> Writing code with `dplyr` imvolves using the "grammar of data" to peform data munging. Each step is done by a single function that represents a verb.
>
> — Jared P. Lander (Lander, 2017)

Data Science for Operational Researchers using R

06 – dplyr

https://github.com/JimDuggan/explore_or

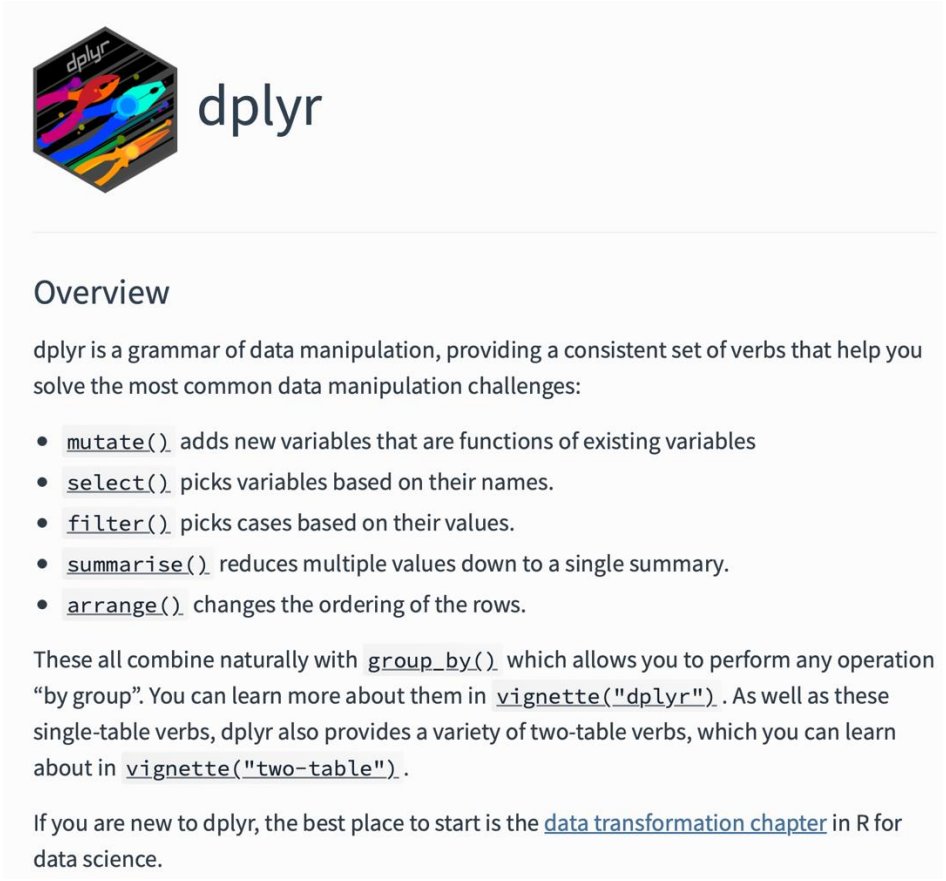
Course Overview

Topic	Description
	Session 1 (half-day)
1	Introduction to R and Posit Cloud
2	Atomic Vectors
3	Functions, Lists and Functionals
4	Data Frames and Tibble
5	**ggplot2**
	Session 2 (half-day)
6	Data transformation with **dplyr**
7	Relational Data with **dplyr**
8	Processing data with **purrr**
9	Exploratory Data Analysis – Examples

Overview

- dplyr provides similar functionality to subset() and transform(), while also supporting aggregation of data
- “tibble in, tibble out”
- Elegant architecture, based on the “5 verbs”
- Makes us of the tidyverse pipe %>%

https://dplyr.tidyverse.org



dplyr

Overview

dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges:

- `mutate()` adds new variables that are functions of existing variables
- `select()` picks variables based on their names.
- `filter()` picks cases based on their values.
- `summarise()` reduces multiple values down to a single summary.
- `arrange()` changes the ordering of the rows.

These all combine naturally with `group_by()` which allows you to perform any operation “by group”. You can learn more about them in `vignette("dplyr")`. As well as these single-table verbs, dplyr also provides a variety of two-table verbs, which you can learn about in `vignette("two-table")`.

If you are new to dplyr, the best place to start is the data transformation chapter in R for data science.

dplyr Basics: 5 key functions

Function	Purpose
filter()	Pick observations by their values
arrange()	Reorder the rows
select()	Pick variables by their names
mutate()	Create new variables with functions of existing variables
summarise()	Collapse many values down to a single summary

- ”A grammar of data manipulation” https://dplyr.tidyverse.org
- All verbs (functions) work similarly
  - The first argument is a data frame/tibble
  - The subsequent arguments decide what to do with the data frame/tibble
  - The result (data frame/tibble) supports chaining of steps – NOTE the “pipe operator” which we will cover later.

The tidyverse pipe - magrittr

```
library(magrittr)
```

```
set.seed(100)
arr   <- rpois(12,50)
arr
#>  [1] 46 38 56 50 52 45 53 47 50 50 48 55
# The function sort takes a parameter to sort in descending order
s_arr <- sort(arr,decreasing = TRUE)
top_6 <- head(s_arr)
top_6
#> [1] 56 55 53 52 50 50
```

```
set.seed(100)
top_6  <- rpois(12,50) %>%
            sort(decreasing = TRUE) %>%
            head()
top_6
#> [1] 56 55 53 52 50 50
```

(1) Filtering rows with filter()

- Used to subset a tibble
- Arguments
  - A data frame/tibble
  - A list of expressions that return a logical value and are defined in terms of variables that are present in the data frame
- All columns returned

```
mpg1 <- filter(mpg,class=="2seater")
mpg1
#> # A tibble: 5 x 11
#>   manufa~1 model displ  year   cyl trans drv     cty   hwy fl    class
#>   <chr>    <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
#> 1 chevrol~ corv~   5.7  1999     8 manu~ r        16    26 p     2sea~
#> 2 chevrol~ corv~   5.7  1999     8 auto~ r        15    23 p     2sea~
#> 3 chevrol~ corv~   6.2  2008     8 manu~ r        16    26 p     2sea~
#> 4 chevrol~ corv~   6.2  2008     8 auto~ r        15    25 p     2sea~
#> 5 chevrol~ corv~   7    2008     8 manu~ r        15    24 p     2sea~
#> # ... with abbreviated variable name 1: manufacturer
```

```
mpg2 <- filter(mpg,class=="2seater",hwy >= 25)
mpg2
#> # A tibble: 3 x 11
#>   manufa~1 model displ  year   cyl trans drv     cty   hwy fl    class
#>   <chr>    <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
#> 1 chevrol~ corv~   5.7  1999     8 manu~ r        16    26 p     2sea~
#> 2 chevrol~ corv~   6.2  2008     8 manu~ r        16    26 p     2sea~
#> 3 chevrol~ corv~   6.2  2008     8 auto~ r        15    25 p     2sea~
#> # ... with abbreviated variable name 1: manufacturer
```

```
mpg3 <- filter(mpg,manufacturer=="lincoln" | manufacturer == "mercury")
mpg3
#> # A tibble: 7 x 11
#>   manufa~1 model displ  year   cyl trans drv     cty   hwy fl    class
#>   <chr>    <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
#> 1 lincoln  navi~   5.4  1999     8 auto~ r        11    17 r     suv
#> 2 lincoln  navi~   5.4  1999     8 auto~ r        11    16 p     suv
#> 3 lincoln  navi~   5.4  2008     8 auto~ r        12    18 r     suv
#> 4 mercury  moun~   4    1999     6 auto~ 4        14    17 r     suv
#> 5 mercury  moun~   4    2008     6 auto~ 4        13    19 r     suv
#> 6 mercury  moun~   4.6  2008     8 auto~ 4        13    19 r     suv
#> 7 mercury  moun~   5    1999     8 auto~ 4        13    17 r     suv
#> # ... with abbreviated variable name 1: manufacturer
```

```r
mpg4 <- filter(mpg,manufacturer %in% c("lincoln","mercury"))
mpg4
#> # A tibble: 7 x 11
#>   manufa~1 model displ  year   cyl trans drv     cty   hwy fl    class
#>   <chr>    <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
#> 1 lincoln  navi~   5.4  1999     8 auto~ r        11    17 r     suv
#> 2 lincoln  navi~   5.4  1999     8 auto~ r        11    16 p     suv
#> 3 lincoln  navi~   5.4  2008     8 auto~ r        12    18 r     suv
#> 4 mercury  moun~   4    1999     6 auto~ 4        14    17 r     suv
#> 5 mercury  moun~   4    2008     6 auto~ 4        13    19 r     suv
#> 6 mercury  moun~   4.6  2008     8 auto~ 4        13    19 r     suv
#> 7 mercury  moun~   5    1999     8 auto~ 4        13    17 r     suv
#> # ... with abbreviated variable name 1: manufacturer
```

The slice() function

```
# Show the first 3 rows
slice(mpg,1:3)
#> # A tibble: 3 x 11
#>   manufa~1 model displ  year   cyl trans drv     cty   hwy fl    class
#>   <chr>    <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
#> 1 audi     a4      1.8  1999     4 auto~ f        18    29 p     comp~
#> 2 audi     a4      1.8  1999     4 manu~ f        21    29 p     comp~
#> 3 audi     a4      2    2008     4 manu~ f        20    31 p     comp~
#> # ... with abbreviated variable name 1: manufacturer
```

(2) Sorting rows with arrange()

```
arrange(mpg,cty) %>% slice(1:3)
#> # A tibble: 3 x 11
#>   manufa~1 model displ  year   cyl trans drv     cty   hwy fl    class
#>   <chr>    <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
#> 1 dodge    dako~   4.7  2008     8 auto~ 4         9    12 e     pick~
#> 2 dodge    dura~   4.7  2008     8 auto~ 4         9    12 e     suv
#> 3 dodge    ram ~   4.7  2008     8 auto~ 4         9    12 e     pick~
#> # ... with abbreviated variable name 1: manufacturer
```

```
arrange(mpg,desc(cty)) %>% slice(1:3)
#> # A tibble: 3 x 11
#>   manufa~1 model displ  year   cyl trans drv     cty   hwy fl    class
#>   <chr>    <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
#> 1 volkswa~ new ~   1.9  1999     4 manu~ f        35    44 d     subc~
#> 2 volkswa~ jetta   1.9  1999     4 manu~ f        33    44 d     comp~
#> 3 volkswa~ new ~   1.9  1999     4 auto~ f        29    41 d     subc~
#> # ... with abbreviated variable name 1: manufacturer
```

Using more than one sorting variable

```
arrange(mpg,class,desc(cty)) %>% slice(1:7)
#> # A tibble: 7 x 11
#>   manufa~1 model displ  year   cyl trans drv     cty   hwy fl    class
#>   <chr>    <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
#> 1 chevrol~ corv~   5.7  1999     8 manu~ r        16    26 p     2sea~
#> 2 chevrol~ corv~   6.2  2008     8 manu~ r        16    26 p     2sea~
#> 3 chevrol~ corv~   5.7  1999     8 auto~ r        15    23 p     2sea~
#> 4 chevrol~ corv~   6.2  2008     8 auto~ r        15    25 p     2sea~
#> 5 chevrol~ corv~   7    2008     8 manu~ r        15    24 p     2sea~
#> 6 volkswa~ jetta   1.9  1999     4 manu~ f        33    44 d     comp~
#> 7 toyota   coro~   1.8  2008     4 manu~ f        28    37 r     comp~
#> # ... with abbreviated variable name 1: manufacturer
```

(3) Choosing columns with select()

- Enables you to subset columns from the input tibble
- select() will always return all the observations
- Simplest use is to write the column names that are required.

```
select(mpg,manufacturer,model,year,displ,cty)
#> # A tibble: 234 x 5
#>    manufacturer model       year displ   cty
#>    <chr>        <chr>      <int> <dbl> <int>
#>  1 audi         a4          1999   1.8    18
#>  2 audi         a4          1999   1.8    21
#>  3 audi         a4          2008   2      20
#>  4 audi         a4          2008   2      21
#>  5 audi         a4          1999   2.8    16
#>  6 audi         a4          1999   2.8    18
#>  7 audi         a4          2008   3.1    18
#>  8 audi         a4 quattro  1999   1.8    18
#>  9 audi         a4 quattro  1999   1.8    16
#> 10 audi         a4 quattro  2008   2      20
```

Use : for selecting a range of consecutive variables

```
mpg %>% select(manufacturer:year,cty)
#> # A tibble: 234 x 5
#>    manufacturer model      displ  year   cty
#>    <chr>        <chr>      <dbl> <int> <int>
#>  1 audi         a4           1.8  1999    18
#>  2 audi         a4           1.8  1999    21
#>  3 audi         a4           2    2008    20
#>  4 audi         a4           2    2008    21
#>  5 audi         a4           2.8  1999    16
#>  6 audi         a4           2.8  1999    18
#>  7 audi         a4           3.1  2008    18
#>  8 audi         a4 quattro   1.8  1999    18
#>  9 audi         a4 quattro   1.8  1999    16
#> 10 audi         a4 quattro   2    2008    20
#> # ... with 224 more rows
#> # i Use `print(n = ...)` to see more rows
```

! For the complement of a collection of variables

```
mpg %>% select(!(manufacturer:year))
#> # A tibble: 234 x 7
#>      cyl trans      drv     cty   hwy fl    class
#>    <int> <chr>      <chr> <int> <int> <chr> <chr>
#>  1     4 auto(l5)   f        18    29 p     compact
#>  2     4 manual(m5) f        21    29 p     compact
#>  3     4 manual(m6) f        20    31 p     compact
#>  4     4 auto(av)   f        21    30 p     compact
#>  5     6 auto(l5)   f        16    26 p     compact
#>  6     6 manual(m5) f        18    26 p     compact
#>  7     6 auto(av)   f        18    27 p     compact
```

```
#>  8     4 manual(m5) 4        18    26 p     compact
#>  9     4 auto(l5)   4        16    25 p     compact
#> 10     4 manual(m6) 4        20    28 p     compact
#> # ... with 224 more rows
#> # i Use `print(n = ...)` to see more rows
```

c() for combining selections

```
mpg %>% select(c(manufacturer:year,cty))
#> # A tibble: 234 x 5
#>    manufacturer model      displ  year   cty
#>    <chr>        <chr>      <dbl> <int> <int>
#>  1 audi         a4           1.8  1999    18
#>  2 audi         a4           1.8  1999    21
#>  3 audi         a4           2    2008    20
#>  4 audi         a4           2    2008    21
#>  5 audi         a4           2.8  1999    16
#>  6 audi         a4           2.8  1999    18
#>  7 audi         a4           3.1  2008    18
#>  8 audi         a4 quattro   1.8  1999    18
#>  9 audi         a4 quattro   1.8  1999    16
#> 10 audi         a4 quattro   2    2008    20
#> # ... with 224 more rows
#> # i Use `print(n = ...)` to see more rows
```

Selection helper: starts_with()

```
mpg %>% select(starts_with("m"))
#> # A tibble: 234 x 2
#>    manufacturer model
#>    <chr>        <chr>
#>  1 audi         a4
#>  2 audi         a4
#>  3 audi         a4
#>  4 audi         a4
#>  5 audi         a4
#>  6 audi         a4
#>  7 audi         a4
#>  8 audi         a4 quattro
#>  9 audi         a4 quattro
#> 10 audi         a4 quattro
#> # ... with 224 more rows
#> # i Use `print(n = ...)` to see more rows
```

Selection helper: ends_with()

```
mpg %>% select(ends_with("l"))
#> # A tibble: 234 x 4
#>    model      displ   cyl fl
#>    <chr>      <dbl> <int> <chr>
#>  1 a4           1.8     4 p
#>  2 a4           1.8     4 p
#>  3 a4           2       4 p
#>  4 a4           2       4 p
#>  5 a4           2.8     6 p
#>  6 a4           2.8     6 p
#>  7 a4           3.1     6 p
#>  8 a4 quattro   1.8     4 p
#>  9 a4 quattro   1.8     4 p
#> 10 a4 quattro   2       4 p
#> # ... with 224 more rows
#> # i Use `print(n = ...)` to see more rows
```

Selection helper: contains()

```
mpg %>% select(contains("an"))
#> # A tibble: 234 x 2
#>    manufacturer trans
#>    <chr>        <chr>
#>  1 audi         auto(l5)
#>  2 audi         manual(m5)
#>  3 audi         manual(m6)
#>  4 audi         auto(av)
#>  5 audi         auto(l5)
#>  6 audi         manual(m5)
#>  7 audi         auto(av)
#>  8 audi         manual(m5)
#>  9 audi         auto(l5)
#> 10 audi         manual(m6)
#> # ... with 224 more rows
#> # i Use `print(n = ...)` to see more rows
```

num_range() – prefix followed by numbers

```
billboard %>% select(artist,track,num_range("wk", 1:3)) %>% slice(1:5)
#> # A tibble: 5 x 5
#>   artist       track                   wk1   wk2   wk3
#>   <chr>        <chr>                 <dbl> <dbl> <dbl>
#> 1 2 Pac        Baby Don't Cry (Keep...    87    82    72
#> 2 2Ge+her      The Hardest Part Of ...    91    87    92
#> 3 3 Doors Down Kryptonite                 81    70    68
#> 4 3 Doors Down Loser                      76    76    72
#> 5 504 Boyz     Wobble Wobble              57    34    25
```

matches() – matches a regular expression pattern

```
 billboard %>% select(matches("wk.3")) %>% slice(1:3)
#> # A tibble: 3 x 7
#>    wk13  wk23  wk33  wk43  wk53  wk63 wk73
#>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <lgl>
#> 1    NA    NA    NA    NA    NA    NA NA
#> 2    NA    NA    NA    NA    NA    NA NA
#> 3    47     6     3    14    49    NA NA
```

The pattern here is to find a column that starts with "wk" and then contains any character (".") and concludes with the character "3".

everything() – matches all remaining columns

```
mpg %>% select(manufacturer:year,cty,hwy,everything())
#> # A tibble: 234 x 11
#>    manuf~1 model displ  year   cty   hwy   cyl trans drv   fl    class
#>    <chr>   <chr> <dbl> <int> <int> <int> <int> <chr> <chr> <chr> <chr>
#>  1 audi    a4      1.8  1999    18    29     4 auto~ f     p     comp~
#>  2 audi    a4      1.8  1999    21    29     4 manu~ f     p     comp~
#>  3 audi    a4      2    2008    20    31     4 manu~ f     p     comp~
#>  4 audi    a4      2    2008    21    30     4 auto~ f     p     comp~
#>  5 audi    a4      2.8  1999    16    26     6 auto~ f     p     comp~
#>  6 audi    a4      2.8  1999    18    26     6 manu~ f     p     comp~
#>  7 audi    a4      3.1  2008    18    27     6 auto~ f     p     comp~
#>  8 audi    a4 q~   1.8  1999    18    26     4 manu~ 4     p     comp~
#>  9 audi    a4 q~   1.8  1999    16    25     4 auto~ 4     p     comp~
#> 10 audi    a4 q~   2    2008    20    28     4 manu~ 4     p     comp~
#> # ... with 224 more rows, and abbreviated variable name
#> #   1: manufacturer
#> # i Use `print(n = ...)` to see more rows
```

(4) Adding columns with mutate()

- The function mutate() adds new variables to a tibble, while keeping the original ones
- It also allows for variables to be overwritten, and variables can be deleted by setting their values to NULL
- Arguments:
  - tibble that is to be transformed
  - name-value pairs. Which can be a vector of length 1 (then recycled), a vector of the same length as the tibble, or NULL.

```
set.seed(100)
mpg_m <- mpg %>%
         filter(class %in% c("compact","midsize")) %>%
         select(manufacturer:year,cty,class) %>%
         sample_n(6)
mpg_m
#> # A tibble: 6 x 6
#>   manufacturer model  displ  year   cty class
#>   <chr>        <chr>  <dbl> <int> <int> <chr>
#> 1 volkswagen   jetta    2    1999    21 compact
#> 2 volkswagen   jetta    2.5  2008    21 compact
#> 3 chevrolet    malibu   3.6  2008    17 midsize
#> 4 volkswagen   gti      2    2008    21 compact
#> 5 audi         a4       2    2008    21 compact
#> 6 toyota       camry    3.5  2008    19 midsize
```

Adding vector of size 1

```
mpg_m %>% mutate(Test="A test")
#> # A tibble: 6 x 7
#>   manufacturer model  displ  year   cty class   Test
#>   <chr>        <chr>  <dbl> <int> <int> <chr>   <chr>
#> 1 volkswagen   jetta    2    1999    21 compact A test
#> 2 volkswagen   jetta    2.5  2008    21 compact A test
#> 3 chevrolet    malibu   3.6  2008    17 midsize A test
#> 4 volkswagen   gti      2    2008    21 compact A test
#> 5 audi         a4       2    2008    21 compact A test
#> 6 toyota       camry    3.5  2008    19 midsize A test
```

Adding a vector of the same length

```
mpg_m %>% dplyr::mutate(cty_kmh=cty*1.6,
                        cty_2=cty_kmh/1.6)
#> # A tibble: 6 x 8
#>   manufacturer model  displ  year   cty class   cty_kmh cty_2
#>   <chr>        <chr>  <dbl> <int> <int> <chr>     <dbl> <dbl>
#> 1 volkswagen   jetta    2    1999    21 compact    33.6    21
#> 2 volkswagen   jetta    2.5  2008    21 compact    33.6    21
#> 3 chevrolet    malibu   3.6  2008    17 midsize    27.2    17
#> 4 volkswagen   gti      2    2008    21 compact    33.6    21
#> 5 audi         a4       2    2008    21 compact    33.6    21
#> 6 toyota       camry    3.5  2008    19 midsize    30.4    19
```

Remove a variable from a tibble (=NULL)

```
mpg_m %>% dplyr::mutate(class=NULL)
#> # A tibble: 6 x 5
#>   manufacturer model  displ  year   cty
#>   <chr>        <chr>  <dbl> <int> <int>
#> 1 volkswagen   jetta    2    1999    21
#> 2 volkswagen   jetta    2.5  2008    21
#> 3 chevrolet    malibu   3.6  2008    17
#> 4 volkswagen   gti      2    2008    21
#> 5 audi         a4       2    2008    21
#> 6 toyota       camry    3.5  2008    19
```

Using group_by() with mutate

- Another function that can be used with mutate() is group_by() which takes a tibble and converts it to a grouped tibble, based on the input variable(s).
- Computations can then be performed on the grouped data.

```
# Group the tibble by class
mpg_mg <- mpg_m %>% dplyr::group_by(class)
mpg_mg
#> # A tibble: 6 x 6
#> # Groups:   class [2]
#>   manufacturer model  displ  year   cty class
#>   <chr>        <chr>  <dbl> <int> <int> <chr>
#> 1 volkswagen   jetta    2    1999    21 compact
#> 2 volkswagen   jetta    2.5  2008    21 compact
#> 3 chevrolet    malibu   3.6  2008    17 midsize
#> 4 volkswagen   gti      2    2008    21 compact
#> 5 audi         a4       2    2008    21 compact
#> 6 toyota       camry    3.5  2008    19 midsize
```

Using mutate() with group_by()

```
mpg_mg %>% dplyr::mutate(MaxCtyByClass=max(cty))
#> # A tibble: 6 x 7
#> # Groups:   class [2]
#>   manufacturer model  displ  year   cty class   MaxCtyByClass
#>   <chr>        <chr>  <dbl> <int> <int> <chr>           <int>
#> 1 volkswagen   jetta    2    1999    21 compact            21
#> 2 volkswagen   jetta    2.5  2008    21 compact            21
#> 3 chevrolet    malibu   3.6  2008    17 midsize            19
#> 4 volkswagen   gti      2    2008    21 compact            21
#> 5 audi         a4       2    2008    21 compact            21
#> 6 toyota       camry    3.5  2008    19 midsize            19
```

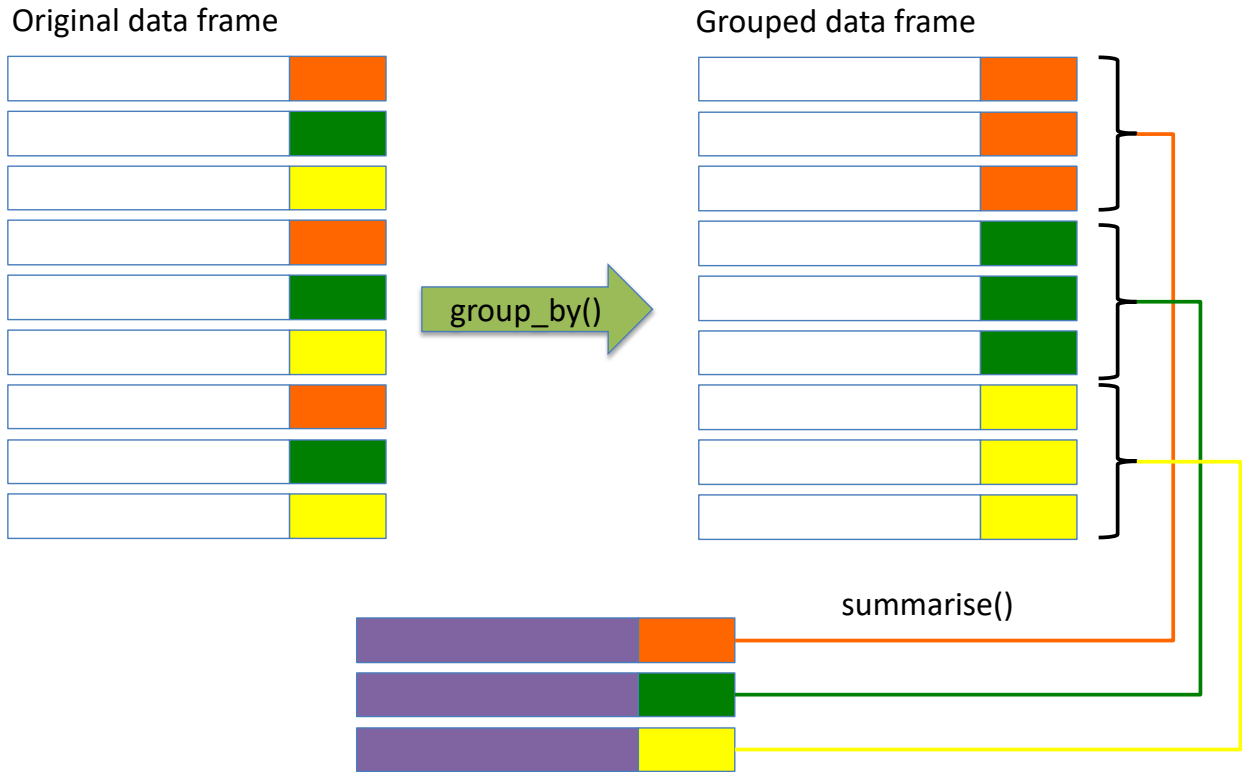
Remove groupings with ungroup()

Groupings can then be removed from a tibble with a call to the function `upgroup()`.

```
mpg_mg %>% dplyr::ungroup()
#> # A tibble: 6 x 6
#>   manufacturer model  displ  year   cty class
#>   <chr>        <chr>  <dbl> <int> <int> <chr>
#> 1 volkswagen   jetta    2    1999    21 compact
#> 2 volkswagen   jetta    2.5  2008    21 compact
#> 3 chevrolet    malibu   3.6  2008    17 midsize
#> 4 volkswagen   gti      2    2008    21 compact
#> 5 audi         a4       2    2008    21 compact
#> 6 toyota       camry    3.5  2008    19 midsize
```

(5) Summarising data with summarise()

- The function summarise() creates a new tibble.
- It will have one (or more) rows for each combination of grouping variable.
- If there are no groupings, then a single row summary of all observations will be displayed.
- Typically, summary functions are applied to the groups
- The original data frame is "collapsed" into fewer values, keyed by the group name(s).



summarise() arguments

- The arguments passed to summarize() include:
  - The tibble/data frame, which will usually have group attributes defined.
  - Name-value pairs of summary functions, where the name will be that of the column in the result. Summary functions include (Wickham and Grolemund, 2016):

Type	Examples
Measures of location	`mean()`, `median()`
Measures of spread	`sd()`, `IQR()`
Measures of rank	`min()`, `max()`, quantile()
Measures of position	first(), nth(), last()
Counts	n(), n_distinct()
Proportions	e.g., sum(x>0)/n()

summarise() with no grouping

```
mpg %>%
  dplyr::summarize(CtyAvr=mean(cty),
                   CtySD=sd(cty),
                   HwyAvr=mean(hwy),
                   HwySD=sd(hwy))
#> # A tibble: 1 x 4
#>   CtyAvr CtySD HwyAvr HwySD
#>    <dbl> <dbl>  <dbl> <dbl>
#> 1   16.9  4.26   23.4  5.95
```

summarise() with a group.

```
mpg %>%
  dplyr::group_by(class) %>%
  dplyr::summarize(CtyAvr=mean(cty),
                   CtySD=sd(cty),
                   HwyAvr=mean(hwy),
                   HwySD=sd(hwy),
                   N=dplyr::n()) %>%
  ungroup()
#> # A tibble: 7 x 6
#>   class      CtyAvr CtySD HwyAvr HwySD     N
#>   <chr>       <dbl> <dbl>  <dbl> <dbl> <int>
#> 1 2seater      15.4 0.548   24.8  1.30     5
#> 2 compact      20.1 3.39    28.3  3.78    47
#> 3 midsize      18.8 1.95    27.3  2.14    41
#> 4 minivan      15.8 1.83    22.4  2.06    11
#> 5 pickup       13   2.05    16.9  2.27    33
#> 6 subcompact   20.4 4.60    28.1  5.38    35
#> 7 suv          13.5 2.42    18.1  2.98    62
```

Extracting additional information from groups

```
mpg %>%
  group_by(class) %>%
  summarize(MaxDispl=max(displ),
            CarMax=dplyr::nth(model,which.max(displ)),
            ManuMax=dplyr::nth(manufacturer,which.max(displ)))
#> # A tibble: 7 x 4
#>   class      MaxDispl CarMax             ManuMax
#>   <chr>         <dbl> <chr>              <chr>
#> 1 2seater         7   corvette           chevrolet
#> 2 compact         3.3 camry solara       toyota
#> 3 midsize         5.3 grand prix         pontiac
#> 4 minivan         4   caravan 2wd        dodge
#> 5 pickup          5.9 ram 1500 pickup 4wd dodge
#> 6 subcompact      5.4 mustang            ford
#> 7 suv             6.5 k1500 tahoe 4wd    chevrolet
```

dplyr mini-case

In this mini-case, based on data stored in `aimsir17`, we use `dplyr` functions to summarize rainfall values, and `ggplot2` to visualize the results, with a focus on answering the following two questions.

1. Calculate the total annual rainfall for each weather station, and visualize using a bar chart.
2. Calculate the total monthly rainfall for two weather stations, "NEW-PORT" and "DUBLIN AIRPORT", and visualize both using a time series graph.

Check the observations tibble

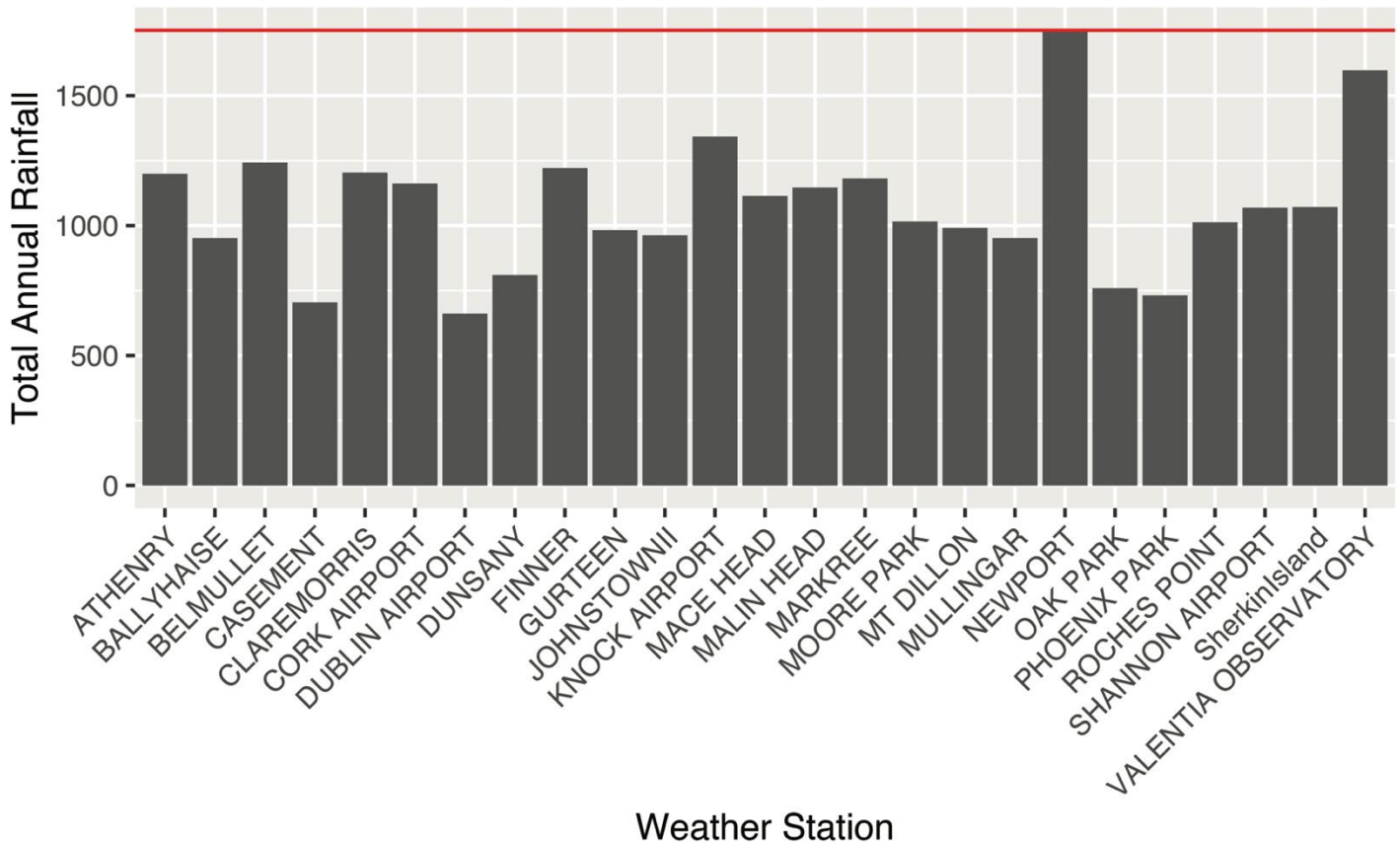
```
library(dplyr)
library(ggplot2)
library(aimsir17)
```

```
observations
#> # A tibble: 219,000 x 12
#>    station  year month   day  hour date                rain
#>      <chr> <dbl> <dbl> <int> <int> <dttm>              <dbl>
#>  1 ATHENRY  2017     1     1     0 2017-01-01 00:00:00   0
#>  2 ATHENRY  2017     1     1     1 2017-01-01 01:00:00   0
#>  3 ATHENRY  2017     1     1     2 2017-01-01 02:00:00   0
#>  4 ATHENRY  2017     1     1     3 2017-01-01 03:00:00   0.1
#>  5 ATHENRY  2017     1     1     4 2017-01-01 04:00:00   0.1
#>  6 ATHENRY  2017     1     1     5 2017-01-01 05:00:00   0
#>  7 ATHENRY  2017     1     1     6 2017-01-01 06:00:00   0
#>  8 ATHENRY  2017     1     1     7 2017-01-01 07:00:00   0
#>  9 ATHENRY  2017     1     1     8 2017-01-01 08:00:00   0
#> 10 ATHENRY  2017     1     1     9 2017-01-01 09:00:00   0
#> # ... with 218,990 more rows, and 5 more variables: temp <dbl>,
#> #   rhum <dbl>, msl <dbl>, wdsp <dbl>, wddir <dbl>
```

Summarising the data… 25 rows, why?

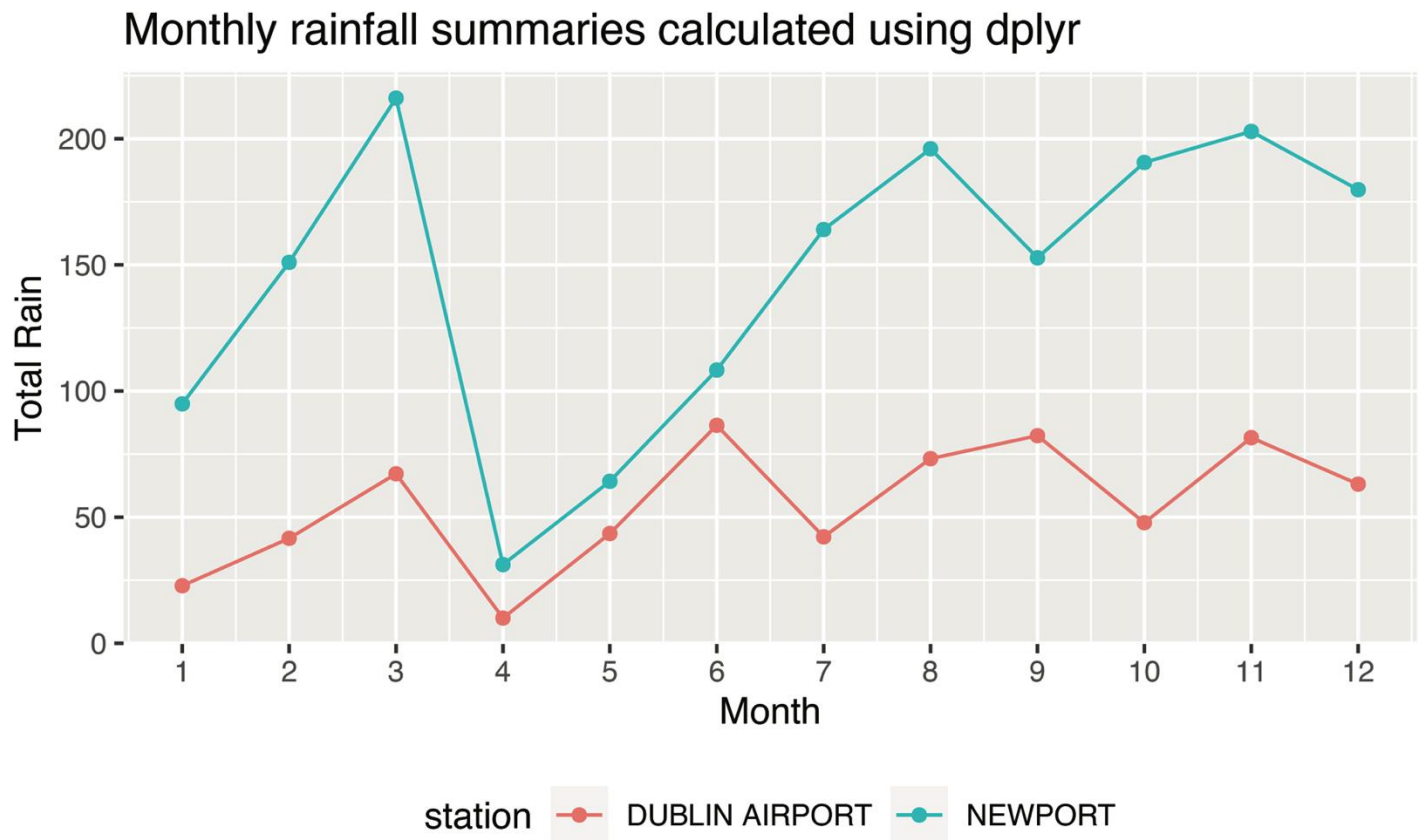
```
annual_rain <- observations %>%
                  dplyr::group_by(station) %>%
                  dplyr::summarize(TotalRain=sum(rain,na.rm=T))
annual_rain
#> # A tibble: 25 x 2
#>    station        TotalRain
#>    <chr>              <dbl>
#>  1 ATHENRY            1199.
#>  2 BALLYHAISE          952.
#>  3 BELMULLET          1243.
#>  4 CASEMENT            705.
#>  5 CLAREMORRIS        1204.
#>  6 CORK AIRPORT       1162.
#>  7 DUBLIN AIRPORT      662.
#>  8 DUNSANY             810.
#>  9 FINNER             1222.
#> 10 GURTEEN             983.
#> # ... with 15 more rows
```

```
ggplot(annual_rain,aes(x=station,y=TotalRain))+
  geom_bar(stat = "identity")+
  theme(axis.text.x=element_text(angle=45,hjust=1))+
  geom_hline(yintercept = max(annual_rain$TotalRain),color="red")+
  xlab("Weather Station") + ylab("Total Annual Rainfall")
```




```
dplyr::filter(annual_rain,
       station %in% c("DUBLIN AIRPORT",
                      "NEWPORT"))
#> # A tibble: 2 x 2
#>   station         TotalRain
#>   <chr>               <dbl>
#> 1 DUBLIN AIRPORT       662.
#> 2 NEWPORT             1752.
```

```r
monthly_rain <- observations %>%
                dplyr::filter(station %in% c("DUBLIN AIRPORT",
                                             "NEWPORT")) %>%
                dplyr::group_by(station, month) %>%
                dplyr::summarize(TotalRain=sum(rain,na.rm = T))
monthly_rain
#> # A tibble: 24 x 3
#> # Groups:   station [2]
#>    station         month TotalRain
#>    <chr>           <dbl>     <dbl>
#>  1 DUBLIN AIRPORT      1      22.8
#>  2 DUBLIN AIRPORT      2      41.6
#>  3 DUBLIN AIRPORT      3      67.2
#>  4 DUBLIN AIRPORT      4      10
#>  5 DUBLIN AIRPORT      5      43.5
#>  6 DUBLIN AIRPORT      6      86.4
#>  7 DUBLIN AIRPORT      7      42.2
#>  8 DUBLIN AIRPORT      8      73.2
#>  9 DUBLIN AIRPORT      9      82.3
#> 10 DUBLIN AIRPORT     10      47.8
#> # ... with 14 more rows
```




```
ggplot(monthly_rain,aes(x=month,y=TotalRain,color=station))+
  geom_point()+geom_line()+
  theme(legend.position = "bottom")+
  scale_x_continuous(limits=c(1,12), breaks=seq(1,12))+
  labs(x="Month",
       y="Total Rain",
       title = "Monthly rainfall summaries calculated using dplyr")
```

(6) Additional dplyr functions case_when()

- The function case_when() enables you to implement similar code to the ifelse() function.
- The arguments are a sequence of two-sided formulas, where the left-hand side (LHS) is used to formulate a condition, and the right-hand side (RHS) provides replacement values.
- If the condition is TRUE, the RHS value is allocated, and no other conditions are evaluated
- The symbol ~ separates the two sides.
- TRUE at the end provides a default value

```
x <- sample(1:99,5,replace = T)
x[5] <- 200

x
#> [1]  70  98   7   7 200

y <- dplyr::case_when(x < 18   ~ "Child",
                      x < 65   ~ "Adult",
                      x <= 99  ~ "Elderly",
                      TRUE     ~ "Unknown")
y
#> [1] "Elderly" "Elderly" "Child"   "Child"   "Unknown"
```

```
mpg %>%
  dplyr::select(manufacturer:model,cty) %>%
  dplyr::mutate(cty_status=case_when(
    cty >= mean(cty) ~ "Above average",
    cty < mean(cty)  ~ "Below average",
    TRUE             ~  "Undefined"
  )) %>%
  dplyr::slice(1:5)
#> # A tibble: 5 x 4
#>   manufacturer model   cty cty_status
#>   <chr>        <chr> <int> <chr>
#> 1 audi         a4       18 Above average
#> 2 audi         a4       21 Above average
#> 3 audi         a4       20 Above average
#> 4 audi         a4       21 Above average
#> 5 audi         a4       16 Below average
```

The pull() function

- A feature of the dplyr is that the functions we've explored always return a tibble.
- However, there may be cases when *we need to return just one column*, and while the $ operator can be used, the function pull() is a better choice, especially when using pipes.
- In this example, an atomic vector is returned

```
mpg %>%
  dplyr::pull(class) %>%
  unique()
#> [1] "compact"    "midsize"    "suv"        "2seater"
#> [5] "minivan"    "pickup"     "subcompact"
```

Exercise 1

1. Based on the `mpg` dataset from `ggplot2`, generate the following tibble which filters all the cars with a `cty` value greater than the median. Ensure that your tibble contains the same columns, and with `set.seed(100)` sample five records using `sample_n()`, and store the result in the tibble `ans`.

```
ans
#> # A tibble: 5 x 7
#>   manufacturer model        displ  year   cty   hwy class
#>   <chr>        <chr>        <dbl> <int> <int> <int> <chr>
#> 1 nissan       maxima         3    1999    18    26 midsize
#> 2 volkswagen   gti            2    1999    19    26 compact
#> 3 subaru       impreza awd    2.5  2008    20    27 compact
#> 4 chevrolet    malibu         3.1  1999    18    26 midsize
#> 5 subaru       forester awd   2.5  2008    19    25 suv
```

Exercise 2

2. Based on the `aimsir17` tibble `observations`, generate the tibble `jan` which contains observations for two weather stations ("DUBLIN AIRPORT" and "MACE HEAD") during the month of January. Add a new column named `WeatherStatus` that contains three possible values: "Warning - Freezing" if the temperature is less than or equal to 0, "Warning - Very Cold" if the temperature is greater than zero and less than or equal to 4, and "No Warning" if the temparture is above 4. Make use of the `case_when()` function, and replicate the plot.

```
jan
#> # A tibble: 1,488 x 7
#>    station    month   day  hour date                 temp Weath~1
#>    <chr>      <dbl> <int> <int> <dttm>              <dbl> <chr>
#>  1 DUBLIN AI~     1     1     0 2017-01-01 00:00:00   5.3 No War~
#>  2 MACE HEAD      1     1     0 2017-01-01 00:00:00   5.6 No War~
#>  3 DUBLIN AI~     1     1     1 2017-01-01 01:00:00   4.9 No War~
#>  4 MACE HEAD      1     1     1 2017-01-01 01:00:00   5.4 No War~
#>  5 DUBLIN AI~     1     1     2 2017-01-01 02:00:00   5   No War~
#>  6 MACE HEAD      1     1     2 2017-01-01 02:00:00   4.7 No War~
#>  7 DUBLIN AI~     1     1     3 2017-01-01 03:00:00   4.2 No War~
#>  8 MACE HEAD      1     1     3 2017-01-01 03:00:00   4.7 No War~
#>  9 DUBLIN AI~     1     1     4 2017-01-01 04:00:00   3.6 Warnin~
#> 10 MACE HEAD      1     1     4 2017-01-01 04:00:00   4.5 No War~
#> # ... with 1,478 more rows, and abbreviated variable name
#> #   1: WeatherStatus
```