Data Science for Operational Researchers Using R Online

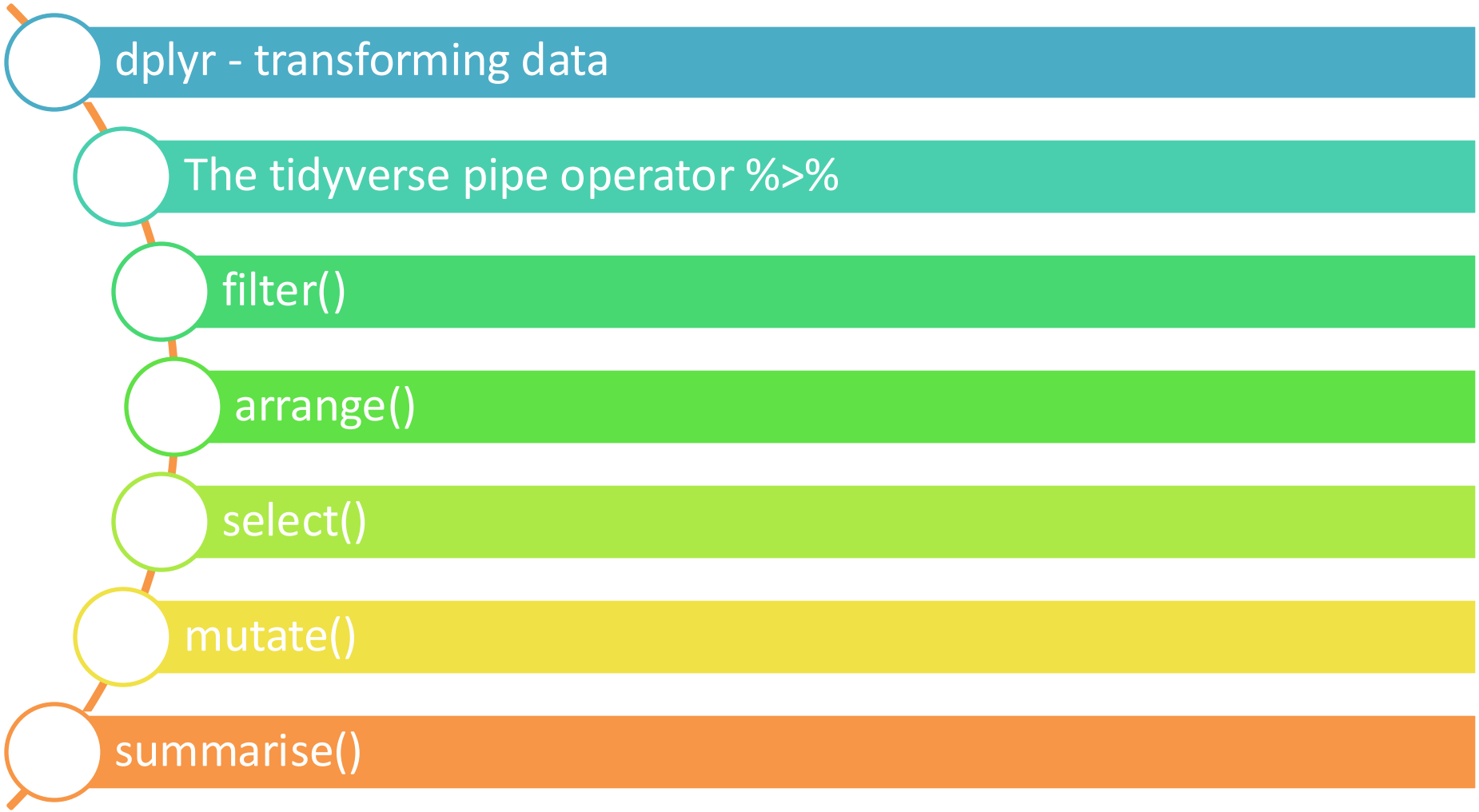
3. Data Transformation with dplyr

Prof. Jim Duggan,
School of Computer Science
University of Galway.

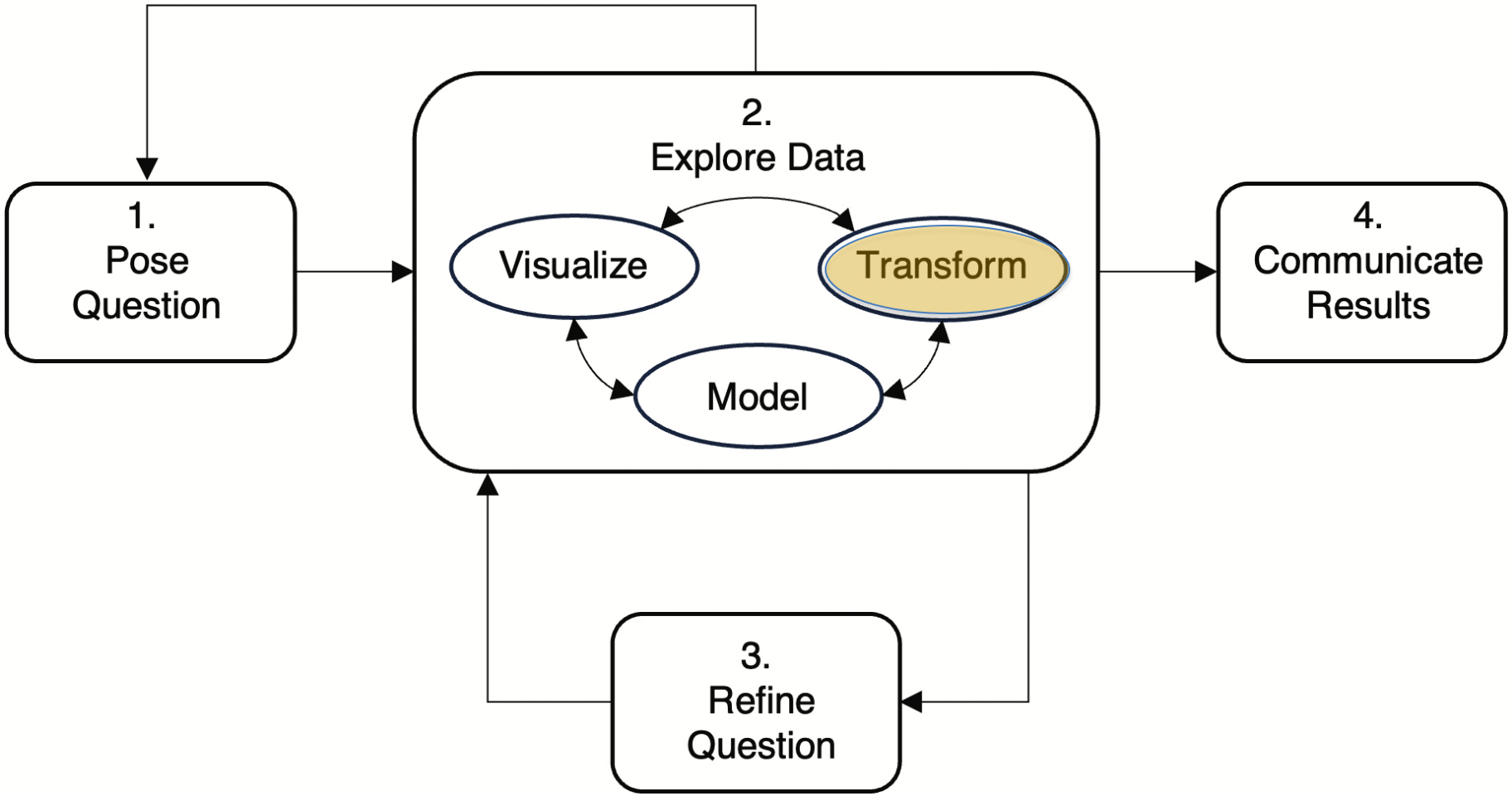
https://github.com/JimDuggan/explore_or

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

Overview


dplyr - transforming data
The tidyverse pipe operator %>%
filter()
arrange()
select()
mutate()
summarise()

1. dplyr – transforming data



Overview

- A tibble may not contain the data that is required, and further transformation of the tibble may be needed.
- The package dplyr provides an alternative set of functions to support data transformations, and also additional functions that can be used to summarize data.
- The underlying structure of dplyr is often termed "tibble-in tibble out" (Wickham and Grolemund, 2016), where each function accepts a tibble and a number of arguments, and then returns a tibble.

2. The tidyverse pipe %>%

- The tidyverse pipe operator (%>%) allows you to chain a number of operations together, without having to assign intermediate variables.
- The general format of the pipe operator is **LHS %>% RHS**, where LHS is the **first argument** of the function defined on the RHS.
- Very useful for "tibble-in, tibble-out" pipeline

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

Another example…

```
> mpg %>% dplyr::sample_n(10) %>% head()
# A tibble: 6 × 11
  manufacturer model              displ  year   cyl trans      drv     cty   hwy fl    class
  <chr>        <chr>              <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
1 nissan       pathfinder 4wd       3.3  1999     6 auto(l4)   4        14    17 r     suv
2 volkswagen   jetta                2    1999     4 manual(m5) f        21    29 r     compact
3 toyota       corolla              1.8  2008     4 auto(l4)   f        26    35 r     compact
4 audi         a4                   2    2008     4 auto(av)   f        21    30 p     compact
5 dodge        dakota pickup 4wd    4.7  2008     8 auto(l5)   4         9    12 e     pickup
6 dodge        ram 1500 pickup 4wd  4.7  2008     8 manual(m6) 4         9    12 e     pickup
```

3. filter()

- The function filter() is used to subset a tibble, in which all rows satisfying a condition are retained.
- The arguments include
  - A data frame, or data frame extensions such as a tibble.
  - A list of expressions that return a logical value and are defined in terms of variables that are present in the input data frame.

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

filter() with two logical expressions

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

filter() with %in% - useful!

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

The slice() function

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

4. arrange() – sorting data

- The arrange() function orders the rows of a tibble by the values of selected columns
- These can be in ascending order (the default), or descending order, by using the function desc().

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

arrange() - Descending order

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

5. select() – Choosing columns

- The function select() allows you to subset columns from the input tibble using the column names
- It will always return all of the observations for the selected columns

```
dplyr::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
#> # ... with 224 more rows
```

: for selecting a range of consecutive variables

```
mpg %>% dplyr::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
```

starts_with()

```
mpg %>% dplyr::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
```

ends_with()

```
mpg %>% dplyr::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
```

contains()

```
mpg %>% dplyr::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)
```

everything() – attaches all remaining columns

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

6. mutate() – Adding columns

- The function mutate() adds new variables to a tibble, while keeping the original ones. There are two main arguments used in mutate()
- The tibble that is to be transformed.
- A set of name-value pairs, where the name provides the column name, and the values can be
  - a vector of length 1 that is subsequently recycled to the tibble length
  - a vector of the same length as the tibble,
  - and (3) the value NULL which will remove the column.

```
set.seed(100)
mpg_m <- mpg %>%
         dplyr::filter(class %in% c("compact","midsize")) %>%
         dplyr::select(manufacturer:year,cty,class) %>%
         dplyr::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 a vector of size 1 - recyles

```
mpg_m %>% dplyr::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
```

Removing a column

```
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
```

7. summarize() – Summarising data

- The function summarize() 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. It takes:
  - 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

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()

summarize() – no groupings

```
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
```

group_by()

- A function that can be used with summarise() is group_by()
- This 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.
- Here, we show how the earlier mpg_m variable is grouped by class.

```
# 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
```

Groupings can be removed with ungroup()

```
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
```

summarize() – 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
```

Using nth() to extract a specific observation

```
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
```

8. Additional useful functions

- A number of additional dplyr functions are worth mentioning, as they can be used alongside the functions we have presented so far.
- These two functions are
  - case_when() and
  - pull()

case_when()

- Similar 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
- The symbol ~ separates the two sides

```
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"
```

pull()

- 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.

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

9. Mini-case: Total rainfall in 2017

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

The observations

```
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>
```

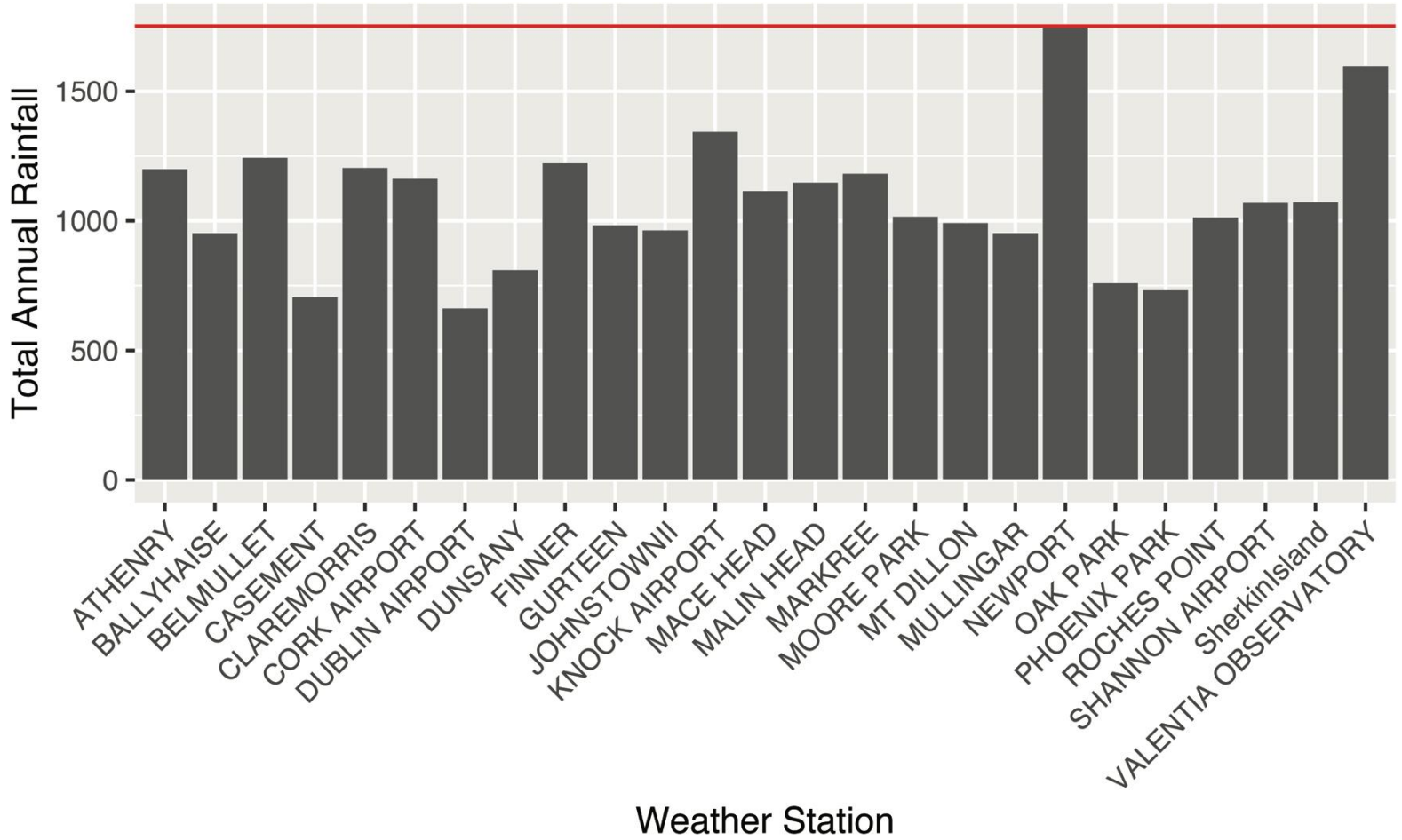
Getting annual totals

```
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
```

```
test <- observations %>%
        dplyr::filter(station=="ATHENRY") %>%
        dplyr::pull(rain) %>%
        sum()
test
#> [1] 1199
annual_rain$TotalRain[1]
#> [1] 1199
```

```
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.
```

```
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")
```

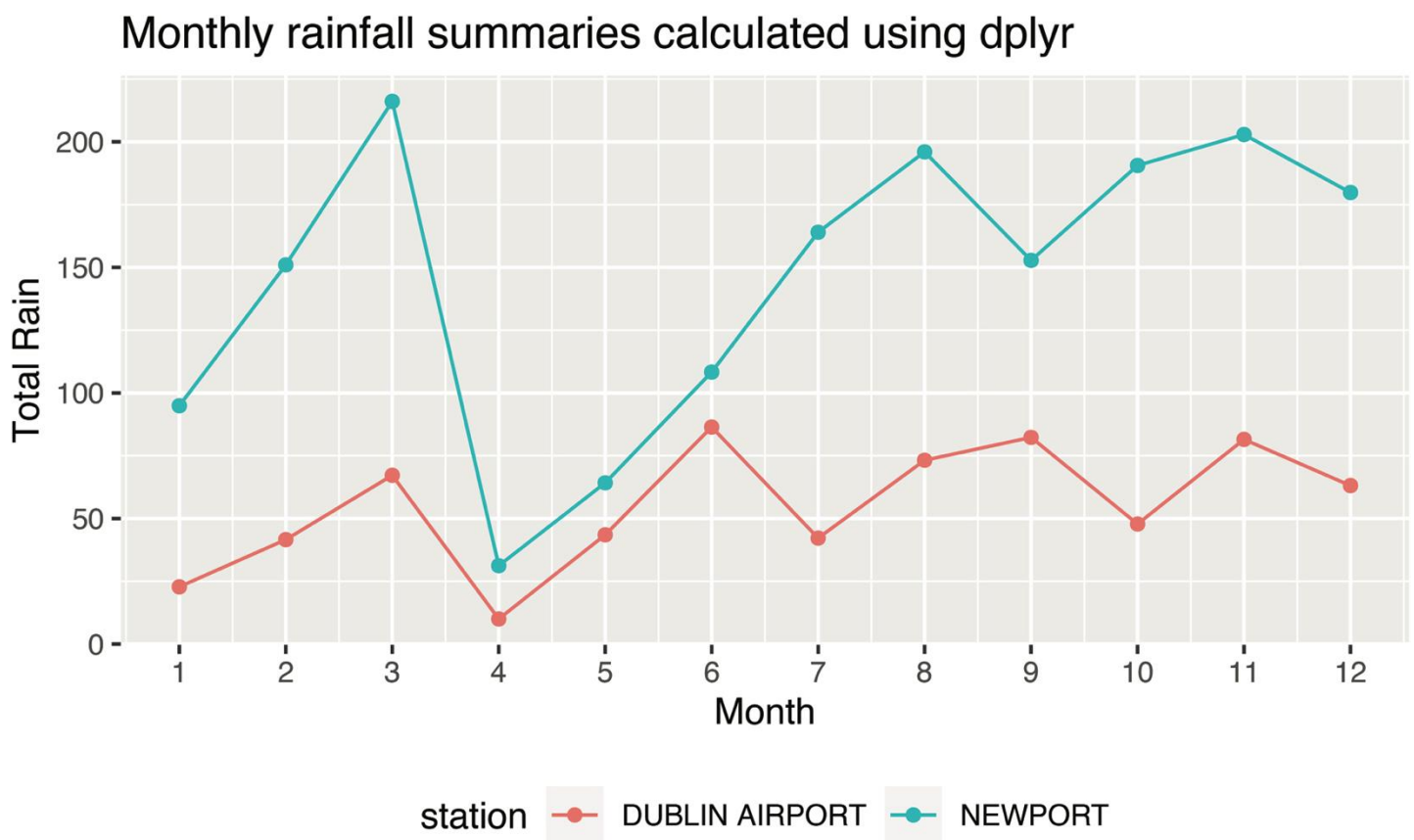


Getting monthly summaries (24 results)

```
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
```

```
dplyr::arrange(monthly_rain,month)
#> # A tibble: 24 x 3
#> # Groups:   station [2]
#>    station         month TotalRain
#>    <chr>           <dbl>     <dbl>
#>  1 DUBLIN AIRPORT      1      22.8
#>  2 NEWPORT             1      94.9
#>  3 DUBLIN AIRPORT      2      41.6
#>  4 NEWPORT             2     151
#>  5 DUBLIN AIRPORT      3      67.2
#>  6 NEWPORT             3     216.
#>  7 DUBLIN AIRPORT      4      10
#>  8 NEWPORT             4      31.2
#>  9 DUBLIN AIRPORT      5      43.5
#> 10 NEWPORT             5      64.2
#> # ... with 14 more rows
```

```r
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")
```



(6.10) Summary Functions

Function	Description
%>%	The tidyverse pipe operator (library magrittr).
%in%	Used to see if the left operand is in a vector.
filter()	Subsets rows based on column values.
slice()	Subsets rows based on their positions.
arrange()	Sorts rows based on column(s).
select()	Subsets columns based on names.
starts_with()	Matches starting column names.
ends_with()	Matches ending column names.
contains()	Matches column names that contain the input.
num_range()	Matches a numerical range.
matches()	Matches a regular expression.
mutate()	Creates new columns from existing variables.
group_by()	Converts a tibble to it into a grouped tibble.
ungroup()	Removes a tibble grouping.
summarize()	Creates a new data frame, based on summaries.
case_when()	Provides vectorization of multiple if_else() statement.
pull()	Similar to $ and useful if deployed with pipe

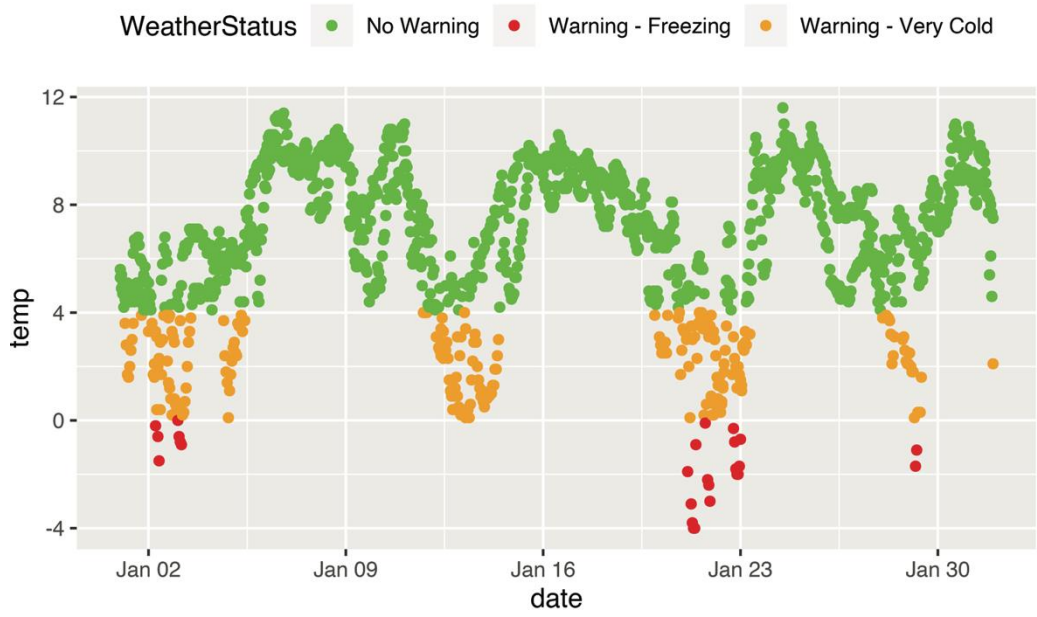
Challenge

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
```

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
```



3. Generate the following tibble (`diam`) based on the `diamonds` tibble from `ggplot2`. Note that the column `PriceMaxColor` is the color of the diamond with the maximum price for a given cut.

```
diam
#> # A tibble: 5 x 5
#>   cut       NumberDiamonds CaratMean PriceMax PriceMaxColor
#>   <ord>              <int>     <dbl>    <int> <ord>
#> 1 Ideal              21551     0.703    18806 G
#> 2 Premium            13791     0.892    18823 I
#> 3 Very Good          12082     0.806    18818 G
#> 4 Good                4906     0.849    18788 G
#> 5 Fair                1610     1.05     18574 G
```

4. For each class of car, create the tibble `mpg1` that contains a new column that stores the rank of city miles per gallon (`cty`), from lowest to highest. Make use of the `rank()` function in R, and in this function call, set the argument `ties.method` = "first".

```
mpg1
#> # A tibble: 234 x 7
#>    manufacturer model      displ  year   cty class   RankCty
#>    <chr>        <chr>      <dbl> <int> <int> <chr>     <int>
#>  1 chevrolet    corvette     5.7  1999    15 2seater       1
#>  2 chevrolet    corvette     6.2  2008    15 2seater       2
#>  3 chevrolet    corvette     7    2008    15 2seater       3
#>  4 chevrolet    corvette     5.7  1999    16 2seater       4
#>  5 chevrolet    corvette     6.2  2008    16 2seater       5
#>  6 audi         a4 quattro   2.8  1999    15 compact       1
#>  7 audi         a4 quattro   3.1  2008    15 compact       2
#>  8 audi         a4           2.8  1999    16 compact       3
#>  9 audi         a4 quattro   1.8  1999    16 compact       4
#> 10 volkswagen   jetta        2.8  1999    16 compact       5
#> # ... with 224 more rows
```

5. Find the stations with the highest (`temp_high`) and lowest (`temp_low`) annual average temperature values. Use these variables to calculate the average monthly temperature values for the two stations (`m_temps`), and display the data in a plot.

```
temp_low
#> [1] "KNOCK AIRPORT"
temp_high
#> [1] "VALENTIA OBSERVATORY"
arrange(m_temps,month,station)
#> # A tibble: 24 x 3
#>    station               month AvrTemp
#>    <chr>                 <dbl>   <dbl>
#>  1 KNOCK AIRPORT             1    5.18
#>  2 VALENTIA OBSERVATORY      1    8.03
#>  3 KNOCK AIRPORT             2    5.03
#>  4 VALENTIA OBSERVATORY      2    8.26
#>  5 KNOCK AIRPORT             3    6.78
#>  6 VALENTIA OBSERVATORY      3    9.36
#>  7 KNOCK AIRPORT             4    7.85
#>  8 VALENTIA OBSERVATORY      4    9.60
#>  9 KNOCK AIRPORT             5   11.6
#> 10 VALENTIA OBSERVATORY      5   12.6
#> # ... with 14 more rows
```