Tidyverse Recap: Travel and Weather - Part I dplyr

BDA 503 - Fall 2017

Nov 21, 2017

Introduction

This exercise is designed a recap to introduction to tidyverse from the very basics. Tidyverse is a collection of R packages used for data manipulation and visualization. We are mainly focused on `dplyr` and `ggplot2` (two most important packages of tidyverse) although we can use functionalities from other packages as well.

Suppose you are an frequent traveler and weather details are important to you because of what to pack to wear in your travels. Our data consists of temperature (in Celsius) history of 4 popular travel destinations (NYC, Amsterdam, London and Venice) between November 2015 and October 2017. Raw data is gathered from Weather Underground and it is only for educational purposes. You are going to explore this data set using the most common tidyverse functions. You will be asked to fill the missing information.

Tip: You can always check the help files of the functions by writing `?` in front of the function name (e.g. `?select`) in the R Console, after you load the package.

Preparation

First we are going to install `tidyverse` and load it. Installing a package is a one time job, essentially equivalent to downloading from server. Though, in each session you need to load the package with either `library` or `require` functions. For this tutorial you also need to download the `travel_weather.RData` file from here.

```
# Install the package if you already haven't
install.packages("tidyverse", repos = "https://cran.r-project.org")
# Load the package to the session
library(tidyverse)
# Set your working directory (the directory which you keep
# the travel data (travel_weather.RData)
setwd("~/MyWorkingDirectory/")
# Load the data set file
load("travel_weather.RData")
```

Main data type of this tutorial is a `data.frame`, or more properly a `tibble`. Data frames are two dimensional, efficient data tables which every column can consist of different data types (i.e. characters, factors, numeric, logical). `tibble` is a special data frame type that comes with tidyverse package but the functionality is very similar (no difference for this tutorial).

Now let's take a look at our data.

Travel Weather Data

```
travel_weather %>%
    tbl_df()
```

```
## # A tibble: 731 x 7
##     year month   day Amsterdam London   NYC Venice
```

```
##   * <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>  <dbl>
##   1  2015    11     1         8      8    16     13
##   2  2015    11     2        10     11    15     10
##   3  2015    11     3         9     11    16      9
##   4  2015    11     4        12     11    17     10
##   5  2015    11     5        13     13    18     12
##   6  2015    11     6        16     14    21     13
##   7  2015    11     7        16     14    17     14
##   8  2015    11     8        12     12    11     13
##   9  2015    11     9        13     12    11     11
##  10  2015    11    10        14     14    12     11
## # ... with 721 more rows
```

Did you notice the %>%? It is called the pipe operator. It starts with the data and connects the operations in the given order (top to bottom or left to right). (*Tip*: You can add line breaks between the operations but pipe operator should always be at the end of the line.)

There are some `tibble` properties you should be aware of. At the first line number of rows and columns are reported (A tibble: 731x7). Also under each column, its data type is given. This way we can be notified of the essentials of this data frame.

A more proper check can be done with `glimpse` function. `glimpse` is especially useful if the number of columns is high.

```
glimpse(travel_weather)
```

```
## Observations: 731
## Variables: 7
## $ year      <dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015, 2015...
## $ month     <dbl> 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, ...
## $ day       <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1...
## $ Amsterdam <dbl> 8, 10, 9, 12, 13, 16, 16, 12, 13, 14, 13, 13, 11, 11...
## $ London    <dbl> 8, 11, 11, 11, 13, 14, 14, 12, 12, 14, 13, 12, 10, 1...
## $ NYC       <dbl> 16, 15, 16, 17, 18, 21, 17, 11, 11, 12, 12, 13, 11, ...
## $ Venice    <dbl> 13, 10, 9, 10, 12, 13, 14, 13, 11, 11, 9, 11, 8, 11,...
```

Our data consists of 731 rows and 7 columns. Each row represents a day. First three columns (`year`,`month` and `day`) define the date. Last four columns (`Amsterdam`, `London`, `NYC` and `Venice`) represent the average temperature of the cities in the given day.

Now let's explore.

dplyr

We are going to see the fundamental functions of dplyr and then some more. It would be very good for you if you follow this tutorial with the dplyr cheat sheet. You can download it from here. Our fundamental functions are.

- `select/rename`
- `filter`
- `arrange`
- `mutate/transmute`
- `group_by/summarise`

We will start really simple and build up.

select/rename

`select` function, as the name suggests, selects the columns. `rename` just renames the columns.

1. Let's start with only one city: Venice. Select the date components (year, month, day) and Venice column. Fill the `YOURANSWERHERE` in your code in order to replicate the result.

```
travel_weather %>% select(year, month, day, YOURANSWERHERE)
```

```
## # A tibble: 731 x 4
##     year month   day Venice
##  * <dbl> <dbl> <dbl>  <dbl>
##  1  2015    11     1     13
##  2  2015    11     2     10
##  3  2015    11     3      9
##  4  2015    11     4     10
##  5  2015    11     5     12
##  6  2015    11     6     13
##  7  2015    11     7     14
##  8  2015    11     8     13
##  9  2015    11     9     11
## 10  2015    11    10     11
## # ... with 721 more rows
```

2. Now let's say you want to have only the cities. You can either write the names of all cities or specify a range with `:`.

```
travel_weather %>% select(YOURANSWERHERE1:YOURANSWERHERE2)
```

```
## # A tibble: 731 x 4
##    Amsterdam London   NYC Venice
##  *     <dbl>  <dbl> <dbl>  <dbl>
##  1         8      8    16     13
##  2        10     11    15     10
##  3         9     11    16      9
##  4        12     11    17     10
##  5        13     13    18     12
##  6        16     14    21     13
##  7        16     14    17     14
##  8        12     12    11     13
##  9        13     12    11     11
## 10        14     14    12     11
## # ... with 721 more rows
```

3. This time we are going to use (`-`) to remove unwanted columns. Suppose we do not want NYC or London columns.

```
travel_weather %>% select(-YOURANSWERHERE1, -YOURANSWERHERE2)
```

```
## # A tibble: 731 x 5
##     year month   day Amsterdam Venice
##  * <dbl> <dbl> <dbl>     <dbl>  <dbl>
##  1  2015    11     1         8     13
##  2  2015    11     2        10     10
##  3  2015    11     3         9      9
##  4  2015    11     4        12     10
##  5  2015    11     5        13     12
##  6  2015    11     6        16     13
```

```
##  7  2015    11     7        16     14
##  8  2015    11     8        12     13
##  9  2015    11     9        13     11
## 10  2015    11    10        14     11
## # ... with 721 more rows
```

4. Now we just want to rename NYC to New York. Although it is not advised to use spaces in your column names, you can do it by taking it between backticks. Remember `rename` will not select any column, just change the name of the specified column.

```
travel_weather %>% rename(`YOUR ANSWER HERE` = NYC)
```

```
## # A tibble: 731 x 7
##     year month   day Amsterdam London `New York` Venice
##  * <dbl> <dbl> <dbl>     <dbl>  <dbl>      <dbl>  <dbl>
##  1  2015    11     1         8      8         16     13
##  2  2015    11     2        10     11         15     10
##  3  2015    11     3         9     11         16      9
##  4  2015    11     4        12     11         17     10
##  5  2015    11     5        13     13         18     12
##  6  2015    11     6        16     14         21     13
##  7  2015    11     7        16     14         17     14
##  8  2015    11     8        12     12         11     13
##  9  2015    11     9        13     12         11     11
## 10  2015    11    10        14     14         12     11
## # ... with 721 more rows
```

Tip: You can also use rename functionality with `select`.

filter

Filter returns rows with the given criteria. You can define any criteria and combine conditions with the "and" (`&`) and "or" (`|`) operators. You can use other operators such as less than (or equal to) (`<`,`<=`), greater than (or equal to) (`>`,`>=`), equal to (not equal to) (`=`, `!=`) and several other operators which return TRUE/FALSE statements as well. You can combine the operations and ensure precedence with parentheses.

1. Suppose we are interested only the first three days of the month.

```
travel_weather %>%
    filter(day <= YOURANSWERHERE)
```

```
## # A tibble: 72 x 7
##     year month   day Amsterdam London   NYC Venice
##    <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>  <dbl>
##  1  2015    11     1         8      8    16     13
##  2  2015    11     2        10     11    15     10
##  3  2015    11     3         9     11    16      9
##  4  2015    12     1         9     11     9      6
##  5  2015    12     2        10     12    11      8
##  6  2015    12     3         9     11    10      8
##  7  2016     1     1         4      3     3      2
##  8  2016     1     2         6     10     2      0
##  9  2016     1     3         7      8     4      3
## 10  2016     2     1        10     12    11      6
## # ... with 62 more rows
```

2. Suppose we are interested in only the dates in November (11th month) which Venice is warmer than NYC.

```
travel_weather %>%
    filter(month == 11 & YOURANSWERHERE)
```

```
## # A tibble: 20 x 7
##     year month   day Amsterdam London   NYC Venice
##    <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>  <dbl>
##  1  2015    11     8        12     12    11     13
##  2  2015    11    14        11     10     8     11
##  3  2015    11    15        12     14     9     11
##  4  2015    11    17        13     13     8      9
##  5  2015    11    23         3      3     4      6
##  6  2015    11    24         5      8     4      6
##  7  2016    11     1        10      9     9     11
##  8  2016    11     6         7      4    11     12
##  9  2016    11     7         4      6     8     11
## 10  2016    11    12         1      8     7      9
## 11  2016    11    19         6      4    10     11
## 12  2016    11    20         7      7     3     11
## 13  2016    11    21        10     10     4     12
## 14  2016    11    22        10      9     4     14
## 15  2016    11    23         8      7     4     14
## 16  2016    11    24         6      9     6     13
## 17  2016    11    25         3      7    10     13
## 18  2016    11    26         3      6     7     12
## 19  2016    11    27         5      7     7     11
## 20  2016    11    28         1      6     7      8
```

3. Suppose we are interested in dates whether Amsterdam is warmer than either London or Venice in July (7th month).

```
travel_weather %>%
    filter(month == 7 & (YOURANSWERHERE1 | YOURANSWERHERE2))
```

```
## # A tibble: 21 x 7
##     year month   day Amsterdam London   NYC Venice
##    <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>  <dbl>
##  1  2016     7     2        16     14    21     25
##  2  2016     7    11        19     18    23     27
##  3  2016     7    12        18     17    24     28
##  4  2016     7    13        16     14    26     27
##  5  2016     7    19        21     20    26     27
##  6  2016     7    20        27     24    25     26
##  7  2016     7    21        21     19    27     26
##  8  2016     7    22        21     19    29     26
##  9  2016     7    23        22     19    31     26
## 10  2016     7    24        21     19    29     25
## # ... with 11 more rows
```

4. Finally, let's add some math. Suppose we are interested in dates which the absolute temperature difference between Amsterdam and Venice is greater than or equal to 12.

```
travel_weather %>%
    filter(abs(YOURANSWERHERE) >= 12)
```

```
## # A tibble: 6 x 7
##    year month   day Amsterdam London   NYC Venice
```

```
##   <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>  <dbl>
## 1  2016     6    25        16     15    24     28
## 2  2017     7    13        14     14    29     27
## 3  2017     8     2        18     17    26     30
## 4  2017     8     4        19     18    25     31
## 5  2017     8     5        17     16    23     31
## 6  2017     8     6        16     17    21     29
```

arrange

`arrange` is simply ordering of values from A to Z or from smallest to largest. Just write the column names in the order you want to arrange. To employ `arrange` in a decreasing order wrap the column of interest between `desc(column_name)` function.

1. Arrange the data by the temperature of NYC.

```
travel_weather %>%
    arrange(YOURANSWERHERE)
```

```
## # A tibble: 731 x 7
##     year month   day Amsterdam London   NYC Venice
##    <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>  <dbl>
##  1  2016     2    14         2      3   -14      6
##  2  2016     2    13         1      2   -10      4
##  3  2016     1     5         6      8    -7      2
##  4  2017     1     9         6      7    -7     -2
##  5  2016     1    19        -2      0    -6      1
##  6  2016     2    12         2      1    -6      6
##  7  2016    12    16         6      6    -6      4
##  8  2017     1     8         4      9    -6     -2
##  9  2017     1     7         1      8    -5     -3
## 10  2017     3    11         7     10    -5      9
## # ... with 721 more rows
```

2. Arrange the data by the temperature of NYC increasing but Amsterdam decreasing.

```
travel_weather %>%
    arrange(YOURANSWERHERE1,desc(YOURANSWERHERE2))
```

```
## # A tibble: 731 x 7
##     year month   day Amsterdam London   NYC Venice
##    <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>  <dbl>
##  1  2016     2    14         2      3   -14      6
##  2  2016     2    13         1      2   -10      4
##  3  2016     1     5         6      8    -7      2
##  4  2017     1     9         6      7    -7     -2
##  5  2016    12    16         6      6    -6      4
##  6  2017     1     8         4      9    -6     -2
##  7  2016     2    12         2      1    -6      6
##  8  2016     1    19        -2      0    -6      1
##  9  2017     3    15         9     11    -5     10
## 10  2017     3    11         7     10    -5      9
## # ... with 721 more rows
```

3. Arrange the data by the decreasing date.

```
travel_weather %>%
    arrange(YOURANSWERHERE)
```

```
## # A tibble: 731 x 7
##     year month   day Amsterdam London   NYC Venice
##    <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>  <dbl>
##  1  2017    10    31         9      9    11     11
##  2  2017    10    30         8      6    12     13
##  3  2017    10    29        11     11    18      9
##  4  2017    10    28        12     10    17     10
##  5  2017    10    27        12      9    13     13
##  6  2017    10    26        13     10    13     13
##  7  2017    10    25        13     14    17     13
##  8  2017    10    24        13     16    21     13
##  9  2017    10    23        13     13    20     13
## 10  2017    10    22        11     11    19     13
## # ... with 721 more rows
```

4. Finally arrange the data by the temperature difference between London and Amsterdam, increasing.

```
travel_weather %>%
    arrange(YOURANSWERHERE1 - YOURANSWERHERE2)
```

```
## # A tibble: 731 x 7
##     year month   day Amsterdam London   NYC Venice
##    <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>  <dbl>
##  1  2016    12    25        10      0     6      6
##  2  2015    12    25         9      0    17      4
##  3  2016     5    31        18     11    26     19
##  4  2016     6     1        19     12    24     17
##  5  2016     4    10        10      4     5     16
##  6  2016     6     7        20     14    24     22
##  7  2016     5     6        17     12    11     18
##  8  2016     5     8        21     16    14     17
##  9  2016     5    10        19     14    14     18
## 10  2016     6     3        16     11    19     19
## # ... with 721 more rows
```

mutate/transmute

`mutate` function is used for calculations between columns. `transmute` is similar but it adds the `select` effect, therefore returning only the columns defined in the `transmute` function.

1. Calculate the temperature difference between Venice and Amsterdam.

```
travel_weather %>%
    mutate(VAdiff = YOURANSWERHERE1 - YOURANSWERHERE2)
```

```
## # A tibble: 731 x 8
##     year month   day Amsterdam London   NYC Venice VAdiff
##    <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>  <dbl>  <dbl>
##  1  2015    11     1         8      8    16     13      5
##  2  2015    11     2        10     11    15     10      0
##  3  2015    11     3         9     11    16      9      0
##  4  2015    11     4        12     11    17     10     -2
```

```
##  5  2015    11     5        13     13    18     12     -1
##  6  2015    11     6        16     14    21     13     -3
##  7  2015    11     7        16     14    17     14     -2
##  8  2015    11     8        12     12    11     13      1
##  9  2015    11     9        13     12    11     11     -2
## 10  2015    11    10        14     14    12     11     -3
## # ... with 721 more rows
```

2. Calculate if Venice is warmer than Amsterdam.

```
travel_weather %>%
    mutate(VwarmerA = YOURANSWERHERE1 > YOURANSWERHERE2)
```

```
## # A tibble: 731 x 8
##     year month   day Amsterdam London   NYC Venice VwarmerA
##    <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>  <dbl>    <lgl>
##  1  2015    11     1         8      8    16     13     TRUE
##  2  2015    11     2        10     11    15     10    FALSE
##  3  2015    11     3         9     11    16      9    FALSE
##  4  2015    11     4        12     11    17     10    FALSE
##  5  2015    11     5        13     13    18     12    FALSE
##  6  2015    11     6        16     14    21     13    FALSE
##  7  2015    11     7        16     14    17     14    FALSE
##  8  2015    11     8        12     12    11     13     TRUE
##  9  2015    11     9        13     12    11     11    FALSE
## 10  2015    11    10        14     14    12     11    FALSE
## # ... with 721 more rows
```

3. If Venice is warmer than Amsterdam write "warmer", else "colder" and just return the date columns and warmer/colder info.

```
travel_weather %>%
    transmute(year,month,day,
        VwarmerA = ifelse(Venice > Amsterdam,YOURANSWERHERE1, YOURANSWERHERE2))
```

```
## # A tibble: 731 x 4
##     year month   day VwarmerA
##    <dbl> <dbl> <dbl>    <chr>
##  1  2015    11     1   warmer
##  2  2015    11     2   colder
##  3  2015    11     3   colder
##  4  2015    11     4   colder
##  5  2015    11     5   colder
##  6  2015    11     6   colder
##  7  2015    11     7   colder
##  8  2015    11     8   warmer
##  9  2015    11     9   colder
## 10  2015    11    10   colder
## # ... with 721 more rows
```

group_by/summarise

`group_by` and `summarise` are used for summary tables (sometimes referred to as pivot tables, especially for Excel users). Summarise can be used on its own or with the grouping function `group_by`. This part is also the first part which you will use more than one pipe (`%>%`).

Tip: If you want to break the grouping, just add the `ungroup()` function at the end.

1. Calculate the mean temperatures of Venice and NYC of data period.

```
travel_weather %>%
    summarise(Venice_mean=mean(YOURANSWERHERE1),NYC_mean=YOURANSWERHERE2)
```

```
## # A tibble: 1 x 2
##   Venice_mean NYC_mean
##         <dbl>    <dbl>
## 1    14.31601 14.41313
```

2. Calculate the mean temperature of Amsterdam for each month. Round the value to two decimals.

```
travel_weather %>%
    group_by(YOURANSWERHERE1) %>%
    summarise(Amsterdam_mean=mean(YOURANSWERHERE2))
```

```
## # A tibble: 12 x 2
##    month Amsterdam_mean
##    <dbl>          <dbl>
##  1     1           3.00
##  2     2           4.32
##  3     3           6.92
##  4     4           8.43
##  5     5          14.48
##  6     6          17.28
##  7     7          18.02
##  8     8          17.68
##  9     9          15.95
## 10    10          11.68
## 11    11           7.65
## 12    12           6.97
```

3. Calculate the number of days Amsterdam is warmer than NYC each year and each month.

```
travel_weather %>%
    group_by(year,month) %>%
    summarise(AwarmerN_n=sum(YOURANSWERHERE1 > YOURANSWERHERE2))
```

```
## # A tibble: 24 x 3
## # Groups:   year [?]
##     year month AwarmerN_n
##    <dbl> <dbl>      <int>
##  1  2015    11         11
##  2  2015    12         12
##  3  2016     1         23
##  4  2016     2         16
##  5  2016     3          5
##  6  2016     4         10
##  7  2016     5          8
##  8  2016     6          1
##  9  2016     7          1
## 10  2016     8          0
## # ... with 14 more rows
```

4. Calculate the maximum, minimum and median temperature values of London for each month and each year.

```
travel_weather %>%
    group_by(year,month) %>%
    summarise(London_min=YOURANSWERHERE1,London_median=median(London),London_max=YOURANSWERHERE2)
```

```
## # A tibble: 24 x 5
## # Groups:   year [?]
##     year month London_min London_median London_max
##    <dbl> <dbl>      <dbl>         <dbl>      <dbl>
##  1  2015    11          1            11         14
##  2  2015    12          0            10         14
##  3  2016     1          0             6         11
##  4  2016     2          1             4         12
##  5  2016     3          2             6         11
##  6  2016     4          4             8         11
##  7  2016     5          8            13         16
##  8  2016     6         11            16         19
##  9  2016     7         14            18         24
## 10  2016     8         14            18         24
## # ... with 14 more rows
```

Advanced Examples

Here is a showcase of some advanced examples of tidyverse data manipulation power.

Lead and Lag

Sometimes you want to have the differences between consecutive rows. Then you can use `lag` and `lead` functions. Suppose we want to calculate the

```
travel_weather %>%
    transmute(year,month,day,Amsterdam,A_prev=lag(Amsterdam),A_next=lead(Amsterdam),
                A_prev_diff=Amsterdam-A_prev,A_next_diff=Amsterdam-A_next)
```

```
## # A tibble: 731 x 8
##     year month   day Amsterdam A_prev A_next A_prev_diff A_next_diff
##    <dbl> <dbl> <dbl>     <dbl>  <dbl>  <dbl>       <dbl>       <dbl>
##  1  2015    11     1         8     NA     10          NA          -2
##  2  2015    11     2        10      8      9           2           1
##  3  2015    11     3         9     10     12          -1          -3
##  4  2015    11     4        12      9     13           3          -1
##  5  2015    11     5        13     12     16           1          -3
##  6  2015    11     6        16     13     16           3           0
##  7  2015    11     7        16     16     12           0           4
##  8  2015    11     8        12     16     13          -4          -1
##  9  2015    11     9        13     12     14           1          -1
## 10  2015    11    10        14     13     13           1           1
## # ... with 721 more rows
```

slice

Slice function returns the rows with the given indexes.

```
travel_weather %>%
    slice(1:3)
```

```
## # A tibble: 3 x 7
##    year month   day Amsterdam London   NYC Venice
##   <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>  <dbl>
## 1  2015    11     1         8      8    16     13
## 2  2015    11     2        10     11    15     10
## 3  2015    11     3         9     11    16      9
```

It can also be combined with the `group_by` function.

```
travel_weather %>%
    group_by(year) %>%
    slice(1:3)
```

```
## # A tibble: 9 x 7
## # Groups:   year [3]
##    year month   day Amsterdam London   NYC Venice
##   <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>  <dbl>
## 1  2015    11     1         8      8    16     13
## 2  2015    11     2        10     11    15     10
## 3  2015    11     3         9     11    16      9
## 4  2016     1     1         4      3     3      2
## 5  2016     1     2         6     10     2      0
## 6  2016     1     3         7      8     4      3
## 7  2017     1     1         1      7     7      2
## 8  2017     1     2         3      2     3      1
## 9  2017     1     3         4      2     5      3
```

But be careful using the slice function as it only returns rows by the index value.

Gather and Spread

You might need to transform your data from wide (many columns) to long format (less columnns) or vice versa. They are also called melting and casting. Then you can use `gather` and `spread` functions respectively. They can be a bit confusing at first but you can quickly get used to them.

Suppose we want to see a summary table of average temperatures of each city for each month. But we want the cities as rows and months as columns.

```
#Transform to long format by melting the data
#Though you should not include date columns
travel_weather_long <-
travel_weather %>%
    gather(key=City,value=Temperature,-year,-month,-day)

travel_weather_long
```

```
## # A tibble: 2,924 x 5
##     year month   day      City Temperature
##    <dbl> <dbl> <dbl>     <chr>       <dbl>
##  1  2015    11     1 Amsterdam           8
##  2  2015    11     2 Amsterdam          10
##  3  2015    11     3 Amsterdam           9
##  4  2015    11     4 Amsterdam          12
```

```
##  5  2015    11     5 Amsterdam          13
##  6  2015    11     6 Amsterdam          16
##  7  2015    11     7 Amsterdam          16
##  8  2015    11     8 Amsterdam          12
##  9  2015    11     9 Amsterdam          13
## 10  2015    11    10 Amsterdam          14
## # ... with 2,914 more rows
```

```
#Now group by and summarise to get average temperatures for each city and month
travel_weather_long %>%
    group_by(month,City) %>%
    summarise(temp_avg=round(mean(Temperature))) %>%
    #Now spread the months to the columns
    spread(month,temp_avg)
```

```
## # A tibble: 4 x 13
##        City   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`  `10`  `11`  `12`
## *     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Amsterdam     3     4     7     8    14    17    18    18    16    12     8     7
## 2    London     4     6     8     9    13    17    18    18    16    12     9     8
## 3       NYC     2     4     7    13    17    22    26    25    22    16    11     7
## 4    Venice     2     7    11    14    18    22    25    25    20    14     9     5
```

_all and _at prefixes

Especially `mutate` and `summarise` has some special functions defined with "all" and "at" (in the previous versions "each") suffixes.

Let's get the average temperatures of all cities. We can do it in two ways. First select the cities and use `summarise_all` or select cities in `summarise_at`.

```
#Method 1
travel_weather %>%
    select(Amsterdam:Venice) %>%
    summarise_all(funs(round(mean(.))))
```

```
## # A tibble: 1 x 4
##   Amsterdam London   NYC Venice
##       <dbl>  <dbl> <dbl>  <dbl>
## 1        11     12    14     14
```

```
#Method 2
travel_weather %>%
    summarise_at(vars(Amsterdam:Venice),funs(round(mean(.))))
```

```
## # A tibble: 1 x 4
##   Amsterdam London   NYC Venice
##       <dbl>  <dbl> <dbl>  <dbl>
## 1        11     12    14     14
```

We can use the `mutate_at` function to see all other cities' temperature differences from NYC.

```
#Method 2
travel_weather %>%
    mutate_at(vars(Amsterdam,London,Venice),funs(diff_NYC=abs(NYC-.))) %>%
    select(-Amsterdam,-London,-Venice)
```

```
## # A tibble: 731 x 7
##     year month   day   NYC Amsterdam_diff_NYC London_diff_NYC Venice_diff_NYC
##    <dbl> <dbl> <dbl> <dbl>              <dbl>           <dbl>            <dbl>
##  1  2015    11     1    16                  8               8                3
##  2  2015    11     2    15                  5               4                5
##  3  2015    11     3    16                  7               5                7
##  4  2015    11     4    17                  5               6                7
##  5  2015    11     5    18                  5               5                6
##  6  2015    11     6    21                  5               7                8
##  7  2015    11     7    17                  1               3                3
##  8  2015    11     8    11                  1               1                2
##  9  2015    11     9    11                  2               1                0
## 10  2015    11    10    12                  2               2                1
## # ... with 721 more rows
```

Final Exercises

These exercises are left to the students to test themselves. Try to write the code to replicate the results.

1. Return the dates which Amsterdam is strictly warmer than London but strictly colder than Venice

```
## # A tibble: 165 x 7
##     year month   day Amsterdam London   NYC Venice
##    <dbl> <dbl> <dbl>     <dbl>  <dbl> <dbl>  <dbl>
##  1  2015    11    21         5      3     9      8
##  2  2015    11    22         3      1     9      8
##  3  2016     1    13         4      3    -3      6
##  4  2016     1    16         2      1     8      4
##  5  2016     2     3         5      4    11      8
##  6  2016     2    11         4      3    -4      7
##  7  2016     2    12         2      1    -6      6
##  8  2016     2    23         4      3     3     11
##  9  2016     2    24         2      1     9     10
## 10  2016     2    25         2      1     9      8
## # ... with 155 more rows
```

2. For each month of each year calculate the average difference between NYC and Amsterdam for the days NYC is strictly warmer than Amsterdam, rounded by 1 decimal. Arrange from the highest difference to the lowest.

```
## # A tibble: 24 x 3
## # Groups:   year [3]
##     year month NYCwA_diff
##    <dbl> <dbl>      <dbl>
##  1  2016     8        8.4
##  2  2016     7        8.1
##  3  2017     9        7.9
##  4  2016     4        7.5
##  5  2017     4        7.4
##  6  2017     7        7.3
##  7  2017     8        6.5
##  8  2016    11        6.4
##  9  2016     3        6.3
## 10  2016     6        6.0
```

```
## # ... with 14 more rows
```

3. Return the warmest city and its temperature of each day.

```
## # A tibble: 731 x 5
## # Groups:   year, month, day [731]
##     year month   day      City Temperature
##    <dbl> <dbl> <dbl>     <chr>       <dbl>
##  1  2015    11     1       NYC          16
##  2  2015    11     2       NYC          15
##  3  2015    11     3       NYC          16
##  4  2015    11     4       NYC          17
##  5  2015    11     5       NYC          18
##  6  2015    11     6       NYC          21
##  7  2015    11     7       NYC          17
##  8  2015    11     8    Venice          13
##  9  2015    11     9 Amsterdam          13
## 10  2015    11    10 Amsterdam          14
## # ... with 721 more rows
```