Tidyverse Intro: Travel and Weather

ETM 58D - Spring 2018

Feb 26, 2018

Introduction

This exercise is designed as an 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 below link.

Download the data set

```
# 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  11.0  1.00      8.00   8.00  16.0  13.0
##  2  2015  11.0  2.00     10.0   11.0   15.0  10.0
##  3  2015  11.0  3.00      9.00  11.0   16.0   9.00
##  4  2015  11.0  4.00     12.0   11.0   17.0  10.0
##  5  2015  11.0  5.00     13.0   13.0   18.0  12.0
##  6  2015  11.0  6.00     16.0   14.0   21.0  13.0
##  7  2015  11.0  7.00     16.0   14.0   17.0  14.0
##  8  2015  11.0  8.00     12.0   12.0   11.0  13.0
##  9  2015  11.0  9.00     13.0   12.0   11.0  11.0
## 10  2015  11.0 10.0      14.0   14.0   12.0  11.0
## # ... 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 as follows.

- `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  11.0  1.00  13.0
##  2  2015  11.0  2.00  10.0
##  3  2015  11.0  3.00   9.00
##  4  2015  11.0  4.00  10.0
##  5  2015  11.0  5.00  12.0
##  6  2015  11.0  6.00  13.0
##  7  2015  11.0  7.00  14.0
##  8  2015  11.0  8.00  13.0
##  9  2015  11.0  9.00  11.0
## 10  2015  11.0 10.0   11.0
## # ... 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      8.00   8.00  16.0  13.0
##  2     10.0   11.0   15.0  10.0
##  3      9.00  11.0   16.0   9.00
##  4     12.0   11.0   17.0  10.0
##  5     13.0   13.0   18.0  12.0
##  6     16.0   14.0   21.0  13.0
##  7     16.0   14.0   17.0  14.0
##  8     12.0   12.0   11.0  13.0
##  9     13.0   12.0   11.0  11.0
## 10     14.0   14.0   12.0  11.0
## # ... 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  11.0  1.00      8.00  13.0
##   2  2015  11.0  2.00     10.0   10.0
##   3  2015  11.0  3.00      9.00   9.00
##   4  2015  11.0  4.00     12.0   10.0
##   5  2015  11.0  5.00     13.0   12.0
##   6  2015  11.0  6.00     16.0   13.0
##   7  2015  11.0  7.00     16.0   14.0
##   8  2015  11.0  8.00     12.0   13.0
##   9  2015  11.0  9.00     13.0   11.0
##  10  2015  11.0 10.0      14.0   11.0
## # ... 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  11.0  1.00      8.00   8.00       16.0  13.0
##   2  2015  11.0  2.00     10.0   11.0        15.0  10.0
##   3  2015  11.0  3.00      9.00  11.0        16.0   9.00
##   4  2015  11.0  4.00     12.0   11.0        17.0  10.0
##   5  2015  11.0  5.00     13.0   13.0        18.0  12.0
##   6  2015  11.0  6.00     16.0   14.0        21.0  13.0
##   7  2015  11.0  7.00     16.0   14.0        17.0  14.0
##   8  2015  11.0  8.00     12.0   12.0        11.0  13.0
##   9  2015  11.0  9.00     13.0   12.0        11.0  11.0
##  10  2015  11.0 10.0      14.0   14.0        12.0  11.0
## # ... 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 11.0   1.00      8.00   8.00 16.0   13.0
##   2  2015 11.0   2.00     10.0   11.0  15.0   10.0
##   3  2015 11.0   3.00      9.00  11.0  16.0    9.00
##   4  2015 12.0   1.00      9.00  11.0   9.00   6.00
##   5  2015 12.0   2.00     10.0   12.0  11.0    8.00
```

```
##  6  2015 12.0   3.00      9.00  11.0  10.0    8.00
##  7  2016  1.00  1.00      4.00   3.00  3.00   2.00
##  8  2016  1.00  2.00      6.00  10.0   2.00   0
##  9  2016  1.00  3.00      7.00   8.00  4.00   3.00
## 10  2016  2.00  1.00     10.0   12.0  11.0    6.00
## # ... 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  11.0  8.00     12.0   12.0  11.0   13.0
##  2  2015  11.0 14.0      11.0   10.0   8.00  11.0
##  3  2015  11.0 15.0      12.0   14.0   9.00  11.0
##  4  2015  11.0 17.0      13.0   13.0   8.00   9.00
##  5  2015  11.0 23.0       3.00   3.00  4.00   6.00
##  6  2015  11.0 24.0       5.00   8.00  4.00   6.00
##  7  2016  11.0  1.00     10.0    9.00  9.00  11.0
##  8  2016  11.0  6.00      7.00   4.00 11.0   12.0
##  9  2016  11.0  7.00      4.00   6.00  8.00  11.0
## 10  2016  11.0 12.0       1.00   8.00  7.00   9.00
## 11  2016  11.0 19.0       6.00   4.00 10.0   11.0
## 12  2016  11.0 20.0       7.00   7.00  3.00  11.0
## 13  2016  11.0 21.0      10.0   10.0   4.00  12.0
## 14  2016  11.0 22.0      10.0    9.00  4.00  14.0
## 15  2016  11.0 23.0       8.00   7.00  4.00  14.0
## 16  2016  11.0 24.0       6.00   9.00  6.00  13.0
## 17  2016  11.0 25.0       3.00   7.00 10.0   13.0
## 18  2016  11.0 26.0       3.00   6.00  7.00  12.0
## 19  2016  11.0 27.0       5.00   7.00  7.00  11.0
## 20  2016  11.0 28.0       1.00   6.00  7.00   8.00
```

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  7.00  2.00      16.0   14.0  21.0   25.0
##  2  2016  7.00 11.0       19.0   18.0  23.0   27.0
##  3  2016  7.00 12.0       18.0   17.0  24.0   28.0
##  4  2016  7.00 13.0       16.0   14.0  26.0   27.0
##  5  2016  7.00 19.0       21.0   20.0  26.0   27.0
##  6  2016  7.00 20.0       27.0   24.0  25.0   26.0
##  7  2016  7.00 21.0       21.0   19.0  27.0   26.0
##  8  2016  7.00 22.0       21.0   19.0  29.0   26.0
##  9  2016  7.00 23.0       22.0   19.0  31.0   26.0
## 10  2016  7.00 24.0       21.0   19.0  29.0   25.0
## # ... 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  6.00 25.0       16.0   15.0  24.0   28.0
## 2  2017  7.00 13.0       14.0   14.0  29.0   27.0
## 3  2017  8.00  2.00      18.0   17.0  26.0   30.0
## 4  2017  8.00  4.00      19.0   18.0  25.0   31.0
## 5  2017  8.00  5.00      17.0   16.0  23.0   31.0
## 6  2017  8.00  6.00      16.0   17.0  21.0   29.0
```

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  2.00 14.0       2.00   3.00 -14.0    6.00
##  2  2016  2.00 13.0       1.00   2.00 -10.0    4.00
##  3  2016  1.00  5.00      6.00   8.00 - 7.00   2.00
##  4  2017  1.00  9.00      6.00   7.00 - 7.00  -2.00
##  5  2016  1.00 19.0      -2.00   0    - 6.00   1.00
##  6  2016  2.00 12.0       2.00   1.00 - 6.00   6.00
##  7  2016 12.0  16.0       6.00   6.00 - 6.00   4.00
##  8  2017  1.00  8.00      4.00   9.00 - 6.00  -2.00
##  9  2017  1.00  7.00      1.00   8.00 - 5.00  -3.00
## 10  2017  3.00 11.0       7.00  10.0  - 5.00   9.00
## # ... 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  2.00 14.0       2.00   3.00 -14.0    6.00
##  2  2016  2.00 13.0       1.00   2.00 -10.0    4.00
##  3  2016  1.00  5.00      6.00   8.00 - 7.00   2.00
##  4  2017  1.00  9.00      6.00   7.00 - 7.00 - 2.00
##  5  2016 12.0  16.0       6.00   6.00 - 6.00   4.00
##  6  2017  1.00  8.00      4.00   9.00 - 6.00 - 2.00
```

```
##  7  2016  2.00 12.0       2.00   1.00 - 6.00   6.00
##  8  2016  1.00 19.0      -2.00   0    - 6.00   1.00
##  9  2017  3.00 15.0       9.00  11.0  - 5.00  10.0
## 10  2017  3.00 11.0       7.00  10.0  - 5.00   9.00
## # ... 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  10.0  31.0      9.00   9.00  11.0  11.0
##  2  2017  10.0  30.0      8.00   6.00  12.0  13.0
##  3  2017  10.0  29.0     11.0   11.0   18.0   9.00
##  4  2017  10.0  28.0     12.0   10.0   17.0  10.0
##  5  2017  10.0  27.0     12.0    9.00  13.0  13.0
##  6  2017  10.0  26.0     13.0   10.0   13.0  13.0
##  7  2017  10.0  25.0     13.0   14.0   17.0  13.0
##  8  2017  10.0  24.0     13.0   16.0   21.0  13.0
##  9  2017  10.0  23.0     13.0   13.0   20.0  13.0
## 10  2017  10.0  22.0     11.0   11.0   19.0  13.0
## # ... 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 12.0  25.0      10.0    0     6.00   6.00
##  2  2015 12.0  25.0       9.00   0    17.0    4.00
##  3  2016  5.00 31.0      18.0   11.0  26.0   19.0
##  4  2016  6.00  1.00     19.0   12.0  24.0   17.0
##  5  2016  4.00 10.0      10.0    4.00  5.00  16.0
##  6  2016  6.00  7.00     20.0   14.0  24.0   22.0
##  7  2016  5.00  6.00     17.0   12.0  11.0   18.0
##  8  2016  5.00  8.00     21.0   16.0  14.0   17.0
##  9  2016  5.00 10.0      19.0   14.0  14.0   18.0
## 10  2016  6.00  3.00     16.0   11.0  19.0   19.0
## # ... 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  11.0  1.00      8.00   8.00  16.0  13.0    5.00
##  2  2015  11.0  2.00     10.0   11.0   15.0  10.0    0
##  3  2015  11.0  3.00      9.00  11.0   16.0   9.00   0
##  4  2015  11.0  4.00     12.0   11.0   17.0  10.0   -2.00
##  5  2015  11.0  5.00     13.0   13.0   18.0  12.0   -1.00
##  6  2015  11.0  6.00     16.0   14.0   21.0  13.0   -3.00
##  7  2015  11.0  7.00     16.0   14.0   17.0  14.0   -2.00
##  8  2015  11.0  8.00     12.0   12.0   11.0  13.0    1.00
##  9  2015  11.0  9.00     13.0   12.0   11.0  11.0   -2.00
## 10  2015  11.0 10.0      14.0   14.0   12.0  11.0   -3.00
## # ... 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  11.0  1.00      8.00   8.00  16.0  13.0  T
##  2  2015  11.0  2.00     10.0   11.0   15.0  10.0  F
##  3  2015  11.0  3.00      9.00  11.0   16.0   9.00 F
##  4  2015  11.0  4.00     12.0   11.0   17.0  10.0  F
##  5  2015  11.0  5.00     13.0   13.0   18.0  12.0  F
##  6  2015  11.0  6.00     16.0   14.0   21.0  13.0  F
##  7  2015  11.0  7.00     16.0   14.0   17.0  14.0  F
##  8  2015  11.0  8.00     12.0   12.0   11.0  13.0  T
##  9  2015  11.0  9.00     13.0   12.0   11.0  11.0  F
## 10  2015  11.0 10.0      14.0   14.0   12.0  11.0  F
## # ... 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  11.0  1.00 warmer
##  2  2015  11.0  2.00 colder
##  3  2015  11.0  3.00 colder
##  4  2015  11.0  4.00 colder
##  5  2015  11.0  5.00 colder
##  6  2015  11.0  6.00 colder
##  7  2015  11.0  7.00 colder
##  8  2015  11.0  8.00 warmer
##  9  2015  11.0  9.00 colder
## 10  2015  11.0 10.0  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        14.3     14.4
```

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  1.00           3.00
##  2  2.00           4.32
##  3  3.00           6.92
##  4  4.00           8.43
##  5  5.00          14.5
##  6  6.00          17.3
##  7  7.00          18.0
##  8  8.00          17.7
##  9  9.00          16.0
## 10 10.0           11.7
## 11 11.0            7.65
## 12 12.0            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 11.0          11
##  2  2015 12.0          12
##  3  2016  1.00         23
##  4  2016  2.00         16
##  5  2016  3.00          5
##  6  2016  4.00         10
##  7  2016  5.00          8
##  8  2016  6.00          1
```

```
##  9  2016  7.00          1
## 10  2016  8.00          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 11.0        1.00         11.0        14.0
##  2  2015 12.0        0            10.0        14.0
##  3  2016  1.00       0             6.00       11.0
##  4  2016  2.00       1.00          4.00       12.0
##  5  2016  3.00       2.00          6.00       11.0
##  6  2016  4.00       4.00          8.00       11.0
##  7  2016  5.00       8.00         13.0        16.0
##  8  2016  6.00      11.0          16.0        19.0
##  9  2016  7.00      14.0          18.0        24.0
## 10  2016  8.00      14.0          18.0        24.0
## # ... 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  11.0  1.00      8.00  NA     10.0        NA          -2.00
##  2  2015  11.0  2.00     10.0    8.00   9.00        2.00        1.00
##  3  2015  11.0  3.00      9.00  10.0   12.0       - 1.00       -3.00
##  4  2015  11.0  4.00     12.0    9.00  13.0         3.00       -1.00
##  5  2015  11.0  5.00     13.0   12.0   16.0         1.00       -3.00
##  6  2015  11.0  6.00     16.0   13.0   16.0         3.00        0
##  7  2015  11.0  7.00     16.0   16.0   12.0         0           4.00
##  8  2015  11.0  8.00     12.0   16.0   13.0       - 4.00       -1.00
##  9  2015  11.0  9.00     13.0   12.0   14.0         1.00       -1.00
## 10  2015  11.0 10.0      14.0   13.0   13.0         1.00        1.00
## # ... 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  11.0  1.00      8.00   8.00  16.0  13.0
## 2  2015  11.0  2.00     10.0   11.0   15.0  10.0
## 3  2015  11.0  3.00      9.00  11.0   16.0   9.00
```

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 11.0   1.00      8.00   8.00 16.0   13.0
## 2  2015 11.0   2.00     10.0   11.0  15.0   10.0
## 3  2015 11.0   3.00      9.00  11.0  16.0    9.00
## 4  2016  1.00  1.00      4.00   3.00  3.00   2.00
## 5  2016  1.00  2.00      6.00  10.0   2.00   0
## 6  2016  1.00  3.00      7.00   8.00  4.00   3.00
## 7  2017  1.00  1.00      1.00   7.00  7.00   2.00
## 8  2017  1.00  2.00      3.00   2.00  3.00   1.00
## 9  2017  1.00  3.00      4.00   2.00  5.00   3.00
```

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  11.0  1.00 Amsterdam        8.00
```

```
##  2  2015  11.0  2.00 Amsterdam       10.0
##  3  2015  11.0  3.00 Amsterdam        9.00
##  4  2015  11.0  4.00 Amsterdam       12.0
##  5  2015  11.0  5.00 Amsterdam       13.0
##  6  2015  11.0  6.00 Amsterdam       16.0
##  7  2015  11.0  7.00 Amsterdam       16.0
##  8  2015  11.0  8.00 Amsterdam       12.0
##  9  2015  11.0  9.00 Amsterdam       13.0
## 10  2015  11.0 10.0  Amsterdam       14.0
## # ... 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`
## * <chr>     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Amsterdam  3.00  4.00  7.00  8.00  14.0  17.0  18.0  18.0  16.0  12.0
## 2 London     4.00  6.00  8.00  9.00  13.0  17.0  18.0  18.0  16.0  12.0
## 3 NYC        2.00  4.00  7.00 13.0   17.0  22.0  26.0  25.0  22.0  16.0
## 4 Venice     2.00  7.00 11.0  14.0   18.0  22.0  25.0  25.0  20.0  14.0
##    `11`  `12`
## * <dbl> <dbl>
## 1  8.00  7.00
## 2  9.00  8.00
## 3 11.0   7.00
## 4  9.00  5.00
```

_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      11.0   12.0  14.0   14.0
```

```
#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      11.0   12.0  14.0   14.0
```

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
##    <dbl> <dbl> <dbl> <dbl>               <dbl>           <dbl>
##  1  2015  11.0  1.00  16.0                8.00            8.00
##  2  2015  11.0  2.00  15.0                5.00            4.00
##  3  2015  11.0  3.00  16.0                7.00            5.00
##  4  2015  11.0  4.00  17.0                5.00            6.00
##  5  2015  11.0  5.00  18.0                5.00            5.00
##  6  2015  11.0  6.00  21.0                5.00            7.00
##  7  2015  11.0  7.00  17.0                1.00            3.00
##  8  2015  11.0  8.00  11.0                1.00            1.00
##  9  2015  11.0  9.00  11.0                2.00            1.00
## 10  2015  11.0 10.0   12.0                2.00            2.00
##    Venice_diff_NYC
##              <dbl>
##  1            3.00
##  2            5.00
##  3            7.00
##  4            7.00
##  5            6.00
##  6            8.00
##  7            3.00
##  8            2.00
##  9            0
## 10            1.00
## # ... 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 11.0  21.0       5.00   3.00   9.00   8.00
##  2  2015 11.0  22.0       3.00   1.00   9.00   8.00
##  3  2016  1.00 13.0       4.00   3.00 - 3.00   6.00
##  4  2016  1.00 16.0       2.00   1.00   8.00   4.00
##  5  2016  2.00  3.00      5.00   4.00  11.0    8.00
##  6  2016  2.00 11.0       4.00   3.00 - 4.00   7.00
##  7  2016  2.00 12.0       2.00   1.00 - 6.00   6.00
##  8  2016  2.00 23.0       4.00   3.00   3.00  11.0
##  9  2016  2.00 24.0       2.00   1.00   9.00  10.0
```

```
## 10  2016  2.00 25.0       2.00   1.00   9.00   8.00
## # ... 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  8.00       8.40
##  2  2016  7.00       8.10
##  3  2017  9.00       7.90
##  4  2016  4.00       7.50
##  5  2017  4.00       7.40
##  6  2017  7.00       7.30
##  7  2017  8.00       6.50
##  8  2016 11.0        6.40
##  9  2016  3.00       6.30
## 10  2016  6.00       6.00
## # ... 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  11.0  1.00 NYC              16.0
##  2  2015  11.0  2.00 NYC              15.0
##  3  2015  11.0  3.00 NYC              16.0
##  4  2015  11.0  4.00 NYC              17.0
##  5  2015  11.0  5.00 NYC              18.0
##  6  2015  11.0  6.00 NYC              21.0
##  7  2015  11.0  7.00 NYC              17.0
##  8  2015  11.0  8.00 Venice           13.0
##  9  2015  11.0  9.00 Amsterdam        13.0
## 10  2015  11.0 10.0  Amsterdam        14.0
## # ... with 721 more rows
```