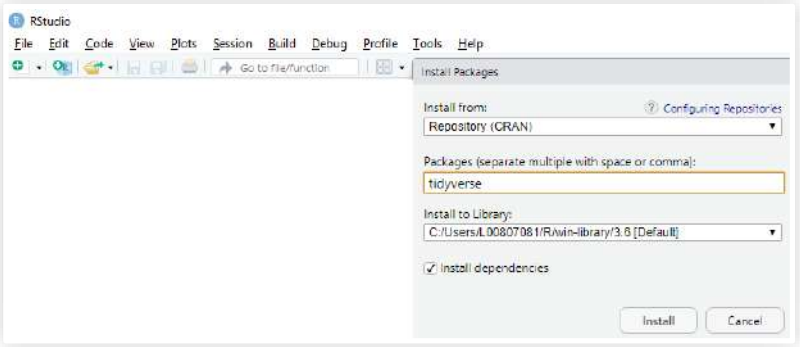
Data wrangling

En este laboratorio nos enfocaremos a la manipulación de datos, debido a que esto puede reducir hasta un 70% el tiempo invertido para un análisis de datos.

Para trabajar con este laboratorio es importante que tengas instalado el paquete de **Tidyverse**. Esto puedes hacerlo desde el menú de **Tools/Install Packages/tidyverse**:



Con esta instalación podrás contar con un combo de paquetes como **tidyr** y **dplyr**, los cuales son muy utilizados en data wrangling. Una vez instalados vamos a llamarlos de la siguiente manera:

```
> library(tidyverse)
-- Attaching packages --------------------------------------- tidyverse 1.3.0 --
<U+2713> tibble  2.1.3     <U+2713> purrr   0.3.3
<U+2713> tidyr   1.0.0     <U+2713> stringr 1.4.0
<U+2713> readr   1.3.1     <U+2713> forcats 0.4.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
> library(dbplyr)

Attaching package: 'dbplyr'

The following objects are masked from 'package:dplyr':

    ident, sql
```

Iniciarás trabajando con el **dataframe airquality**:

```
> head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

Se trata de un dataframe de 6 variables con 153 observaciones sobre la calidad del aire en Nueva York.

A continuación, utilizarás la función *select()* para cuando desees crear un subconjunto de columnas, por ejemplo, si quieres extraer las observaciones de la variable Ozone, deberás escribir la siguiente instrucción:

```
> select(airquality, Ozone)
    Ozone
1      41
2      36
3      12
4      18
5      NA
6      28
7      23
8      19
9       8
10     NA
```

Por su parte, la función *filter()* es muy útil en la manipulación de datos porque permite filtrar datos de filas con base en especificaciones dadas. A manera de ejemplo, filtraremos la información del dataframe airquality para aquellas filas donde el nivel de concentración de ozone superó las 100 partes por billón. La instrucción es la siguiente:

```
> filter(airquality, Ozone > 100)
  Ozone Solar.R Wind Temp Month Day
1   115     223  5.7   79     5  30
2   135     269  4.1   84     7   1
3   108     223  8.0   85     7  25
4   122     255  4.0   89     8   7
5   110     207  8.0   90     8   9
6   168     238  3.4   81     8  25
7   118     225  2.3   94     8  29
```

También podrías poner más condicionantes, por ejemplo, aquellos días en donde el ozono superó los 50 ppb, pero que también se tuvo una temperatura mayor a 80° Fahrenheit:

```
> filter(airquality, Ozone > 50 & Temp > 80)
   Ozone Solar.R Wind Temp Month Day
1     71     291 13.8   90     6   9
2    135     269  4.1   84     7   1
3     64     175  4.6   83     7   5
4     77     276  5.1   88     7   7
5     97     267  6.3   92     7   8
6     97     272  5.7   92     7   9
7     85     175  7.4   89     7  10
8     61     285  6.3   84     7  18
9     79     187  5.1   87     7  19
10    63     220 11.5   85     7  20
11    80     294  8.6   86     7  24
12   108     223  8.0   85     7  25
13    52      82 12.0   86     7  27
14    82     213  7.4   88     7  28
15    64     253  7.4   83     7  30
16    59     254  9.2   81     7  31
17    78      NA  6.9   86     8   4
18    66      NA  4.6   87     8   6
19   122     255  4.0   89     8   7
20    89     229 10.3   90     8   8
21   110     207  8.0   90     8   9
22   168     238  3.4   81     8  25
23    73     215  8.0   86     8  26
24    76     203  9.7   97     8  28
25   118     225  2.3   94     8  29
26    84     237  6.3   96     8  30
27    85     188  6.3   94     8  31
28    96     167  6.9   91     9   1
29    78     197  5.1   92     9   2
30    73     183  2.8   93     9   3
31    91     189  4.6   93     9   4
```

Una instrucción que es de mucha utilidad en la manipulación de datos es *Pipes* (%>%), que sirve como una especie de retención en memoria de la última salida ejecutada por una instrucción para utilizarla en una nueva, es decir, (retomando el ejemplo que hemos estado trabajando), cuando se desea filtrar la información para que únicamente obtengas las mediciones donde el ozono estuvo por debajo de 100 en el mes 7 y que te muestre solo la columna del ozono se realiza de la siguiente forma:

```
> airquality %>%
+ filter(Ozone < 100 & Month == 7) %>%
+ select(Ozone)
   Ozone
1     49
2     32
3     64
4     40
5     77
6     97
7     97
8     85
9     10
10    27
11     7
12    48
13    35
14    61
15    79
16    63
17    16
18    80
19    20
20    52
21    82
22    50
23    64
24    59
```

Para corroborar, agrega la opción para que muestre también el mes:

```
> airquality %>%
+ filter(Ozone < 100 & Month == 7) %>%
+ select(Ozone, Month)
   Ozone Month
1     49     7
2     32     7
3     64     7
4     40     7
5     77     7
6     97     7
7     97     7
8     85     7
9     10     7
10    27     7
11     7     7
12    48     7
13    35     7
14    61     7
15    79     7
16    63     7
17    16     7
18    80     7
19    20     7
20    52     7
21    82     7
22    50     7
23    64     7
24    59     7
```

Por su parte, la función *mutate()* se utiliza para crear o transformar variables, es decir, permite crear variables nuevas o transformarlas, preservando las originales, por ejemplo, (en el dataframe que hemos estado trabajando) imagina que deseas agregar una columna que tenga la temperatura en grados centígrados. Para esto, tendrías que ejecutar las siguientes instrucciones:

```
> mutate(airquality, TempC = (Temp-32)/1.8)
    Ozone Solar.R Wind Temp Month Day    TempC
1      41     190  7.4   67     5   1 19.44444
2      36     118  8.0   72     5   2 22.22222
3      12     149 12.6   74     5   3 23.33333
4      18     313 11.5   62     5   4 16.66667
5      NA      NA 14.3   56     5   5 13.33333
6      28      NA 14.9   66     5   6 18.88889
7      23     299  8.6   65     5   7 18.33333
8      19      99 13.8   59     5   8 15.00000
9       8      19 20.1   61     5   9 16.11111
10     NA     194  8.6   69     5  10 20.55556
11      7      NA  6.9   74     5  11 23.33333
12     16     256  9.7   69     5  12 20.55556
13     11     290  9.2   66     5  13 18.88889
14     14     274 10.9   68     5  14 20.00000
15     18      65 13.2   58     5  15 14.44444
16     14     334 11.5   64     5  16 17.77778
17     34     307 12.0   66     5  17 18.88889
18      6      78 18.4   57     5  18 13.88889
19     30     322 11.5   68     5  19 20.00000
20     11      44  9.7   62     5  20 16.66667
```

Para redondear los datos en grados centígrados se puede usar la función *round()*:

```
> mutate(airquality, TempC = round((Temp-32)/1.8))
    Ozone Solar.R Wind Temp Month Day TempC
1      41     190  7.4   67     5   1    19
2      36     118  8.0   72     5   2    22
3      12     149 12.6   74     5   3    23
4      18     313 11.5   62     5   4    17
5      NA      NA 14.3   56     5   5    13
6      28      NA 14.9   66     5   6    19
7      23     299  8.6   65     5   7    18
8      19      99 13.8   59     5   8    15
9       8      19 20.1   61     5   9    16
10     NA     194  8.6   69     5  10    21
```

Ahora es tu turno de practicar con el siguiente ejercicio:

Práctica:

Utilizando el dataframe de Airquality (utilizado en los ejemplos anteriores) realiza lo siguiente:

a) Filtra la información del dataframe considerando aquellos días donde el nivel de concentración de Ozono fue inferior a 50 y en donde adicionalmente la temperatura haya estado por debajo de los 70°F.

b) Modifica el dataframe de Airquality de modo que los valores de la variable Wind sean redondeados para que no contengan decimales.

c) Agrega una columna al dataframe donde se incluya la temperatura en Kelvin.

Otra función muy útil en el manejo de datos es *group_by()*, la cual se usa para agrupar los datos en conjuntos comunes, por ejemplo, en este caso queremos agrupar los datos de airquality por mes, por lo cual deberás ejecutar la siguiente instrucción:

```
> by_month <- airquality %>% group_by(Month)

> View(by_month)
```

Estas instrucciones nos arrojan el siguiente resultado:

	Ozone	Solar.R	Wind	Temp	Month	Day
1	41	190	7.4	67	5	1
2	36	118	8.0	72	5	2
3	12	149	12.6	74	5	3
4	18	313	11.5	62	5	4
5	NA	NA	14.3	56	5	5
6	28	NA	14.9	66	5	6
7	23	299	8.6	65	5	7
8	19	99	13.8	59	5	8
9	8	19	20.1	61	5	9
10	NA	194	8.6	69	5	10

Como podrás observar, los datos están agrupados ahora por mes, por lo tanto, podrás realizar operaciones con esos datos agrupados, por ejemplo, si ahora lo que deseas es saber los promedios de temperaturas por cada mes, podrás hacerlo fácilmente con las instrucciones que se muestran a continuación:

```
> by_month %>% summarise(Promedio_temp = mean(Temp))
# A tibble: 5 x 2
  Month Promedio_temp
  <int>         <dbl>
1     5          65.5
2     6          79.1
3     7          83.9
4     8          84.0
5     9          76.9
```

Si lo que deseas es saber las temperaturas máximas y mínimas de cada mes:

```
> by_month %>% summarise(max_temp = max(Temp), min_temp = min(Temp))
# A tibble: 5 x 3
  Month max_temp min_temp
  <int>    <int>    <int>
1     5       81       56
2     6       93       65
3     7       92       73
4     8       97       72
5     9       93       63
```

Cuando tengas columnas de datos donde no cuentes con observaciones, los cálculos que deseas realizar se pueden afectar, por ejemplo, si desearas encontrar el promedio de concentración de ozono por mes, tendrías el siguiente resultado:

```
> by_month %>% summarise(Promedio_Ozone = mean(Ozone))
# A tibble: 5 x 2
  Month Promedio_Ozone
  <int>          <dbl>
1     5             NA
2     6             NA
3     7             NA
4     8             NA
5     9             NA
```

Para eso podrías hacer uso de *filter* en combinación con *group_by*, por ejemplo, si deseas obtener los promedios de concentraciones por cada mes, excluyendo las observaciones con NA, podrías hacerlo como se muestra a continuación:

```
> by_month %>% filter(!is.na(Ozone)) %>% summarise(Promedio_ppb = mean(Ozone))
# A tibble: 5 x 2
  Month Promedio_ppb
  <int>        <dbl>
1     5         23.6
2     6         29.4
3     7         59.1
4     8         60.0
5     9         31.4
```

Se puede usar la función *tally()* cuando lo que deseas es contar elementos de un conjunto de datos. Imagina que deseas contar cuántos elementos se encuentran en cada grupo por mes, lo cual podrías hacerlo de la siguiente manera:

```
> airquality %>% group_by(Month) %>% tally()
# A tibble: 5 x 2
  Month     n
  <int> <int>
1     5    31
2     6    30
3     7    31
4     8    31
5     9    30
```

La suma de los elementos de cada grupo debe dar el total de las observaciones:

```
> airquality %>% tally()
    n
1 153
```

Puedes hacer uso de la función *pivot_wider()* para convertir filas en columnas, obteniendo así una mejor observación de los datos.

En este ejemplo utilizarás el grupo de datos fish_encounters, donde se tiene un grupo de observaciones sobre ciertos peces en estaciones, así como información (número 1) en caso de que el pez haya sido visto por la estación correspondiente:

```
> fish_encounters
# A tibble: 114 x 3
   fish  station  seen
   <fct> <fct>   <int>
 1 4842  Release     1
 2 4842  I80_1       1
 3 4842  Lisbon      1
 4 4842  Rstr        1
 5 4842  Base_TD     1
 6 4842  BCE         1
 7 4842  BCW         1
 8 4842  BCE2        1
 9 4842  BCW2        1
10 4842  MAE         1
# … with 104 more rows
```

Como podrás observar, dentro de las filas se encuentran los datos de las estaciones, las cuales pueden convertirse en columnas para visualizar mejor los datos.
Para realizar esto harás uso de la función *pivot_wider()*:

```
> fish_encounters %>%
+     pivot_wider(names_from = station, values_from = seen)
# A tibble: 19 x 12
   fish  Release I80_1 Lisbon  Rstr Base_TD   BCE   BCW  BCE2  BCW2   MAE   MAW
   <fct>   <int> <int>  <int> <int>   <int> <int> <int> <int> <int> <int> <int>
 1 4842        1     1      1     1       1     1     1     1     1     1     1
 2 4843        1     1      1     1       1     1     1     1     1     1     1
 3 4844        1     1      1     1       1     1     1     1     1     1     1
 4 4845        1     1      1     1       1    NA    NA    NA    NA    NA    NA
 5 4847        1     1      1    NA      NA    NA    NA    NA    NA    NA    NA
 6 4848        1     1      1     1      NA    NA    NA    NA    NA    NA    NA
 7 4849        1     1     NA    NA      NA    NA    NA    NA    NA    NA    NA
 8 4850        1     1     NA     1       1     1     1    NA    NA    NA    NA
 9 4851        1     1     NA    NA      NA    NA    NA    NA    NA    NA    NA
10 4854        1     1     NA    NA      NA    NA    NA    NA    NA    NA    NA
11 4855        1     1      1     1       1    NA    NA    NA    NA    NA    NA
12 4857        1     1      1     1       1     1     1     1     1    NA    NA
13 4858        1     1      1     1       1     1     1     1     1     1     1
14 4859        1     1      1     1       1    NA    NA    NA    NA    NA    NA
15 4861        1     1      1     1       1     1     1     1     1     1     1
16 4862        1     1      1     1       1     1     1     1     1    NA    NA
17 4863        1     1     NA    NA      NA    NA    NA    NA    NA    NA    NA
18 4864        1     1     NA    NA      NA    NA    NA    NA    NA    NA    NA
19 4865        1     1      1    NA      NA    NA    NA    NA    NA    NA    NA
```

Con ese ajuste la información se observa mejor, sin embargo, donde no existen observaciones se encuentra un NA, por lo tanto, vas a sustituir NA por 0 para trabajar mejor los datos:

```
> fish_encounters %>%
+     pivot_wider(
+         names_from = station,
+         values_from = seen,
+         values_fill = list(seen = 0)
+     )
# A tibble: 19 x 12
   fish  Release I80_1 Lisbon  Rstr Base_TD   BCE   BCW  BCE2  BCW2   MAE   MAW
   <fct>   <int> <int>  <int> <int>   <int> <int> <int> <int> <int> <int> <int>
 1 4842        1     1      1     1       1     1     1     1     1     1     1
 2 4843        1     1      1     1       1     1     1     1     1     1     1
 3 4844        1     1      1     1       1     1     1     1     1     1     1
 4 4845        1     1      1     1       1     0     0     0     0     0     0
 5 4847        1     1      1     0       0     0     0     0     0     0     0
 6 4848        1     1      1     1       0     0     0     0     0     0     0
 7 4849        1     1      0     0       0     0     0     0     0     0     0
 8 4850        1     1      0     1       1     1     1     0     0     0     0
 9 4851        1     1      0     0       0     0     0     0     0     0     0
10 4854        1     1      0     0       0     0     0     0     0     0     0
11 4855        1     1      1     1       1     0     0     0     0     0     0
12 4857        1     1      1     1       1     1     1     1     1     0     0
13 4858        1     1      1     1       1     1     1     1     1     1     1
14 4859        1     1      1     1       1     0     0     0     0     0     0
15 4861        1     1      1     1       1     1     1     1     1     1     1
16 4862        1     1      1     1       1     1     1     1     1     0     0
17 4863        1     1      0     0       0     0     0     0     0     0     0
18 4864        1     1      0     0       0     0     0     0     0     0     0
19 4865        1     1      1     0       0     0     0     0     0     0     0
```

Ahora es tu turno para practicar:

Ejercicio:

Utiliza el dataset mtcars para realizar una agrupación de datos con base en los siguientes criterios:

a) Aquellos con 8 cilindros y transmisión automática.
b) Que presente un resumen de la información contando todos los vehículos observados por número de carburadores.
c) Calcular el promedio de HP de los vehículos por número de cilindros.