DSCI 325: Handout 21 – Introduction to the dplyr package in R

Spring 2016

In this handout, we will introduce the `dplyr` package which can be used to manipulate data in R. Most of the examples discussed below were taken from the following R documentation:

http://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html

As stated in this documentation, the `dplyr` package provides "simple functions that correspond to the most common data manipulation verbs, so that you can easily translate your thoughts into code."

Start by installing both this package and the data set that will be used for illustration purposes.

```
> install.packages("dplyr")
> install.packages(c("nycflights13", "Lahman"))

> library(dplyr)
> library(nycflights13)
```

Filter rows with `filter()`

With the `dplyr` package, the `filter()` function allows you to select a subset of the rows of a data frame.

```
> filter(flights, month == 1, day ==1)
```

```
   year month day dep_time dep_delay arr_time arr_delay carrier tailnum flight origin dest air_time distance hour
1  2013     1   1      517         2      830        11      UA  N14228   1545    EWR  IAH      227     1400    5
2  2013     1   1      533         4      850        20      UA  N24211   1714    LGA  IAH      227     1416    5
3  2013     1   1      542         2      923        33      AA  N619AA   1141    JFK  MIA      160     1089    5
4  2013     1   1      544        -1     1004       -18      B6  N804JB    725    JFK  BQN      183     1576    5
5  2013     1   1      554        -6      812       -25      DL  N668DN    461    LGA  ATL      116      762    5
6  2013     1   1      554        -4      740        12      UA  N39463   1696    EWR  ORD      150      719    5
7  2013     1   1      555        -5      913        19      B6  N516JB    507    EWR  FLL      158     1065    5
8  2013     1   1      557        -3      709       -14      EV  N829AS   5708    LGA  IAD       53      229    5
9  2013     1   1      557        -3      838        -8      B6  N593JB     79    JFK  MCO      140      944    5
10 2013     1   1      558        -2      753         8      AA  N3ALAA    301    LGA  ORD      138      733    5
..  ...   ... ...      ...       ...      ...       ...     ...     ...    ...    ...  ...      ...      ...  ...
Variables not shown: minute (dbl)
```

Note that this is equivalent to what we would have obtained with the following command:

```
> attach(flights)
> flights[month == 1 & day == 1, ]
```

With the `filter()` function you can give any number of filtering conditions which are joined together with "&" or other Boolean operators. For example, consider the following commands using this function.

```
> filter(flights, month == 1 & day == 1)
> filter(flights, month == 1 & day == 1 & carrier == "UA")
> filter(flights, origin == "EWR" | origin == "LGA")
> filter(flights, (origin == "EWR" | origin == "LGA") & dest == "IAH")
```

Arrange rows with `arrange()`

This function from the `dplyr` package can be used to reorder the rows of a data set.

```
> arrange(flights, year, month, day)
```

```
   year month day dep_time dep_delay arr_time arr_delay carrier tailnum flight origin dest air_time distance hour
1  2013     1   1      517         2      830        11      UA  N14228   1545    EWR  IAH      227     1400    5
2  2013     1   1      533         4      850        20      UA  N24211   1714    LGA  IAH      227     1416    5
3  2013     1   1      542         2      923        33      AA  N619AA   1141    JFK  MIA      160     1089    5
4  2013     1   1      544        -1     1004       -18      B6  N804JB    725    JFK  BQN      183     1576    5
5  2013     1   1      554        -6      812       -25      DL  N668DN    461    LGA  ATL      116      762    5
6  2013     1   1      554        -4      740        12      UA  N39463   1696    EWR  ORD      150      719    5
7  2013     1   1      555        -5      913        19      B6  N516JB    507    EWR  FLL      158     1065    5
8  2013     1   1      557        -3      709       -14      EV  N829AS   5708    LGA  IAD       53      229    5
9  2013     1   1      557        -3      838        -8      B6  N593JB     79    JFK  MCO      140      944    5
10 2013     1   1      558        -2      753         8      AA  N3ALAA    301    LGA  ORD      138      733    5
..  ...   ... ...      ...       ...      ...       ...     ...     ...    ...    ...  ...      ...      ...  ...
Variables not shown: minute (dbl)
```

```
> arrange(flights, distance, air_time)
```

```
   year month day dep_time dep_delay arr_time arr_delay carrier tailnum flight origin dest air_time distance hour
1  2013     7  27       NA        NA       NA        NA      US            1632    EWR  LGA       NA       17   NA
2  2013     2   3     2153        24     2247        23      EV  N13913   4619    EWR  PHL       21       80   21
3  2013     2  12     2123        -7     2211       -14      EV  N12921   4619    EWR  PHL       21       80   21
4  2013     1   6     2125        -4     2224         0      EV  N22909   4619    EWR  PHL       22       80   21
5  2013     1  23     2128        -1     2221        -3      EV  N12135   4619    EWR  PHL       23       80   21
6  2013     2  10     2127        -2     2209       -15      EV  N41104   4619    EWR  PHL       23       80   21
7  2013     2   1     2128        -1     2216        -8      EV  N13969   4619    EWR  PHL       24       80   21
8  2013     3  30     1942        -8     2026       -18      EV  N12569   4457    EWR  PHL       24       80   19
9  2013     1   7     2124        -5     2212       -12      EV  N33182   4619    EWR  PHL       25       80   21
10 2013     1  14     2128        -1     2215        -9      EV  N14953   4619    EWR  PHL       25       80   21
..  ...   ... ...      ...       ...      ...       ...     ...     ...    ...    ...  ...      ...      ...  ...
Variables not shown: minute (dbl)
```

Note that this is equivalent to what we saw earlier using the `order()` function:

```
> flights[order(distance, air_time),]
```

To order a column in descending order, use `desc()`:

```
> arrange(flights, desc(distance), desc(air_time))
```

```
   year month day dep_time dep_delay arr_time arr_delay carrier tailnum flight origin dest air_time distance hour
1  2013     2   6      853        -7     1542         2      HA  N389HA     51    JFK  HNL      691     4983    8
2  2013     3  15     1001         1     1551        21      HA  N388HA     51    JFK  HNL      686     4983   10
3  2013     3  17     1006         6     1607        37      HA  N380HA     51    JFK  HNL      686     4983   10
4  2013     3  16     1001         1     1544        14      HA  N384HA     51    JFK  HNL      683     4983   10
5  2013     2   5      900         0     1555        15      HA  N386HA     51    JFK  HNL      679     4983    9
6  2013     3  14      958        -2     1542        12      HA  N380HA     51    JFK  HNL      676     4983    9
7  2013    11  20     1006         6     1639        44      HA  N386HA     51    JFK  HNL      675     4983   10
8  2013     4   3      957        -3     1535        25      HA  N383HA     51    JFK  HNL      671     4983    9
9  2013    11  11      957        -3     1627        32      HA  N383HA     51    JFK  HNL      669     4983    9
10 2013    11  10      957        -3     1625        30      HA  N393HA     51    JFK  HNL      667     4983    9
..  ...   ... ...      ...       ...      ...       ...     ...     ...    ...    ...  ...      ...      ...  ...
Variables not shown: minute (dbl)
```

Select columns with `select()`

If you're working with a large data set and only a few variables are of actual interest to you, you can select that subset of variables easily with `dplyr`. For example, consider the following:

```
> select(flights, year, month, day)
```

```
   year month day
1  2013     1   1
2  2013     1   1
3  2013     1   1
4  2013     1   1
5  2013     1   1
6  2013     1   1
7  2013     1   1
8  2013     1   1
9  2013     1   1
10 2013     1   1
..  ...   ... ...
```

```
> select(flights, year:day)
```

```
  year month day
1  2013     1   1
2  2013     1   1
3  2013     1   1
4  2013     1   1
5  2013     1   1
6  2013     1   1
7  2013     1   1
8  2013     1   1
9  2013     1   1
10 2013     1   1
..  ...   ... ...
```

```
> select(flights, -(year:day))
```

```
   dep_time dep_delay arr_time arr_delay carrier tailnum flight origin dest air_time distance hour mi
nute
1       517         2      830        11      UA  N14228   1545    EWR  IAH      227     1400    5
17
2       533         4      850        20      UA  N24211   1714    LGA  IAH      227     1416    5
33
3       542         2      923        33      AA  N619AA   1141    JFK  MIA      160     1089    5
42
4       544        -1     1004       -18      B6  N804JB    725    JFK  BQN      183     1576    5
44
5       554        -6      812       -25      DL  N668DN    461    LGA  ATL      116      762    5
54
6       554        -4      740        12      UA  N39463   1696    EWR  ORD      150      719    5
54
7       555        -5      913        19      B6  N516JB    507    EWR  FLL      158     1065    5
55
8       557        -3      709       -14      EV  N829AS   5708    LGA  IAD       53      229    5
57
9       557        -3      838        -8      B6  N593JB     79    JFK  MCO      140      944    5
57
10      558        -2      753         8      AA  N3ALAA    301    LGA  ORD      138      733    5
58
..      ...       ...      ...       ...     ...     ...    ...    ...  ...      ...      ...  ...
...
```

You can also use various "helper functions" within `select()`, as shown below.

- `starts_with()`
- `ends_with()`
- `matches()`
- `contains()`

```
> select(flights, starts_with("arr"))
```

```
  arr_time arr_delay
1      830        11
2      850        20
3      923        33
4     1004       -18
5      812       -25
6      740        12
7      913        19
8      709       -14
9      838        -8
10     753         8
..     ...       ...
```

```
> select(flights, ends_with("delay"))
```

```
  dep_delay arr_delay
1         2        11
2         4        20
3         2        33
4        -1       -18
5        -6       -25
6        -4        12
7        -5        19
8        -3       -14
9        -3        -8
10       -2         8
..      ...       ...
```

```
> select(flights, contains("_"))
```

```
   dep_time dep_delay arr_time arr_delay air_time
1       517         2      830        11      227
2       533         4      850        20      227
3       542         2      923        33      160
4       544        -1     1004       -18      183
5       554        -6      812       -25      116
6       554        -4      740        12      150
7       555        -5      913        19      158
8       557        -3      709       -14       53
9       557        -3      838        -8      140
10      558        -2      753         8      138
..      ...       ...      ...       ...      ...
```

Also, a common use of the `select()` function is to determine how many unique (or distinct) values a variable (or a set of variables) takes on.

```
> distinct(select(flights, carrier))
```

```
   carrier
1       UA
2       AA
3       B6
4       DL
5       EV
6       MQ
7       US
8       WN
9       VX
10      FL
11      AS
12      9E
13      F9
14      HA
15      YV
16      OO
```

```
> distinct(select(flights, carrier, dest))
```

```
   carrier dest
1       UA  IAH
2       AA  MIA
3       B6  BQN
4       DL  ATL
5       UA  ORD
6       B6  FLL
7       EV  IAD
8       B6  MCO
9       AA  ORD
10      B6  PBI
..     ...  ...
```

Add new columns with `mutate()`

In addition to selecting from existing columns, you can add new columns that are functions of existing columns.

```
> mutate(flights, gain = arr_delay - dep_delay, speed = distance / air_time*60)
```

```
   year month day dep_time dep_delay arr_time arr_delay carrier tailnum flight origin dest air_time distance hour
1  2013     1   1      517         2      830        11      UA  N14228   1545    EWR  IAH      227     1400    5
2  2013     1   1      533         4      850        20      UA  N24211   1714    LGA  IAH      227     1416    5
3  2013     1   1      542         2      923        33      AA  N619AA   1141    JFK  MIA      160     1089    5
4  2013     1   1      544        -1     1004       -18      B6  N804JB    725    JFK  BQN      183     1576    5
5  2013     1   1      554        -6      812       -25      DL  N668DN    461    LGA  ATL      116      762    5
6  2013     1   1      554        -4      740        12      UA  N39463   1696    EWR  ORD      150      719    5
7  2013     1   1      555        -5      913        19      B6  N516JB    507    EWR  FLL      158     1065    5
8  2013     1   1      557        -3      709       -14      EV  N829AS   5708    LGA  IAD       53      229    5
9  2013     1   1      557        -3      838        -8      B6  N593JB     79    JFK  MCO      140      944    5
10 2013     1   1      558        -2      753         8      AA  N3ALAA    301    LGA  ORD      138      733    5
..  ...   ... ...      ...       ...      ...       ...     ...     ...    ...    ...  ...      ...      ...  ...
Variables not shown: minute (dbl), gain (dbl), speed (dbl)
```

Note that if you wanted to save this modified version to a new data set, you could use the following command:

```
> flights2 = mutate(flights, gain = arr_delay - dep_delay, speed = distance / air_time*60)
```

The `transmute()` function also allows you to create new columns that are functions of existing columns. The difference is that this saves only the new columns that you create.

```
> transmute(flights, gain = arr_delay - dep_delay, speed = distance / air_time*60)
```

```
Source: local data frame [336,776 x 2]

   gain    speed
1     9 370.0441
2    16 374.2731
3    31 408.3750
4   -17 516.7213
5   -19 394.1379
6    16 287.6000
7    24 404.4304
8   -11 259.2453
9    -5 404.5714
10   10 318.6957
..  ...      ...
```

Summarize values with `summarize()`

This lets you create summaries that collapse a data frame to a single row. For example, consider the following:

```
> summarize(flights, mean_delay = mean(dep_delay, na.rm = TRUE))
```

```
  mean_delay
1   12.63907
```

This function is more useful when used in conjunction with others (which you will see later on). Finally, note that you can also call this function as follows:

```
> summarise(flights, mean_delay = mean(dep_delay, na.rm = TRUE))
```

Randomly sample rows with `sample_n()` and `sample_frac()`

You can take a random sample of a fixed number of rows (here, 10) as follows:

```
> sample_n(flights,10)
```

```
Source: local data frame [10 x 16]

   year month day dep_time dep_delay arr_time arr_delay carrier tailnum flight origin dest air_time distance hour
1  2013     4  21     1438        -2     1601       -23      EV  N14543   4438    EWR  STL      130      872   14
2  2013     4   2     1921       112     2134       117      EV  N12957   4104    EWR  CVG      111      569   19
3  2013     5  15     1833         4     2025       -14      DL  N320US   2019    LGA  MSP      151     1020   18
4  2013    10   3      815        -7     1006        -8      EV  N16954   4691    EWR  DAY       80      533    8
5  2013     2  27     1845        15     2102       -19      B6  N708JB    711    JFK  LAS      285     2248   18
6  2013     8   9     1045        55     1317        57      VX  N361VA    251    JFK  LAS      299     2248   10
7  2013     2  26      829        -6      944       -28      9E  N937XJ   3317    JFK  BUF       55      301    8
8  2013     2  19     1454        -4     1803       -13      UA  N844UA    374    EWR  AUS      230     1504   14
9  2013     1  15     1703        -2     1946       -22      B6  N805JB    143    JFK  PBI      144     1028   17
10 2013     6  12     1720         4     1934        -8      UA  N481UA    485    EWR  DEN      227     1605   17
Variables not shown: minute (dbl)
```

Similarly, you can take a random sample of a fixed percentage of your rows (here, 1%):

```
> sample_frac(flights,.01)
```

```
Source: local data frame [3,368 x 16]

   year month day dep_time dep_delay arr_time arr_delay carrier tailnum flight origin dest air_time distance hour
1  2013    11  26      613         0      830        -4      EV  N14907   4393    EWR  IND      113      645    6
2  2013    11   8      713        -7      832       -23      WN  N288WN   1975    EWR  BNA      118      748    7
3  2013     5   2     1550        30     1704        25      EV  N15572   4141    EWR  MKE      107      725   15
4  2013     4   5     1839        -8     2156       -23      UA  N512UA    346    JFK  SFO      348     2586   18
5  2013     7  18     1729        30     1921        58      EV  N14153   4091    EWR  IAD       46      212   17
6  2013    10  18     1056        -4     1210       -13      UA  N822UA    798    EWR  ORD      117      719   10
7  2013     5  23     1658       313     1837       315      EV  N13161   4376    EWR  RDU       76      416   16
8  2013     9  27     1924        -1     2150       -58      DL  N316US    498    JFK  SAT      184     1587   19
9  2013     2  27     1847        48     2111        29      DL  N627DL   1047    LGA  ATL      118      762   18
10 2013     6  25      940        10     1218       -12      WN  N242WN   1983    EWR  AUS      190     1504    9
..  ...   ... ...      ...       ...      ...       ...     ...     ...    ...    ...  ...      ...      ...  ...
Variables not shown: minute (dbl)
```

Commonalities of functions in the `dplyr` package

Note that all of these functions are similar in the following ways:

- The first argument is a data frame
- Subsequent arguments tell R what to do with that data frame
- The result is a new data frame

As stated in the aforementioned R documentation, these five functions together "provide the basis of a language of data manipulation." At the most basic level, we alter data sets in the following ways:

- Reorder rows (`arrange()`)
- Select observations (rows) of interest (`filter()`)
- Select variables (columns) of interest (`select()`)
- Add new variables that are functions of existing variables (`mutate()`)
- Collapse many values to a summary (`summarize()`)

Grouped operations with `group_by()`

Finally, note that you can also use all of the above functions to process a data set "by group."

```
> by_carrier = group_by(flights, carrier)
```

```
> summarize(by_carrier, n())
```

```
carrier   n()
1       9E 18460
2       AA 32729
3       AS   714
4       B6 54635
5       DL 48110
6       EV 54173
7       F9   685
8       FL  3260
9       HA   342
10      MQ 26397
11      OO    32
12      UA 58665
13      US 20536
14      VX  5162
15      WN 12275
16      YV   601
```

```
> summarize(by_carrier, mean(dep_delay, na.rm=TRUE), mean(arr_delay, na.rm=TRUE))
```

```
carrier mean(dep_delay, na.rm = TRUE) mean(arr_delay, na.rm = TRUE)
1       9E                   16.725769                    7.3796692
2       AA                    8.586016                    0.3642909
3       AS                    5.804775                   -9.9308886
4       B6                   13.022522                    9.4579733
5       DL                    9.264505                    1.6443409
6       EV                   19.955390                   15.7964311
7       F9                   20.215543                   21.9207048
8       FL                   18.726075                   20.1159055
9       HA                    4.900585                   -6.9152047
10      MQ                   10.552041                   10.7747334
11      OO                   12.586207                   11.9310345
12      UA                   12.106073                    3.5580111
13      US                    3.782418                    2.1295951
14      VX                   12.869421                    1.7644644
15      WN                   17.711744                    9.6491199
16      YV                   18.996330                   15.5569853
```

```
> summarize(by_carrier, sd(dep_delay, na.rm=TRUE), sd(arr_delay, na.rm=TRUE))
```

```
carrier sd(dep_delay, na.rm = TRUE) sd(arr_delay, na.rm = TRUE)
1       9E                  45.90604                   50.08678
2       AA                  37.35486                   42.51618
3       AS                  31.36303                   36.48263
4       B6                  38.50337                   42.84230
5       DL                  39.73505                   44.40229
6       EV                  46.55235                   49.86147
7       F9                  58.36265                   61.64600
8       FL                  52.66160                   54.08767
9       HA                  74.10990                   75.12942
10      MQ                  39.18457                   43.17431
11      OO                  43.06599                   48.58493
12      UA                  35.71660                   40.98434
13      US                  28.05633                   33.06695
14      VX                  44.81510                   49.96645
15      WN                  43.34435                   46.87770
16      YV                  49.17227                   52.92223
```

```
> summarize(by_carrier, min(dep_delay, na.rm=TRUE), min(arr_delay, na.rm=TRUE))
```

```
   carrier min(dep_delay, na.rm = TRUE) min(arr_delay, na.rm = TRUE)
1       9E                          -24                          -68
2       AA                          -24                          -75
3       AS                          -21                          -74
4       B6                          -43                          -71
5       DL                          -33                          -71
6       EV                          -32                          -62
7       F9                          -27                          -47
8       FL                          -22                          -44
9       HA                          -16                          -70
10      MQ                          -26                          -53
11      OO                          -14                          -26
12      UA                          -20                          -75
13      US                          -19                          -70
14      VX                          -20                          -86
15      WN                          -13                          -58
16      YV                          -16                          -46
```

```
> summarize(by_carrier, max(dep_delay, na.rm=TRUE), max(arr_delay, na.rm=TRUE))
```

```
carrier max(dep_delay, na.rm = TRUE) max(arr_delay, na.rm = TRUE)
1       9E                          747                          744
2       AA                         1014                         1007
3       AS                          225                          198
4       B6                          502                          497
5       DL                          960                          931
6       EV                          548                          577
7       F9                          853                          834
8       FL                          602                          572
9       HA                         1301                         1272
10      MQ                         1137                         1127
11      OO                          154                          157
12      UA                          483                          455
13      US                          500                          492
14      VX                          653                          676
15      WN                          471                          453
16      YV                          387                          381
```

Chaining operations

Suppose you want to do many operations at once:

```
> a1 = group_by(flights, year, month, day)
> a2 = select(a1, arr_delay, dep_delay)
> a3 = summarize (a2, arr_delay_avg = mean(arr_delay, na.rm = TRUE),
       dep_delay_avg = mean(dep_delay, na.rm = TRUE))
> a4 = filter(a3, arr_delay_avg > 30 | dep_delay_avg > 30)
> a4
```

```
   year month day arr_delay_avg dep_delay_avg
1  2013     1  16      34.24736      24.61287
2  2013     1  31      32.60285      28.65836
3  2013     2  11      36.29009      39.07360
4  2013     2  27      31.25249      37.76327
5  2013     3   8      85.86216      83.53692
6  2013     3  18      41.29189      30.11796
7  2013     4  10      38.41231      33.02368
8  2013     4  12      36.04814      34.83843
9  2013     4  18      36.02848      34.91536
10 2013     4  19      47.91170      46.12783
..  ...   ... ...           ...           ...
```

Note that you could have alternatively used the chain operator (%>%) to rewrite these multiple operations. The %>% operator uses the output from the left-hand side as the first input to the function on the right-hand side.

```
flights %>%
  group_by(year, month, day) %>%
  select(arr_delay, dep_delay) %>%
  summarize(
    arr_delay_avg = mean(arr_delay, na.rm = TRUE),
    dep_delay_avg = mean(dep_delay, na.rm = TRUE)
    ) %>%
  filter(arr_delay_avg > 30 | dep_delay_avg >30)
```

```
   year month day arr_delay_avg dep_delay_avg
1  2013     1  16      34.24736      24.61287
2  2013     1  31      32.60285      28.65836
3  2013     2  11      36.29009      39.07360
4  2013     2  27      31.25249      37.76327
5  2013     3   8      85.86216      83.53692
6  2013     3  18      41.29189      30.11796
7  2013     4  10      38.41231      33.02368
8  2013     4  12      36.04814      34.83843
9  2013     4  18      36.02848      34.91536
10 2013     4  19      47.91170      46.12783
..  ...   ... ...           ...           ...
```

Merging Data Sets

In the previous handout, we joined two data frames by a common variable (i.e., an inner join) using the `merge()` function.

EmpsAU

Obs	First	Gender	EmpID
1	Togar	M	121150
2	Kylie	F	121151
3	Birin	M	121152

PhoneH

Obs	EmpID	Phone	First
1	121151	+61(2)5555-1849	Kylie
2	121152	+61(2)5555-1665	Birin
3	121150	+61(2)5555-1793	Togur

```
> EmpsAUH = merge(EmpsAU,PhoneH, by="EmpID")
```

Result:

	EmpID	First.x	Gender	Phone	First.y
1	121150	Togar	M	+61(2)5555-1793	Togur
2	121151	Kylie	F	+61(2)5555-1849	Kylie
3	121152	Birin	M	+61(2)5555-1665	Birin

What if you don't specify a by variable?

```
> EmpsAUH = merge(EmpsAU,PhoneH)
```

	First	EmpID	Gender	Phone
1	Birin	121152	M	+61(2)5555-1665
2	Kylie	121151	F	+61(2)5555-1849

Next, note that Togar's name was misspelled in the `PhoneH` data set. To correct this, we can use the following command.

```
> PhoneH$First = as.character(PhoneH$First)
> PhoneH2=mutate(PhoneH,First=ifelse(First=="Togur","Togar",First))
```

Now, when we merge the two data sets with the following commands, we see the result shown below:

```
> EmpsAUH = merge(EmpsAU,PhoneH2, by="EmpID")
```

	EmpID	First.x	Gender	Phone	First.y
1	121150	Togar	M	+61(2)5555-1793	Togar
2	121151	Kylie	F	+61(2)5555-1849	Kylie
3	121152	Birin	M	+61(2)5555-1665	Birin

```
> EmpsAUH = merge(EmpsAU,PhoneH2)
```

	First	EmpID	Gender	Phone
1	Birin	121152	M	+61(2)5555-1665
2	Kylie	121151	F	+61(2)5555-1849
3	Togar	121150	M	+61(2)5555-1793

```
> EmpsAUH = merge(EmpsAU,PhoneH2, by=c("EmpID","First"))
```

	EmpID	First	Gender	Phone
1	121150	Togar	M	+61(2)5555-1793
2	121151	Kylie	F	+61(2)5555-1849
3	121152	Birin	M	+61(2)5555-1665

Merging Data Sets Using dplyr

Consider the data sets once again.

EmpsAU

Obs	First	Gender	EmpID
1	Togar	M	121150
2	Kylie	F	121151
3	Birin	M	121152

PhoneH

Obs	EmpID	Phone	First
1	121151	+61(2)5555-1849	Kylie
2	121152	+61(2)5555-1665	Birin
3	121150	+61(2)5555-1793	Togur

PhoneH2

Obs	EmpID	Phone	First
1	121151	+61(2)5555-1849	Kylie
2	121152	+61(2)5555-1665	Birin
3	121150	+61(2)5555-1793	Togar

The following command merges the data using an "**inner join**":

```
> EmpsAUH = inner_join(EmpsAU,PhoneH)
```

	First	Gender	EmpID	Phone
1	Kylie	F	121151	+61(2)5555-1849
2	Birin	M	121152	+61(2)5555-1665

```
> EmpsAUH = inner_join(EmpsAU,PhoneH2)
```

	First	Gender	EmpID	Phone
1	Togar	M	121150	+61(2)5555-1793
2	Kylie	F	121151	+61(2)5555-1849
3	Birin	M	121152	+61(2)5555-1665

A "**left join**" can be accomplished as follows:

```
> EmpsAUH = left_join(EmpsAU,PhoneH)
```

	First	Gender	EmpID	Phone
1	Togar	M	121150	NA
2	Kylie	F	121151	+61(2)5555-1849
3	Birin	M	121152	+61(2)5555-1665

```
> EmpsAUH = left_join(EmpsAU,PhoneH2)
```

	First	Gender	EmpID	Phone
1	Togar	M	121150	+61(2)5555-1793
2	Kylie	F	121151	+61(2)5555-1849
3	Birin	M	121152	+61(2)5555-1665

```
> EmpsAUH = left_join(PhoneH,EmpsAU)
```

	EmpID	Phone	First	Gender
1	121151	+61(2)5555-1849	Kylie	F
2	121152	+61(2)5555-1665	Birin	M
3	121150	+61(2)5555-1793	Togur	NA

```
> EmpsAUH = left_join(PhoneH2,EmpsAU)
```

	EmpID	Phone	First	Gender
1	121151	+61(2)5555-1849	Kylie	F
2	121152	+61(2)5555-1665	Birin	M
3	121150	+61(2)5555-1793	Togar	M

Finally, we can also consider a "**full join**".

```
> EmpsAUH = full_join(EmpsAU,PhoneH)
```

	First	Gender	EmpID	Phone
1	Togar	M	121150	NA
2	Kylie	F	121151	+61(2)5555-1849
3	Birin	M	121152	+61(2)5555-1665
4	Togur	NA	121150	+61(2)5555-1793

```
> EmpsAUH = full_join(EmpsAU,PhoneH2)
```

	First	Gender	EmpID	Phone
1	Togar	M	121150	+61(2)5555-1793
2	Kylie	F	121151	+61(2)5555-1849
3	Birin	M	121152	+61(2)5555-1665