Going deeper with dplyr: New features in 0.3 and 0.4

Introduction

In August 2014, I created a 40-minute video tutorial introducing the key functionality of the dplyr package in R, using dplyr version 0.2. Since then, there have been two significant updates to dplyr (0.3 and 0.4), introducing a ton of new features.

This document (created in March 2015) covers the most useful new features in 0.3 and 0.4, as well as other functionality that I didn't cover last time (though it is not necessarily new). My new video tutorial walks through the code below in detail.

If you have not watched the previous tutorial, I recommend you do so first since it covers some dplyr basics that will not be covered in this tutorial.

Loading dplyr and the nycflights13 dataset

Although my last tutorial used data from the hflights package, Hadley Wickham has rewritten the dplyr vignettes to use the nycflights13 package instead, and so I'm also using nycflights13 for the sake of consistency.

```r
# remove flights data if you just finished my previous tutorial
rm(flights)
```

```r
# load packages
suppressMessages(library(dplyr))
#Make sure to install.packages("nycflights13")
library(nycflights13)

# print the flights dataset from nycflights13
flights
```

```
## # A tibble: 336,776 × 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1   2013     1     1      517            515         2      830
## 2   2013     1     1      533            529         4      850
## 3   2013     1     1      542            540         2      923
## 4   2013     1     1      544            545        -1     1004
## 5   2013     1     1      554            600        -6      812
## 6   2013     1     1      554            558        -4      740
## 7   2013     1     1      555            600        -5      913
## 8   2013     1     1      557            600        -3      709
## 9   2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Choosing columns: select, rename

```
# besides just using select() to pick columns...
flights %>% select(carrier, flight)
```

```
## # A tibble: 336,776 × 2
##    carrier flight
##      <chr>  <int>
## 1       UA   1545
## 2       UA   1714
## 3       AA   1141
## 4       B6    725
## 5       DL    461
## 6       UA   1696
## 7       B6    507
## 8       EV   5708
## 9       B6     79
## 10      AA    301
## # ... with 336,766 more rows
```

```
# ...you can use the minus sign to hide columns
flights %>% select(-month, -day)
```

```
## # A tibble: 336,776 × 17
##     year dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int>    <int>          <int>     <dbl>    <int>          <int>
## 1   2013      517            515         2      830            819
## 2   2013      533            529         4      850            830
## 3   2013      542            540         2      923            850
## 4   2013      544            545        -1     1004           1022
## 5   2013      554            600        -6      812            837
## 6   2013      554            558        -4      740            728
## 7   2013      555            600        -5      913            854
## 8   2013      557            600        -3      709            723
## 9   2013      557            600        -3      838            846
## 10  2013      558            600        -2      753            745
## # ... with 336,766 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

```
# hide a range of columns
flights %>% select(-(dep_time:arr_delay))

# hide any column with a matching name
flights %>% select(-contains("time"))
```

```
# pick columns using a character vector of column names
cols <- c("carrier", "flight", "tailnum")
flights %>% select(one_of(cols))
```

```
## # A tibble: 336,776 × 3
##    carrier flight tailnum
##      <chr>  <int>   <chr>
## 1       UA   1545  N14228
## 2       UA   1714  N24211
## 3       AA   1141  N619AA
## 4       B6    725  N804JB
## 5       DL    461  N668DN
## 6       UA   1696  N39463
## 7       B6    507  N516JB
## 8       EV   5708  N829AS
## 9       B6     79  N593JB
## 10      AA    301  N3ALAA
## # ... with 336,766 more rows
```

```
# select() can be used to rename columns, though all columns not mentioned are dropped
flights %>% select(tail = tailnum)
```

```
## # A tibble: 336,776 × 1
##      tail
##     <chr>
## 1  N14228
## 2  N24211
## 3  N619AA
## 4  N804JB
## 5  N668DN
## 6  N39463
## 7  N516JB
## 8  N829AS
## 9  N593JB
## 10 N3ALAA
## # ... with 336,766 more rows
```

```
# rename() does the same thing, except all columns not mentioned are kept
flights %>% rename(tail = tailnum)
```

```
## # A tibble: 336,776 × 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1   2013     1     1      517            515         2      830
## 2   2013     1     1      533            529         4      850
## 3   2013     1     1      542            540         2      923
## 4   2013     1     1      544            545        -1     1004
## 5   2013     1     1      554            600        -6      812
## 6   2013     1     1      554            558        -4      740
## 7   2013     1     1      555            600        -5      913
## 8   2013     1     1      557            600        -3      709
## 9   2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 336,766 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tail <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

Choosing rows: filter, between, slice, sample_n, top_n, distinct

```
# filter() supports the use of multiple conditions
flights %>% filter(dep_time >= 600, dep_time <= 605)
```

```
## # A tibble: 2,460 × 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1   2013     1     1      600            600         0      851
## 2   2013     1     1      600            600         0      837
## 3   2013     1     1      601            600         1      844
## 4   2013     1     1      602            610        -8      812
## 5   2013     1     1      602            605        -3      821
## 6   2013     1     2      600            600         0      814
## 7   2013     1     2      600            605        -5      751
## 8   2013     1     2      600            600         0      819
## 9   2013     1     2      600            600         0      846
## 10  2013     1     2      600            600         0      737
## # ... with 2,450 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

```r
# between() is a concise alternative for determing if numeric values fall in a range
flights %>% filter(between(dep_time, 600, 605))

# side note: is.na() can also be useful when filtering
flights %>% filter(!is.na(dep_time))
```

```r
# slice() filters rows by position
flights %>% slice(1000:1005)
```

```
## # A tibble: 6 × 19
##    year month   day dep_time sched_dep_time dep_delay arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1  2013     1     2      809            810        -1      950
## 2  2013     1     2      810            800        10     1008
## 3  2013     1     2      811            815        -4     1100
## 4  2013     1     2      811            815        -4     1126
## 5  2013     1     2      811            820        -9      944
## 6  2013     1     2      815            815         0     1109
## # ... with 12 more variables: sched_arr_time <int>, arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

```r
# keep the first three rows within each group
flights %>% group_by(month, day) %>% slice(1:3)
```

```
## Source: local data frame [1,095 x 19]
## Groups: month, day [365]
##
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1   2013     1     1      517            515         2      830
## 2   2013     1     1      533            529         4      850
## 3   2013     1     1      542            540         2      923
## 4   2013     1     2       42           2359        43      518
## 5   2013     1     2      126           2250       156      233
## 6   2013     1     2      458            500        -2      703
## 7   2013     1     3       32           2359        33      504
## 8   2013     1     3       50           2145       185      203
## 9   2013     1     3      235           2359       156      700
## 10  2013     1     4       25           2359        26      505
## # ... with 1,085 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

```
# sample three rows from each group
flights %>% group_by(month, day) %>% sample_n(3)
```

```
## Source: local data frame [1,095 x 19]
## Groups: month, day [365]
##
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1   2013     1     1     2053           2055        -2     2254
## 2   2013     1     1     1558           1359       119     1718
## 3   2013     1     1     2209           2155        14     2400
## 4   2013     1     2     1902           1905        -3     2203
## 5   2013     1     2     1257           1300        -3     1529
## 6   2013     1     2     2051           1929        82     2256
## 7   2013     1     3     2046           2045         1     2340
## 8   2013     1     3     1455           1446         9     1801
## 9   2013     1     3     1510           1510         0     1736
## 10  2013     1     4     1643           1615        28     1857
## # ... with 1,085 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

```
# keep three rows from each group with the top dep_delay
flights %>% group_by(month, day) %>% top_n(3, dep_delay)
```

```
## Source: local data frame [1,108 x 19]
## Groups: month, day [365]
##
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1   2013     1     1      848           1835       853     1001
## 2   2013     1     1     1815           1325       290     2120
## 3   2013     1     1     2343           1724       379      314
## 4   2013     1     2     1412            838       334     1710
## 5   2013     1     2     1607           1030       337     2003
## 6   2013     1     2     2131           1512       379     2340
## 7   2013     1     3     2008           1540       268     2339
## 8   2013     1     3     2012           1600       252     2314
## 9   2013     1     3     2056           1605       291     2239
## 10  2013     1     4     2058           1730       208        2
## # ... with 1,098 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

```
# also sort by dep_delay within each group
flights %>% group_by(month, day) %>% top_n(3, dep_delay) %>% arrange(desc(dep_delay))
```

```
## Source: local data frame [1,108 x 19]
## Groups: month, day [365]
## 
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1   2013     1     9      641            900      1301     1242
## 2   2013     6    15     1432           1935      1137     1607
## 3   2013     1    10     1121           1635      1126     1239
## 4   2013     9    20     1139           1845      1014     1457
## 5   2013     7    22      845           1600      1005     1044
## 6   2013     4    10     1100           1900       960     1342
## 7   2013     3    17     2321            810       911      135
## 8   2013     6    27      959           1900       899     1236
## 9   2013     7    22     2257            759       898      121
## 10  2013    12     5      756           1700       896     1058
## # ... with 1,098 more rows, and 12 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

```
# unique rows can be identified using unique() from base R
flights %>% select(origin, dest) %>% unique()
```

```
## # A tibble: 224 × 2
##    origin  dest
##     <chr> <chr>
## 1     EWR   IAH
## 2     LGA   IAH
## 3     JFK   MIA
## 4     JFK   BQN
## 5     LGA   ATL
## 6     EWR   ORD
## 7     EWR   FLL
## 8     LGA   IAD
## 9     JFK   MCO
## 10    LGA   ORD
## # ... with 214 more rows
```

```
# dplyr provides an alternative that is more "efficient"
flights %>% select(origin, dest) %>% distinct()

# side note: when chaining, you don't have to include the parentheses if there are no arguments
flights %>% select(origin, dest) %>% distinct
```

Adding new variables: mutate, transmute, add_rownames

```
# mutate() creates a new variable (and keeps all existing variables)
flights %>% mutate(speed = distance/air_time*60)
```

```
## # A tibble: 336,776 × 20
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1   2013     1     1      517            515         2      830
## 2   2013     1     1      533            529         4      850
## 3   2013     1     1      542            540         2      923
## 4   2013     1     1      544            545        -1     1004
## 5   2013     1     1      554            600        -6      812
## 6   2013     1     1      554            558        -4      740
## 7   2013     1     1      555            600        -5      913
## 8   2013     1     1      557            600        -3      709
## 9   2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## # ... with 336,766 more rows, and 13 more variables: sched_arr_time <int>,
## #   arr_delay <dbl>, carrier <chr>, flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>, speed <dbl>
```

```
# transmute() only keeps the new variables
flights %>% transmute(speed = distance/air_time*60)
```

```
## # A tibble: 336,776 × 1
##       speed
##       <dbl>
## 1  370.0441
## 2  374.2731
## 3  408.3750
## 4  516.7213
## 5  394.1379
## 6  287.6000
## 7  404.4304
## 8  259.2453
## 9  404.5714
## 10 318.6957
## # ... with 336,766 more rows
```

```
# example data frame with row names
mtcars %>% head()
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
# add_rownames() turns row names into an explicit variable
mtcars %>% add_rownames("model") %>% head()
```

```
## Warning: Deprecated, use tibble::rownames_to_column() instead.
```

```
## # A tibble: 6 × 12
##               model   mpg   cyl  disp    hp  drat    wt  qsec    vs    am
##               <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1         Mazda RX4  21.0     6   160   110  3.90 2.620 16.46     0     1
## 2     Mazda RX4 Wag  21.0     6   160   110  3.90 2.875 17.02     0     1
## 3        Datsun 710  22.8     4   108    93  3.85 2.320 18.61     1     1
## 4    Hornet 4 Drive  21.4     6   258   110  3.08 3.215 19.44     1     0
## 5 Hornet Sportabout  18.7     8   360   175  3.15 3.440 17.02     0     0
## 6           Valiant  18.1     6   225   105  2.76 3.460 20.22     1     0
## # ... with 2 more variables: gear <dbl>, carb <dbl>
```

```r
# side note: dplyr no longer prints row names (ever) for local data frames
mtcars %>% tbl_df()
```

```
## # A tibble: 32 × 11
##      mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
## *  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1   21.0     6 160.0   110  3.90 2.620 16.46     0     1     4     4
## 2   21.0     6 160.0   110  3.90 2.875 17.02     0     1     4     4
## 3   22.8     4 108.0    93  3.85 2.320 18.61     1     1     4     1
## 4   21.4     6 258.0   110  3.08 3.215 19.44     1     0     3     1
## 5   18.7     8 360.0   175  3.15 3.440 17.02     0     0     3     2
## 6   18.1     6 225.0   105  2.76 3.460 20.22     1     0     3     1
## 7   14.3     8 360.0   245  3.21 3.570 15.84     0     0     3     4
## 8   24.4     4 146.7    62  3.69 3.190 20.00     1     0     4     2
## 9   22.8     4 140.8    95  3.92 3.150 22.90     1     0     4     2
## 10  19.2     6 167.6   123  3.92 3.440 18.30     1     0     4     4
## # ... with 22 more rows
```

Grouping and counting: summarise, tally, count, group_size, n_groups, ungroup

```r
# summarise() can be used to count the number of rows in each group
flights %>% group_by(month) %>% summarise(cnt = n())
```

```
## # A tibble: 12 × 2
##    month   cnt
##    <int> <int>
## 1      1 27004
## 2      2 24951
## 3      3 28834
## 4      4 28330
## 5      5 28796
## 6      6 28243
## 7      7 29425
## 8      8 29327
## 9      9 27574
## 10    10 28889
## 11    11 27268
## 12    12 28135
```

```r
# tally() and count() can do this more concisely
flights %>% group_by(month) %>% tally()
flights %>% count(month)
```

```r
# you can sort by the count
flights %>% group_by(month) %>% summarise(cnt = n()) %>% arrange(desc(cnt))
```

```
## # A tibble: 12 × 2
##    month   cnt
##    <int> <int>
## 1      7 29425
## 2      8 29327
## 3     10 28889
## 4       3 28834
## 5       5 28796
## 6       4 28330
## 7       6 28243
## 8      12 28135
## 9       9 27574
## 10     11 27268
## 11      1 27004
## 12      2 24951
```

```
# tally() and count() have a sort parameter for this purpose
flights %>% group_by(month) %>% tally(sort=TRUE)
flights %>% count(month, sort=TRUE)
```

```
# you can sum over a specific variable instead of simply counting rows
flights %>% group_by(month) %>% summarise(dist = sum(distance))
```

```
## # A tibble: 12 × 2
##    month     dist
##    <int>    <dbl>
## 1      1 27188805
## 2      2 24975509
## 3      3 29179636
## 4      4 29427294
## 5      5 29974128
## 6      6 29856388
## 7      7 31149199
## 8      8 31149334
## 9      9 28711426
## 10    10 30012086
## 11    11 28639718
## 12    12 29954084
```

```
# tally() and count() have a wt parameter for this purpose
flights %>% group_by(month) %>% tally(wt = distance)
flights %>% count(month, wt = distance)
```

```
# group_size() returns the counts as a vector
flights %>% group_by(month) %>% group_size()
```

```
##  [1] 27004 24951 28834 28330 28796 28243 29425 29327 27574 28889 27268
## [12] 28135
```

```
# n_groups() simply reports the number of groups
flights %>% group_by(month) %>% n_groups()
```

```
## [1] 12
```

```
# group by two variables, summarise, arrange (output is possibly confusing)
flights %>% group_by(month, day) %>% summarise(cnt = n()) %>% arrange(desc(cnt)) %>% print(n = 40)
```

```
## Source: local data frame [365 x 3]
## Groups: month [12]
##
##    month   day   cnt
##    <int> <int> <int>
## 1     11    27  1014
## 2      7    11  1006
## 3      7     8  1004
## 4      7    10  1004
## 5     12     2  1004
## 6      7    18  1003
## 7      7    25  1003
## 8      7    12  1002
## 9      7     9  1001
## 10     7    17  1001
## 11     7    31  1001
## 12     8     7  1001
## 13     8     8  1001
## 14     8    12  1001
## 15     7    22  1000
## 16     7    24  1000
## 17     8     1  1000
## 18     8     5  1000
## 19     8    15  1000
## 20    11    21  1000
## 21     7    15   999
## 22     7    19   999
## 23     7    26   999
## 24     7    29   999
## 25     8     2   999
## 26     8     9   999
## 27    11    22   999
## 28     8    16   998
## 29     7    23   997
## 30     7    30   997
## 31     8    14   997
## 32     7    16   996
## 33     8     6   996
## 34     8    19   996
## 35     9    13   996
## 36     9    26   996
## 37     9    27   996
## 38     4    15   995
## 39     6    20   995
## 40     6    26   995
## # ... with 325 more rows
```

```
# ungroup() before arranging to arrange across all groups
flights %>% group_by(month, day) %>% summarise(cnt = n()) %>% ungroup() %>% arrange(desc(cnt))
```

```
## # A tibble: 365 × 3
##    month   day   cnt
##    <int> <int> <int>
## 1     11    27  1014
## 2      7    11  1006
## 3      7     8  1004
## 4      7    10  1004
## 5     12     2  1004
## 6      7    18  1003
## 7      7    25  1003
## 8      7    12  1002
## 9       7      9  1001
## 10      7     17  1001
## # ... with 355 more rows
```

Creating data frames: data_frame

`data_frame()` is a better way than `data.frame()` for creating data frames. Benefits of `data_frame()`:

- You can use previously defined columns to compute new columns.
- It never coerces column types.
- It never munges column names.
- It never adds row names.
- It only recycles length 1 input.
- It returns a local data frame (a tbl_df).

```
# data_frame() example
data_frame(a = 1:6, b = a*2, c = 'string', 'd+e' = 1) %>% glimpse()
```

```
## Observations: 6
## Variables: 4
## $ a   <int> 1, 2, 3, 4, 5, 6
## $ b   <dbl> 2, 4, 6, 8, 10, 12
## $ c   <chr> "string", "string", "string", "string", "string", "string"
## $ d+e <dbl> 1, 1, 1, 1, 1, 1
```

```
# data.frame() example
data.frame(a = 1:6, c = 'string', 'd+e' = 1) %>% glimpse()
```

```
## Observations: 6
## Variables: 3
## $ a   <int> 1, 2, 3, 4, 5, 6
## $ c   <fctr> string, string, string, string, string, string
## $ d.e <dbl> 1, 1, 1, 1, 1, 1
```

Joining (merging) tables: left_join, right_join, inner_join, full_join, semi_join, anti_join

```
# create two simple data frames
(a <- data_frame(color = c("green","yellow","red"), num = 1:3))
```

```
## # A tibble: 3 × 2
##    color   num
##    <chr> <int>
## 1  green     1
## 2 yellow     2
## 3    red     3
```

```
(b <- data_frame(color = c("green","yellow","pink"), size = c("S","M","L")))
```

```
## # A tibble: 3 × 2
##    color  size
##    <chr> <chr>
## 1  green     S
## 2 yellow     M
## 3   pink     L
```

```r
# only include observations found in both "a" and "b" (automatically joins on variables that appear in
inner_join(a, b)
```

```
## Joining, by = "color"
```

```
## # A tibble: 2 × 3
##    color   num  size
##    <chr> <int> <chr>
## 1  green     1     S
## 2 yellow     2     M
```

```r
# include observations found in either "a" or "b"
full_join(a, b)
```

```
## Joining, by = "color"
```

```
## # A tibble: 4 × 3
##    color   num  size
##    <chr> <int> <chr>
## 1  green     1     S
## 2 yellow     2     M
## 3    red     3  <NA>
## 4   pink    NA     L
```

```r
# include all observations found in "a"
left_join(a, b)
```

```
## Joining, by = "color"
```

```
## # A tibble: 3 × 3
##    color   num  size
##    <chr> <int> <chr>
## 1  green     1     S
## 2 yellow     2     M
## 3    red     3  <NA>
```

```r
# include all observations found in "b"
right_join(a, b)
```

```
## Joining, by = "color"
```

```
## # A tibble: 3 × 3
##    color   num  size
##    <chr> <int> <chr>
## 1  green     1     S
## 2 yellow     2     M
## 3   pink    NA     L
```

```r
# right_join(a, b) is identical to left_join(b, a) except for column ordering
left_join(b, a)
```

```
## Joining, by = "color"
```

```
## # A tibble: 3 × 3
##    color  size   num
##    <chr> <chr> <int>
## 1  green     S     1
## 2 yellow     M     2
## 3   pink     L    NA
```

```
# filter "a" to only show observations that match "b"
semi_join(a, b)
```

```
## Joining, by = "color"

## # A tibble: 2 × 2
##    color   num
##    <chr> <int>
## 1  green     1
## 2 yellow     2
```

```
# filter "a" to only show observations that don't match "b"
anti_join(a, b)
```

```
## Joining, by = "color"

## # A tibble: 1 × 2
##   color   num
##   <chr> <int>
## 1   red     3
```

```
# sometimes matching variables don't have identical names
b <- b %>% rename(col = color)

# specify that the join should occur by matching "color" in "a" with "col" in "b"
inner_join(a, b, by=c("color" = "col"))
```

```
## # A tibble: 2 × 3
##    color   num  size
##    <chr> <int> <chr>
## 1  green     1     S
## 2 yellow     2     M
```

Viewing more output: print, View

```
# specify that you want to see more rows
flights %>% print(n = 15)
```

```
## # A tibble: 336,776 × 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1   2013     1     1      517            515         2      830
## 2   2013     1     1      533            529         4      850
## 3   2013     1     1      542            540         2      923
## 4   2013     1     1      544            545        -1     1004
## 5   2013     1     1      554            600        -6      812
## 6   2013     1     1      554            558        -4      740
## 7   2013     1     1      555            600        -5      913
## 8   2013     1     1      557            600        -3      709
## 9   2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
## 11  2013     1     1      558            600        -2      849
## 12  2013     1     1      558            600        -2      853
## 13  2013     1     1      558            600        -2      924
## 14  2013     1     1      558            600        -2      923
## 15  2013     1     1      559            600        -1      941
## # ... with 3.368e+05 more rows, and 12 more variables:
## #   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
# specify that you want to see ALL rows (don't run this!)
flights %>% print(n = Inf)
```

```
# specify that you want to see all columns
flights %>% print(width = Inf)
```

```
## # A tibble: 336,776 × 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1   2013     1     1      517            515         2      830
## 2   2013     1     1      533            529         4      850
## 3   2013     1     1      542            540         2      923
## 4   2013     1     1      544            545        -1     1004
## 5   2013     1     1      554            600        -6      812
## 6   2013     1     1      554            558        -4      740
## 7   2013     1     1      555            600        -5      913
## 8   2013     1     1      557            600        -3      709
## 9   2013     1     1      557            600        -3      838
## 10  2013     1     1      558            600        -2      753
##    sched_arr_time arr_delay carrier flight tailnum origin  dest air_time
##             <int>     <dbl>   <chr>  <int>   <chr>  <chr> <chr>    <dbl>
## 1             819        11      UA   1545  N14228    EWR   IAH      227
## 2             830        20      UA   1714  N24211    LGA   IAH      227
## 3             850        33      AA   1141  N619AA    JFK   MIA      160
## 4            1022       -18      B6    725  N804JB    JFK   BQN      183
## 5             837       -25      DL    461  N668DN    LGA   ATL      116
## 6             728        12      UA   1696  N39463    EWR   ORD      150
## 7             854        19      B6    507  N516JB    EWR   FLL      158
## 8             723       -14      EV   5708  N829AS    LGA   IAD       53
## 9             846        -8      B6     79  N593JB    JFK   MCO      140
## 10            745         8      AA    301  N3ALAA    LGA   ORD      138
##    distance  hour minute           time_hour
##       <dbl> <dbl>  <dbl>              <dttm>
## 1      1400     5     15 2013-01-01 05:00:00
## 2      1416     5     29 2013-01-01 05:00:00
## 3      1089     5     40 2013-01-01 05:00:00
## 4      1576     5     45 2013-01-01 05:00:00
## 5       762     6      0 2013-01-01 06:00:00
## 6       719     5     58 2013-01-01 05:00:00
## 7      1065     6      0 2013-01-01 06:00:00
## 8       229     6      0 2013-01-01 06:00:00
## 9       944     6      0 2013-01-01 06:00:00
## 10      733     6      0 2013-01-01 06:00:00
## # ... with 336,766 more rows
```

```
# show up to 1000 rows and all columns
flights %>% View()

# set option to see all columns and fewer rows
```

```r
options(dplyr.width = Inf, dplyr.print_min = 6)

# reset options (or just close R)
options(dplyr.width = NULL, dplyr.print_min = 10)
```

Resources

- Release announcements for version 0.3 and version 0.4
- dplyr reference manual and vignettes
- Two-table vignette covering joins and set operations
- RStudio's Data Wrangling Cheat Sheet for dplyr and tidyr
- dplyr GitHub repo and list of releases

Data School

- Blog
- Email newsletter
- YouTube channel

< END OF DOCUMENT >