# Block 03 - Data Handling

*Marcelo*

*20 November 2017*

```
#rm(list = ls())
library(dplyr)
library(nycflights13)
```

dplyr aims to provide a function for each basic verb of data manipulating:

- filter() (and slice())
- arrange()
- select() (and rename())
- distinct()
- mutate() (and transmute())
- summarise()
- sample_n() and sample_frac()

```
class(flights)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
dfFlights <- as.data.frame(flights)
```

```
flights[flights$month == 7 &
          flights$day == 1,  ]
```

```
## # A tibble: 966 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     7     1        1           2029       212      236
##  2  2013     7     1        2           2359         3      344
##  3  2013     7     1       29           2245       104      151
##  4  2013     7     1       43           2130       193      322
##  5  2013     7     1       44           2150       174      300
##  6  2013     7     1       46           2051       235      304
##  7  2013     7     1       48           2001       287      308
##  8  2013     7     1       58           2155       183      335
##  9  2013     7     1      100           2146       194      327
## 10  2013     7     1      100           2245       135      337
## # ... with 956 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>
```

```
system.time(filter(flights, month == 7, day == 1))
```

```
##    user  system elapsed
##    0.01    0.00    0.01
```

```
system.time(filter(dfFlights, dep_delay < 0 & arr_delay > 0))
```

```
##    user  system elapsed
##    0.02    0.00    0.02
```

```
#arrange(flights, year, month, day)
arrange(flights, desc(arr_delay))
```

```
## # A tibble: 336,776 x 19
##     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     7    22     2257            759       898      121
##  9  2013    12     5      756           1700       896     1058
## 10  2013     5     3     1133           2055       878     1250
## # ... 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>
```

all the following outputs are identical

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

```
## # A tibble: 336,776 x 3
##     year month   day
##    <int> <int> <int>
##  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
## # ... with 336,766 more rows
```

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

```
## # A tibble: 336,776 x 3
##     year month   day
##    <int> <int> <int>
##  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
## # ... with 336,766 more rows
```

```
select(flights, 1:3)
```

```
## # A tibble: 336,776 x 3
##     year month   day
##    <int> <int> <int>
##  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
## # ... with 336,766 more rows
```

## Excercises: Subset, Filter and Arrange data frames

### 1 and 2

```
dataPath <- "./Datasets/Schweiz/"
filesToLoad <- list.files(path = dataPath, full.names = TRUE)
filesToLoad
```

```
## [1] "./Datasets/Schweiz/Schweiz1.txt" "./Datasets/Schweiz/Schweiz2.txt"
## [3] "./Datasets/Schweiz/Schweiz3.txt" "./Datasets/Schweiz/Schweiz4.txt"
```

### 3

```
dat1 <- read.csv2(file = filesToLoad[1])
str(dat1, vec.len = 1)
```

```
## 'data.frame':    47 obs. of  7 variables:
##  $ Province        : Factor w/ 47 levels "Aigle","Aubonne",..: 8 9 ...
##  $ Fertility       : num  80.2 83.1 ...
##  $ Agriculture     : num  17 45.1 ...
##  $ Examination     : int  15 6 ...
##  $ Education       : int  12 9 ...
##  $ Catholic        : num  9.96 ...
##  $ Infant.Mortality: num  22.2 22.2 ...
```

```
dat2 <- read.csv(file = filesToLoad[2])
```

```
dat1 <- read.table(file= filesToLoad[1],
                   header = TRUE,
                   sep = ";",
                   dec = ",",
                   stringsAsFactors = T)
head(dat1)
```

```
##       Province Fertility Agriculture Examination Education Catholic
## 1   Courtelary      80.2        17.0          15        12     9.96
```

```
## 2     Delemont      83.1        45.1           6         9    84.84
## 3 Franches-Mnt      92.5        39.7           5         5    93.40
## 4      Moutier      85.8        36.5          12         7    33.77
## 5   Neuveville      76.9        43.5          17        15     5.16
## 6   Porrentruy      76.1        35.3           9         7    90.57
##   Infant.Mortality
## 1             22.2
## 2             22.2
## 3             20.2
## 4             20.3
## 5             20.6
## 6             26.6
```

```
dat2 <- read.table(file= filesToLoad[2],
                   header = TRUE,
                   sep = ",",
                   dec = ".",
                   stringsAsFactors = T)
head(dat2)
```

```
##       Province Fertility Agriculture Examination Education Catholic
## 1   Courtelary      80.2        17.0          15        12     9.96
## 2     Delemont      83.1        45.1           6         9    84.84
## 3 Franches-Mnt      92.5        39.7           5         5    93.40
## 4      Moutier      85.8        36.5          12         7    33.77
## 5   Neuveville      76.9        43.5          17        15     5.16
## 6   Porrentruy      76.1        35.3           9         7    90.57
##   Infant.Mortality
## 1             22.2
## 2             22.2
## 3             20.2
## 4             20.3
## 5             20.6
## 6             26.6
```

```
dat3 <- read.table(file= filesToLoad[3],
                   header = TRUE,
                   sep = "_",
                   dec = "-",
                   stringsAsFactors = T)
head(dat3)
```

```
##       Province Fertility Agriculture Examination Education Catholic
## 1   Courtelary      80.2        17.0          15        12     9.96
## 2     Delemont      83.1        45.1           6         9    84.84
## 3 Franches-Mnt      92.5        39.7           5         5    93.40
## 4      Moutier      85.8        36.5          12         7    33.77
## 5   Neuveville      76.9        43.5          17        15     5.16
## 6   Porrentruy      76.1        35.3           9         7    90.57
##   Infant.Mortality
## 1             22.2
## 2             22.2
## 3             20.2
## 4             20.3
## 5             20.6
## 6             26.6
```

```
dat4 <- read.table(file= filesToLoad[4],
                   header = TRUE,
                   sep = "\t",
                   dec = ",",
                   stringsAsFactors = T)
head(dat4)
```

```
##       Province Fertility Agriculture Examination Education Catholic
## 1   Courtelary      80.2        17.0          15        12     9.96
## 2     Delemont      83.1        45.1           6         9    84.84
## 3 Franches-Mnt      92.5        39.7           5         5    93.40
## 4      Moutier      85.8        36.5          12         7    33.77
## 5   Neuveville      76.9        43.5          17        15     5.16
## 6   Porrentruy      76.1        35.3           9         7    90.57
##   Infant.Mortality
## 1             22.2
## 2             22.2
## 3             20.2
## 4             20.3
## 5             20.6
## 6             26.6
```

```
all(identical(dat1, dat2),
    identical(dat3, dat4),
    identical(dat1, dat3))
```

```
## [1] TRUE
```

```
datSub1 <- dat1[-(1:3), 1:6]
datSub2 <- dat1[-(4:10), c(1, 7)]
```

**9.**

Use the * join functions from the dplyr package to reunite the two subsets to one data frame named datJoin. Read the help file for inner join. Which column should be used for joining? Your result should look like:

```
datJoin <- full_join(x = datSub1,
                     y = datSub2,
                     by = "Province")
str(datJoin, vec.len = 1)
```

```
## 'data.frame':    47 obs. of  7 variables:
##  $ Province        : Factor w/ 47 levels "Aigle","Aubonne",..: 26 28 ...
##  $ Fertility       : num  85.8 76.9 ...
##  $ Agriculture     : num  36.5 43.5 ...
##  $ Examination     : int  12 17 ...
##  $ Education       : int  7 15 ...
##  $ Catholic        : num  33.8 ...
##  $ Infant.Mortality: num  NA NA ...
```

```
identical(dim(dat1), dim(datJoin))
```

```
## [1] TRUE
```

```
identical(dat1, datJoin)
```

```
## [1] FALSE
```

## Subset, Filter and Arrange data frames

```
data(swiss)
?swiss
```

```
## starting httpd help server ... done
```

**Create a new column swiss$Province in which you store the name of the provinces corresponding to the observations. Delete the rownames afterwards.**

```
swiss <- swiss %>%
  mutate(Province = rownames(swiss)) %>%
  select(7, 1:6)
swiss
```

```
##        Province Fertility Agriculture Examination Education Catholic
## 1    Courtelary      80.2        17.0          15        12     9.96
## 2      Delemont      83.1        45.1           6         9    84.84
## 3  Franches-Mnt      92.5        39.7           5         5    93.40
## 4       Moutier      85.8        36.5          12         7    33.77
## 5    Neuveville      76.9        43.5          17        15     5.16
## 6    Porrentruy      76.1        35.3           9         7    90.57
## 7         Broye      83.8        70.2          16         7    92.85
## 8         Glane      92.4        67.8          14         8    97.16
## 9       Gruyere      82.4        53.3          12         7    97.67
## 10       Sarine      82.9        45.2          16        13    91.38
## 11      Veveyse      87.1        64.5          14         6    98.61
## 12        Aigle      64.1        62.0          21        12     8.52
## 13      Aubonne      66.9        67.5          14         7     2.27
## 14     Avenches      68.9        60.7          19        12     4.43
## 15     Cossonay      61.7        69.3          22         5     2.82
## 16    Echallens      68.3        72.6          18         2    24.20
## 17     Grandson      71.7        34.0          17         8     3.30
## 18     Lausanne      55.7        19.4          26        28    12.11
## 19    La Vallee      54.3        15.2          31        20     2.15
## 20       Lavaux      65.1        73.0          19         9     2.84
## 21       Morges      65.5        59.8          22        10     5.23
## 22       Moudon      65.0        55.1          14         3     4.52
## 23        Nyone      56.6        50.9          22        12    15.14
## 24         Orbe      57.4        54.1          20         6     4.20
## 25         Oron      72.5        71.2          12         1     2.40
## 26      Payerne      74.2        58.1          14         8     5.23
## 27 Paysd'enhaut      72.0        63.5           6         3     2.56
## 28        Rolle      60.5        60.8          16        10     7.72
## 29        Vevey      58.3        26.8          25        19    18.46
## 30      Yverdon      65.4        49.5          15         8     6.10
## 31      Conthey      75.5        85.9           3         2    99.71
## 32    Entremont      69.3        84.9           7         6    99.68
## 33       Herens      77.3        89.7           5         2   100.00
## 34     Martigwy      70.5        78.2          12         6    98.96
## 35      Monthey      79.4        64.9           7         3    98.22
```

```
## 36   St Maurice      65.0        75.9          9         9    99.06
## 37       Sierre      92.2        84.6          3         3    99.46
## 38         Sion      79.3        63.1         13        13    96.83
## 39       Boudry      70.4        38.4         26        12     5.62
## 40 La Chauxdfnd      65.7         7.7         29        11    13.79
## 41     Le Locle      72.7        16.7         22        13    11.22
## 42    Neuchatel      64.4        17.6         35        32    16.92
## 43   Val de Ruz      77.6        37.6         15         7     4.97
## 44 ValdeTravers      67.6        18.7         25         7     8.65
## 45 V. De Geneve      35.0         1.2         37        53    42.34
## 46  Rive Droite      44.7        46.6         16        29    50.43
## 47  Rive Gauche      42.8        27.7         22        29    58.33
##    Infant.Mortality
## 1              22.2
## 2              22.2
## 3              20.2
## 4              20.3
## 5              20.6
## 6              26.6
## 7              23.6
## 8              24.9
## 9              21.0
## 10             24.4
## 11             24.5
## 12             16.5
## 13             19.1
## 14             22.7
## 15             18.7
## 16             21.2
## 17             20.0
## 18             20.2
## 19             10.8
## 20             20.0
## 21             18.0
## 22             22.4
## 23             16.7
## 24             15.3
## 25             21.0
## 26             23.8
## 27             18.0
## 28             16.3
## 29             20.9
## 30             22.5
## 31             15.1
## 32             19.8
## 33             18.3
## 34             19.4
## 35             20.2
## 36             17.8
## 37             16.3
## 38             18.1
## 39             20.3
## 40             20.5
## 41             18.9
```

```
## 42             23.0
## 43             20.0
## 44             19.5
## 45             18.0
## 46             18.2
## 47             19.3
```

**Identify the five provinces with the lowest percentage of males involved in agriculture as occupation.**

```
swiss %>%
  arrange(Agriculture) %>%
  select(Province, Agriculture) %>%
  slice(1:5)
```

```
## # A tibble: 5 x 2
##       Province Agriculture
##          <chr>       <dbl>
## 1 V. De Geneve         1.2
## 2 La Chauxdfnd         7.7
## 3    La Vallee        15.2
## 4     Le Locle        16.7
## 5   Courtelary        17.0
```

### Find the five provinces with the highest percentage of catholic population.

```
swiss %>%
  arrange(desc(Catholic)) %>%
  select(Province, Catholic) %>%
  slice(1:5)
```

```
## # A tibble: 5 x 2
##     Province Catholic
##        <chr>    <dbl>
## 1     Herens   100.00
## 2    Conthey    99.71
## 3  Entremont    99.68
## 4     Sierre    99.46
## 5 St Maurice    99.06
```

### Data Manipulation

```
rm(list = ls())
library(dplyr)
```

```
urlRedWine <-
"http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
urlWhiteWine <-
"http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv"
readLines(urlRedWine, n = 5)
```

```
## [1] "\"fixed acidity\";\"volatile acidity\";\"citric acid\";\"residual sugar\";\"chlorides\";\"free s
## [2] "7.4;0.7;0;1.9;0.076;11;34;0.9978;3.51;0.56;9.4;5"
## [3] "7.8;0.88;0;2.6;0.098;25;67;0.9968;3.2;0.68;9.8;5"
## [4] "7.8;0.76;0.04;2.3;0.092;15;54;0.997;3.26;0.65;9.8;5"
## [5] "11.2;0.28;0.56;1.9;0.075;17;60;0.998;3.16;0.58;9.8;6"
```

```
readLines(urlWhiteWine, n = 5)
```

```
## [1] "\"fixed acidity\";\"volatile acidity\";\"citric acid\";\"residual sugar\";\"chlorides\";\"free s
## [2] "7;0.27;0.36;20.7;0.045;45;170;1.001;3;0.45;8.8;6"
## [3] "6.3;0.3;0.34;1.6;0.049;14;132;0.994;3.3;0.49;9.5;6"
## [4] "8.1;0.28;0.4;6.9;0.05;30;97;0.9951;3.26;0.44;10.1;6"
## [5] "7.2;0.23;0.32;8.5;0.058;47;186;0.9956;3.19;0.4;9.9;6"
```

```
redWine <- read.table(file = urlRedWine,
                      header = TRUE,
                      sep = ";",
                      dec = ".")

whiteWine <- read.table(file = urlWhiteWine,
                      header = TRUE,
                      sep = ";",
                      dec = ".")
```

```
redWine <- mutate(.data = redWine, color = "red")
whiteWine <- mutate(.data = whiteWine, color = "white")
```

```
wine <- bind_rows(redWine, whiteWine)
str(wine, vec.len = 2)
```

```
## 'data.frame':    6497 obs. of  13 variables:
##  $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 ...
##  $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 ...
##  $ free.sulfur.dioxide : num  11 25 15 17 11 ...
##  $ total.sulfur.dioxide: num  34 67 54 60 34 ...
##  $ density             : num  0.998 0.997 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 ...
##  $ quality             : int  5 5 5 6 5 ...
##  $ color               : chr  "red" "red" ...
```

```
rm(redWine, whiteWine)
```

```
# apply condition
wine <- wine %>%
  mutate(alcoholClass = ifelse(test = alcohol < 10,
                               yes = "low",
                               no = ifelse(test = alcohol < 12,
                                           yes = "medium",
                                           no = "high")))

# apply ordered factor
wine$alcoholClass <- factor(x = wine$alcoholClass,
```

```
                              levels = c("low", "medium", "high"),
                              ordered = TRUE)

#display aggregated table
table(wine$alcoholClass, wine$color)
```

```
##
##           red white
##   low     680  1923
##   medium  757  2162
##   high    162   813
```

```
prop.table(table(wine$alcoholClass, wine$color), margin = 2)
```

```
##
##                red     white
##   low    0.4252658 0.3926092
##   medium 0.4734209 0.4414047
##   high   0.1013133 0.1659861
```

**6. Figure out how the same results can be obtained using dplyr functions:**

```
wine %>%
  select(color, alcoholClass) %>%
  count(color, alcoholClass)
```

```
## # A tibble: 6 x 3
##   color alcoholClass     n
##   <chr>        <ord> <int>
## 1   red          low   680
## 2   red       medium   757
## 3   red         high   162
## 4 white          low  1923
## 5 white       medium  2162
## 6 white         high   813
```

```
# not quite right
wine %>%
  select(color, alcoholClass) %>%
  count(color, alcoholClass) %>%
  group_by(color) %>%  mutate(ratio = n / sum(n))
```

```
## # A tibble: 6 x 4
## # Groups:   color [2]
##   color alcoholClass     n     ratio
##   <chr>        <ord> <int>     <dbl>
## 1   red          low   680 0.4252658
## 2   red       medium   757 0.4734209
## 3   red         high   162 0.1013133
## 4 white          low  1923 0.3926092
## 5 white       medium  2162 0.4414047
## 6 white         high   813 0.1659861
```

**7. Calculate the mean values for alcohol, density and pH for each combination of quality and color using the aggregate function.**

```
wineAgg <- wine %>%
  select(quality, color, alcohol, density, pH)
```

```
aggregate(x = wineAgg[,-(1:2)],
          by = list(wineAgg$quality, wineAgg$color),
          FUN = mean)
```

```
##    Group.1 Group.2   alcohol   density       pH
## 1        3     red  9.955000 0.9974640 3.398000
## 2        4     red 10.265094 0.9965425 3.381509
## 3        5     red  9.899706 0.9971036 3.304949
## 4        6     red 10.629519 0.9966151 3.318072
## 5        7     red 11.465913 0.9961043 3.290754
## 6        8     red 12.094444 0.9952122 3.267222
## 7        3   white 10.345000 0.9948840 3.187500
## 8        4   white 10.152454 0.9942767 3.182883
## 9        5   white  9.808840 0.9952626 3.168833
## 10       6   white 10.575372 0.9939613 3.188599
## 11       7   white 11.367936 0.9924524 3.213898
## 12       8   white 11.636000 0.9922359 3.218686
## 13       9   white 12.180000 0.9914600 3.308000
```

# Block 03.02 - 27.11.17

```
rm(list = ls())
```

```
require(nycflights13)
require(dplyr)
```

Rename at select

```
select(flights, depTime = dep_time)
```

```
## # A tibble: 336,776 x 1
##    depTime
##      <int>
##  1     517
##  2     533
##  3     542
##  4     544
##  5     554
##  6     554
##  7     555
##  8     557
##  9     557
## 10     558
## # ... with 336,766 more rows
```

rename (returns full DataFrame)

```
rename(flights, depTime = dep_time)
```

```
## # A tibble: 336,776 x 19
##     year month   day depTime 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>
```

analogous to unique, but for whole dataFrame

```
distinct(flights, tailnum)
```

```
## # A tibble: 4,044 x 1
##    tailnum
##      <chr>
##  1  N14228
##  2  N24211
##  3  N619AA
##  4  N804JB
##  5  N668DN
##  6  N39463
##  7  N516JB
##  8  N829AS
##  9  N593JB
## 10  N3ALAA
## # ... with 4,034 more rows
```

list all flights connections (unique origin AND destination)

```
distinct(select(flights, origin, dest))
```

```
## # A tibble: 224 x 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
```

mutate

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

```
## # A tibble: 336,776 x 6
##     gain    speed distance air_time arr_delay dep_delay
##    <dbl>    <dbl>    <dbl>    <dbl>     <dbl>     <dbl>
##  1     9 370.0441     1400      227        11         2
##  2    16 374.2731     1416      227        20         4
##  3    31 408.3750     1089      160        33         2
##  4   -17 516.7213     1576      183       -18        -1
##  5   -19 394.1379      762      116       -25        -6
##  6    16 287.6000      719      150        12        -4
##  7    24 404.4304     1065      158        19        -5
##  8   -11 259.2453      229       53       -14        -3
##  9    -5 404.5714      944      140        -8        -3
## 10    10 318.6957      733      138         8        -2
## # ... with 336,766 more rows
```

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

```
## # A tibble: 336,776 x 2
##     gain    speed
##    <dbl>    <dbl>
##  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
## # ... with 336,766 more rows
```

```
summarise(flights,
          delay = mean(arr_delay - dep_delay, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##       delay
##       <dbl>
## 1 -5.659779
```

```
sample_n(flights, 20)
```

```
## # A tibble: 20 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     9    19     2105           1805       180     2217
```

```
##  2  2013     3    16     1558           1605        -7     1741
##  3  2013     6    25     2217           2030       107     2340
##  4  2013    11    16     1128           1119         9     1432
##  5  2013    12    15     1013           1019        -6     1331
##  6  2013     8    10      558            602        -4      737
##  7  2013     4     7     1435           1200       155     1646
##  8  2013     7    23       NA           1500        NA       NA
##  9  2013     6    21     1733           1725         8     2024
## 10  2013     1    19      759            759         0      904
## 11  2013     7     6      853            853         0     1125
## 12  2013    12    15      926            900        26     1232
## 13  2013    10    10      559            600        -1      817
## 14  2013     2    20     1957           2000        -3     2111
## 15  2013     2    23      915            920        -5     1420
## 16  2013    11     9     2354           2359        -5      417
## 17  2013     1     5     1746           1745         1     2041
## 18  2013     7    14     1414           1425       -11     1545
## 19  2013    10    19      825            825         0     1136
## 20  2013     8    18     1529           1514        15     1638
## # ... 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>
```

```
sample_frac(flights, 0.001)
```

```
## # A tibble: 337 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013    11    21     1700           1700         0     1821
##  2  2013    12     2     2130           2100        30     2241
##  3  2013    12    20     2257           2141        76      152
##  4  2013     9    30     1146           1154        -8     1251
##  5  2013     4     4     2102           2059         3       17
##  6  2013    12    18     1308            930       218     1836
##  7  2013     6    29     1909           1826        43     2139
##  8  2013     5    21     1632           1529        63     1719
##  9  2013     6     4     1827           1830        -3     2200
## 10  2013     1     7     1455           1455         0     1623
## # ... with 327 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>
```

## Grouped operations

At the most basic level, you can only alter a data frame in five useful ways:

1. you can reorder the rows (arrange()),
2. pick observations (filter() ) and
3. variables of interest (select()),
4. add new variables that are functions of existing variables (mutate()) or
5. collapse many values to a summary (summarise()). The remainder of the language comes from applying the five functions to different types of data, like to grouped data, as described next.