Solutions to G. Grolemund & H. Wickhams's R for Data Science, Chapter 5

Krista DeStasio

8/2/2017

Contents

A Brief Introduction to This File

This R file walks through G. Grolemund & H. Wickhams's online text, "R for Data Science." Much of the code is sourced directly from the book and credit belongs to the authors. Here, some sections of code are heavily commented so that the beginning R programmer can read through and understand what each line of code does and compare it to their own as they work through the text. Throughout, the book provides the primary and most thorough explanation. **For the greatest learning benefit, I suggest you attempt each exercise on your own before looking at the code or write-ups provided here.** Of course, there is more than one way to write code and you may find a more elegant solution that you prefer.

For those new to R and RStudio, it may be of additional benefit to knit the document and examine how the code in the Rmd file is visually expressed in the resultant knitted document. For example, see how the `["R for Data Science."](http://r4ds.had.co.nz/index.html)` is expressed as a hyperlink in the preceeding paragraph where it was not surrounded by tick-marks and compare that to how the same text is expressed in this paragraph when surrounded by ticks. See also the difference in appearance when knitting to different document types (HTML, PDF, Word).

Tip: *If you are using RStudio, click the text next to the orange # box at the bottom of the editor window to easily navigate the code chunks.*

Tip: *Use the ? before any command to view the documentation on that function. Do this often. For example, type `?setwd` to see a description, usage, arguments, and more for the function `setwd()`.*

Tip: Find RStudio Cheatsheets at https://www.rstudio.com/resources/cheatsheets/

Chapter 5, Data transformation

Introduction

```
?flights
flights
```

```
## # 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     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>
```

```
#View(flights) # This line is commented out because the View function will prevent the document from kn
```

The six key dplyr functions:

filter()
arrange()
select()
mutate()
summarise()
groupby()

filter()

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

```
## # A tibble: 842 x 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 832 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>
```

```
jan1 <- filter(flights, month == 1, day == 1)
(dec25 <- filter(flights, month ==12, day == 25))
```

```
## # A tibble: 719 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013    12    25      456            500        -4      649
##  2  2013    12    25      524            515         9      805
##  3  2013    12    25      542            540         2      832
##  4  2013    12    25      546            550        -4     1022
##  5  2013    12    25      556            600        -4      730
##  6  2013    12    25      557            600        -3      743
##  7  2013    12    25      557            600        -3      818
##  8  2013    12    25      559            600        -1      855
##  9  2013    12    25      559            600        -1      849
## 10  2013    12    25      600            600         0      850
## # ... with 709 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>
```

Comparisons

Comparison operators:

```
>
>=
<
<=
!=
==
```

```
# filter(flights, month = 1) # wrong
filter(flights, month == 1) # corrected
```

```
## # A tibble: 27,004 x 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 26,994 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>
```

```
sqrt(2) ^ 2 == 2
```

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

```
1/49 * 49 == 1
```

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

```
near(sqrt(2) ^ 2, 2)
```

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

```
near(1/49 * 49, 1)
```

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

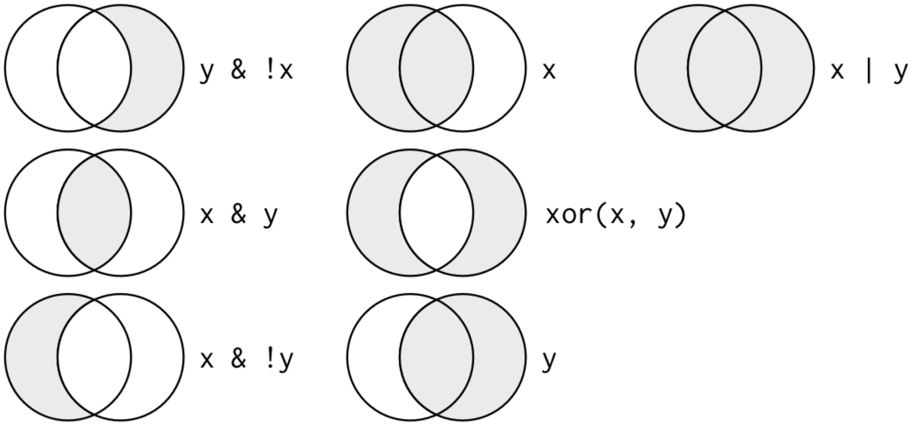

Figure 1: Boolean operations

Logical operators

```
# Find all flights that departed in November or December
filter(flights, month == 11 | month == 12)
```

```
## # A tibble: 55,403 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013    11     1        5           2359         6      352
##  2  2013    11     1       35           2250       105      123
##  3  2013    11     1      455            500        -5      641
##  4  2013    11     1      539            545        -6      856
##  5  2013    11     1      542            545        -3      831
##  6  2013    11     1      549            600       -11      912
##  7  2013    11     1      550            600       -10      705
##  8  2013    11     1      554            600        -6      659
##  9  2013    11     1      554            600        -6      826
## 10  2013    11     1      554            600        -6      749
## # ... with 55,393 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>
```

```
nov_dec <- filter(flights, month %in% c(11,12))

# Find all flights that weren't delayed (on arrival or departure) by more than two hours
filter(flights, !(arr_delay > 120 | dep_delay > 120))
```

```
## # A tibble: 316,050 x 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 316,040 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>
```

```
filter(flights, arr_delay <= 120, dep_delay <= 120)
```

```
## # A tibble: 316,050 x 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 316,040 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>
```

Missing values

```
NA > 5
```

```
## [1] NA
```

```
10 == NA
```

```
## [1] NA
```

```
NA + 10
```

```
## [1] NA
```

```
NA / 2
```

```
## [1] NA
```

```
NA == NA
```

```
## [1] NA
```

```
# Let x be Mary's age. We don't know how old she is.
x <- NA

# Let y be John's age. We don't know how old he is.
y <- NA
```

```
# Are John and Mary the same age?
x == y
```

```
## [1] NA
```

```
# We don't know!

is.na(x) # Is the value of x missing
```

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

Exercises 5.2.4

1. Find all flights that

```
1. Had an arrival delay of two or more hours
```

```
filter(flights, arr_delay >= 120)
```

```
## # A tibble: 10,200 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1      811            630       101     1047
##  2  2013     1     1      848           1835       853     1001
##  3  2013     1     1      957            733       144     1056
##  4  2013     1     1     1114            900       134     1447
##  5  2013     1     1     1505           1310       115     1638
##  6  2013     1     1     1525           1340       105     1831
##  7  2013     1     1     1549           1445        64     1912
##  8  2013     1     1     1558           1359       119     1718
##  9  2013     1     1     1732           1630        62     2028
## 10  2013     1     1     1803           1620       103     2008
## # ... with 10,190 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>
```

```
2. Flew to Houston (IAH or HOU)
```

```
filter(flights, dest %in% c("IAH", "HOU"))
```

```
## # A tibble: 9,313 x 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      623            627        -4      933
##  4  2013     1     1      728            732        -4     1041
##  5  2013     1     1      739            739         0     1104
##  6  2013     1     1      908            908         0     1228
##  7  2013     1     1     1028           1026         2     1350
##  8  2013     1     1     1044           1045        -1     1352
##  9  2013     1     1     1114            900       134     1447
## 10  2013     1     1     1205           1200         5     1503
## # ... with 9,303 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
# or:
filter(flights, dest == "IAH" | dest == "HOU")
```

```
## # A tibble: 9,313 x 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      623            627        -4      933
##  4  2013     1     1      728            732        -4     1041
##  5  2013     1     1      739            739         0     1104
##  6  2013     1     1      908            908         0     1228
##  7  2013     1     1     1028           1026         2     1350
##  8  2013     1     1     1044           1045        -1     1352
##  9  2013     1     1     1114            900       134     1447
## 10  2013     1     1     1205           1200         5     1503
## # ... with 9,303 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>
```

3. Were operated by United, American, or Delta

```r
filter(flights, carrier %in% c("AA", "UA"))
```

```
## # A tibble: 91,394 x 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      554            558        -4      740
##  5  2013     1     1      558            600        -2      753
##  6  2013     1     1      558            600        -2      924
##  7  2013     1     1      558            600        -2      923
##  8  2013     1     1      559            600        -1      941
##  9  2013     1     1      559            600        -1      854
## 10  2013     1     1      606            610        -4      858
## # ... with 91,384 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
filter(flights, carrier == "AA" | carrier == "UA")
```

```
## # A tibble: 91,394 x 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      554            558        -4      740
##  5  2013     1     1      558            600        -2      753
##  6  2013     1     1      558            600        -2      924
##  7  2013     1     1      558            600        -2      923
##  8  2013     1     1      559            600        -1      941
##  9  2013     1     1      559            600        -1      854
## 10  2013     1     1      606            610        -4      858
## # ... with 91,384 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>
```

4. Departed in summer (July, August, and September)

```
filter(flights, month %in% c(7, 8, 9))
```

```
## # A tibble: 86,326 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 86,316 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>
```

```
filter(flights, month == 7 | month == 8 | month == 9)
```

```
## # A tibble: 86,326 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 86,316 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>
```

5. Arrived more than two hours late, but didn't leave late

```
filter(flights, arr_delay > 120 & dep_delay < 1)
```

```
## # A tibble: 29 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1    27     1419           1420        -1     1754
##  2  2013    10     7     1350           1350         0     1736
##  3  2013    10     7     1357           1359        -2     1858
##  4  2013    10    16      657            700        -3     1258
##  5  2013    11     1      658            700        -2     1329
##  6  2013     3    18     1844           1847        -3       39
##  7  2013     4    17     1635           1640        -5     2049
##  8  2013     4    18      558            600        -2     1149
##  9  2013     4    18      655            700        -5     1213
## 10  2013     5    22     1827           1830        -3     2217
## # ... with 19 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>
```

6. Were delayed by at least an hour, but made up over 30 minutes in flight

```r
filter(flights, dep_delay >= 60 & arr_delay < -30)
```

```
## # A tibble: 0 x 19
## # ... with 19 variables: year <int>, month <int>, day <int>,
## #   dep_time <int>, sched_dep_time <int>, dep_delay <dbl>, arr_time <int>,
## #   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>
```

7. Departed between midnight and 6am (inclusive)

```r
filter(flights, dep_time > 0 & dep_time < 600)
```

```
## # A tibble: 8,730 x 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 8,720 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>
```

2. Another useful dplyr filtering helper is between(). What does it do? Can you use it to simplify the code needed to answer the previous challenges?

```r
?between

# 1.5 simplified
filter(flights, between(flights$month, 7, 9))
```

```
## # A tibble: 86,326 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 86,316 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>
```

```
# 1.7 simplified
filter(flights, between(flights$dep_time, 0, 600))
```

```
## # A tibble: 9,344 x 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 9,334 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>
```

3. How many flights have a missing dep_time? What other variables are missing? What might these rows represent?

```
filter(flights, is.na(dep_time))
```

```
## # A tibble: 8,255 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>
##  1  2013     1     1       NA           1630        NA       NA
##  2  2013     1     1       NA           1935        NA       NA
##  3  2013     1     1       NA           1500        NA       NA
##  4  2013     1     1       NA            600        NA       NA
##  5  2013     1     2       NA           1540        NA       NA
##  6  2013     1     2       NA           1620        NA       NA
##  7  2013     1     2       NA           1355        NA       NA
##  8  2013     1     2       NA           1420        NA       NA
##  9  2013     1     2       NA           1321        NA       NA
## 10  2013     1     2       NA           1545        NA       NA
## # ... with 8,245 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>
```

```
# or
table(is.na(flights$dep_time))
```

```
## 
##  FALSE   TRUE 
## 328521   8255
```

4. Why is NA ^ 0 not missing? Why is NA | TRUE not missing? Why is FALSE & NA not missing? Can you figure out the general rule? (NA * 0 is a tricky counterexample!)

- `NA^0` is not missing because any value to the 0th power is 1

- `NA | TRUE` is not missing because the | operand will return `TRUE` as long as one condition is true. `TRUE` is `TRUE`.

- FALSE & NA is not missng because the NA is ignored

- In operations, any value interacting with an `NA` becomes missing. Missing values are ignored in conditional exressions.

`arrange()`

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

```
## # 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     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>
```

```
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>
```

```
df <- tibble(x = c(5, 2, NA))
arrange(df, x)
```

```
## # A tibble: 3 x 1
##       x
##   <dbl>
## 1     2
## 2     5
## 3    NA
```

```
arrange(df, desc(x))
```

```
## # A tibble: 3 x 1
##       x
##   <dbl>
## 1     5
## 2     2
## 3    NA
```

Exercises 5.3.1

1. How could you use `arrange()` to sort all missing values to the start? (Hint: use `is.na()`).

```
# Example using arrival times
arrange(flights, !is.na(arr_time))
```

```
## # 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     1     2016           1930        46       NA
##   2  2013     1     1       NA           1630        NA       NA
##   3  2013     1     1       NA           1935        NA       NA
##   4  2013     1     1       NA           1500        NA       NA
##   5  2013     1     1       NA            600        NA       NA
##   6  2013     1     2     2041           2045        -4       NA
##   7  2013     1     2     2145           2129        16       NA
##   8  2013     1     2       NA           1540        NA       NA
##   9  2013     1     2       NA           1620        NA       NA
##  10  2013     1     2       NA           1355        NA       NA
## # ... 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>
```

2. Sort flights to find the most delayed flights. Find the flights that left earliest.

```
# Most delayed at departure
arrange(flights, desc(dep_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     6    27      959           1900       899     1236
##  9  2013     7    22     2257            759       898      121
## 10  2013    12     5      756           1700       896     1058
## # ... 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>
```

```
# Left earliest
arrange(flights, dep_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    12     7     2040           2123       -43       40
##  2  2013     2     3     2022           2055       -33     2240
##  3  2013    11    10     1408           1440       -32     1549
##  4  2013     1    11     1900           1930       -30     2233
##  5  2013     1    29     1703           1730       -27     1947
##  6  2013     8     9      729            755       -26     1002
##  7  2013    10    23     1907           1932       -25     2143
##  8  2013     3    30     2030           2055       -25     2213
##  9  2013     3     2     1431           1455       -24     1601
## 10  2013     5     5      934            958       -24     1225
## # ... 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>
```

3. Sort flights to find the fastest flights

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

```
## # 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     5    25     1709           1700         9     1923
##  2  2013     7     2     1558           1513        45     1745
##  3  2013     5    13     2040           2025        15     2225
##  4  2013     3    23     1914           1910         4     2045
##  5  2013     1    12     1559           1600        -1     1849
##  6  2013    11    17      650            655        -5     1059
##  7  2013     2    21     2355           2358        -3      412
##  8  2013    11    17      759            800        -1     1212
##  9  2013    11    16     2003           1925        38       17
## 10  2013    11    16     2349           2359       -10      402
## # ... 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>
```

4. Which flights travelled the longest? Which travelled the shortest?

```
# Flights with the longest travel time
arrange(flights, desc(air_time))
```

```
## # 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     3    17     1337           1335         2     1937
##  2  2013     2     6      853            900        -7     1542
##  3  2013     3    15     1001           1000         1     1551
##  4  2013     3    17     1006           1000         6     1607
##  5  2013     3    16     1001           1000         1     1544
##  6  2013     2     5      900            900         0     1555
##  7  2013    11    12      936            930         6     1630
##  8  2013     3    14      958           1000        -2     1542
##  9  2013    11    20     1006           1000         6     1639
## 10  2013     3    15     1342           1335         7     1924
## # ... 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>
```

```
# Flights with the shortest travel time
arrange(flights, air_time)
```

```
## # 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    16     1355           1315        40     1442
##  2  2013     4    13      537            527        10      622
##  3  2013    12     6      922            851        31     1021
##  4  2013     2     3     2153           2129        24     2247
##  5  2013     2     5     1303           1315       -12     1342
##  6  2013     2    12     2123           2130        -7     2211
##  7  2013     3     2     1450           1500       -10     1547
##  8  2013     3     8     2026           1935        51     2131
##  9  2013     3    18     1456           1329        87     1533
## 10  2013     3    19     2226           2145        41     2305
## # ... 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>
```

select()

```
# Select columns by name
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 all columns between year and day (inclusive)
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 all columns except those from year to day
select(flights, -(year:day))
```

```
## # A tibble: 336,776 x 16
##    dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay
##       <int>          <int>     <dbl>    <int>          <int>     <dbl>
##  1      517            515         2      830            819        11
##  2      533            529         4      850            830        20
##  3      542            540         2      923            850        33
##  4      544            545        -1     1004           1022       -18
##  5      554            600        -6      812            837       -25
##  6      554            558        -4      740            728        12
##  7      555            600        -5      913            854        19
##  8      557            600        -3      709            723       -14
##  9      557            600        -3      838            846        -8
## 10      558            600        -2      753            745         8
## # ... with 336,766 more rows, and 10 more variables: carrier <chr>,
## #   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

Helper functions to use with `select()`

- `starts_with("abc")`: matches names that begin with "abc".
- `ends_with("xyz")`: matches names that end with "xyz".
- `contains("ijk")`: matches names that contain "ijk".
- `matches("(.)\\1")`: selects variables that match a regular expression. This one matches any variables that contain repeated characters. You'll learn more about regular expressions in strings.
- `num_range("x", 1:3)`: matches x1, x2 and x3.

rename()

```
rename(flights, tail_num = tailnum)
```

```
## # 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     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_num <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

everything()

```
# Move handful of variables to the start of the data frame
select(flights, time_hour, air_time, everything())
```

```
## # A tibble: 336,776 x 19
##              time_hour air_time  year month   day dep_time sched_dep_time
##                 <dttm>    <dbl> <int> <int> <int>    <int>          <int>
##  1 2013-01-01 05:00:00      227  2013     1     1      517            515
##  2 2013-01-01 05:00:00      227  2013     1     1      533            529
##  3 2013-01-01 05:00:00      160  2013     1     1      542            540
##  4 2013-01-01 05:00:00      183  2013     1     1      544            545
##  5 2013-01-01 06:00:00      116  2013     1     1      554            600
##  6 2013-01-01 05:00:00      150  2013     1     1      554            558
##  7 2013-01-01 06:00:00      158  2013     1     1      555             600
##  8 2013-01-01 06:00:00       53  2013     1     1      557             600
##  9 2013-01-01 06:00:00      140  2013     1     1      557             600
## 10 2013-01-01 06:00:00      138  2013     1     1      558             600
## # ... with 336,766 more rows, and 12 more variables: dep_delay <dbl>,
## #   arr_time <int>, sched_arr_time <int>, arr_delay <dbl>, carrier <chr>,
## #   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, distance <dbl>,
## #   hour <dbl>, minute <dbl>
```

Exercises 5.4.1

1. Brainstorm as many ways as possible to select `dep_time`, `dep_delay`, `arr_time`, and `arr_delay` from `flights`.

```r
# select
select(flights, dep_time, dep_delay, arr_time, arr_delay)
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_delay arr_time arr_delay
##       <int>     <dbl>    <int>     <dbl>
##  1      517         2      830        11
##  2      533         4      850        20
##  3      542         2      923        33
##  4      544        -1     1004       -18
##  5      554        -6      812       -25
##  6      554        -4      740        12
##  7      555        -5      913        19
##  8      557        -3      709       -14
##  9      557        -3      838        -8
## 10      558        -2      753         8
## # ... with 336,766 more rows
```

```r
# starts_with
select(flights, starts_with("dep_"), starts_with("arr_"))
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_delay arr_time arr_delay
##       <int>     <dbl>    <int>     <dbl>
##  1      517         2      830        11
##  2      533         4      850        20
##  3      542         2      923        33
##  4      544        -1     1004       -18
##  5      554        -6      812       -25
##  6      554        -4      740        12
##  7      555        -5      913        19
##  8      557        -3      709       -14
##  9      557        -3      838        -8
## 10      558        -2      753         8
## # ... with 336,766 more rows
```

```r
# ends_with
select(flights, ends_with("time"), ends_with("delay"), -(starts_with("sched")), -(starts_with("air")))
```

```
## # A tibble: 336,776 x 4
##    dep_time arr_time dep_delay arr_delay
##       <int>    <int>     <dbl>     <dbl>
##  1      517      830         2        11
##  2      533      850         4        20
##  3      542      923         2        33
##  4      544     1004        -1       -18
##  5      554      812        -6       -25
##  6      554      740        -4        12
##  7      555      913        -5        19
##  8      557      709        -3       -14
##  9      557      838        -3        -8
## 10      558      753        -2         8
## # ... with 336,766 more rows
```

```
# contains
select(flights, contains("dep"), contains("arr_"), -(contains("sched")))
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_delay arr_time arr_delay
##       <int>     <dbl>    <int>     <dbl>
##  1      517         2      830        11
##  2      533         4      850        20
##  3      542         2      923        33
##  4      544        -1     1004       -18
##  5      554        -6      812       -25
##  6      554        -4      740        12
##  7      555        -5      913        19
##  8      557        -3      709       -14
##  9      557        -3      838        -8
## 10      558        -2      753         8
## # ... with 336,766 more rows
```

2. What happens if you include the name of a variable multiple times in a `select()` call?

The variable is included only once in the data frame

```
select(flights, arr_delay, dep_delay, dep_delay)
```

```
## # A tibble: 336,776 x 2
##    arr_delay dep_delay
##        <dbl>     <dbl>
##  1        11         2
##  2        20         4
##  3        33         2
##  4       -18        -1
##  5       -25        -6
##  6        12        -4
##  7        19        -5
##  8       -14        -3
##  9        -8        -3
## 10         8        -2
## # ... with 336,766 more rows
```

3. What does the `one_of()` function do? Why might it be helpful in conjunction with this vector?

```
vars <- c("year", "month", "day", "dep_delay", "arr_delay")
```

`one_of()` allows one to select variables that match those in a character vector.

```
vars <- c("year", "month", "day", "dep_delay", "arr_delay")
select(flights, one_of(vars))
```

```
## # A tibble: 336,776 x 5
##     year month   day dep_delay arr_delay
##    <int> <int> <int>     <dbl>     <dbl>
##  1  2013     1     1         2        11
##  2  2013     1     1         4        20
##  3  2013     1     1         2        33
##  4  2013     1     1        -1       -18
##  5  2013     1     1        -6       -25
##  6  2013     1     1        -4        12
##  7  2013     1     1        -5        19
##  8  2013     1     1        -3       -14
##  9  2013     1     1        -3        -8
## 10  2013     1     1        -2         8
## # ... with 336,766 more rows
```

4. Does the result of running the following code surprise you? How do the select helpers deal with case by default? How can you change that default?

```
select(flights, contains("TIME"))
```

The `select()` helpers are not case sensitive by default and will select all variables that contain the character string `time` regardless of case (e.g. “TIME”, “Time”, time“,”TiMe“, etc.). To change the default, set `ignore.case = FALSE` within the helper function.

```
# Default
select(flights, contains("TIME"))
```

```
## # A tibble: 336,776 x 6
##    dep_time sched_dep_time arr_time sched_arr_time air_time
##       <int>          <int>    <int>          <int>    <dbl>
##  1      517            515      830            819      227
##  2      533            529      850            830      227
##  3      542            540      923            850      160
##  4      544            545     1004           1022      183
##  5      554            600      812            837      116
##  6      554            558      740            728      150
##  7      555            600      913            854      158
##  8      557            600      709            723       53
##  9      557            600      838            846      140
## 10      558            600      753            745      138
## # ... with 336,766 more rows, and 1 more variables: time_hour <dttm>
```

```
# Set ignore.case = FALSE
select(flights, contains("TIME", ignore.case = FALSE))
```

```
## # A tibble: 336,776 x 0
```

`mutate()`

```r
# Create a narrower dataset
flights_sml <- select(flights, year:day, ends_with("delay"), distance, air_time)

# Create new columns
mutate(flights_sml,
       gain = arr_delay - dep_delay,
       speed = distance / air_time * 60,
       hours = air_time / 60,
       gain_per_hour = gain / hours)
```

```
## # A tibble: 336,776 x 11
##     year month   day dep_delay arr_delay distance air_time  gain    speed
##    <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl> <dbl>    <dbl>
##  1  2013     1     1         2        11     1400      227     9 370.0441
##  2  2013     1     1         4        20     1416      227    16 374.2731
##  3  2013     1     1         2        33     1089      160    31 408.3750
##  4  2013     1     1        -1       -18     1576      183   -17 516.7213
##  5  2013     1     1        -6       -25      762      116   -19 394.1379
##  6  2013     1     1        -4        12      719      150    16 287.6000
##  7  2013     1     1        -5        19     1065      158    24 404.4304
##  8  2013     1     1        -3       -14      229       53   -11 259.2453
##  9  2013     1     1        -3        -8      944      140    -5 404.5714
## 10  2013     1     1        -2         8      733      138    10 318.6957
## # ... with 336,766 more rows, and 2 more variables: hours <dbl>,
## #   gain_per_hour <dbl>
```

`transmute()`

Only keep the new variables.

```r
transmute(flights,
          gain = arr_delay - dep_delay,
          hours = air_time / 60,
          gain_per_hour = gain / hours)
```

```
## # A tibble: 336,776 x 3
##     gain     hours gain_per_hour
##    <dbl>     <dbl>         <dbl>
##  1     9 3.7833333      2.378855
##  2    16 3.7833333      4.229075
##  3    31 2.6666667     11.625000
##  4   -17 3.0500000     -5.573770
##  5   -19 1.9333333     -9.827586
##  6    16 2.5000000      6.400000
##  7    24 2.6333333      9.113924
##  8   -11 0.8833333    -12.452830
##  9    -5 2.3333333     -2.142857
## 10    10 2.3000000      4.347826
## # ... with 336,766 more rows
```

Useful creation functions

1. Modular arithmatic, including:

- Division

```
100 / 3
```

```
## [1] 33.33333
```

```
- Integer division
```

```
100 %/% 3
```

```
## [1] 33
```

```
- Remainder division
```

```
100 %% 3
```

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

```
# Breaking up is(n't) hard to do
transmute(flights,
          dep_time,
          hour = dep_time %/%100,
          minute = dep_time %% 100)
```

```
## # A tibble: 336,776 x 3
##    dep_time  hour minute
##       <int> <dbl>  <dbl>
##  1      517     5     17
##  2      533     5     33
##  3      542     5     42
##  4      544     5     44
##  5      554     5     54
##  6      554     5     54
##  7      555     5     55
##  8      557     5     57
##  9      557     5     57
## 10      558     5     58
## # ... with 336,766 more rows
```

2. logs
3. Offsets (lead & lag)

```
(x <-  1:10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
lag(x)
```

```
##  [1] NA  1  2  3  4  5  6  7  8  9
```

```
lead(x)
```

```
##  [1]  2  3  4  5  6  7  8  9 10 NA
```

4. Cumulative and rolling aggregates
    - `cumsum()`
    - `cumprod()`
    - `cummin()`
    - `cummax()`

- cummean()

```
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
cumsum(x)
```

```
##  [1]  1  3  6 10 15 21 28 36 45 55
```

```
cummean(x)
```

```
##  [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
```

5. Logical comparisons
6. Ranking

```
(y <- c(1, 2, 2, NA, 3, 4))
```

```
## [1]  1  2  2 NA  3  4
```

```
min_rank(y) # Default: smallest values get small ranks
```

```
## [1]  1  2  2 NA  4  5
```

```
min_rank(desc(y)) # Give the largest values the smallest ranks
```

```
## [1]  5  3  3 NA  2  1
```

```
row_number(y)
```

```
## [1]  1  2  3 NA  4  5
```

```
dense_rank(y)
```

```
## [1]  1  2  2 NA  3  4
```

```
percent_rank(y)
```

```
## [1] 0.00 0.25 0.25   NA 0.75 1.00
```

```
cume_dist(y)
```

```
## [1] 0.2 0.6 0.6  NA 0.8 1.0
```

Exercises 5.5.2

1. Currently dep_time and sched_dep_time are convenient to look at, but hard to compute with because they're not really continuous numbers. Convert them to a more convenient representation of number of minutes since midnight.

```
transmute(flights,
          dep_time,
          dep_minutes = dep_time %/% 100 * 60 + dep_time %% 100,
          sched_dep_time,
          sched_dep_minutes = sched_dep_time %/% 100 * 60 + sched_dep_time %% 100)
```

```
## # A tibble: 336,776 x 4
##    dep_time dep_minutes sched_dep_time sched_dep_minutes
##       <int>       <dbl>          <int>             <dbl>
##  1      517         317            515               315
##  2      533         333            529               329
##  3      542         342            540               340
##  4      544         344            545               345
##  5      554         354            600               360
##  6      554         354            558               358
##  7      555         355            600               360
##  8      557         357            600               360
##  9      557         357            600               360
## 10      558         358            600               360
## # ... with 336,766 more rows
```

Alternatively, write a function to convert time to minutes as seen here: https://jrnold.github.io/e4qf/data-transformation.html#mutate

```
time2mins <- function(x) {
  x %/% 100 * 60 + x %% 100
}

transmute(flights,
       dep_minutes = time2mins(dep_time),
       sched_dep_minutes = time2mins(sched_dep_time))
```

```
## # A tibble: 336,776 x 2
##    dep_minutes sched_dep_minutes
##          <dbl>             <dbl>
##  1         317               315
##  2         333               329
##  3         342               340
##  4         344               345
##  5         354               360
##  6         354               358
##  7         355               360
##  8         357               360
##  9         357               360
## 10         358               360
## # ... with 336,766 more rows
```

2. Compare air_time with arr_time - dep_time. What do you expect to see? What do you see? What do you need to do to fix it?

What we want is for `air_time` to equal `arr_time - dep_time`. However, since the times are not continuous values, we will get a meaningless value. We must first convert `arr_time` and `dep_time` to a format such as minuted that represents the elapsed time on a continuous scale. However, since the plane may depart from and arrive in different time zones, and since arrival and departure times are reported in local time, some of the calculated values will be different than the `air_time` values.

```
# Without converting to minutes, note that the columns do not match
transmute(flights,
          air_time,
          calculated_airtime = arr_time - dep_time)
```

```
## # A tibble: 336,776 x 2
##    air_time calculated_airtime
##       <dbl>              <int>
##  1      227                313
##  2      227                317
##  3      160                381
##  4      183                460
##  5      116                258
##  6      150                186
##  7      158                358
##  8       53                152
##  9      140                281
## 10      138                195
## # ... with 336,766 more rows
```

```
# Now that we have converted the time format, the calculation will be correct. But wait, some still don
(flights2 <- transmute(flights,
          arr_time = time2mins(arr_time),
          dep_time = time2mins(dep_time),
          air_time,
          airtime_new = arr_time - dep_time))
```

```
## # A tibble: 336,776 x 4
##    arr_time dep_time air_time airtime_new
##       <dbl>    <dbl>    <dbl>       <dbl>
##  1      510      317      227         193
##  2      530      333      227         197
##  3      563      342      160         221
##  4      604      344      183         260
##  5      492      354      116         138
##  6      460      354      150         106
##  7      553      355      158         198
##  8      429      357       53          72
##  9      518      357      140         161
## 10      473      358      138         115
## # ... with 336,766 more rows
```

To fix this, let's simply view those flights that did not change time zones.

```
flights2 %>% filter(air_time == airtime_new)
```

```
## # A tibble: 196 x 4
##    arr_time dep_time air_time airtime_new
##       <dbl>    <dbl>    <dbl>       <dbl>
##  1      821      710      111         111
##  2     1338     1187      151         151
##  3      554      411      143         143
##  4      666      427      239         239
##  5      731      583      148         148
##  6     1106      979      127         127
##  7      754      616      138         138
##  8      974      859      115         115
##  9     1427     1315      112         112
## 10     1245     1133      112         112
## # ... with 186 more rows
```

3. Compare dep_time, sched_dep_time, and dep_delay. How would you expect those three numbers to be related?

4. Find the 10 most delayed flights using a ranking function. How do you want to handle ties? Carefully read the documentation for min_rank().

5. **What does 1:3 + 1:10 return? Why?**

6. **What trigonometric functions does R provide?**