BADM 371 Intro to Data Analytics

BADM 371

2022-02-24

Contents

Chapter 1

Introduction

This is a book written which contains all the materials and lessons for **BADM 371 Intro to Data Analytics**. If you miss a day, want to review something we covered in class, or for any reason want to look for a worksheet, this is where you can go.

Each chapter contains a different topic we will cover during the semester. Some larger topics are split into two chapters to make accessing the materials a little more intuitive.

This book is updated automatically with any changes made to the documents during the semester, so if at any point you are told there was a change in the assignment, you can come here to get the updated version.

Also, this book has benefited greatly from lots of free, readily available resources posted on the web and we leverage these extensively. I would encourage you to review these resources in your analytics journey. Some that we specifically use with great frequency are these (and I say loud **THANK YOU** to the authors!):

- R for Data Science
- An Introduction to Statistical Learning with Applications in R
- R: A self-learn tutorial
- Data Science in a Box
- stackoverflow.com, for example

Chapter 2

DPLYR

2.1 Introduction

For more help **PLEASE** check out Introduction to dplyr introducing the key functionality of the dplyr package.

> Your life is about to change. For the better, even.

2.2 A Neat Resource

- RStudio's Data Wrangling Cheat Sheet for dplyr and tidyr

2.3 Single table verbs

`dplyr` aims to provide a function for each basic verb of data manipulation. These verbs can be organised into three categories based on the component of the dataset that they work with:

Rows:

- `filter()` chooses rows based on column values.
- `slice()` chooses rows based on location.
- `arrange()` changes the order of the rows.

Columns:

- `select()` changes whether or not a column is included.
- `rename()` changes the name of columns. `mutate()` changes the values of columns and creates new columns.
- `relocate()` changes the order of the columns. Groups of rows:
- `summarise()` collapses a group into a single row. It's not that useful until we learn the `group_by()` verb below.

2.4 The pipe

All of the `dplyr` functions take a data frame (or tibble) as the first argument. Rather than forcing the user to either save intermediate objects or nest functions, dplyr provides the `%>%` operator from magrittr. x %>% f(y) turns into f(x, y) so the result from one step is then "piped" into the next step. You can use the pipe to rewrite multiple operations that you can read left-to-right, top-to-bottom (reading the pipe operator as "then").

2.5 Loading dplyr and the nycflights13 dataset

```
# load packages
suppressMessages(library(dplyr))
library(nycflights13)

# print the flights dataset from nycflights13
head(flights)
#> # A tibble: 6 x 19
#>    year month   day dep_time sched_dep_time dep_delay
#>   <int> <int> <int>    <int>          <int>     <dbl>
#> 1  2013     1     1      517            515         2
#> 2  2013     1     1      533            529         4
#> 3  2013     1     1      542            540         2
#> 4  2013     1     1      544            545        -1
#> 5  2013     1     1      554            600        -6
#> 6  2013     1     1      554            558        -4
#> # ... with 13 more variables: 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>
```

2.6 Choosing columns: select, rename

```
# besides just using select() to pick columns...
flights %>% select(carrier, flight)
#> # A tibble: 336,776 x 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 x 17
#>     year dep_time sched_dep_time dep_delay arr_time
#>    <int>    <int>          <int>     <dbl>    <int>
#>  1  2013      517            515         2      830
#>  2  2013      533            529         4      850
#>  3  2013      542            540         2      923
#>  4  2013      544            545        -1     1004
#>  5  2013      554            600        -6      812
#>  6  2013      554            558        -4      740
#>  7  2013      555            600        -5      913
#>  8  2013      557            600        -3      709
#>  9  2013      557            600        -3      838
#> 10  2013      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>
```

```
# 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 x 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 x 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 x 19
#>     year month   day dep_time sched_dep_time dep_delay
#>    <int> <int> <int>    <int>          <int>     <dbl>
#>  1  2013     1     1      517            515         2
#>  2  2013     1     1      533            529         4
#>  3  2013     1     1      542            540         2
```

```
#>  4  2013     1     1      544            545        -1
#>  5  2013     1     1      554            600        -6
#>  6  2013     1     1      554            558        -4
#>  7  2013     1     1      555            600        -5
#>  8  2013     1     1      557            600        -3
#>  9  2013     1     1      557            600        -3
#> 10  2013     1     1      558            600        -2
#> # ... with 336,766 more rows, and 13 more variables:
#> #   arr_time <int>, 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>
```

2.7 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 x 19
#>     year month   day dep_time sched_dep_time dep_delay
#>    <int> <int> <int>    <int>          <int>     <dbl>
#>  1  2013     1     1      600            600         0
#>  2  2013     1     1      600            600         0
#>  3  2013     1     1      601            600         1
#>  4  2013     1     1      602            610        -8
#>  5  2013     1     1      602            605        -3
#>  6  2013     1     2      600            600         0
#>  7  2013     1     2      600            605        -5
#>  8  2013     1     2      600            600         0
#>  9  2013     1     2      600            600         0
#> 10  2013     1     2      600            600         0
#> # ... with 2,450 more rows, and 13 more variables:
#> #   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>
```

```
# 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))
```

```
# slice() filters rows by position
flights %>% slice(1000:1005)
#> # A tibble: 6 x 19
#>    year month   day dep_time sched_dep_time dep_delay
#>   <int> <int> <int>    <int>          <int>     <dbl>
#> 1  2013     1     2      809            810        -1
#> 2  2013     1     2      810            800        10
#> 3  2013     1     2      811            815        -4
#> 4  2013     1     2      811            815        -4
#> 5  2013     1     2      811            820        -9
#> 6  2013     1     2      815            815         0
#> # ... with 13 more variables: 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>

# keep the first three rows within each group
flights %>% group_by(month, day) %>% slice(1:3)
#> # A tibble: 1,095 x 19
#> # Groups:   month, day [365]
#>     year month   day dep_time sched_dep_time dep_delay
#>    <int> <int> <int>    <int>          <int>     <dbl>
#>  1  2013     1     1      517            515         2
#>  2  2013     1     1      533            529         4
#>  3  2013     1     1      542            540         2
#>  4  2013     1     2       42           2359        43
#>  5  2013     1     2      126           2250       156
#>  6  2013     1     2      458            500        -2
#>  7  2013     1     3       32           2359        33
#>  8  2013     1     3       50           2145       185
#>  9  2013     1     3      235           2359       156
#> 10  2013     1     4       25           2359        26
#> # ... with 1,085 more rows, and 13 more variables:
#> #   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>

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

```
#> # A tibble: 1,095 x 19
#> # Groups:   month, day [365]
#>     year month   day dep_time sched_dep_time dep_delay
#>    <int> <int> <int>    <int>          <int>     <dbl>
#>  1  2013     1     1     2058           2100        -2
#>  2  2013     1     1     1945           1940         5
#>  3  2013     1     1     1615           1602        13
#>  4  2013     1     2     2041           1942        59
#>  5  2013     1     2     1331           1201        90
#>  6  2013     1     2     1156           1200        -4
#>  7  2013     1     3     2030           2029         1
#>  8  2013     1     3      835            835         0
#>  9  2013     1     3      622            630        -8
#> 10  2013     1     4     1033           1000        33
#> # ... with 1,085 more rows, and 13 more variables:
#> #   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>

# keep three rows from each group with the top dep_delay
flights %>% group_by(month, day) %>% top_n(3, dep_delay)
#> # A tibble: 1,108 x 19
#> # Groups:   month, day [365]
#>     year month   day dep_time sched_dep_time dep_delay
#>    <int> <int> <int>    <int>          <int>     <dbl>
#>  1  2013     1     1      848           1835       853
#>  2  2013     1     1     1815           1325       290
#>  3  2013     1     1     2343           1724       379
#>  4  2013     1     2     1412            838       334
#>  5  2013     1     2     1607           1030       337
#>  6  2013     1     2     2131           1512       379
#>  7  2013     1     3     2008           1540       268
#>  8  2013     1     3     2012           1600       252
#>  9  2013     1     3     2056           1605       291
#> 10  2013     1     4     2058           1730       208
#> # ... with 1,098 more rows, and 13 more variables:
#> #   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>

# also sort by dep_delay within each group
```

```
flights %>% group_by(month, day) %>% top_n(3, dep_delay) %>% arrange(desc(dep_delay))
#> # A tibble: 1,108 x 19
#> # Groups:   month, day [365]
#>     year month   day dep_time sched_dep_time dep_delay
#>    <int> <int> <int>    <int>          <int>     <dbl>
#>  1  2013     1     9      641            900      1301
#>  2  2013     6    15     1432           1935      1137
#>  3  2013     1    10     1121           1635      1126
#>  4  2013     9    20     1139           1845      1014
#>  5  2013     7    22      845           1600      1005
#>  6  2013     4    10     1100           1900       960
#>  7  2013     3    17     2321            810       911
#>  8  2013     6    27      959           1900       899
#>  9  2013     7    22     2257            759       898
#> 10  2013    12     5      756           1700       896
#> # ... with 1,098 more rows, and 13 more variables:
#> #   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>
```

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

```
# 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
flights %>% select(origin, dest) %>% distinct
```

2.8 Excercies

Using the nycflights13 dataset and the dplyr package, answer these questions. Some answers are given in square brackets for you to check your answers.

1. How many flights in Sept were late departing flights? [7815]
2. How many flights in Sept were late departing flights that originated at JFK airport? [2649]
3. How many flights in Sept were late departing flights with an origin of JFK airport and had an destination of anywhere except MIA? [2572]
4. Which carrier had the most flights in this data set? [UA with 58665]
5. Which destination had the most flights in this data set? [ORD with 17283]
6. Which destination had the most flights with departure delays of greater than 60 minutes in this data set? [ORD with 1480]
7. What was the longest arrival delay in this dataset? [1272]
8. Which carrier in September had the most late departing flights? [UA with 1559]
9. Create a variable called total.annoyance which arrival delay plus the departure delay for each flight.
10. Which carrier with more than 10 flights in September had greatest % late departing flights?

2.9 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 x 20
#>     year month   day dep_time sched_dep_time dep_delay
#>    <int> <int> <int>    <int>          <int>     <dbl>
#>  1  2013     1     1      517            515         2
#>  2  2013     1     1      533            529         4
#>  3  2013     1     1      542            540         2
#>  4  2013     1     1      544            545        -1
#>  5  2013     1     1      554            600        -6
#>  6  2013     1     1      554            558        -4
#>  7  2013     1     1      555            600        -5
#>  8  2013     1     1      557            600        -3
#>  9  2013     1     1      557            600        -3
#> 10  2013     1     1      558            600        -2
#> # ... with 336,766 more rows, and 14 more variables:
#> #   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>, speed <dbl>

# transmute() only keeps the new variables
flights %>% transmute(speed = distance/air_time*60)
#> # A tibble: 336,776 x 1
#>    speed
#>    <dbl>
#>  1  370.
#>  2  374.
#>  3  408.
#>  4  517.
#>  5  394.
#>  6  288.
#>  7  404.
#>  8  259.
#>  9  405.
#> 10  319.
#> # ... with 336,766 more rows
```

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

# add_rownames() turns row names into an explicit variable
mtcars %>% add_rownames("model") %>% head()
#> Warning: `add_rownames()` was deprecated in dplyr 1.0.0.
#> Please use `tibble::rownames_to_column()` instead.
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated
```

```
#> # A tibble: 6 x 12
#>   model        mpg   cyl  disp    hp  drat    wt  qsec    vs
#>   <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1 Mazda RX4   21       6   160   110  3.9   2.62  16.5     0
#> 2 Mazda RX4~  21       6   160   110  3.9   2.88  17.0     0
#> 3 Datsun 710  22.8     4   108    93  3.85  2.32  18.6     1
#> 4 Hornet 4 ~  21.4     6   258   110  3.08  3.22  19.4     1
#> 5 Hornet Sp~  18.7     8   360   175  3.15  3.44  17.0     0
#> 6 Valiant     18.1     6   225   105  2.76  3.46  20.2     1
#> # ... with 3 more variables: am <dbl>, gear <dbl>,
#> #   carb <dbl>

# side note: dplyr no longer prints row names (ever) for local data frames
mtcars %>% tbl_df()
#> Warning: `tbl_df()` was deprecated in dplyr 1.0.0.
#> Please use `tibble::as_tibble()` instead.
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
#> # A tibble: 32 x 11
#>      mpg   cyl  disp    hp  drat    wt  qsec    vs    am
#>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#>  1  21       6  160    110  3.9   2.62  16.5     0     1
#>  2  21       6  160    110  3.9   2.88  17.0     0     1
#>  3  22.8     4  108     93  3.85  2.32  18.6     1     1
#>  4  21.4     6  258    110  3.08  3.22  19.4     1     0
#>  5  18.7     8  360    175  3.15  3.44  17.0     0     0
#>  6  18.1     6  225    105  2.76  3.46  20.2     1     0
#>  7  14.3     8  360    245  3.21  3.57  15.8     0     0
#>  8  24.4     4  147.    62  3.69  3.19  20       1     0
#>  9  22.8     4  141.    95  3.92  3.15  22.9     1     0
#> 10  19.2     6  168.   123  3.92  3.44  18.3     1     0
#> # ... with 22 more rows, and 2 more variables: gear <dbl>,
#> #   carb <dbl>
```

Chapter 3

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

```
# summarise() can be used to count the number of rows in each group
flights %>% group_by(month) %>% summarise(cnt = n())
#> # A tibble: 12 x 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
```

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

```
# you can sort by the count
flights %>% group_by(month) %>% summarise(cnt = n()) %>% arrange(desc(cnt))
#> # A tibble: 12 x 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 x 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
#> [10] 28889 27268 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
#> `summarise()` has grouped output by 'month'. You can override using the `.groups` argument.
#> # A tibble: 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(des
#> `summarise()` has grouped output by 'month'. You can override using the `.groups` a
#> # A tibble: 365 x 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
```

Chapter 4

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()
#> Warning: `data_frame()` was deprecated in tibble 1.1.0.
#> Please use `tibble()` instead.
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was generated.
#> Rows: 6
#> Columns: 4
#> $ a     <int> 1, 2, 3, 4, 5, 6
#> $ b     <dbl> 2, 4, 6, 8, 10, 12
#> $ c     <chr> "string", "string", "string", "string", "str~
#> $ `d+e` <dbl> 1, 1, 1, 1, 1, 1

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

```
#> Columns: 3
#> $ a   <int> 1, 2, 3, 4, 5, 6
#> $ c   <chr> "string", "string", "string", "string", "strin~
#> $ d.e <dbl> 1, 1, 1, 1, 1, 1
```

Chapter 5

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 x 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 x 2
#>   color  size
#>   <chr>  <chr>
#> 1 green  S
#> 2 yellow M
#> 3 pink   L

# only include observations found in both "a" and "b" (automatically joins on variables that appe
inner_join(a, b)
#> Joining, by = "color"
#> # A tibble: 2 x 3
#>   color    num size
```

```
#>   <chr>  <int> <chr>
#> 1 green      1 S
#> 2 yellow     2 M

# include observations found in either "a" or "b"
full_join(a, b)
#> Joining, by = "color"
#> # A tibble: 4 x 3
#>   color    num size
#>   <chr>  <int> <chr>
#> 1 green      1 S
#> 2 yellow     2 M
#> 3 red        3 <NA>
#> 4 pink      NA L

# include all observations found in "a"
left_join(a, b)
#> Joining, by = "color"
#> # A tibble: 3 x 3
#>   color    num size
#>   <chr>  <int> <chr>
#> 1 green      1 S
#> 2 yellow     2 M
#> 3 red        3 <NA>

# include all observations found in "b"
right_join(a, b)
#> Joining, by = "color"
#> # A tibble: 3 x 3
#>   color    num size
#>   <chr>  <int> <chr>
#> 1 green      1 S
#> 2 yellow     2 M
#> 3 pink      NA L

# 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 x 3
#>   color  size    num
#>   <chr>  <chr> <int>
#> 1 green  S         1
#> 2 yellow M         2
#> 3 pink   L        NA
```

```r
# filter "a" to only show observations that match "b"
semi_join(a, b)
#> Joining, by = "color"
#> # A tibble: 2 x 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 x 2
#>   color   num
#>   <chr> <int>
#> 1 red       3
```

```r
# 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 x 3
#>   color    num size
#>   <chr>  <int> <chr>
#> 1 green      1 S
#> 2 yellow     2 M
```

Chapter 6

Viewing more output: print, View

```
# specify that you want to see more rows
flights %>% print(n = 15)
#> # A tibble: 336,776 x 19
#>     year month   day dep_time sched_dep_time dep_delay
#>    <int> <int> <int>    <int>          <int>     <dbl>
#>  1  2013     1     1      517            515         2
#>  2  2013     1     1      533            529         4
#>  3  2013     1     1      542            540         2
#>  4  2013     1     1      544            545        -1
#>  5  2013     1     1      554            600        -6
#>  6  2013     1     1      554            558        -4
#>  7  2013     1     1      555            600        -5
#>  8  2013     1     1      557            600        -3
#>  9  2013     1     1      557            600        -3
#> 10  2013     1     1      558            600        -2
#> 11  2013     1     1      558            600        -2
#> 12  2013     1     1      558            600        -2
#> 13  2013     1     1      558            600        -2
#> 14  2013     1     1      558            600        -2
#> 15  2013     1     1      559            600        -1
#> # ... with 336,761 more rows, and 13 more variables:
#> #   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>
```

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

```
#>  2 2013-01-01 05:00:00
#>  3 2013-01-01 05:00:00
#>  4 2013-01-01 05:00:00
#>  5 2013-01-01 06:00:00
#>  6 2013-01-01 05:00:00
#>  7 2013-01-01 06:00:00
#>  8 2013-01-01 06:00:00
#>  9 2013-01-01 06:00:00
#> 10 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
options(dplyr.width = Inf, dplyr.print_min = 6)

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

Chapter 7

DPLYR Excercies

Using the nycflights13 dataset and the dplyr package, answer these questions. Some answers are given in square brackets for you to check your answers.

1. How many flights in Sept were late departing flights? [7815]
2. How many flights in Sept were late departing flights that originated at JFK airport? [2649]
3. How many flights in Sept were late departing flights with an origin of JFK airport and had an destination of anywhere except MIA? [2572]
4. Which carrier had the most flights in this data set? [UA with 58665]
5. Which destination had the most flights in this data set? [ORD with 17283]
6. Which destination had the most flights with departure delays of greater than 60 minutes in this data set? [ORD with 1480]
7. What was the longest arrival delay in this dataset? [1272]
8. Which carrier in September had the most late departing flights? [UA with 1559]
9. Create a variable called total.annoyance which arrival delay plus the departure delay for each flight.
10. Which carrier with more than 10 flights in September had greatest % late departing flights?

Chapter 8

More DPLYR

8.1 Introduction

For more help **PLEASE** check out Introduction to dplyr introducing the key functionality of the dplyr package.

> Your life is about to change. **For the better, even.**

8.2 Single table verbs

`dplyr` aims to provide a function for each basic verb of data manipulation. These verbs can be organised into three categories based on the component of the dataset that they work with:

Rows:

- `filter()` chooses rows based on column values.
- `slice()` chooses rows based on location.
- `arrange()` changes the order of the rows.

Columns:

- `select()` changes whether or not a column is included.
- `rename()` changes the name of columns.
- `mutate()` changes the values of columns and creates new columns.
- `relocate()` changes the order of the columns. Groups of rows:
- `summarise()` collapses a group into a single row. Itâ s not that useful until we learn `group_by()`
- `group_by()` usually works with `summarise()`

8.3 More with the pipe

All of the `dplyr` functions take a data frame (or tibble) as the first argument. Rather than forcing the user to either save intermediate objects or nest functions, dplyr provides the `%>%` operator from `magrittr`. One step is then â pipedâ into the next step. You can use the pipe to rewrite multiple operations that you can read left-to-right, top-to-bottom (reading the pipe operator as â thenâ).

What is this: `%>%`?

8.4 Loading `dplyr` and the `starwars` dataset

```
# You should already have done this but you'll need it
install.packages("dplyr")
```

```
library(dplyr)

starwars %>%
  filter(species == "Droid")


starwars %>%
  select(name, ends_with("color"))


starwars %>%
  mutate(name, bmi = mass / ((height / 100)  ^ 2)) %>%
  select(name:mass, bmi)


starwars %>%
  arrange(desc(mass))

starwars %>%
  group_by(species) %>%
  summarise(
    n = n(),
    mass = mean(mass, na.rm = TRUE)
  ) %>%
  filter(
    n > 1,
```

```
    mass > 50
  )
```

8.5 starwars Excercises

Please use the `starwars` dataset from the `dplyr` package to answer the following questions:

1. How may humans are in this dataset?
2. How many characters are taller than 89 cm?
3. How many characters are taller than 37 inches?
4. How many characters are taller than 37 inches and weigh more than 55 pounds?
5. How many characters are not human or droid?
6. How many characters are not human or droid and are taller than 47 inches?
7. Which species has the most individuals included in this data set?
8. Which species has the tallest individuals on average?
9. What is the tallest individual for each species?
10. Calculate the BMI for each individual and determine which individual has the highest BMI. Use the formula `bmi = mass/((height/100)^2)` to calculate bmi.
11. Which homeworld has the most individuals included in this data set?
12. Which homeworld has the tallest individuals on average?
13. What is the tallest individual for each eye color?