



SM-2302 Software for Mathematicians

R3: The tidyverse *[handout version]*

Dr. Haziq Jamil

Mathematical Sciences, Faculty of Science, UBD

<https://haziqj.ml>

Semester I 2022/23

Overview

Tidy data

Tibbles

Pipeline

Data manipulation

- `filter`

- `slice`

- `select`

- `relocate`

- `pull`

- `arrange`

- Others

Overview

Reshaping data (Wide vs. Long)

Tidy data

country	year	cases	population
Afghanistan	1999	16645	15557071
Afghanistan	2000	16666	20495360
Brazil	1999	30737	172006362
Brazil	2000	80488	174004898
China	1999	210258	1272015272
China	2000	210766	1280023583

variables

country	year	cases	population
Afghanistan	1999	16645	15557071
Afghanistan	2000	16666	20495360
Brazil	1999	30737	172006362
Brazil	2000	80488	174004898
China	1999	210258	1272015272
China	2000	210766	1280023583

observations

country	year	cases	population
Afghanistan	1999	16645	15557071
Afghanistan	2000	16666	20495360
Brazil	1999	30737	172006362
Brazil	2000	80488	174004898
China	1999	210258	1272015272
China	2000	210766	1280023583

values

From R4DS, tidy data.

Tidy vs Untidy

Happy families are all alike; every unhappy family is unhappy in its own way.

—Leo Tolstoy, Anna Karenina

```
tidyr::billboard[, 1:7]
```

```
## # A tibble: 317 x 7
```

##	artist	track	date.entered	wk1	wk2	wk3	wk4
##	<chr>	<chr>	<date>	<dbl>	<dbl>	<dbl>	<dbl>
## 1	2 Pac	Baby Don't Cry (Keep...	2000-02-26	87	82	72	77
## 2	2Ge+her	The Hardest Part Of ...	2000-09-02	91	87	92	NA
## 3	3 Doors Down	Kryptonite	2000-04-08	81	70	68	67
## 4	3 Doors Down	Loser	2000-10-21	76	76	72	69
## 5	504 Boyz	Wobble Wobble	2000-04-15	57	34	25	17
## 6	98^0	Give Me Just One Nig...	2000-08-19	51	39	34	26
## 7	A*Teens	Dancing Queen	2000-07-08	97	97	96	95
## 8	Aaliyah	I Don't Wanna	2000-01-29	84	62	51	41
## 9	Aaliyah	Try Again	2000-03-18	59	53	38	28
## 10	Adams, Yolanda	Open My Heart	2000-08-26	76	76	74	69
## #	... with 307 more rows						

More tidy vs untidy

A

Untidy Data

species	habitat	weight	length	latitude/longitude	date
Alligator mississippiensis	swamp	431 lb	4 ft 2	29.531,-82.184	Sept 15, 2015
Puma concolor	forest	125 lb	2.2m	29.125,-81.682	08/10/2015
Ursus americanus	forest	88 kg	133 cm	N29°7'30"/W81°40'55.2"	07-13-2015

B

Tidy Data

meta-data

data

species_code	date	station_code	weight_kg	length_cm
TSN 551771	2015-09-15	1	196	127
TSN 55247	2015-08-10	2	57	220
TSN 180544	2015-07-13	2	88	133

station_code	habitat	latitude	longitude
1	swamp	29.531	-82.184
2	forest	29.125	-81.682

species_code	class	genus	species
TSN 551771	Reptilia	Alligator	mississippiensis
TSN 55247	Mammalia	Puma	concolor
TSN 180544	Mammalia	Ursus	americanus

Hart EM, Barmby P, LeBauer D, Michonneau F, Mount S, et al. (2016) Ten Simple Rules for Digital Data Storage. PLOS Computational Biology 12(10): e1005097. DOI: 10.1371/journal.pcbi.1005097

Tidy data

Tibbles

Pipeline

Data manipulation

Reshaping data (Wide vs. Long)

Modern data frames

Hadley Wickham / RStudio have a package that modifies data frames to be a bit more modern.

```
library(tibble)
```

```
iris
```

	Sepal.Length	Sepal.Width	Petal.Length
## 1	5.1	3.5	1.4
## 2	4.9	3.0	1.4
## 3	4.7	3.2	1.3
## 4	4.6	3.1	1.5
## 5	5.0	3.6	1.4
## 6	5.4	3.9	1.7
## 7	4.6	3.4	1.4
## 8	5.0	3.4	1.5
## 9	4.4	2.9	1.4
## 10	4.9	3.1	1.5
## 11	5.4	3.7	1.5
## 12	4.8	3.4	1.6
## 13	4.8	3.0	1.4
## 14	4.3	3.0	1.1

```
(tbl_iris = as_tibble(iris))
```

```
## # A tibble: 150 x 5
```

	Sepal.Length	Sepal.Wi~1	Petal~2	Petal~3	S
##	<dbl>	<dbl>	<dbl>	<dbl>	<
## 1	5.1	3.5	1.4	0.2	se
## 2	4.9	3	1.4	0.2	se
## 3	4.7	3.2	1.3	0.2	se
## 4	4.6	3.1	1.5	0.2	se
## 5	5	3.6	1.4	0.2	se
## 6	5.4	3.9	1.7	0.4	se
## 7	4.6	3.4	1.4	0.3	se
## 8	5	3.4	1.5	0.2	se
## 9	4.4	2.9	1.4	0.2	se
## 10	4.9	3.1	1.5	0.1	se

Tibbles are lazy

By default, subsetting tibbles always results in another tibble (`$` or `[[` can still be used to subset for a specific column).

```
tbl_iris[1, ]
```

```
## # A tibble: 1 x 5
##   Sepal.Length Sepal.Width Petal~1 Petal~2 Species
##         <dbl>         <dbl>   <dbl>   <dbl> <fct>
## 1         5.1         3.5     1.4     0.2 setosa
## # ... with abbreviated variable names
## #   1: Petal.Length, 2: Petal.Width
```

```
tbl_iris[[1]]
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4
##  [12] 4.8 4.8 4.3 5.8 5.7 5.4 5.1 5.7 5.1 5.4 5.1
##  [23] 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2
##  [34] 5.5 4.9 5.0 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0
##  [45] 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5 6.5
##  [56] 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7
##  [67] 5.6 5.8 6.2 5.6 5.9 6.1 6.3 6.1 6.4 6.6 6.8
```

```
tbl_iris[, 1]
```

```
## # A tibble: 150 x 1
##   Sepal.Length
##         <dbl>
## 1         5.1
## 2         4.9
## 3         4.7
## 4         4.6
## 5         5
## 6         5.4
## 7         4.6
## 8         5
## 9         4.4
## 10        4.9
## # ... with 140 more rows
```

More laziness: partial matching

Tibbles do not use partial matching when the \$ operator is used.

```
head( iris$Sp )  
## [1] setosa setosa setosa setosa setosa setosa  
## Levels: setosa versicolor virginica
```

```
head( iris$Species )  
## [1] setosa setosa setosa setosa setosa setosa  
## Levels: setosa versicolor virginica
```

```
head( tbl_iris$Sp )  
## Warning: Unknown or uninitialised column: `Sp`.  
## NULL
```

```
head( tbl_iris$Species )  
## [1] setosa setosa setosa setosa setosa setosa  
## Levels: setosa versicolor virginica
```

More laziness: stringsAsFactors

Tibbles also have always had `stringsAsFactors = FALSE` as default behavior.

```
(t = tibble(  
  x = 1:3,  
  y = c("A", "B", "C"),  
  z = factor(c("X", "Y", "Z"))  
))
```

```
## # A tibble: 3 x 3  
##       x y      z  
##   <int> <chr> <fct>  
## 1     1 A      X  
## 2     2 B      Y  
## 3     3 C      Z
```

```
(d = data.frame(  
  x = 1:3,  
  y = c("A", "B", "C"),  
  z = factor(c("X", "Y", "Z")),  
  stringsAsFactors = TRUE  
))
```

```
##    x y z  
## 1 1 A X  
## 2 2 B Y  
## 3 3 C Z
```

Tibbles and length coercion

Only vectors with length 1 will undergo length coercion. Everything else will throw an error.

```
data.frame(x = 1:4, y = 1)
```

```
##   x y
## 1 1 1
## 2 2 1
## 3 3 1
## 4 4 1
```

```
data.frame(x = 1:4, y = 1:2)
```

```
##   x y
## 1 1 1
## 2 2 2
## 3 3 1
## 4 4 2
```

```
tibble(x = 1:4, y = 1)
```

```
## # A tibble: 4 x 2
##       x     y
##   <int> <dbl>
## 1     1     1
## 2     2     1
## 3     3     1
## 4     4     1
```

```
tibble(x = 1:4, y = 1:2)
```

```
## Error:
## ! Tibble columns must have compatible sizes.
## * Size 4: Existing data.
## * Size 2: Column `y`.
## i Only values of size one are recycled.
```

Tidy data

Tibbles

Pipeline

Data manipulation

Reshaping data (Wide vs. Long)

What is a pipe

In software engineering, a pipeline consists of a chain of processing elements (processes, threads, coroutines, functions, etc.), arranged so that the output of each element is the input of the next.

—Wikipedia

Magrittr's pipe is a new infix operator that allows us to link two functions together in a way that is readable from left to right.

The two code examples below are equivalent:

```
f(g(x=1, y=2), n=2)
```

```
g(x=1, y=2) %>% f(n=2)
```

Readability

Consider the following sequence of actions that describe the process of getting to campus in the morning:

I need to find my key, then unlock my car, then start my car, then drive to school, then park.

Expressed as a set of nested functions in R pseudocode this would look like:

```
park(drive(start_car(find("keys")), to = "campus"))
```

Writing it out using pipes give it a more natural (and easier to read) structure:

```
find("keys") %>%  
  start_car() %>%  
  drive(to = "campus") %>%  
  park()
```

Approaches

All of the following are fine, it comes down to personal preference:

Nested:

```
h(g(f(x), y = 1), z = 1)
```

Piped:

```
f(x) %>%  
  g(y = 1) %>%  
  h(z = 1)
```

Intermediate:

```
res <- f(x)  
res <- g(res, y = 1)  
res <- h(res, z = 1)
```


What about other arguments?

Sometimes we want to send our results to an function argument other than first one or we want to use the previous result for multiple arguments. In these cases we can refer to the previous result using `'.'`.

```
data.frame(a = 1:3, b = 3:1) %>% lm(a ~ b, data = .)
```

```
##
```

```
## Call:
```

```
## lm(formula = a ~ b, data = .)
```

```
##
```

```
## Coefficients:
```

```
## (Intercept)          b
```

```
##           4          -1
```

```
data.frame(a = 1:3, b = 3:1) %>% .[[1]]
```

```
## [1] 1 2 3
```

```
data.frame(a = 1:3, b = 3:1) %>% .[[length(.)]]
```

```
## [1] 3 2 1
```

Tidy data

Tibbles

Pipeline

Data manipulation

- `filter`

- `slice`

- `select`

- `relocate`

- `pull`

- `arrange`

- Others

Reshaping data (Wide vs. Long)

A Grammar of Data Manipulation

dplyr is based on the concepts of functions as verbs that manipulate data frames.

Core single data frame functions / verbs:

- `filter()` / `slice()`: pick rows based on criteria
- `select()` / `rename()`: select columns by name
- `pull()`: grab a column as a vector
- `arrange()`: reorder rows
- `mutate()` / `transmute()`: create or modify columns
- `distinct()`: filter for unique rows
- `summarise()` / `count()`: reduce variables to values
- `group_by()` / `ungroup()`: modify other verbs to act on subsets
- `relocate()`: change column order
- ... (many more)

dplyr rules

1. First argument is *always* a data frame
2. Subsequent arguments say what to do with that data frame
3. *Always* return a data frame
4. Don't modify in place
5. Lazy evaluation magic

Example data

```
library(dplyr)
library(nycflights13)

flights
## # A tibble: 336,776 x 19
##   year month   day dep_time sched~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier flight
##   <int> <int> <int>   <int>   <int>   <dbl>   <int>   <int>   <dbl> <chr>   <int>
## 1  2013     1     1     517     515     2     830     819     11 UA     1545
## 2  2013     1     1     533     529     4     850     830     20 UA     1714
## 3  2013     1     1     542     540     2     923     850     33 AA     1141
## 4  2013     1     1     544     545    -1    1004    1022    -18 B6      725
## 5  2013     1     1     554     600    -6     812     837    -25 DL      461
## 6  2013     1     1     554     558    -4     740     728     12 UA    1696
## 7  2013     1     1     555     600    -5     913     854     19 B6      507
## 8  2013     1     1     557     600    -3     709     723    -14 EV    5708
## 9  2013     1     1     557     600    -3     838     846     -8 B6       79
## 10 2013     1     1     558     600    -2     753     745      8 AA     301
## # ... with 336,766 more rows, 8 more variables: tailnum <chr>, origin <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dtm>, and abbreviated variable names 1: sched_dep_time,
## #   2: dep_delay, 3: arr_time, 4: sched_arr_time, 5: arr_delay
```

filter(): March flights

```
flights %>% filter(month == 3)
## # A tibble: 28,834 x 19
##   year month   day dep_time sched~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier flight
##   <int> <int> <int>   <int>   <int>   <dbl>   <int>   <int>   <dbl> <chr>   <int>
## 1  2013     3     1       4    2159    125    318     56    142 B6        11
## 2  2013     3     1      50    2358     52    526    438     48 B6       707
## 3  2013     3     1    117    2245    152    223   2354    149 B6       608
## 4  2013     3     1    454     500     -6    633    648    -15 US     1117
## 5  2013     3     1    505     515    -10    746    810    -24 UA       475
## 6  2013     3     1    521     530     -9    813    827    -14 UA     1714
## 7  2013     3     1    537     540     -3    856    850     6 AA     1141
## 8  2013     3     1    541     545     -4   1014   1023     -9 B6       725
## 9  2013     3     1    549     600    -11    639    703    -24 US     2114
## 10 2013     3     1    550     600    -10    747    801    -14 EV     4911
## # ... with 28,824 more rows, 8 more variables: tailnum <chr>, origin <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dtm>, and abbreviated variable names 1: sched_dep_time,
## #   2: dep_delay, 3: arr_time, 4: sched_arr_time, 5: arr_delay
```

filter(): Flights in the first 7 days of March

```
flights %>% filter(month == 3, day <= 7)
## # A tibble: 6,530 x 19
##   year month   day dep_time sched~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier flight
##   <int> <int> <int>   <int>   <int>   <dbl>   <int>   <int>   <dbl> <chr>   <int>
## 1  2013     3     1         4    2159    125    318      56    142 B6        11
## 2  2013     3     1        50    2358     52    526     438     48 B6       707
## 3  2013     3     1       117    2245    152    223    2354    149 B6      608
## 4  2013     3     1       454     500     -6    633     648    -15 US     1117
## 5  2013     3     1       505     515    -10    746     810    -24 UA      475
## 6  2013     3     1       521     530     -9    813     827    -14 UA     1714
## 7  2013     3     1       537     540     -3    856     850      6 AA     1141
## 8  2013     3     1       541     545     -4   1014    1023     -9 B6      725
## 9  2013     3     1       549     600    -11    639     703    -24 US     2114
## 10 2013     3     1       550     600    -10    747     801    -14 EV     4911
## # ... with 6,520 more rows, 8 more variables: tailnum <chr>, origin <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dtm>, and abbreviated variable names 1: sched_dep_time,
## #   2: dep_delay, 3: arr_time, 4: sched_arr_time, 5: arr_delay
```

filter(): Flights to LAX or JFK in March

```
flights %>% filter(dest == "LAX" | dest == "JFK", month == 3)
## # A tibble: 1,178 x 19
##   year month   day dep_time sched~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier flight
##   <int> <int> <int>   <int>   <int>   <dbl>   <int>   <int>   <dbl> <chr>   <int>
## 1  2013     3     1     607     610     -3     832     925     -53 UA       797
## 2  2013     3     1     629     632     -3     844     952     -68 UA      1702
## 3  2013     3     1     657     700     -3     953    1034     -41 DL       763
## 4  2013     3     1     714     715     -1     939    1037     -58 B6       671
## 5  2013     3     1     716     710      6     958    1035     -37 VX       399
## 6  2013     3     1     727     730     -3    1007    1100     -53 AA        33
## 7  2013     3     1     836     840     -4    1111    1157     -46 UA     1553
## 8  2013     3     1     857     900     -3    1202    1221     -19 DL       120
## 9  2013     3     1     903     900      3    1157    1220     -23 AA        1
## 10 2013     3     1     904     831     33    1150    1151      -1 UA     1223
## # ... with 1,168 more rows, 8 more variables: tailnum <chr>, origin <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dtm>, and abbreviated variable names 1: sched_dep_time,
## #   2: dep_delay, 3: arr_time, 4: sched_arr_time, 5: arr_delay
```


slice(): First 10 flights

```
flights %>% slice(1:10)
## # A tibble: 10 x 19
##   year month   day dep_time sched~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier flight
##   <int> <int> <int>   <int>   <int>   <dbl>   <int>   <int>   <dbl> <chr>   <int>
## 1  2013     1     1     517     515     2     830     819     11  UA     1545
## 2  2013     1     1     533     529     4     850     830     20  UA     1714
## 3  2013     1     1     542     540     2     923     850     33  AA     1141
## 4  2013     1     1     544     545    -1    1004    1022    -18 B6      725
## 5  2013     1     1     554     600    -6     812     837    -25 DL      461
## 6  2013     1     1     554     558    -4     740     728     12  UA     1696
## 7  2013     1     1     555     600    -5     913     854     19  B6      507
## 8  2013     1     1     557     600    -3     709     723    -14 EV     5708
## 9  2013     1     1     557     600    -3     838     846     -8  B6       79
## 10 2013     1     1     558     600    -2     753     745      8  AA     301
## # ... with 8 more variables: tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dtm>, and
## #   abbreviated variable names 1: sched_dep_time, 2: dep_delay, 3: arr_time,
## #   4: sched_arr_time, 5: arr_delay
```

slice(): Last 5 flights

```
flights %>% slice((n() - 4):n())
```

```
## # A tibble: 5 x 19
```

```
##   year month   day dep_time sched_d~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier flight
##   <int> <int> <int>   <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>   <int>
## 1  2013     9    30      NA      1455     NA     NA     1634     NA  9E     3393
## 2  2013     9    30      NA      2200     NA     NA     2312     NA  9E     3525
## 3  2013     9    30      NA      1210     NA     NA     1330     NA  MQ     3461
## 4  2013     9    30      NA      1159     NA     NA     1344     NA  MQ     3572
## 5  2013     9    30      NA       840     NA     NA     1020     NA  MQ     3531
## # ... with 8 more variables: tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dtm>, and
## #   abbreviated variable names 1: sched_dep_time, 2: dep_delay, 3: arr_time,
## #   4: sched_arr_time, 5: arr_delay
```

slice(): Last 5 flights (cont.)

```
flights %>% slice_tail(n = 5)
```

```
## # A tibble: 5 x 19
```

```
##   year month   day dep_time sched_d~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier flight
##   <int> <int> <int>   <int>      <int>   <dbl>   <int>   <int>   <dbl> <chr>   <int>
## 1  2013     9    30      NA      1455     NA     NA     1634     NA  9E     3393
## 2  2013     9    30      NA      2200     NA     NA     2312     NA  9E     3525
## 3  2013     9    30      NA      1210     NA     NA     1330     NA  MQ     3461
## 4  2013     9    30      NA      1159     NA     NA     1344     NA  MQ     3572
## 5  2013     9    30      NA       840     NA     NA     1020     NA  MQ     3531
## # ... with 8 more variables: tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dtm>, and
## #   abbreviated variable names 1: sched_dep_time, 2: dep_delay, 3: arr_time,
## #   4: sched_arr_time, 5: arr_delay
```

select(): Individual Columns

```
flights %>% select(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(): Exclude Columns

```
flights %>% select(-year, -month, -day)
## # A tibble: 336,776 x 16
##   dep_time sched_dep~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier flight tailnum origin
##   <int>      <int>    <dbl>  <int>    <int>    <dbl> <chr>    <int> <chr>    <chr>
## 1      517        515      2      830      819      11 UA      1545 N14228 EWR
## 2      533        529      4      850      830      20 UA      1714 N24211 LGA
## 3      542        540      2      923      850      33 AA      1141 N619AA JFK
## 4      544        545     -1     1004     1022     -18 B6       725 N804JB JFK
## 5      554        600     -6      812      837     -25 DL       461 N668DN LGA
## 6      554        558     -4      740      728      12 UA      1696 N39463 EWR
## 7      555        600     -5      913      854      19 B6       507 N516JB EWR
## 8      557        600     -3      709      723     -14 EV      5708 N829AS LGA
## 9      557        600     -3      838      846      -8 B6        79 N593JB JFK
## 10     558        600     -2      753      745       8 AA       301 N3ALAA LGA
## # ... with 336,766 more rows, 6 more variables: dest <chr>, air_time <dbl>,
## # distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dtm>, and abbreviated
## # variable names 1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## # 5: arr_delay
```

select(): Ranges

This will select all columns within the two variables specified.

```
flights %>% select(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(): Exclusion Ranges

```
flights %>% select(-(year:day))
## # A tibble: 336,776 x 16
##   dep_time sched_dep~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier flight tailnum origin
##   <int>      <int>    <dbl>   <int>    <int>    <dbl> <chr>    <int> <chr>   <chr>
## 1      517        515      2     830      819      11 UA      1545 N14228 EWR
## 2      533        529      4     850      830      20 UA      1714 N24211 LGA
## 3      542        540      2     923      850      33 AA      1141 N619AA JFK
## 4      544        545     -1    1004    1022     -18 B6       725 N804JB JFK
## 5      554        600     -6     812      837     -25 DL       461 N668DN LGA
## 6      554        558     -4     740      728      12 UA      1696 N39463 EWR
## 7      555        600     -5     913      854      19 B6       507 N516JB EWR
## 8      557        600     -3     709      723     -14 EV      5708 N829AS LGA
## 9      557        600     -3     838      846      -8 B6        79 N593JB JFK
## 10     558        600     -2     753      745       8 AA       301 N3ALAA LGA
## # ... with 336,766 more rows, 6 more variables: dest <chr>, air_time <dbl>,
## # distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dtm>, and abbreviated
## # variable names 1: sched_dep_time, 2: dep_delay, 3: arr_time, 4: sched_arr_time,
## # 5: arr_delay
```

select(): Matching

```
flights %>% select(contains("dep"),  
                  contains("arr"))
```

```
## # A tibble: 336,776 x 7
```

##	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier
##	<int>	<int>	<dbl>	<int>	<int>	<dbl>	<chr>
## 1	517	515	2	830	819	11	UA
## 2	533	529	4	850	830	20	UA
## 3	542	540	2	923	850	33	AA
## 4	544	545	-1	1004	1022	-18	B6
## 5	554	600	-6	812	837	-25	DL
## 6	554	558	-4	740	728	12	UA
## 7	555	600	-5	913	854	19	B6
## 8	557	600	-3	709	723	-14	EV
## 9	557	600	-3	838	846	-8	B6
## 10	558	600	-2	753	745	8	AA

```
## # ... with 336,766 more rows
```


select(): Matching (cont.)

```
flights %>% select(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
```

Other helpers provide by tidyselect: `starts_with`, `ends_with`, `everything`, `matches`, `num_range`, `one_of`, `everything`, `last_col`.

select() + where(): Get numeric columns

```
flights %>% select(where(is.numeric))
## # A tibble: 336,776 x 14
##   year month   day dep_time sched_~1 dep_d~2 arr_t~3 sched~4 arr_d~5 flight air_t~6
##   <int> <int> <int>   <int>   <int>   <dbl>   <int>   <int>   <dbl>   <int>   <dbl>
## 1  2013     1     1     517     515         2     830     819        11    1545     227
## 2  2013     1     1     533     529         4     850     830        20    1714     227
## 3  2013     1     1     542     540         2     923     850        33    1141     160
## 4  2013     1     1     544     545        -1    1004    1022       -18     725     183
## 5  2013     1     1     554     600        -6     812     837       -25     461     116
## 6  2013     1     1     554     558        -4     740     728        12    1696     150
## 7  2013     1     1     555     600        -5     913     854        19     507     158
## 8  2013     1     1     557     600        -3     709     723       -14    5708        53
## 9  2013     1     1     557     600        -3     838     846        -8        79     140
## 10 2013     1     1     558     600        -2     753     745         8     301     138
## # ... with 336,766 more rows, 3 more variables: distance <dbl>, hour <dbl>,
## #   minute <dbl>, and abbreviated variable names 1: sched_dep_time, 2: dep_delay,
## #   3: arr_time, 4: sched_arr_time, 5: arr_delay, 6: air_time
```

relocate(): to the front

```
flights %>% relocate(carrier, origin, dest)
## # A tibble: 336,776 x 19
##   carrier origin dest   year month   day dep_time sched_de~1 dep_d~2 arr_t~3 sched~4
##   <chr>    <chr> <chr> <int> <int> <int>   <int>      <int>   <dbl>   <int>   <int>
## 1 UA      EWR   IAH   2013     1     1     517        515     2     830     819
## 2 UA      LGA   IAH   2013     1     1     533        529     4     850     830
## 3 AA      JFK   MIA   2013     1     1     542        540     2     923     850
## 4 B6      JFK   BQN   2013     1     1     544        545    -1    1004    1022
## 5 DL      LGA   ATL   2013     1     1     554        600    -6     812     837
## 6 UA      EWR   ORD   2013     1     1     554        558    -4     740     728
## 7 B6      EWR   FLL   2013     1     1     555        600    -5     913     854
## 8 EV      LGA   IAD   2013     1     1     557        600    -3     709     723
## 9 B6      JFK   MCO   2013     1     1     557        600    -3     838     846
## 10 AA     LGA   ORD   2013     1     1     558        600    -2     753     745
## # ... with 336,766 more rows, 8 more variables: arr_delay <dbl>, flight <int>,
## #   tailnum <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dtm>, and abbreviated variable names 1: sched_dep_time,
## #   2: dep_delay, 3: arr_time, 4: sched_arr_time
```

relocate(): to the end

```
flights %>%
  relocate(year, month, day, .after = last_col())
## # A tibble: 336,776 x 19
##   dep_time sched_dep~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier flight tailnum origin
##   <int>      <int>    <dbl>   <int>   <int>    <dbl> <chr>    <int> <chr>   <chr>
## 1      517        515      2     830     819     11 UA      1545 N14228 EWR
## 2      533        529      4     850     830     20 UA      1714 N24211 LGA
## 3      542        540      2     923     850     33 AA      1141 N619AA JFK
## 4      544        545     -1    1004    1022    -18 B6       725 N804JB JFK
## 5      554        600     -6     812     837    -25 DL       461 N668DN LGA
## 6      554        558     -4     740     728     12 UA      1696 N39463 EWR
## 7      555        600     -5     913     854     19 B6       507 N516JB EWR
## 8      557        600     -3     709     723    -14 EV      5708 N829AS LGA
## 9      557        600     -3     838     846     -8 B6        79 N593JB JFK
## 10     558        600     -2     753     745      8 AA       301 N3ALAA LGA
## # ... with 336,766 more rows, 9 more variables: dest <chr>, air_time <dbl>,
## # distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dtm>, year <int>,
## # month <int>, day <int>, and abbreviated variable names 1: sched_dep_time,
## # 2: dep_delay, 3: arr_time, 4: sched_arr_time, 5: arr_delay
```

rename(): Change column names

```
flights %>% rename(tail_number = tailnum)
```

```
## # A tibble: 336,776 x 19
```

##	year	month	day	dep_time	sched_~1	dep_d~2	arr_t~3	sched~4	arr_d~5	carrier	flight
##	<int>	<int>	<int>	<int>	<int>	<dbl>	<int>	<int>	<dbl>	<chr>	<int>
## 1	2013	1	1	517	515	2	830	819	11	UA	1545
## 2	2013	1	1	533	529	4	850	830	20	UA	1714
## 3	2013	1	1	542	540	2	923	850	33	AA	1141
## 4	2013	1	1	544	545	-1	1004	1022	-18	B6	725
## 5	2013	1	1	554	600	-6	812	837	-25	DL	461
## 6	2013	1	1	554	558	-4	740	728	12	UA	1696
## 7	2013	1	1	555	600	-5	913	854	19	B6	507
## 8	2013	1	1	557	600	-3	709	723	-14	EV	5708
## 9	2013	1	1	557	600	-3	838	846	-8	B6	79
## 10	2013	1	1	558	600	-2	753	745	8	AA	301

```
## # ... with 336,766 more rows, 8 more variables: tail_number <chr>, origin <chr>,  
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,  
## #   time_hour <dtm>, and abbreviated variable names 1: sched_dep_time,  
## #   2: dep_delay, 3: arr_time, 4: sched_arr_time, 5: arr_delay
```

pull

```
names(flights)
```

```
## [1] "year"          "month"         "day"           "dep_time"
## [5] "sched_dep_time" "dep_delay"     "arr_time"      "sched_arr_time"
## [9] "arr_delay"     "carrier"       "flight"        "tailnum"
## [13] "origin"        "dest"          "air_time"      "distance"
## [17] "hour"          "minute"        "time_hour"
```

```
flights %>% pull("year") %>% head()
```

```
## [1] 2013 2013 2013 2013 2013 2013
```

```
flights %>% pull(1) %>% head() # position from the left
```

```
## [1] 2013 2013 2013 2013 2013 2013
```

```
flights %>% pull(-1) %>% head() # position from the right
```

```
## [1] "2013-01-01 05:00:00 EST" "2013-01-01 05:00:00 EST" "2013-01-01 05:00:00 EST"
```

```
## [4] "2013-01-01 05:00:00 EST" "2013-01-01 06:00:00 EST" "2013-01-01 05:00:00 EST"
```

```
flights$year %>% head()
```

```
## [1] 2013 2013 2013 2013 2013 2013
```

arrange(): Sort data

```
flights %>% filter(month == 3, day == 2) %>% arrange(origin, dest)
```

```
## # A tibble: 765 x 19
```

```
##   year month   day dep_time sched~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier flight
##   <int> <int> <int>   <int>   <int>   <dbl>   <int>   <int>   <dbl> <chr>   <int>
## 1  2013     3     2    1336    1329     7    1426    1432     -6 EV     4263
## 2  2013     3     2     628     629    -1     837     849    -12 DL      575
## 3  2013     3     2     637     640    -3     903     915    -12 EV    4209
## 4  2013     3     2     743     745    -2     945    1010    -25 DL      807
## 5  2013     3     2     857     900    -3    1117    1126     -9 DL      485
## 6  2013     3     2    1027    1030    -3    1234    1247    -13 DL    2343
## 7  2013     3     2    1134    1145   -11    1332    1359    -27 DL      401
## 8  2013     3     2    1412    1415    -3    1636    1630     6 DL      935
## 9  2013     3     2    1633    1636    -3    1848    1908    -20 EV    3273
## 10 2013     3     2    1655    1700    -5    1857    1924    -27 DL    2042
## # ... with 755 more rows, 8 more variables: tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dtm>, and
## #   abbreviated variable names 1: sched_dep_time, 2: dep_delay, 3: arr_time,
## #   4: sched_arr_time, 5: arr_delay
```

arrange() & desc(): Descending order

```
flights %>%  
  filter(month == 3, day == 2) %>%  
  arrange(desc(origin), dest) %>%  
  select(origin, dest, tailnum)
```

```
## # A tibble: 765 x 3  
##   origin dest  tailnum  
##   <chr>  <chr> <chr>  
## 1 LGA    ATL   N928AT  
## 2 LGA    ATL   N623DL  
## 3 LGA    ATL   N680DA  
## 4 LGA    ATL   N996AT  
## 5 LGA    ATL   N510MQ  
## 6 LGA    ATL   N663DN  
## 7 LGA    ATL   N942DL  
## 8 LGA    ATL   N511MQ  
## 9 LGA    ATL   N910DE  
## 10 LGA   ATL   N902DE  
## # ... with 755 more rows
```


mutate(): Modify columns

```
flights %>%  
  select(year:day) %>%  
  mutate(date = paste(year, month, day, sep = "/"))
```

```
## # A tibble: 336,776 x 4  
##   year month   day date  
##   <int> <int> <int> <chr>  
## 1  2013     1     1 2013/1/1  
## 2  2013     1     1 2013/1/1  
## 3  2013     1     1 2013/1/1  
## 4  2013     1     1 2013/1/1  
## 5  2013     1     1 2013/1/1  
## 6  2013     1     1 2013/1/1  
## 7  2013     1     1 2013/1/1  
## 8  2013     1     1 2013/1/1  
## 9  2013     1     1 2013/1/1  
## 10 2013     1     1 2013/1/1  
## # ... with 336,766 more rows
```

distinct(): Find unique rows

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

```
## # A tibble: 224 x 2
```

```
##   origin dest  
##   <chr> <chr>
```

```
## 1 EWR    ALB
```

```
## 2 EWR    ANC
```

```
## 3 EWR    ATL
```

```
## 4 EWR    AUS
```

```
## 5 EWR    AVL
```

```
## 6 EWR    BDL
```

```
## 7 EWR    BNA
```

```
## 8 EWR    BOS
```

```
## 9 EWR    BQN
```

```
## 10 EWR   BTV
```

```
## # ... with 214 more rows
```

summarise()

```
flights %>%  
  summarize(n(), min(dep_delay), max(dep_delay))
```

```
## # A tibble: 1 x 3  
##   `n()` `min(dep_delay)` `max(dep_delay)`  
##   <int>         <dbl>         <dbl>  
## 1 336776           NA           NA
```

```
flights %>%  
  summarize(  
    n = n(),  
    min_dep_delay = min(dep_delay, na.rm = TRUE),  
    max_dep_delay = max(dep_delay, na.rm = TRUE)  
  )
```

```
## # A tibble: 1 x 3  
##       n min_dep_delay max_dep_delay  
##   <int>         <dbl>         <dbl>  
## 1 336776          -43          1301
```

group_by()

```
flights %>% group_by(origin)
## # A tibble: 336,776 x 19
## # Groups:   origin [3]
##   year month   day dep_time sched_~1 dep_d~2 arr_t~3 sched~4 arr_d~5 carrier flight
##   <int> <int> <int>   <int>   <int>   <dbl>   <int>   <int>   <dbl> <chr>   <int>
## 1  2013     1     1     517     515     2     830     819     11 UA     1545
## 2  2013     1     1     533     529     4     850     830     20 UA     1714
## 3  2013     1     1     542     540     2     923     850     33 AA     1141
## 4  2013     1     1     544     545    -1    1004    1022    -18 B6      725
## 5  2013     1     1     554     600    -6     812     837    -25 DL      461
## 6  2013     1     1     554     558    -4     740     728     12 UA    1696
## 7  2013     1     1     555     600    -5     913     854     19 B6      507
## 8  2013     1     1     557     600    -3     709     723    -14 EV    5708
## 9  2013     1     1     557     600    -3     838     846     -8 B6       79
## 10 2013     1     1     558     600    -2     753     745      8 AA     301
## # ... with 336,766 more rows, 8 more variables: tailnum <chr>, origin <chr>,
## #   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dtm>, and abbreviated variable names 1: sched_dep_time,
## #   2: dep_delay, 3: arr_time, 4: sched_arr_time, 5: arr_delay
```

summarise() with group_by()

```
flights %>%  
  group_by(origin) %>%  
  summarise(  
    n = n(),  
    min_dep_delay = min(dep_delay, na.rm = TRUE),  
    max_dep_delay = max(dep_delay, na.rm = TRUE)  
  )
```

```
## # A tibble: 3 x 4
```

```
##   origin      n min_dep_delay max_dep_delay  
##   <chr>   <int>         <dbl>         <dbl>  
## 1 EWR    120835          -25           1126  
## 2 JFK    111279          -43           1301  
## 3 LGA    104662          -33            911
```

count()

```
flights %>%  
  group_by(origin, carrier) %>%  
  summarise(n = n(), .groups = "drop")  
# or "drop_last" or "keep"
```

```
## # A tibble: 35 x 3  
##   origin carrier      n  
##   <chr>   <chr>   <int>  
## 1 EWR     9E       1268  
## 2 EWR     AA       3487  
## 3 EWR     AS        714  
## 4 EWR     B6       6557  
## 5 EWR     DL       4342  
## 6 EWR     EV      43939  
## 7 EWR     MQ       2276  
## 8 EWR     OO         6  
## 9 EWR     UA      46087  
## 10 EWR    US      4405  
## # ... with 25 more rows
```

```
flights %>%  
  count(origin, carrier)
```

```
## # A tibble: 35 x 3  
##   origin carrier      n  
##   <chr>   <chr>   <int>  
## 1 EWR     9E       1268  
## 2 EWR     AA       3487  
## 3 EWR     AS        714  
## 4 EWR     B6       6557  
## 5 EWR     DL       4342  
## 6 EWR     EV      43939  
## 7 EWR     MQ       2276  
## 8 EWR     OO         6  
## 9 EWR     UA      46087  
## 10 EWR    US      4405  
## # ... with 25 more rows
```

mutate() with group_by()

```
flights %>% group_by(origin) %>%  
  mutate(n = n()) %>%  
  select(origin, n)
```

```
## # A tibble: 336,776 x 2
```

```
## # Groups:   origin [3]
```

```
##   origin      n
```

```
##   <chr>   <int>
```

```
## 1 EWR    120835
```

```
## 2 LGA    104662
```

```
## 3 JFK    111279
```

```
## 4 JFK    111279
```

```
## 5 LGA    104662
```

```
## 6 EWR    120835
```

```
## 7 EWR    120835
```

```
## 8 LGA    104662
```

```
## 9 JFK    111279
```

```
## 10 LGA    104662
```

```
## # ... with 336,766 more rows
```

Tidy data

Tibbles

Pipeline

Data manipulation

Reshaping data (Wide vs. Long)

Wide → Long

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K



country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

`pivot_longer` (previously `gather`)

From data import cheatsheet.

Syntax

```
(d <- tibble::tribble(
  ~country, ~"1999", ~"2000",
    "A", "0.7K",    "2K",
    "B",  "37K",    "80K",
    "C", "212K",    "213K"
))
```

```
## # A tibble: 3 x 3
##   country `1999` `2000`
##   <chr>   <chr>  <chr>
## 1 A      0.7K    2K
## 2 B      37K    80K
## 3 C     212K   213K
```

```
pivot_longer(d, cols = "1999":"2000",
              names_to = "year",
              values_to = "cases")
```

```
## # A tibble: 6 x 3
##   country year  cases
##   <chr>   <chr> <chr>
## 1 A      1999  0.7K
## 2 A      2000   2K
## 3 B      1999  37K
## 4 B      2000  80K
## 5 C      1999 212K
## 6 C      2000 213K
```

Long → Wide

country	year	type	count		country	year	cases	pop
A	1999	cases	0.7K	→	A	1999	0.7K	19M
A	1999	pop	19M		A	2000	2K	20M
A	2000	cases	2K		B	1999	37K	172M
A	2000	pop	20M		B	2000	80K	174M
B	1999	cases	37K		C	1999	212K	1T
B	1999	pop	172M		C	2000	213K	1T
B	2000	cases	80K					
B	2000	pop	174M					
C	1999	cases	212K					
C	1999	pop	1T					
C	2000	cases	213K					
C	2000	pop	1T					

`pivot_wider` (previously `spread`)

Syntax


```
d <- tibble::tribble(  
  ~country, ~year, ~type, ~count,  
    "A", 1999, "cases", "0.7K",  
    "A", 1999, "pop", "19M",  
    "A", 2000, "cases", "2K",  
    "A", 2000, "pop", "20M",  
    "B", 1999, "cases", "37K",  
    "B", 1999, "pop", "172M",  
    "B", 2000, "cases", "80K",  
    "B", 2000, "pop", "174M",  
    "C", 1999, "cases", "212K",  
    "C", 1999, "pop", "1T",  
    "C", 2000, "cases", "213K",  
    "C", 2000, "pop", "1T"  
)
```

```
pivot_wider(d, id_cols = country:year,  
            names_from = type,  
            values_from = count)
```

```
## # A tibble: 6 x 4  
##   country year cases pop  
##   <chr>   <dbl> <chr> <chr>  
## 1 A      1999 "0.7K" 19M  
## 2 A      2000 "2K"   20M  
## 3 B      1999 "37K" 172M  
## 4 B      2000 "80K" 174M  
## 5 C      1999 "212K" 1T  
## 6 C      2000 "213K" 1T
```

separate()

country	year	rate	
A	1999	0.7K/19M	
A	2000	2K/20M	
B	1999	37K/172M	
B	2000	80K/174M	



country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172
B	2000	80K	174

```
separate(d, rate, sep = "/", into = c("cases", "pop"))
```

`unite()`



country	century	year
A	19	99
A	20	00
B	19	99
B	20	00

country	year
A	1999
A	2000
B	1999
B	2000

```
unite(d, century, year, col = "year", sep = "")
```

Merging two tibbles

Check out the explanation here: <https://github.com/gadenbuie/tidyexplain>

- `left_join(x, y)`
- `right_join(x, y)`
- `full_join(x, y)`
- `inner_join(x, y)`
- `union(x, y)` and `union_all(x, y)`
- `intersect(x, y)`
- `setdiff(x, y)`