Tidy Data, dplyr, and tidyr

Colin Rundel

2019-09-30

Functions

Function Parts

The two parts of a function are the arguments (`formals`) and the code (`body`).

```
gcd = function(long1, lat1, long2, lat2) {
  R = 6371 # Earth mean radius in km
  # distance in km
  acos(sin(lat1)*sin(lat2) + cos(lat1)*cos(lat2) * cos(long2-long1)) * R
}
```

Function Parts

The two parts of a function are the arguments (`formals`) and the code (`body`).

```
gcd = function(long1, lat1, long2, lat2) {
  R = 6371 # Earth mean radius in km
  # distance in km
  acos(sin(lat1)*sin(lat2) + cos(lat1)*cos(lat2) * cos(long2-long1)) * R
}
```

```
formals(gcd)
```

```
## $long1
## 
## 
## $lat1
## 
## 
## $long2
## 
## 
## $lat2
```

```
body(gcd)
```

```
## {
##     R = 6371
##     acos(sin(lat1) * sin(lat2) + cos(lat1) * co
##         long1)) * R
## }
```

Return values

There are two ways of returning values in R: explicitly or implicitly.

Explicit - includes one or more `return` statements

```
f = function(x) {
  return(x*x)
}
```

Implicit - value of the last statement is returned.

```
f = function(x) {
  x*x
}
```

Argument names

When defining a function we are also implicitly defining names for the arguments, when calling the function we can use these names to pass arguments in a different order.

```r
f = function(x,y,z) {
  paste0("x=",x," y=",y," z=",z)
}
```

```r
f(1,2,3)
```

```
## [1] "x=1 y=2 z=3"
```

```r
f(y=2,1,3)
```

```
## [1] "x=1 y=2 z=3"
```

```r
f(z=1,x=2,y=3)
```

```
## [1] "x=2 y=3 z=1"
```

```r
f(y=2,1,x=3)
```

```
## [1] "x=3 y=2 z=1"
```

```r
f(1,2,3,m=1)
```

```
## Error in f(1, 2, 3, m = 1): unused argument (m = 1)
```

Argument defaults

It is also possible to give function arguments default values so that they don't need to be provided every time the function is called.

```r
f = function(x,y=1,z=1) {
  paste0("x=",x," y=",y," z=",z)
}
```

```r
f()
```

```
## Error in paste0("x=", x, " y=", y, " z=", z): argument "x" is missing, with no default
```

```r
f(x=3)
```

```
## [1] "x=3 y=1 z=1"
```

```r
f(y=2,2)
```

```
## [1] "x=2 y=2 z=1"
```

Return values

Many of the built in functions in R will return a value, even if you haven't noticed that this is the case. This can be particularly problematic if you are using implicit return values, since you might be returning something you didn't expect.
Some examples,

```
x = y = 5
```

```
x
```

```
## [1] 5
```

```
y
```

```
## [1] 5
```

```
z = if (rnorm(1) > 0) {
  "pos"
} else {
  "neg"
}
```

```
z
```

```
## [1] "neg"
```

Return values

Many of the built in functions in R will return a value, even if you haven't noticed that this is the case. This can be particularly problematic if you are using implicit return values, since you might be returning something you didn't expect.
Some examples,

```
x = y = 5
```

```
x
```

```
## [1] 5
```

```
y
```

```
## [1] 5
```

```
y = 5
```

```
z = if (rnorm(1) > 0) {
  "pos"
} else {
  "neg"
}
```

```
z
```

```
## [1] "neg"
```

```
if (rnorm(1) > 0) {
  "pos"
} else {
  "neg"
}
```

```
## [1] "pos"
```

More oddness

```r
r = rnorm(1)
```

```r
if (r > 0) {
  print("pos")
} else {
  print("neg")
}
```

```
## [1] "neg"
```

```r
z = if (r > 0) {
  print("pos")
} else {
  print("neg")
}
```

```
## [1] "neg"
```

```r
z
```

```
## [1] "neg"
```

More oddness

```r
r = rnorm(1)
```

```r
if (r > 0) {
  print("pos")
} else {
  print("neg")
}
```

```
## [1] "neg"
```

```r
z = if (r > 0) {
  print("pos")
} else {
  print("neg")
}
```

```
## [1] "neg"
```

```r
z
```

```
## [1] "neg"
```

```r
typeof(print("ABC"))
```

```
## [1] "ABC"
```

```
## [1] "character"
```

```r
z = typeof(print("ABC"))
```

```
## [1] "ABC"
```

```r
z
```

```
## [1] "character"
```

Invisible values

```r
f = function(x) {
  invisible(x)
}
```

```r
g = function(x) {
  x
}
```

Invisible values

```r
f = function(x) {
  invisible(x)
}
```

```r
f(1)
```

```r
g = function(x) {
  x
}
```

```r
g(1)
```

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

Invisible values

```r
f = function(x) {
  invisible(x)
}
```

```r
f(1)
```

```r
x = f(1)
x
```

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

```r
g = function(x) {
  x
}
```

```r
g(1)
```

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

```r
y = g(1)
y
```

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

Even Operators are functions

```r
`+`
```

```
## function (e1, e2)  .Primitive("+")
```

```r
typeof(`+`)
```

```
## [1] "builtin"
```

```r
`+`(4:1,2)
```

```
## [1] 6 5 4 3
```

```r
4:1 + 2
```

```
## [1] 6 5 4 3
```

```r
`|`
```

```
## function (e1, e2)  .Primitive("|")
```

```r
typeof(`|`)
```

```
## [1] "builtin"
```

```r
`|`(TRUE,FALSE)
```

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

```r
TRUE | FALSE
```

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

Even Operators are functions

```
`+`
```

```
## function (e1, e2)  .Primitive("+")
```

```
typeof(`+`)
```

```
## [1] "builtin"
```

```
`+`(4:1,2)
```

```
## [1] 6 5 4 3
```

```
4:1 + 2
```

```
## [1] 6 5 4 3
```

```
`|`
```

```
## function (e1, e2)  .Primitive("|")
```

```
typeof(`|`)
```

```
## [1] "builtin"
```

```
`|`(TRUE,FALSE)
```

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

```
TRUE | FALSE
```

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

```
`$`
```

```
## .Primitive("$")
```

```
`[[`
```

```
## .Primitive("[[")
```

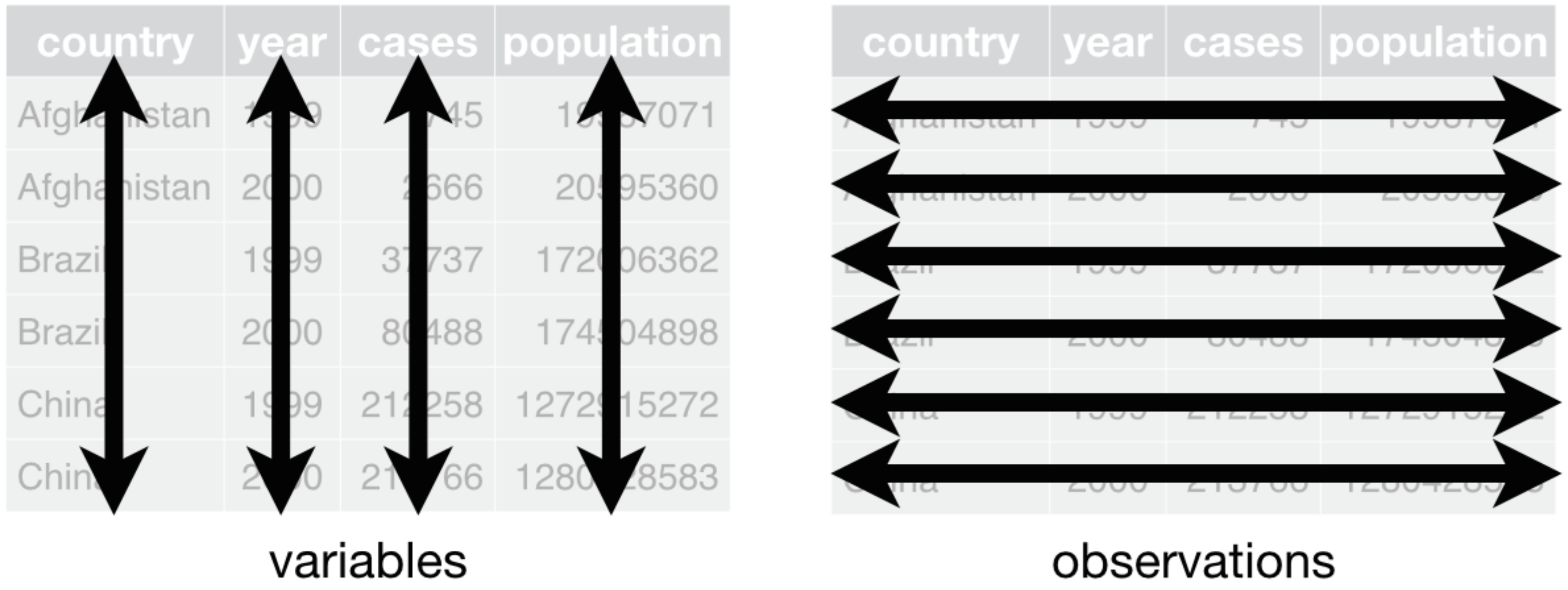
```
`names<-`
```

```
## function (x, value)  .Primitive("names<-")
```

tidyverse

www.rstudio.com

Tidy data



- One variable per column
- One observation per row
- Each type of observational unit forms a table

From R4DS - tidy data

TIBBLE
www.rstudio.com

Modern data frames

Hadley Wickham / RStudio have a package that modifies data frames to be more modern, or as he calls them surly and lazy.

```
library(tibble)
class(iris)
```

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

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

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

Fancy Printing

```
tbl_iris
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>  
##  1          5.1         3.5          1.4         0.2 setosa 
##  2          4.9         3            1.4         0.2 setosa 
##  3          4.7         3.2          1.3         0.2 setosa 
##  4          4.6         3.1          1.5         0.2 setosa 
##  5          5           3.6          1.4         0.2 setosa 
##  6          5.4         3.9          1.7         0.4 setosa 
##  7          4.6         3.4          1.4         0.3 setosa 
##  8          5           3.4          1.5         0.2 setosa 
##  9          4.4         2.9          1.4         0.2 setosa 
## 10          4.9         3.1          1.5         0.1 setosa 
## # … with 140 more rows
```

```
iris
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1            5.1         3.5          1.4         0.2     setosa
## 2            4.9         3.0          1.4         0.2     setosa
## 3            4.7         3.2          1.3         0.2     setosa
## 4            4.6         3.1          1.5         0.2     setosa
## 5            5.0         3.6          1.4         0.2     setosa
## 6            5.4         3.9          1.7         0.4     setosa
## 7            4.6         3.4          1.4         0.3     setosa
## 8            5.0         3.4          1.5         0.2     setosa
## 9            4.4         2.9          1.4         0.2     setosa
## 10           4.9         3.1          1.5         0.1     setosa
## 11           5.4         3.7          1.5         0.2     setosa
## 12           4.8         3.4          1.6         0.2     setosa
```

```r
df = data.frame(x = rnorm(10,sd=5), y = rnorm(10), z = runif(10))
```

```r
as_tibble(df)
```

```
## # A tibble: 10 x 3
##        x      y      z
##    <dbl>  <dbl>  <dbl>
##  1 -1.58  0.301 0.575
##  2  2.79  0.380 0.590
##  3 -2.53 -1.04  0.525
##  4 -2.64 -1.12  0.835
##  5 -3.20 -0.324 0.673
##  6 -3.77 -0.344 0.823
##  7 -6.88 -0.154 0.710
##  8 -1.24  0.657 0.642
##  9  9.01  0.437 0.0562
## 10  3.01  1.60  0.417
```

```r
df
```

```
##            x          y          z
## 1  -1.581470  0.3008048 0.57475339
## 2   2.789190  0.3803478 0.58980821
## 3  -2.526325 -1.0411151 0.52470878
## 4  -2.644518 -1.1245520 0.83510024
## 5  -3.196826 -0.3237571 0.67307743
## 6  -3.771731 -0.3442898 0.82257162
## 7  -6.875631 -0.1535736 0.70978067
## 8  -1.244910  0.6570045 0.64154025
## 9   9.011300  0.4373594 0.05624825
## 10  3.005200  1.5967680 0.41691734
```

Tibbles are lazy

```r
tbl_iris[1,]
```

```
## # A tibble: 1 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>  
## 1          5.1         3.5          1.4         0.2 setosa 
```

Tibbles are lazy

```r
tbl_iris[1,]
```

```
## # A tibble: 1 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          5.1         3.5          1.4         0.2 setosa
```

```r
tbl_iris[,"Species"]
```

```
## # A tibble: 150 x 1
##    Species
##    <fct>
##  1 setosa
##  2 setosa
##  3 setosa
##  4 setosa
##  5 setosa
##  6 setosa
##  7 setosa
##  8 setosa
##  9 setosa
## 10 setosa
## # … with 140 more rows
```

Tibbles are lazy

```r
tbl_iris[1,]
```

```
## # A tibble: 1 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>  
## 1          5.1         3.5          1.4         0.2 setosa
```

```r
tbl_iris[,"Species"]
```

```
## # A tibble: 150 x 1
##    Species
##    <fct>  
##  1 setosa 
##  2 setosa 
##  3 setosa 
##  4 setosa 
##  5 setosa 
##  6 setosa 
##  7 setosa 
##  8 setosa 
##  9 setosa 
## 10 setosa 
## # … with 140 more rows
```

```r
tibble(
  x = 1:3,
  y = c("A","B","C")
)
```

```
## # A tibble: 3 x 2
##       x y    
##   <int> <chr>
## 1     1 A    
## 2     2 B    
## 3     3 C
```

More laziness

```r
head( tbl_iris[1] )
```

```
## # A tibble: 6 x 1
##   Sepal.Length
##          <dbl>
## 1          5.1
## 2          4.9
## 3          4.7
## 4          4.6
## 5          5
## 6          5.4
```

More laziness

```
head( tbl_iris[1] )
```

```
## # A tibble: 6 x 1
##   Sepal.Length
##          <dbl>
## 1          5.1
## 2          4.9
## 3          4.7
## 4          4.6
## 5          5
## 6          5.4
```

```
head( tbl_iris[[1]] )
```

```
## [1] 5.1 4.9 4.7 4.6 5.0 5.4
```

More laziness

```r
head( tbl_iris[1] )
```

```
## # A tibble: 6 x 1
##   Sepal.Length
##          <dbl>
## 1          5.1
## 2          4.9
## 3          4.7
## 4          4.6
## 5          5
## 6          5.4
```

```r
head( tbl_iris[[1]] )
```

```
## [1] 5.1 4.9 4.7 4.6 5.0 5.4
```

```r
head( iris$Sp )
```

```
## [1] setosa setosa setosa setosa setosa setosa
## Levels: setosa versicolor virginica
```

```r
tbl_iris$Sp
```

```
## Warning: Unknown or uninitialised column: 'Sp'.
```

```
## NULL
```

```r
head( tbl_iris$Species )
```

```
## [1] setosa setosa setosa setosa setosa setosa
## Levels: setosa versicolor virginica
```

Tibbles and length coercion

```r
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
```

Tibbles and length coercion

```r
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
```

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

```
## Tibble columns must have consistent lengths, only values of length one are recycled:
## * Length 2: Column `y`
## * Length 4: Column `x`
```

Tibbles and length coercion

```r
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
```

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

```
## Tibble columns must have consistent lengths, only values of length one are recycled:
## * Length 2: Column `y`
## * Length 4: Column `x`
```

```r
tibble(x = 1:4, y = 1:3)
```

```
## Tibble columns must have consistent lengths, only values of length one are recycled:
## * Length 3: Column `y`
## * Length 4: Column `x`
```

Tibbles and S3

```r
d = tibble(
  x = 1:3,
  y = c("A","B","C")
)

class(d)
```

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

Tibbles and S3

```r
d = tibble(
  x = 1:3,
  y = c("A","B","C")
)

class(d)
```

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

```r
class(d) = rev(class(d))
class(d)
```

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

```r
d
```

```
##   x y
## 1 1 A
## 2 2 B
## 3 3 C
```

magrittr

Pipes in R

You can think about the following sequence of actions - find key, unlock car, start car, drive to school, 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 `.`.

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
```

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
```

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 `.`.

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

```
##
## Call:
## lm(formula = a ~ b, data = .)
##
## Coefficients:
## (Intercept)            b
##           4           -1
```

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

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

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

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

dplyr

www.rstudio.com

A Grammar of Data Manipulation

dplyr is based on the concepts of functions as verbs that manipulate data frames.
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()`: add new variables
- `distinct()`: filter for unique rows
- `sample_n()` / `sample_frac()`: randomly sample rows
- `summarise()` / `count()`: reduce variables to values
- `group_by()` / `ungroup()`: modify other verbs to act on subsets
- ... (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

We will demonstrate dplyr's functionality using the nycflights13 data.

```r
library(dplyr)
library(nycflights13)

flights
```

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # … with 336,766 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

filter() - March flights

```r
flights %>% filter(month == 3)
```

```
## # A tibble: 28,834 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     3     1        4           2159       125      318             56
##  2  2013     3     1       50           2358        52      526            438
##  3  2013     3     1      117           2245       152      223           2354
##  4  2013     3     1      454            500        -6      633            648
##  5  2013     3     1      505            515       -10      746            810
##  6  2013     3     1      521            530        -9      813            827
##  7  2013     3     1      537            540        -3      856            850
##  8  2013     3     1      541            545        -4     1014           1023
##  9  2013     3     1      549            600       -11      639            703
## 10  2013     3     1      550            600       -10      747            801
## # … with 28,824 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

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

```r
flights %>% filter(month == 3, day <= 7)
```

```
## # A tibble: 6,530 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     3     1        4           2159       125      318             56
##  2  2013     3     1       50           2358        52      526            438
##  3  2013     3     1      117           2245       152      223           2354
##  4  2013     3     1      454            500        -6      633            648
##  5  2013     3     1      505            515       -10      746            810
##  6  2013     3     1      521            530        -9      813            827
##  7  2013     3     1      537            540        -3      856            850
##  8  2013     3     1      541            545        -4     1014           1023
##  9  2013     3     1      549            600       -11      639            703
## 10  2013     3     1      550            600       -10      747            801
## # … with 6,520 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

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_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     3     1      607            610        -3      832            925
##  2  2013     3     1      629            632        -3      844            952
##  3  2013     3     1      657            700        -3      953           1034
##  4  2013     3     1      714            715        -1      939           1037
##  5  2013     3     1      716            710         6      958           1035
##  6  2013     3     1      727            730        -3     1007           1100
##  7  2013     3     1      836            840        -4     1111           1157
##  8  2013     3     1      857            900        -3     1202           1221
##  9  2013     3     1      903            900         3     1157           1220
## 10  2013     3     1      904            831        33     1150           1151
## # … with 1,168 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

slice() - First 10 flights

```r
flights %>% slice(1:10)
```

```
## # A tibble: 10 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # … with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

slice() - Last 5 flights

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

```
## # A tibble: 5 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     9    30       NA           1455        NA       NA           1634
## 2  2013     9    30       NA           2200        NA       NA           2312
## 3  2013     9    30       NA           1210        NA       NA           1330
## 4  2013     9    30       NA           1159        NA       NA           1344
## 5  2013     9    30       NA            840        NA       NA           1020
## # … with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

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_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, and 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

select() - Ranges

```
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_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, and 9 more variables: flight <int>, tailnum <chr>,
## #   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
## #   minute <dbl>, time_hour <dttm>
```

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
```

```
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
```

Some other helpers (provide by tidyselect):

`starts_with`, `ends_with`, `everything`, `matches`, `num_range`, `one_of`, `everything`, `last_col`.

select_if() - Get non-numeric columns

```
flights %>% select_if(function(x) !is.numeric(x))
```

```
## # A tibble: 336,776 x 5
##    carrier tailnum origin dest  time_hour
##    <chr>   <chr>   <chr>  <chr> <dttm>
##  1 UA      N14228  EWR    IAH   2013-01-01 05:00:00
##  2 UA      N24211  LGA    IAH   2013-01-01 05:00:00
##  3 AA      N619AA  JFK    MIA   2013-01-01 05:00:00
##  4 B6      N804JB  JFK    BQN   2013-01-01 05:00:00
##  5 DL      N668DN  LGA    ATL   2013-01-01 06:00:00
##  6 UA      N39463  EWR    ORD   2013-01-01 05:00:00
##  7 B6      N516JB  EWR    FLL   2013-01-01 06:00:00
##  8 EV      N829AS  LGA    IAD   2013-01-01 06:00:00
##  9 B6      N593JB  JFK    MCO   2013-01-01 06:00:00
## 10 AA      N3ALAA  LGA    ORD   2013-01-01 06:00:00
## # … with 336,766 more rows
```

rename() - Change column names

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

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # … with 336,766 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tail_number <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

select() vs. rename()

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

```
## # A tibble: 336,776 x 1
##    tail_number
##    <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
```

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

```
## # A tibble: 336,776 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # … with 336,766 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tail_number <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

pull()

```r
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"
```

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

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

```r
flights %>% pull(1) %>% head()
```

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

```r
flights %>% pull(-1) %>% head()
```

```
## [1] "2013-01-01 05:00:00 EST" "2013-01-01 05:00:00 EST"
## [3] "2013-01-01 05:00:00 EST" "2013-01-01 05:00:00 EST"
## [5] "2013-01-01 06:00:00 EST" "2013-01-01 05:00:00 EST"
```

arrange() - Sort data

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

```
## # A tibble: 765 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     3     2     1336           1329         7     1426           1432
##  2  2013     3     2      628            629        -1      837            849
##  3  2013     3     2      637            640        -3      903            915
##  4  2013     3     2      743            745        -2      945           1010
##  5  2013     3     2      857            900        -3     1117           1126
##  6  2013     3     2     1027           1030        -3     1234           1247
##  7  2013     3     2     1134           1145       -11     1332           1359
##  8  2013     3     2     1412           1415        -3     1636           1630
##  9  2013     3     2     1633           1636        -3     1848           1908
## 10  2013     3     2     1655           1700        -5     1857           1924
## # … with 755 more rows, and 11 more variables: arr_delay <dbl>, carrier <chr>,
## #   flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
## #   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

arrange() & desc() - Descending order

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

```
## # 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

```r
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
```

transmute() - Create new tibble from existing columns

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

```
## # A tibble: 336,776 x 1
##    date
##    <chr>
##  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
```

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
```

Sampling rows

```
flights %>% sample_n(10)
```

```
## # A tibble: 10 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     6     4      859            900        -1     1208           1204
##  2  2013     1     3     1614           1555        19     1733           1715
##  3  2013     8    27      749            740         9     1029           1055
##  4  2013    11     4      757            757         0     1004           1024
##  5  2013     6     9     1636           1630         6     1839           1920
##  6  2013     5     8      624            630        -6      829            849
##  7  2013     9    11     1133           1130         3     1305           1311
##  8  2013     3    12      837            830         7      931            944
##  9  2013    10     4     1816           1800        16     1949           1920
## 10  2013     5    31      658            700        -2      957           1034
## # … with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
flights %>% sample_frac(0.00003)
```

```
## # A tibble: 10 x 19
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     4     1     1255           1120        95     1430           1302
##  2  2013    12     5      937            945        -8     1307           1235
##  3  2013     4     2     1051           1056        -5     1206           1215
##  4  2013    10     6     1040           1046        -6     1322           1330
##  5  2013    10    17     1128           1130        -2     1335           1334
##  6  2013     3    23     1045           1049        -4     1354           1340
##  7  2013    12     6      552            600        -8      649            701
##  8  2013     1     2     2141           2145        -4     2301           2311
##  9  2013    12     3      653            650         3      915            919
## 10  2013     2    25      604            610        -6      800            818
## # … with 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

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
```

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()

```r
flights %>% group_by(origin)
```

```
## # A tibble: 336,776 x 19
## # Groups:   origin [3]
##     year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##    <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
##  1  2013     1     1      517            515         2      830            819
##  2  2013     1     1      533            529         4      850            830
##  3  2013     1     1      542            540         2      923            850
##  4  2013     1     1      544            545        -1     1004           1022
##  5  2013     1     1      554            600        -6      812            837
##  6  2013     1     1      554            558        -4      740            728
##  7  2013     1     1      555            600        -5      913            854
##  8  2013     1     1      557            600        -3      709            723
##  9  2013     1     1      557            600        -3      838            846
## 10  2013     1     1      558            600        -2      753            745
## # … with 336,766 more rows, and 11 more variables: arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
```

summarise() with group_by()

```
flights %>% group_by(origin) %>%
  summarize(
    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
```

```
flights %>% group_by(origin, carrier) %>%
  summarize(
    n = n(),
    min_dep_delay = min(dep_delay, na.rm = TRUE),
    max_dep_delay = max(dep_delay, na.rm = TRUE)
  ) %>%
  filter(n > 10000)
```

```
## # A tibble: 10 x 5
## # Groups:   origin [3]
##    origin carrier     n min_dep_delay max_dep_delay
##    <chr>  <chr>   <int>         <dbl>         <dbl>
##  1 EWR    EV      43939           -25           548
##  2 EWR    UA      46087           -18           424
##  3 JFK    9E      14651           -24           747
##  4 JFK    AA      13783           -15          1014
##  5 JFK    B6      42076           -43           453
##  6 JFK    DL      20701           -18           960
##  7 LGA    AA      15459           -24           803
##  8 LGA    DL      23067           -33           911
##  9 LGA    MQ      16928           -26           366
## 10 LGA    US      13136           -18           500
```

count()

```r
flights %>%
  group_by(origin, carrier) %>%
  summarize(n = n()) %>%
  ungroup()
```

```
## # 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
```

```r
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
```

Demos

1. How many flights to Los Angeles (LAX) did each of the legacy carriers (AA, UA, DL or US) have in May from JFK, and what was their average duration?

1. What was the shortest flight out of each airport in terms of distance? In terms of duration?

Exercise 1

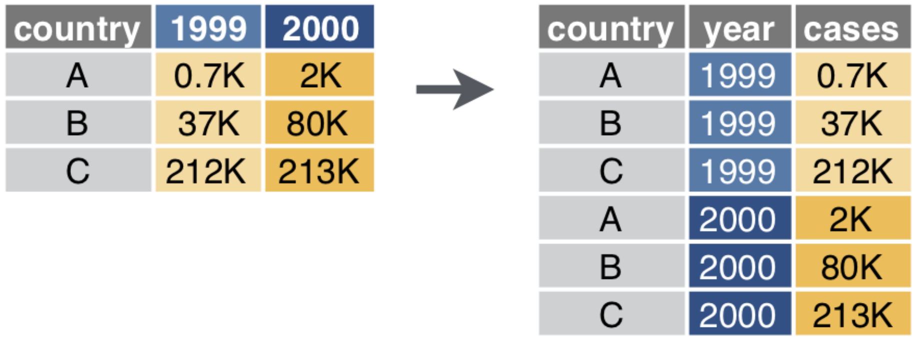
1. Which plane (check the tail number) flew out of each New York airport the most?

1. Which date should you fly on if you want to have the lowest possible average departure delay? What about arrival delay?

tidyr

www.rstudio.com

Gather

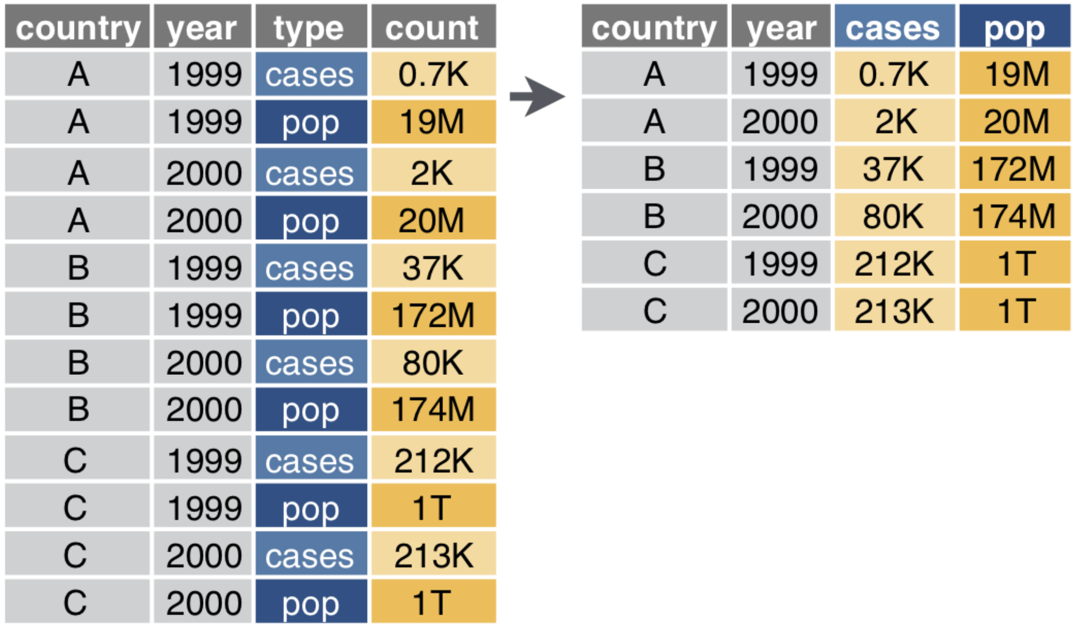


country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

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

From data import cheatsheet

Spread

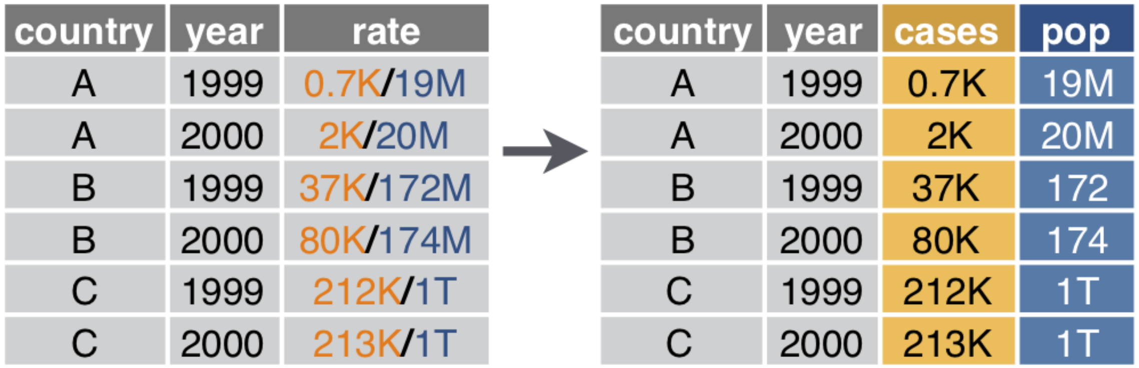


country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

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

From data import cheatsheet

Separate

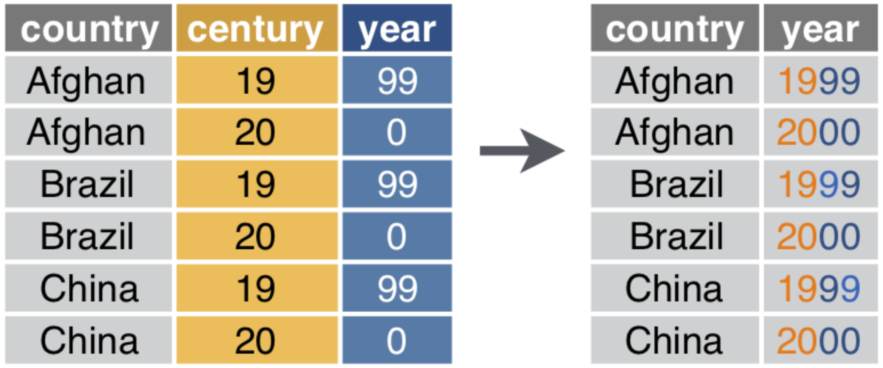


country	year	rate
A	1999	0.7K/19M
A	2000	2K/20M
B	1999	37K/172M
B	2000	80K/174M
C	1999	212K/1T
C	2000	213K/1T

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

From data import cheatsheet

Unite



country	century	year
Afghan	19	99
Afghan	20	0
Brazil	19	99
Brazil	20	0
China	19	99
China	20	0

country	year
Afghan	1999
Afghan	2000
Brazil	1999
Brazil	2000
China	1999
China	2000

From data import cheatsheet

Example 1 - Grades

Is the following data tidy?

```
(grades = tibble(
  name   = c("Alice", "Bob", "Carol", "Dave"),
  hw_1    = c(19, 18, 18, 19),
  hw_2    = c(19, 20, 20, 19),
  hw_3    = c(18, 18, 18, 18),
  hw_4    = c(20, 16, 17, 19),
  exam_1 = c(89, 77, 96, 86),
  exam_2 = c(95, 88, 99, 82)
))
```

```
## # A tibble: 4 x 7
##   name   hw_1  hw_2  hw_3  hw_4 exam_1 exam_2
##   <chr> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>
## 1 Alice    19    19    18    20     89     95
## 2 Bob      18    20    18    16     77     88
## 3 Carol    18    20    18    17     96     99
## 4 Dave     19    19    18    19     86     82
```

Example 1 - Grades

Is the following data tidy?

```
(grades = tibble(
  name   = c("Alice", "Bob", "Carol", "Dave"),
  hw_1    = c(19, 18, 18, 19),
  hw_2    = c(19, 20, 20, 19),
  hw_3    = c(18, 18, 18, 18),
  hw_4    = c(20, 16, 17, 19),
  exam_1 = c(89, 77, 96, 86),
  exam_2 = c(95, 88, 99, 82)
))
```

```
## # A tibble: 4 x 7
##   name   hw_1  hw_2  hw_3  hw_4 exam_1 exam_2
##   <chr> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>
## 1 Alice    19    19    18    20     89     95
## 2 Bob      18    20    18    16     77     88
## 3 Carol    18    20    18    17     96     99
## 4 Dave     19    19    18    19     86     82
```

How would we calculate a final score based on the following formula,

$$\text{score} = 0.6\,\frac{\sum \text{hw}_i}{80} + 0.4\,\frac{\sum \text{exam}_j}{200}$$

Semi-tidy approach

```
grades %>%
  mutate(
    hw_avg = (hw_1+hw_2+hw_3+hw_4)/4,
    exam_avg = (exam_1+exam_2)/2
  ) %>%
  mutate(
    overall = 0.4*(exam_avg/100) + 0.6*(hw_avg/20)
  )
```

```
## # A tibble: 4 x 10
##   name   hw_1  hw_2  hw_3  hw_4 exam_1 exam_2 hw_avg exam_avg overall
##   <chr> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>    <dbl>   <dbl>
## 1 Alice    19    19    18    20     89     95   19       92     0.938
## 2 Bob      18    20    18    16     77     88   18       82.5   0.87
## 3 Carol    18    20    18    17     96     99   18.2     97.5   0.938
## 4 Dave     19    19    18    19     86     82   18.8     84     0.899
```

Wide -> Long (pivot_longer)

```
tidyr::pivot_longer(grades, cols = hw_1:exam_2,
                    names_to = "assignment",
                    values_to = "score")
```

```
## # A tibble: 24 x 3
##    name  assignment score
##    <chr> <chr>      <dbl>
##  1 Alice hw_1          19
##  2 Alice hw_2          19
##  3 Alice hw_3          18
##  4 Alice hw_4          20
##  5 Alice exam_1        89
##  6 Alice exam_2        95
##  7 Bob   hw_1          18
##  8 Bob   hw_2          20
##  9 Bob   hw_3          18
## 10 Bob   hw_4          16
## # … with 14 more rows
```

```r
tidyr::pivot_longer(grades, cols = hw_1:exam_2,
                    names_to = c("type", "id"), names_sep = "_",
                    values_to = "score")
```

```
## # A tibble: 24 x 4
##    name  type  id    score
##    <chr> <chr> <chr> <dbl>
##  1 Alice hw    1        19
##  2 Alice hw    2        19
##  3 Alice hw    3        18
##  4 Alice hw    4        20
##  5 Alice exam  1        89
##  6 Alice exam  2        95
##  7 Bob   hw    1        18
##  8 Bob   hw    2        20
##  9 Bob   hw    3        18
## 10 Bob   hw    4        16
## # … with 14 more rows
```

Tidy approach?

```r
grades %>%
  tidyr::pivot_longer(
    cols = hw_1:exam_2,
    names_to = c("type", "id"), names_sep = "_",
    values_to = "score"
  ) %>%
  group_by(name, type) %>%
  summarize(total = sum(score))
```

```
## # A tibble: 8 x 3
## # Groups:   name [4]
##   name  type  total
##   <chr> <chr> <dbl>
## 1 Alice exam    184
## 2 Alice hw       76
## 3 Bob   exam    165
## 4 Bob   hw       72
## 5 Carol exam    195
## 6 Carol hw       73
## 7 Dave  exam    168
## 8 Dave  hw       75
```

Long -> Wide (pivot_wider)

```
grades %>%
  tidyr::pivot_longer(
    cols = hw_1:exam_2,
    names_to = c("type", "id"), names_sep = "_",
    values_to = "score"
  ) %>%
  group_by(name, type) %>%
  summarize(total = sum(score)) %>%
  tidyr::pivot_wider(
    names_from = type, values_from = total
  )
```

```
## # A tibble: 4 x 3
## # Groups:   name [4]
##   name   exam    hw
##   <chr> <dbl> <dbl>
## 1 Alice   184    76
## 2 Bob     165    72
## 3 Carol   195    73
## 4 Dave    168    75
```

Finishing up

```
grades %>%
  tidyr::pivot_longer(
    cols = hw_1:exam_2,
    names_to = c("type", "id"), names_sep = "_",
    values_to = "score"
  ) %>%
  group_by(name, type) %>%
  summarize(total = sum(score)) %>%
  tidyr::pivot_wider(
    names_from = type, values_from = total
  ) %>%
  mutate(
    score = 0.6*(hw/80) + 0.4*(exam/200)
  )
```

```
## # A tibble: 4 x 4
## # Groups:   name [4]
##   name   exam    hw score
##   <chr> <dbl> <dbl> <dbl>
## 1 Alice   184    76 0.938
## 2 Bob     165    72 0.87
## 3 Carol   195    73 0.938
## 4 Dave    168    75 0.899
```