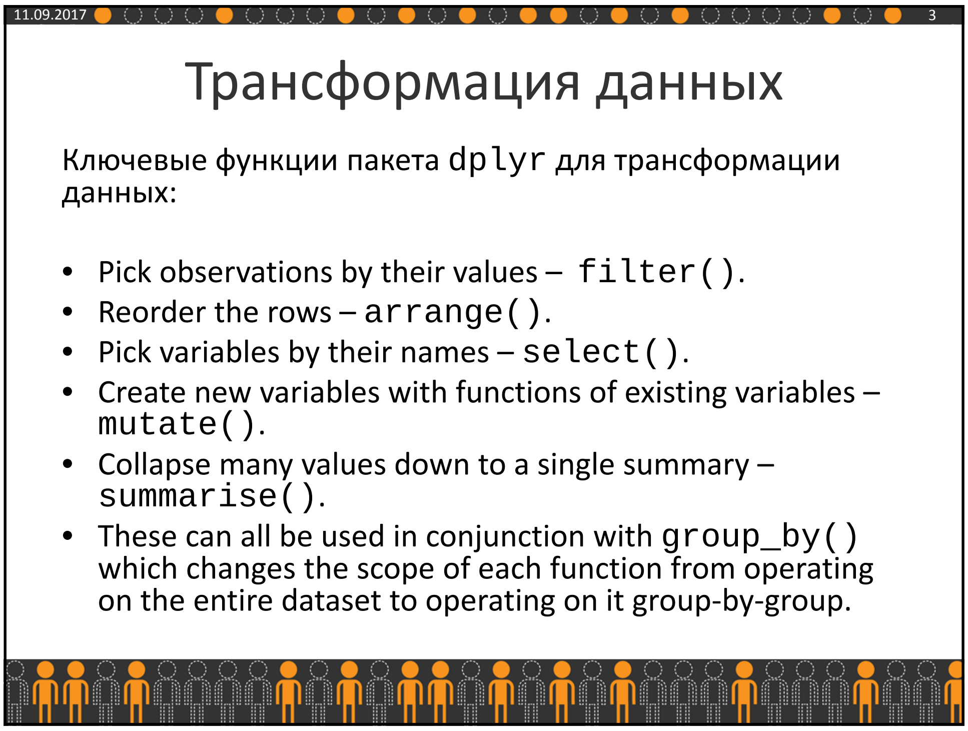
Анализ данных

Лекция 4 Трансформация данных в R

Гедранович Ольга Брониславовна,
старший преподаватель кафедры ИТ, МИУ
volha.b.k@gmail.com

Вопросы лекции

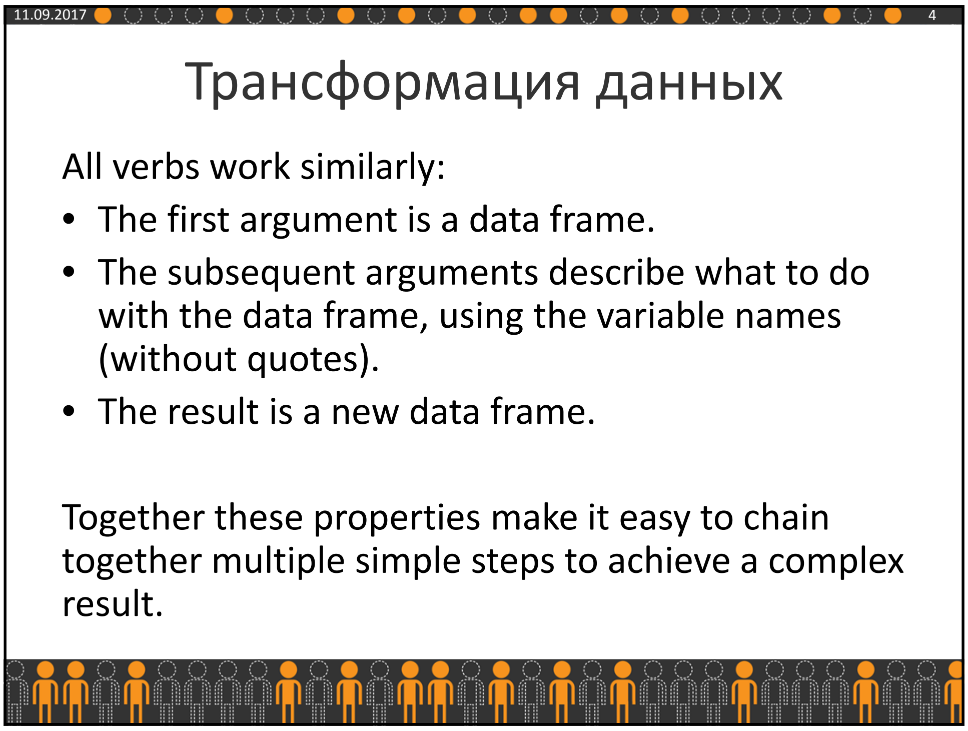
- Фильтрация (filter)
- Сортировка (arrange)
- Выборка (select)
- Модификация (mutate)
- Групповые операции
- Принципы Tidy Data и пакет `tidyr`



Трансформация данных

Ключевые функции пакета `dplyr` для трансформации данных:

- Pick observations by their values – `filter()`.
- Reorder the rows – `arrange()`.
- Pick variables by their names – `select()`.
- Create new variables with functions of existing variables – `mutate()`.
- Collapse many values down to a single summary – `summarise()`.
- These can all be used in conjunction with `group_by()` which changes the scope of each function from operating on the entire dataset to operating on it group-by-group.

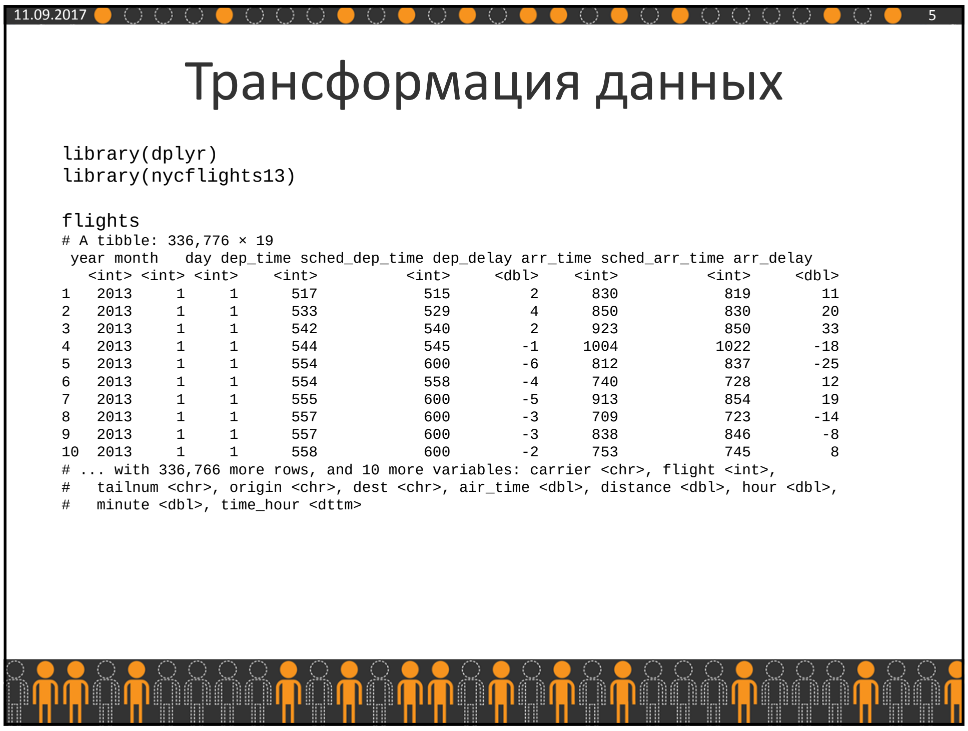


Трансформация данных

All verbs work similarly:

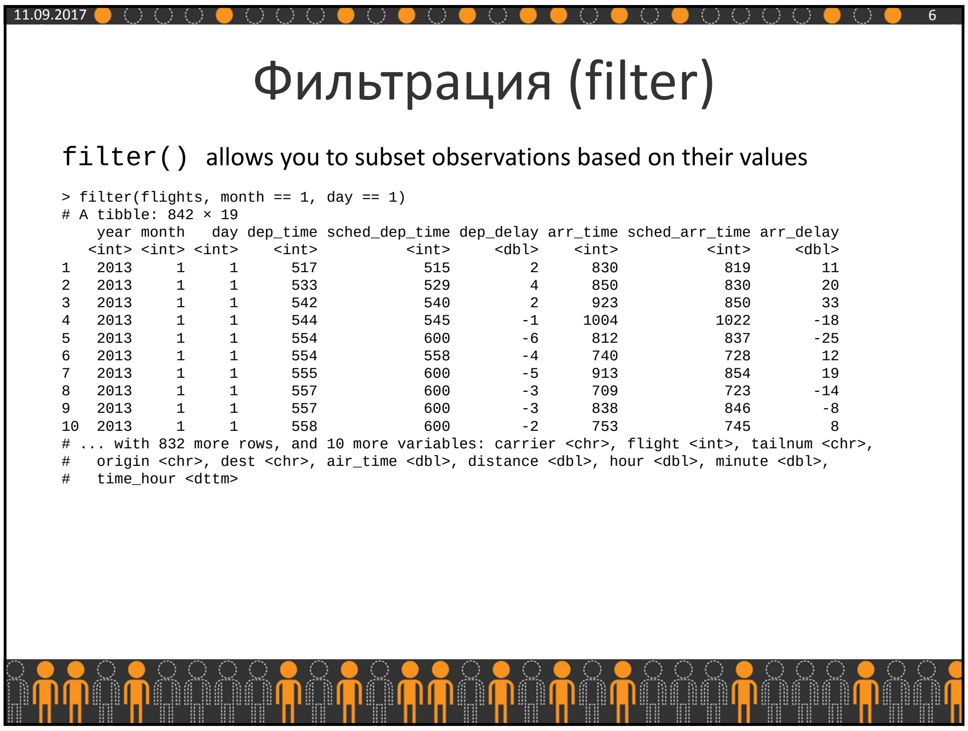
- The first argument is a data frame.
- The subsequent arguments describe what to do with the data frame, using the variable names (without quotes).
- The result is a new data frame.

Together these properties make it easy to chain together multiple simple steps to achieve a complex result.



Трансформация данных

```
library(dplyr)
library(nycflights13)

flights
# A tibble: 336,776 × 19
 year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl>
1   2013     1     1      517            515         2      830            819        11
2   2013     1     1      533            529         4      850            830        20
3   2013     1     1      542            540         2      923            850        33
4   2013     1     1      544            545        -1     1004           1022       -18
5   2013     1     1      554            600        -6      812            837       -25
6   2013     1     1      554            558        -4      740            728        12
7   2013     1     1      555            600        -5      913            854        19
8   2013     1     1      557            600        -3      709            723       -14
9   2013     1     1      557            600        -3      838            846        -8
10  2013     1     1      558            600        -2      753            745         8
# ... with 336,766 more rows, and 10 more variables: carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>
```



Фильтрация (filter)

`filter()` allows you to subset observations based on their values

```
> filter(flights, month == 1, day == 1)
# A tibble: 842 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl>
1   2013     1     1      517            515         2      830            819        11
2   2013     1     1      533            529         4      850            830        20
3   2013     1     1      542            540         2      923            850        33
4   2013     1     1      544            545        -1     1004           1022       -18
5   2013     1     1      554            600        -6      812            837       -25
6   2013     1     1      554            558        -4      740            728        12
7   2013     1     1      555            600        -5      913            854        19
8   2013     1     1      557            600        -3      709            723       -14
9   2013     1     1      557            600        -3      838            846        -8
10  2013     1     1      558            600        -2      753            745         8
# ... with 832 more rows, and 10 more variables: carrier <chr>, flight <int>, tailnum <chr>,
#   origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
#   time_hour <dttm>
```

Сортировка (arrange)

`arrange()` changes rows' order.

It takes a data frame and a set of column names (or more complicated expressions) to order by.

If you provide more than one column name, each additional column will be used to break ties in the values of preceding columns.

Сортировка (arrange)

```
arrange(flights, year, month, day)
# A tibble: 336,776 × 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl>
1   2013     1     1      517            515         2      830            819        11
2   2013     1     1      533            529         4      850            830        20
3   2013     1     1      542            540         2      923            850        33
4   2013     1     1      544            545        -1     1004           1022       -18
5   2013     1     1      554            600        -6      812            837       -25
6   2013     1     1      554            558        -4      740            728        12
7   2013     1     1      555            600        -5      913            854        19
8   2013     1     1      557            600        -3      709            723       -14
9   2013     1     1      557            600        -3      838            846        -8
10  2013     1     1      558            600        -2      753            745         8
# ... with 336,766 more rows, and 10 more variables: carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>
```

Выборка (select)

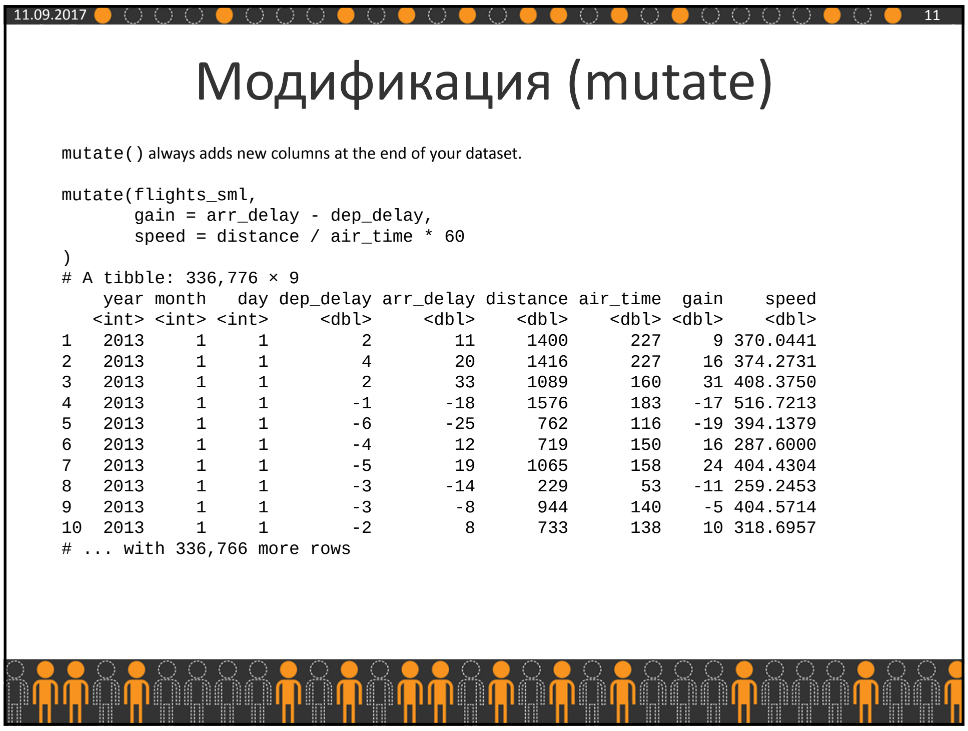
`select()` allows to zoom in on a useful subset basing on the names of the variables.

```
select(flights, year, month, day)
# A tibble: 336,776 × 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
```

There are a number of helper functions you can use within `select()`:

- `starts_with("abc")`: matches names that begin with “abc”
- `ends_with("xyz")`: matches names that end with “xyz”
- `contains("ijk")`: matches names that contain “ijk”
- `matches("(.)\\1")`: selects variables that match a regular expression. This one matches any variables that contain repeated characters
- `num_range("x", 1:3)`: matches x1, x2 and x3.

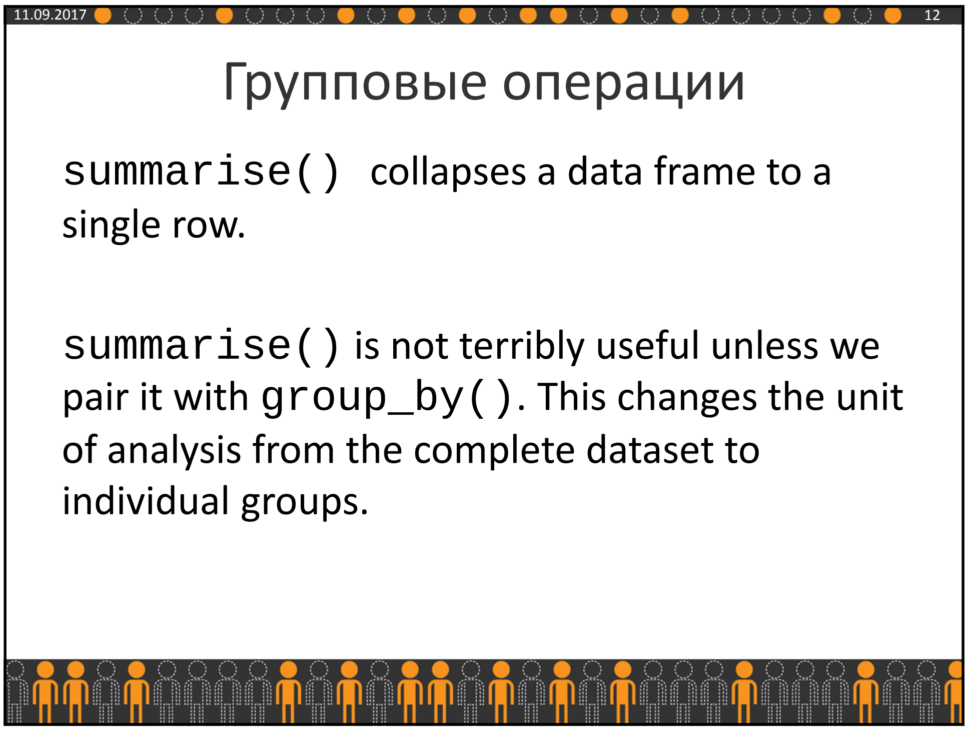
```
rename(flights, tail_num = tailnum)

select(flights, time_hour, air_time, everything())
```



Модификация (mutate)

`mutate()` always adds new columns at the end of your dataset.

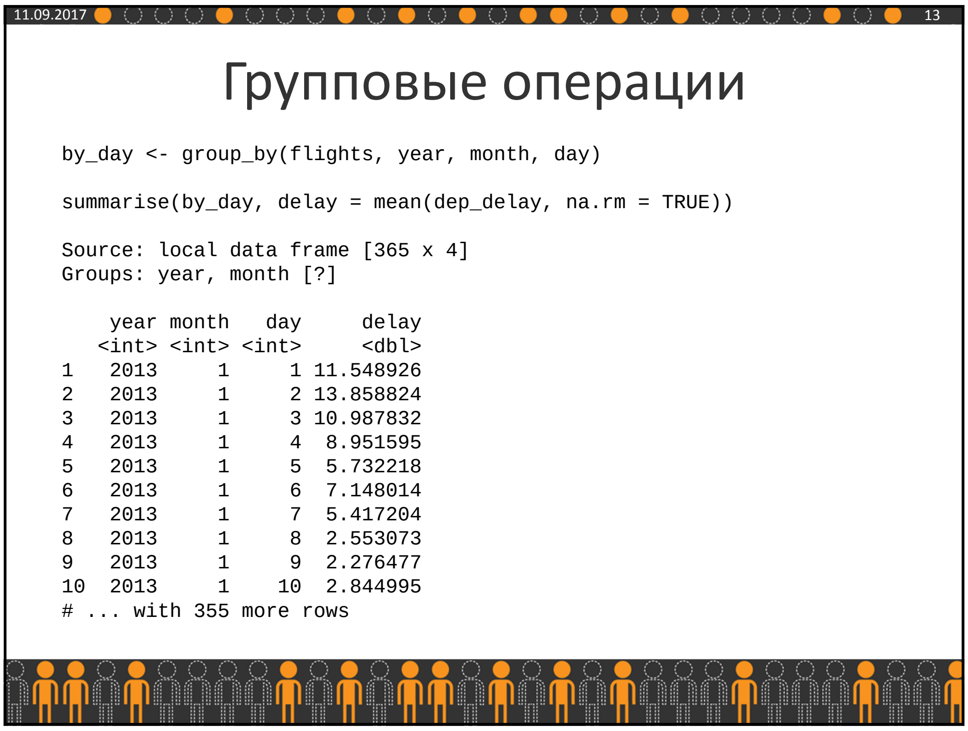
```
mutate(flights_sml,
       gain = arr_delay - dep_delay,
       speed = distance / air_time * 60
)
# A tibble: 336,776 × 9
    year month   day dep_delay arr_delay distance air_time  gain    speed
   <int> <int> <int>     <dbl>     <dbl>    <dbl>    <dbl> <dbl>    <dbl>
1   2013     1     1         2        11     1400      227     9 370.0441
2   2013     1     1         4        20     1416      227    16 374.2731
3   2013     1     1         2        33     1089      160    31 408.3750
4   2013     1     1        -1       -18     1576      183   -17 516.7213
5   2013     1     1        -6       -25      762      116   -19 394.1379
6   2013     1     1        -4        12      719      150    16 287.6000
7   2013     1     1        -5        19     1065      158    24 404.4304
8   2013     1     1        -3       -14      229       53   -11 259.2453
9   2013     1     1        -3        -8      944      140    -5 404.5714
10  2013     1     1        -2         8      733      138    10 318.6957
# ... with 336,766 more rows
```



Групповые операции

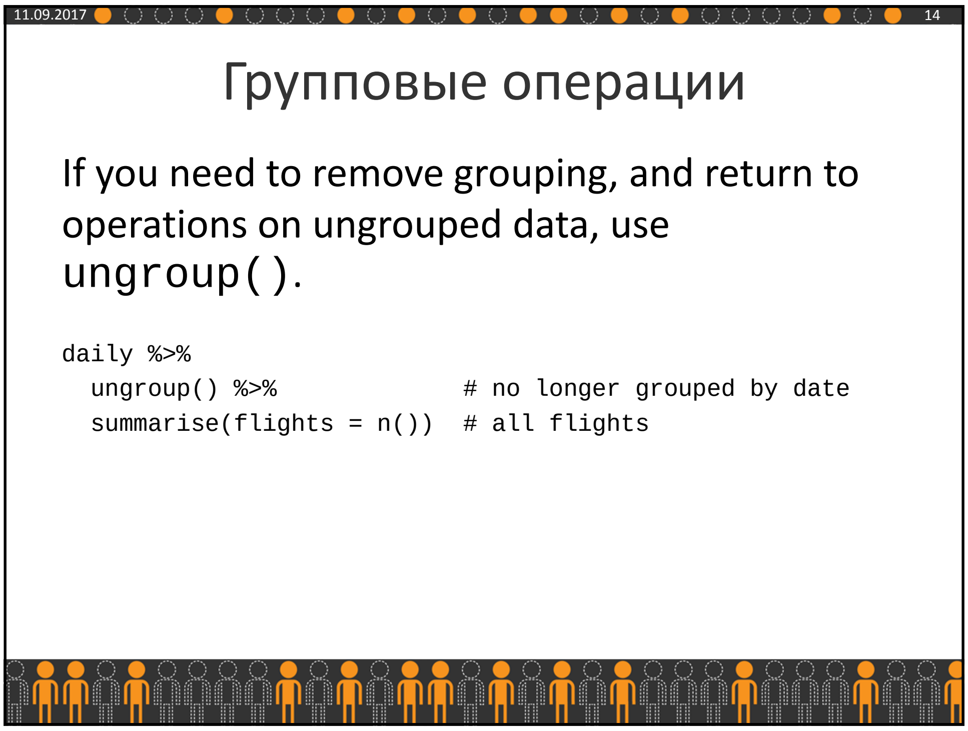
`summarise()` collapses a data frame to a single row.

`summarise()` is not terribly useful unless we pair it with `group_by()`. This changes the unit of analysis from the complete dataset to individual groups.



Групповые операции

```
by_day <- group_by(flights, year, month, day)

summarise(by_day, delay = mean(dep_delay, na.rm = TRUE))

Source: local data frame [365 x 4]
Groups: year, month [?]

    year month   day     delay
   <int> <int> <int>     <dbl>
1   2013     1     1 11.548926
2   2013     1     2 13.858824
3   2013     1     3 10.987832
4   2013     1     4  8.951595
5   2013     1     5  5.732218
6   2013     1     6  7.148014
7   2013     1     7  5.417204
8   2013     1     8  2.553073
9   2013     1     9  2.276477
10  2013     1    10  2.844995
# ... with 355 more rows
```



Групповые операции

If you need to remove grouping, and return to operations on ungrouped data, use `ungroup()`.

```
daily %>%
  ungroup() %>%             # no longer grouped by date
  summarise(flights = n())  # all flights
```

Принципы Tidy Data и пакет `tidyr`

Hadley Wickham is a statistician from New Zealand who is currently Chief Scientist at RStudio.

He is best known for his development of packages for R, that implement logics of data visualisation and data transformation, including tidyr, ggplot2, plyr, dplyr, and reshape2.

Wickham's data analysis packages for R are collectively known as the *'tidyverse'*.

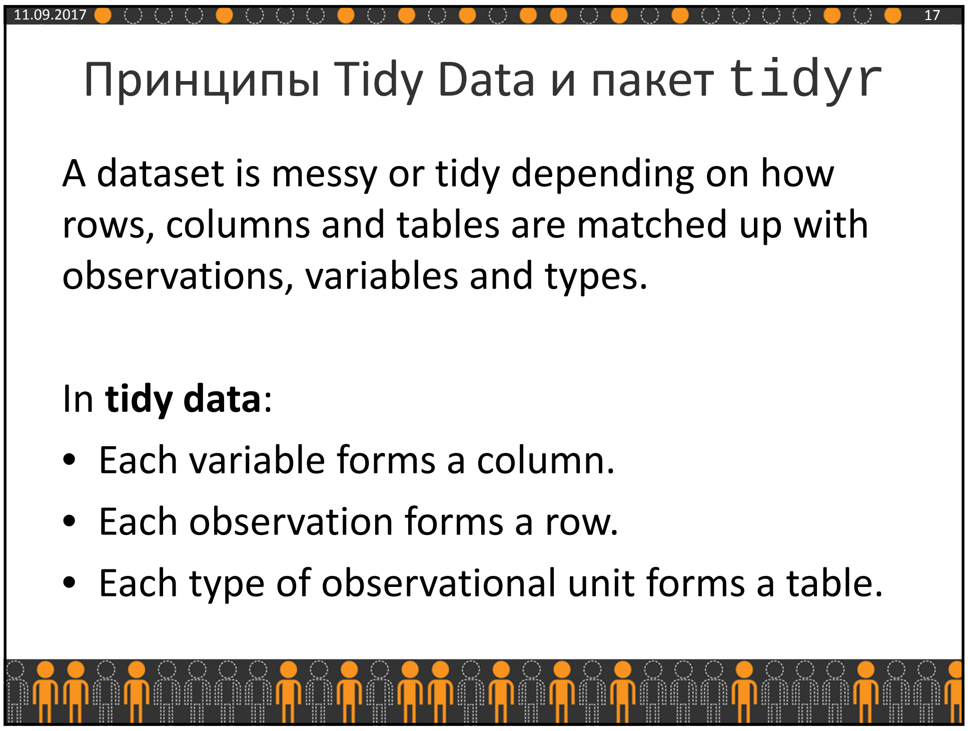
Принципы Tidy Data и пакет `tidyr`

A dataset is a collection of **values**, usually either numbers (if quantitative) or strings (if qualitative).

Values are organized in two ways. Every value belongs to a **variable** and an **observation**.

A variable contains all values that measure the same underlying attribute across units.

An observation contains all values measured on the same unit across attributes.

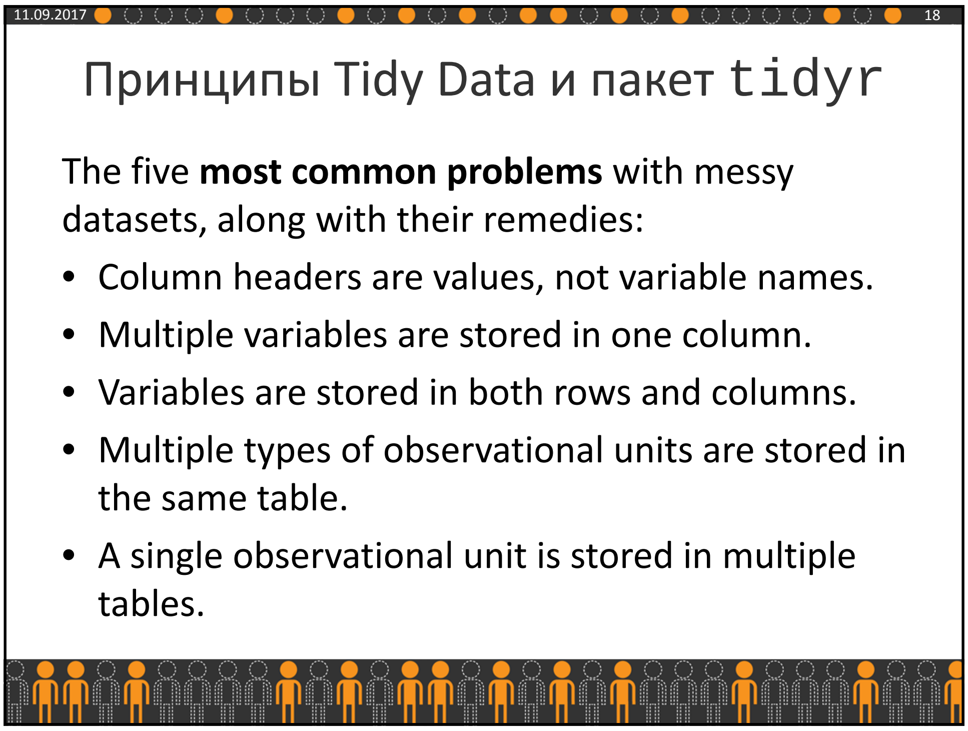


Принципы Tidy Data и пакет `tidyr`

A dataset is messy or tidy depending on how rows, columns and tables are matched up with observations, variables and types.

In **tidy data**:

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



Принципы Tidy Data и пакет `tidyr`

The five **most common problems** with messy datasets, along with their remedies:

- Column headers are values, not variable names.
- Multiple variables are stored in one column.
- Variables are stored in both rows and columns.
- Multiple types of observational units are stored in the same table.
- A single observational unit is stored in multiple tables.

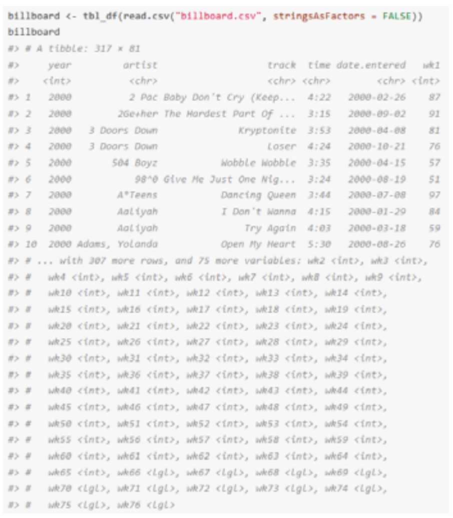
Column headers are values, not variable names

```
pew <- tbl_df(read.csv("pew.csv", stringsAsFactors = FALSE, check.names = FALSE))
pew
#> # A tibble: 18 × 11
#>                  religion `<$10k` `$10-20k` `$20-30k` `$30-40k` `$40-50k`
#>                     <chr>   <int>     <int>     <int>     <int>     <int>
#> 1                Agnostic      27        34        60        81        76
#> 2                 Atheist      12        27        37        52        35
#> 3                Buddhist      27        21        30        34        33
#> 4                Catholic     418       617       732       670       638
#> 5      Don’t know/refused      15        14        15        11        10
#> 6        Evangelical Prot     575       869      1064       982       881
#> 7                   Hindu       1         9         7         9        11
#> 8  Historically Black Prot    228       244       236       238       197
#> 9        Jehovah's Witness      20        27        24        24        21
#> 10                  Jewish      19        19        25        25        30
#> # ... with 8 more rows, and 5 more variables: `$50-75k` <int>,
#> #   `$75-100k` <int>, `$100-150k` <int>, `>150k` <int>, `Don't
#> #   know/refused` <int>
```

Column headers are values, not variable names

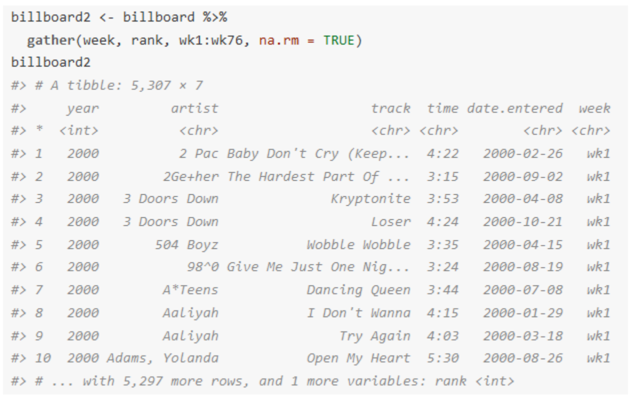
```
pew %>%
  gather(income, frequency, -religion)
#> # A tibble: 180 × 3
#>                  religion income frequency
#>                     <chr>  <chr>     <int>
#> 1                Agnostic  <$10k        27
#> 2                 Atheist  <$10k        12
#> 3                Buddhist  <$10k        27
#> 4                Catholic  <$10k       418
#> 5      Don’t know/refused  <$10k        15
#> 6        Evangelical Prot  <$10k       575
#> 7                   Hindu  <$10k         1
#> 8  Historically Black Prot <$10k       228
#> 9       Jehovah's Witness  <$10k        20
#> 10                 Jewish  <$10k        19
#> # ... with 170 more rows
```

Column headers are values, not variable names



```
billboard <- tbl_df(read.csv("billboard.csv", stringsAsFactors = FALSE))
billboard
#> # A tibble: 317 × 81
#>     year         artist                     track  time date.entered   wk1
#>    <int>          <chr>                     <chr> <chr>        <chr> <int>
#> 1   2000          2 Pac Baby Don't Cry (Keep...  4:22   2000-02-26    87
#> 2   2000        2Ge+her The Hardest Part Of ...  3:15   2000-09-02    91
#> 3   2000   3 Doors Down                Kryptonite  3:53   2000-04-08    81
#> 4   2000   3 Doors Down                     Loser  4:24   2000-10-21    76
#> 5   2000       504 Boyz             Wobble Wobble  3:35   2000-04-15    57
#> 6   2000           98^0 Give Me Just One Nig...  3:24   2000-08-19    51
#> 7   2000        A*Teens             Dancing Queen  3:44   2000-07-08    97
#> 8   2000        Aaliyah             I Don't Wanna  4:15   2000-01-29    84
#> 9   2000        Aaliyah                 Try Again  4:03   2000-03-18    59
#> 10  2000 Adams, Yolanda             Open My Heart  5:30   2000-08-26    76
#> # ... with 307 more rows, and 75 more variables: wk2 <int>, wk3 <int>,
#> #   wk4 <int>, wk5 <int>, wk6 <int>, wk7 <int>, wk8 <int>, wk9 <int>,
#> #   wk10 <int>, wk11 <int>, wk12 <int>, wk13 <int>, wk14 <int>,
#> #   wk15 <int>, wk16 <int>, wk17 <int>, wk18 <int>, wk19 <int>,
#> #   wk20 <int>, wk21 <int>, wk22 <int>, wk23 <int>, wk24 <int>,
#> #   wk25 <int>, wk26 <int>, wk27 <int>, wk28 <int>, wk29 <int>,
#> #   wk30 <int>, wk31 <int>, wk32 <int>, wk33 <int>, wk34 <int>,
#> #   wk35 <int>, wk36 <int>, wk37 <int>, wk38 <int>, wk39 <int>,
#> #   wk40 <int>, wk41 <int>, wk42 <int>, wk43 <int>, wk44 <int>,
#> #   wk45 <int>, wk46 <int>, wk47 <int>, wk48 <int>, wk49 <int>,
#> #   wk50 <int>, wk51 <int>, wk52 <int>, wk53 <int>, wk54 <int>,
#> #   wk55 <int>, wk56 <int>, wk57 <int>, wk58 <int>, wk59 <int>,
#> #   wk60 <int>, wk61 <int>, wk62 <int>, wk63 <int>, wk64 <int>,
#> #   wk65 <int>, wk66 <lgl>, wk67 <lgl>, wk68 <lgl>, wk69 <lgl>,
#> #   wk70 <lgl>, wk71 <lgl>, wk72 <lgl>, wk73 <lgl>, wk74 <lgl>,
#> #   wk75 <lgl>, wk76 <lgl>
```

Column headers are values, not variable names

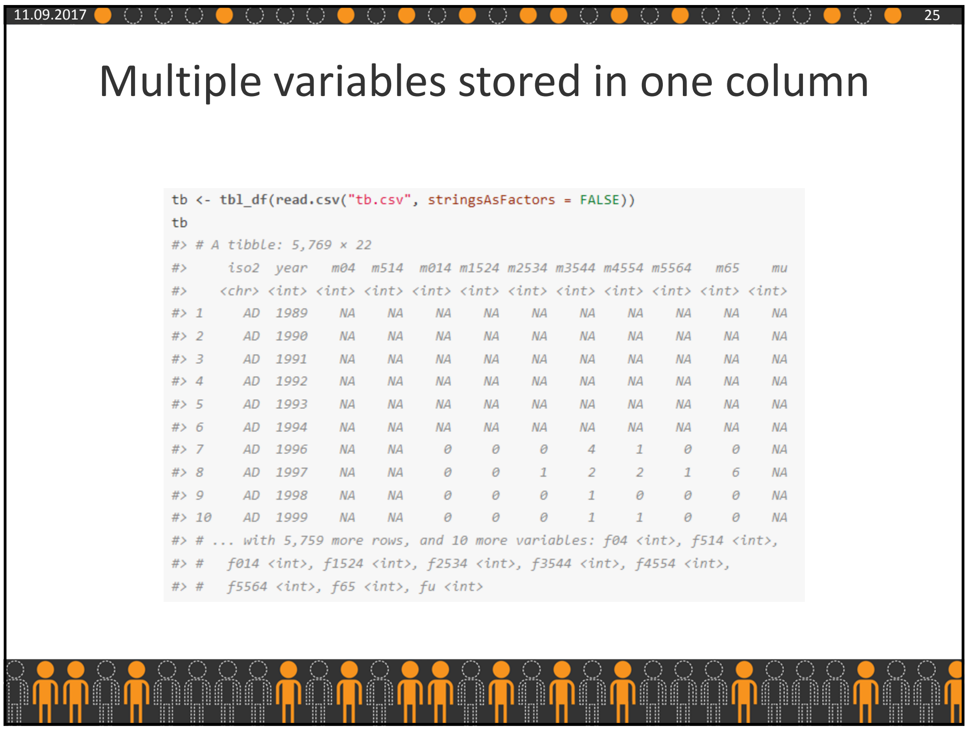


```
billboard2 <- billboard %>%
  gather(week, rank, wk1:wk76, na.rm = TRUE)
billboard2
#> # A tibble: 5,307 × 7
#>      year         artist                     track  time date.entered  week
#> *   <int>          <chr>                     <chr> <chr>        <chr> <chr>
#> 1    2000          2 Pac Baby Don't Cry (Keep...  4:22   2000-02-26   wk1
#> 2    2000        2Ge+her The Hardest Part Of ...  3:15   2000-09-02   wk1
#> 3    2000   3 Doors Down                Kryptonite  3:53   2000-04-08   wk1
#> 4    2000   3 Doors Down                     Loser  4:24   2000-10-21   wk1
#> 5    2000       504 Boyz             Wobble Wobble  3:35   2000-04-15   wk1
#> 6    2000           98^0 Give Me Just One Nig...  3:24   2000-08-19   wk1
#> 7    2000        A*Teens             Dancing Queen  3:44   2000-07-08   wk1
#> 8    2000        Aaliyah             I Don't Wanna  4:15   2000-01-29   wk1
#> 9    2000        Aaliyah                 Try Again  4:03   2000-03-18   wk1
#> 10   2000 Adams, Yolanda             Open My Heart  5:30   2000-08-26   wk1
#> # ... with 5,297 more rows, and 1 more variables: rank <int>
```

```
billboard3 <- billboard2 %>%
  mutate(
    week = extract_numeric(week),
    date = as.Date(date.entered) + 7 * (week - 1)) %>%
  select(-date.entered)
#> extract_numeric() is deprecated: please use readr::parse_number() instead
billboard3
#> # A tibble: 5,307 × 7
#>     year          artist                   track  time  week  rank
#>    <int>           <chr>                   <chr> <chr> <dbl> <int>
#> 1   2000           2 Pac Baby Don't Cry (Keep...  4:22     1    87
#> 2   2000         2Ge+her The Hardest Part Of ...  3:15     1    91
#> 3   2000    3 Doors Down              Kryptonite  3:53     1    81
#> 4   2000    3 Doors Down                   Loser  4:24     1    76
#> 5   2000        504 Boyz           Wobble Wobble  3:35     1    57
#> 6   2000            98^0 Give Me Just One Nig...  3:24     1    51
#> 7   2000         A*Teens           Dancing Queen  3:44     1    97
#> 8   2000         Aaliyah           I Don't Wanna  4:15     1    84
#> 9   2000         Aaliyah               Try Again  4:03     1    59
#> 10  2000 Adams, Yolanda            Open My Heart  5:30     1    76
#> # ... with 5,297 more rows, and 1 more variables: date <date>
```

Column headers are values, not variable names

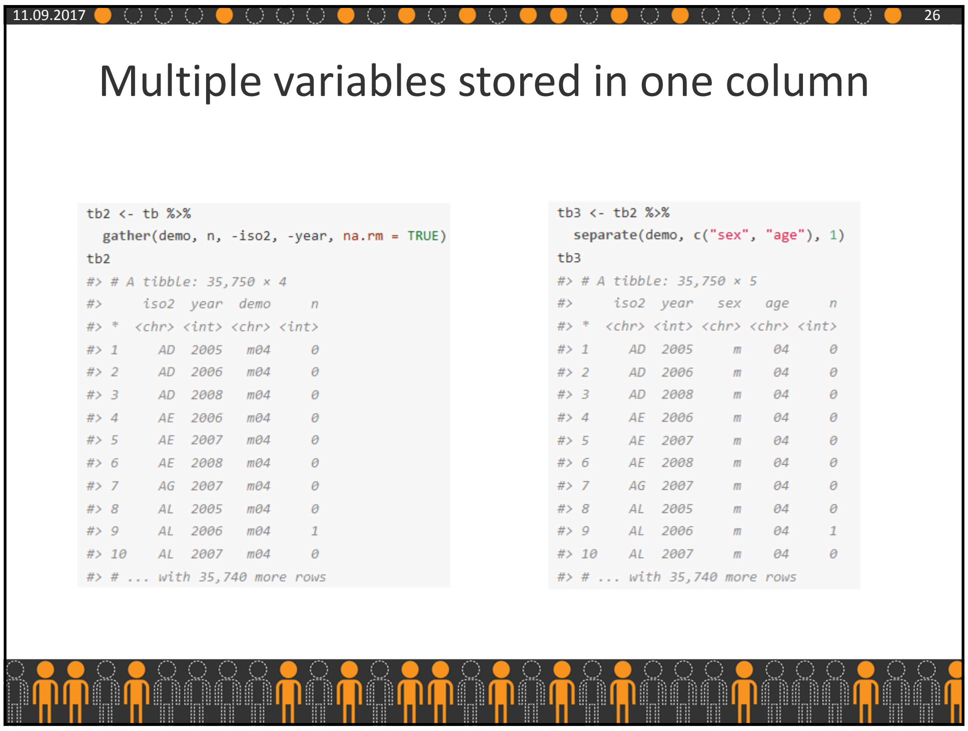
```
billboard3 %>% arrange(artist, track, week)
#> # A tibble: 5,307 × 7
#>     year  artist                    track  time  week  rank       date
#>    <int>  <chr>                     <chr> <chr> <dbl> <int>     <date>
#> 1   2000   2 Pac Baby Don't Cry (Keep...  4:22     1    87 2000-02-26
#> 2   2000   2 Pac Baby Don't Cry (Keep...  4:22     2    82 2000-03-04
#> 3   2000   2 Pac Baby Don't Cry (Keep...  4:22     3    72 2000-03-11
#> 4   2000   2 Pac Baby Don't Cry (Keep...  4:22     4    77 2000-03-18
#> 5   2000   2 Pac Baby Don't Cry (Keep...  4:22     5    87 2000-03-25
#> 6   2000   2 Pac Baby Don't Cry (Keep...  4:22     6    94 2000-04-01
#> 7   2000   2 Pac Baby Don't Cry (Keep...  4:22     7    99 2000-04-08
#> 8   2000 2Ge+her The Hardest Part Of ...  3:15     1    91 2000-09-02
#> 9   2000 2Ge+her The Hardest Part Of ...  3:15     2    87 2000-09-09
#> 10  2000 2Ge+her The Hardest Part Of ...  3:15     3    92 2000-09-16
#> # ... with 5,297 more rows
```



11.09.2017 25

Multiple variables stored in one column

```
tb <- tbl_df(read.csv("tb.csv", stringsAsFactors = FALSE))
tb
#> # A tibble: 5,769 × 22
#>      iso2  year   m04  m514  m014 m1524 m2534 m3544 m4554 m5564   m65    mu
#>     <chr> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
#> 1      AD  1989    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
#> 2      AD  1990    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
#> 3      AD  1991    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
#> 4      AD  1992    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
#> 5      AD  1993    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
#> 6      AD  1994    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
#> 7      AD  1996    NA    NA     0     0     0     4     1     0     0    NA
#> 8      AD  1997    NA    NA     0     0     1     2     2     1     6    NA
#> 9      AD  1998    NA    NA     0     0     0     1     0     0     0    NA
#> 10     AD  1999    NA    NA     0     0     0     1     1     0     0    NA
#> # ... with 5,759 more rows, and 10 more variables: f04 <int>, f514 <int>,
#> #   f014 <int>, f1524 <int>, f2534 <int>, f3544 <int>, f4554 <int>,
#> #   f5564 <int>, f65 <int>, fu <int>
```



11.09.2017 26

Multiple variables stored in one column

```
tb2 <- tb %>%
  gather(demo, n, -iso2, -year, na.rm = TRUE)
tb2
#> # A tibble: 35,750 × 4
#>      iso2  year  demo     n
#> *   <chr> <int> <chr> <int>
#> 1      AD  2005   m04     0
#> 2      AD  2006   m04     0
#> 3      AD  2008   m04     0
#> 4      AE  2006   m04     0
#> 5      AE  2007   m04     0
#> 6      AE  2008   m04     0
#> 7      AG  2007   m04     0
#> 8      AL  2005   m04     0
#> 9      AL  2006   m04     1
#> 10     AL  2007   m04     0
#> # ... with 35,740 more rows
```

```
tb3 <- tb2 %>%
  separate(demo, c("sex", "age"), 1)
tb3
#> # A tibble: 35,750 × 5
#>      iso2  year   sex   age     n
#> *   <chr> <int> <chr> <chr> <int>
#> 1      AD  2005     m    04     0
#> 2      AD  2006     m    04     0
#> 3      AD  2008     m    04     0
#> 4      AE  2006     m    04     0
#> 5      AE  2007     m    04     0
#> 6      AE  2008     m    04     0
#> 7      AG  2007     m    04     0
#> 8      AL  2005     m    04     0
#> 9      AL  2006     m    04     1
#> 10     AL  2007     m    04     0
#> # ... with 35,740 more rows
```

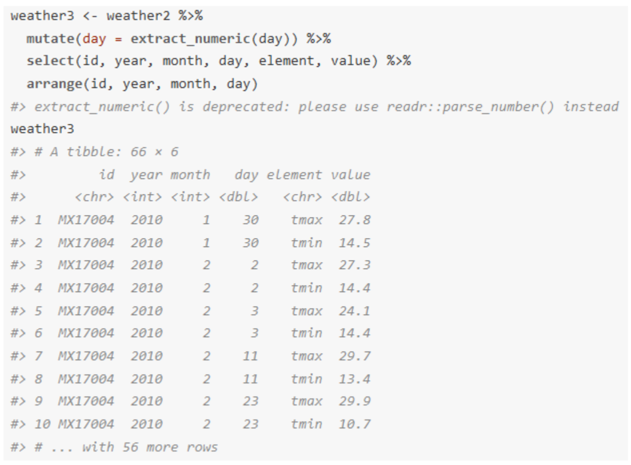
Variables are stored in both rows and columns

```
weather <- tbl_df(read.csv("weather.csv", stringsAsFactors = FALSE))
weather
#> # A tibble: 22 × 35
#>          id  year month element    d1    d2    d3    d4    d5    d6    d7
#>       <chr> <int> <int>   <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1  MX17004  2010     1    tmax    NA    NA    NA    NA    NA    NA    NA
#> 2  MX17004  2010     1    tmin    NA    NA    NA    NA    NA    NA    NA
#> 3  MX17004  2010     2    tmax    NA  27.3  24.1    NA    NA    NA    NA
#> 4  MX17004  2010     2    tmin    NA  14.4  14.4    NA    NA    NA    NA
#> 5  MX17004  2010     3    tmax    NA    NA    NA    NA  32.1    NA    NA
#> 6  MX17004  2010     3    tmin    NA    NA    NA    NA  14.2    NA    NA
#> 7  MX17004  2010     4    tmax    NA    NA    NA    NA    NA    NA    NA
#> 8  MX17004  2010     4    tmin    NA    NA    NA    NA    NA    NA    NA
#> 9  MX17004  2010     5    tmax    NA    NA    NA    NA    NA    NA    NA
#> 10 MX17004  2010     5    tmin    NA    NA    NA    NA    NA    NA    NA
#> # ... with 12 more rows, and 24 more variables: d8 <dbl>, d9 <lgl>,
#> #   d10 <dbl>, d11 <dbl>, d12 <lgl>, d13 <dbl>, d14 <dbl>, d15 <dbl>,
#> #   d16 <dbl>, d17 <dbl>, d18 <lgl>, d19 <lgl>, d20 <lgl>, d21 <lgl>,
#> #   d22 <lgl>, d23 <dbl>, d24 <lgl>, d25 <dbl>, d26 <dbl>, d27 <dbl>,
#> #   d28 <dbl>, d29 <dbl>, d30 <dbl>, d31 <dbl>
```

Variables are stored in both rows and columns

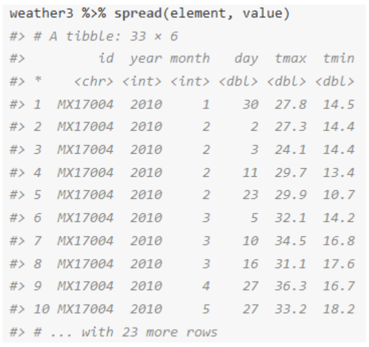
```
weather2 <- weather %>%
  gather(day, value, d1:d31, na.rm = TRUE)
weather2
#> # A tibble: 66 × 6
#>          id  year month element   day value
#> *     <chr> <int> <int>   <chr> <chr> <dbl>
#> 1  MX17004  2010    12    tmax    d1  29.9
#> 2  MX17004  2010    12    tmin    d1  13.8
#> 3  MX17004  2010     2    tmax    d2  27.3
#> 4  MX17004  2010     2    tmin    d2  14.4
#> 5  MX17004  2010    11    tmax    d2  31.3
#> 6  MX17004  2010    11    tmin    d2  16.3
#> 7  MX17004  2010     2    tmax    d3  24.1
#> 8  MX17004  2010     2    tmin    d3  14.4
#> 9  MX17004  2010     7    tmax    d3  28.6
#> 10 MX17004  2010     7    tmin    d3  17.5
#> # ... with 56 more rows
```

Variables are stored in both rows and columns

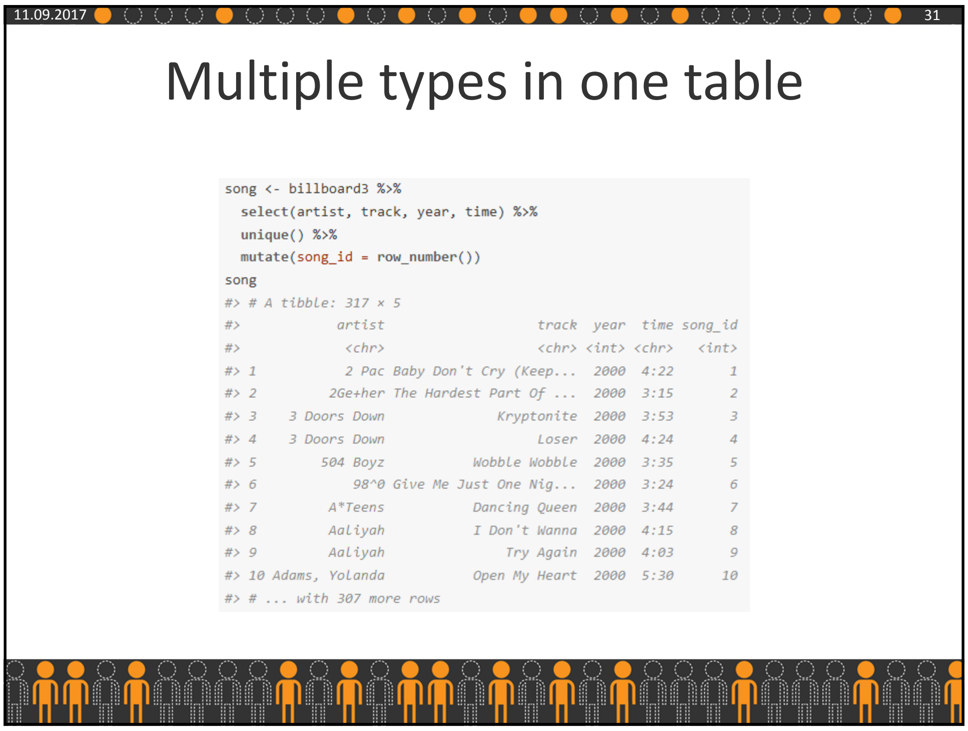


```
weather3 <- weather2 %>%
  mutate(day = extract_numeric(day)) %>%
  select(id, year, month, day, element, value) %>%
  arrange(id, year, month, day)
#> extract_numeric() is deprecated: please use readr::parse_number() instead
weather3
#> # A tibble: 66 × 6
#>          id  year month   day element value
#>       <chr> <int> <int> <dbl>   <chr> <dbl>
#> 1  MX17004  2010     1    30    tmax  27.8
#> 2  MX17004  2010     1    30    tmin  14.5
#> 3  MX17004  2010     2     2    tmax  27.3
#> 4  MX17004  2010     2     2    tmin  14.4
#> 5  MX17004  2010     2     3    tmax  24.1
#> 6  MX17004  2010     2     3    tmin  14.4
#> 7  MX17004  2010     2    11    tmax  29.7
#> 8  MX17004  2010     2    11    tmin  13.4
#> 9  MX17004  2010     2    23    tmax  29.9
#> 10 MX17004  2010     2    23    tmin  10.7
#> # ... with 56 more rows
```

Variables are stored in both rows and columns

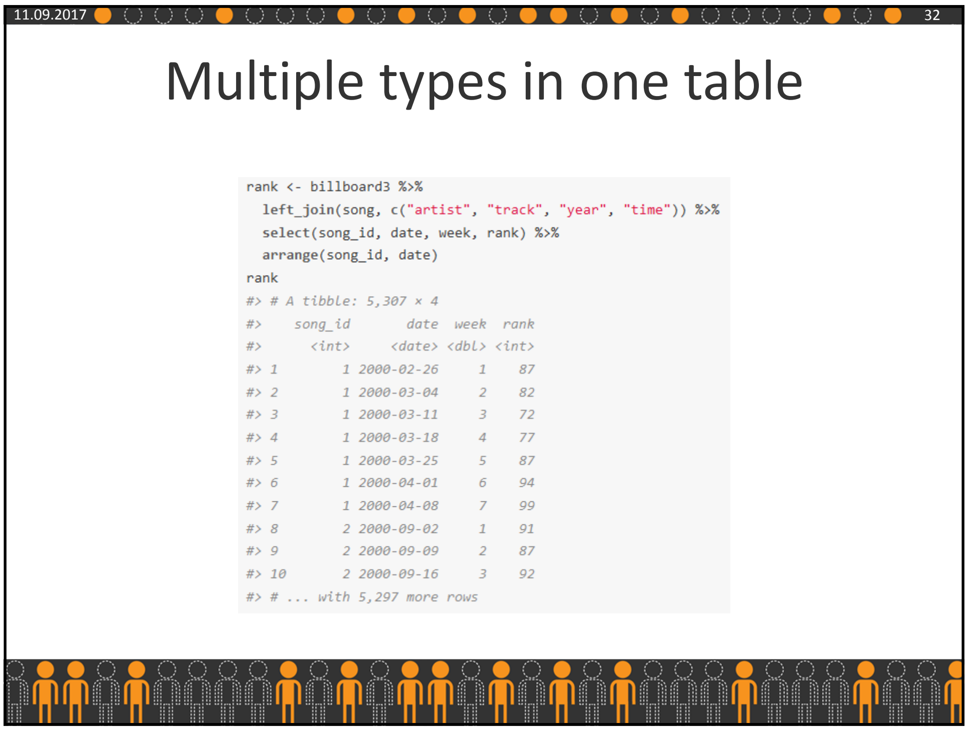


```
weather3 %>% spread(element, value)
#> # A tibble: 33 × 6
#>          id  year month   day  tmax  tmin
#> *     <chr> <int> <int> <dbl> <dbl> <dbl>
#> 1  MX17004  2010     1    30  27.8  14.5
#> 2  MX17004  2010     2     2  27.3  14.4
#> 3  MX17004  2010     2     3  24.1  14.4
#> 4  MX17004  2010     2    11  29.7  13.4
#> 5  MX17004  2010     2    23  29.9  10.7
#> 6  MX17004  2010     3     5  32.1  14.2
#> 7  MX17004  2010     3    10  34.5  16.8
#> 8  MX17004  2010     3    16  31.1  17.6
#> 9  MX17004  2010     4    27  36.3  16.7
#> 10 MX17004  2010     5    27  33.2  18.2
#> # ... with 23 more rows
```



Multiple types in one table

```
song <- billboard3 %>%
  select(artist, track, year, time) %>%
  unique() %>%
  mutate(song_id = row_number())
song
#> # A tibble: 317 × 5
#>             artist                   track  year  time song_id
#>              <chr>                   <chr> <int> <chr>   <int>
#> 1            2 Pac Baby Don't Cry (Keep...  2000  4:22       1
#> 2          2Ge+her The Hardest Part Of ...  2000  3:15       2
#> 3     3 Doors Down              Kryptonite  2000  3:53       3
#> 4     3 Doors Down                   Loser  2000  4:24       4
#> 5         504 Boyz           Wobble Wobble  2000  3:35       5
#> 6             98^0 Give Me Just One Nig...  2000  3:24       6
#> 7          A*Teens           Dancing Queen  2000  3:44       7
#> 8          Aaliyah           I Don't Wanna  2000  4:15       8
#> 9          Aaliyah               Try Again  2000  4:03       9
#> 10 Adams, Yolanda           Open My Heart  2000  5:30      10
#> # ... with 307 more rows
```



Multiple types in one table

```
rank <- billboard3 %>%
  left_join(song, c("artist", "track", "year", "time")) %>%
  select(song_id, date, week, rank) %>%
  arrange(song_id, date)
rank
#> # A tibble: 5,307 × 4
#>    song_id       date  week  rank
#>      <int>     <date> <dbl> <int>
#> 1        1 2000-02-26     1    87
#> 2        1 2000-03-04     2    82
#> 3        1 2000-03-11     3    72
#> 4        1 2000-03-18     4    77
#> 5        1 2000-03-25     5    87
#> 6        1 2000-04-01     6    94
#> 7        1 2000-04-08     7    99
#> 8        2 2000-09-02     1    91
#> 9        2 2000-09-09     2    87
#> 10       2 2000-09-16     3    92
#> # ... with 5,297 more rows
```

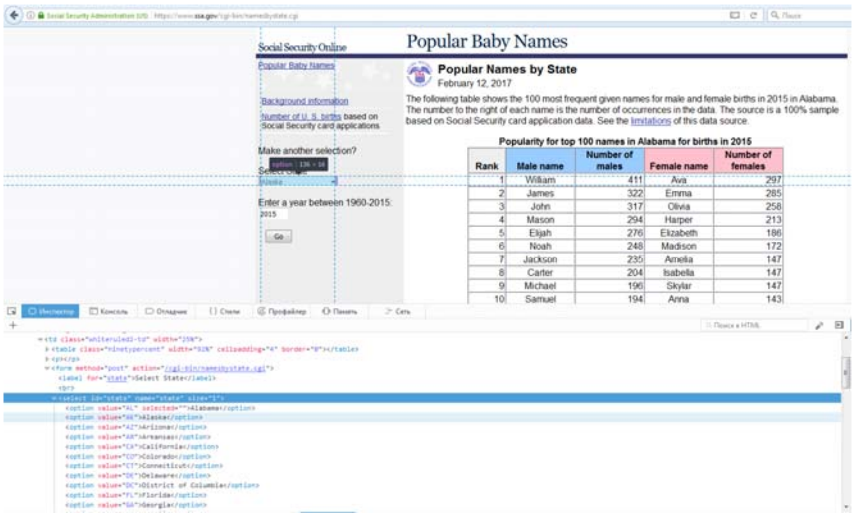
One type in multiple tables

It's also common to find data values about a single type of observational unit spread out over multiple tables or files. These tables and files are often split up by another variable, so that each represents a single year, person, or location.

As long as the format for individual records is consistent, this is an easy problem to fix:

- Read the files into a list of tables.
- For each table, add a new column that records the original file name (the file name is often the value of an important variable).
- Combine all tables into a single table.

One type in multiple tables



Social Security Online

Popular Baby Names

Background information

Number of U. S. births based on Social Security card applications

Make another selection?

Select State

Enter a year between 1960-2015:

2015

Go

Popular Names by State

February 12, 2017

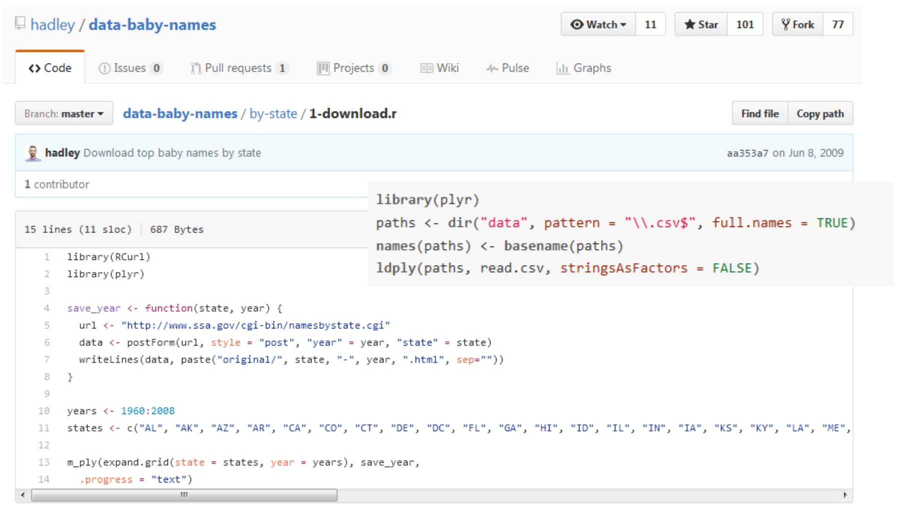
The following table shows the 100 most frequent given names for male and female births in 2015 in Alabama. The number to the right of each name is the number of occurrences in the data. The source is a 100% sample based on Social Security card application data. See the limitations of this data source.

Popularity for top 100 names in Alabama for births in 2015

Rank	Male name	Number of males	Female name	Number of females
1	William	411	Ava	297
2	James	322	Emma	285
3	John	317	Olivia	258
4	Mason	294	Harper	213
5	Elijah	276	Elizabeth	186
6	Noah	248	Madison	172
7	Jackson	235	Amelia	147
8	Carter	204	Isabella	147
9	Michael	196	Skylar	147
10	Samuel	194	Anna	143

```
▾ <td class="whiteruled2-td" width="25%">
  ▸ <table class="ninetypercent" width="92%" cellpadding="4" border="0"></table>
  ▸ <p></p>
  ▾ <form method="post" action="/cgi-bin/namesbystate.cgi">
      <label for="state">Select State</label>
      <br>
    ▾ <select id="state" name="state" size="1">
        <option value="AL" selected="">Alabama</option>
        <option value="AK">Alaska</option>
        <option value="AZ">Arizona</option>
        <option value="AR">Arkansas</option>
        <option value="CA">California</option>
        <option value="CO">Colorado</option>
        <option value="CT">Connecticut</option>
        <option value="DE">Delaware</option>
        <option value="DC">District of Columbia</option>
        <option value="FL">Florida</option>
        <option value="GA">Georgia</option>
```

One type in multiple tables



```r
library(RCurl)
library(plyr)

save_year <- function(state, year) {
  url <- "http://www.ssa.gov/cgi-bin/namesbystate.cgi"
  data <- postForm(url, style = "post", "year" = year, "state" = state)
  writeLines(data, paste("original/", state, "-", year, ".html", sep=""))
}

years <- 1960:2008
states <- c("AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "DC", "FL", "GA", "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME",

m_ply(expand.grid(state = states, year = years), save_year,
  .progress = "text")
```

```r
library(plyr)
paths <- dir("data", pattern = "\\.csv$", full.names = TRUE)
names(paths) <- basename(paths)
ldply(paths, read.csv, stringsAsFactors = FALSE)
```

Основная литература

- Grolemund, G. R for Data Science [Electronic resource] / Garrett Grolemund, Hadley Wickham. – 2016. – Mode of access: http://r4ds.had.co.nz/index.html. – Date of access: 01.09.2016.
- Wickham, H. Tidy data [Electronic resource] / Hadley Wickham // The Journal of Statistical Software. – Vol. 59. – 2014 . – Mode of access: http://vita.had.co.nz/papers/tidy-data.html. – Date of access: 01.09.2016.
- Wickham, H. Tidy data [Electronic resource]. – 2017. – Mode of access: ftp://cran.r-project.org/pub/R/web/packages/tidyr/vignettes/tidy-data.html. – Date of access: 01.02.2017.
- Ogurtsov, A. Tidy data (Перевод) [Electronic resource]. – 2015. – Mode of access: http://biostat-r.blogspot.com.by/2016/01/tidy-data.html. – Date of access: 01.02.2017.