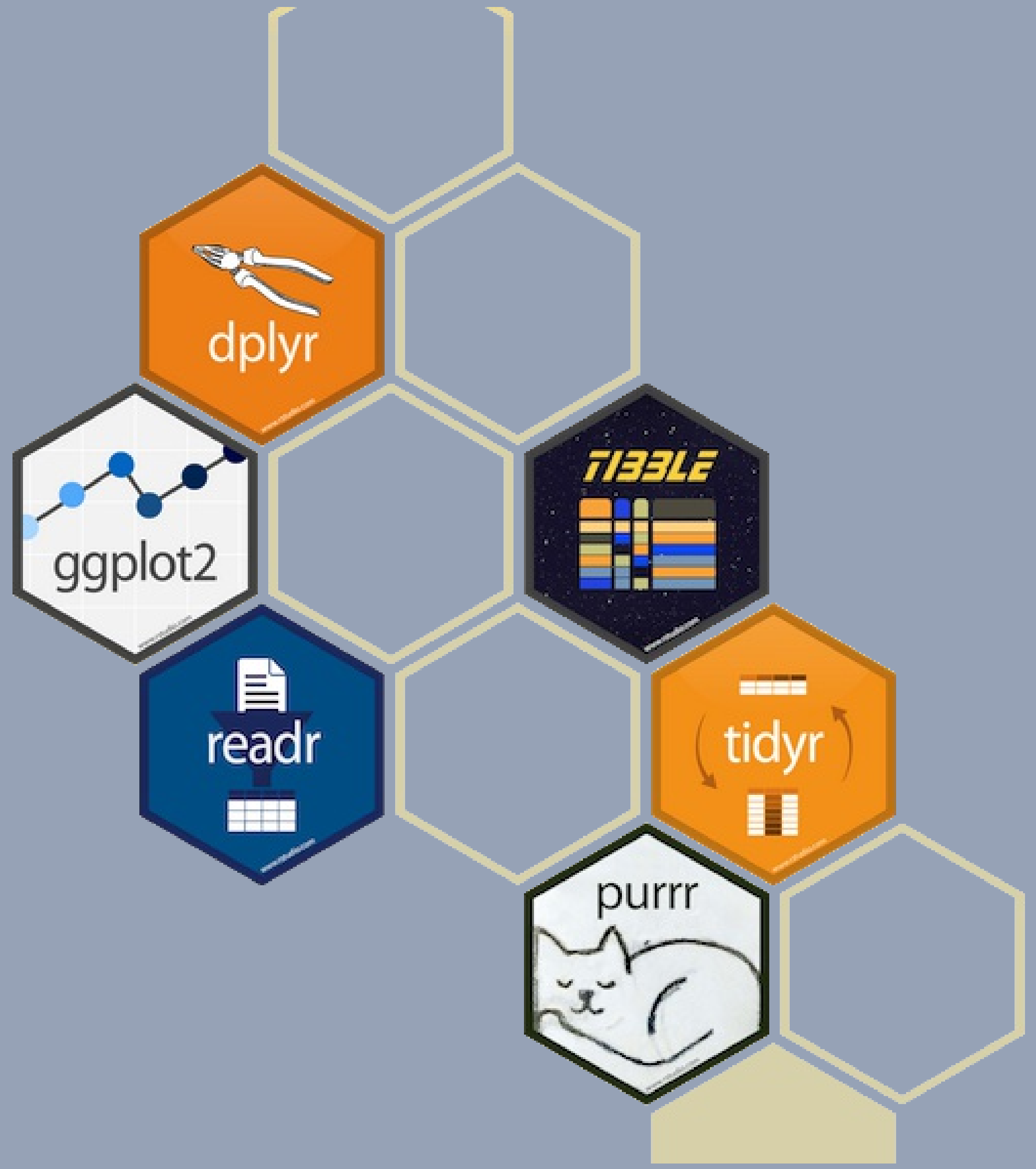
An Introduction to the Tidyverse

David Zimmermann
R User Group Oxford
02 October 2017



Outline

1. An Overview of the Tidyverse
2. Pipes %>%
3. Tidy Data and tidyr
4. Data Munging and dplyr

An Overview of the Tidyverse

A collection of libraries for data science with a consistent design and API

tibble	data.frame extension
tidyr	reshaping data
dplyr	data munging and manipulation
readr	read and write to text-files (csv, txt, etc)
ggplot2	data visualization
purrr	functional programming
…	

Lives here https://www.tidyverse.org/ and here https://github.com/tidyverse/tidyverse

Usage

Install once

```
install.packages("tidyverse")
```

Usage in Scripts

```
library(tidyverse)

# and you are good to go
df <- data_frame(id = 1:3, name = c("Alice", "Bob", "Charlie"))
df
```

```
# A tibble: 3 x 2
     id    name
  <int>   <chr>
1     1   Alice
2     2     Bob
3     3 Charlie
```

data?frame

```r
df_dot        <- data.frame(id = 1:2, name = c("Alice", "Bob"))
df_underscore <- data_frame(id = 1:2, name = c("Alice", "Bob"))

df_dot
```

```
  id  name
1  1 Alice
2  2   Bob
```

```r
df_underscore
```

```
# A tibble: 2 x 2
     id  name
  <int> <chr>
1     1 Alice
2     2   Bob
```

data?frame cont’d

data_frame()/tibble():

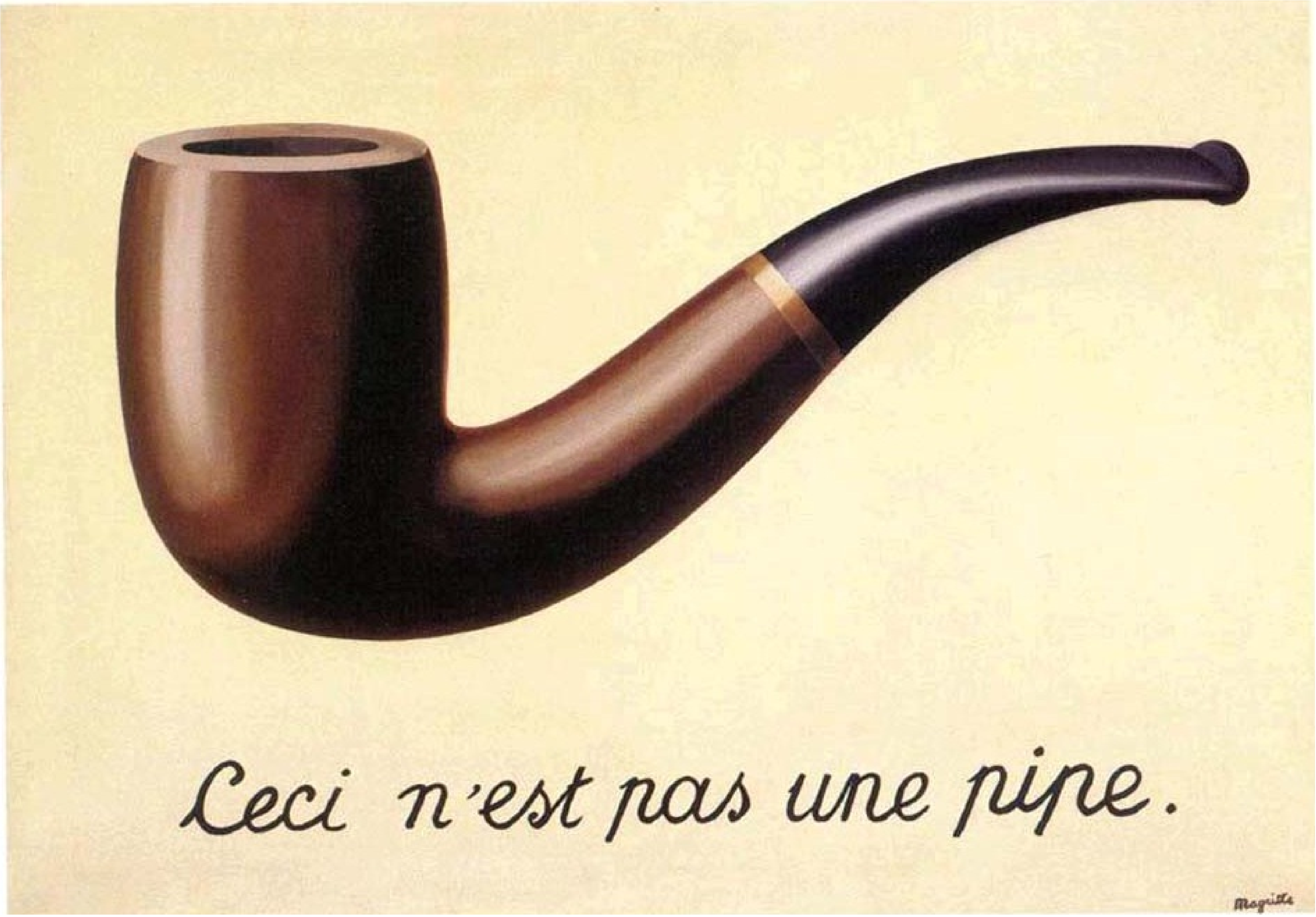
- doesn't convert strings to factors and allows for lists as elements!
- prints more information in a nicer format (try with a dataset of >10 columns/rows)
- is still a data.frame, thus works with older code!

```
str(df_dot)
```

```
'data.frame':   2 obs. of  2 variables:
 $ id  : int  1 2
 $ name: Factor w/ 2 levels "Alice","Bob": 1 2
```

```
str(df_underscore)
```

```
Classes 'tbl_df', 'tbl' and 'data.frame':   2 obs. of  2 variables:
```


Ceci n'est pas une pipe.
Magritte

Pipes

Then:

```
value2 <- foo(value)
value3 <- bar(value2)

# Or
bar(foo(value))
```

Pipes

Then:

```
value2 <- foo(value)
value3 <- bar(value2)

# Or
bar(foo(value))
```

Take the output (value) of one function and use it in another function.

Now using the pipe-operator **%>%** (read it as “then”):

```
value %>% foo() %>% bar()
# or with better formattting
value %>%
  foo() %>%
  bar()
```

Pipes cont'd

In general

```
f(g(h(x)))
# becomes
x %>% h() %>% g() %>% f()
```

Pipes cont’d

In general

```
f(g(h(x)))
# becomes
x %>% h() %>% g() %>% f()
```

Example

Non-pipe

```
flights_dec <- filter(flights, month == 12)
flights_dec_grouped <- group_by(flights_dec, day)
summarise(flights_dec_grouped, mean_delay = mean(arr_delay))
```

vs. pipe

```
flights %>%
  filter(month == 12) %>%
  group_by(day) %>%
  summarise(mean_delay = mean(arr_delay))
```

Pipes cont'd

Named arguments and arg-numbers using the .-argument

```
foo(w, x)
# becomes
x %>% foo(w, .)

foo(w, x, y, z)
# becomes
x %>% foo(w, ., y, z)

# Named Arguments
x %>% foo(w, x = .)
```

More information: http://r4ds.had.co.nz/pipes.html

Tidy Data

How should we store data in a “good” format?

Say, we have sales data for

- Alice
- Bob
- Charlie

For the years 2010 and 2011

Tidy Data cont'd

Wide-format

Year	Alice	Bob	Charlie
2010	105	100	90
2011	110	97	95

Long-format

Name	Year	Sales
Alice	2010	105
Alice	2011	110
Bob	2010	100
Bob	2011	97
Charlie	2010	90
Charlie	2011	95

Tidy Data cont'd

Wide-format

Year	Alice	Bob	Charlie
2010	105	100	90
2011	110	97	95

Long/Tidy-Data

Name	Year	Sales
Alice	2010	105
Alice	2011	110
Bob	2010	100
Bob	2011	97
Charlie	2010	90
Charlie	2011	95

Ease of use vs. size of data?

What happens if we receive data for 2012?

What happens if Dave joins?

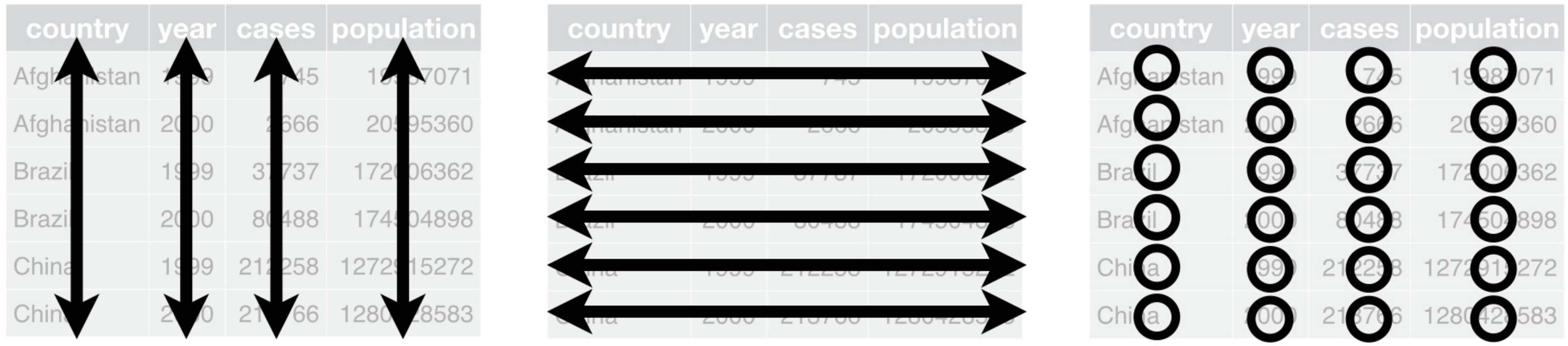
What happens if we receive the number of visits per salesperson?

What happens if Alice and Charlie belong to Company X, but Bob to Company Y?

Formalised Tidy Data

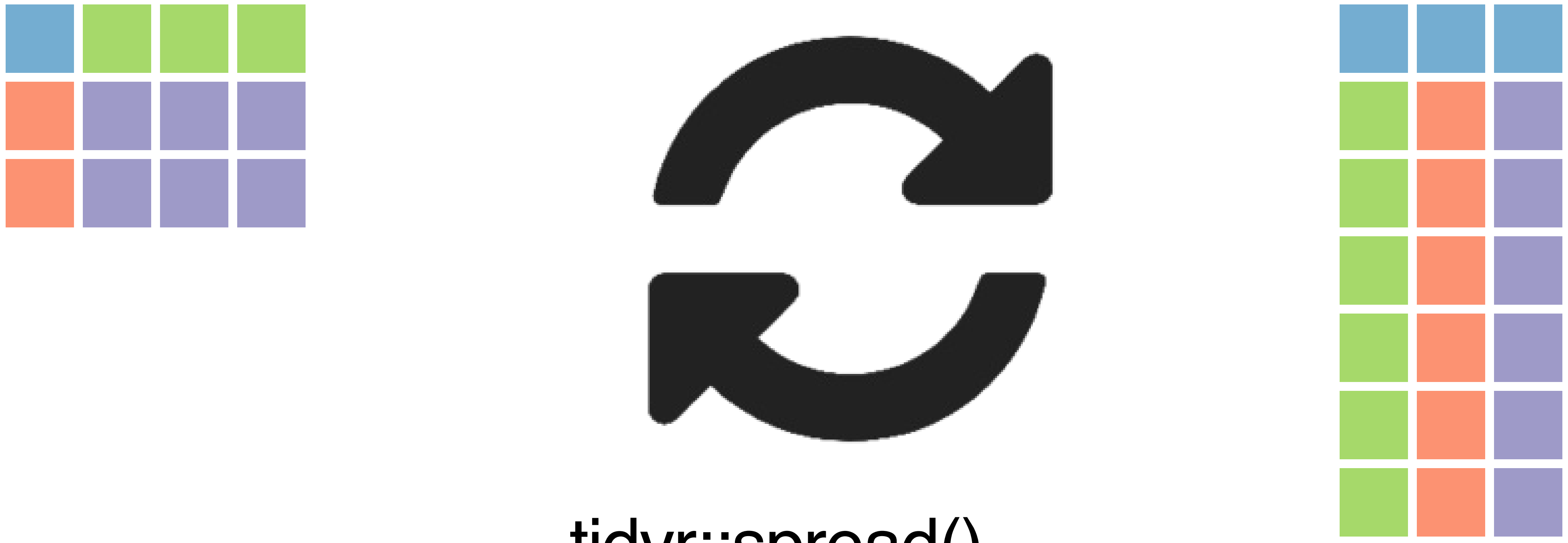
One Variable per **Column**

One Observation per **Row**



variables observations values

Reshaping Data using TidyR (Tidyverse)



Wide-Data

tidyr::gather()

Long/Tidy-Data

tidyr::spread()

Reshaping Data

Recreate the Data

```
wide <- data_frame(year    = c(2010, 2011),
                   Alice   = c(105, 110),
                   Bob     = c(100, 97),
                   Charlie = c(90, 95))
wide
```

```
# A tibble: 2 x 4
   year Alice   Bob Charlie
  <dbl> <dbl> <dbl>   <dbl>
1  2010   105   100      90
2  2011   110    97      95
```

Reshaping Data Wide to Long

gather() the data to a long/tidy-format

```
long <- wide %>% gather(key = "name", value = "sales", -year)
long
```

```
# A tibble: 6 x 3
   year    name sales
  <dbl>   <chr> <dbl>
1  2010   Alice   105
2  2011   Alice   110
3  2010     Bob   100
4  2011     Bob    97
5  2010 Charlie    90
6  2011 Charlie    95
```

- **key**: the name of the (future) variable that holds the key (the values in the header)
- **value**: the name of the (future) variable that holds the values (in the body)

Reshaping Data Long to Wide

spread() the data to a wide-format

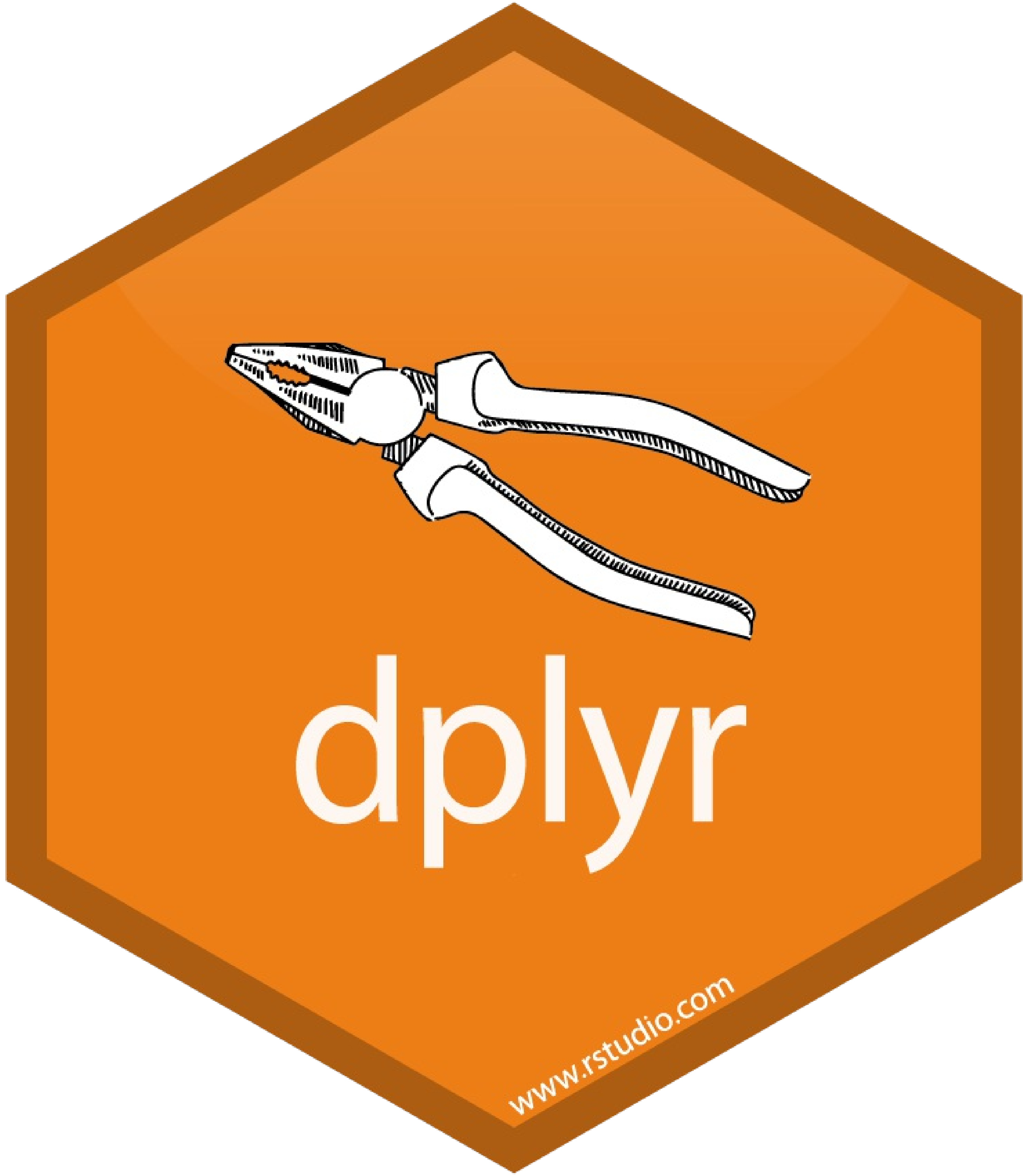
```
wide2 <- long %>% spread(key = "name", value = "sales")
wide2
```

```
# A tibble: 2 x 4
   year Alice   Bob Charlie
* <dbl> <dbl> <dbl>   <dbl>
1  2010   105   100      90
2  2011   110    97      95
```

- **key**: the name of the variable that will go in the header
- **value**: the name of the variable that will go to the body

```
identical(wide, wide2)
```

```
[1] TRUE
```


dplyr
www.rstudio.com

dplyr Overview

Grammer of Data Manipulation

Reasonably fast and consistent API

- filter() filter observations / rows
- arrange() arrange (sort) the dataset by a column
- select() select variables / columns
- mutate() change or create a variable
- summarise() summarise the dataset to a single row
- + group_by() operate by a grouping-variable

Lives here http://dplyr.tidyverse.org/ and here https://github.com/tidyverse/dplyr

Dataset

```r
# install.packages("nycflights13")
library(nycflights13)
flights %>% glimpse()
```

```
Observations: 336,776
Variables: 19
$ year           <int> 2013, 2013, 2013, 2013, 2013, 2013,...
$ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
$ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
$ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, ...
$ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, ...
$ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2...
$ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913,...
$ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854,...
$ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, ...
$ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA",...
$ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 5...
$ tailnum        <chr> "N14228", "N24211", "N619AA", "N804...
$ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", ...
$ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", ...
$ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, ...
$ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1...
$ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6,...
$ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, ...
```

Filter Observations

filter() the rows of a dataset for certain values only

Input Data

var		
x		
y		
z		
x		
z		
x		

```
df %>%
   filter(var == "x")
```

Output Data

var		
x		
x		
x		

Filter Observations cont’d

Query: Find the long-distance flights in the spring of 2013.

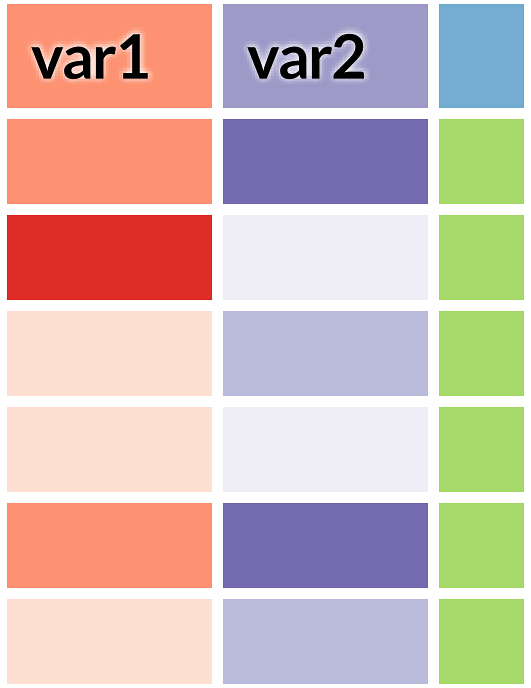
```
flights %>%
  filter(month <= 03 & distance > 2500)
```

```
# A tibble: 2,916 x 19
    year month   day dep_time sched_dep_time dep_delay
   <int> <int> <int>    <int>          <int>     <dbl>
 1  2013     1     1      558            600        -2
 2  2013     1     1      611            600        11
 3  2013     1     1      655            700        -5
 4  2013     1     1      729            730        -1
 5  2013     1     1      734            737        -3
 6  2013     1     1      745            745         0
 7  2013     1     1      746            746         0
 8  2013     1     1      803            800         3
 9  2013     1     1      826            817         9
10  2013     1     1      857            900        -3
# ... with 2,906 more rows, and 13 more variables:
#   arr_time <int>, sched_arr_time <int>, arr_delay <dbl>,
#   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>
```

Arrange Observations

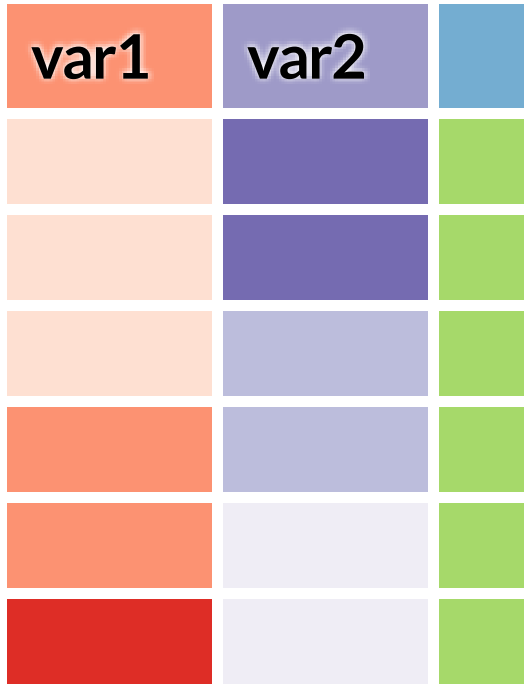
arrange() the rows of a dataset for certain variables / columns

Input Data



```
df %>%
   arrange(var1,
           -var2)
```

Output Data



Arrange Observations cont’d

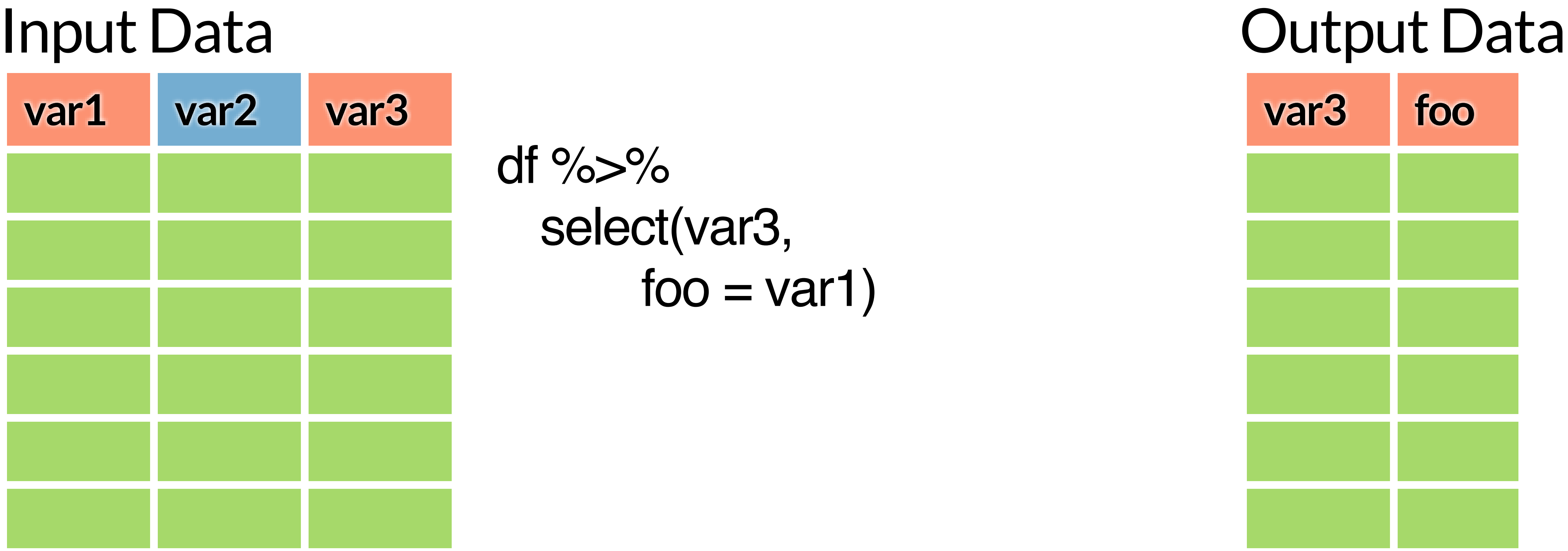
Query: Sort the flights by day (ascending) and departure-delay (descending).

```
flights %>%
  arrange(day, -dep_delay)
```

```
# A tibble: 336,776 x 19
    year month   day dep_time sched_dep_time dep_delay
   <int> <int> <int>    <int>          <int>     <dbl>
 1  2013     1     1      848           1835       853
 2  2013    12     1      657           1930       687
 3  2013     5     1        9           1655       434
 4  2013     1     1     2343           1724       379
 5  2013     8     1     2311           1659       372
 6  2013     3     1     1528            920       368
 7  2013     7     1     1602            959       363
 8  2013     3     1     1449            855       354
 9  2013     7     1     2118           1525       353
10  2013     7     1     1410            820       350
# ... with 336,766 more rows, and 13 more variables:
#   arr_time <int>, sched_arr_time <int>, arr_delay <dbl>,
#   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
```

Select Variables

select() certain columns of a dataset



```
df %>%
   select(var3,
          foo = var1)
```

Select Variables cont'd

Query: Select the carrier and the tail-number of the flights.

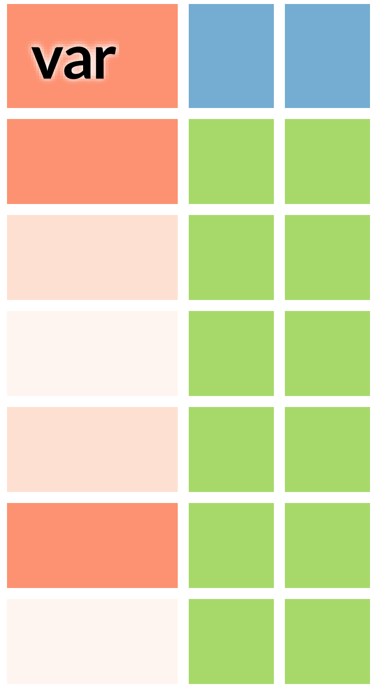
```
flights %>%
  select(carrier, tail_number = tailnum)
```

```
# A tibble: 336,776 x 2
   carrier tail_number
     <chr>      <chr>
 1      UA     N14228
 2      UA     N24211
 3      AA     N619AA
 4      B6     N804JB
 5      DL     N668DN
 6      UA     N39463
 7      B6     N516JB
 8      EV     N829AS
 9      B6     N593JB
10      AA     N3ALAA
# ... with 336,766 more rows
```

Mutate Variables

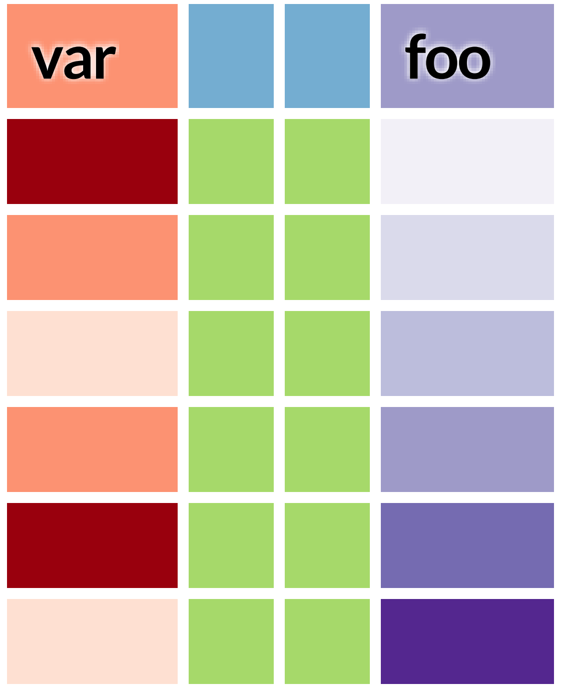
mutate() (change or create new) variables of a dataset

Input Data



```
df %>%
  mutate(var = var * 2,
         foo = 1:n())
```

Output Data



Mutate Variables cont’d

Query: Create a unique ID for each flight and compute the log distance.

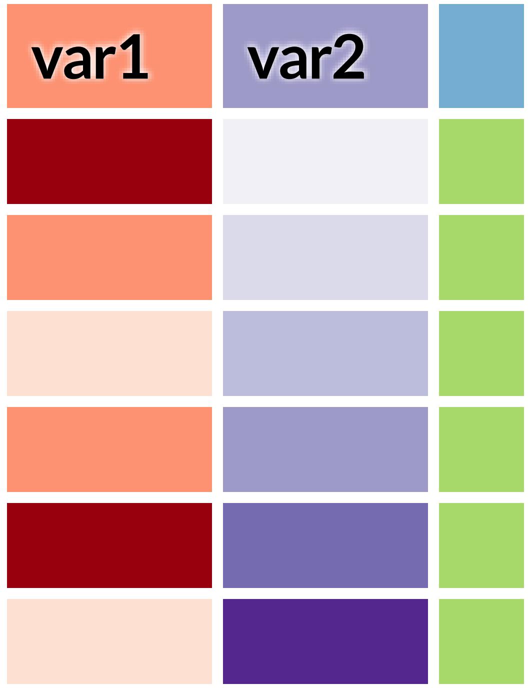
```
flights %>%
  mutate(id = 1:n(), log_dist = log(distance)) %>%
  select(id, log_dist)
```

```
# A tibble: 336,776 x 2
      id log_dist
   <int>    <dbl>
 1     1 7.244228
 2     2 7.255591
 3     3 6.993015
 4     4 7.362645
 5     5 6.635947
 6     6 6.577861
 7     7 6.970730
 8     8 5.433722
 9     9 6.850126
10    10 6.597146
# ... with 336,766 more rows
```

Summarise Variables

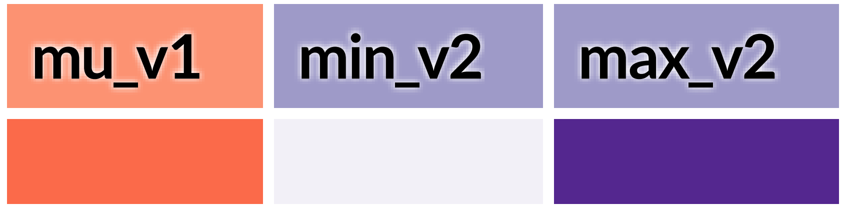
summarise() (compute a summary of) variables of a dataset

Input Data



```
df %>%
  summarise(
    mu_v1 = mean(var1),
    min_v2 = min(var2),
    max_v2 = max(var2),
  )
```

Output Data



Summarise Variables cont'd

Query: Find the minimum, average, and maximum arrival delay for all flights.

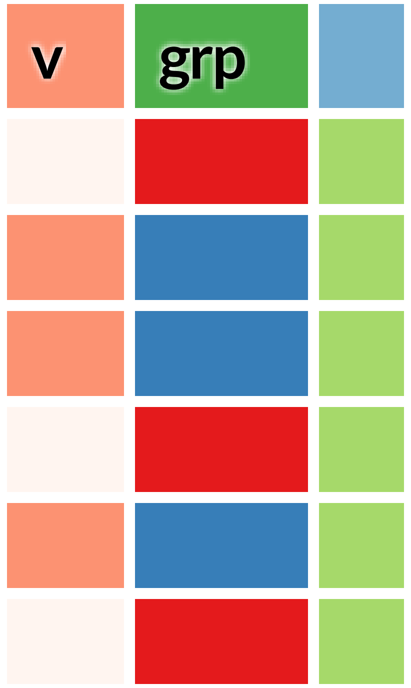
```
flights %>%
  filter(!is.na(arr_delay)) %>%
  summarise(min_delay = min(arr_delay),
            avg_delay = mean(arr_delay),
            max_delay = max(arr_delay))
```

```
# A tibble: 1 x 3
  min_delay avg_delay max_delay
      <dbl>     <dbl>     <dbl>
1       -86  6.895377      1272
```

Group Mutate

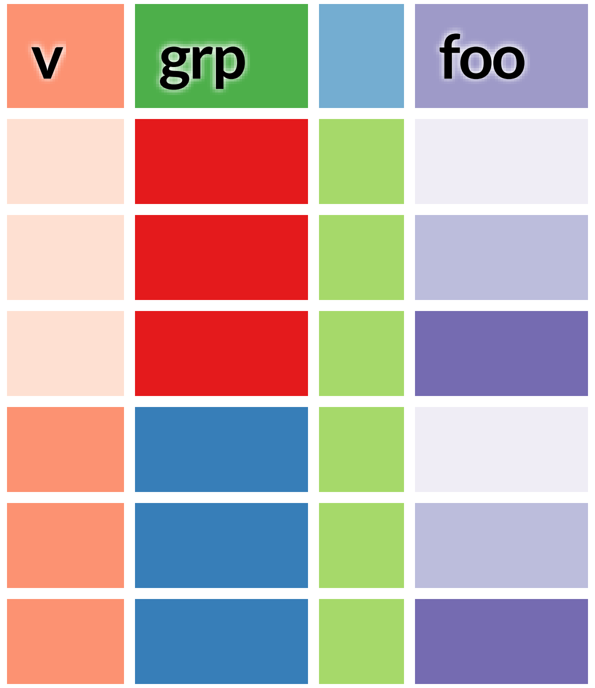
group_by() mutate (change or create new) variables of a dataset per group

Input Data



```
df %>%
  group_by(grp) %>%
  mutate(v = mean(v),
         foo = 1:n())
```

Output Data



The data output is still grouped, to ungroup use

df %>% ... %>% ungroup()

Group Mutate cont'd

Query: For each flight, find the difference of the arrival-delay to the average arrival-delay of the respective carrier (airline).

```
flights %>%
  filter(!is.na(arr_delay)) %>%
  group_by(carrier) %>%
  mutate(delta_arr_delay = arr_delay - mean(arr_delay)) %>%
  select(delta_arr_delay)
```

```
# A tibble: 327,346 x 2
# Groups:   carrier [16]
   carrier delta_arr_delay
     <chr>           <dbl>
 1      UA        7.441989
 2      UA       16.441989
 3      AA       32.635709
 4      B6      -27.457973
 5      DL      -26.644341
 6      UA        8.441989
 7      B6        9.542027
 8      EV      -29.796431
 9      B6      -17.457973
10      AA        7.635709
```

Group Summarise

group_by() summarise (compute a summary of) variables of a dataset per group

Input Data

var1	var2	grp
🟥	⬜	🟥
🟧	⬜	🟦
⬜	🟪	🟥
🟧	🟪	🟦
🟥	🟪	🟦
⬜	🟪	🟥

```
df %>%
  group_by(grp) %>%
  summarise(
    mu_v1 = mean(var1),
    min_v2 = min(var2),
    max_v2 = max(var2),
  )
```

Output Data

grp	mu_v1	min_v2	max_v2
🟥	🟧	⬜	🟪
🟦	🟧	⬜	🟪

Group Summarise cont’d

Query: For each carrier (airline), find the mean and the median arrival delay over all flights in 2013.

```
flights %>%
  filter(!is.na(arr_delay)) %>%
  group_by(carrier) %>%
  summarise(mean_delay = mean(arr_delay),
            median_delay = median(arr_delay))
```

```
# A tibble: 16 x 3
   carrier mean_delay median_delay
     <chr>      <dbl>        <dbl>
 1      9E  7.3796692           -7
 2      AA  0.3642909           -9
 3      AS -9.9308886          -17
 4      B6  9.4579733           -3
 5      DL  1.6443409           -8
 6      EV 15.7964311           -1
 7      F9 21.9207048            6
 8      FL 20.1159055            5
 9      HA -6.9152047          -13
10      MQ 10.7747334           -1
#     with 6 more rows
```

Full Pipeline Example

Query: Find the 5 aircrafts (by tail number) that have the most time made-up (on average) and have at least 20 flights.

Full Pipeline Example

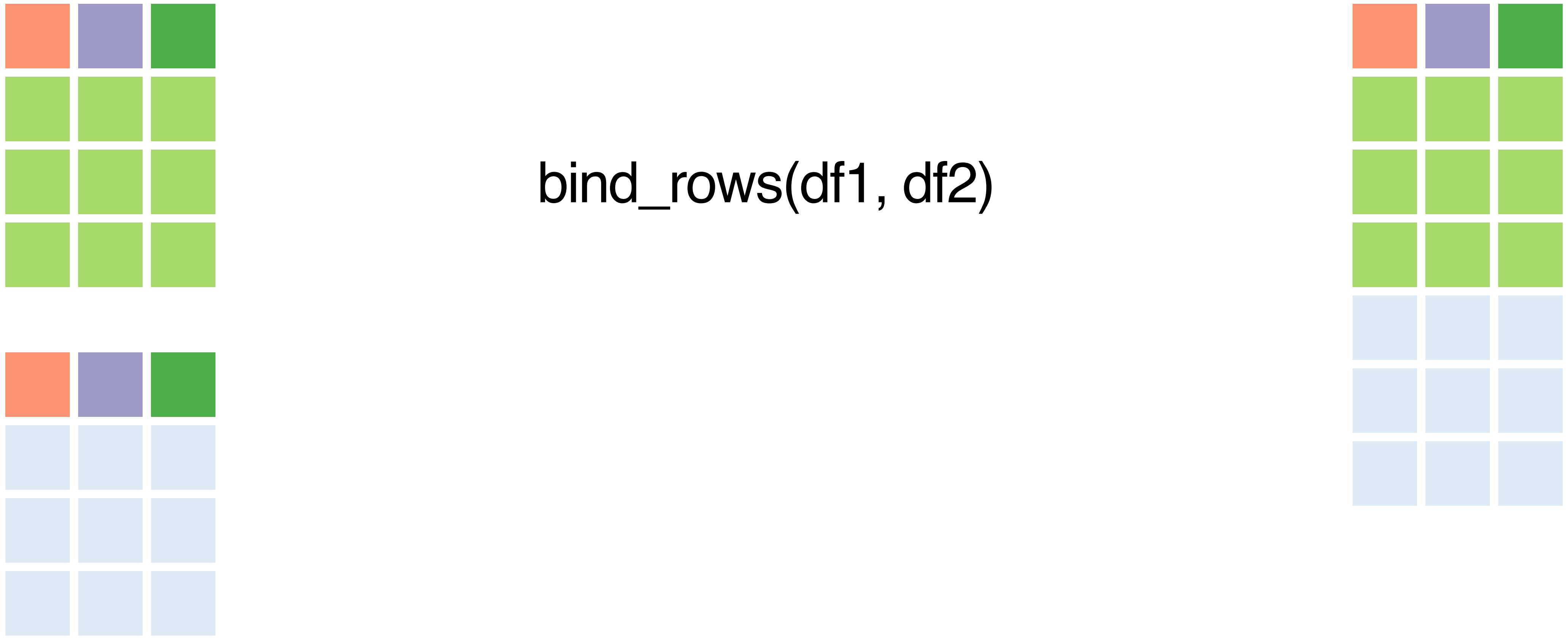
Query: Find the 5 aircrafts (by tail number) that have the most time made-up (on average) and have at least 20 flights.

```
flights %>%
  filter(!is.na(arr_delay) & !is.na(dep_delay)) %>%
  mutate(time_made_up = dep_delay - arr_delay) %>%
  group_by(tailnum) %>%
  summarise(n_flights = n(),
            time_made_up = mean(time_made_up)) %>%
  filter(n_flights > 20) %>%
  arrange(-time_made_up) %>%
  top_n(5)
```

```
# A tibble: 5 x 3
  tailnum n_flights time_made_up
    <chr>     <int>        <dbl>
1  N423AS        29     26.89655
2  N382HA        26     23.80769
3  N419AS        32     20.56250
4  N540AA        34     17.26471
5  N847VA        91     17.09890
```

Combining Rows

bind_rows() bind together two or more datasets by row



Input Datasets

bind_rows(df1, df2)

Output Data

Combining Rows cont’d

Task: Add two datasets (with the same variable-names) together by row.

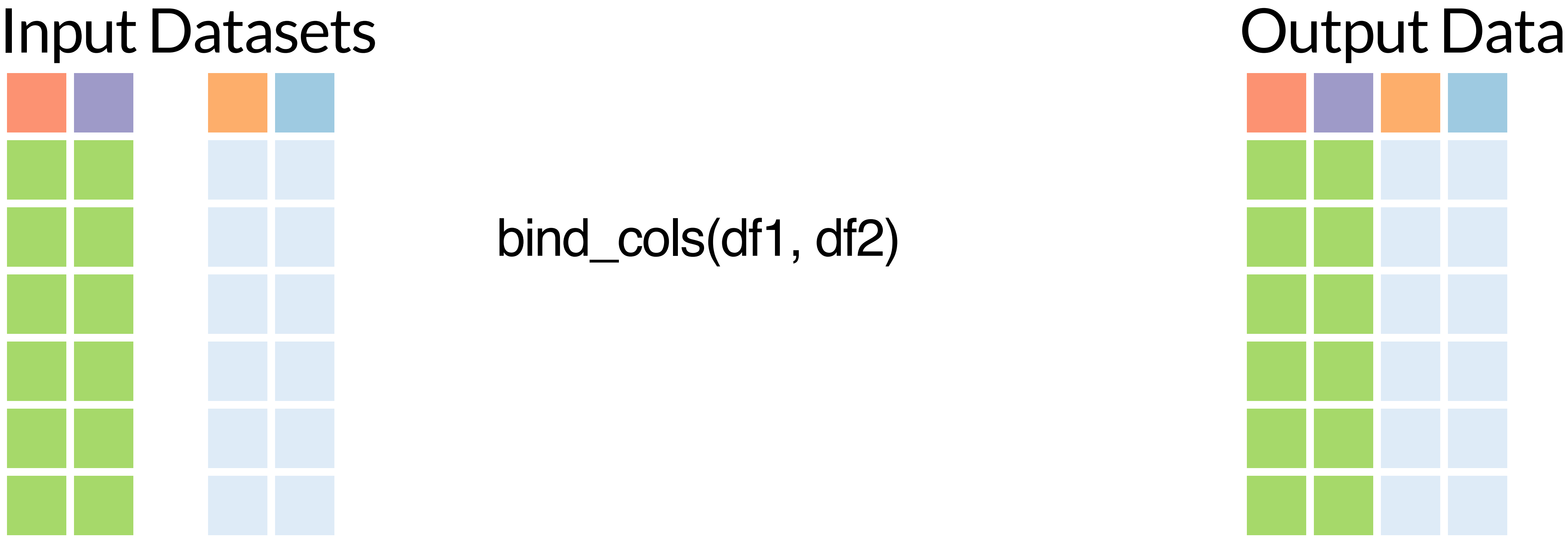
```
df1 <- data_frame(id = 1:2,
                  name = c("Alice", "Bob"))
df2 <- data_frame(id = 3:4,
                  name = c("Charlie", "Dave"))

bind_rows(df1, df2)
```

```
# A tibble: 4 x 2
     id    name
  <int>   <chr>
1     1   Alice
2     2     Bob
3     3 Charlie
4     4    Dave
```

Combining Columns

bind_cols() bind together two or more datasets by column



Input Datasets

bind_cols(df1, df2)

Output Data

Combining Columns cont'd

Task: Add two datasets (with the same row-numbers) together by column.

```r
df1 <- data_frame(id = 1:2,
                  name = c("Alice", "Bob"))
df2 <- data_frame(sales = c(100, 95),
                  region = c("North", "South"))

bind_cols(df1, df2)
```

```
# A tibble: 2 x 4
     id  name sales region
  <int> <chr> <dbl>  <chr>
1     1 Alice   100  North
2     2   Bob    95  South
```

Joins

Combine the datasets by the variable publisher.

“Left” data-frame: superheroes

superhero	alignment	publisher
Batman	good	DC
Joker	bad	DC
Xavier	good	Marvel
Hellboy	good	Dark Horse

“Right” data-frame: address

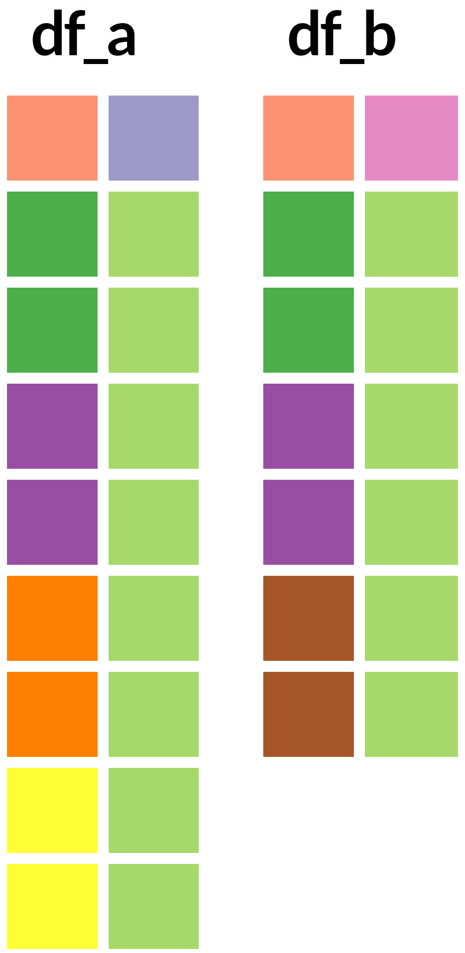
publisher	address
DC	Burbank (CA)
Marvel	NY City (NY)
Image Comics	Berkeley (CA)

Joined data-frame

superhero	alignment	publisher	address
Batman	good	DC	Burbank (CA)
Joker	bad	DC	Burbank (CA)
Xavier	good	Marvel	NY City (NY)
???	???	???	???

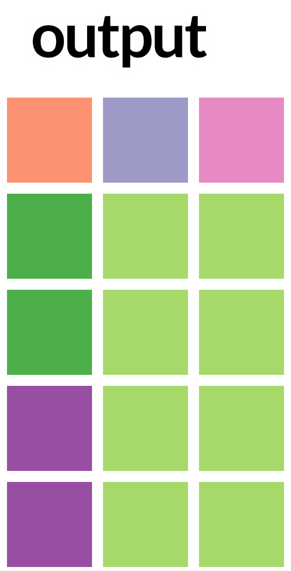
Inner Join

inner_join() to take only observations found in both datasets



```
inner_join(df_a, df_b,
        by = "id")
```

Output Data



Inner Join cont’d

Task: Find for all superheroes the matching addresses if there is information.

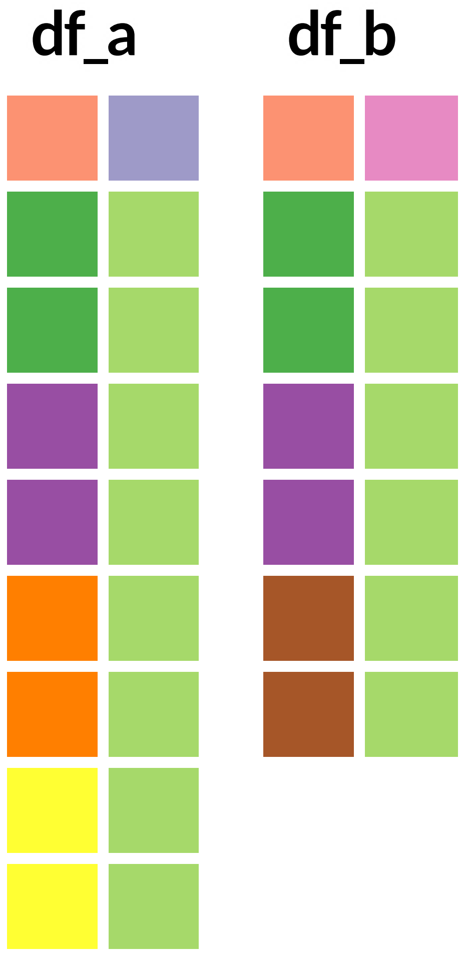
```
# recreate the data
superheroes <- data_frame(
  superhero = c("Batman", "Joker", "Xavier", "Hellboy"),
  alignment = c("good", "bad", "good", "good"),
  publisher = c("DC", "DC", "Marvel", "Dark Horse")
)
address <- data_frame(
  publisher = c("DC", "Marvel", "Image Comics"),
  address   = c("Burbank (CA)", "NY City (NY)",
                "Portland (OR)")
)

# perform the join
inner_join(superheroes, address, by = "publisher")
```

```
# A tibble: 3 x 4
  superhero alignment publisher     address
      <chr>     <chr>     <chr>       <chr>
1    Batman      good        DC Burbank (CA)
2     Joker       bad        DC Burbank (CA)
```

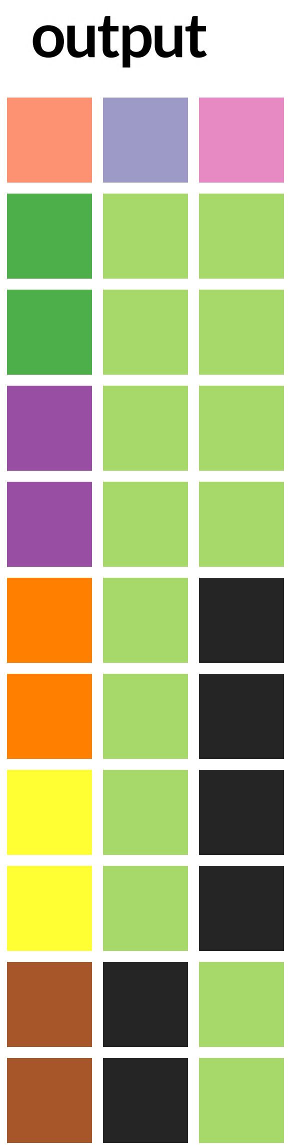
Full Join

full_join() to take all observations



```
full_join(df_a, df_b,
          by = "id")
```

Output Data



Full Join cont'd

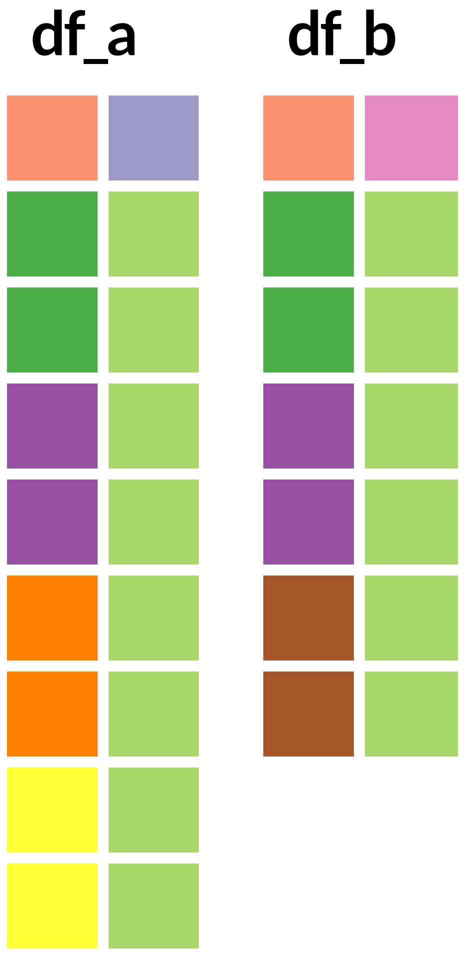
Task: List all known information about publishing houses and their address.

```
full_join(superheroes, address, by = "publisher")
```

```
# A tibble: 5 x 4
  superhero alignment   publisher      address
      <chr>     <chr>       <chr>        <chr>
1    Batman      good          DC  Burbank (CA)
2     Joker       bad          DC  Burbank (CA)
3    Xavier      good      Marvel  NY City (NY)
4   Hellboy      good  Dark Horse         <NA>
5      <NA>      <NA> Image Comics Portland (OR)
```

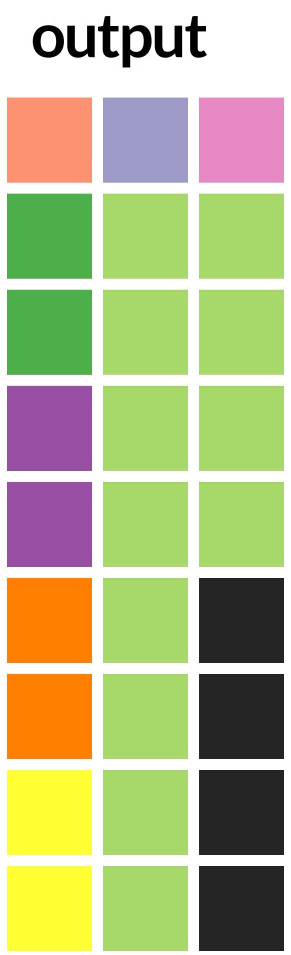
Left Join

left_join() to take all observations of the left dataset



```
left_join(df_a, df_b,
          by = "id")
```

Output Data



Left Join cont’d

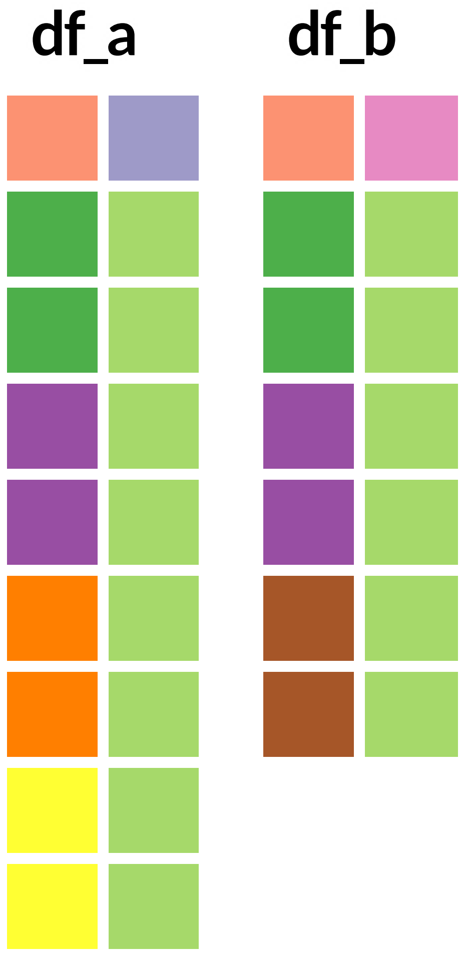
Task: For all superheroes, add their address.

```
left_join(superheroes, address, by = "publisher")
```

```
# A tibble: 4 x 4
  superhero alignment  publisher      address
      <chr>     <chr>      <chr>        <chr>
1    Batman      good         DC Burbank (CA)
2     Joker       bad         DC Burbank (CA)
3    Xavier      good     Marvel NY City (NY)
4   Hellboy      good Dark Horse         <NA>
```

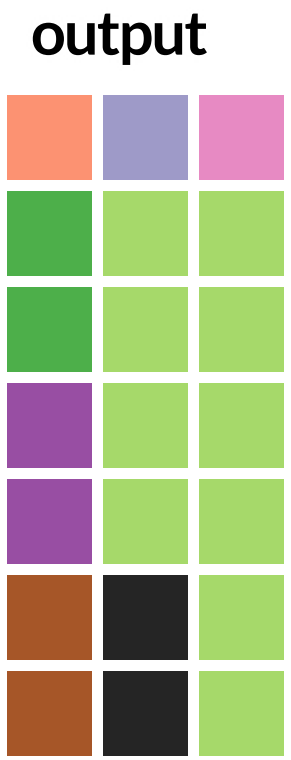
Right Join

right_join() to take all observations of the right dataset



```
right_join(df_a, df_b,
        by = "id")
```

Output Data



Right Join cont’d

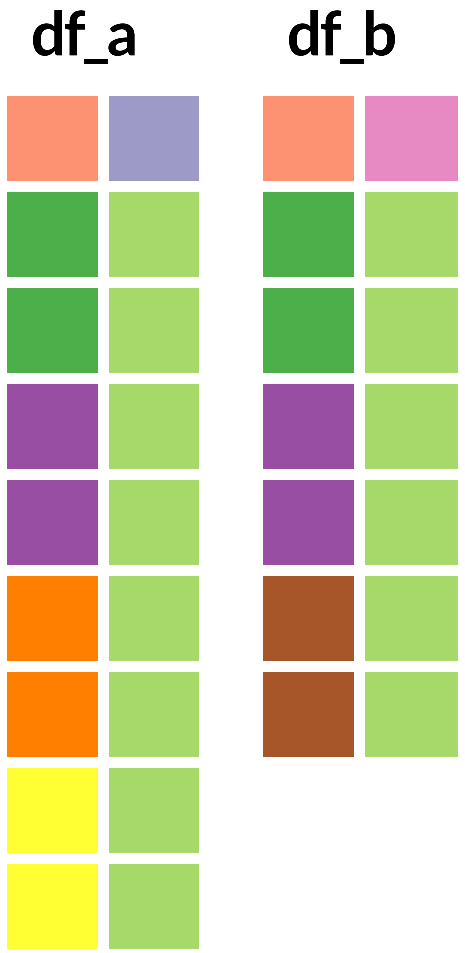
Task: For all publishers, add the superheroes.

```
right_join(superheroes, address, by = "publisher")
```

```
# A tibble: 4 x 4
  superhero alignment   publisher      address
      <chr>     <chr>       <chr>        <chr>
1    Batman      good          DC  Burbank (CA)
2     Joker       bad          DC  Burbank (CA)
3    Xavier      good      Marvel  NY City (NY)
4      <NA>      <NA> Image Comics Portland (OR)
```

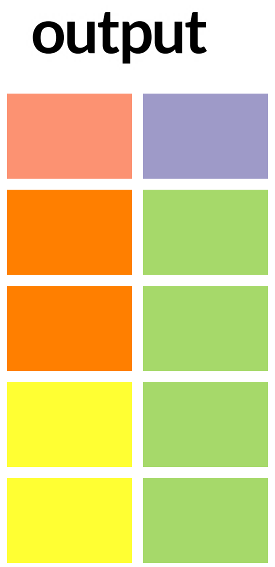
Anti Join

anti_join() to take all observations in df_a that are not in df_b



anti_join(df_a, df_b,
by = "id")

Output Data



Anti Join cont'd

Task: Find all superheroes that have no address.

```
anti_join(superheroes, address, by = "publisher")
```

```
# A tibble: 1 x 3
  superhero alignment  publisher
      <chr>     <chr>      <chr>
1   Hellboy      good Dark Horse
```

Task: Find all publishers that have no superhero.

```
anti_join(address, superheroes, by = "publisher")
```

```
# A tibble: 1 x 2
     publisher       address
         <chr>         <chr>
1 Image Comics Portland (OR)
```

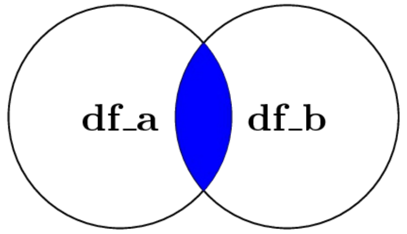
Merging Two Datasets

`dplyr-syntax`

for two data_frames: `df_a, df_b`

Inner Join

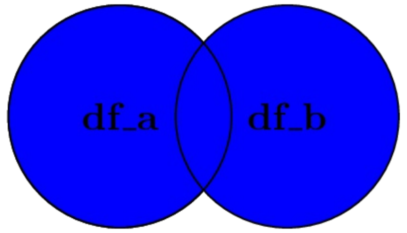
`inner_join()`



`inner_join(df_a, df_b, by = "id")`

Full Join

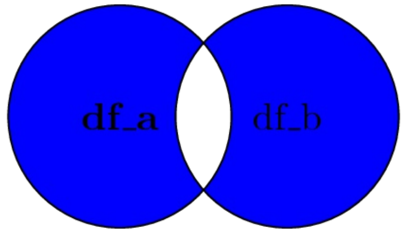
`full_join()`



`full_join(df_a, df_b, by = "id")`

Outer Join

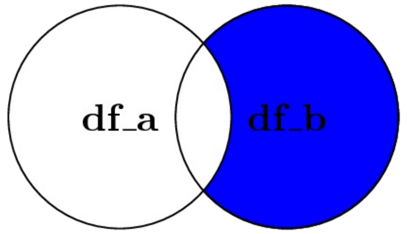
`NA`



`NA`

Right Outer Join

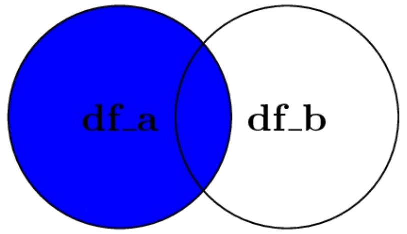
`anti_join()` (reverse)



`anti_join(df_b, df_a, by = "id")`

Left Join

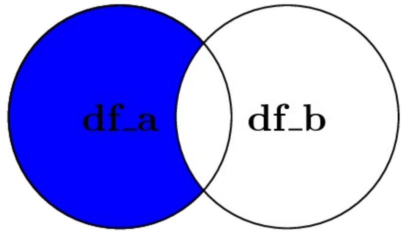
`left_join()`



`left_join(df_a, df_b, by = "id")`

Left Outer Join

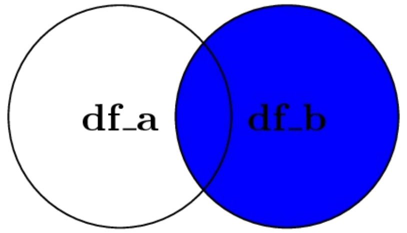
`anti_join()`



`anti_join(df_a, df_b, by = "id")`

Right Join

`right_join()`



`right_join(df_a, df_b, by = "id")`

Additional Resources

- *R for Data Science* by Hadley Wickham Online http://r4ds.had.co.nz/ or paperback Amazon
- *Data Wrangling CheatSheet* https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf

Questions?

About and Contact

My Ideas: https://datashenanigan.wordpress.com/

My Projects: https://github.com/DavZim/

My Contact: david_j_zimmermann@hotmail.com

Sources

Images:

- https://www.tidyverse.org/
- Rene Magritte: The Treachery of Images 1929
- Ursus Wehrli: The Art of Clean Up 2013
- http://r4ds.had.co.nz/tidy-data.html
- http://fontawesome.io/icon/refresh/
- http://www.wallpaperspots.com/fruit-mix-hd-wallpaper/