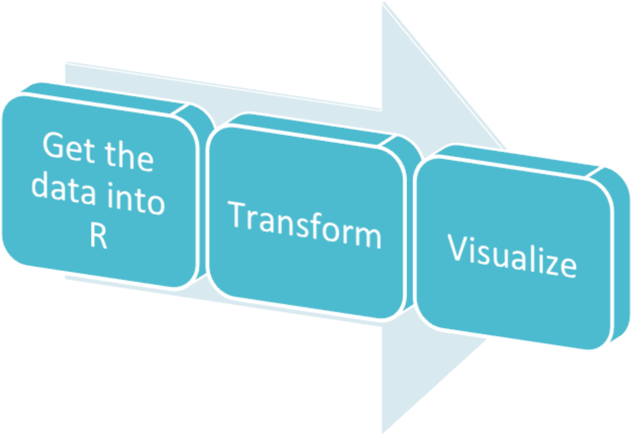
R language and data analysis:data manipulation advanced

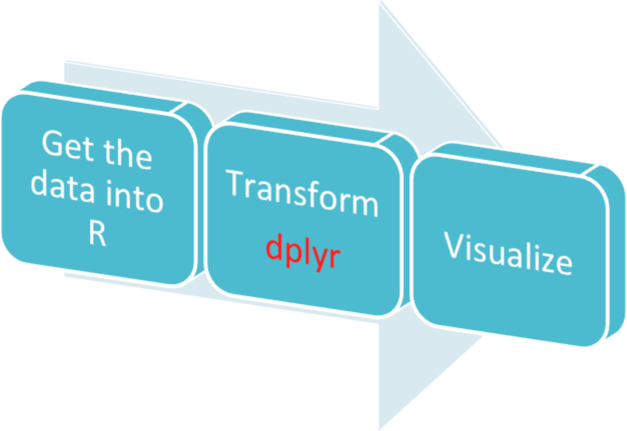
Qiang Shen

Dec. 17, 2017

data analysis procedure



data analysis procedure



Get the data into R

Transform

dplyr

Visualize

A quick demo 1

- split-apply-combine in base

```
with(iris, tapply(iris[, 1], Species,
    mean))
```

```
    setosa versicolor  virginica
     5.006      5.936      6.588
```

A quick demo 2

- split-apply-combine in base

```r
s <- split(iris, iris$Species)
tmp.data <- iris[, 1:3]
# head(tmp.data)
sapply(s, function(x) colMeans(tmp.data,
    na.rm = T))
```

```
               setosa versicolor
Sepal.Length 5.843333   5.843333
Sepal.Width  3.057333   3.057333
Petal.Length 3.758000   3.758000
             virginica
Sepal.Length  5.843333
Sepal.Width   3.057333
Petal.Length  3.758000
```

dplyr

```
library(dplyr)
iris %>% group_by(Species) %>% select(one_of("Sepal.Length"
    summarise_each(funs(mean(.)))
iris %>% group_by(Species) %>% select(one_of(names(iris[,
    1:3]))) %>% summarise_each(funs(mean(.)))
```

dplyr

```
# A tibble: 3 x 2
     Species Sepal.Length
      <fctr>        <dbl>
1     setosa        5.006
2 versicolor        5.936
3  virginica        6.588

# A tibble: 3 x 4
     Species Sepal.Length
      <fctr>        <dbl>
1     setosa        5.006
2 versicolor        5.936
3  virginica        6.588
# ... with 2 more variables:
#   Sepal.Width <dbl>,
#   Petal.Length <dbl>
```

dplyr: tibble

```r
library(tibble)
iris
head(iris)
as.tibble(iris)
tbl_df(iris)
```

dplyr: tibble

```
tibble(x = 1:5, y = 1, z = x^2 + y)
```

```
# A tibble: 5 x 3
      x     y     z
  <int> <dbl> <dbl>
1     1     1     2
2     2     1     5
3     3     1    10
4     4     1    17
5     5     1    26
```

dplyr: tibble

```r
tribble(~x, ~y, ~z, "a", 2, 3.6, "b",
    1, 8.5)
```

```
# A tibble: 2 x 3
      x     y     z
  <chr> <dbl> <dbl>
1     a     2   3.6
2     b     1   8.5
```

dplyr: tibble

```
df1 <- data.frame(x = 1:3, y = 3:1)
class(df1[, 1:2])
```

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

```
class(df1[, 1])
```

```
[1] "integer"
```

```
df2 <- tibble(x = 1:3, y = 3:1)
class(df2[, 1:2])
```

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

```
class(df2[, 1])
```

```
[1] "tbl_df"     "tbl"
```

dplyr features

- a powerful R-package to transform and summarize tabular data with rows and columns.

1. intutive to translate your thoughts into codes/scripts.
2. spend less time waiting for the computer.

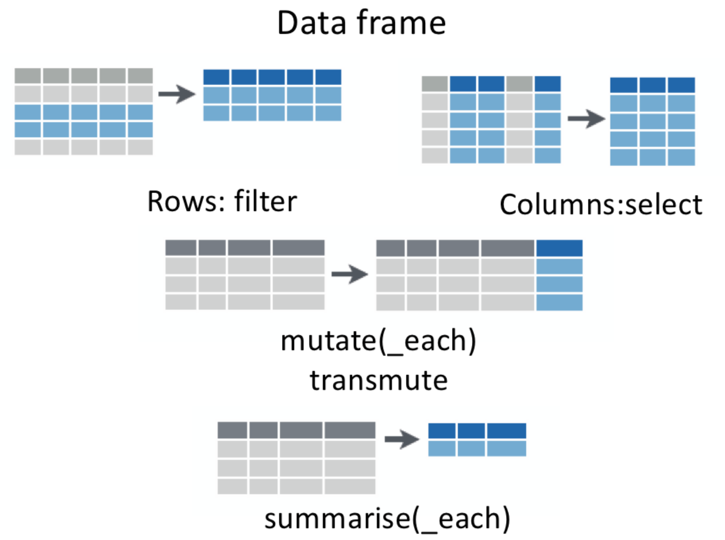
key verbs in dplyr package

- filter() (and slice())
- arrange()
- select() (and rename())
- distinct()
- mutate() (and transmute())
- summarise()
- sample_n() and sample_frac()

key verbs

- filter: keep rows matching criteria
- select: pick columns by name
- arrange: reorder rows
- mutate: add new variables
- summarise: reduce variables to values

key verbs



Data frame

Rows: filter

Columns:select

mutate(_each)

transmute

summarise(_each)

strcture of verbs

- First argument is a data frame, say *flights*.
- Subsequent arguments say what to do with the data frame.
- The result is a data frame

basics of dataset

- dataset contains all 336776 flights that departed from New York City in 2013.

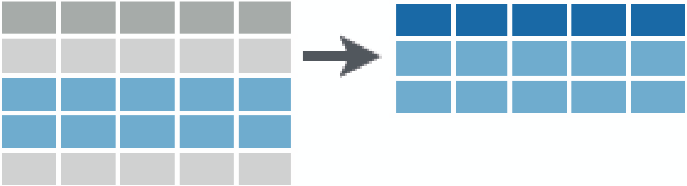
```
library(dplyr)
library(nycflights13)
dim(flights)
head(flights)
flights  #tbl_df
print(tbl_df(mtcars), n = 5)
tbl_df(iris)
glimpse(flights)
```

key verbs

- **filter() (and slice())**
- arrange()
- select() (and rename())
- distinct()
- mutate() (and transmute())
- summarise()
- sample_n() and sample_frac()

base function

- all flights on January 1st



```
flights[flights$month == 1 & flights$day ==
    1, ]
```

base function

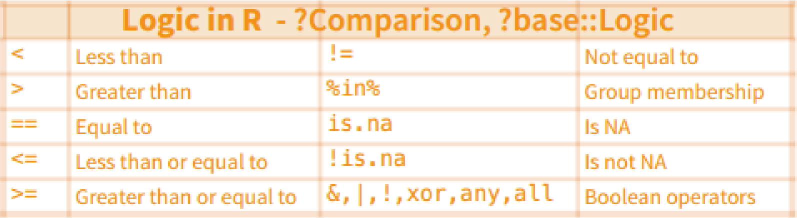
- all flights on January 1st

```
library(nycflights13)
subset(flights, month == 1 & day == 1)
```

filter()

```
library(nycflights13)
filter(flights, month == 1, day == 1)  ##tidyr ':'' vs. ',
filter(flights, month == 1 | month ==
    2)
filter(flights, carrier == "AA" | carrier ==
    "UA")
filter(flights, carrier %in% c("AA",
    "UA"))
```

logic for filter()



Logic in R - ?Comparison, ?base::Logic			
<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&,|,!,xor,any,all	Boolean operators

select rows by position: base

```
library(dplyr)
library(nycflights13)
flights[1:10, ]
```

select rows by position: `slice`

```r
library(nycflights13)
slice(flights, 1:10)
```

key verbs

- filter() (and slice())
- **arrange()**
- select() (and rename())
- distinct()
- mutate() (and transmute())
- summarise()
- sample_n() and sample_frac()

order data: base

- Order data based on specified columns.

```
flights[order(flights$year, flights$month,
    flights$day), ]
flights[order(-flights$arr_delay), ]  #descend
```

order data: arrange

- Order data based on specified columns.

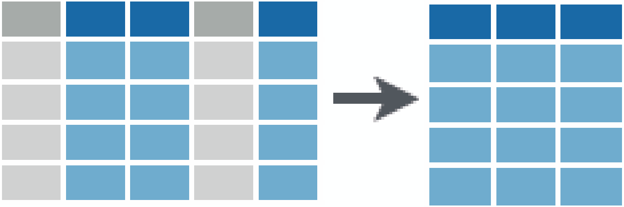
```
library(nycflights13)
arrange(flights, year, month, day)
arrange(flights, desc(arr_delay))
```

key verbs

- filter() (and slice())
- arrange()
- **select() (and rename())**
- distinct()
- mutate() (and transmute())
- summarise()
- sample_n() and sample_frac()

Select columns by name: base

- select columns of year, month and age



```
library(nycflights13)
flights[, c("year", "month", "day")]
subset(flights, select = (year:day))
```

Select columns by name: `select`

- Select columns between year and day (inclusive)

```r
library(nycflights13)
select(flights, year, month, day)
select(flights, year:day)
```

Select columns by name: `select`

- Select all columns except those from year to day (inclusive)

```r
library(nycflights13)
select(flights, -(year:day))
```

select:rename

```
library(nycflights13)
select(flights, tail_num = tailnum)
```

verb rename in dplyr

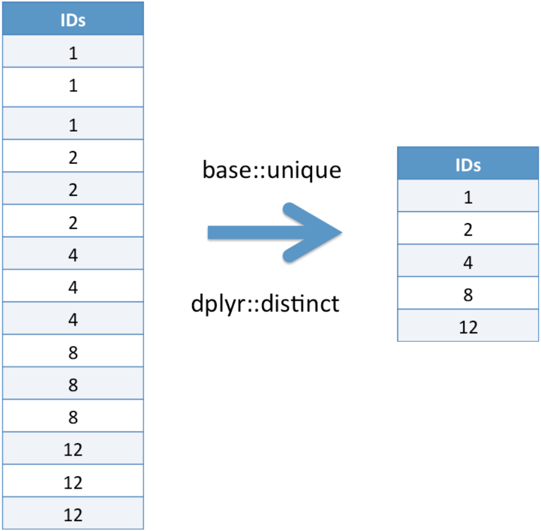
- rename a variable

```
library(nycflights13)
rename(flights, tail_num = tailnum)
flights
```

key verbs

- filter() (and slice())
- arrange()
- select() (and rename())
- **distinct()**
- mutate() (and transmute())
- summarise()
- sample_n() and sample_frac()

distinct vs. unique



IDs
1
1
1
2
2
2
4
4
4
8
8
8
12
12
12

base::unique

dplyr::distinct

IDs
1
2
4
8
12

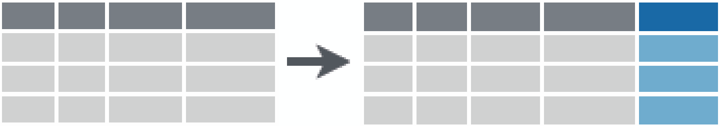
distinct vs. unique

```r
library(nycflights13)
unique(select(flights, tailnum))
distinct(select(flights, tailnum))
distinct(select(flights, origin, dest))
```

key verbs: new column

- filter() (and slice())
- arrange()
- select() (and rename())
- distinct()
- **mutate() (and transmute())**
- summarise()
- sample_n() and sample_frac()

new column: base



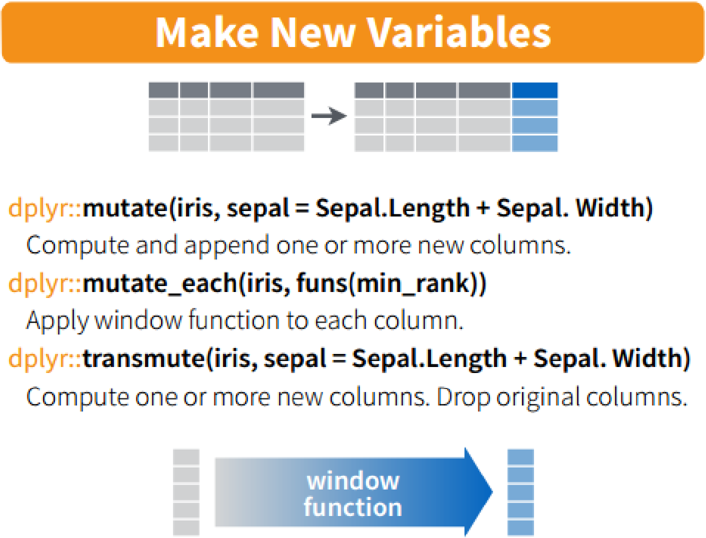
```
iris$Sepal_sum <- iris$Sepal.Length +
    iris$Sepal.Width
head(iris)[, c(1:2, 5:6)]
```

```
  Sepal.Length Sepal.Width Species
1          5.1         3.5  setosa
2          4.9         3.0  setosa
3          4.7         3.2  setosa
4          4.6         3.1  setosa
5          5.0         3.6  setosa
6          5.4         3.9  setosa
```

new column: `transform` in `base`

```
library(nycflights13)
transform(flights, gain = arr_delay -
    dep_delay, speed = distance/air_time *
    60)
```

create new coulumn



Make New Variables

dplyr::**mutate(iris, sepal = Sepal.Length + Sepal. Width)**

Compute and append one or more new columns.

dplyr::**mutate_each(iris, funs(min_rank))**

Apply window function to each column.

dplyr::**transmute(iris, sepal = Sepal.Length + Sepal. Width)**

Compute one or more new columns. Drop original columns.

new column: mutate

```r
library(nycflights13)
mutate(flights, gain = arr_delay - dep_delay,
    speed = distance/air_time * 60)
```

transform vs. mutate

```r
mutate(flights, gain = arr_delay - dep_delay,
    gain_per_hour = gain/(air_time/60))

transform(flights, gain = arr_delay -
    delay, gain_per_hour = gain/(air_time/60))
```

transmute

```
transmute(flights, gain = arr_delay -
    dep_delay, gain_per_hour = gain/(air_time/60))
```

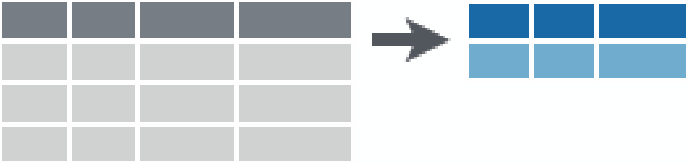
mutate_each

```r
iris_new <- mutate_each(iris[, 1:4],
    funs(ratio = ./max(.), avg = mean(.)))
head(iris_new)
```

key verbs in dplyr package

- filter() (and slice())
- arrange()
- select() (and rename())
- distinct()
- mutate() (and transmute())
- **summarise()**
- sample_n() and sample_frac()

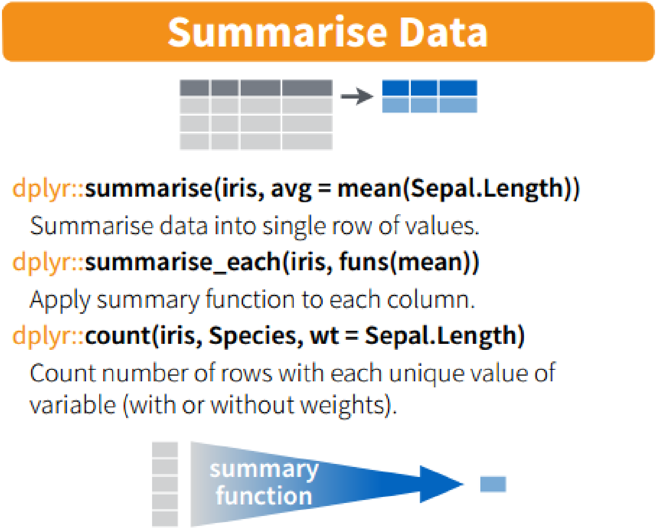
summarise



```
library(nycflights13)
summarise(flights, delay = mean(dep_delay,
    na.rm = TRUE))
```

```
# A tibble: 1 x 1
     delay
     <dbl>
1 12.63907
```

summarise



Summarise Data

dplyr::**summarise(iris, avg = mean(Sepal.Length))**

Summarise data into single row of values.

dplyr::**summarise_each(iris, funs(mean))**

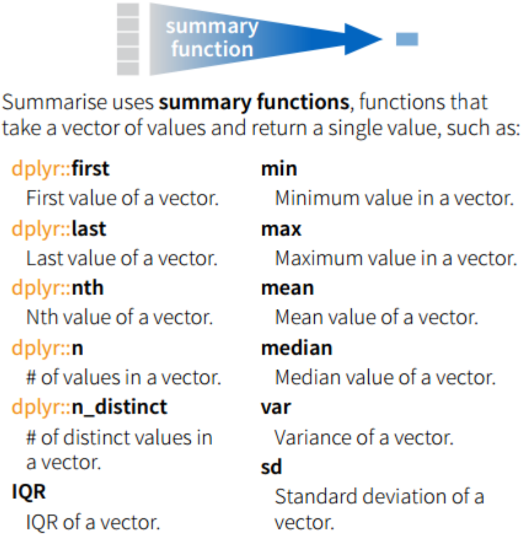
Apply summary function to each column.

dplyr::**count(iris, Species, wt = Sepal.Length)**

Count number of rows with each unique value of variable (with or without weights).

summary
function

summarise



Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

dplyr::**first**
First value of a vector.

dplyr::**last**
Last value of a vector.

dplyr::**nth**
Nth value of a vector.

dplyr::**n**
# of values in a vector.

dplyr::**n_distinct**
# of distinct values in a vector.

IQR
IQR of a vector.

min
Minimum value in a vector.

max
Maximum value in a vector.

mean
Mean value of a vector.

median
Median value of a vector.

var
Variance of a vector.

sd
Standard deviation of a vector.

summarise_each

```r
library(nycflights13)
summarise_each(iris[, 1:4], funs(avg = mean(.,
    na.rm = T)))
```

```
  Sepal.Length_avg Sepal.Width_avg
1         5.843333        3.057333
  Petal.Length_avg Petal.Width_avg
1            3.758        1.199333
```

count

```
# count(iris, Species)
count(iris, Species, wt = Sepal.Length)
```

```
# A tibble: 3 x 2
     Species     n
      <fctr> <dbl>
1     setosa 250.3
2 versicolor 296.8
3  virginica 329.4
```

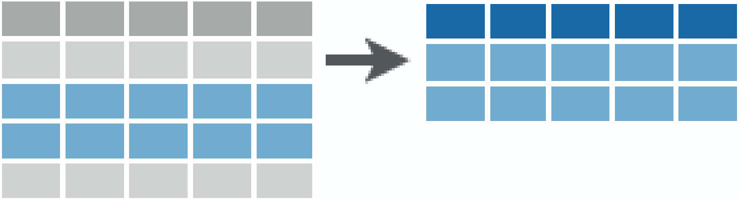
```
with(iris, tapply(iris[, 1], Species,
    sum))
```

```
    setosa versicolor  virginica
     250.3      296.8      329.4
```

key verbs in dplyr package

- filter() (and slice())
- arrange()
- select() (and rename())
- distinct()
- mutate() (and transmute())
- summarise()
- **sample_n() and sample_frac()**

Randomly sample rows.



```
sample_n(flights, 10)
sample_frac(flights, 0.01)
sample_frac(flights, 0.01, replace = T)  ##bootstrap
```

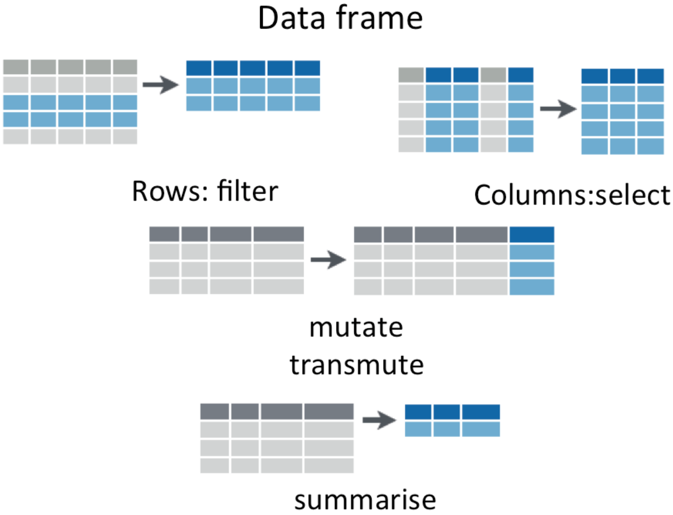
key verbs in dplyr package

- filter() (and slice())
- arrange()
- select() (and rename())
- distinct()
- mutate() (and transmute())
- summarise()
- sample_n() and sample_frac()

key verbs

- filter: keep rows matching criteria
- select: pick columns by name
- mutate: add new variables
- summarise: reduce variables to values

key verbs



Data frame

Rows: filter

Columns:select

mutate
transmute

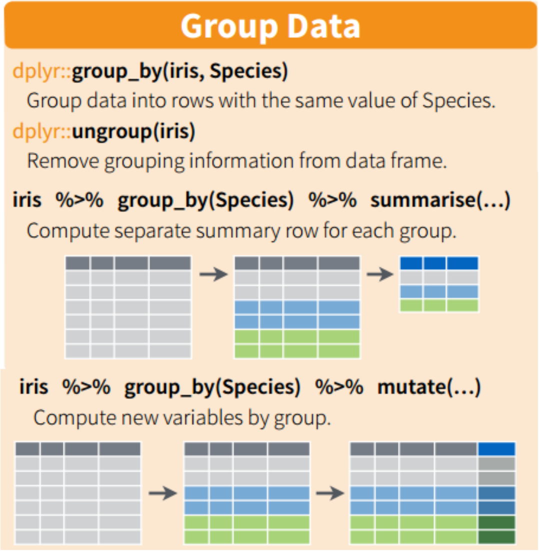
summarise

strcture of verbs

- The first argument is a data frame. (import)
- The subsequent arguments describe what to do with it.
- The result is a new data frame. (export)

Practice!

group_by



Group Data

```
dplyr::group_by(iris, Species)
```

Group data into rows with the same value of Species.

```
dplyr::ungroup(iris)
```

Remove grouping information from data frame.

```
iris %>% group_by(Species) %>% summarise(...)
```

Compute separate summary row for each group.

```
iris %>% group_by(Species) %>% mutate(...)
```

Compute new variables by group.

group_by

```r
library(nycflights13)
by_tailnum <- group_by(flights, tailnum)
by_tailnum
delay <- summarise(by_tailnum, count = n(),
    dist = mean(distance, na.rm = TRUE),
    delay = mean(arr_delay, na.rm = TRUE))
delay <- filter(delay, count > 20, dist <
    2000)
with(delay, plot(dist, delay))
abline(with(delay, lm(delay ~ dist)))
```

aggregate functions

- package base: min(), max(), mean(), sum(), sd(), median(), and IQR().
- package dplyr: n(), n_distinct(x), first(x), last(x) and nth(x, n)

```
destinations <- group_by(flights, dest)
destinations
dest <- summarise(destinations, planes = n_distinct(tailnum
    flights = n())
arrange(dest, desc(planes))
summarise(dest, sum(flights))
```

aggregate functions

```
daily <- group_by(flights, month, day)
(per_day <- summarise(daily, flights = n()))
(per_month <- summarise(per_day, flights = sum(flights)))
(per_year <- summarise(per_month, flights = sum(flights)))
```

group_by: mutate with summarise

```
flights <- mutate(flights, hrs = air_time/60,
    speed = distance/hrs)
flights0 <- group_by(flights, tailnum)
flight <- select(flights0, speed)
flight
summarise(flight, speed_mean = mean(speed,
    na.rm = T))
```

Chaining: example

```
a1 <- group_by(flights, year, month,
    day)
a2 <- select(a1, arr_delay, dep_delay)
a3 <- summarise(a2, arr = mean(arr_delay,
    na.rm = TRUE), dep = mean(dep_delay,
    na.rm = TRUE))
a4 <- filter(a3, arr > 30 | dep > 30)
```

Chaining: example

```r
filter(summarise(select(group_by(flights,
    year, month, day), arr_delay, dep_delay),
    arr = mean(arr_delay, na.rm = TRUE),
    dep = mean(dep_delay, na.rm = TRUE)),
    arr > 30 | dep > 30)
```

Chaining: example

- %>%

```
flights %>% group_by(year, month, day) %>%
    select(arr_delay, dep_delay) %>%
    summarise(arr = mean(arr_delay, na.rm = TRUE),
        dep = mean(dep_delay, na.rm = TRUE)) %>%
    filter(arr > 30 | dep > 30)
```

summary

- filter: keep rows matching criteria
- select: pick columns by name
- mutate: add new variables
- summarise: reduce variables to values
- chaining: %>%