1.transform

Joao Lopes

2024-10-07

Contents

1. NUMERIC VECTORS

[from https://r4ds.hadley.nz/numbers]

```r
library("nycflights13") #collection of datasets
library("skimr")        #function skim() for descriptive statistics
library("tidyverse")    #collection of packages for data analysis
```

```r
#metadata
?flights
```

```r
#data inspection
glimpse(flights)

#descriptive statistics
skim(flights)
```

1.1. COUNTS

```r
#simple count
flights |> count(dest)

#sorted count
flights |> count(dest, sort = TRUE)

#count + summary statisctics
flights |>
  group_by(dest) |>
  summarize(
    n = n(),
    delay = mean(arr_delay, na.rm = TRUE)
  )

#count distinct
flights |>
  group_by(dest) |>
  summarize(carriers = n_distinct(carrier)) |>
  arrange(desc(carriers))

#sum is weighted count
flights |>
  group_by(tailnum) |>
  summarize(miles = sum(distance))

flights |> count(tailnum, wt = distance)

#count missing values
flights |>
  group_by(dest) |>
  summarize(n_cancelled = sum(is.na(dep_time)))
```

1.2. NUMERIC TRANSFORMATION

Arithmetic and recycling rules

```r
x <- c(1, 2, 10, 20)
x / 5
x / c(5, 5, 5, 5)
x * c(1, 2)
x * c(1, 2, 3)
```

Minimum and maximum

```r
df <- tribble(
  ~x, ~y,
  1,  3,
  5,  2,
  7, NA,
)

df |>
  mutate(
    min = pmin(x, y, na.rm = TRUE),
    max = pmax(x, y, na.rm = TRUE)
  )

df |>
  mutate(
    min = min(x, y, na.rm = TRUE),
    max = max(x, y, na.rm = TRUE)
  )
```

Modular arithmetic

```r
1:10 %/% 3
1:10 %% 3

flights |>
  mutate(
    hour = sched_dep_time %/% 100,
    minute = sched_dep_time %% 100,
    .keep = "used"
  )

flights |>
  group_by(hour = sched_dep_time %/% 100) |>
  summarize(prop_cancelled = mean(is.na(dep_time)), n = n()) |>
  filter(hour > 1) |>
  ggplot(aes(x = hour, y = prop_cancelled)) +
  geom_line(color = "grey50") +
  geom_point(aes(size = n))
```

Rounding

```
#simple round
round(123.456)

#round to nearest n digit
round(123.456, 2)  # two digits
round(123.456, 1)  # one digit
round(123.456, -1) # round to nearest ten
round(123.456, -2) # round to nearest hundred

#floor() vs ceiling()
round(c(1.5, 2.5))

x <- 123.456
floor(x)
ceiling(x)

#round down/up to nearest n digit
floor(x / 0.01) * 0.01
ceiling(x / 0.01) * 0.01

#round to nearest multiple
round(x / 4) * 4       # round to nearest multiple of 4
round(x / 0.25) * 0.25 # round to nearest 0.25
```

Cutting numbers into ranges

```
#simple cut
x <- c(1, 2, 5, 10, 15, 20)
cut(x, breaks = c(0, 5, 10, 15, 20))
cut(x, breaks = c(0, 5, 10, 100))
cut(x,
  breaks = c(0, 5, 10, 15, 20),
  labels = c("sm", "md", "lg", "xl")
)

#cut with values outside the range
y <- c(NA, -10, 5, 10, 30)
cut(y, breaks = c(0, 5, 10, 15, 20))
```

Cumulative and rolling aggregates

```
x <- 1:10
cumsum(x)
```

1.3. GENERAL TRANSFORMATION

Ranks

```
#simple ranks
x <- c(1, 2, 2, 3, 4, NA)
min_rank(x)
```

```r
min_rank(desc(x))

#more ranks
df <- tibble(x = x)
df |>
  mutate(
    row_number = row_number(x),
    dense_rank = dense_rank(x),
    percent_rank = percent_rank(x),
    cume_dist = cume_dist(x)
  )

#using ranks to divide data
df <- tibble(id = 1:10)
df |>
  mutate(
    row0 = row_number() - 1,
    three_groups = row0 %% 3,
    three_in_each_group = row0 %/% 3
  )
```

Offsets

```r
x <- c(2, 5, 11, 11, 19, 35)
lag(x)
lead(x)

x - lag(x)
x == lag(x)
```

Consecutive identifiers

```r
events <- tibble(
  time = c(0, 1, 2, 3, 5, 10, 12, 15, 17, 19, 20, 27, 28, 30)
)

events <- events |>
  mutate(
    diff = time - lag(time, default = first(time)),
    has_gap = diff >= 5
  )
events

events |> mutate(
  group = cumsum(has_gap)
)
```

1.4. SUMMARY STATISTICS

Center

```r
flights |>
  group_by(year, month, day) |>
  summarize(
    mean = mean(dep_delay, na.rm = TRUE),
    median = median(dep_delay, na.rm = TRUE),
    n = n(),
    .groups = "drop"
  ) |>
  ggplot(aes(x = mean, y = median)) +
  geom_abline(slope = 1, intercept = 0, color = "white", linewidth = 2) +
  geom_point()
```

Minimum, maximum, and quantiles

```r
flights |>
  group_by(year, month, day) |>
  summarize(
    max = max(dep_delay, na.rm = TRUE),
    q95 = quantile(dep_delay, 0.95, na.rm = TRUE),
    .groups = "drop"
  )
```

Spread

```r
flights |>
  group_by(origin, dest) |>
  summarize(
    distance_sd = IQR(distance),
    n = n(),
    .groups = "drop"
  ) |>
  filter(distance_sd > 0)
```

Distributions

```r
flights |>
  filter(dep_delay < 120) |>
  ggplot(aes(x = dep_delay, group = interaction(day, month))) +
  geom_freqpoly(binwidth = 5, alpha = 1/5)
```

Positions

```r
flights |>
  group_by(year, month, day) |>
  summarize(
    first_dep = first(dep_time, na_rm = TRUE),
    fifth_dep = nth(dep_time, 5, na_rm = TRUE),
    last_dep = last(dep_time, na_rm = TRUE)
  )

flights |>
  group_by(year, month, day) |>
  mutate(r = min_rank(sched_dep_time)) |>
  filter(r %in% c(1, max(r)))
```

2. FACTORS

[from https://r4ds.hadley.nz/factors]

```
library("tidyverse") #collection of packages for data analysis
```

2.1. BASICS

```
#use just strings
x1 <- c("Dec", "Apr", "Jan", "Mar")
x2 <- c("Dec", "Apr", "Jam", "Mar")
sort(x1)

#simple factors
month_levels <- c(
  "Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
)
y1 <- factor(x1, levels = month_levels)
y1
sort(y1)
y2 <- factor(x2, levels = month_levels)
```

```
#use fct()
y2 <- fct(x2, levels = month_levels)
```

```
levels(y2)

factor(x1)
fct(x1)

#use col_factor()
csv <- "
month,value
Jan,12
Feb,56
Mar,12"

df <- read_csv(csv, col_types = cols(month = col_factor(month_levels)))
df$month
```

2.2. DATASET gss_cat

```
#General Social Survey
gss_cat
```

```
?gss_cat
```

```
#look at factors
gss_cat |>
  count(race)
```

```
#reorder in ggplot
relig_summary <- gss_cat |>
  group_by(relig) |>
  summarize(
    tvhours = mean(tvhours, na.rm = TRUE),
    n = n()
  )

ggplot(relig_summary, aes(x = tvhours, y = relig)) +
  geom_point()

ggplot(relig_summary, aes(x = tvhours, y = fct_reorder(relig, tvhours))) +
  geom_point()

#reorder in tibble
relig_summary |>
  mutate(
    relig = fct_reorder(relig, tvhours)
  ) |>
  ggplot(aes(x = tvhours, y = relig)) +
  geom_point()

#reorder and relevel
rincome_summary <- gss_cat |>
  group_by(rincome) |>
  summarize(
    age = mean(age, na.rm = TRUE),
    n = n()
  )

ggplot(rincome_summary, aes(x = age, y = fct_reorder(rincome, age))) +
  geom_point()

ggplot(rincome_summary, aes(x = age, y = fct_relevel(rincome, "Not applicable"))) +
  geom_point()

by_age <- gss_cat |>
  filter(!is.na(age)) |>
  count(age, marital) |>
  group_by(age) |>
  mutate(
    prop = n / sum(n)
  )

#reorder and legend order
ggplot(by_age, aes(x = age, y = prop, color = marital)) +
  geom_line(linewidth = 1) +
  scale_color_brewer(palette = "Set1")

ggplot(by_age, aes(x = age, y = prop, color = fct_reorder2(marital, age, prop))) +
  geom_line(linewidth = 1) +
  scale_color_brewer(palette = "Set1") +
  labs(color = "marital")

#reorder and bar plots
gss_cat |>
  mutate(marital = marital |> fct_infreq() |> fct_rev()) |>
  ggplot(aes(x = marital)) +
  geom_bar()
```

2.4. MODIFYING FACTOR LEVELS

```
gss_cat |> count(partyid)

#ommit levels using fct_recode()
gss_cat |>
  mutate(
    partyid = fct_recode(partyid,
      "Republican, strong"    = "Strong republican",
      "Republican, weak"      = "Not str republican",
      "Independent, near rep" = "Ind,near rep",
      "Independent, near dem" = "Ind,near dem",
      "Democrat, weak"        = "Not str democrat",
      "Democrat, strong"      = "Strong democrat"
    )
  ) |>
  count(partyid)

#combine levels using fct_recode()
gss_cat |>
  mutate(
    partyid = fct_recode(partyid,
      "Republican, strong"    = "Strong republican",
      "Republican, weak"      = "Not str republican",
      "Independent, near rep" = "Ind,near rep",
      "Independent, near dem" = "Ind,near dem",
      "Democrat, weak"        = "Not str democrat",
      "Democrat, strong"      = "Strong democrat",
      "Other"                 = "No answer",
      "Other"                 = "Don't know",
      "Other"                 = "Other party"
    )
  )

#combine levels using fct_collapse()
gss_cat |>
  mutate(
    partyid = fct_collapse(partyid,
      "other" = c("No answer", "Don't know", "Other party"),
      "rep" = c("Strong republican", "Not str republican"),
      "ind" = c("Ind,near rep", "Independent", "Ind,near dem"),
      "dem" = c("Not str democrat", "Strong democrat")
    )
  ) |>
  count(partyid)
```

```r
#lump unfrequent groups using fct_lump_lowfreq()
gss_cat |>
  mutate(relig = fct_lump_lowfreq(relig)) |>
  count(relig)

#lump unfrequent groups using fct_lump_n()
gss_cat |>
  mutate(relig = fct_lump_n(relig, n = 10)) |>
  count(relig, sort = TRUE)
```

---

3. LOGICAL VECTORS

[from https://r4ds.hadley.nz/logicals]

```
library("nycflights13") #collection of datasets
library("tidyverse")    #collection of packages for data analysis
```

3.1. COMPARISONS

```
#create logical vector inline
flights |>
  filter(dep_time > 600 & dep_time < 2000 & abs(arr_delay) < 20)

#create logical vector outside logical condition
flights |>
  mutate(
    daytime = dep_time > 600 & dep_time < 2000,
    approx_ontime = abs(arr_delay) < 20,
  ) |>
  filter(daytime & approx_ontime)
```

Missing values

```
#logical conditions
NA > 5
10 == NA
NA == NA
flights |>
  filter(dep_time == NA)

#is.na()
is.na(c(TRUE, NA, FALSE))
is.na(c(1, NA, 3))
is.na(c("a", NA, "b"))
flights |>
  filter(is.na(dep_time))
```

3.2. BOOLEAN ALGEBRA

Boolean operations

```
x <- c(rep(TRUE,6),rep(FALSE,3))
y <- c(rep(FALSE,3),rep(TRUE,6))
which(x)          #x
which(y)          #y
which(x & !y)     #x & !y
which(!x & y)     #!x & y
which(xor(x, y)) #xor(x, y)
which(x | y)      # x | y
```

Missing values

```r
df <- tibble(x = c(TRUE, FALSE, NA))

df |>
  mutate(
    and = x & NA,
    or = x | NA
  )
```

Operator %in%

```r
flights |>
  filter(month == 11 | month == 12)
flights |>
  filter(month %in% c(11, 12))

c(1, 2, NA) == NA
c(1, 2, NA) %in% NA
```

3.3. SUMMARIES

Logical summaries

```r
flights |>
  group_by(year, month, day) |>
  summarize(
    all_delayed = all(dep_delay <= 60, na.rm = TRUE),
    any_long_delay = any(arr_delay >= 300, na.rm = TRUE),
    .groups = "drop"
  )
```

Numeric summaries of logical vectors

```r
flights |>
  group_by(year, month, day) |>
  summarize(
    all_delayed = mean(dep_delay <= 60, na.rm = TRUE),
    any_long_delay = sum(arr_delay >= 300, na.rm = TRUE),
    .groups = "drop"
  )
```

Logical subsetting

```r
flights |>
  filter(arr_delay > 0) |>
  group_by(year, month, day) |>
  summarize(
    behind = mean(arr_delay),
    n = n(),
    .groups = "drop"
  )

flights |>
```

```
  group_by(year, month, day) |>
  summarize(
    behind = mean(arr_delay[arr_delay > 0], na.rm = TRUE),
    ahead = mean(arr_delay[arr_delay < 0], na.rm = TRUE),
    n = n(),
    .groups = "drop"
  )
```

3.4. CONDITIONAL TRANSFORMATIONS

Function if_else()

```
#simple if_else()
x <- c(-3:3, NA)
if_else(x > 0, "+ve", "-ve")
if_else(x > 0, "+ve", "-ve", "???")
if_else(x < 0, -x, x)

#sequence of if_else()
if_else(x == 0, "0",
if_else(x < 0,  "-ve",
                "+ve"),
                "???")
```

Function case_when()

```
case_when(
  x == 0   ~ "0",
  x < 0    ~ "-ve",
  x > 0    ~ "+ve",
  is.na(x) ~ "???"
)

case_when(
  x < 0 ~ "-ve",
  x > 0 ~ "+ve"
)

case_when(
  x < 0 ~ "-ve",
  x > 0 ~ "+ve",
  TRUE ~ "???"
)

case_when(
  x > 0 ~ "+ve",
  x > 2 ~ "big"
)

flights |>
  mutate(
    status = case_when(
      is.na(arr_delay)      ~ "cancelled",
      arr_delay < -30        ~ "very early",
      arr_delay < -15        ~ "early",
      abs(arr_delay) <= 15  ~ "on time",
      arr_delay < 60         ~ "late",
      arr_delay < Inf        ~ "very late",
    ),
    .keep = "used"
  )
```

Compatible types

```
if_else(TRUE, "a", 1)

case_when(
  x < -1 ~ TRUE,
  x > 0  ~ now()
)
```