Data management

Working with a single data frame

Matilde Trevisani

We...

have a single data frame

want to slice it, and dice it, and juice it, and process it

Data: Hotel bookings

- Data from two hotels: one resort and one city hotel
- Observations: Each row represents a hotel booking

```
hotels <- read_csv("data/hotels.csv")
```

`select`, `arrange`, and `slice`

`select` to keep variables

```
hotels %>%
  select(hotel, lead_time)
```

```
## # A tibble: 119,390 × 2
##   hotel        lead_time
##   <chr>            <dbl>
## 1 Resort Hotel       342
## 2 Resort Hotel       737
## 3 Resort Hotel         7
## 4 Resort Hotel        13
## 5 Resort Hotel        14
## 6 Resort Hotel        14
## # ℹ 119,384 more rows
```

`select` to exclude variables

```
hotels %>%
  select(-agent)
```

```
## # A tibble: 119,390 × 31
##   hotel        is_canceled lead_time arrival_date_year
##   <chr>              <dbl>     <dbl>             <dbl>
## 1 Resort Hotel           0       342              2015
## 2 Resort Hotel           0       737              2015
## 3 Resort Hotel           0         7              2015
## 4 Resort Hotel           0        13              2015
## 5 Resort Hotel           0        14              2015
## 6 Resort Hotel           0        14              2015
## # ℹ 119,384 more rows
## # ℹ 27 more variables: arrival_date_month <chr>,
## #   arrival_date_week_number <dbl>,
## #   arrival_date_day_of_month <dbl>,
## #   stays_in_weekend_nights <dbl>, stays_in_week_nights <dbl>,
## #   adults <dbl>, children <dbl>, babies <dbl>, meal <chr>,
## #   country <chr>, market_segment <chr>, …
```

`select` a range of variables

```
hotels %>%
  select(hotel:arrival_date_month)
```

```
## # A tibble: 119,390 × 5
##   hotel        is_canceled lead_time arrival_date_year
##   <chr>              <dbl>     <dbl>             <dbl>
## 1 Resort Hotel           0       342              2015
## 2 Resort Hotel           0       737              2015
## 3 Resort Hotel           0         7              2015
## 4 Resort Hotel           0        13              2015
## 5 Resort Hotel           0        14              2015
## 6 Resort Hotel           0        14              2015
## # ℹ 119,384 more rows
## # ℹ 1 more variable: arrival_date_month <chr>
```

`select` variables with certain characteristics

```
hotels %>%
  select(starts_with("arrival"))
```

```
## # A tibble: 119,390 × 4
##   arrival_date_year arrival_date_month arrival_date_week_number
##               <dbl> <chr>                                  <dbl>
## 1              2015 July                                      27
## 2              2015 July                                      27
## 3              2015 July                                      27
## 4              2015 July                                      27
## 5              2015 July                                      27
## 6              2015 July                                      27
## # ℹ 119,384 more rows
## # ℹ 1 more variable: arrival_date_day_of_month <dbl>
```

`select` variables with certain characteristics

```
hotels %>%
  select(ends_with("type"))
```

```
## # A tibble: 119,390 × 4
##   reserved_room_type assigned_room_type deposit_type
##   <chr>              <chr>              <chr>
## 1 C                  C                  No Deposit
## 2 C                  C                  No Deposit
## 3 A                  C                  No Deposit
## 4 A                  A                  No Deposit
## 5 A                  A                  No Deposit
## 6 A                  A                  No Deposit
## # ℹ 119,384 more rows
## # ℹ 1 more variable: customer_type <chr>
```

Select helpers

- `starts_with()`: Starts with a prefix
- `ends_with()`: Ends with a suffix
- `contains()`: Contains a literal string
- `num_range()`: Matches a numerical range like x01, x02, x03
- `one_of()`: Matches variable names in a character vector
- `everything()`: Matches all variables
- `last_col()`: Select last variable, possibly with an offset
- `matches()`: Matches a regular expression (a sequence of symbols/characters expressing a string/pattern to be searched for within text)

See help for any of these functions for more info, e.g. `?everything`.

`pull` to select one variable as a vector

```
hotels %>%
  pull(lead_time)
```

`pull` creates a vector whereas `select` creates a data frame.

Basically, `pull` is the equivalent to writing `hotels$lead_time` or `hotels[,'lead_time']`, whereas `select` removes all of the columns except for `lead_time` but maintains the data frame structure.

```
hotels %>%
  pull(lead_time) %>%
  class()
```

```
## [1] "numeric"
```

arrange in ascending / descending order

```
hotels %>%
  select(adults, children, babies) %>%
  arrange(babies)
```

```
## # A tibble: 119,390 × 3
##   adults children babies
##    <dbl>    <dbl>  <dbl>
## 1      2        0      0
## 2      2        0      0
## 3      1        0      0
## 4      1        0      0
## 5      2        0      0
## 6      2        0      0
## # ℹ 119,384 more rows
```

```
hotels %>%
  select(adults, children, babies) %>%
  arrange(desc(babies))
```

```
## # A tibble: 119,390 × 3
##   adults children babies
##    <dbl>    <dbl>  <dbl>
## 1      2        0     10
## 2      1        0      9
## 3      2        0      2
## 4      2        0      2
## 5      2        0      2
## 6      2        0      2
## # ℹ 119,384 more rows
```

`slice` for certain row numbers

```
# first five
hotels %>%
  slice(1:5)
```

```
## # A tibble: 5 × 32
##   hotel        is_canceled lead_time arrival_date_year
##   <chr>              <dbl>     <dbl>             <dbl>
## 1 Resort Hotel           0       342              2015
## 2 Resort Hotel           0       737              2015
## 3 Resort Hotel           0         7              2015
## 4 Resort Hotel           0        13              2015
## 5 Resort Hotel           0        14              2015
## # ℹ 28 more variables: arrival_date_month <chr>,
## #   arrival_date_week_number <dbl>,
## #   arrival_date_day_of_month <dbl>,
## #   stays_in_weekend_nights <dbl>, stays_in_week_nights <dbl>,
## #   adults <dbl>, children <dbl>, babies <dbl>, meal <chr>,
## #   country <chr>, market_segment <chr>,
## #   distribution_channel <chr>, is_repeated_guest <dbl>, …
```

```r
hotels %>%
  # slice the first five rows  # this line is a comment
  # select(hotel) %>%           # this one doesn't run
  slice(1:5)                   # this line runs
```

```
## # A tibble: 5 × 32
##   hotel        is_canceled lead_time arrival_date_year
##   <chr>              <dbl>     <dbl>             <dbl>
## 1 Resort Hotel           0       342              2015
## 2 Resort Hotel           0       737              2015
## 3 Resort Hotel           0         7              2015
## 4 Resort Hotel           0        13              2015
## 5 Resort Hotel           0        14              2015
## # ℹ 28 more variables: arrival_date_month <chr>,
## #   arrival_date_week_number <dbl>,
...
```

In R, you can use the # for adding comments to your code. Any text following # will be printed as is, and won't be run as R code. This is useful for leaving comments in your code and for temporarily disabling certain lines of code while debugging.

filter

`filter` to select a subset of rows

```
# bookings in City Hotels
hotels %>%
  filter(hotel == "City Hotel")
```

```
## # A tibble: 79,330 × 32
##   hotel      is_canceled lead_time arrival_date_year
##   <chr>            <dbl>     <dbl>             <dbl>
## 1 City Hotel           0         6              2015
## 2 City Hotel           1        88              2015
## 3 City Hotel           1        65              2015
## 4 City Hotel           1        92              2015
## 5 City Hotel           1       100              2015
## 6 City Hotel           1        79              2015
## # ℹ 79,324 more rows
## # ℹ 28 more variables: arrival_date_month <chr>,
## #   arrival_date_week_number <dbl>,
## #   arrival_date_day_of_month <dbl>,
## #   stays_in_weekend_nights <dbl>, stays_in_week_nights <dbl>,
## #   adults <dbl>, children <dbl>, babies <dbl>, meal <chr>,
## #   country <chr>, market_segment <chr>, …
```

`filter` for many conditions at once

```
hotels %>%
  filter(
    adults == 0,
    children >= 1
    ) %>%
  select(adults, babies, children)
```

```
## # A tibble: 223 × 3
##   adults babies children
##    <dbl>  <dbl>    <dbl>
## 1      0      0        3
## 2      0      0        2
## 3      0      0        2
## 4      0      0        2
## 5      0      0        2
## 6      0      0        3
## # ℹ 217 more rows
```

`filter` for more complex conditions

```
# bookings with no adults and some children or babies in the room
hotels %>%
  filter(
    adults == 0,
    children >= 1 | babies >= 1     # | means or
    ) %>%
  select(adults, babies, children)
```

```
## # A tibble: 223 × 3
##   adults babies children
##    <dbl>  <dbl>    <dbl>
## 1      0      0        3
## 2      0      0        2
## 3      0      0        2
## 4      0      0        2
## 5      0      0        2
## 6      0      0        3
## # ℹ 217 more rows
```

Logical operators in R

operator	definition	operator	definition
`<`	less than	`x | y`	`x` OR `y`
`<=`	less than or equal to	`is.na(x)`	test if `x` is `NA`
`>`	greater than	`!is.na(x)`	test if `x` is not `NA`
`>=`	greater than or equal to	`x %in% y`	test if `x` is in `y`
`==`	exactly equal to	`!(x %in% y)`	test if `x` is not in `y`
`!=`	not equal to	`!x`	not `x`
`x & y`	`x` AND `y`		

distinct and count

`distinct` to filter for unique rows

... and `arrange` to order alphabetically

```
hotels %>%
  distinct(market_segment) %>%
  arrange(market_segment)
```

```
## # A tibble: 8 × 1
##   market_segment
##   <chr>
## 1 Aviation
## 2 Complementary
## 3 Corporate
## 4 Direct
## 5 Groups
## 6 Offline TA/TO
## 7 Online TA
## 8 Undefined
```

```
hotels %>%
  distinct(hotel, market_segment) %>%
  arrange(hotel, market_segment)
```

```
## # A tibble: 14 × 2
##    hotel        market_segment
##    <chr>        <chr>
##  1 City Hotel   Aviation
##  2 City Hotel   Complementary
##  3 City Hotel   Corporate
##  4 City Hotel   Direct
##  5 City Hotel   Groups
##  6 City Hotel   Offline TA/TO
##  7 City Hotel   Online TA
##  8 City Hotel   Undefined
##  9 Resort Hotel Complementary
## 10 Resort Hotel Corporate
...
```

`count` to create frequency tables

```
# alphabetical order by default
hotels %>%
  count(market_segment)
```

```
## # A tibble: 8 × 2
##   market_segment     n
##   <chr>          <int>
## 1 Aviation         237
## 2 Complementary    743
## 3 Corporate       5295
## 4 Direct         12606
## 5 Groups         19811
## 6 Offline TA/TO  24219
## 7 Online TA      56477
## 8 Undefined          2
```

```
# descending frequency order
hotels %>%
  count(market_segment, sort = TRUE)
```

```
## # A tibble: 8 × 2
##   market_segment     n
##   <chr>          <int>
## 1 Online TA      56477
## 2 Offline TA/TO  24219
## 3 Groups         19811
## 4 Direct         12606
## 5 Corporate       5295
## 6 Complementary    743
## 7 Aviation         237
## 8 Undefined          2
```

count and arrange

```
# ascending frequency order
hotels %>%
  count(market_segment) %>%
  arrange(n)
```

```
## # A tibble: 8 × 2
##   market_segment     n
##   <chr>          <int>
## 1 Undefined          2
## 2 Aviation         237
## 3 Complementary    743
## 4 Corporate       5295
## 5 Direct         12606
## 6 Groups         19811
## 7 Offline TA/TO  24219
## 8 Online TA      56477
```

```
# descending frequency order
# just like adding sort = TRUE
hotels %>%
  count(market_segment) %>%
  arrange(desc(n))
```

```
## # A tibble: 8 × 2
##   market_segment     n
##   <chr>          <int>
## 1 Online TA      56477
## 2 Offline TA/TO  24219
## 3 Groups         19811
## 4 Direct         12606
## 5 Corporate       5295
## 6 Complementary    743
## 7 Aviation         237
## 8 Undefined          2
```

`count` for multiple variables

```
hotels %>%
  count(hotel, market_segment)
```

```
## # A tibble: 14 × 3
##    hotel        market_segment     n
##    <chr>        <chr>          <int>
##  1 City Hotel   Aviation         237
##  2 City Hotel   Complementary    542
##  3 City Hotel   Corporate       2986
##  4 City Hotel   Direct          6093
##  5 City Hotel   Groups         13975
##  6 City Hotel   Offline TA/TO  16747
##  7 City Hotel   Online TA      38748
##  8 City Hotel   Undefined          2
##  9 Resort Hotel Complementary    201
## 10 Resort Hotel Corporate       2309
## 11 Resort Hotel Direct          6513
## 12 Resort Hotel Groups          5836
## 13 Resort Hotel Offline TA/TO   7472
## 14 Resort Hotel Online TA      17729
```

order matters when you count

```
# hotel type first
hotels %>%
  count(hotel, market_segment)
```

```
## # A tibble: 14 × 3
##    hotel        market_segment     n
##    <chr>        <chr>          <int>
##  1 City Hotel   Aviation         237
##  2 City Hotel   Complementary    542
##  3 City Hotel   Corporate       2986
##  4 City Hotel   Direct          6093
##  5 City Hotel   Groups         13975
##  6 City Hotel   Offline TA/TO  16747
##  7 City Hotel   Online TA      38748
##  8 City Hotel   Undefined          2
##  9 Resort Hotel Complementary    201
## 10 Resort Hotel Corporate       2309
## 11 Resort Hotel Direct          6513
## 12 Resort Hotel Groups          5836
## 13 Resort Hotel Offline TA/TO   7472
## 14 Resort Hotel Online TA      17729
```

```
# market segment first
hotels %>%
  count(market_segment, hotel)
```

```
## # A tibble: 14 × 3
##    market_segment hotel            n
##    <chr>          <chr>        <int>
##  1 Aviation       City Hotel     237
##  2 Complementary  City Hotel     542
##  3 Complementary  Resort Hotel   201
##  4 Corporate      City Hotel    2986
##  5 Corporate      Resort Hotel  2309
##  6 Direct         City Hotel    6093
##  7 Direct         Resort Hotel  6513
##  8 Groups         City Hotel   13975
##  9 Groups         Resort Hotel  5836
## 10 Offline TA/TO  City Hotel   16747
## 11 Offline TA/TO  Resort Hotel  7472
## 12 Online TA      City Hotel   38748
## 13 Online TA      Resort Hotel 17729
## 14 Undefined      City Hotel       2
```

Your turn!

- Laboratorio "Datamanaging"
  - Esercizio 1
  - Esercizio 2, parte I.

mutate

- `mutate` changes an existing variable if you use in the function the name of an already existing variable
- `mutate` adds a new variable if you define a new variable using a different name from the variable names already exixting
- Use `transmute` if you only want to keep the new variables.

`mutate` to add a new variable

```
hotels %>%
  mutate(little_ones = children + babies) %>%
  select(children, babies, little_ones) %>%
  arrange(desc(little_ones))
```

```
## # A tibble: 119,390 × 3
##   children babies little_ones
##      <dbl>  <dbl>       <dbl>
## 1       10      0          10
## 2        0     10          10
## 3        0      9           9
## 4        2      1           3
## 5        2      1           3
## 6        2      1           3
## # ℹ 119,384 more rows
```

Little ones in resort and city hotels

```
# Resort Hotel
hotels %>%
  mutate(little_ones = children + babies) %>%
  filter(
    little_ones >= 1,
    hotel == "Resort Hotel"
    ) %>%
  select(hotel, little_ones)
```

```
## # A tibble: 3,929 × 2
##   hotel        little_ones
##   <chr>              <dbl>
## 1 Resort Hotel           1
## 2 Resort Hotel           2
## 3 Resort Hotel           2
## 4 Resort Hotel           2
## 5 Resort Hotel           1
## 6 Resort Hotel           1
## # ℹ 3,923 more rows
```

```
# City Hotel
hotels %>%
  mutate(little_ones = children + babies) %>%
  filter(
    little_ones >= 1,
    hotel == "City Hotel"
    )  %>%
  select(hotel, little_ones)
```

```
## # A tibble: 5,403 × 2
##   hotel      little_ones
##   <chr>            <dbl>
## 1 City Hotel           1
## 2 City Hotel           1
## 3 City Hotel           2
## 4 City Hotel           1
## 5 City Hotel           1
## 6 City Hotel           1
## # ℹ 5,397 more rows
```

What is happening in the following chunk?

```
hotels %>%
  mutate(little_ones = children + babies) %>%
  count(hotel, little_ones) %>%
  mutate(prop = n / sum(n))
```

```
## # A tibble: 12 × 4
##    hotel        little_ones     n       prop
##    <chr>              <dbl> <int>      <dbl>
##  1 City Hotel             0 73923 0.619
##  2 City Hotel             1  3263 0.0273
##  3 City Hotel             2  2056 0.0172
##  4 City Hotel             3    82 0.000687
##  5 City Hotel             9     1 0.00000838
##  6 City Hotel            10     1 0.00000838
##  7 City Hotel            NA     4 0.0000335
##  8 Resort Hotel           0 36131 0.303
##  9 Resort Hotel           1  2183 0.0183
## 10 Resort Hotel           2  1716 0.0144
## 11 Resort Hotel           3    29 0.000243
## 12 Resort Hotel          10     1 0.00000838
```

`summarise` and `group_by`

`summarise` for summary stats

```
# mean average daily rate for all bookings
# (calculated by dividing the sum of all lodging transactions by the total number of staying night
hotels %>%
  summarise(mean_adr = mean(adr))
```

```
## # A tibble: 1 × 1
##   mean_adr
##      <dbl>
## 1     102.
```

`summarise()` changes the data frame entirely, it collapses rows down to a single summary statistic, and removes all columns that are irrelevant to the calculation.

❌

```
hotels %>%
  summarise(mean(adr))
```

```
## # A tibble: 1 × 1
##   `mean(adr)`
##         <dbl>
## 1        102.
```

✅

```
hotels %>%
  summarise(mean_adr = mean(adr))
```

```
## # A tibble: 1 × 1
##   mean_adr
##      <dbl>
## 1     102.
```

`summarise()` also lets you get away with being sloppy and not naming your new column, but that's not recommended!

group_by for grouped operations

```
# mean average daily rate for all booking at city and resort hotels
hotels %>%
  group_by(hotel) %>%
  summarise(mean_adr = mean(adr))
```

```
## # A tibble: 2 × 2
##   hotel        mean_adr
##   <chr>           <dbl>
## 1 City Hotel      105.
## 2 Resort Hotel     95.0
```

Together `group_by()` and `summarise()` provide one of the tools that you'll use most commonly when working with dplyr: **grouped summaries**.

Calculating frequencies

The following two give the same result, so `count` is simply short for `group_by` then determine frequencies

```
hotels %>%
  group_by(hotel) %>%
  summarise(n = n())
```

```
## # A tibble: 2 × 2
##   hotel            n
##   <chr>        <int>
## 1 City Hotel   79330
## 2 Resort Hotel 40060
```

```
hotels %>%
  count(hotel)
```

```
## # A tibble: 2 × 2
##   hotel            n
##   <chr>        <int>
## 1 City Hotel   79330
## 2 Resort Hotel 40060
```

Multiple summary statistics

`summarise` can be used for multiple summary statistics as well

```
hotels %>%
  summarise(
    min_adr = min(adr),
    mean_adr = mean(adr),
    median_adr = median(adr),
    max_adr = max(adr)
    )
```

```
## # A tibble: 1 × 4
##   min_adr mean_adr median_adr max_adr
##     <dbl>    <dbl>      <dbl>   <dbl>
## 1   -6.38     102.       94.6    5400
```

Your turn!

- Laboratorio "Datamanaging"
  - Esercizio 2, parte II.