vectors and data frames in R

prof. dr. nicolas meseth

1. from a value to a table
2. vectors
3. data frames and tibbles

a type case keeps many small
things in order because every
column holds only one kind of thing.



image: ai-generated (gpt-image-2.5-flare)

part 1

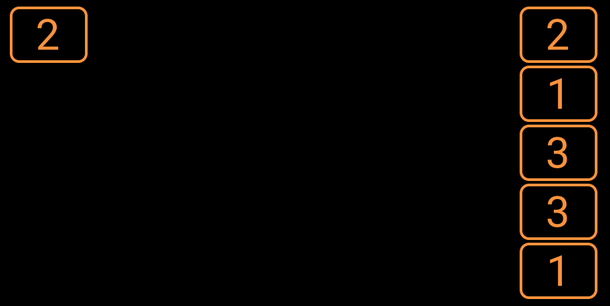
from a value to a table

four shapes of data

2

in a matrix every cell has the same type, in a data frame every column has its own.

four shapes of data

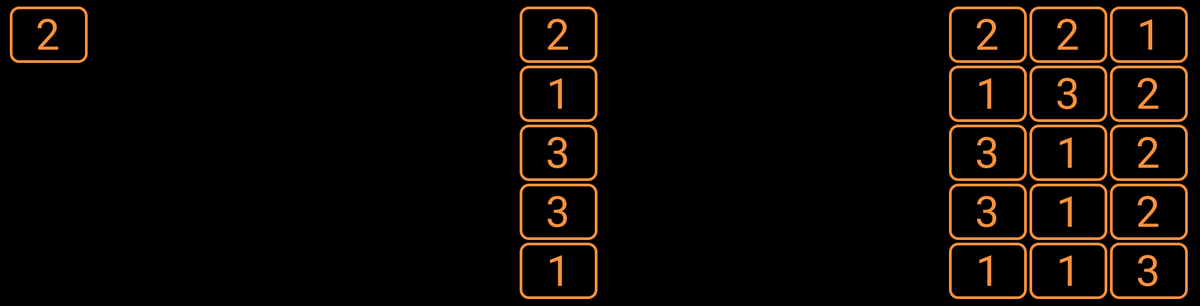


scalar value

vector

in a matrix every cell has the same type, in a data frame every column has its own.

four shapes of data



scalar value

vector

matrix

in a matrix every cell has the same type, in a data frame every column has its own.

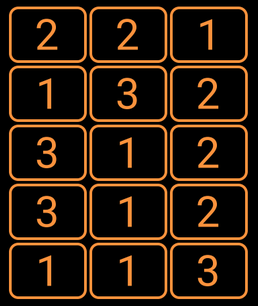
four shapes of data



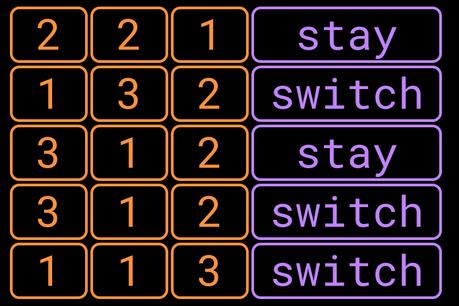
scalar value



vector

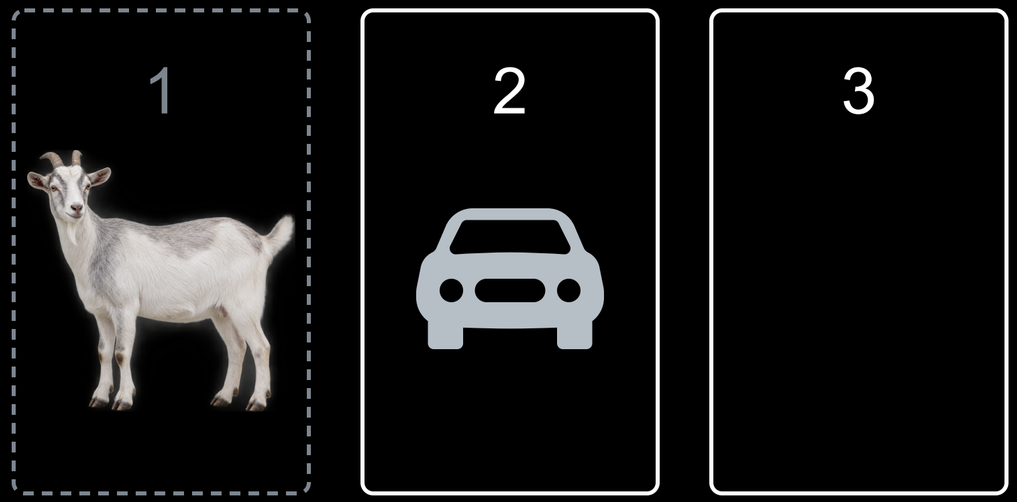


matrix



data frame

in a matrix every cell has the same type, in a data frame every column has its own.



opened by the host your pick

```
prize_door          2
contestant_choice   2
monty_opens         1
decision            "stay"
```

a game show with three doors: you pick one, the host opens another with a goat behind it, and you stay with your door or switch.

image: ai-generated (gpt-image-2.5-flare)

play the game on the site, and stay or switch.

after a few rounds the game shows your rounds as R code, one line per fact.

hands-on-computer-science.de/ds/en/items/monty-hall-game



```
prize_door <- c(2, 1, 3, 3, 1)
contestant_choice <- c(2, 3, 1, 1, 1)
monty_opens <- c(1, 2, 2, 2, 3)
decision <- c("stay", "switch", "stay", "switch", "switch")
```

c() combines values into a vector, and the first value of each vector belongs to the first round.

part 2

vectors

many values of one kind

the same position, the same round

	1	2	3	4	5
prize_door	2	1	3	3	1
contestant_choice	2	3	1	1	1
monty_opens	1	2	2	2	3
decision	stay	switch	stay	switch	switch

four vectors of the same length describe five rounds, as long as nobody mixes up the order.

the same position, the same round

	1	2	3	4	5
prize_door	2	1	3	3	1
contestant_choice	2	3	1	1	1
monty_opens	1	2	2	2	3
decision	stay	switch	stay	switch	switch

round 1: car behind door 2, door 2 picked, host opened door 1, player stayed

four vectors of the same length describe five rounds, as long as nobody mixes up the order.

the same position, the same round

	1	2	3	4	5
prize_door	2	1	3	3	1
contestant_choice	2	3	1	1	1
monty_opens	1	2	2	2	3
decision	stay	switch	stay	switch	switch

round 4: car behind door 3, door 1 picked, host opened door 2, player switched

four vectors of the same length describe five rounds, as long as nobody mixes up the order.

one vector, one type

you write	R holds	typeof()
`c(2, 1, 3)`	2 1 3	`"double"`
`c(2L, 1L, 3L)`	2 1 3	`"integer"`
`c("stay", "switch")`	"stay" "switch"	`"character"`
`c(TRUE, FALSE)`	TRUE FALSE	`"logical"`

every value in a vector has the same type, and typeof() tells you which one.

```
length(prize_door)     # 5
typeof(decision)       # "character"
mean(prize_door)       # 2
table(decision)        # stay 2, switch 3
```

sum, mean, median, min, max, range, sd, sort, unique, table: functions that take a whole vector at once.

```
c(1, TRUE)
```




```
c(1, TRUE, "stay")
```



R never mixes types in a vector, it quietly converts to the most general one.

```
c(1, TRUE)
```




```
"double"
```

```
c(1, TRUE, "stay")
```



R never mixes types in a vector, it quietly converts to the most general one.

`c(1, TRUE)` → `1` `1` → `"double"`

`c(1, TRUE, "stay")` → `"1"` `"TRUE"` `"stay"` → `"character"`

`logical` ⟶ `double` ⟶ `character`

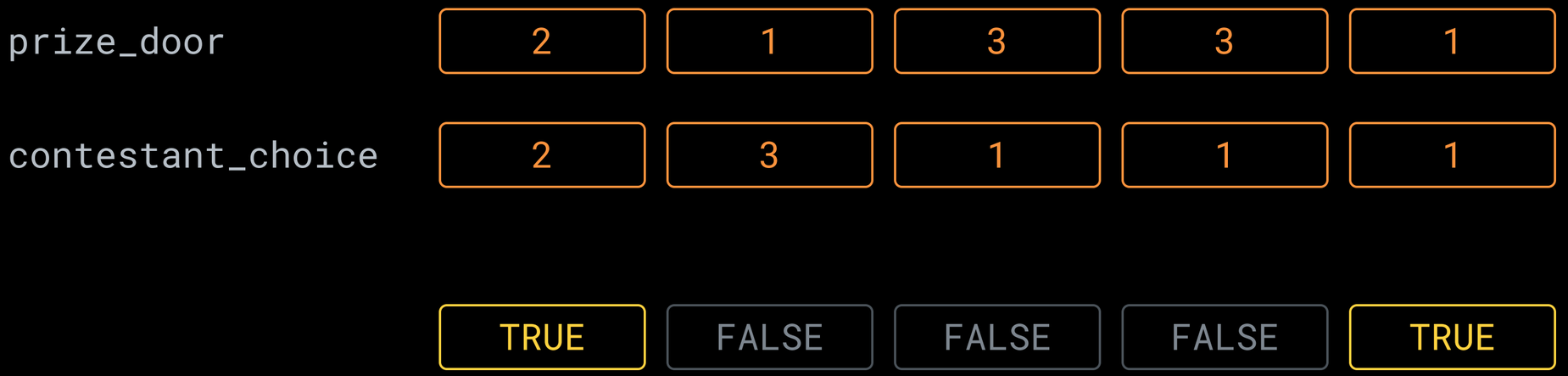
R never mixes types in a vector, it quietly converts to the most general one.

comparing whole vectors



an operation on two vectors works position by position, with no loop to write.

comparing whole vectors



an operation on two vectors works position by position, with no loop to write.

```
first_pick_right <- prize_door == contestant_choice
sum(first_pick_right)      # 2
mean(first_pick_right)     # 0.4
```

R counts TRUE as 1 and FALSE as 0, so the mean of a logical vector is a share.

picking values by position

```
decision[1]          # "stay"
```

square brackets select, a minus leaves out, and the result is again a vector.

picking values by position

```
decision[1]          # "stay"
decision[-1]         # all but the first
```

square brackets select, a minus leaves out, and the result is again a vector.

picking values by position

```
decision[1]          # "stay"
decision[-1]         # all but the first
decision[2:4]        # "switch" "stay" "switch"
```

square brackets select, a minus leaves out, and the result is again a vector.

```
decision[1]          # "stay"
decision[-1]         # all but the first
decision[2:4]        # "switch" "stay" "switch"
decision[c(1, 3)]    # "stay" "stay"
```

square brackets select, a minus leaves out, and the result is again a vector.

`decision == "stay"`

TRUE	FALSE	TRUE	FALSE	FALSE

`first_pick_right`

TRUE	FALSE	FALSE	FALSE	TRUE

a logical vector in square brackets keeps exactly the positions that are TRUE.

picking values by a condition

```
decision == "stay"
```

TRUE	FALSE	TRUE	FALSE	FALSE

```
first_pick_right
```

TRUE	FALSE

```
first_pick_right[decision == "stay"]
```

a logical vector in square brackets keeps exactly the positions that are TRUE.

```
x <- c(2, NA, 3)
```




```
mean(x)                  → NA
mean(x, na.rm = TRUE)    → 2.5
```

related, but not the same

`NULL`	nothing at all
`NaN`	0 / 0, not a number
`Inf`	1 / 0, infinite

NA means "not known", so anything computed with it is not known either, until you say what to do.

```
decision_factor <- factor(
  decision,
  levels = c("stay", "switch")
)
levels(decision_factor)    # "stay" "switch"
```

a factor knows all its possible values and their order, even those that do not occur.

a vector is a row of values of one type, and position `i` belongs to observation `i`.

part 3

data frames and tibbles

vectors side by side

vectors become a table

```
games <- data.frame(prize_door, contestant_choice, monty_opens, decision)
```

prize_door	contestant_choice	monty_opens	decision
2	2	1	stay
1	3	2	switch
3	1	2	stay
3	1	2	switch
1	1	3	switch

a data frame is a list of equally long vectors, and every vector holds values of one type.

vectors become a table

```
games <- data.frame(prize_door, contestant_choice, monty_opens, decision)
```

	prize_door	contestant_choice	monty_opens	decision
1	2	2	1	stay
2	1	3	2	switch
3	3	1	2	stay
4	3	1	2	switch
5	1	1	3	switch

a data frame is a list of equally long vectors, and every vector holds values of one type.

vectors become a table

```
games <- data.frame(prize_door, contestant_choice, monty_opens, decision)
```

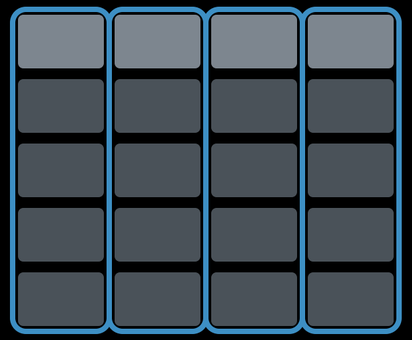
	prize_door	contestant_choice	monty_opens	decision
1	2	2	1	stay
2	1	3	2	switch
3	3	1	2	stay
4	3	1	2	switch
5	1	1	3	switch

one column is one vector

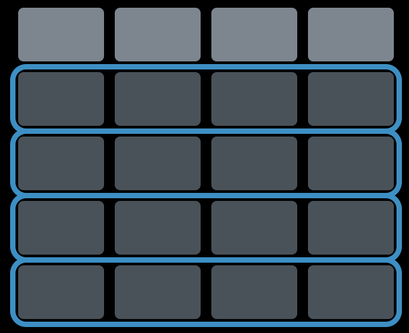
one row is one game

a data frame is a list of equally long vectors, and every vector holds values of one type.

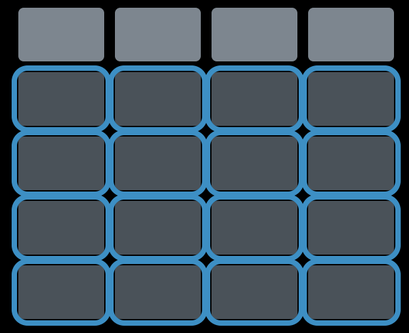
tidy data



each variable is a column



each observation is a row



each value is a cell

when a table is tidy, every column is a vector of one variable, and the tools of R fit it.

after Wickham, Çetinkaya-Rundel & Grolemund: R for Data Science, chapter "data tidying"

`read.csv("lets_make_a_deal.csv")`

```
86       Linda         2                1         3     stay
87      Robert         3                3         2     stay
88     Barbara         1                3         2     stay
89       Karen         2                2         1     stay
90       Sarah         1                3         2   switch
91      Joseph         1                1         3     stay
92     Richard         1                2         3   switch
93     Charles         1                1         3     stay
94       Susan         3                1         2     stay
95    Patricia         1                3         2     stay
96     Richard         1                2         3     stay
97    Patricia         1                3         2   switch
98       Susan         3                3         1     stay
99     Jessica         1                2         3     stay
100   Patricia         1                1         3     stay
```

`read_csv("lets_make_a_deal.csv")`

```
# A tibble: 100 × 5
   contestant prize_door contestant_choice monty_opens decision
   <chr>           <dbl>             <dbl>       <dbl> <chr>
 1 Charles             3                 3           2 stay
 2 Patricia            1                 3           2 stay
 3 Karen               3                 3           1 stay
 4 Jessica             3                 1           2 stay
 5 Joseph              1                 3           2 stay
 6 Robert              1                 3           2 stay
 7 John                3                 1           2 stay
 8 Richard             2                 3           1 switch
 9 Michael             3                 1           2 stay
10 Elizabeth           3                 2           1 switch
# ℹ 90 more rows
```

a tibble is a data frame with better manners: it shows the types and does not flood the console.

```
library(tidyverse)
monty_hall <- read_csv("lets_make_a_deal.csv")
glimpse(monty_hall)
```

```
Rows: 100
Columns: 5
$ contestant        <chr> "Charles", "Patricia", "Karen", "Jessica", "…
$ prize_door        <dbl> 3, 1, 3, 3, 1, 1, 3, 2, 3, 3, 3, 3, 1, 3, 2,…
$ contestant_choice <dbl> 3, 3, 3, 1, 3, 3, 1, 3, 1, 2, 3, 2, 1, 3, 1,…
$ monty_opens       <dbl> 2, 2, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 2, 3,…
$ decision          <chr> "stay", "stay", "stay", "stay", "stay", "sta…
```

glimpse() turns the table on its side and shows every column with its type and first values; it is the first thing to run on new data.

a data frame is a list of equally long vectors, and every vector holds values of one type.

to read on: Hands-On Programming with R, chapter "R objects", and R for Data Science, chapters "data tidying", "logical vectors", "factors" and "missing values"