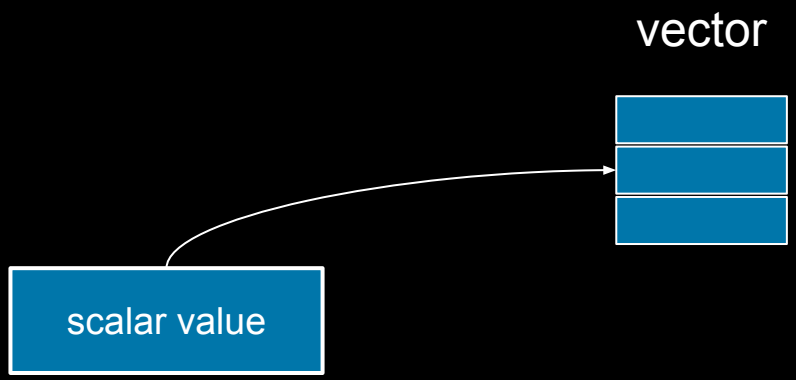
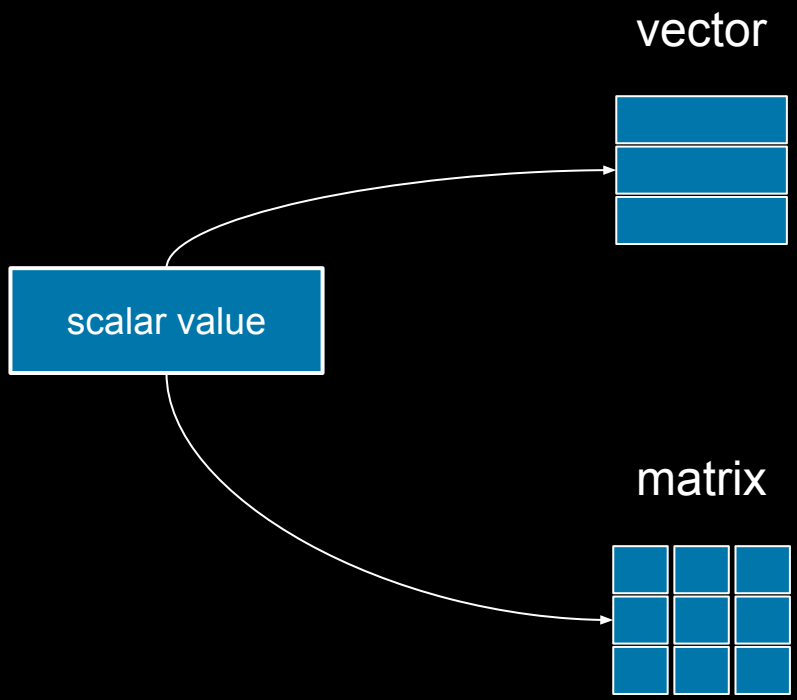
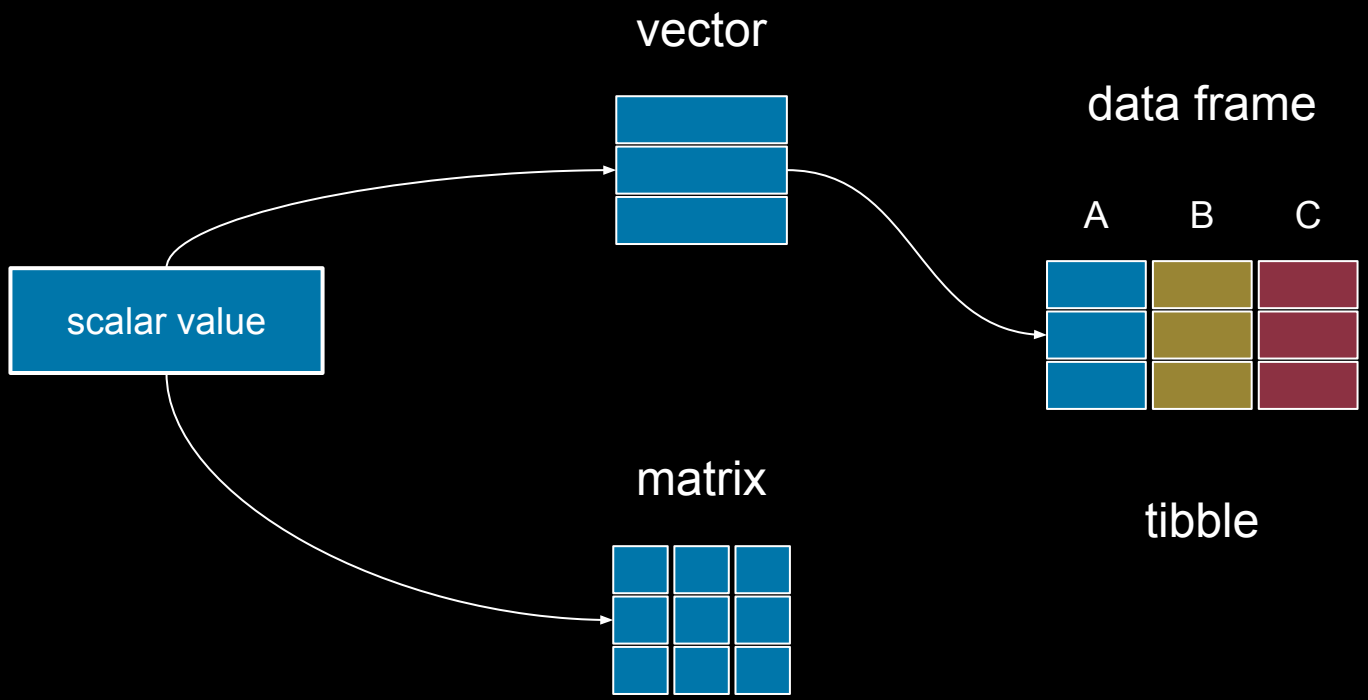
DATA REPRESENTATION

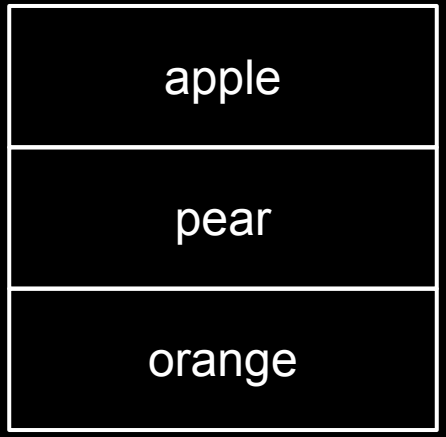

scalar value


vector
scalar value


vector
scalar value
matrix


vector
data frame
A
B
C
scalar value
matrix
tibble

VECTORS


apple
pear
orange

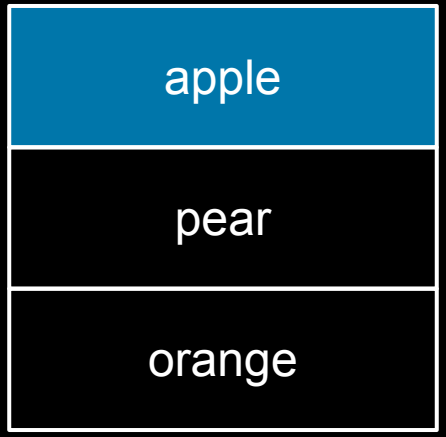
list of values with
the same storage mode

list of values with
the same storage mode

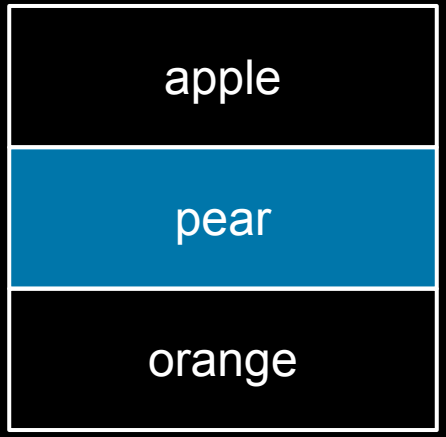
character
double
integer
logical

```
v <- c("apple", "pear", "orange")
```

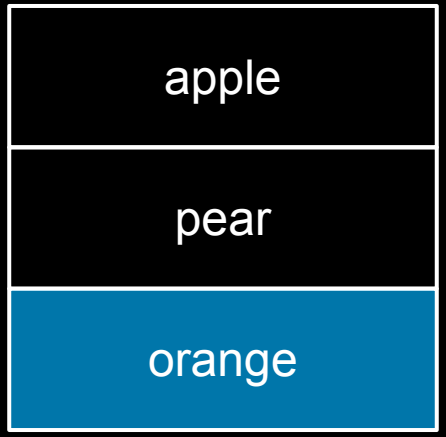
`v[1]`



v[2]



`v[3]`

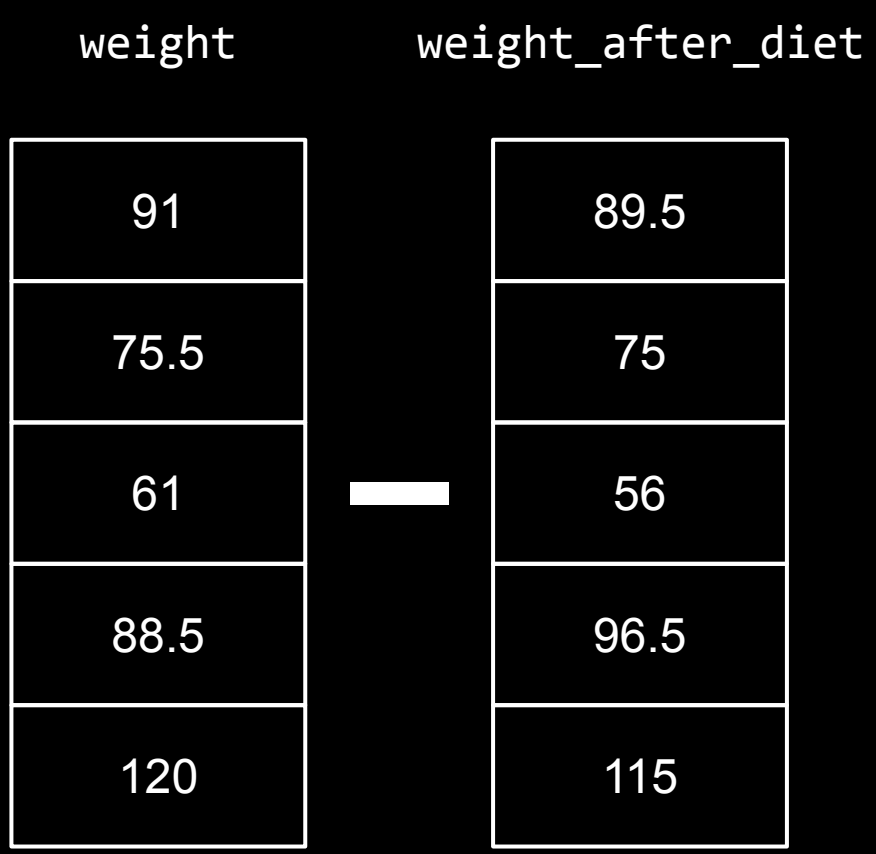


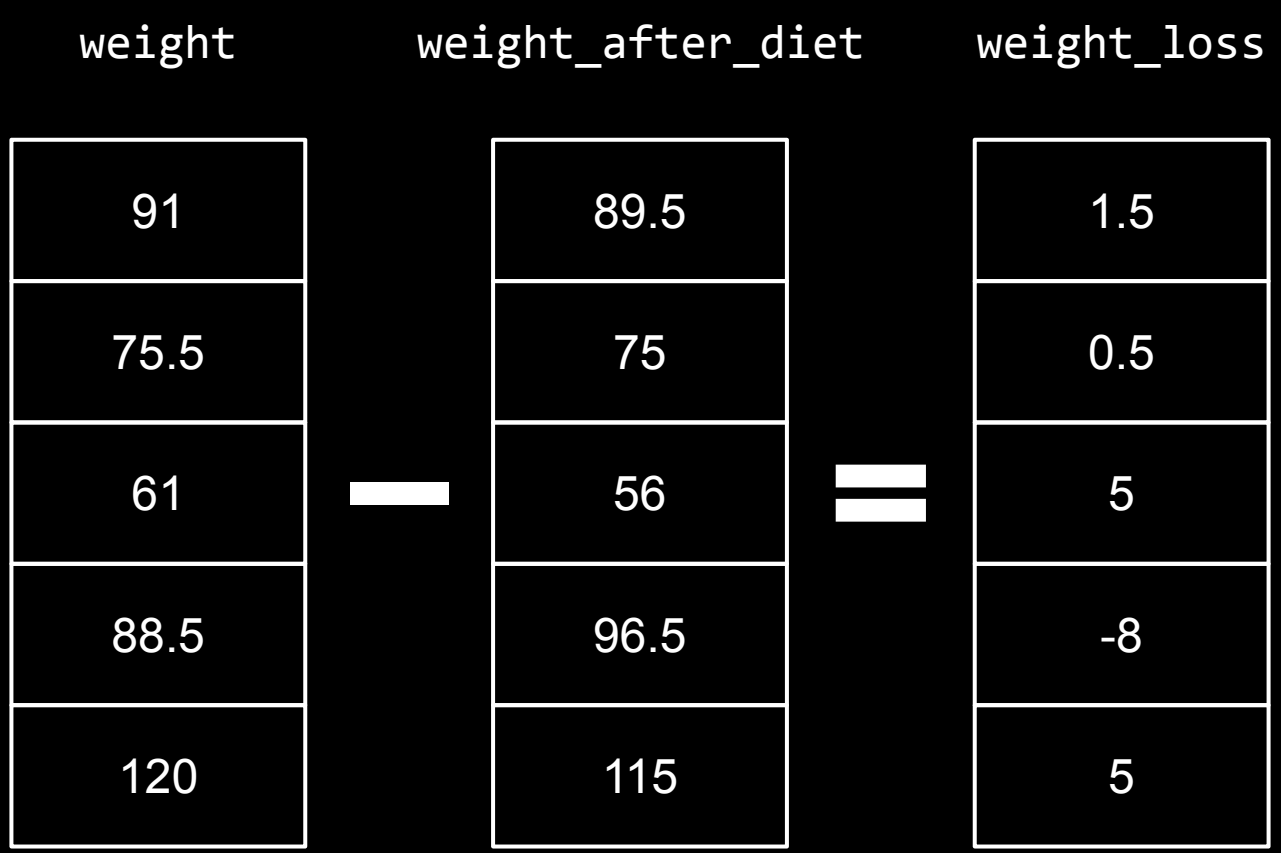
```
weight <- c(91, 75.5, 61, 88.5, 120)
```

```
weight <- c(91, 75.5, 61, 88.5, 120)
mean(weight)
```

```
sum         length
mean        sort
median      cumsum
sd          prod
var         quantile
min         abs
max         range
```

```
weight_after_diet <-
   c(89.5, 75, 56, 96.5, 115)
```


weight
weight_after_diet
91
75.5
61
88.5
120
89.5
75
56
96.5
115


weight
weight_after_diet
weight_loss
91
75.5
61
88.5
120
89.5
75
56
96.5
115
1.5
0.5
5
-8
5

```
weight_loss <-
   weight - weight_after_diet
```

subsetting vectors

weight[1]

```
weight[1]
weight[-1]
```

```
weight[1]
weight[-1]
weight[2:5]
```

```
weight[1]

weight[-1]

weight[2:5]

weight[1:length(weight)-1]
```

```
weight[1]
weight[-1]
weight[2:5]
weight[1:length(weight)-1]
weight[c(TRUE, FALSE, TRUE, TRUE, FALSE)]
```

```
weight[1]

weight[-1]

weight[2:5]

weight[1:length(weight)-1]

weight[c(TRUE, FALSE, TRUE, TRUE, FALSE)]

weight[weight > 80]
```

```
weight[1]

weight[-1]

weight[2:5]

weight[1:length(weight)-1]

weight[c(TRUE, FALSE, TRUE, TRUE, FALSE)]

weight[weight > 80]

weight[weight > 80 & weight < 100]
```

special values

NA
NULL
NaN
Inf
-Inf

factors

```r
category <- factor(c("heavy", "medium", "light", "medium", "heavy"))
```

```r
category <- factor(c("heavy", "medium", "light", "medium", "heavy"))

levels(weight_category)
```

```r
category <- factor(c("heavy", "medium", "light", "medium", "heavy"))

levels(weight_category)

category_reordered <- factor(category,
                             levels = c("light", "medium", "heavy"))
```

```r
category <- factor(c("heavy", "medium", "light", "medium", "heavy"))

levels(weight_category)

category_reordered <- factor(category,
                             levels = c("light", "medium", "heavy"))

category_ordered <- factor(category,
                           levels = c("light", "medium", "heavy"),
                           ordered = TRUE)
```

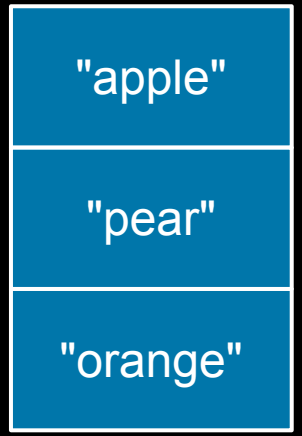
{{ forcats }}

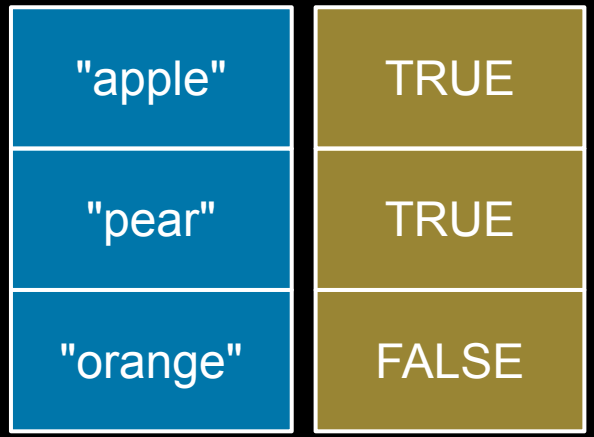
as_factor()

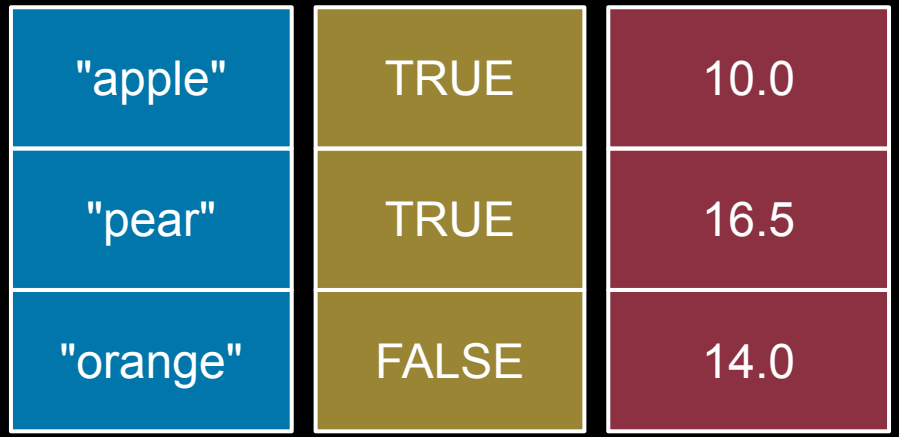
```
fct_reorder
fct_relevel
fct_infreq
fct_rev
fct_lump
```

DATA FRAMES

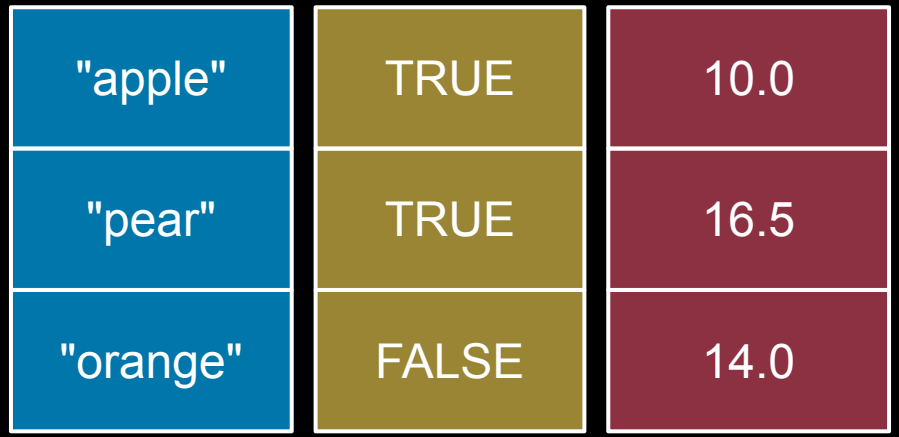
BACK

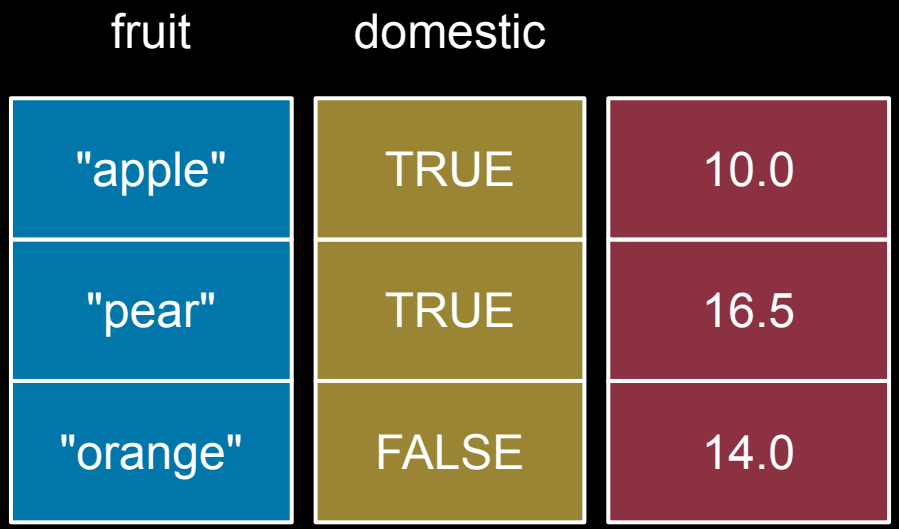

"apple"
"pear"
"orange"

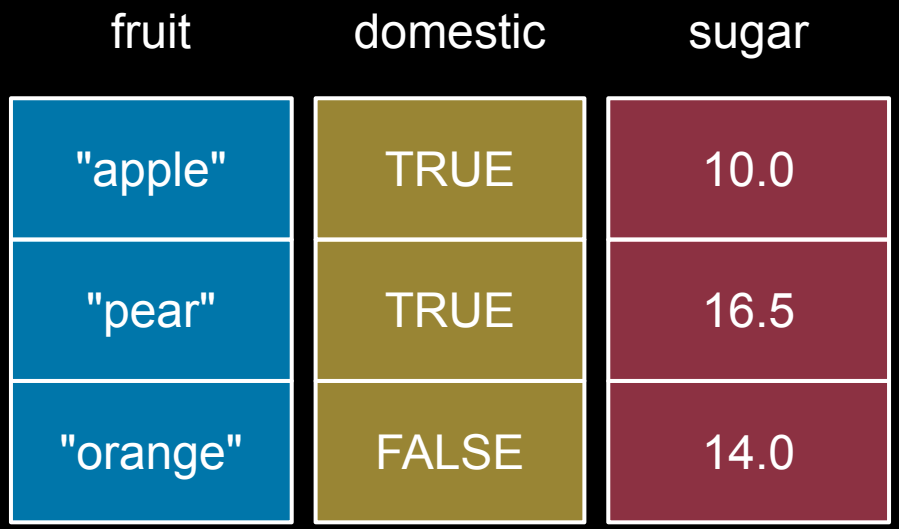

"apple"
TRUE
"pear"
TRUE
"orange"
FALSE

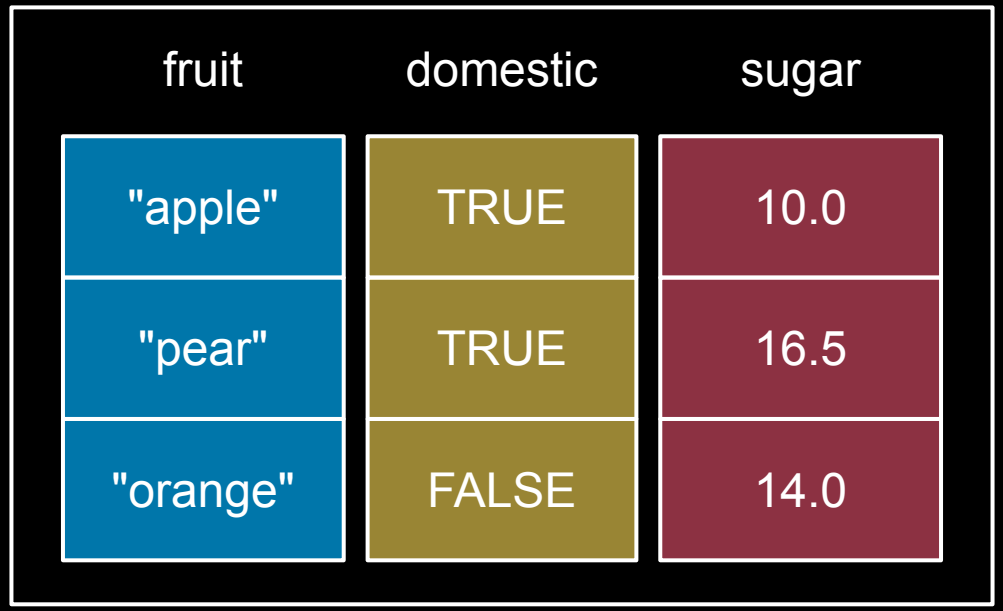

"apple"
TRUE
10.0
"pear"
TRUE
16.5
"orange"
FALSE
14.0

fruit

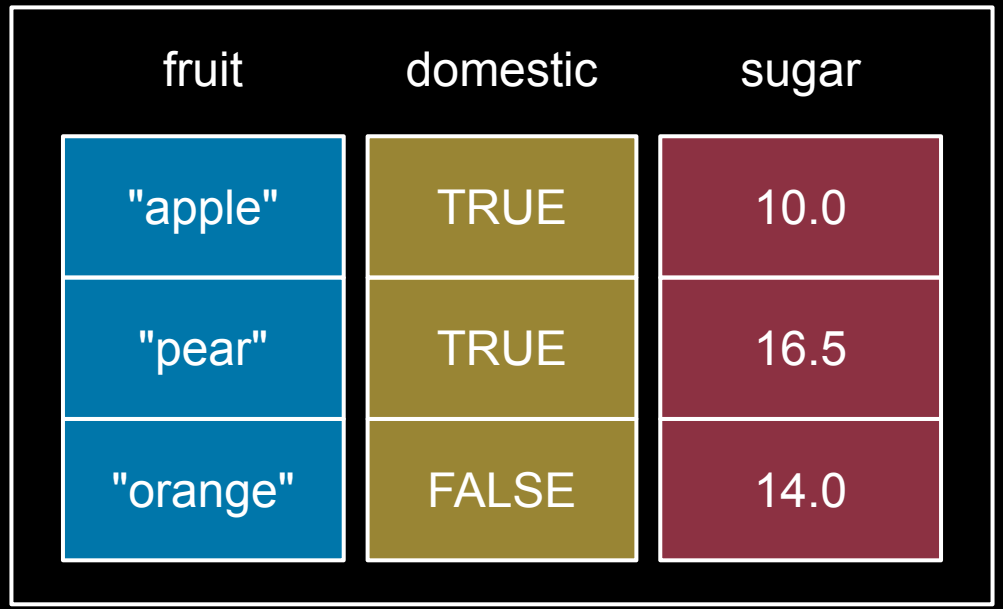



fruit
domestic
"apple"
TRUE
10.0
"pear"
TRUE
16.5
"orange"
FALSE
14.0


fruit
domestic
sugar
"apple"
TRUE
10.0
"pear"
TRUE
16.5
"orange"
FALSE
14.0


fruit
domestic
sugar
"apple"
TRUE
10.0
"pear"
TRUE
16.5
"orange"
FALSE
14.0

data frame "fruits"



creating data frames

```
data.frame()
read.csv()
```

comma separated values (CSV)

data frame meta data

```
ncol()
nrow()
dim()
colnames()
```

accessing data frames

accessing data frames
accessing columns

```
monty$prize_door
monty$contestant_choice
monty$decision
```

```
monty$prize_door
monty$contestant_choice
monty$decision
```

return a vector

```
monty["prize_door"]
monty["contestant_choice"]
monty["decision"]
```

```
monty["prize_door"]
monty["contestant_choice"]
monty["decision"]
```

return a data frame

```
# multiple columns by name
monty(c("prize_door", "contestant_choice"))
```

```
monty[, 1]               # first column
monty[, 1:2]             # first two columns
monty[, ncol(monty)]     # last column
```

accessing data frames

accessing rows

```
monty[1,]                # first row
monty[1:10,]             # first 10 rows
monty[nrow(monty),]      # last row
```

changing columns

```
monty$decision <- as.factor(monty$decision)
```

adding columns

```
monty$correct_guess <-
   monty$contestant_choice == monty$prize_door
```

rename columns

```
colnames(monty)[2] <- "choice"
```

subsetting data frames

```
switched <-
   monty[monty_hall$decision == "switch", ]
```

```
switched <-
   monty[
      monty_hall$decision == "switch &
      monty$won == TRUE, ]
```

subset()

```
subset(monty, decision == "switch")
```

```
subset(
      monty,
      decision == "switch" & won == TRUE
)
```

sorting rows

```
monty[order(monty$prize_door),]
```

```
monty[order(monty$prize_door),]

monty[order(
         monty$prize_door,
         decreasing = TRUE
         ),]
```

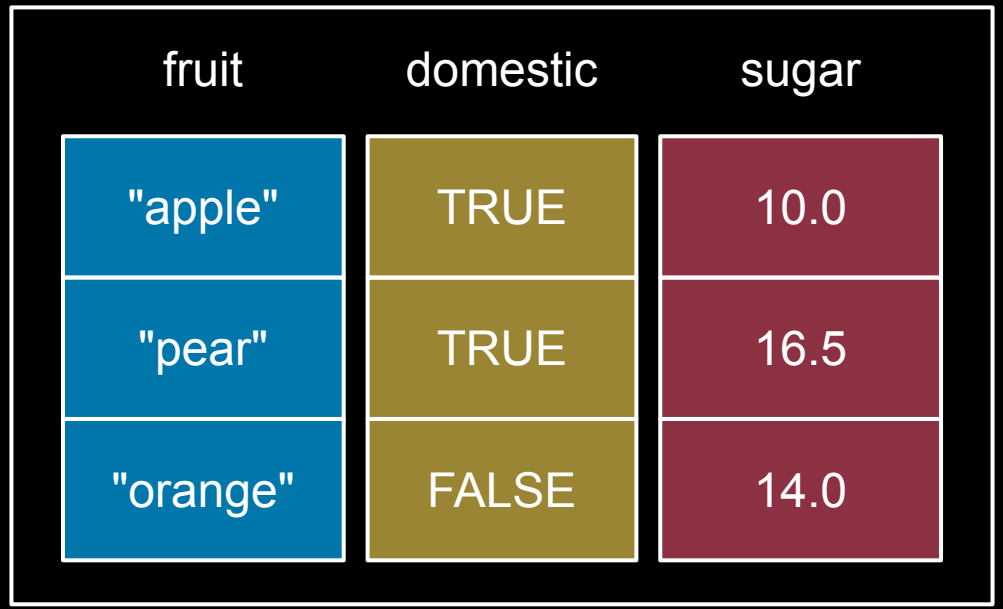
saving data frames

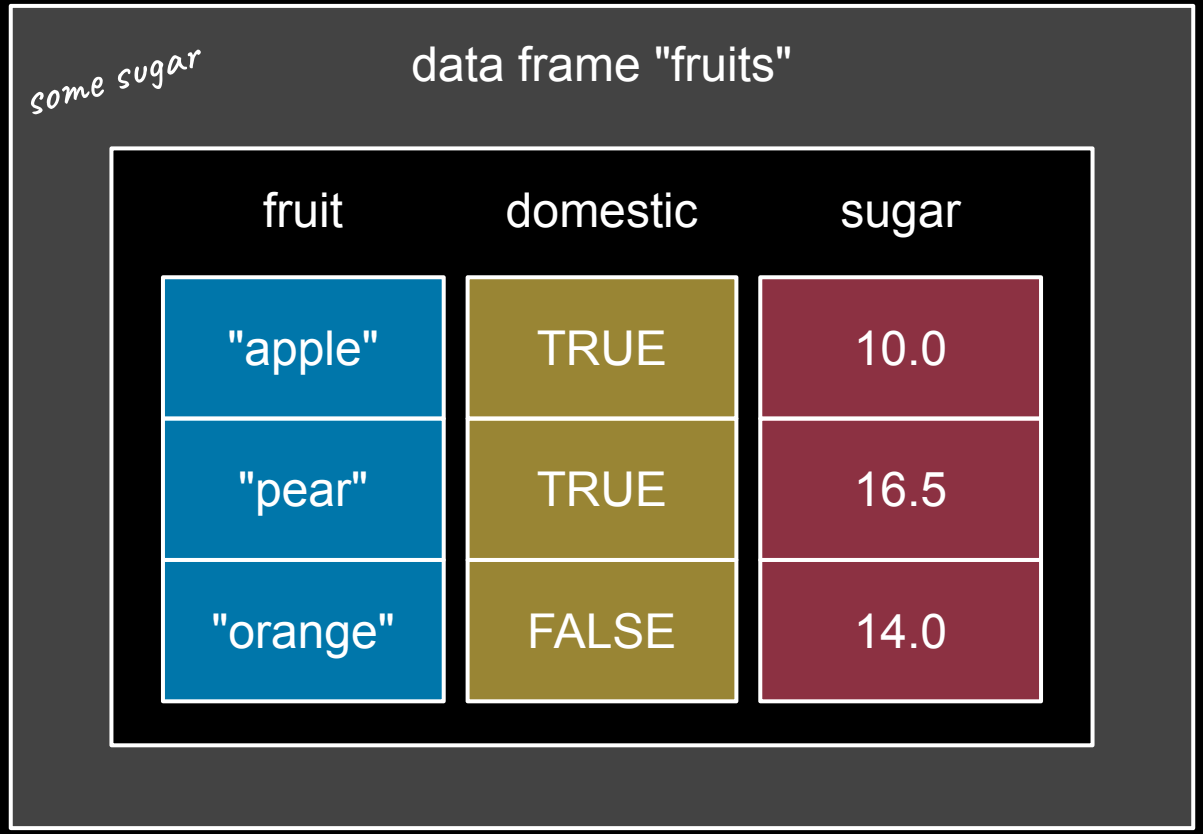
```
write.csv()
```

tibbles

{{ tibble }}

data frame "fruits"




some sugar
data frame "fruits"
fruit
domestic
sugar
"apple"
TRUE
10.0
"pear"
TRUE
16.5
"orange"
FALSE
14.0
tibble "tbl_fruits"

```
as_tibble()
```

some sugar

better printing
subsets stay tibbles
better data type guessing
support for extended data types
…