Objects & Functions

The R Bootcamp

www.therbootcamp.com

@therbootcamp

July 2018

3 Object types for data

R has 3 main data objects...

`list` - R's multi-purpose container

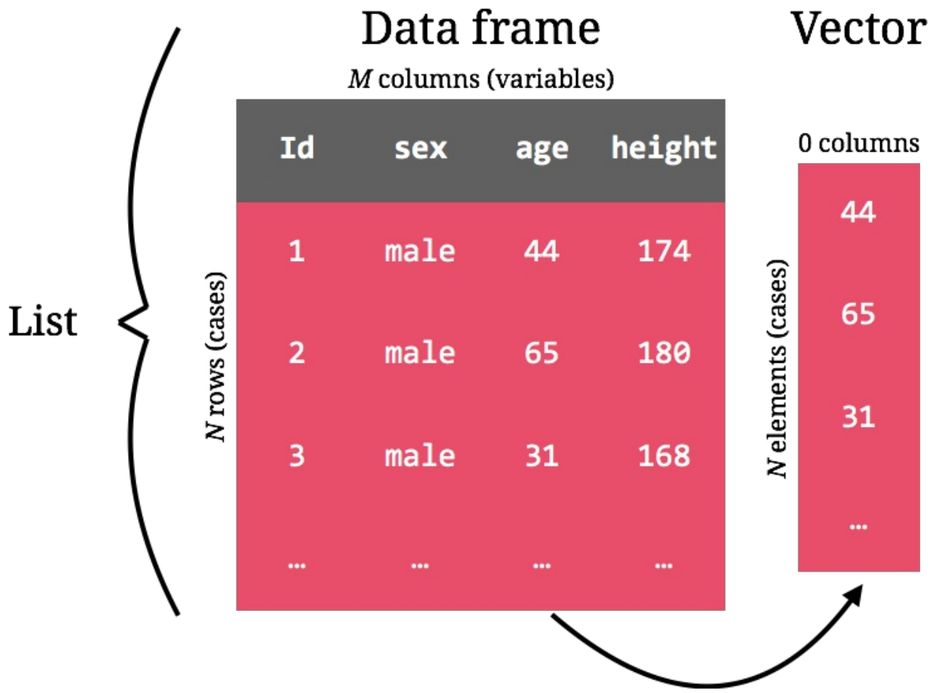
- Can carry any data, incl. lists
- Often used for function outputs

`data_frame` - R's spreadsheet

- Specific type of `list`
- Typical data format
- For multi-variable data sets

`vectors` - R's data container

- Actually carries the data
- Contain data of 1 of many types



list

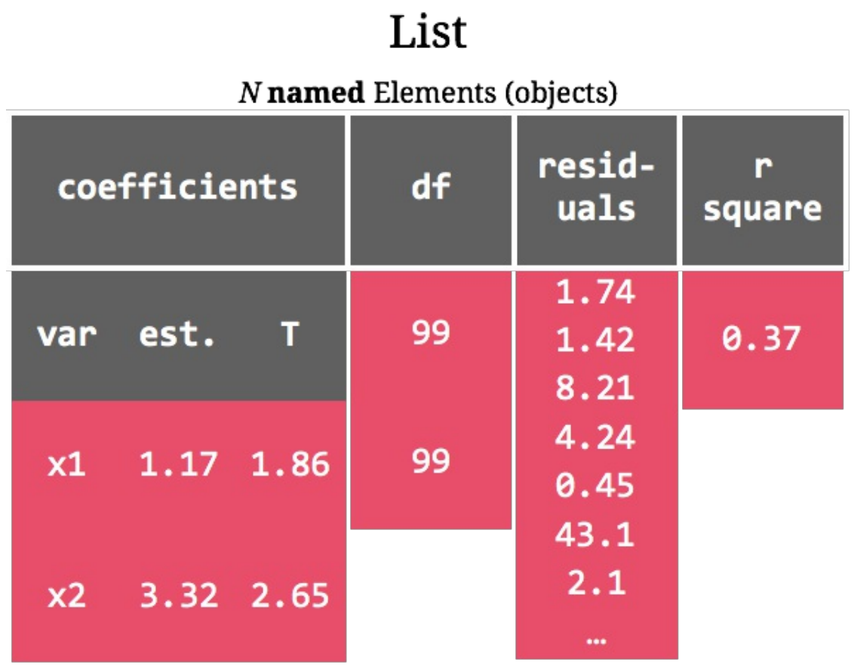
1 - Can **carry any data**, incl. `lists`, `data_frames`, `vectors`, etc.

2 - Are often used for **function outputs**

3 - Have **named elements**.

4 - Elements can be **inspect**ed via `names()` or `str()`.

5 - Elements are (typically) **select**ed by `$`.



`list`: Select element using $

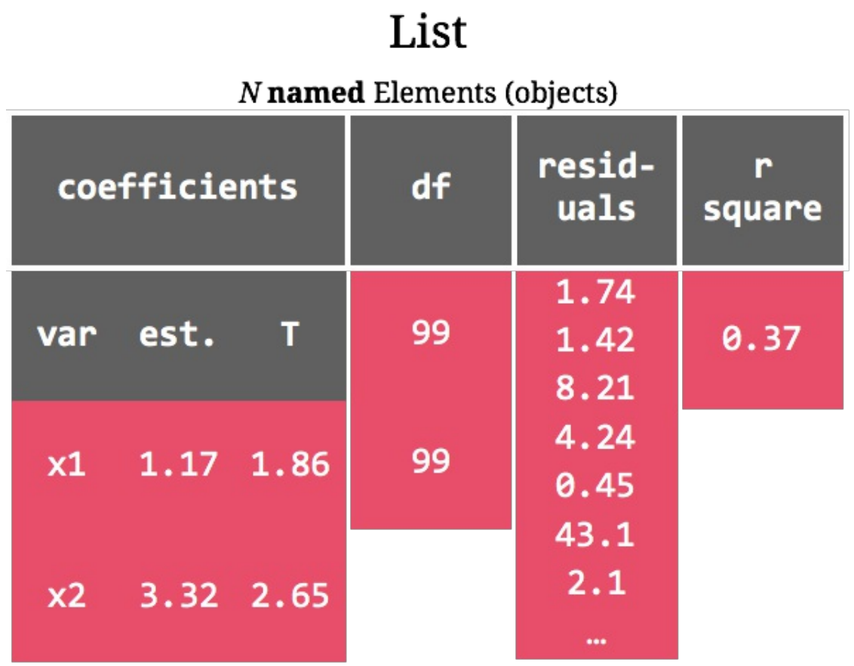
```
# regression
reg_model <- lm(height ~ sex + age,
                data = baselers)
reg_results <- summary(reg_model)

# get element names
names(reg_results)
```

```
## [1] "call"          "terms"
## [3] "residuals"     "coefficients"
## [5] "aliased"       "sigma"
## [7] "df"            "r.squared"
```

```
# select element using $
reg_results$coefficients
```

```
##                Estimate  t value
## (Intercept) 164.171266 499.5339
## sexmale      13.993699  66.4724
## age          -0.003753  -0.5819
```



data_frame

1 - Are `lists` containing `vectors` **of equal length** representing the variables.

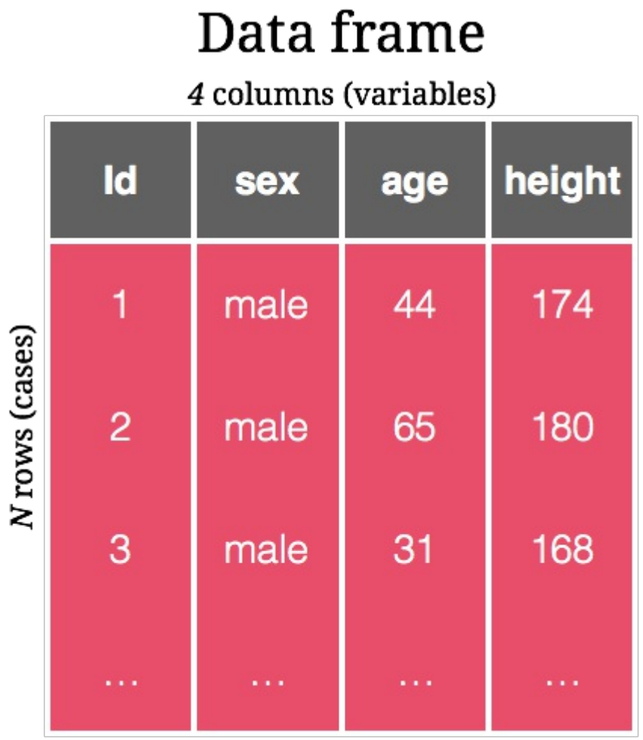
2 - Contain `vectors` of different types: `numeric`, `character`, etc.

3 - Have named elements.

4 - Elements can be **inspect**ed via `names()`, `str()`, `print()`, `View()`, or `skimr::skim()`.

5 - Elements are (typically) **select**ed by `$`.

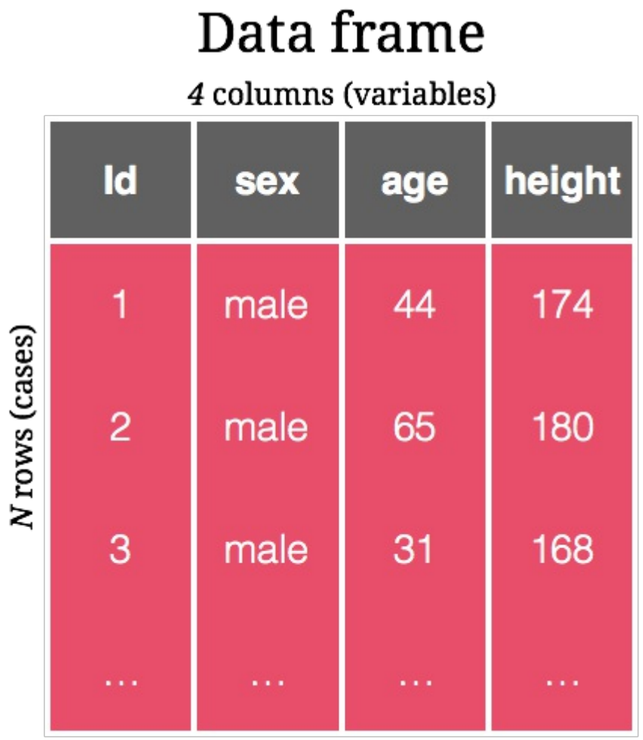
6 - Come in different flavors: `data.frame()`, `data.table()`, `tibble()`.



Inspect content

```
# inspect baselers via print
baselers
```

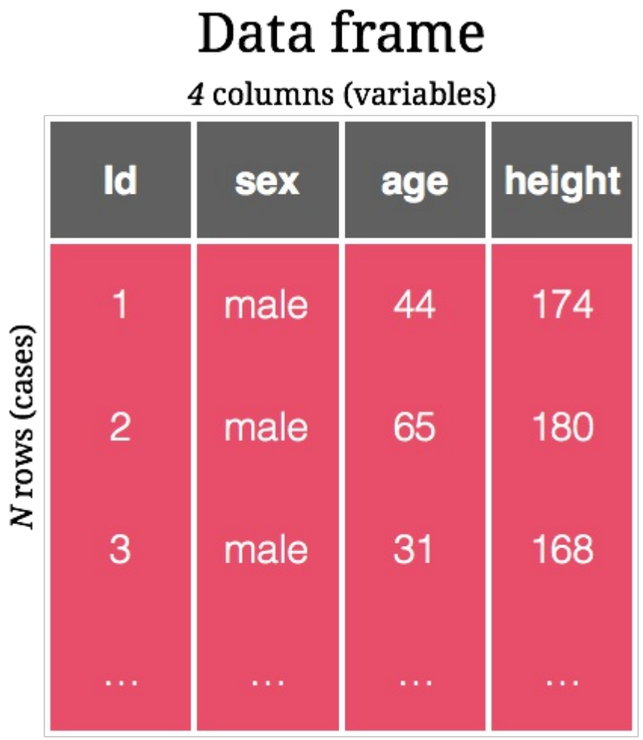
```
## # A tibble: 10,000 x 20
##       id sex      age height weight
##    <int> <chr>  <int>  <dbl>  <dbl>
##  1     1 male      44   174.  113.
##  2     2 male      65   180.   75.2
##  3     3 female    31   168.   55.5
##  4     4 male      27   209    93.8
##  5     5 male      24   177.   NA
##  6     6 male      63   187.   67.4
##  7     7 male      71   152.   83.3
##  8     8 female    41   156.   67.8
##  9     9 male      43   176.   69.3
## 10    10 female    31   166.   66.3
## # ... with 9,990 more rows, and 15 more
## #   variables
```



Inspect content

```
# inspect baselers via print
View(baselers)
```

	id	sex	age	height	weight	income
1	1	male	44	174.3	113.4	6300
2	2	male	65	180.3	75.2	10900
3	3	female	31	168.3	55.5	5100
4	4	male	27	209.0	93.8	4200
5	5	male	24	176.7	*NA*	4000
6	6	male	63	186.6	67.4	11400
7	7	male	71	151.6	83.3	12000
8	8	female	41	155.7	67.8	7600
9	9	male	43	176.1	69.3	8500
10	10	female	31	166.1	66.3	6100
11	11	female	42	157.8	51.9	8000



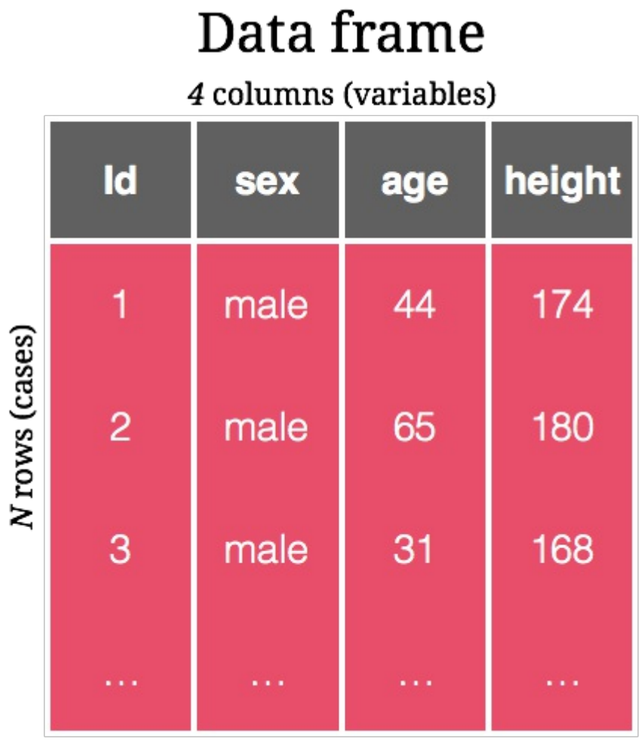
Select via $

```
# select age variable
baselers$age
```

```
##  [1] 44 65 31 27 24 63 71 41 43 31 42 31
## [13] 38 49 39 54 78 62 88 74
```

```
# select age variable
baselers$education
```

```
##  [1] "SEK_III"
##  [2] "obligatory_school"
##  [3] "SEK_III"
##  [4] "SEK_III"
##  [5] "SEK_III"
##  [6] "SEK_III"
##  [7] "SEK_III"
##  [8] "SEK_III"
##  [9] "apprenticeship"
## [10] "SEK_II"
```

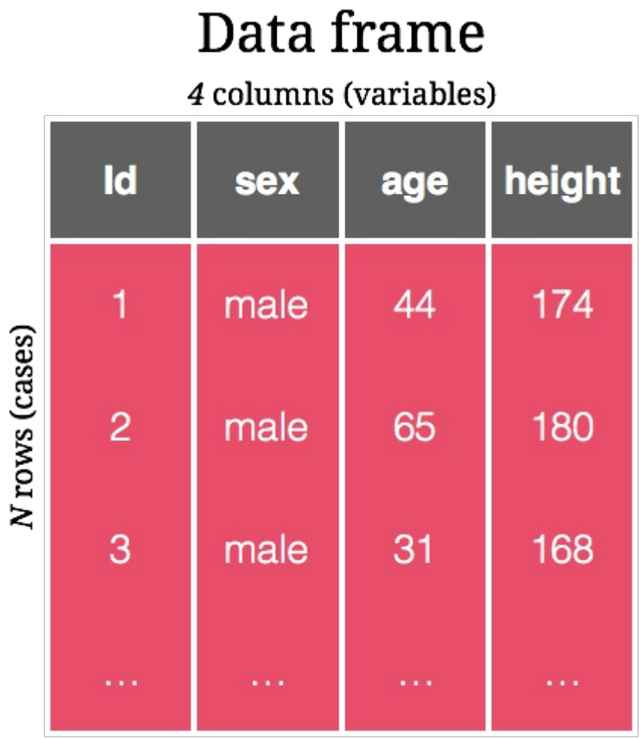


Change/Add via $

```
# compute age in months
baselers$age <- baselers$age * 2

# inspect baselers
baselers
```

```
## # A tibble: 10,000 x 20
##       id sex      age height weight
##    <int> <chr>  <dbl>  <dbl>  <dbl>
##  1     1 male      88   174.  113.
##  2     2 male     130   180.   75.2
##  3     3 female    62   168.   55.5
##  4     4 male      54   209    93.8
##  5     5 male      48   177.   NA
##  6     6 male     126   187.   67.4
##  7     7 male     142   152.   83.3
##  8     8 female    82   156.   67.8
##  9     9 male      86   176.   69.3
## 10    10 female    62   166.   66.3
## # ... with 9,990 more rows, and 15 more
## #   variables
```



Tidy data

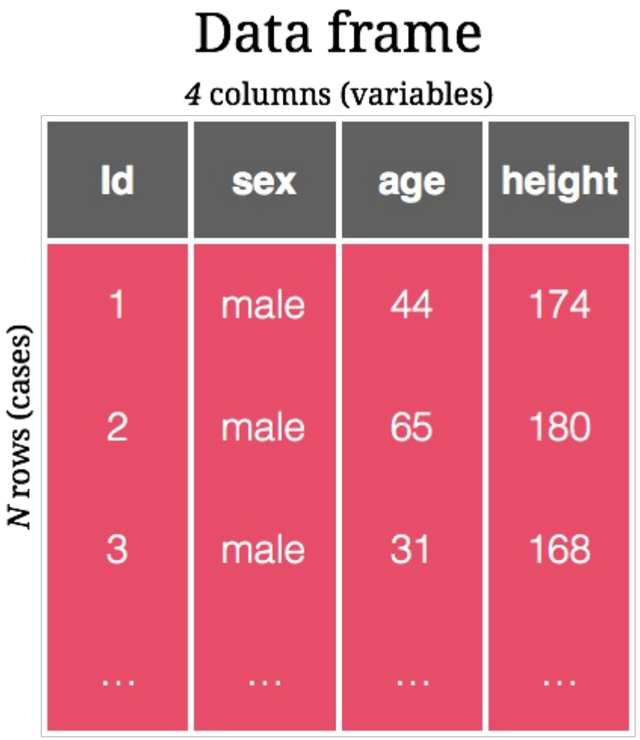
1 - Each variable you measure should be in one column.

2 - Each different observation of that variable should be in a different row.

3 - There should be one table for each "kind" of variable.

4 - If you have multiple tables, they should include a column in the table that allows them to be linked.

see **The Elements of Data Analytic Style** by Jeff Leek



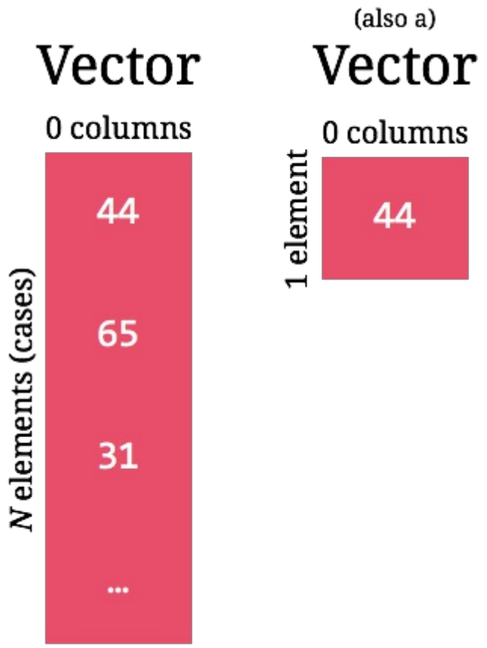
vector

1 - R's **basic and, in a way, only data container**.

2 - Can contain only a **single type of data** and missing values.

3 - Data types

`numeric` - All numbers
`character` - All characters (e.g., names)
`logical` - `TRUE` or `FALSE`
...
`NA` - missing values



Select/Change/(Add) via [ ]

```
# extract vector containing age
age <- baselers$age
age
```

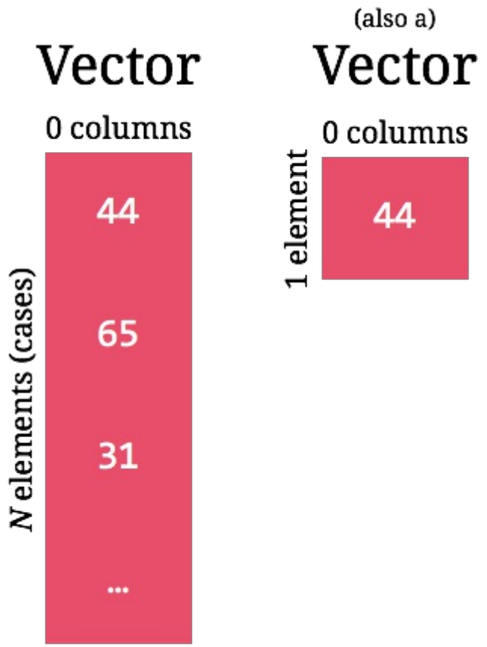
```
## [1]  88 130  62  54  48 126 142  82  86
```

```
# select value
age[2]
```

```
## [1] 130
```

```
# change value
age[2] <- 2
age
```

```
## [1]  88   2  62  54  48 126 142  82  86
```



Find more info on indexing **here**.

Data types: numeric

numeric vectors are used to store numbers and only numbers.

```
baselers$age
```

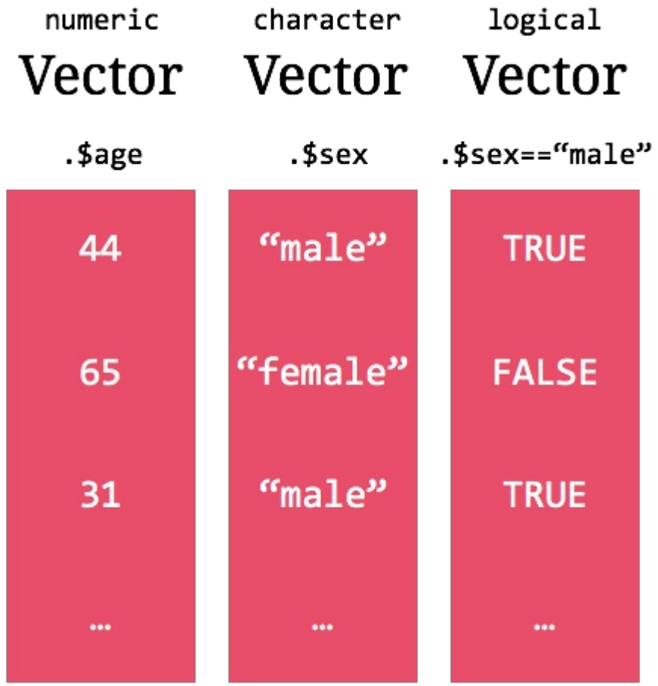
```
## [1]  88 130  62  54  48 126 142  82  86
```

```
# evaluate type
typeof(baselers$age)
```

```
## [1] "double"
```

```
is.numeric(baselers$age)
```

```
## [1] TRUE
```



Data types: `character`

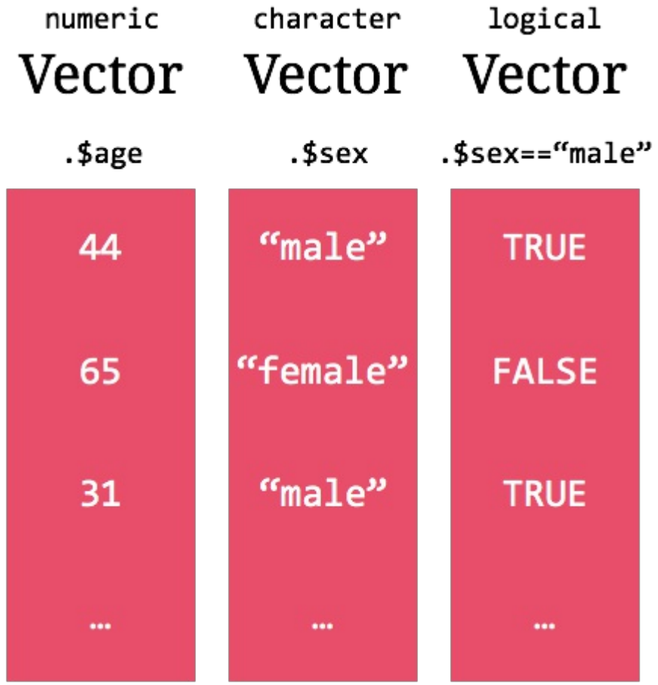
`character` vector are used to store data represented by **letters and symbols, and all other data**.

```
baselers$sex
```

```
## [1] "male"   "male"   "female" "male"
## [5] "male"   "male"   "male"   "female"
```

```
# evaluate type
as.character(baselers$age)
```

```
## [1] "88"  "130" "62"  "54"  "48"  "126"
## [7] "142" "82"  "86"
```



Data types: `logical`

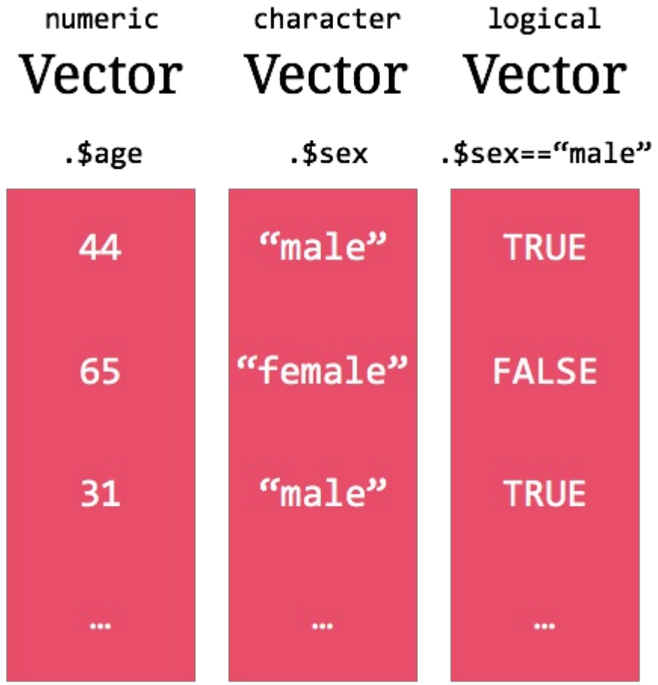
`logical` vector are used to **data** aka to select elements or rows. `logical` are typically created from other vectors via **logical comparisons**.

```
baselers$sex == "male"
```

```
## [1]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## [7]  TRUE FALSE
```

```
# evaluate type
baselers$age < 30
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [8] TRUE TRUE
```



Data types: `logical`

`logical` vector are used to **data** aka to select elements or rows. `logical` are typically created from other vectors via **logical comparisons**.

Logical operators

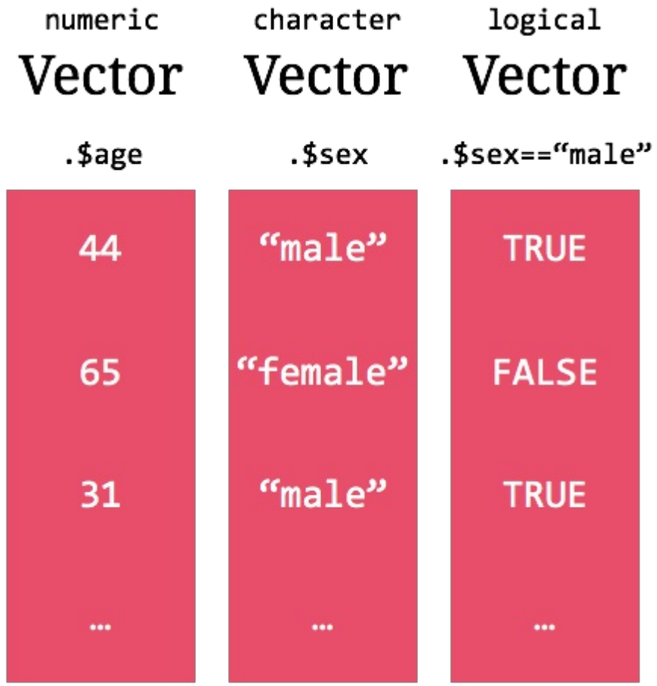
== - is equal to

<, > - smaller/greater than

≤, ≥ - smaller/greater than or equal

&, && - logical AND

|, || - logical OR

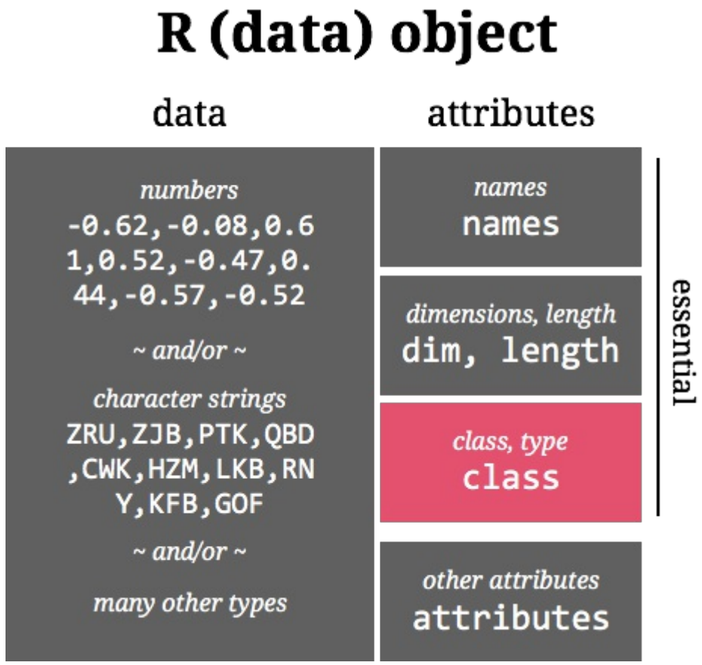


Object Classes

1 - R's objects have **content and attributes**.

2 - Attributes include always **names**, **dimensions**, and the **class** (or type) of the object.

3 - **Classes** are critical because they determine **when and how they can be used in functions**!



Functions

Functions have 3 elements:

1 - **Name**: Used to refer to the function and call (execute) it.

2 - **Arguments**: Used to provide (data) inputs and to control what the function does. Arguments with default values (e.g., `use = "everything"`) need not be specified. Arguments without default values (e.g., `x`) need be specified. **Inputs must have the appropriate class!**

3 - **Body**: The code that uses the inputs (arguments) to produce the desired output. The code of the functions body is based **copies of the inputs**, which are named according to the arguments names.

A function

```
cor(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))
```

```
na.method <- pmatch(use, c("all.obs",
"complete.obs", "pairwise.complete.obs",
        "everything", "na.or.complete"))
…
if (method == "pearson")
   .Call(C_cor, x, y, na.method, FALSE)
else if (na.method %in% c(2L, 5L)) {
   if (is.null(y))
      .Call(C_cor, Rank(na.omit(x)), NULL,
      na.method, method == “kendall")
      }
…
```

Documentation

R documentation (**help files** and **vignettes**) will become very to use once you are familiar with the basic R vocabulary.

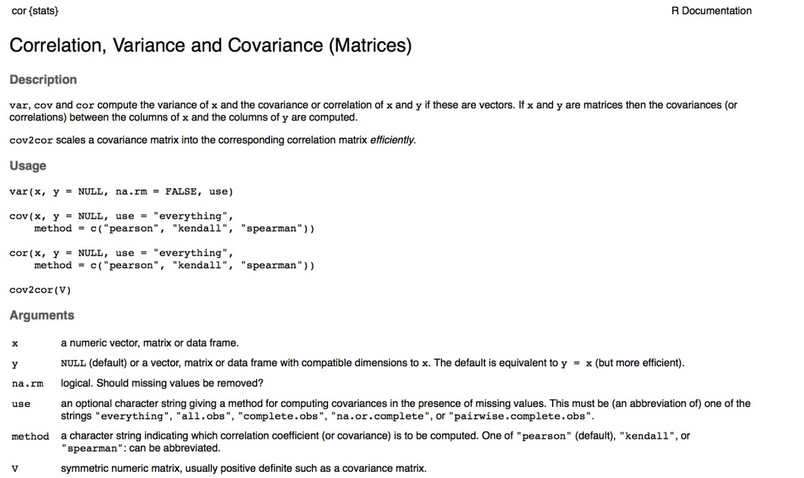
Pay attention to...

Usage - shows how to use function, its arguments and their defaults.
Arguments - describes arguments, and their `class`.
Value - describes what the function returns.
Examples - provide working R code.

```
# To access help files
?name_of_function

# search help files
??name_of_function
```

```
?cor
```



cor {stats} R Documentation

Correlation, Variance and Covariance (Matrices)

Description

var, cov and cor compute the variance of x and the covariance or correlation of x and y if these are vectors. If x and y are matrices then the covariances (or correlations) between the columns of x and the columns of y are computed.

cov2cor scales a covariance matrix into the corresponding correlation matrix *efficiently*.

Usage

```
var(x, y = NULL, na.rm = FALSE, use)

cov(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))

cor(x, y = NULL, use = "everything",
    method = c("pearson", "kendall", "spearman"))

cov2cor(V)
```

Arguments

x	a numeric vector, matrix or data frame.
y	NULL (default) or a vector, matrix or data frame with compatible dimensions to x. The default is equivalent to y = x (but more efficient).
na.rm	logical. Should missing values be removed?
use	an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs".
method	a character string indicating which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman": can be abbreviated.
V	symmetric numeric matrix, usually positive definite such as a covariance matrix.

Practical

Link to practical