Assigning a variable

alt -

example
a <- ‘apple’

Running a line of code

cmd return

ctrl return

Pipe

cmd shift M

ctrl shift M

example
diamonds %>% str()

practice
ds **alt -** diamonds **cmd shift M** str() **cmd return**

ds <- diamonds %>% str()

Types

Converting between common data types in R. Can always go from a higher value in the table to a lower value.

`as.logical`	`TRUE, FALSE, TRUE`	Boolean values (TRUE or FALSE).
`as.numeric`	`1, 0, 1`	Integers or floating point numbers.
`as.character`	`'1', '0', '1'`	Character strings. Generally preferred to factors.
`as.factor`	`'1', '0', '1', levels: '1', '0'`	Character strings with preset levels. Needed for some statistical models.

Maths Functions

`log(x)`	Natural log.	`sum(x)`	Sum.
`exp(x)`	Exponential.	`mean(x)`	Mean.
`max(x)`	Largest element.	`median(x)`	Median.
`min(x)`	Smallest element.	`quantile(x)`	Percentage quantiles.
`round(x, n)`	Round to n decimal places.	`rank(x)`	Rank of elements.
`sig.fig(x, n)`	Round to n significant figures.	`var(x)`	The variance.
`cor(x, y)`	Correlation.	`sd(x)`	The standard deviation.

Variable Assignment

```
> a <- 'apple'
> a
[1] 'apple'
```

John	Ringo	Paul	George
1940	1940	1942	1943

names(beatles)	John	Ringo	Paul	George	**name**
beatles	1940	1940	1942	1943	**value**
	[1]	**[2]**	**[3]**	**[4]**	**index**

Selecting Vector Elements

By Position

`x[4]`	The fourth element.
`x[-4]`	All but the fourth.
`x[2:4]`	Elements two to four.
`x[-(2:4)]`	All elements except two to four.
`x[c(1, 5)]`	Elements one and five.

By Value

`x[x == 10]`	Elements which are equal to 10.
`x[x < 0]`	All elements less than zero.
`x[x %in% c(1, 2, 5)]`	Elements in the set 1, 2, 5.

Named Vectors

`x['apple']`	Element with name 'apple'.

Selecting Vector Elements

By Position

`x[4]`	The fourth element.
`x[-4]`	All but the fourth.
`x[2:4]`	Elements two to four.
`x[-(2:4)]`	All elements except two to four.
`x[c(1, 5)]`	Elements one and five.

By Value

`x[x == 10]`	Elements which are equal to 10.
`x[x < 0]`	All elements less than zero.
`x[x %in% c(1, 2, 5)]`	Elements in the set 1, 2, 5.

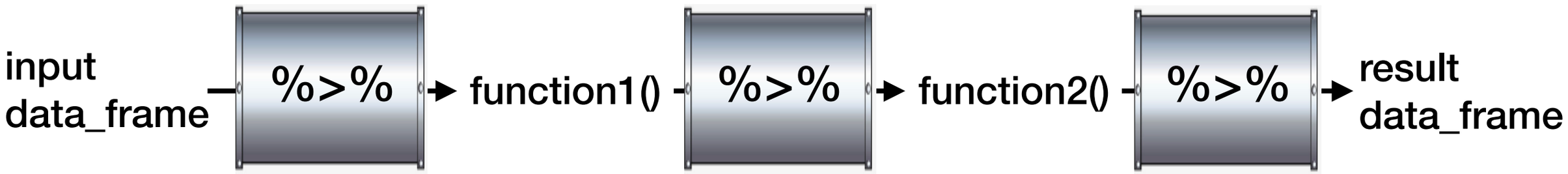
Named Vectors

`x['apple']`	Element with name 'apple'.

Conditions

`a == b`	Are equal	`a > b`	Greater than
`a != b`	Not equal	`a < b`	Less than
`a >= b`	Greater than or equal to	`is.na(a)`	Is missing
`a <= b`	Less than or equal to	`is.null(a)`	Is null

The pipe %>%



cmd shift M

ctrl shift M

dplyr

Five key dplyr functions allow you to solve the vast majority of data manipulation challenges

- Pick rows by value `filter()`
- Pick columns by name `select()`
- Reorder rows `arrange()`
- Create new columns by transforming existing columns `mutate()`
- Collapse many values down to a single summary `summarise()`

Format
data_frame %>% dplyr::function()

Output
another data_frame

Pipe multiple simple steps together to achieve a complex result

my_df %>% **`filter()`**

my_df %>% **`select()`**

my_df %>% **`mutate()`**

my_df %>% **`summarize()`**