FUNCTION CHAINING

Using the %>% operator

THE PROBLEM

Start with a vector of numbers

Get the absolute value of each element

Multiply each element by three

Take the square root of all elements

Sum all elements

LINE-BY-LINE SOLUTION

```r
# 1] A vector of numbers
vct_numbers <- -10:10


# 2] Get the absolute value
step_2 <- abs(vct_numbers)


# 3] Multiply each number by 3
step_3 <- step_2 * 3


# 4] Square root
step_4 <- sqrt(step_3)


# 5] sum all elements
final_step <- sum(step_4)
```

*The final result is **77.8324***

LINE-BY-LINE SOLUTION

Advantages

- Easy to read — *top to bottom*
- Easy to debug — *Can observe intermediate results*

Disadvantages

- Verbose — *lots of typing*
- Many variables — *clogs up the workspace*

NESTING SOLUTION

```
# Put everything in brackets

sum(sqrt((abs(-10:10)) * 3))
```

*The final result is **77.8324***

NESTING SOLUTION

Advantages

- Terse — *Not much typing*
- No additional variables

Disadvantages

- Difficult to read — *People don't read inside out*

CHAINING SOLUTION

```r
-10:10 %>% abs() %>% '*'(3) %>% sqrt() %>% sum()
```

*The final result is **77.8324***

```r
# note that ‘*’(3, 2) is the same as 3 * 2

# left hand side is piped to the right hand side

# example: c(-2, -1, 1, 2) %>% abs() ==> 2, 1, 1, 2
```

CHAINING SOLUTION

Advantages

- Terse — *Not much typing*
- No additional variables
- Can read sequentially — left to right

Disadvantages

- Debugging — *Can highlight steps 1:N and run*

MULTI ARGUMENT FUNCTIONS

```r
# define a simple function
fn_power <- function(number, power) number ^ power

# following results in 1 4 9 16
fn_power(1:4, 2)

# following also results in 1 4 9 16
1:4 %>% fn_power(2)
```

The left hand side of the %>% is piped to the first argument of the proceeding function

PIPE TO DIFFERENT ARGUMENTS

```r
# What if our function was defined in reverse order…?
fn_power_rev <- function(power, number) number ^ power
```

```r
# We can use the '.' operator as a placeholder
# following also results in 1 4 9 16
1:4 %>% fn_power(2, .)
```

PIPE TO DIFFERENT ARGUMENTS

```r
# Another example using  “gsub()”

# replaces “morning” with “afternoon”
# results in “good afternoon”
gsub("morning",  "afternoon", "good morning")

# using the %>% operator
# use the “.” as the argument is in the 3rd position
"good morning" %>% gsub("morning",  "afternoon", .)
```

HOW TO LOAD

- Explicitly using *library(magrittr)*
- Implicitly using *library(dplyr)*