
```
1 # Define function to calculate average value in a vector
2
3 average <- function(x) {
4   sum(x) / length(x)
5 }
```

```
1 # Handle non-numeric input
2
3 average <- function(x) {
4   if (!is.numeric(x)) {
5     return(NA)
6   }
7   sum(x) / length(x)
8 }
```

```
1 # Message about returning NA
2
3 average <- function(x) {
4   if (!is.numeric(x)) {
5     message("`x` must be a numeric vector. Returning NA instead.")
6     return(NA)
7   }
8   sum(x) / length(x)
9 }
```

```
1 # Warn about returning NA
2
3 average <- function(x) {
4   if (!is.numeric(x)) {
5     warning("`x` must be a numeric vector. Returning NA instead.")
6     return(NA)
7   }
8   sum(x) / length(x)
9 }
```

```
1 # Stop instead of warn
2
3 average <- function(x) {
4   if (!is.numeric(x)) {
5     stop("`x` must be a numeric vector.")
6   }
7   sum(x) / length(x)
8 }
```

```
1 # Handle NA values
2
3 average <- function(x) {
4   if (!is.numeric(x)) {
5     stop("`x` must be a numeric vector.")
6   }
7   if (any(is.na(x))) {
8     warning("`x` contains one or more NA values.")
9     return(NA)
10  }
11  sum(x) / length(x)
12 }
```

```
1 # Write test function
2
3 source("average6.R")
4
5 test_average <- function() {
6   if (average(c(1, 2, 3)) == 2) {
7     cat("`average` passed test :)\n")
8   } else {
9     cat("`average` failed test :(\\n")
10  }
11 }
12
13 test_average()
```

```
1 # Add test cases
2
3 source("average6.R")
4
5 test_average <- function() {
6   if (average(c(1, 2, 3)) == 2) {
7     cat("`average` passed test :)\n")
8   } else {
9     cat("`average` failed test :(\n")
10  }
11
12  if (average(c(-1, -2, -3)) == -2) {
13    cat("`average` passed test :)\n")
14  } else {
15    cat("`average` failed test :(\n")
16  }
17
18  if (average(c(-1, 0, 1)) == 0) {
19    cat("`average` passed test :)\n")
20  } else {
21    cat("`average` failed test :(\n")
22  }
23 }
24
25 test_average()
```



```
1 # Introduce test_that and add representative test cases to catch corner cases
2
3 source("average6.R")
4
5 test_that("`average` calculates mean", {
6   expect_equal(average(c(1, 2, 3)), 2)
7   expect_equal(average(c(-1, -2, -3)), -2)
8   expect_equal(average(c(-1, 0, 1)), 0)
9   expect_equal(average(c(-2, -1, 1, 2)), 0)
10 })
```

```
1 # Test warning about NA values
2
3 source("average6.R")
4
5 test_that("`average` calculates mean", {
6   expect_equal(average(c(1, 2, 3)), 2)
7   expect_equal(average(c(-1, -2, -3)), -2)
8   expect_equal(average(c(-1, 0, 1)), 0)
9   expect_equal(average(c(-2, -1, 1, 2)), 0)
10 })
11
12 test_that("`average` warns about NAs in input", {
13   expect_warning(average(c(1, NA, 3)))
14   expect_warning(average(c(NA, NA, NA)))
15 })
```

```
1 # Fix ordering of error handling
2
3 average <- function(x) {
4   if (any(is.na(x))) {
5     warning("`x` contains one or more NA values.")
6     return(NA)
7   }
8   if (!is.numeric(x)) {
9     stop("`x` must be a numeric vector.")
10  }
11  sum(x) / length(x)
12 }
```

```
1 # Test NA return values
2
3 source("average7.R")
4
5 test_that("`average` calculates mean", {
6   expect_equal(average(c(1, 2, 3)), 2)
7   expect_equal(average(c(-1, -2, -3)), -2)
8   expect_equal(average(c(-1, 0, 1)), 0)
9   expect_equal(average(c(-2, -1, 1, 2)), 0)
10 })
11
12 test_that("`average` returns NA with NAs in input", {
13   expect_equal(suppressWarnings(average(c(1, NA, 3))), NA)
14   expect_equal(suppressWarnings(average(c(NA, NA, NA))), NA)
15 })
16
17 test_that("`average` warns about NAs in input", {
18   expect_warning(average(c(1, NA, 3)))
19   expect_warning(average(c(NA, NA, NA)))
20 })
```

```
1 # Test stop if argument is non-numeric
2
3 source("average7.R")
4
5 test_that("`average` calculates mean", {
6   expect_equal(average(c(1, 2, 3)), 2)
7   expect_equal(average(c(-1, -2, -3)), -2)
8   expect_equal(average(c(-1, 0, 1)), 0)
9   expect_equal(average(c(-2, -1, 1, 2)), 0)
10 })
11
12 test_that("`average` returns NA with NAs in input", {
13   expect_equal(suppressWarnings(average(c(1, NA, 3))), NA)
14   expect_equal(suppressWarnings(average(c(NA, NA, NA))), NA)
15 })
16
17 test_that("`average` warns about NAs in input", {
18   expect_warning(average(c(1, NA, 3)))
19   expect_warning(average(c(NA, NA, NA)))
20 })
21
22 test_that("`average` stops if `x` is non-numeric", {
23   expect_error(average(c("quack!")))
24   expect_error(average(c("1", "2", "3")))
25 })
```

```
1 # Demonstrates floating-point imprecision
2
3 print(0.3)
4 print(0.3, digits = 17)
```

```
1 # Test doubles
2
3 source("average7.R")
4
5 test_that("`average` calculates mean", {
6   expect_equal(average(c(1, 2, 3)), 2)
7   expect_equal(average(c(-1, -2, -3)), -2)
8   expect_equal(average(c(-1, 0, 1)), 0)
9   expect_equal(average(c(-2, -1, 1, 2)), 0)
10  expect_equal(average(c(0.1, 0.5)), 0.3)
11 })
12
13 test_that("`average` returns NA with NAs in input", {
14   expect_equal(suppressWarnings(average(c(1, NA, 3))), NA)
15   expect_equal(suppressWarnings(average(c(NA, NA, NA))), NA)
16 })
17
18 test_that("`average` warns about NAs in input", {
19   expect_warning(average(c(1, NA, 3)))
20   expect_warning(average(c(NA, NA, NA)))
21 })
22
23 test_that("`average` stops if `x` is non-numeric", {
24   expect_error(average(c("quack!")))
25   expect_error(average(c("1", "2", "3")))
26 })
```

```
1 # Test greet
2
3 source("greet1.R")
4
5 test_that("`greet` says hello to a user", {
6   expect_equal(greet("Carter"), "hello, Carter")
7 })
```

```
1 # Greet a user
2
3 greet <- function(to) {
4   return(paste("hello,", to))
5 }
```

```
1 # Describe greet
2
3 source("greet1.R")
4
5 describe("greet()", {
6   it("can say hello to a user", {
7     name <- "Carter"
8     expect_equal(greet(name), "hello, Carter")
9   })
10 })
```

```
1 # Describe greet
2
3 source("greet2.R")
4
5 describe("greet()", {
6   it("can say hello to a user", {
7     name <- "Carter"
8     expect_equal(greet(name), "hello, Carter")
9   })
10  it("can say hello to the world", {
11    expect_equal(greet(), "hello, world")
12  })
13 })
```

```
1 # Provides default argument value
2
3 greet <- function(to = "world") {
4   return(paste("hello,", to))
5 }
```