In R , how to get count of a value?

In R there are a few super common ways to **count values**, depending on what you mean. Here are the most useful patterns 👇

1️⃣ Count how many times each value appears

r Copy code

```r
x <- c("a", "b", "a", "c", "b", "a")

table(x)
```

Output

r Copy code

```r
a b c
3 2 1
```

2️⃣ Count a specific value

r Copy code

```r
sum(x == "a")
```

➡️ Counts how many `"a"` values are in `x`

3️⃣ Count values using dplyr (very common in data work)

r Copy code

```
library(dplyr)

df %>%
  count(column_name)
```

Example:

r Copy code

```r
df <- data.frame(type = c("A", "B", "A", "C", "A"))

df %>% count(type)
```

4️⃣ Count values with group_by

r Copy code

```r
df %>%
  group_by(type) %>%
  summarise(n = n())
```

5️⃣ Count non-NA values

r Copy code

```r
sum(!is.na(x))
```

6️⃣ Count NA values

r Copy code

```r
sum(is.na(x))
```

7️⃣ Quick base-R alternative

```r
length(which(x == "a"))
```

If you want, tell me:

- vector or data frame?
- count **one value** or **all values**?
- base R or `dplyr` ?

I'll tailor it exactly to your case 👍

How to understand this line "summarise(n = n())" , I dont see, it declare a var "n" ?

Great question — this line *looks* mysterious at first, but once you see the pattern it clicks fast 👍

```r
summarise(n = n())
```

Let's break it down **piece by piece**.

1️⃣ What `summarise()` does

`summarise()` **creates a new data frame** where:

- each row represents a group (from `group_by()`), and
- each column is a summary you define.

So inside `summarise()`, you are **defining new columns**.

2️⃣ Is `n` a variable?

Yes — but **not beforehand**.

```r
n = n()
```

means:

> "Create a new column called `n`, whose value is the result of `n()`"

So:

- **left** `n` → column name (you are declaring it here)
- **right** `n()` → a function

Think of it like:

```r
new_column_name = calculation()
```

3️⃣ What is `n()` ?

`n()` is a **dplyr helper function** that means:

> “How many rows are in the current group?”

Example:

r

Copy code

```
df <- data.frame(
  type = c("A", "A", "B", "C", "A")
)

df %>%
  group_by(type) %>%
  summarise(n = n())
```

How dplyr sees it:

- Group `"A"` → 3 rows → `n()` = 3
- Group `"B"` → 1 row → `n()` = 1
- Group `"C"` → 1 row → `n()` = 1

Output

r

Copy code

```
# A tibble: 3 × 2
  type      n
  <chr> <int>
1 A         3
2 B         1
3 C         1
```

4️⃣ Why use the same name twice?

This is a **very common R pattern**:

r

Copy code

```
column_name = function()
```

Other examples:

r

Copy code

```r
summarise(mean_age = mean(age))
summarise(total = sum(sales))
summarise(max_score = max(score))
```

So `n = n()` just happens to reuse the same word — but they mean **different things**:

- first `n` → column name
- second `n()` → counting function

5️⃣ Shortcut: `count()`

Because this pattern is so common, dplyr gives you:

r

Copy code

```r
df %>% count(type)
```

Which is exactly the same as:

r

Copy code

```r
df %>%
  group_by(type) %>%
  summarise(n = n())
```

Mental model (remember this)

> **Inside `summarise()` you are *creating columns*, not referencing old variables.**

If you want, I can also explain:

- how `summarise()` differs from `mutate()`

- why `n()` only works inside dplyr verbs
- or how this works under the hood (tidy evaluation)

Just say the word 👌