5 How to count with count, add_count, and add_tally

What will this tutorial cover?

In this tutorial you will learn how to count values with count(), add_count(), and add_tally(). Also, you will find out how to count with continuous variables.

Who do I have to thank?

For this post, I have to thank David Robinson, from whom I learned about the decade trick. Thanks also go to Olivier Gimenez who wrote a really nice article explaining this trick.

Counting is one of the most common tasks you do when working with data. Counting may sound simple, but it can get complicated quickly. Consider these examples:

- Sometimes we want to count with continuous variables. Suppose you have a year variable in your data frame that is of data type integer (e.g. 1982, 1945, 1990). You want to know the number of people for each decade. To do this, you must first convert your year variable to decades before you start counting.
- Often you want to count things per group (for example, the number of players on a particular sports team) and add the counts per group as a new variable to your data frame. You could use joins to do this, but could you do it with less code and more efficiently?
- Counting can also mean that you calculate the sum of a variable within groups (for example, the number of goals scored by a particular team during a season). Normally you would use group_by and summarize for this calculation. But then you should not forget to ungroup. Could you do this without the group_by function and with less code?

If you feel that you can't complete these tasks in a snap, read on. I'll introduce you to four tricks and functions that will help you accomplish these tasks effortlessly.

- The first trick lets you count with continuous variables (counting decades, for example).
- With the second trick, you can calculate the sum of a variable within groups without using `group_by`

- The third trick allows you to add the count of a variable as a new variable to your data frame.
- The fourth trick allows you to add a new variable to your data frame that represents the sum of a given variable.

5.1 How to count with continuous variables

Let's start with the first trick. The Starwars dataset from dplyr represents all Starwars characters in a data frame. The data frame has a variable `birth_year`. Suppose you want to know how many of these characters were born in a given decade, not a given year. To calculate decades from years, you need to know how integer division works. Example: I was born in 1986. I might ask myself: How many times does the number ten fit into the number 1986? I could divide 1986 by ten and get 198.6, but that's not the question I'm asking. I want to know how many times 10 fits into the number 1986. For this we need integer division. In R, integer division can be done with the `%/%` operator. If I apply this operator to my birth year, I get the number 198 (`1986 %/% 10`). To convert this number to my decade, I need to multiply this number by 10:

```
10 * (1986 %/% 10)
```

```
[1] 1980
```

> **Note**
>
> Kudos go to David Robinson who as far as I know came up with this trick and Olivier Gimenez, who wrote an excellent blog describing it.

Now that we know how to perform integer divisions and calculate decades from them, we can combine them with the counting function. Suppose we want to calculate how many of the Starwars characters were born in a given decade. This is how we would do it:

```
starwars %>%
  count(decade = 10 * (birth_year %/% 10),
        name = "characters_per_decade") %>%
  glimpse()
```

```
Rows: 16
Columns: 2
$ decade               <dbl> 0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110,~
```

```
$ characters_per_decade <int> 1, 3, 4, 4, 9, 6, 4, 2, 2, 3, 1, 1, 1, 1, 1, 44
```

The real trick is that you create a new variable inside the count function. Up to now, you have probably put an existing variable into the count function. In this example, we have created a new variable `characters_per_decade` that is calculated from the variable `year_of_birth`. We also used the `name` argument to give the count column a more descriptive name.

Now that we know how we can create new variables inside `count`, we can think of new ways to create discrete variables. ggplot2, for example, has a function that creates bins from continuous variables. Suppose we want to create bins in the range of 10 for the variable `birth_year`:

```
starwars %>%
  count(height_intervals = cut_width(height, 10))
```

```
# A tibble: 18 x 2
   height_intervals     n
   <fct>            <int>
 1 [65,75]              1
 2 (75,85]              1
 3 (85,95]              2
 4 (95,105]             3
 5 (105,115]            1
 6 (115,125]            1
 7 (135,145]            1
 8 (145,155]            2
 9 (155,165]            7
10 (165,175]           14
11 (175,185]           20
12 (185,195]           12
13 (195,205]            7
14 (205,215]            3
15 (215,225]            2
16 (225,235]            3
17 (255,265]            1
18 <NA>                 6
```

You can see that the bins each have a range of 10. Also, the bins are surrounded by square brackets and parentheses. A square bracket means that a number is included in the bin, a parenthesis means that the number is not included in the bin In our second example, this would mean that the year 75 is included, but not the year 85.

5.2 How to calculate the sum of a variable based on groups without using group_by.

The second trick took me some time to understand. Intuitively, one would think that the `count` function counts the values of discrete variables: The number of players on a team, the number of cars, etc. However, count can also be used to calculate the sum of a variable for a particular group or groups. Let's first look at how this would be done without the trick. Let's say we want to calculate the sum of unemployed people in the US per year (I wouldn't trust these numbers here, this is just an example of using this technique). Using `group_by` and `summarise` you would do the following:

```
economics %>%
  mutate(
    year = format(date, "%Y")
  ) %>%
  group_by(year) %>%
  summarise(sum_unemploy = sum(unemploy, na.rm = TRUE))
```

```
# A tibble: 49 x 2
   year  sum_unemploy
   <chr>        <dbl>
 1 1967         18074
 2 1968         33569
 3 1969         33962
 4 1970         49528
 5 1971         60260
 6 1972         58510
 7 1973         52312
 8 1974         62080
 9 1975         95275
10 1976         88778
# ... with 39 more rows
```

To achieve the same result with count, you need to know the argument `wt`. wt stands for weighted counts. While `count` calculates the frequency of values within a group without specifying the `wt` argument (`n = n()`), `wt` calculates the sum of a continuous variable for certain groups (`n = sum(<VARIABLE>)`):

```
economics %>%
  count(year = format(date, "%Y"), wt = unemploy,
```

```
          name = "sum_unemploy")
```

```
# A tibble: 49 x 2
   year  sum_unemploy
   <chr>        <dbl>
 1 1967         18074
 2 1968         33569
 3 1969         33962
 4 1970         49528
 5 1971         60260
 6 1972         58510
 7 1973         52312
 8 1974         62080
 9 1975         95275
10 1976         88778
# ... with 39 more rows
```

Again, you can see that we created a new variable on the fly. Also, we used the `wt` argument and set it to the `unemploy` variable. This technique has its advantages and disadvantages. On the positive side, we only need three lines of code instead of six. On the downside, the code is less explicit, and without knowing the inner workings of count, it's hard to tell that the function is calculating sums. However, it could be a new option in your toolbox to calculate sums.

5.3 How to add counts as a variable to your data frame

In the previous examples, you saw that the `count` creates a new data frame with the grouping variable and the frequency or sum variable. This is not always what you want. Sometimes you want to add counts to your existing data frame. Let's take the mpg data frame. The data frame contains statistics about cars from different manufacturers. Now suppose you want to count how many cars there are per manufacturer, and add those numbers to the mpg data frame. This can be done with `add_count()`:

```
mpg %>%
  add_count(manufacturer, name = "number_of_cars_by_manufacturer") %>%
  select(manufacturer, model, number_of_cars_by_manufacturer) %>%
  glimpse()
```

```
Rows: 234
Columns: 3
$ manufacturer                   <chr> "audi", "audi", "audi", "audi", "audi",~
$ model                          <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4~
$ number_of_cars_by_manufacturer <int> 18, 18, 18, 18, 18, 18, 18, 18, 18, 18,~
```

As you can see, there are 18 cars from Audi. Why is this useful? Because you keep all the other columns of your data frame. For example, you can use the counts to calculate proportions (number of cars per manufacturer / total number of cars). Also, you avoid using joins to combine two data frames (the counts and the original dataset). Of course, add_count also works with weighted counts.

5.4 How to add a new variable to your data frame that contains the sum of a specific variable

add_tally() does something similar than add_count(). The only difference is that add_tally calculates the sum of a given variable instead of a count. For example, we could add a new variable to mpg that shows the sum of the variables per model.

```
mpg %>%
  group_by(model) %>%
  add_tally(wt = displ, name = "sum_display_per_model") %>%
  select(manufacturer, model, sum_display_per_model) %>%
  glimpse()
```

```
Rows: 234
Columns: 3
Groups: model [38]
$ manufacturer          <chr> "audi", "audi", "audi", "audi", "audi", "audi", ~
$ model                 <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 qu~
$ sum_display_per_model <dbl> 16.3, 16.3, 16.3, 16.3, 16.3, 16.3, 16.3, 19.4, ~
```

Again, you can see that we calculated the sum by providing the argument `wt` with a continuous variable. The result is a new variable called `sum_display_per_model`. You could get the same result with `add_count` and using `displ` for the argument wt. Note that `add_tally` has no argument for grouping the data. You must accomplish this with group_by.

Summary

Here is what you can take from this tutorial.

- With the integer division trick you convert years to decades.
- With add_count you can add a count variable to an existing data frame
- With the `wt` argument in `count`, `add_count` and `add_tally` you can calculate the sum of a variable (per group if you want to)