I very often use Hash[array.group_by{|x|x}.map{|x,y|[x,y.size]}}.

Would be nice with to have a method called count_by:

```ruby
array = ['aa', 'aA', 'bb', 'cc']
p array.count_by(&:downcase) #=> {'aa'=>2,'bb'=>1,'cc'=>1}
```

## Associated revisions

**Revision 673dc51c - 02/07/2019 08:14 AM - nobu (Nobuyoshi Nakada)**

enum.c: Enumerable#tally

- enum.c (enum_tally): new methods Enumerable#tally, which group and count elements of the collection. [Feature #11076]

**Revision 67020 - 02/07/2019 08:14 AM - nobu (Nobuyoshi Nakada)**

enum.c: Enumerable#tally

- enum.c (enum_tally): new methods Enumerable#tally, which group and count elements of the collection. [Feature #11076]

**Revision 5133bfad - 02/08/2019 09:57 AM - nobu (Nobuyoshi Nakada)**

enum.c: [DOC] tally does not take a block

[Feature #11076]

**Revision 67039 - 02/08/2019 09:57 AM - nobu (Nobuyoshi Nakada)**

enum.c: [DOC] tally does not take a block

[Feature #11076]

## History

**#1 - 04/20/2015 01:32 AM - shevegen (Robert A. Heiler)**

Can you also add a sentence or two for documentation? :)

It may lower the entry barrier for adding a method such as the above (I assume it must be documented by someone before it could be added).

**#2 - 04/20/2015 06:50 AM - nobu (Nobuyoshi Nakada)**

- Description updated

[https://github.com/ruby/ruby/compare/trunk...nobu:feature/11076-Enumerable%23count_by](https://github.com/ruby/ruby/compare/trunk...nobu:feature/11076-Enumerable%23count_by)

**#3 - 04/20/2015 08:14 AM - duerst (Martin Dürst)**

Having this would definitely be very useful. I remember having searched for a `count_by` method more than once in the past.

**#4 - 04/20/2015 02:08 PM - ko1 (Koichi Sasada)**

+1
Robert A. Heiler wrote:

Can you also add a sentence or two for documentation? :-)

I am sorry but I am not sure to properly format this, but the documentation would be like:

Syntax:
```ruby
group_by { |obj| block } → a_hash
```
```ruby
group_by → an_enumerator
```

Description:
Groups the collection by result of the block. Returns a hash where the keys are the evaluated result from the block and the values are the number of the elements in the collection that correspond to the key.

If no block is given an enumerator is returned.

Examples:
```ruby
['a','a','a','b','c'].group_by { |x| x } #=> {'a'=>3, 'b'=>1, 'c'=>1}
```
```ruby
(1..7).group_by { |i| i%3 } #=> {0=>2, 1=>3, 2=>2}
```

#6 - 06/29/2018 01:01 AM - baweaver (Brandon Weaver)

Has there been any thought on this as a language feature?

There was an earlier conversation demonstrating a practical use for this feature, and I had mentioned a few of the core maintainers to bring the subject back into consideration:

https://twitter.com/keystonelemur/status/1012434696909852672

nobu had recently updated his patch here:

https://github.com/ruby/ruby/compare/trunk...nobu:feature/11076-Enumerable%23count_by

I still believe this would be an incredibly useful feature to have in the core of the language, as a very common pattern to work around it is unintuitive for newer programmers:

```ruby
# Most common
array
  .group_by { |v| v }
  .map { |k, v| [k, v.size] }
  .to_h

# In older versions:
Hash[array.group_by { |v| v }.map { |k, v| [k, v.size] }]

# or in more recent versions:
array
  .group_by { |v| v }
  .transform_values(&:size)

# or using reduce / ewo:
array.each_with_object(Hash.new(0)) { |v, h| h[v] += 1 }
```

By giving a name to this concept, we've made it more accessible as well. Given the current trend of 2.6, I believe this would be a welcome addition.

#7 - 08/09/2018 07:58 AM - knu (Akinori MUSHA)

In today's developer meeting, Matz understood the need for the feature but didn't like the name. One point he made was that existing pairs like sort/sort_by and max/max_by share their features, so count_by() might not go well with count().

#8 - 08/09/2018 06:15 PM - baweaver (Brandon Weaver)

group_count? It's half-way between group_by and count

#9 - 08/10/2018 11:37 AM - janfri (Jan Friedrich)

As Naruse in DevelopersMeeting20180809 mentioned: It is a histogram function. How about histogram_by (and for the block-less counterpart histogram)?

#10 - 08/21/2018 05:55 PM - djones (David Jones)

How about tally?
array = ["aa", "aA", "bb", "cc"]
p.array.tally(&:downcase) #=> ({"aa"=>2,"bb"=>1,"cc"=>1})

tally describes quite well to me what this method does and avoids clashing with group or count.
tally_by might be worthy of consideration too.

**Definition of "Tally"**

Current score or amount: *that takes his tally to 10 goals in 10 games.*

1. a record of a score or amount: *I kept a running tally of David's debt on a note above my desk.*
2. a particular number taken as a group or unit to facilitate counting.
3. a mark registering a number or amount.
4. an account kept by means of a tally.

---

#11 - 12/13/2018 12:23 AM - baweaver (Brandon Weaver)

matz (Yukihiro Matsumoto) / ko1 (Koichi Sasada): Any chance of this making it into 2.6? The code is already done (thanks nobu (Nobuyoshi Nakada)) and the only consideration left is the name. Would tally_by be an acceptable compromise?

#12 - 12/13/2018 10:32 AM - janfri (Jan Friedrich)

Just my 2 cents: I'm not a native English speaker. Never heard the word "tally" before. So I would never remember it and has always to look at the api docs.

#13 - 12/14/2018 05:35 PM - odlp (Oliver Peate)

For me the definition of tally does seem to fit the use case, so +1 to tally_by.

Couple of alternatives, how about:

- census (as in census_by(&:downcase))
- inventory (either inventory or inventory_by)

Both are more widely used than tally (although I think tally is the better choice):


#14 - 01/24/2019 04:22 AM - inopinatus (Joshua GOODALL)

A histogram refers to counts of items in ranges of otherwise continuous data. But this function is more general than that, so I think histogram is too specific a term.

For this native English speaker, tally is the most precisely fitted method name.

#15 - 01/24/2019 02:21 PM - mame (Yusuke Endoh)

I have learnt the word "tally" in this thread. Thank you. It looks good to me, a non-native speaker. I have put this on the agenda of the next developers' meeting.

By the way, what is the precise semantics of the method?

**Question 1. What identity is the object in the keys?**

```ruby
str1 = "a"
str2 = "a"
t = [str1, str2].tally
p t #=> { "a" => 2 }
p t.keys.first.object_id #=> str1.object_id or str2.object_id ?

IMO: I think it should prefer the first element, so it should be equal to str1.object_id.
```

**Question 2. What is the key of tally_by?**

```ruby
str1 = "a"
str2 = "A"
t = [str1, str2].tally_by(&:upcase)
p t #=> { "a" => 2 } or { "A" => 2 } ?
p t.keys.first.object_id #=> str1.object_id, str2.object_id, or otherwise?

IMO: The return values of sort_by and max_by contains the original elements, not the return value of the block. According to the analogy to them, I
think that t should be { "a" => 2 } and its key be str1.object_id.

#16 - 01/25/2019 12:11 AM - mrkn (Kenta Murata)
enumerable-statistics provides value_counts method.
It is designed to follow pandas’s Series.value_counts.

#17 - 01/29/2019 01:04 AM - baweaver (Brandon Weaver)
mame (Yusuke Endoh) wrote:

I have learnt the word “tally” in this thread. Thank you. It looks good to me, a non-native speaker. I have put this on the agenda of the next developers’ meeting.

By the way, what is the precise semantics of the method?

Question 1. What identity is the object in the keys?

str1 = "a"
str2 = "a"
t = [str1, str2].tally
p t #=> { "a" => 2 }
p t.keys.first.object_id #=> str1.object_id or str2.object_id ?

IMO: I think it should prefer the first element, so it should be equal to str1.object_id.

Question 2. What is the key of tally_by?

str1 = "a"
str2 = "A"
t = [str1, str2].tally_by(&:upcase)
p t #=> { "a" => 2 } or { "A" => 2 } ?
p t.keys.first.object_id #=> str1.object_id, str2.object_id, or otherwise?

IMO: The return values of sort_by and max_by contains the original elements, not the return value of the block. According to the analogy to them, I think that t should be { "a" => 2 } and its key be str1.object_id.

Answer 1: I would say the first, but tally could also be effectively represented by tally_by(&:itself) as shown in an implementation below:

Answer 2: The transformed value, like group_by:

[1, 2, 3].group_by(&:even?)
=> {false=>[1, 3], true=>[2]}

[1, 2, 3].tally_by(&:even?)
=> {false => 2, true => 1}

The implementation is similar to this:

module Enumerable
  # Implementing via group_by
  def tally_by(&fn)
    group_by(&fn).to_h { |k, vs| [k, vs.size] }
  end

  # Implementing via reduction
  def tally_by2(&fn)
    each_with_object(Hash.new(0)) { |v, a| a[fn[v]] += 1 }
  end
end

...which would result in the first object_id I believe.

#18 - 01/29/2019 02:52 AM - nobu (Nobuyoshi Nakada)
https://github.com/nobu/ruby/pull/new/feature/11076-Enumerable%23tally

As Hash[]= copies string keys, the object_id will be unique unless the item is frozen.
For this kind of method, I wish we would implement it in Ruby even in MRI: it's much simpler, more readable, and every Ruby implementation could use it.

knu (Akinori MUSHA) wrote:

In today's developer meeting, Matz understood the need for the feature but didn't like the name. One point he made was that existing pairs like sort/sort_by and max/max_by share their features, so count_by() might not go well with count().

Since this feature is an inferior variant of group_by in the sense that it reduces the value arrays into their lengths, what about naming the method group?

Then, group can be read as "group the block evaluation (with their counts provided as additional information)" while group_by can be read as "group the receiver by the block evaluation".

I personally feel that it is overkill to give a new unrelated name (such as tally) for such a feature that looks quite specific and narrow in nature.

And it is also a good opportunity to fill in the empty slot for the by-less variant of group_by, which has made group_by stand out and a bit awkward.

Answer 2: The transformed value, like group_by:

```ruby
[1, 2, 3].group_by(&:even?)
=> {false=>[1, 3], true=>[2]}

[1, 2, 3].tally_by(&:even?)
=> {false => 2, true => 1}
```

If we have tally, we can implement this behavior easily: `[1, 2, 3].map { |x| x.even? }.tally. Is a new method really needed just for a shorthand of this behavior?

OK, tally sounds reasonable. Accepted.

Matz.

- Assignee set to mame (Yusuke Endoh)
- Status changed from Open to Assigned

Thanks, I'll implement it.

Note that tally_by is not accepted yet. We need to discuss the detail later (if needed).

Nobu has already started creating a patch. Leave it to him.
enum.c: Enumerable#tally

- enum.c (enum_tally): new methods Enumerable#tally, which group and count elements of the collection.  [Feature #11076]

#27 - 02/14/2019 07:36 PM - baweaver (Brandon Weaver)

mame (Yusuke Endoh) wrote:

Answer 2: The transformed value, like group_by:

```
[1, 2, 3].group_by(&:even?)
  => [false=>[1, 3], true=>[2]]
```

```
[1, 2, 3].tally_by(&:even?)
  => [false => 2, true => 1]
```

If we have tally, we can implement this behavior easily: `[1, 2, 3].map{|x|x.even?}.tally. Is a new method really needed just for a shorthand of this behavior?

It's a common enough that the syntax may be justified. It could be argued that a lot of shorthand expressions aren't technically necessary, but I feel that this makes Ruby Ruby, the ability to say something common with less.

That, and there's established precedent of count / count_by, max / max_by, and others that would make this an easily adopted syntax. If it's not adopted I would not be surprised to see a follow-up request to add it.

I would see tally_by and other *_by methods as the base for their counterparts, such that:

```
[1,2,3].tally => [1,2,3].tally_by(&:itself)
```

Where the non-*_by method is effectively the *_by method implemented with the itself identity function.

#28 - 02/15/2019 12:30 AM - mame (Yusuke Endoh)

baweaver (Brandon Weaver) wrote:

It's a common enough that the syntax may be justified.

That's just because "map + something" is frequent. However, blindly adding a "map" feature to anything does not make sense to me. In fact, "map + select" is much more frequent, but it is not introduced yet (#5663, #15323). If we add "tally_by" as a shorthand to "map + tally", we should confirm if the combination is truly frequent (i.e., "tally" is rarely used without "map"). We can do it after only "tally" is released.

#29 - 05/02/2019 02:41 PM - jonathanhefner (Jonathan Hefner)

"map + select" is much more frequent, but it is not introduced yet

I think it would also be nice if filter_map was added. However, a specific justification for adding tally_by is to avoid an extra array allocation. filter_map can already be expressed as map {...}.compact! to avoid allocating an extra array. But there is no way to avoid an extra allocation with map {...}.tally.