Ruby master - Feature #10552

[PATCH] Add Enumerable#frequencies and Enumerable#relative_frequencies

11/27/2014 07:59 AM - brianhempel (Brian Hempel)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Counting how many times a value appears in some collection has always been a bit clumsy in Ruby. While Ruby has enough constructs to do it in one line, it still requires knowing the folklore of the optimum solution as well as some acrobatic typing:

```
%w[cat bird bird horse].each_with_object(Hash.new(0)) { |word, hash| hash[word] += 1 }
# => {"cat" => 1, "bird" => 2, "horse" => 1}
```

What if Ruby could count for us? This patch adds two methods to enumerables:

```
%w[cat bird bird horse].frequencies
# => {"bird" => 2, "horse" => 1, "cat" => 1}

%w[cat bird bird horse].relative_frequencies
# => {"bird" => 0.5, "horse" => 0.25, "cat" => 0.25}
```

To make programmers happier, the returned hash has the most common values first. This is nice because, for example, finding the most common element of a collection becomes trivial:

```
most_common, count = %w[cat bird bird horse].frequencies.first
```

Whereas the best you can do with vanilla Ruby is:

```
most_common, count = %w[cat bird bird horse].each_with_object(Hash.new(0)) { |word, hash| hash[
word] += 1 }.max_by(&:last)

# or...

most_common, count = %w[cat bird bird horse].group_by(&:to_s).map { |word, arr| [word, arr.size]
}.max_by(&:last)
```

While I don't like the long method names, "frequencies" and "relative frequencies" are the terms used in basic statistics.
http://en.wikipedia.org/wiki/Frequency_%28statistics%29

Related issues:

Related to Ruby master - Feature #9970: Add `Hash#map_keys` and `Hash#map_val...	**Open**	
Related to Ruby master - Feature #7793: New methods on Hash	**Assigned**	
Related to Ruby master - Feature #10228: Statistics module	**Feedback**	**09/11/2014**

History

#1 - 11/27/2014 03:03 PM - workmad3 (David Workman)

I like this idea, but I think it could be improved by allowing .frequencies to take a block and it will count the frequencies of the return value of the block, similar to .all?, .any? and .none?

This would allow the frequencies method to be useful not just on arrays of strings but also able to be used on more complex data structures without having to do a .map first to massage data into the desired format first.

#2 - 11/27/2014 07:29 PM - brianhempel (Brian Hempel)

Thanks for the feedback David. I can see a map functionality being useful, but here I will play some arguments against integrating map:

1. In the future, I was thinking the block could be used to change the weighting: some elements might count as 1, but others are less important so each of them only counts as 0.5. However, I can't think of a good use case for that yet.
2. any? all? and none? return booleans, not collections. All of the other enumerable methods that return a collection return elements from the original enumerable. For example, my_enum.group_by(&:relation) has elements from my_enum in the hash value arrays. It's a small code smell that my_enum.frequencies(&:relation) would return a potentially large collection that contains nothing from my_enum.

3. any? all? and none? can exit early, so there's a performance improvement to .any?(&:finished?) compared to .map(&:finished?).any?. There would be little performance improvement here because frequencies always has to walk the entire collection.

On the other hand, here is one good argument for integrating map:

1. Enumerable#count takes a block to specify what to count, and frequencies is basically count, but on all elements at once.

#3 - 11/28/2014 07:15 AM - duerst (Martin Dürst)

frequencies is essentially a group_by with the values mapped with size/count.

So assuming something like issue #9970 or issue #7793 gets accepted, it could simply be written as
%w[cat bird bird horse].group_by {|x| x}.map_values {|v| v.count }
or, if we get an identity method (*), as:
%w[cat bird bird horse].group_by(&:identity).map_values &:count

While this may not be very short, it's a concise description of what actually happens. I think it would be better for Ruby to improve how such general transformations can be written, rather than add more and more specialized methods methods such as (relative_)frequency. Such methods better would go into a statistics package (see 10228; would be good to have, too, of course.)

(*) I thought we had an issue for this, but couldn't find it.

#4 - 11/28/2014 08:38 AM - brianhempel (Brian Hempel)

Yes, I would rather see Hash#map_values in Ruby before Enumerable#frequencies. However, if both map_values and frequencies were added, then we might not need relative_frequencies, since calculating it becomes cleaner:

```
array = %w[cat bird bird horse]
array.frequencies.map_values { |n| n.to_f / array.size }
```

I think that counting everything up is more like pre-statistics. I want to count things more often than I want to take a mean or a standard deviation. Also, most statistical measures operate only on collections of numbers. In contrast, counting frequencies works on collections of anything, not just numbers.

We could call the method counts instead of frequencies to make it sound less like statistics and more like counting.

To revise your example in favor of this patch: if you want the frequencies sorted, the "manual way" becomes longer:

```
%w[cat bird bird horse].group_by(&:identity).map_values(&:count).sort_by(&:last).reverse.to_h
```

#5 - 11/29/2014 03:04 AM - duerst (Martin Dürst)

- Related to Feature #9970: Add `Hash#map_keys` and `Hash#map_values` added

#6 - 11/29/2014 03:04 AM - duerst (Martin Dürst)

- Related to Feature #7793: New methods on Hash added

#7 - 11/29/2014 03:04 AM - duerst (Martin Dürst)

- Related to Feature #10228: Statistics module added

#8 - 11/30/2014 03:57 AM - Ajedi32 (Andrew M)

Personally, I'd prefer the form Enumerable#count_by with a block, as this method seems very similar to group_by in my opinion. I also think relative_frequencies is unnecessary, as I think map_values (or transform_values or whatever we end up calling it) would serve that purpose well enough.

#9 - 11/30/2014 11:56 AM - shevegen (Robert A. Heiler)

I like the word .frequencies - it seems nicer than each_with_object(Hash.new(0)) and also than group_by.

I do not like the word .relative_frequencies but I can understand why you want this - it seems more a subpart of statistics though, and would perhaps be better placed into some extension into ruby (either into math, or perhaps statistics, which could be a subproject of ruby math).

On a side-note, perhaps we can also improve on ruby statistics functionality a bit. I'd rather use Ruby than R, speed difference is no issue for me, but Ruby is much nicer to work with than R.

Files

add_enum_frequencies.patch	5.81 KB	11/27/2014	brianhempel (Brian Hempel)