[])transpose should behave specially

08/10/2012 08:36 PM - Anonymous

Status: Rejected
Priority: Normal
Assignee:
Target version: 2.0.0

Description

p = [1, 2, 3]
q = [4, 5, 6]
[p, q].transpose

=> 1, 4], [2, 5], [3, 6

As expected, 2 x 3 vector was converted into 3 x 2.

[p].transpose

=> 1], [2], [3

As expected, 1 x 3 => 3 x 1.

[]Transpose

=> []

Unexpected, 0 x 3 did not become 3 x 0: [], [], []

In other words, when [] is the receiver, transpose has no way to know what kind of ** 2 dimensional ** object is it - whether 0 x 3, 0 x 4, 0 x 1 or perhaps 0 x 0. #transpose should not assume it is 0 x 0. It should raise, or warn, or complain, or require argument for this case, in short, it should behave differently than today.

History

#1 - 08/10/2012 09:16 PM - alexeymuranov (Alexey Muranov)

I think it can be agreed that all of the following represent the same "empty matrix": [], [], [], [], []]. Otherwise they all would need to be treated exceptionally (how to express the transposition of each of them?). After all, 0 x 3 = 0 x 2 = 3 x 0 = 0 x 0 = 0.

#2 - 08/11/2012 01:57 AM - marcandre (Marc-Andre Lafortune)
- Category set to core
- Target version set to 2.0.0

Hi,

alexeymuranov (Alexey Muranov) wrote:

I think it can be agreed that all of the following represent the same "empty matrix": [], [], [], [], []]. Otherwise they all would need to be treated exceptionally (how to express the transposition of each of them?). After all, 0 x 3 = 0 x 2 = 3 x 0 = 0 x 0 = 0.

Actually, no. Empty matrices are well defined: http://en.wikipedia.org/wiki/Matrix_(mathematics)#Empty_matrices

The matrix library handles then well too:

require 'matrix'
Matrix[][[] == Matrix[] # => false
Matrix[][[]].transpose.column_size # => 3
boris_stitnicky (Boris Stitnicky) wrote:

In other words, when [] is the receiver, ... it should raise, or warn, or complain

Strictly speaking you are right.

I'm not sure it's that much more useful to raise, though. I think the current behavior might be more helpful.

#3 - 08/11/2012 02:29 AM - marcandre (Marc-Andre Lafortune)

Hi,

boris_stitnicky (Boris Stitnicky) wrote:

In other words, when [] is the receiver, ... it should raise, or warn, or complain

Strictly speaking you are right.

Oups, I was not thinking straight. [] corresponds to a 0x0 matrix (i.e. Matrix[] or Matrix.empty(0,0)). It is its own transpose.

So the current behavior is correct.

What could be said is that [], [], [] .transpose is not completely accurate in returning [], but no other result is possible. For more accurate handling... use the matrix library.

I'll close this if there are no other objections.

#4 - 08/11/2012 06:20 AM - alexeymuranov (Alexey Muranov)

marcandre (Marc-Andre Lafortune) wrote:

Oups, I was not thinking straight. [] corresponds to a 0x0 matrix (i.e. Matrix[] or Matrix.empty(0,0)). It is its own transpose.

Then it is not possible to express in the same way a 3 x 0 matrix (or 0 x 3?) I think this was the reason for the original question. The result of [], [], [] .transpose is then wrong.

I think it is not completely wrong to allow oneself for certain purposes, like for addition and multiplication, to identify matrices of different dimensions which can be obtained from one another by inserting zeros, and also to identify all empty and all zero matrices. The exact dimension is probably needed only for the determinant and the inverse matrix.

It is just a possible explanation for the current behavior.

#5 - 08/13/2012 09:12 PM - marcandre (Marc-Andre Lafortune)

Hi,

alexeymuranov (Alexey Muranov) wrote:

Then it is not possible to express in the same way a 3 x 0 matrix (or 0 x 3?) I think this was the reason for the original question.

A 3x0 matrix corresponds to [], [], [], but there is no correspondence for a 0x3 matrix. So strictly speaking, [], [], [] .transpose has no valid answer, but returning [] is more useful than raising I believe.

#6 - 08/13/2012 09:23 PM - Anonymous

A 3x0 matrix corresponds to [], [], [], but there is no correspondence for a 0x3 matrix. So strictly speaking, [], [], [] .transpose has no valid answer, but returning [] is more useful than raising I believe.

It would be interesting to see code that found returning [] to be useful. As in, code that expected to operate on the transposed result.

To me, [], [], [] .transpose smells like division by zero, and because I can't imagine enjoying the result, I'm fine with an exception.
A 3x0 matrix corresponds to [], [], []], but there is no correspondence for a 0x3 matrix. So strictly speaking, [], [], []].transpose has no valid answer, but returning [] is more useful than raising I believe.

It would be interesting to see code that found returning [] to be useful. As in, code that expected to operate on the transposed result.

I can imagine code that wants to iterate on all elements, but going through columns first. E.g:

```ruby
# Instead of
exams.each_with_index do |exam, i|
  # do something with exam and grades[i]
end
# Using transpose (or zip) is nicer:
[exams, grades].transpose.each do |exam, grade|
  # do the same with exam and grade
end
```

The later would fail for an empty set of exams and grades

Note that [[]], [[]], [[]].transpose is consistent with []].zip([], []]

So there are at least some situations when [] would be a useful result. And while there are situations where no result would be useful, it doesn't much matter in those situations whether you get an exception or a result that isn't useful.

I'm convinced. :-)

I didn't understand this sentence.

Anyway, I just want to give another explanation why it is possible to assume that there is only one empty matrix. With this assumption the function of #transpose will become correct, as [] == [] == [[]] == ... (as matrices).

One of the ways to define a matrix is as an \( m \times n \) rectangular table of numbers, more precisely a function from \( \{0, \ldots, m-1\} \times \{0, \ldots, n-1\} \) to the numbers. If \( m \) or \( n \) is equal to 0, then the domain is the empty set, and so is the only function on this domain: \( [] \times [] = [] \times \{0,1,2\} = \{0,1,2\} \times [] = [] = \emptyset \), and the only function defined on \( \emptyset \) is \( \emptyset \). I used here set theoretic conventions.

Of course matrices can also be viewed as functions from \( \mathbb{R}^m \) to \( \mathbb{R}^n \) then \( \mathbb{R}^0 = \{\emptyset\} \) is a one-element set, and “empty matrices” of different dimension
will not be the same. But I think this viewpoint is less standard.

Alexey.

#10 - 08/15/2012 01:21 PM - marcandre (Marc-Andre Lafortune)
- Status changed from Open to Rejected

Hi,

alexeymuranov (Alexey Muranov) wrote:

marcandre (Marc-Andre Lafortune) wrote:

The later would fail for an empty set of exams and grades

I didn't understand this sentence.

The example using each_with_index will work even for empty arrays. The "equivalent" example with transpose currently works with empty arrays but would not if transpose returned nil or raised an error on \[
\]
. It would make "equivalent" code behave differently in that case, and I feel that it wouldn't be more useful.

#11 - 08/15/2012 04:09 PM - alexeymuranov (Alexey Muranov)

Ok, thanks, I've understood.

#12 - 08/16/2012 07:57 AM - Anonymous

I see, so as long as the exact behavior is provided by the matrix library, the \[].transpose can afford to be pragmatic.