Ruby master - Bug #12548

Rounding modes inconsistency between round versus sprintf

07/04/2016 09:20 AM - unclekiki (Kieran McCusker)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	2.4.0	**Backport:**	2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN

Description

Hi

I've found this possible bug with sprintf in our production code, but it seems very odd that no one has reported it so I'm doubting myself.

I've tested a few versions (CRuby 2.3.1 and 2.1.9) and it was present in both.

To reproduce in irb:

```
sprintf('%1.0f', 12.5)
```

Expected result 13 actual 12

In fact if you look at the number sequence 0.5 -> 99.5 you find the even half will all round down and the odd half round up. 12.5.round produces the correct result (13)

If you do the same in jruby 2.2.1 you get the expected result of 13.

Many thanks

Kieran

Related issues:

Related to Ruby master - Bug #12958: Breaking change in how `#round` works	**Closed**

Associated revisions

Revision dfe91fcd - 11/05/2016 09:49 AM - nobu (Nobuyoshi Nakada)

numeric.c: round to nearest even

- numeric.c (flo_round, int_round): support round-to-nearest-even semantics of IEEE 754 to match sprintf behavior, and add half: optional keyword argument for the old behavior. [ruby-core:76273] [Bug #12548]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@56590 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 56590 - 11/05/2016 09:49 AM - nobu (Nobuyoshi Nakada)

numeric.c: round to nearest even

- numeric.c (flo_round, int_round): support round-to-nearest-even semantics of IEEE 754 to match sprintf behavior, and add half: optional keyword argument for the old behavior. [ruby-core:76273] [Bug #12548]

Revision 56590 - 11/05/2016 09:49 AM - nobu (Nobuyoshi Nakada)

numeric.c: round to nearest even

- numeric.c (flo_round, int_round): support round-to-nearest-even semantics of IEEE 754 to match sprintf behavior, and add half: optional keyword argument for the old behavior. [ruby-core:76273] [Bug #12548]

Revision 56590 - 11/05/2016 09:49 AM - nobu (Nobuyoshi Nakada)

numeric.c: round to nearest even

- numeric.c (flo_round, int_round): support round-to-nearest-even semantics of IEEE 754 to match sprintf behavior, and add half: optional keyword argument for the old behavior. [ruby-core:76273] [Bug #12548]

Revision 56590 - 11/05/2016 09:49 AM - nobu (Nobuyoshi Nakada)

numeric.c: round to nearest even

- numeric.c (flo_round, int_round): support round-to-nearest-even semantics of IEEE 754 to match sprintf behavior, and add half: optional keyword argument for the old behavior. [ruby-core:76273] [Bug #12548]

Revision 114d1751 - 12/21/2016 01:29 AM - nobu (Nobuyoshi Nakada)

numeric.c: allow nil as rounding mode option

- numeric.c (rb_num_get_rounding_option): allow nil same as the default behavior, per [ruby-core:77961]. [Bug #12548]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@57130 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 57130 - 12/21/2016 01:29 AM - nobu (Nobuyoshi Nakada)

numeric.c: allow nil as rounding mode option

- numeric.c (rb_num_get_rounding_option): allow nil same as the default behavior, per [ruby-core:77961]. [Bug #12548]

Revision 57130 - 12/21/2016 01:29 AM - nobu (Nobuyoshi Nakada)

numeric.c: allow nil as rounding mode option

- numeric.c (rb_num_get_rounding_option): allow nil same as the default behavior, per [ruby-core:77961]. [Bug #12548]

Revision 57130 - 12/21/2016 01:29 AM - nobu (Nobuyoshi Nakada)

numeric.c: allow nil as rounding mode option

- numeric.c (rb_num_get_rounding_option): allow nil same as the default behavior, per [ruby-core:77961]. [Bug #12548]

Revision 57130 - 12/21/2016 01:29 AM - nobu (Nobuyoshi Nakada)

numeric.c: allow nil as rounding mode option

- numeric.c (rb_num_get_rounding_option): allow nil same as the default behavior, per [ruby-core:77961]. [Bug #12548]

Revision 581b995c - 12/21/2016 01:29 AM - nobu (Nobuyoshi Nakada)

numeric.c: rdoc of half option [ci skip]

- numeric.c (flo_round): [DOC] mention half option. [Bug #12548]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@57131 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 57131 - 12/21/2016 01:29 AM - nobu (Nobuyoshi Nakada)

numeric.c: rdoc of half option [ci skip]

- numeric.c (flo_round): [DOC] mention half option. [Bug #12548]

Revision 57131 - 12/21/2016 01:29 AM - nobu (Nobuyoshi Nakada)

numeric.c: rdoc of half option [ci skip]

- numeric.c (flo_round): [DOC] mention half option. [Bug #12548]

Revision 57131 - 12/21/2016 01:29 AM - nobu (Nobuyoshi Nakada)

numeric.c: rdoc of half option [ci skip]

- numeric.c (flo_round): [DOC] mention half option. [Bug #12548]

Revision 57131 - 12/21/2016 01:29 AM - nobu (Nobuyoshi Nakada)

numeric.c: rdoc of half option [ci skip]

- numeric.c (flo_round): [DOC] mention half option. [Bug #12548]

History

#1 - 07/05/2016 03:01 AM - shyouhei (Shyouhei Urabe)

I doubt if this is a bug or not. The rounding mode you describe is so-called "round to the nearest even" which is described in ISO 31-0 as "generally preferable". Maybe the author of sprintf explicitly chose such mode (not sure).

It seems the problem here is whether should we specify sprintf's rounding mode or leave it implementation-defined.

#2 - 07/05/2016 09:07 AM - unclekiki (Kieran McCusker)

Having read a bit more I see that there are indeed multiple possible ways of handling tie-breaking i.e. when a value is n.5, but is it reasonable for two implementations in ruby core (sprintf and round) to use a different method? It's at least surprising (very in my case) and will catch out anyone coming from other programming backgrounds.

#3 - 07/06/2016 03:28 AM - shyouhei (Shyouhei Urabe)

- Subject changed from Ruby sprintf bug to Rounding modes inconsistency between round versus sprintf

I agree that can be a point. I changed the subject.

#4 - 08/09/2016 08:37 AM - matz (Yukihiro Matsumoto)

It should be consistent. Will be fixed.

Matz.

#5 - 08/16/2016 04:26 PM - noahgibbs (Noah Gibbs)

It looks like floats are rounded via numeric.c, using the flo_round function. That definitely does *not* use round-even. Maybe it should? It already implements a rounding function, so it shouldn't be too hard to implement round-even.

It looks like sprintf uses ruby_sprintf, through a series of calls to BSD_vfprintf in vsnprintf.c. At least on my machine. And BSD_vprintf winds up calling ruby_dtoa (#defined from BSD__dtoa) with mode 3. If you look for "round-even" in util.c, you'll see an extensive comment explaining why they used round-even mode.

So basically: right now, Numeric#round uses flo_round(), while sprintf uses util.c's round-even mode.

Presumably we should change flo_round to do round-even.

#6 - 08/17/2016 02:54 AM - nobu (Nobuyoshi Nakada)

I agree that round-even is preferable, but it introduces an incompatibility.
Some libraries, e.g., rexml, depend on the current behavior.
So I'm thinking an optional parameter to select rounding mode.

```
2.5.round        #=> 3
2.5.round(:even) #=> 2
```

#7 - 08/17/2016 07:32 PM - noahgibbs (Noah Gibbs)

Nobu: but then, default will still be inconsistent.

Maybe add a parameter and require libraries to supply it if they care? Then later we could change the default from :even to :closest or :biased or whatever we call the common rounding behavior.

#8 - 08/18/2016 01:15 AM - shyouhei (Shyouhei Urabe)

I doubt if we need multiple rounding modes. I guess for every situation when a specific rounding mode is mandatory, what is needed is the round-even mode. I have never experienced the needs of other modes. Almost everybody don't care what mode they use.

Defaulting to that rounding mode should just suffice. Adding parameter solves a problem that doesn't exist.

#9 - 08/18/2016 04:49 PM - noahgibbs (Noah Gibbs)

Sounds good to me. Do we need to fix ReXML, if it depends on the current rounding mode?

#10 - 08/22/2016 01:51 AM - noahgibbs (Noah Gibbs)

Another reason not to add an argument: Float#round already takes an argument for the number of digits to round to.

#11 - 08/22/2016 09:01 AM - nobu (Nobuyoshi Nakada)

Noah Gibbs wrote:

> Another reason not to add an argument: Float#round already takes an argument for the number of digits to round to.

It's easy to distinguish a Fixnum and a Symbol.
Or keyword arguments will work well.

#12 - 08/22/2016 09:26 AM - nobu (Nobuyoshi Nakada)

Noah Gibbs wrote:

> Sounds good to me. Do we need to fix ReXML, if it depends on the current rounding mode?

The rdoc of REXML::Functions.number states:

> a string that consists of optional whitespace followed by an optional minus sign followed by a Number followed by whitespace is converted to the IEEE 754 number that is nearest (according to the **IEEE 754 round-to-nearest** rule) to the mathematical value represented by the string; any other string is converted to NaN

#13 - 08/22/2016 01:18 PM - noahgibbs (Noah Gibbs)

Okay, so sounds like just a documentation fix for ReXML. That makes sense. Thanks!

#14 - 09/13/2016 07:55 PM - noahgibbs (Noah Gibbs)

Ah - looks like the ReXML docs don't need to change. The round-even mode Ruby uses is still referred to as "round to nearest, ties to even" - https://en.wikipedia.org/wiki/IEEE_floating_point#Roundings_to_nearest

#15 - 09/21/2016 07:58 PM - noahgibbs (Noah Gibbs)

- ruby -v changed from 2.3.1 to 2.4.0

- File round_even.patch added

Here's a patch that makes floating-point rounding behavior match sprintf()'s round-to-even. It also adds a unit test for the new behavior. The change doesn't break any existing unit tests.

#16 - 09/22/2016 01:13 AM - nobu (Nobuyoshi Nakada)

Your patch breaks two tests.

```
[24/42] TestVector#test_round = 0.00 s
  1) Failure:
TestVector#test_round [test/matrix/test_vector.rb:162]:
<Vector[1.23, 2.34, 3.4]> expected but was
<Vector[1.23, 2.35, 3.4]>.

[30/42] REXMLTests::FunctionsTester#test_floor_ceiling_round = 0.02 s
  2) Failure:
REXMLTests::FunctionsTester#test_floor_ceiling_round [test/rexml/test_functions.rb:168]:
<[<b id='1'/>]> expected but was
<[]>.
```

#17 - 09/22/2016 04:39 AM - noahgibbs (Noah Gibbs)

- File round_even.patch added

You're right. I apologize. I had been running test-ruby instead of test-all without realizing it.

It appears both tests won't work if we correctly implement the round-even behavior, so the new patch fixes those.

New patch is attached.

#18 - 09/22/2016 04:39 AM - noahgibbs (Noah Gibbs)

- File deleted (round_even.patch)

#19 - 09/22/2016 04:40 AM - noahgibbs (Noah Gibbs)

- File round_even.patch added

#20 - 09/22/2016 04:40 AM - noahgibbs (Noah Gibbs)

- File deleted (round_even.patch)

#21 - 09/22/2016 06:33 AM - nobu (Nobuyoshi Nakada)

Noah Gibbs wrote:

> It appears both tests won't work if we correctly implement the round-even behavior, so the new patch fixes those.

That is what I had mentioned.

#22 - 09/22/2016 07:01 AM - nobu (Nobuyoshi Nakada)

I have a patch too, but it breaks compatibilities in rubyspec.

```
1)
Float#round returns rounded values for big values FAILED
Expected 20000000000000000000
to have same value and type as 30000000000000000000

spec/rubyspec/core/float/round_spec.rb:70:in `block (2 levels) in <top (required)>'
spec/rubyspec/core/float/round_spec.rb:3:in `<top (required)>'

2)
Integer#round returns itself rounded if passed a negative value FAILED
Expected 200
 to have same value and type as 300

spec/rubyspec/core/integer/round_spec.rb:22:in `block (2 levels) in <top (required)>'
spec/rubyspec/core/integer/round_spec.rb:4:in `<top (required)>'

3)
Rational#round with no arguments (precision = 0) returns the truncated value toward the nearest integer FAILED
Expected 0
 to equal 1

spec/rubyspec/shared/rational/round.rb:17:in `block (3 levels) in <top (required)>'
spec/rubyspec/core/rational/round_spec.rb:3:in `<top (required)>'

4)
Time#strftime rounds an offset to the nearest second when formatting with %z FAILED
Expected "+01:01:04"
 to equal "+01:01:05"

spec/rubyspec/core/time/strftime_spec.rb:50:in `block (2 levels) in <top (required)>'
spec/rubyspec/core/time/strftime_spec.rb:7:in `<top (required)>'
```

#23 - 09/22/2016 03:10 PM - noahgibbs (Noah Gibbs)

nobu (Nobuyoshi Nakada): makes sense. I'll file an issue with RubySpec to let them know that at least some of these tests aren't compatible with IEEE 754 round-nearest-to-even mode. Examining the first test failure (spec/core/float/round_spec.rb line 70), it clearly isn't since it expects 2.5 to round to 3 at that decimal place.

I'll examine the other failures and try to make sure they're all equally clear.

#24 - 09/22/2016 03:25 PM - noahgibbs (Noah Gibbs)

Also: looks like your patch fixes it for integer rounding too, based on your RubySpec failures. Mine does not. Sprintf doesn't have an integer rounding mode, but your way is probably still better.

#25 - 09/22/2016 03:29 PM - noahgibbs (Noah Gibbs)

Opened an issue with RubySpec: https://github.com/ruby/spec/issues/294

#26 - 10/20/2016 07:29 PM - noahgibbs (Noah Gibbs)

Benoit Daloze says that RubySpec should test previous rounding behavior for Ruby before 2.4.0, and test round-even for 2.4.0+. That makes sense to me. Opened RubySpec PR #322 (https://github.com/ruby/spec/pull/322) to reflect this.

When/if we merge Nobu's branch for this bug, we should also ask RubySpec to merge PR #322 or something very similar.

#27 - 10/21/2016 09:30 PM - noahgibbs (Noah Gibbs)

Benoit says RubySpec PR #322 looks good. Just waiting for us to commit round-to-nearest-even mode to MRI so he can merge it. I've used Nobu's branch locally, including to create PR #322, and it works well for me. +1

#28 - 11/05/2016 08:59 AM - nobu (Nobuyoshi Nakada)

We had a discussion today, and the conclusions:

- add half: optional keyword argument to round methods,
- its value must be one of :up, :even, and nil,
- :up is compatible with the existing behavior,
- :even is the new behavior compatible with sprintf, and
- if half: option isn't given or is nil, same as :even.

#29 - 11/05/2016 09:49 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset r56590.

---

numeric.c: round to nearest even

- numeric.c (flo_round, int_round): support round-to-nearest-even semantics of IEEE 754 to match sprintf behavior, and add half: optional keyword argument for the old behavior. [ruby-core:76273] [Bug #12548]

#30 - 11/18/2016 08:40 PM - Eregon (Benoit Daloze)

- Related to Bug #12958: Breaking change in how `#round` works added

#31 - 12/14/2016 01:11 AM - shyouhei (Shyouhei Urabe)

- Status changed from Closed to Open

Reopening.

The fix was reverted due to issue #12958. This issue is now back alive.

#32 - 12/21/2016 01:30 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset r57130.

---

numeric.c: allow nil as rounding mode option

- numeric.c (rb_num_get_rounding_option): allow nil same as the default behavior, per [ruby-core:77961]. [Bug #12548]

#33 - 12/21/2016 01:32 AM - nobu (Nobuyoshi Nakada)

- Status changed from Closed to Open

- Description updated

Probably, [Feature #10000] can add half option to String#%.

#34 - 02/19/2018 10:46 AM - unclekiki (Kieran McCusker)

Hi again

After I raised this bug and saw the comment from matz I stopped watching assuming that sprintf would be "fixed" not that round would be changed instead. IEEE 754 is useful when you are taking them mean of a large number of rounded values and care about the mean. Rounding to even will give you a more accurate rounded mean in large sets of randomly distributed data. When rounding individual numbers it seems very unintuitive

This suggests that the typical use of sprintf is to take a large number of floating point values round them so they can be converted back from string to floats and then summed thus preserving their summed mean. Hummm, would it not be more likely to be used for formatting percentage values etc. where users would not expect 11.5 and 12.5 to be formatted to 12%.

Wouldn't it have been more sensible to make sprintf use Round half away from zero as round does (and JRuby's implementation did when I last looked) this is what I assumed would happen. There may well be a separate issue that round could be made aware of IEEE 754 for the fairly specialized case I first outlined.

As it stands in ruby 2.5 my original issue is still outstanding and the two uses are still inconsistent.

Thanks

Kieran

#35 - 07/10/2018 05:41 PM - vor_lord (Brett Williams)

I don't think that Float#round and sprintf should have different default behavior. Whichever rounding mode is considered default should apply to both. Is there any consensus on this issue?

#36 - 07/11/2018 01:55 AM - shyouhei (Shyouhei Urabe)

The current status is:

- We once have fixed this bug.
- That caused rails breakage.
- Everybody complained.
- So we reverted.

I personally believe this issue is a real bug but it seems I am a minority here. The consensus, if any, is that it should be kept as it is.

See also: https://bugs.ruby-lang.org/issues/12958

#37 - 07/11/2018 12:33 PM - unclekiki (Kieran McCusker)

It is a real bug, it's just that, as I have said before, the "fix" took the wrong direction. sprintf should behave like round (as it does, or at least did, in jruby with no apparent issues). Round even is a very specialised use case.

Imagine you take the height of 100 school children to the nearest half centimetre and then want to output frequencies using sprintf to the nearest whole centimetre. Do you really want the frequencies to spaced like (117), (117.5, 118, 118.5), (119), (119.5, 120, 120.5)

The real solution is to make sprintf perform like round.

As Matz said 2 years ago "it should be consistent. Will be fixed."

#38 - 07/12/2018 12:53 AM - nobu (Nobuyoshi Nakada)

unclekiki (Kieran McCusker) wrote:

> Imagine you take the height of 100 school children to the nearest half centimetre and then want to output frequencies using sprintf to the nearest whole centimetre. Do you really want the frequencies to spaced like (117), (117.5, 118, 118.5), (119), (119.5, 120, 120.5)

Why grouping by sprintf, instead of Float#round?
It doesn't sound a reasonable use-case.

#39 - 07/12/2018 07:53 AM - unclekiki (Kieran McCusker)

Ok

How about listing them in the format

name ..... height (rounded to whole centimetre)

sprintf would seem a great choice for that but fails due to it's rounding choices. By the way, I noticed this in the first place because I was using sprintf to format frequencies on charts using code like sprintf('%1.0f%%', frequency) so it was a reasonable use case for me.

Could you give me a use case for sprintf where using round-even is a good enough choice for the decision to make it behave differently to round is justified? I know it wasn't really a conscious decision of course, but given that we have an implementation of sprintf in jruby that matches round and the sky does not appear to be falling down I find it difficult to understand the reasoning why sprintf rounding wasn't just brought in line with round.

#40 - 04/13/2019 12:46 AM - mackuba (Kuba Suder)

I came across this thread now, because I've noticed that sprintf's behavior has actually changed in Ruby 2.4 too, just... not the same way as round's did...

```
> echo "n = [5.005, 5.015, 5.025]; puts 'round: ' + n.map { |x| x.round(2) }.join(' '); puts 'sprintf: ' + n.m
ap { |x| sprintf('%.2f', x) }.join(' ')" > /tmp/floats.rb

> rvm 2.3.7 do ruby /tmp/floats.rb
round: 5.01 5.02 5.03
sprintf: 5.00 5.01 5.03

> rvm 2.4.0 do ruby /tmp/floats.rb
round: 5.01 5.02 5.03
sprintf: 5.00 5.02 5.02
```

#41 - 03/16/2020 08:02 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Closed

It is inconsistent but fixing either way could cause problems. I vote for keeping as it is.

Matz.

Files

round_even.patch	4.43 KB	09/22/2016	noahgibbs (Noah Gibbs)