Ruby - Bug #16440

Date range inclusion behaviors are inconsistent

12/20/2019 04:22 PM - st0012 (Stan Lo)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.6.5p114 (2019-10-01 revision 67812) [x86_64-darwin19]	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN

Description

It's weird that a Date range can include Time and DateTime objects that were converted from a Date object. But it can't include a newly generated DateTime object. For example:

```
may1 = Date.parse("2019-05-01")
may3 = Date.parse("2019-05-03")
noon_of_may3 = DateTime.parse("2019-05-03 12:00")
may31 = Date.parse("2019-05-31")

(may1..may31).include? may3 # => True
(may1..may31).include? may3.to_time # => True
(may1..may31).include? may3.to_datetime # => True
(may1..may31).include? noon_of_may3 # => False
```

Shouldn't the last case return true as well?

Related Rails issue: https://github.com/rails/rails/issues/36175

History

#1 - 12/20/2019 05:28 PM - wishdev (John Higgins)

Nothing strange with your example - but that doesn't mean it totally works right.

First your example is a DATE range - so adding this line

(may1..may31).each { |x| puts x }

That shows that your set is each day within the range at Midnight - therefore any other time is not included (and in fact on my system - the to_time option returns false instead of true).

BUT, on the other hand - one might imagine that something like

may1 = DateTime.parse("2019-05-01") may31 = DateTime.parse("2019-05-31") noon_of_may3 = DateTime.parse("2019-05-03 12:00") (may1..may31).include? noon_of_may3

Should get a true for the include

It appears though, that DateTime ranges only use the exact Time each day of the range

(may1..may31).each { |x| puts x }

Shows this with the DateTime range.

So I don't believe there is an issue with the code as you have it - but there might be a conversation as to why a DateTime range does not appear to work for your example.

John

#2 - 12/20/2019 05:55 PM - zverok (Victor Shepelev)

Range#include? works as #to_a.include?. E.g. this:

```
(may1..may31).include? noon_of_may3
# => false
```

Is equivalent to this:

```
dates = (may1..may31).to_a # => each Date between May 1 and 31
dates.include? noon_of_may3
# => false
```

What works as you expect (compare value with range begin and end) is Range#cover?:

```
(may1..may31).cover? noon_of_may3
# => true
```

To make things a bit more complicated, there is a special reimplementation for numbers, so (1...2).include?(1.5) is true.

The Range's docs point explain th behavior (though, a bit sparingly):

> Returns true if obj is *an element of the range*, false otherwise. If begin and end are numeric, comparison is done according to the magnitude of the values.

Docs for cover explain how it behaves, too:

> Returns true if obj is between the begin and end of the range.

#3 - 12/21/2019 03:51 AM - shevegen (Robert A. Heiler)

I have no strong opinion either way but I can understand the assumption by st0012 to some extent. For example, I personally always seem to think more about .include? than .cover?, largely because I simply use .include? a lot more. I once even added some .partial_include? method to Enumerable (or somewhere else, I don't remember ... was years ago).

The other thing is DateTime, Date, and Time. Personally I'd love if we could have just one-ring-to-rule-them-all one day, perhaps in ruby 4.0 or so - I think that is a partial complaint by Stan, in the sense of the behaviour he showed (but I am assuming this here). But again, I have no real strong opinion either way.

Would be interesting to ask Stan Lo whether he knew about .cover? or not. :)

#4 - 12/21/2019 03:05 PM - st0012 (Stan Lo)

To: wishdev (John Higgins)

> (and in fact on my system - the to_time option returns false instead of true).

Sorry that I accidentally tested my code in a Rails console instead of pure irb. The result for that case should be false on my machine as well.

Let me correct this:

```
may1 = Date.parse("2019-05-01")
may3 = Date.parse("2019-05-03")
noon_of_may3 = DateTime.parse("2019-05-03 12:00")
may31 = Date.parse("2019-05-31")

(may1..may31).include? may3 # => True
(may1..may31).include? may3.to_time # => False
(may1..may31).include? may3.to_datetime # => True
(may1..may31).include? noon_of_may3 # => False
```

To: zverok (Victor Shepelev) and shevegen (Robert A. Heiler)

I think semantically, cover might be a better API for such cases. But I'm like shevegen don't use cover that often. In fact, I completely forgot about it!

However, I think my question of this issue is:

Does a Date range represent a series of individual days between May 1st and May 31th, like [2019-05-01 00:00:00, 2019-05-02 00:00:00..... 2019-05-31 00:00:00] ? Or it represents a continuous time range that starts from May 1st's 00:00 to May 31th's 00:00?

If it's the first case, I can understand that include? doesn't return true for noon_of_may3. Because it's not at 00:00:00 of that day. But at the same time, I think it should return true for (may1..may31).include? may3.to_time as well because it's at 00:00:00 of that day.

```
may3.to_time #=> 2019-05-03 00:00:00 +0000
```

If it's the second case, we should make all 4 cases return true because they're all covered by the range.

What do you guys think?

#5 - 12/21/2019 03:25 PM - zverok (Victor Shepelev)

> Does a Date range represent a series of individual days between May 1st and May 31th, like [2019-05-01 00:00:00, 2019-05-02 00:00:00..... 2019-05-31 00:00:00] ? Or it represents a continuous time range that starts from May 1st's 00:00 to May 31th's 00:00?

It (Range in general, nothing special about date Range) represents both, depending on the context.

- In Enumerable context (for example, if you'll try to do (may1..may31).to_a, or .select or .any?; or include?) it represents a series.
- In the diapason context (cover?, ===) it represents the entire space between beginning and end.

That's true for every kind of range, and even if not entirely obvious always, is easy to explain and remember without edge cases (which the linked Rails PR tries to introduce: "if you don't know include? is discontinous, we got you covered, bro!").

> I think it should return true for (may1..may31).include? may3.to_time.

The reason it doesn't is not related to the Range itself, but to the fact that Date is library class and Time is core class, and they are not compatible. This is also false:

```
may3 == may3.to_time
# => false
```

I hate this fact myself and tried to argue about it (that we need core Date class), but Powers That Be think about date as "scientific" dates library rarely needed, while Rails team and Rails users used to think about it as a generic "just date" class.

#6 - 12/29/2019 07:10 PM - jeremyevans0 (Jeremy Evans)

- Status changed from Open to Rejected

As explained in some previous comments, if you want to check if a value is on or after the beginning of the range and on or before the end of the range, use cover?. include? should only be used if you want to check the argument is one of the members of the range (i.e. included in the array returned by to_a).