Win-Vector Blog

The Win-Vector LLC data science blog

Prefer = for assignment in R

April 23, 2013 John Mount Mathematics, Programming assignment, R, style

We share our opinion that = should be preferred to the more standard <- for assignment in R. This is from a draft of the appendix of our upcoming book. This has the risk of becoming an R version of Javascript's semicolon controversy, but here you have it.

R has five common assignment operators: "=", "<-", "->", "<<-" and "->>". Traditionally in R <- is the preferred assignment operator and = is thought as an amateurish alias for it.

The <- notation is preferred by some for the very good reason that <- always means assignment. Whereas = can mean assignment, function argument binding or case statement depending on context. However, in our opinion, you are allowed by R to type <- too many places (such as inside expressions) and it usually an easier to find bug when you typed = when you meant <- than the other way around.

We prefer to get into the habit of never typing <-, because accidentally typing <- instead of = in a function call can cause a non-reported error. Consider the following code fragment demonstrating how we can use = to bind values to function arguments:

```
> divide = function(numerator,denominator) { numerator/denominator }
> divide(1,2)
[1] 0.5
> divide(2,1)
[1] 2
> divide(denominator=2,numerator=1)
[1] 0.5
```

Now consider the following (deliberate) error, by habit we typed <- instead of =:

```
> divide(denominator<-2,numerator<-1)
[1] 2
> denominator
[1] 2
```

We quietly get the wrong answer and contaminate the values of `numerator` and `denominator` in the global name space. This is a simple example of where typing <- where = was intended causes a non-signaling bug. We don't know of any simple example (other than building examples that intend side-effects) where typing = where you meant <- is an error. So we prefer =.

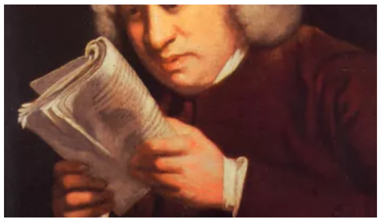
The -> operator is just a right to left assignment that lets you write things like `x -> 5`. It is cute, but not game changing. The <<- and ->> are to be avoided unless you actually need their special abilities. They undo one of the important safety point about functions. When a variable is assigned inside a function this assignment is local to the function. That is nobody outside of the function every sees the effect, the function can safely use variables to store intermediate calculations without clobbering same-named outside variables. The <<- and ->> operators are the operators to reach outside of this protected scope and cause outside side effects. Side effects seem great when you need them, but on the balance they make code maintenance, debugging and documentation much harder.

Edit 5/23/2016: Frankly I still believe there is good evidence that the currently observed semantics of R as it is implemented make "=" the better choice. The "<-" operator currently does not preserve differences that really improve readability or code safety. Obviously it may have been different in the past (before some language changes and when interoperating with S-PLUS was critical. However, we no longer encourage students to use "=" as we can't in good conscience risk wasting student's time on a dealing with the inevitable bickering and backlash. I have enough experience to try and make my case (right or wrong), a newer R user will not be able to respond to a number of the standard criticisms.

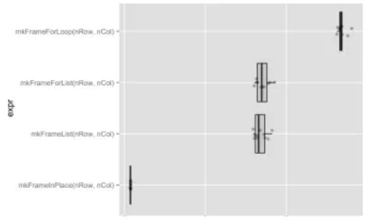
SHARE THIS:

9 2

RELATED



An R function return and assignment puzzle
In "Programming"



Efficient accumulation in R
In "Coding"

R annoyances
Readers returning to our blog will know that Win-Vector LLC is fairly "pro-R." You can take that to mean "in favor or R" or "professionally using R" (both
In "Coding"

April 23, 2013 John Mount Mathematics, Programming assignment, R, style

20 thoughts on “Prefer = for assignment in R”

Johan Viklund

April 24, 2013 at 1:03 am

(WordPress ate parts of my comment, hopefully this works, you can remove my the previous one)

I have an example of using = instead of <- (from my own code):

> system.time(result <- long_running_op())

Now, I don’t know if this is a significant example enough. I’ve found it useful to be able to do this sometimes. Now, this can of course be expressed like this instead (given that <<- is also disallowed):

> system.time((function() res <<- long_running_op())())

I don’t think this is a very strong case, but it is sometimes convenient to be able to use <- inside statements. I would like this to be cleared up though, because it is potentially confusing. An alternative would be to disallow = to mean assignment anywhere (I don’t view argument binding as assignment). But your suggestion is better (and more consistent with other programming languages).

My pet peeve in R is the inconsistent usage of humpback and dot-separation for argument names, read.table() is a perfect example of this, “blank.lines.skip” vs “stringsAsFactors”. Even if I know the correct words of the argument I never remember how to join them. Would be nice if someone could go over at least the standard library and canonize this (at least make aliases from one to the other as a first step).

And finally thanks for a very good blog. I really appreciate it.

---

jmount

April 24, 2013 at 6:11 am

@Johan Viklund Johan, sorry about any problems our crude commenting system gave you. Your two examples are interesting and worth thinking about. In both cases you are setting up intentional side-effects to get a result you want out of a wrapping context. system.time() is but one good example and it got me to thinking about the importance of a wrapping context (like a finally on a try/catch block in Java parlance). There may be more ideas along those lines. But overall I agree- I wish R had some notations that pretty much always meant the same thing and (to my point) were always a syntax error when used in the wrong context.

Jim

April 28, 2013 at 10:17 am

I completely disagree. You're example CLEARLY shows that <- is for assignment ONLY, and that equals is sometimes something else.

Francis Smart

April 28, 2013 at 10:34 am

I use = signs exclusively on my blog posts (www.EconometricsbySimulation.com). This is primarily because I don't use any extra tools tell blogger that my R code is not HTML code which results in the occasional unforeseen mangling of my code.

Good points. Though I would make sure to include the function "assign" in your list of techniques.

By the way, do you know how to assign values to a subset of an original vector with a variable reference command? For instance:
> v=1:10
> get("v")[3]=99

Does not work.
> assign("v[3]", 99)
> v[3]
Does not work either.

Just a question that I figure everybody knows but I did not get the memo.

Fr.

April 28, 2013 at 12:03 pm

I'd like to be on your side, but check this Gist that downloads a DTA file and saves it as a CSV file: if you replace "<-" by "=" at line 4, the `dta` object is not created and the code will break.

Fr.

April 28, 2013 at 12:04 pm

Note: I forgot to mention the Gist depends on the `downloader` and `foreign` libraries, although the problem emerges with the base `file.exists()` function.

Alex Foss

April 28, 2013 at 1:13 pm

[Apologies for double post, part of my comment got snipped, not sure why]

Interesting post, and good examples. The last year or so of using R I have switched from <- to = for two main reasons: (1) Ease of sharing code with python/matlab/etc users who are not R junkies, and (2) this example:

foo<-3

Of course, formatting code well avoids this problem, but did the author mean to compare foo to -3, or to assign 3 to foo? If my rule is always use =, then I never have to infer from context.

I agree that the special features of <- and <<- are not useful, and in fact may lead to bad/dangerous programming habits.

Alex Foss

April 28, 2013 at 1:15 pm

Apologies. Two of my comments seem to have been partially deleted. I don't know what's going on, so please just delete all three of mine.

---

jmount

April 28, 2013 at 4:19 pm

@Alex Foss

Alex, your second comment looks good to me. Sorry the stupid comment software is cutting things (I have no good idea why it is, other than the need to escape html entities like R's traditional right to left assignment looking like an un-closed HTML/XML comment).

---

jmount

April 28, 2013 at 4:23 pm

@Jim

It is a matter of opinion (and I know i have a minority opinion). Yes, arrow assignment always means assignment (which is very desirable). But in my opinion R loses most of the advantage by allowing assignment in too many contexts (like in function arguments, this is likely a version of C's idea that the expression A=B has a type other than void).

So my point is to try and use a syntax that makes more errors signaling (syntax errors and observable exceptions) and fewer sneaking errors (wrong values to arguments and unexpected side-effects). You always want to be far (more than one mistake away) from hard to find errors, even if this moves you closer to cheap to find errors.

This, of course, may not change your mind. Coding has a lot of aspects of taste and a lot of assumptions that we may not share. It just happens this is the way I think about it. You may find this is an ugly and futile precaution.

I am not trying so much to convince, but push people to see the issue.

---

jmount

April 28, 2013 at 4:32 pm

The style of coding I am pushing may seem very unnatural to some. It is very similar to what is called "Yoda conditions" in the C and Javascript worlds. The idea there is to check if a variable is equal to a value you write

if(5==var) { … }

instead of

if(var==5) { … }

The idea is if you accidentally type assignment equals (=) instead of test equals first is a syntax error and the second would clobber the var (unless you declared it final) skip the test (as non-zero can stand in for true in many C derived languages, Java fortunately would have a type error in this case). So to mess up you have to make two mistakes: forget to use the Yoda form, and type the wrong operator.

Though as an aside, I think in some of the truly ancient dialects of Fortran statements like if(5=var) { … } were disasters as they would over-write the the value of 5 in the constant pool (meaning anywhere else you typed a literal 5 in your program might get a new value).

I happen to like these ideas, some people do not. My article http://www.win-vector.com/blog/2012/02/why-i-dont-like-dynamic-typing/ demonstrates some of my hairshirt taste in programming.

---

jmount

April 28, 2013 at 5:06 pm

Thought I would add one of my old favorites: "x < -2" versus "x <-2". I actually grew up on Pascal's ":=" assignment operator (and liked it a lot). I do know that "=" is a late addition to R. But in addition to looking at language intent and history we have to empirically look at the language in front of us. "=" is now in R, and I happen to think it offers a slight advantage. It should be an empirically testable statement which assignment operator is empirically safer in the R language as it now exists (and as it is likely to continue). If you assume no mistakes, then there is no difference in safety (you are completely safe always). If you assume you may sometime type one thing when you mean another, then it is fair to ask what will show up sooner. Appealing to the style guides is a good point (as a lot of thought presumably went into them, and I agree with most of the advice in the Google style guide). Saying you see the issue and but still don't like = is perfectly valid. Saying "=" was added to appease script kiddies is a bit less interesting of an argument. I don't think there is a case where = as an assignment operator has different semantics than <- as an assignment operator. It is just somethings = is not an assignment operator (like when in a function argument context). If this position is sufficiently unpopular I will definitely take the advice out of our book. No sense pointing others in a non-productive direction.

---

taylor

April 28, 2013 at 5:16 pm

good point, but I doubt I'll change from <- to =. I doubt I will ever use <- in a function call. Google's style guides also recommend using <- : http://google-styleguide.googlecode.com/svn/trunk/google-r-style.html

---

jmount

April 28, 2013 at 5:28 pm

@Francis Smart

I confess I don't see how to do it (other than maybe calling parse/eval on tex).
I thought '[< -'(get('v'),2,15) might do it, but it does not (idea from skimming https://stat.ethz.ch/pipermail/r-help/2010-October/255326.html). It looks like you can work with named slots, but I don't know how to do this with indices.

---

jmount

April 28, 2013 at 5:38 pm

@Fr.

Yes, the <- is needed if you want to have an assignment escape out of a function argument context. When I say use "=" I don't mean run a regexp and replace <- with = everywhere (which is in fact wrong wrong wrong). More like when writing consider:

dta = "data/qog-cs.dta"

if(!file.exists(dta)) …

instead of

if(!file.exists(dta <- "data/qog-cs.dta")) ... (though this does kill some important while() loop idioms!).

Ricardo

April 28, 2013 at 10:39 pm

@Fr.
I think it’s not a good practice to do assingment inside function calls.

I’d replace line 4 by:

dta = “data/qog-cs.dta”
if(!file.exists(dta)) {

---

al3xa

April 29, 2013 at 3:03 am

http://cdn.memegenerator.net/instances/400x/37363149.jpg

---

jmount

April 29, 2013 at 6:34 am

So what are the opinions on [] versus [[]]? http://www.win-vector.com/blog/2012/06/selection-in-r/

---

jmount

April 30, 2013 at 2:28 pm

Thought about it more. The facts are: “=” is unpopular as assignment, so if you don’t have your own personal reason you should use “<-". So it would be wrong to teach beginners to use "=". I'll continue to share my opinion, but it makes sense to teach what is standard unless unless there is a large advantage in the non-standard. And there certainly is not a large advantage. Thanks all for the steering.

Fr.

May 4, 2013 at 3:58 pm

@jmount and @Ricardo :

Thanks for the replies. Any chance that assigning in a function call is a bad practice precisely because of the discrepancy I point to? I find it rather efficient at compacting the code at no cost of legibility. I might be wrong here, details welcome.

I have another point to make because I just finished teaching an undergraduate course with R and students are absolute beginners with social science backgrounds. I clearly remember the students sighing in relief when I mentioned = could also assign, because they really hated <-. We talked a bit about it, and the students explained that they would have chosen = for assignment if they had been writing the R syntax themselves. The <- approach sounded oddly backwards to them (which is a nice intuition).

So my complete position on the issue is actually:

1) there is one situation that I often find myself in where = does not work for assignment
2) but aside from that, I would actually very much recommend John's position!

Comments are closed.