Ordering ggplot2 bars with flipped coordinates using Rpy2 in Python

I have an R dataframe object in Python, using Rpy2, that looks like this:

```
cat name num
1    a   bob    1
2    b   bob    2
3    a mary    3
4    b mary    4
```

I want to plot it as a `geom_bar` with flipped coords, and have the order of the bars match the order of the legend (without changing the legend). This has been asked on here and other mailing lists, but I am stuck on how to translate the R ordering code to rpy2. This is my current code:

```
# attempting to reorder bars so that value 'a' of cat is first (in red)
# and value 'b' of cat is second (in green)
labels = tuple(["a", "b"])
labels = robj.StrVector(labels)
variable_i = r_df.names.index("cat")
r_df[variable_i] = robj.FactorVector(r_df[variable_i],
                                     levels=labels)
r.pdf("test.pdf"))
p = ggplot2.ggplot(r_df) + \
    ggplot2.geom_bar(aes_string(x="name",
                                y="num",
                                fill="cat"),
                     position=ggplot2.position_dodge(width=0.5)) + \
    ggplot2.coord_flip()
p.plot()
```

this gives a graph that has the correct legend ('a' is first in red, 'b' is second in green) but the bars appear in the order green/red ('b', 'a') for each value of 'name'. my impression from prev. posts is that this is because `coord_flip()` flips the order. how can I change the *bars order* (not legend order) to match the current legend, plotting red bars first in each dodged bar group? thanks.

edit:*

tried @joran's solution in rpy2:

```
labels = tuple(["a", "b"])
labels = robj.StrVector(labels[::-1])
```

```
variable_i = r_df.names.index("cat")
r_df[variable_i] = robj.FactorVector(r_df[variable_i],
                                     levels=labels)
p = ggplot2.ggplot(r_df) + \
    ggplot2.geom_bar(aes_string(x="name",
                                y="num",
                                fill="cat"),
                     stat="identity",
                     position=ggplot2.position_dodge()) + \
    ggplot2.scale_fill_manual(values = np.array(['blue','red'])) + \
    ggplot2.guides(fill=ggplot2.guide_legend(reverse=True)) + \
    ggplot2.coord_flip()
p.plot()
```

this doesn't work because `ggplot2.guides` is not found, but that's only for setting the legend. My question is: why is the `scale_fill_manual` necessary? I don't want to specify my own colors, I want the default ggplot2 colors, but I just want to set the ordering to be a particular way. It makes sense that with `coord_flip` I'd need a reverse ordering but that is not enough to reorder the bars apparently (see my `labels[::-1]`) independently of the legend. thoughts on this?

python r ggplot2 rpy2

edited Jul 15 '13 at 3:31

asked Jul 14 '13 at 22:35

user248237dfsf
15.4k 74 223 339

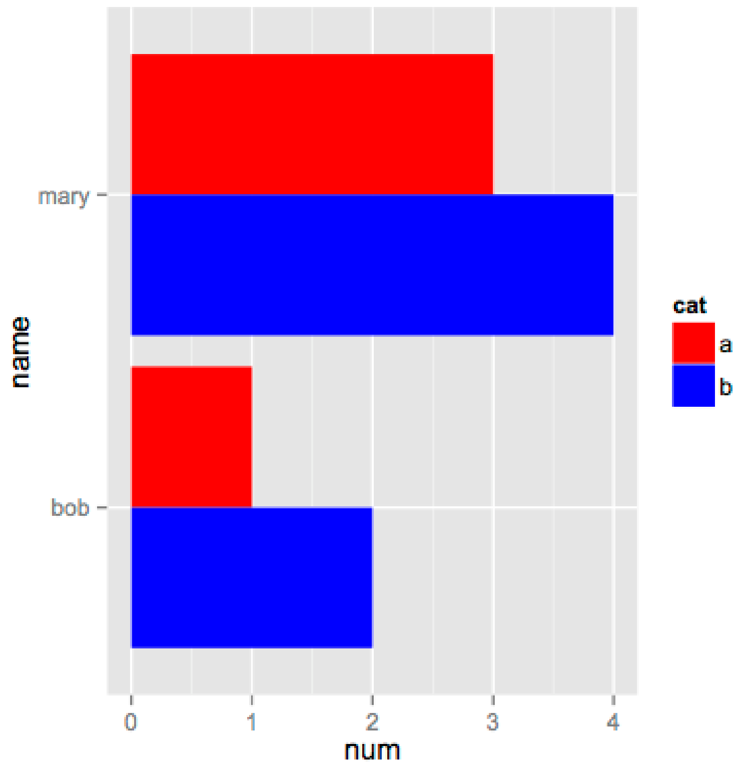
1 ggplot2's `guides` is not custom-mapped by the module `rpy2.robjects.lib.ggplot2` (`guides` is relatively recent, I missed its introduction and no one complained until now). In the meantime, try with just modifying your code with `ggplot2.ggplot2.guides()` . – lgautier Jul 16 '13 at 8:06

1 Answer

I apologize I do not have rpy setup on my machine, but I was able to get what I think you are describing with this R/ggplot2 code:

```
dat$cat <- factor(dat$cat,levels = c('b','a'))

ggplot(dat,aes(x = name,y = num, fill = cat)) +
    geom_bar(stat = "identity",position = "dodge") +
    coord_flip() +
    scale_fill_manual(values = c('blue','red')) +
    guides(fill = guide_legend(reverse = TRUE))
```



Basically, I just reversed the factor levels, and reversed the legend order. Convoluted, but it looks like what you described. (Also, you did want `stat = "identity"` , right?)

answered Jul 15 '13 at 2:03
joran
106k 13 235 284

@jordan: thank you - see my edit for response – user248237dfsf Jul 15 '13 at 3:31

@user248237dfsf The only reason `guides` would throw an error like that is if (a) you have a rather old version of ggplot installed, or (b) somehow rpy is behaving badly. I only used `scale_manual` because otherwise you get a and b mapped to the default colors in the reverse order; displayed how you want them, but with the 1st default color mapped to b. If you don't care about that, omit it. – joran Jul 15 '13 at 3:58

@jordan: so why isn't it enough to do `dat$cat <- factor(dat$cat,levels = c('b','a'))` to reverse the bars? at least the rpy2 equivalent does not do that, it needs the `scale_fill_manual` . about colors: can I reverse them without explicitly stating them? I want ggplot to pick the colors without me inputting hex colors or color names – user248237dfsf Jul 15 '13 at 4:08

@user248237dfsf My name is joran. Reordering the levels *does* reorder the bars. But as I said before, the order in which ggplot assigns its default colors remains the same. So b gets the first default color and a gets the second one. – joran Jul 15 '13 at 4:38

sorry the typo! usually don't mess up names. I think I understand what ggplot is doing now but still cannot translate your solution to rpy2 – user248237dfsf Jul 15 '13 at 15:28