R Workshop week 3: making graphs part 2

Teal Potter

9/10/2021

Table of Contents

Setup 1
Renaming groups in a graph 2
RENAME groups by overwriting names in data frame 2
RENAME groups by overwriting names in data frame 2
RENAME groups by adding column with new names in data frame 2
RENAME groups in graph code (not in data frame) 3
REORDER groups in graph 4
REORDER groups in legend 5
3 ways to calculate standard error 6
Option 1 6
Use standard error function in a package 6
Calculate standard error the tidyverse way 6
Write your own standard error function 8
Example satellite plot 9
Example text to add around graphed area 10
A way to save high quality graphs 10

Setup

```
library(tidyverse)  # tidyverse is a combination of packages including ggplot2
library(car)

head(Soils) # our example dataset
```

```
##   Group Contour Depth Gp Block   pH     N Dens   P    Ca   Mg    K   Na Conduc
## 1     1     Top  0-10 T0     1 5.40 0.188 0.92 215 16.35 7.65 0.72 1.14   1.09
## 2     1     Top  0-10 T0     2 5.65 0.165 1.04 208 12.25 5.15 0.71 0.94   1.35
## 3     1     Top  0-10 T0     3 5.14 0.260 0.95 300 13.02 5.68 0.68 0.60   1.41
## 4     1     Top  0-10 T0     4 5.14 0.169 1.10 248 11.92 7.88 1.09 1.01   1.64
## 5     2     Top 10-30 T1     1 5.14 0.164 1.12 174 14.17 8.12 0.70 2.17   1.85
## 6     2     Top 10-30 T1     2 5.10 0.094 1.22 129  8.55 6.92 0.81 2.67   3.18
```

Renaming groups in a graph

RENAME groups by overwriting names in data frame

```
# Check how many names to replace
class(Soils$Depth)

## [1] "factor"

levels(Soils$Depth)

## [1] "0-10"  "10-30" "30-60" "60-90"
```

RENAME groups by overwriting names in data frame

```
soils <- Soils #saving pre-loaded Soils dataset to new object called 'soils' that I can manipulate

levels(soils$Depth) <- c("Top", "Mid", "Deep", "Deepest")
head(soils)[1:5]

##   Group Contour Depth Gp Block
## 1     1     Top   Top T0     1
## 2     1     Top   Top T0     2
## 3     1     Top   Top T0     3
## 4     1     Top   Top T0     4
## 5     2     Top   Mid T1     1
## 6     2     Top   Mid T1     2

soils <- Soils  # resetting dataset to use original Depth column names
```

RENAME groups by adding column with new names in data frame

Use dplyr::mutate to make a new column in data frame that specifies order of groups from a different column. Note: the package 'dplyr' is loaded with tidyverse

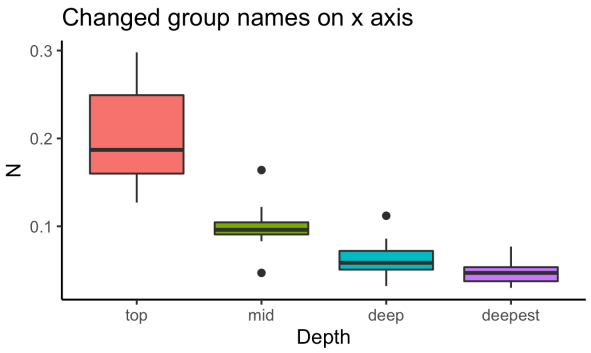
```
soils <- mutate(soils, Depth_labels = fct_recode(
          Depth,   # New column name will be "Depth_labels"
```

```
              "TOP" = "0-10",    # Note order: "new name" = "orginal name"
              "MID" =  "10-30",
              "DEEP" = "30-60",
              "DEEPEST" = "60-90"))


ncol(soils)  # Since new columns are added to the end of the dataset (right side),
I use ncol to see total number of columns

## [1] 15

head(soils)[10:15] # looking at to rows of columns 10:15

##      Ca   Mg    K   Na Conduc Depth_labels
## 1 16.35 7.65 0.72 1.14   1.09          TOP
## 2 12.25 5.15 0.71 0.94   1.35          TOP
## 3 13.02 5.68 0.68 0.60   1.41          TOP
## 4 11.92 7.88 1.09 1.01   1.64          TOP
## 5 14.17 8.12 0.70 2.17   1.85          MID
## 6  8.55 6.92 0.81 2.67   3.18          MID
```

RENAME groups in graph code (not in data frame)

Rename x axis labels in ggplot code with scale_x_discreet(labels = …)

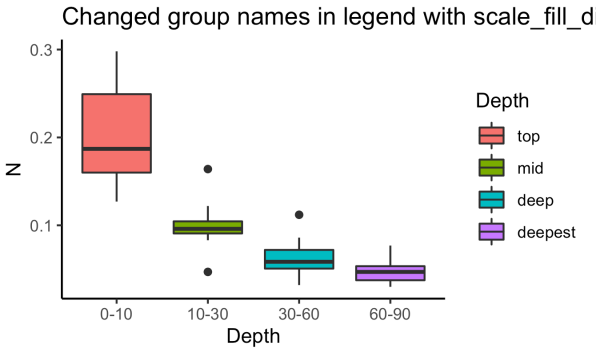
```
ggplot(Soils, aes(x = Depth, y = N, fill = Depth))+      # note: if using
scale_fill_discrete use fill() here too
  geom_boxplot()+
  theme_classic()+
  theme(legend.position = "none")+ # removing legend
  labs(title = "Changed group names on x axis")+
  scale_x_discrete(labels = c("top", "mid", "deep", "deepest"))
```



Rename labels in lengend using scale_fill_discrete(labels = …)

```
ggplot(Soils, aes(x = Depth, y = N, fill = Depth))+      # note: if using
scale_fill_discrete use fill() here too
  geom_boxplot()+
```

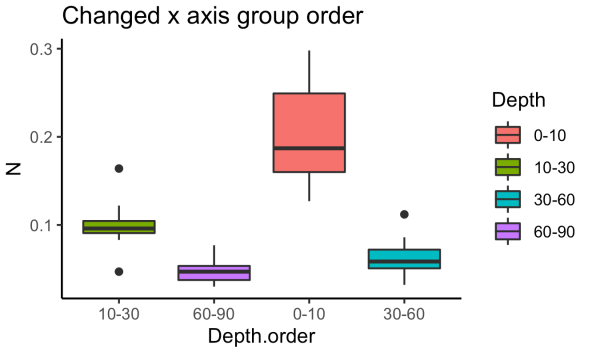
```
  theme_classic()+
  labs(title = "Changed group names in legend with scale_fill_discrete")+
  scale_fill_discrete(labels = c("top", "mid", "deep", "deepest"))   # make sure to
provide correct # of names
```



REORDER groups in graph

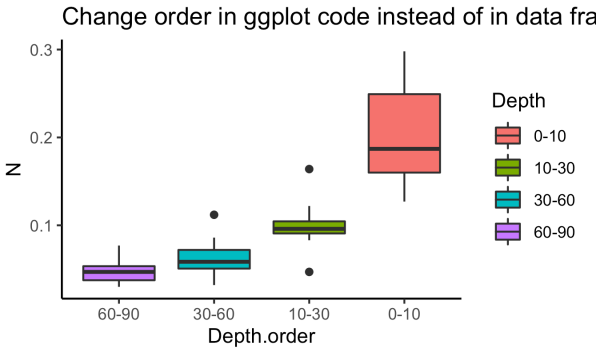
Note: reordering a gradient color palette for continuous variables will not be included in this document. I haven't found an easy way to do it yet!

```
soils <- Soils %>%
  mutate(Depth.order = factor(Depth, levels = c("10-30", "60-90", "0-10","30-60")))
# must use existing level names in the column


ggplot(soils, aes(x = Depth.order, y = N, fill = Depth))+  # using new column;
Depth.order
  geom_boxplot()+
  theme_classic()+
  labs(title = "Changed x axis group order")
```



Alternatively, use scale_x_discrete() in the ggplot code to specify order without making any changes to your data frame. Both options are commonly used.

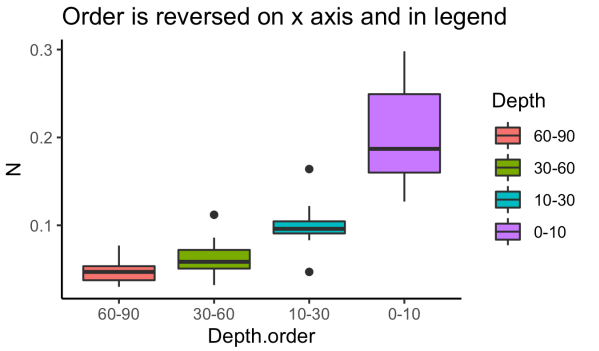
```
ggplot(soils, aes(x = Depth.order, y = N, fill = Depth))+  # using new column; Depth.order
  geom_boxplot()+
  theme_classic()+
  labs(title = "Change order in ggplot code instead of in data frame")+
  scale_x_discrete(limits = c("60-90", "30-60", "10-30", "0-10"))
```



REORDER groups in legend

Also showing a alternative coding option: specify group order as an object and referencing in ggplot code

```
order.object <- c("60-90", "30-60", "10-30", "0-10")  # save order in a new object

ggplot(soils, aes(x = Depth.order, y = N, fill = Depth))+  # using new column; Depth.order
  geom_boxplot()+
  theme_classic()+
  labs(title = "Order is reversed on x axis and in legend")+
   scale_x_discrete(limits = order.object)+  # keeping x axis order from previous graph, but referencing new object this time
  scale_fill_discrete(limits = order.object) # this code changes legend labels order
```



3 ways to calculate standard error

It is good practice to include a measure of variance in your graph if you are reporting a measure of centeral tendency (e.g. mean or median) of groups as opposed to showing all the data points.

Strangely enough, neither base R nor ggplot2 have included a function to calculate standard error to make error bars.

The following examples show 3 ways you can do this demonstrated with the `Soils` dataset.

Option 1

Use standard error function in a package

Here is how you calculate standard error on an entire column (Soils$pH). You can also use this function in option 2 to calculate standard error for groups say pH for every soil depth.

```
#install.packages(plotrix)   # I have plotrix downloaded already so I added '#' to make this line of code a comment so I don't run this line again.

library(plotrix)   #plotrix::std.error

std.error(Soils$pH)  # standard error for the mean of all pH values in Soils dataset

## [1] 0.0969739
```

Option 2

Calculate standard error the tidyverse way

Standard error for the mean of all pH values in Soils dataset

```
sd(Soils$pH / sqrt(length(Soils$pH)))

## [1] 0.0969739
```

Standard error of the means is calculated on pH for each soil depth.

```
grouped <- group_by(Soils, Depth)                          #tidyverse::group_by()

summarise(grouped, se = sd(pH/sqrt(length(pH))),           #tidyverse::summarise()
          means = mean(pH))
```

```
##   Depth     se means
## 1 0-10  0.0718  5.40
## 2 10-30 0.172   5.00
## 3 30-60 0.0954  4.28
## 4 60-90 0.0587  4.00
```

Same code with piping

Piping means using %>% to connect lines of code instead of assigning each line of code to a new object and referring to that object in the next line.

```
Soils %>%
  group_by(Depth) %>%
  summarise(se = sd(pH/sqrt(length(pH))))
```

```
##   Depth     se
## 1 0-10  0.0718
## 2 10-30 0.172
## 3 30-60 0.0954
## 4 60-90 0.0587
```

```
#Note: this will not work if you have any NAs or non-numeric data in your pH
column. Fixed below.

se.table <- Soils %>%
  group_by(Depth) %>%
  summarise(
    se = sd(pH, na.rm = TRUE)/sqrt(length(na.omit(pH))),
    means = mean(pH))

se.table
```

```
##   Depth     se means
## 1 0-10  0.0718  5.40
## 2 10-30 0.172   5.00
## 3 30-60 0.0954  4.28
## 4 60-90 0.0587  4.00
```

Option 1 and 2 together

You can use the tidyverse way and the premade plotrix::std.error function to calculate standard error for groups in a column with summarise_each()…

```
Soils %>%
  group_by(Depth) %>%
  summarise(
    pH.se = std.error(pH))
```

```
##   Depth  pH.se
## 1 0-10  0.0718
## 2 10-30 0.172
## 3 30-60 0.0954
## 4 60-90 0.0587
```

…or for multiple columns with summarise_at()

```
Soils %>%
  group_by(Depth) %>%
    summarise_at(c("pH", "N"), std.error)
```

```
##   Depth     pH       N
## 1 0-10  0.0718 0.0159
## 2 10-30 0.172  0.00785
## 3 30-60 0.0954 0.00605
## 4 60-90 0.0587 0.00372
```

Option 3

Write your own standard error function

Note: a good way to learn how to write functions is by doing a self led exercise in your RStudio console using the swirl package.

Run this code to get started: `install.packages("swirl")`

```
library(swirl)
```

```
SE <- function(x) sd(x, na.rm = TRUE)/sqrt(length(na.omit(x)))
```

Now you can put your a vector in your function as an argument and run the code

```
SE(Soils$pH)
```

```
## [1] 0.0969739
```

Writing a function to calculate standard error for multiple groups gets complicated quickly

```
se_by_groups <- function(x, num_var){
  num_var <- enquo(num_var)

  x %>%   # this bit of code defines what x is in the function
    summarize(avg = mean(!!num_var),
              n = n(),
              sd = sd(!!num_var),
```

```
              se = sd/sqrt(n))
}
```

And this is how you would use the se_by_groups function

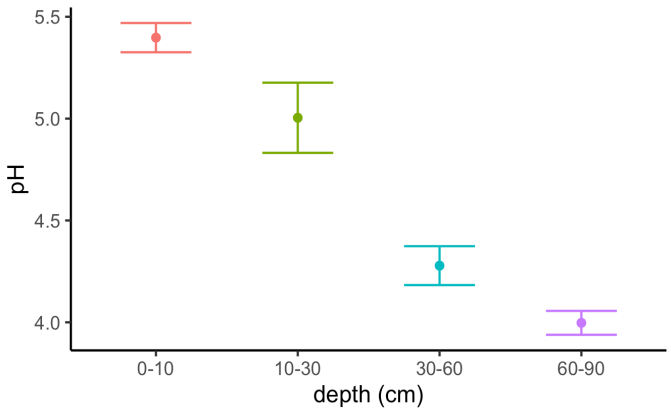
```
Soils %>%
  group_by(Depth) %>%
  se_by_groups(pH)


##   Depth   avg     n    sd     se
## 1 0-10   5.40    12 0.249 0.0718
## 2 10-30  5.00    12 0.597 0.172
## 3 30-60  4.28    12 0.331 0.0954
## 4 60-90  4.00    12 0.203 0.0587
```

Example satellite plot

previously calculated standard error can now be added as error bars with geom_errorbar(). Note: the se.table produced via option 2 are the data used.

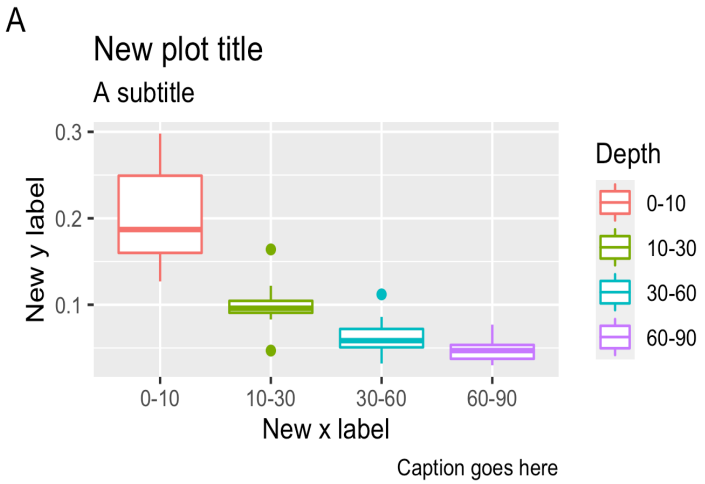
Notice that this function allows you to directly calculate where the error bar should go based on adding and subtracting the standard error from the calculated means.

```
satellite <- ggplot(se.table, aes(x = Depth, y = means, col = Depth))+
  geom_point() +  # data are points
  geom_errorbar(aes(x=Depth, ymax=means + se ,ymin = means - se), width = 0.5)+
  theme_classic()+
  ylab("pH")+
  xlab("depth (cm)") +
  theme(legend.position = "none")

satellite
```



Example text to add around graphed area

```
#boxplot <-
  ggplot(Soils, aes(x = Depth, y = N, col = Depth))+
  geom_boxplot()+
labs(
    title = "New plot title",
    subtitle = "A subtitle",
    x = "New x label",
    y = "New y label",
    fill = "New legend title",
    caption = "Caption goes here",
    tag = "A")
```



A way to save high quality graphs

ggsave() is an easy function to use to save graphs to several file types include .pdf. Specify width and height (default is inches), and the dpi which determines your graph's resolution or quality. Setting dpi to 600 for most online publications (at least 300 for printing) is a good way to ensure a sharp looking image.

```
ggsave(file =
"/Users/firstnamelastname/topfoldername/folderintopfolder/file_name.pdf", fig1,
width = 6.5, height = 6, dpi = 600)
```