FES 524 Winter 2022 Lab 4

Cleaning up results to put in a table

Here is an example of how I might neaten a table up in R that I could take to Excel for final "prettifying". This is what `diff_q2` looked like after we calculated the comparisons of interest.

```
diff_q2
```

```
                     contrast estimate        SE df   lower.CL    upper.CL
1  contain Jan 2 - barert Jan 2   -0.268 0.1266186 20 -0.6166444  0.08064436
2  barert Jan 28 - barert Jan 2   -0.154 0.1266186 20 -0.5026444  0.19464436
3 contain Jan 28 - barert Jan 2    0.108 0.1266186 20 -0.2406444  0.45664436
4  barert Feb 25 - barert Jan 2   -1.180 0.1266186 20 -1.5286444 -0.83135564
5 contain Feb 25 - barert Jan 2   -1.268 0.1266186 20 -1.6166444 -0.91935564
```

And here's the work I'd do to clean things up. This involves general data manipulation such as rounding, pasting columns together, and cleaning up column names.

```
# Here I'll make a table of results of interest to give
    # an idea of the work it takes to get things cleaned up in R,
    # with the estimated differences in means and confidence intervals
diff_q2_tab = diff_q2 %>%
    select(-SE, -df) %>% # keep all columns other than SE, df
    mutate(across(.cols = where(is.numeric),
                  .fns = round,
                  digits = 2) ) # Round numeric columns to two digits

# I am going to use paste() to make a single column for the confidence intervals
# Using sprintf() to force 2 decimal places on Upper.CI
diff_q2_tab = mutate(diff_q2_tab,
                     ci = paste(sprintf("%.2f", lower.CL), upper.CL, sep = ", ") )
# I need to make better names for the comparisons
    # and make column names looks nicer
diff_q2_tab$contrast = c("Jan 2 container minus control",
                         "Jan 28 bare minus control",
                         "Jan 28 container minus control",
                         "Feb 25 bare minus control",
                         "Feb 25 container minus control")
colnames(diff_q2_tab) = c("Comparison", "Difference in mean growth increment (cm)",
                     "Lower", "Upper", "Dunnett-adjusted 95% CI")
```

Here is an example table for the write-up, although it could use some further tweaking on alignment.

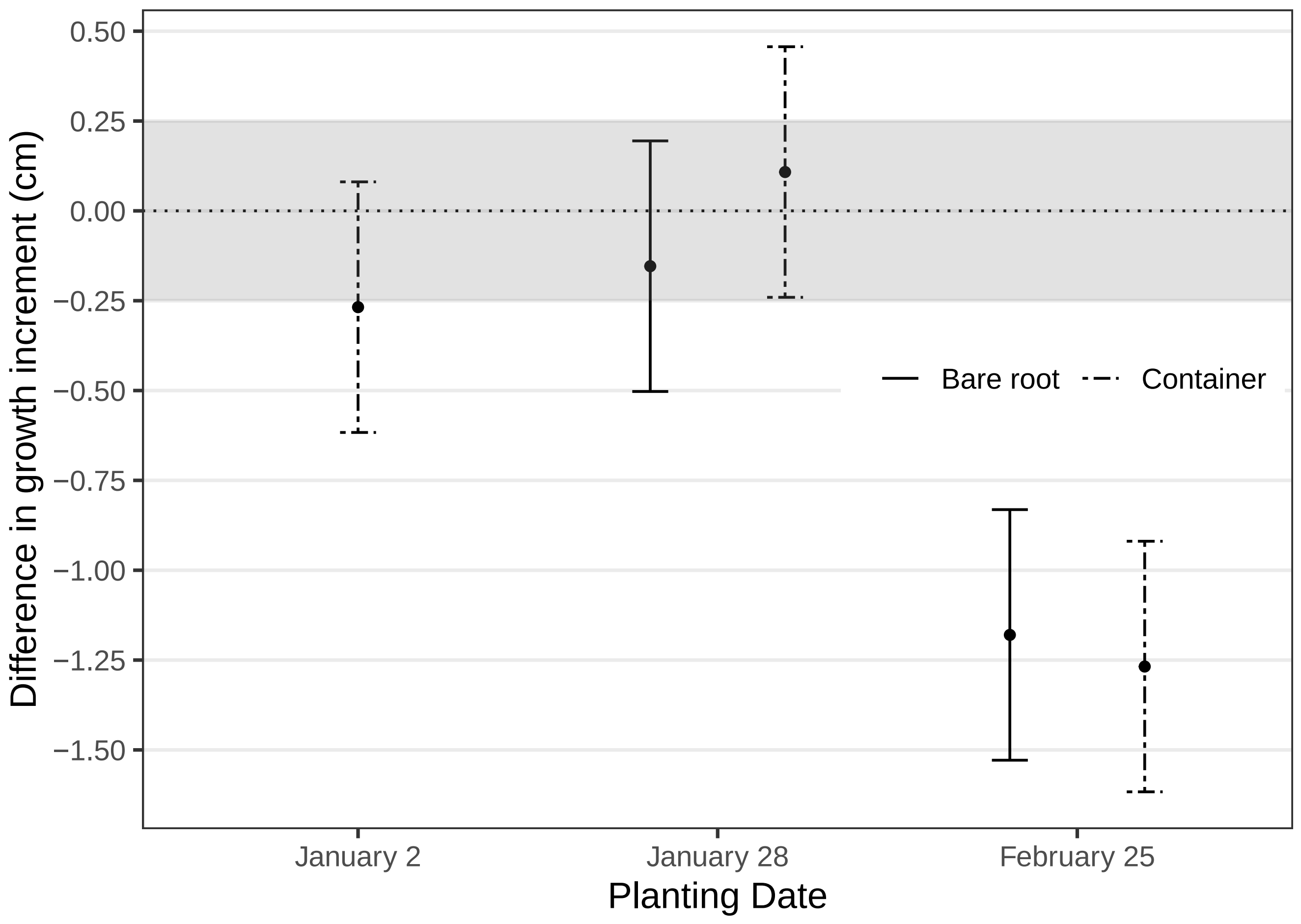
```
select(diff_q2_tab, -Lower, -Upper)
```

Comparison	Difference in mean growth increment (cm)	Dunnett-adjusted 95% CI
Jan 2 container minus control	-0.27	-0.62, 0.08
Jan 28 bare minus control	-0.15	-0.50, 0.19
Jan 28 container minus control	0.11	-0.24, 0.46
Feb 25 bare minus control	-1.18	-1.53, -0.83
Feb 25 container minus control	-1.27	-1.62, -0.92

Bonus graphics

We will create a second graphic showing the raw data and then embed this figure into the one we made in Lab 4. The figures we will embed the new one in is named `g1`.

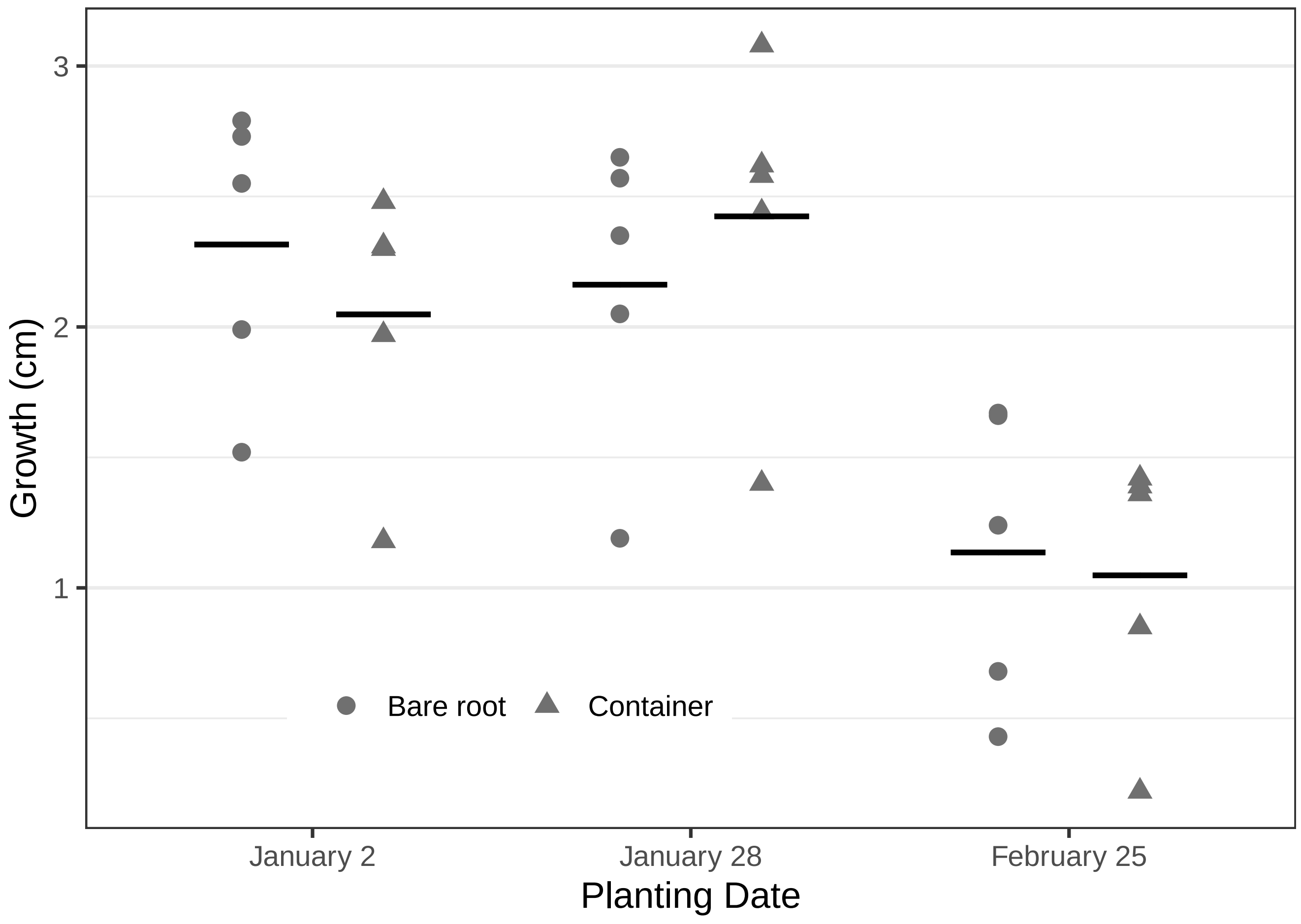
This is what the figure we'll be working with looked like at the end of Lab 4:



We will make a figure representing the raw data with lines for group means as an alternative to showing these data in a table. We will name this graphic `g2`. To add the lines for the means, I created a second dataset named `growthmeans` which I then used for the `geom_errorbar` layer. This is the first time we've seen plotting from two separate datasets inside one graphic. Notice we add the `data` argument to `geom_errorbar` to use the summarize dataset for that layer.

```
# I want to add the means as a line, so I'm going to make a "means" dataset
    # to do that
growthmeans = growthdata %>%
     group_by(plantdate, stock) %>%
     summarise(growth = mean(growth),
               .groups = "drop")

# Notice I change the dataset for the geom_errorbar() layer
( g2 = ggplot(growthdata, aes(x = plantdate, y = growth, group = stock ) ) +
          geom_point( size = 3, colour = "grey44", aes(shape = stock),
                      position = position_dodge(width = .75) ) +
          geom_errorbar(data = growthmeans, aes(ymax = growth, ymin = growth),
                        position = position_dodge(width = .75), width = .5, size = 1) +
          # put in line for means using growthmeans dataset
          theme_bw(base_size = 14) +
          ylab("Growth (cm)") + # Change y label
          scale_x_discrete(name = "Planting Date",
                           labels = c("January 2", "January 28", "February 25")) +
          # Change x axis labels and tick labels
          theme(legend.position = c(.35, .15), # change legend position
                legend.direction = "horizontal", # make legend horiz
                legend.key = element_blank(), # remove boxes around legend symbols
                panel.grid.major.x = element_blank() ) + # remove vertical gridlines
          scale_shape_discrete(name = "", labels = c("Bare root", "Container") ) ) # change legend names
```



We are going to embed the figure showing the raw data and the group means (figure `g2`) into the graph showing the differences in means between the control group and all other groups (figure `g1`).

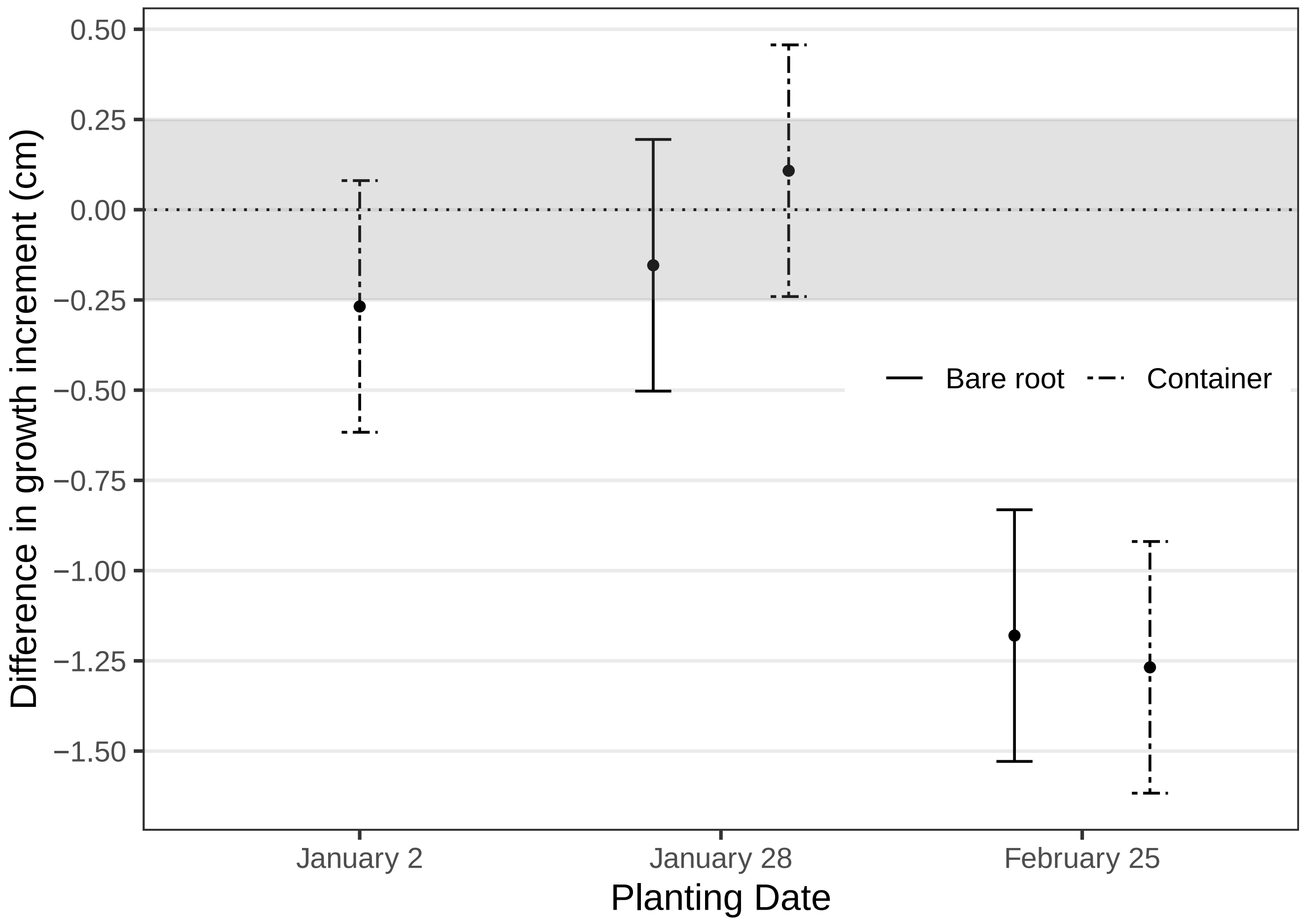
I'll use package **cowplot** for this, a package that has a lot of flexibility for combining plots. You may need to install this if working on your own computer.

```
library(cowplot)
```

Embedding one plot in another in **cowplot** is based on `ggdraw()` followed by `draw_plot()` layers.

Here's what it looks to simply draw the `g1` plot again but using **cowplot** code.

```
ggdraw() +
    draw_plot(g1)
```



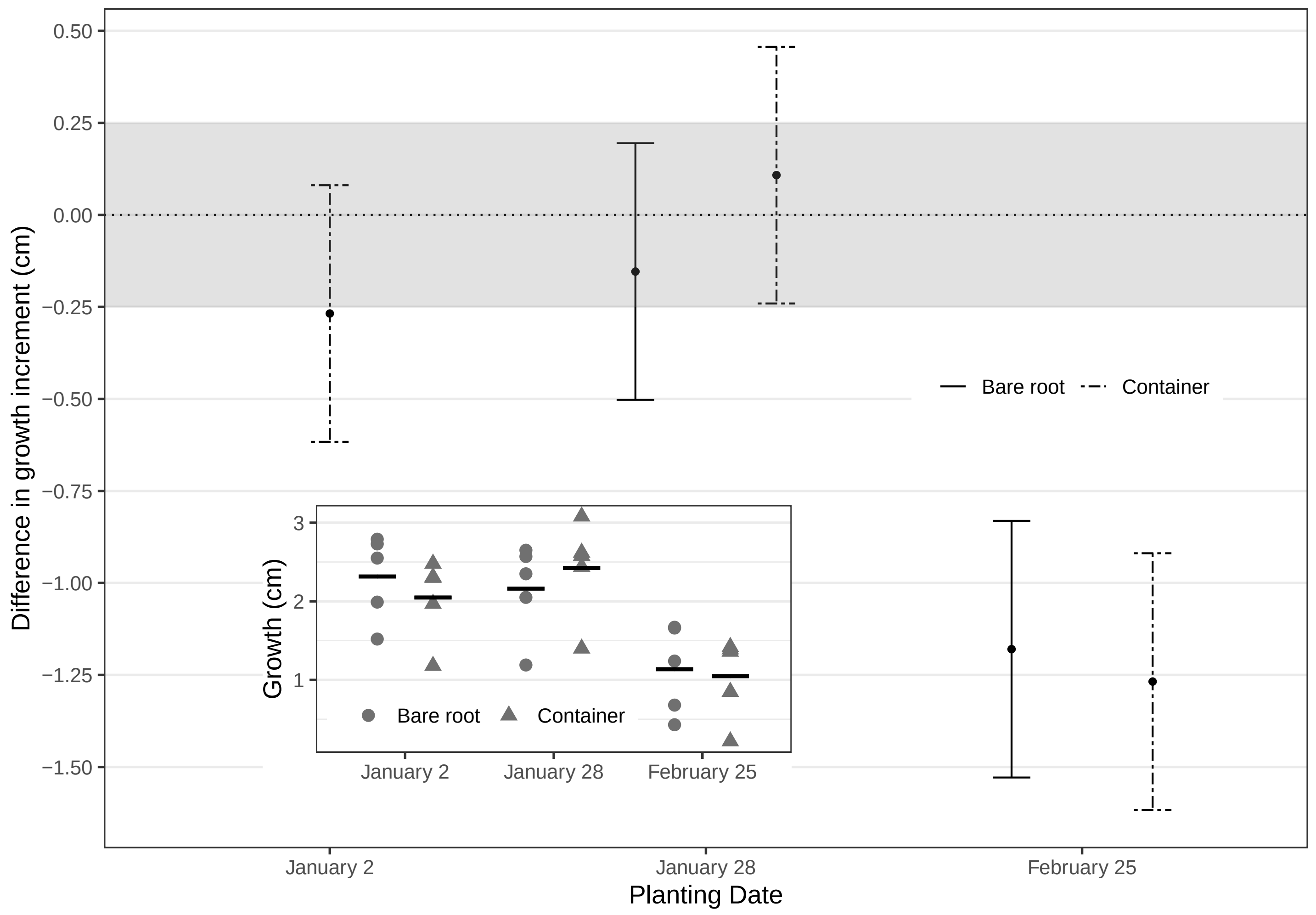
The second graph is embedded in the first by adding another `draw_plot()` layer. Make sure you make your graphics window in RStudio big enough to see the result. I've increased the plot size below to make sure things fit appropriately at the final graphics size I used (10 x 7 inches).

I decided to change some theme elements of the plot, removing the x axis label and making the spacing around the plot narrower, so there is an additional `theme` layer added to `g2`.

In `draw_plot()`, the position of the plot is defined via the `x` and `y` arguments. This is the position of the lower left corner of the plot, and is a number between 0 and 1 for each axis.

The size of the plot is controlled by the `height` and `width` arguments. These also take values between 0 and 1. Note if both are set to 1 (the default), the graph would be as large as the plotting space.

```
ggdraw() +
    draw_plot(g1) +
    draw_plot(g2 + theme(plot.margin = unit(rep(0,4), "lines"),
                         axis.title.x = element_blank() ),
            x = .2, y = .15,
            height = .3, width = .4)
```

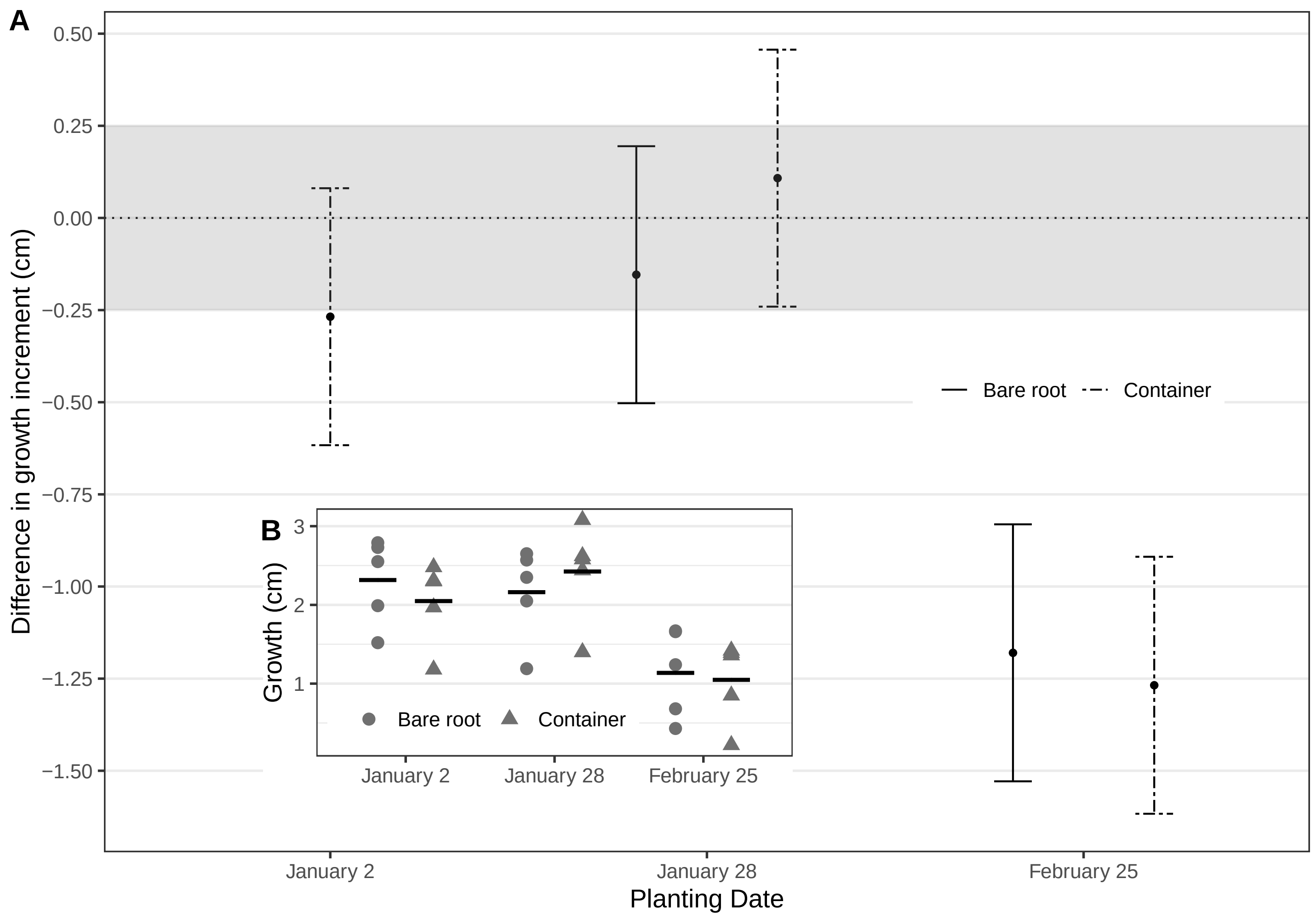


Adding labels to combined plots is a strength of **cowplot**. These can be added to embedded plots via `draw_plot_label()`. It can be tricky getting the placement of the labels just right, although it is based on the `x` and `y` and the `height` and `width` values from the `draw_plot()` layers.

Adding labels would help with writing a clear caption for this graphic.

Here is a final version of my plot. Getting the labels right took some back and forth on my part, working with the plot at the final size.

```
( combined = ggdraw() +
    draw_plot(g1) +
    draw_plot(g2 + theme(plot.margin = unit(rep(0,4), "lines"),
                         axis.title.x = element_blank() ),
              x = .2, y = .15,
              height = .3, width = .4) +
    draw_plot_label(label = c("A", "B"),
                    x = c(0, .19),
                    y = c(1, .45) ) )
```



If you wanted to save the final figure, you can use the **cowplot** version of `ggsave()`.

```
ggsave("combined_plot.pdf", plot = combined, height = 7, width = 10)
```