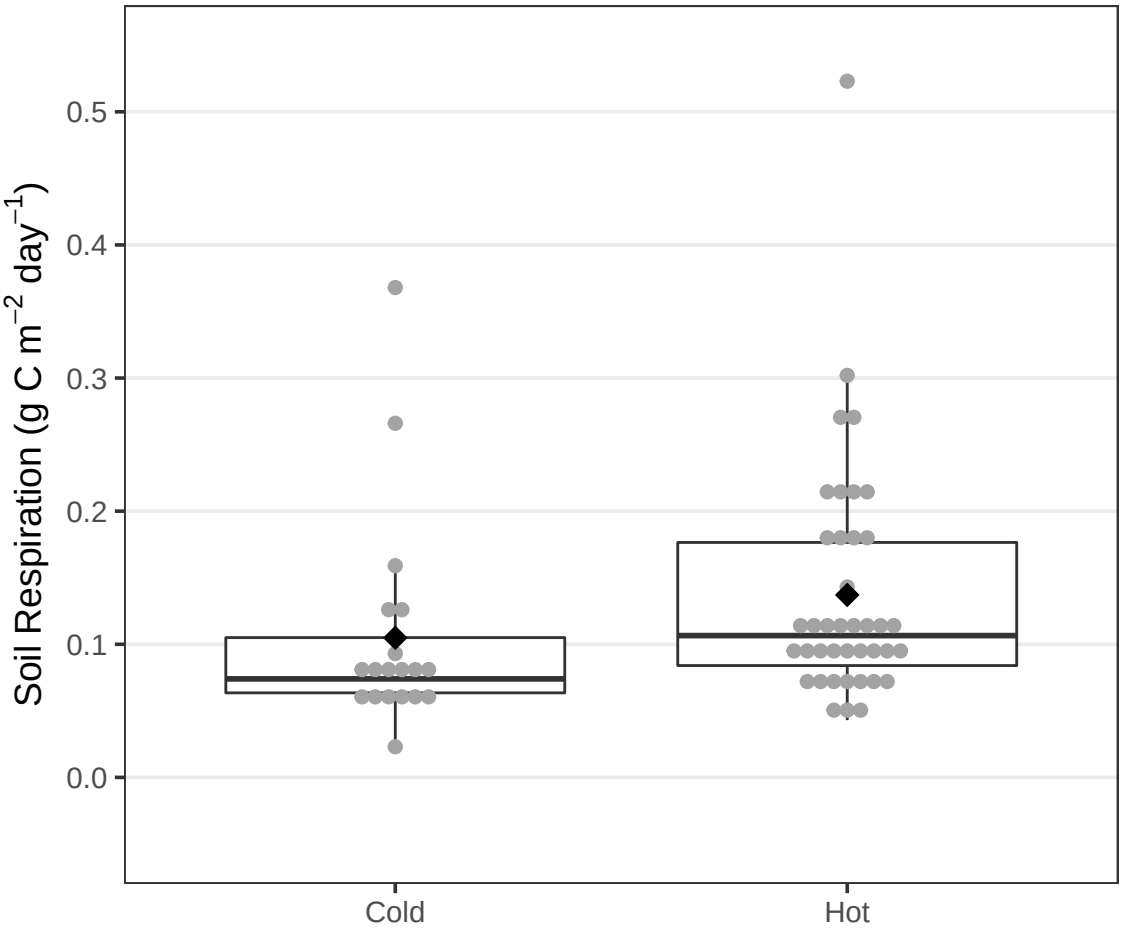
FES 524 Winter 2022 Lab 1

Bonus graphics

The bonus graphics section each week provides some "extra" R graphics code. You do not need this to complete the assignment, and this can always be skipped.

We will use some of the more advanced graphics in R to combine the figure, the results of the two-sample t-test, and the summary table that was made in Lab 1. The figure we will build on is named `g1` and the summary table is named `sumresp`. You'll see this code is fairly complicated and I skip a lot of the explanation.

This is what the figure we'll be working with looked like at the end of Lab 1:



First let's write out the results from the test as character strings in R and combine them as rows of a `data.frame`. We will take advantage of the `expression()` function in order to write text using mathematical annotation. We used this to create appropriate units in the figure `g1`. The use of `expression()` here is what will allow us to have nice looking subscripts and superscripts and to put a hat over the symbol d so we know the results is an estimated difference.

```
# Write out the results from the two-sample t-test with unequal variances
# A title for the test
titleres = as.character(expression(bold("Welch"~t-test*~"for"~a~difference*~"in"~means)))
# This is the t stat with df and p-value
ttestres = as.character(expression(t[38.3]==-1.36*","~italic(p)==0.18))
# The estimated difference in group means (I calculated from the test results)
diff = as.character(expression(hat(d)==0.032~g~C~m^-2*day^-1))
```

```
# The 95% CI of the difference
ci = as.character(expression(95*"%"~CI*":"~-0.080~to~0.016))
# Put these into a single column of a data.frame
d1 = data.frame(col = rbind(titleres, ttestres, diff, ci) )
```

In order to combine a table with a graph, we will need to turn the table into a graphical object. Graphical objects are referred to as "grobs". Use `tableGrob()` from the package **gridExtra** to do this. The figure `g1` is already a graphical object.

We'll load packages **grid** and **gtable**, which we'll need to customize the tables, as well.

We control what the table looks like within the `theme` argument. In this case, we'll make the background of the table white and control the size of the table with `padding`. In order to have all the symbols in the table print correctly we need `parse = TRUE`.

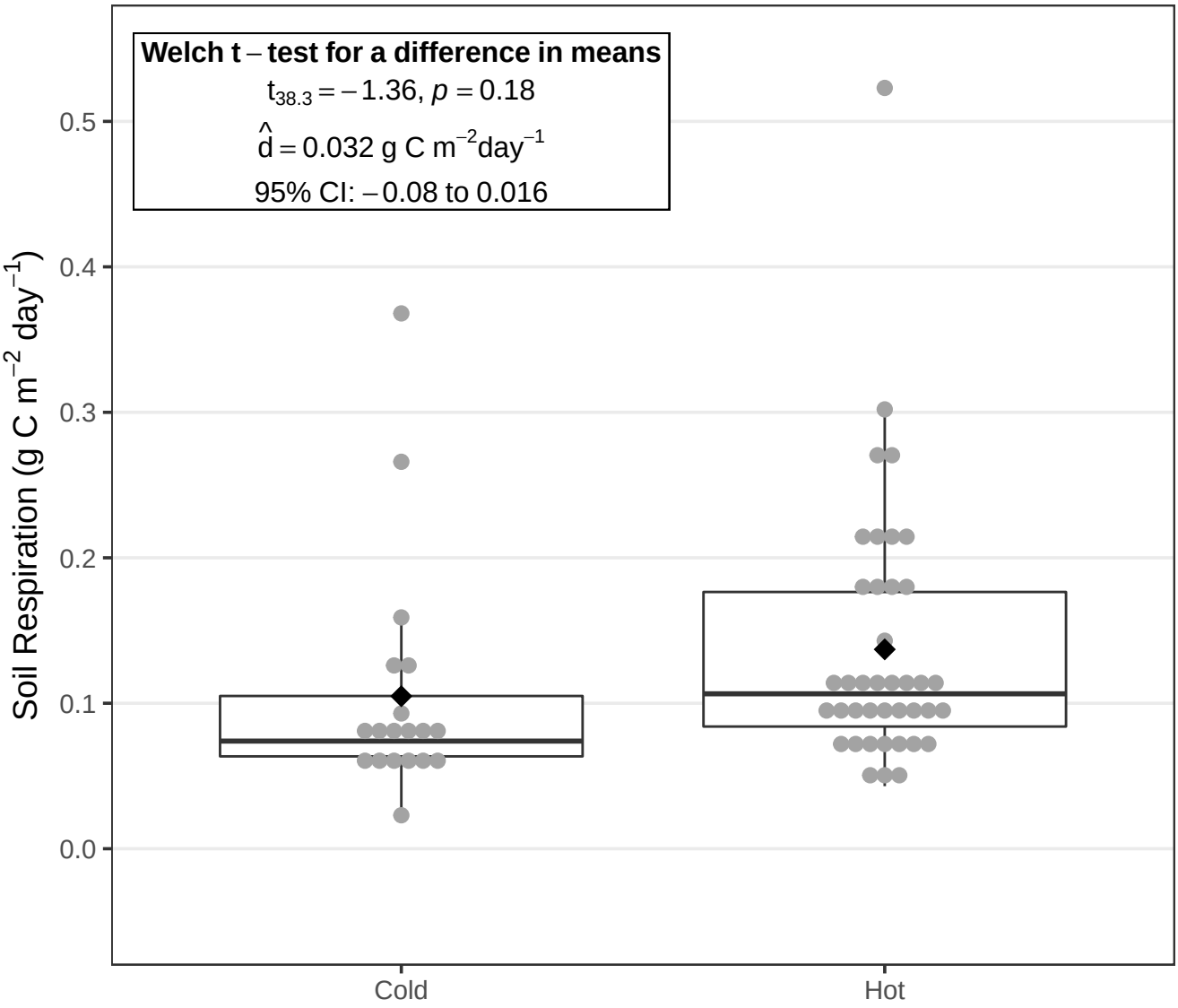
```
library(grid) # v. 4.1.2
library(gridExtra) # v. 2.3
library(gtable) # v. 0.3.0

t1 = tableGrob(d1, rows = NULL, cols = NULL,
            theme = ttheme_default(core = list(bg_params = list(fill = "white"),
                                               fg_params = list(parse = TRUE),
                                              padding = unit(c(2.5, 2.5), "mm"))))
```

To add a rectangle around the table before adding it to the plot we can use the function `gtable_add_grob()` from package **gtable**.

```
t1 = gtable_add_grob(t1,
                     grobs = rectGrob(gp = gpar(fill = NA, lwd = 2)),
                     t = 1,
                     b = nrow(t1),
                     r = ncol(t1),
                     l = 1)
```

We will add the graphical object table `t1` to the grob `g1` using `annotation_custom()`. We'll name the new graphic `g1res`.

```
( g1res = g1 +
    annotation_custom(grob = t1,
                      xmin = .5, xmax = 1.5,
                      ymin = .4, ymax = .6) )
```



Including some simple summary statistics by group in the figure will be informative for anyone reading about our study results. We will split the summary table to create a different table for each group and then convert the tables into two different graphical objects, named `t2` and `t3`, using `tableGrob()`. I want these two objects to look the same, so I'll define all the `tableGrob()` theme information first.

We are going to have column names in these two tables, so have to set the background of the header to white. We'll also change the font size to make it easier to read.

```
ttsum = ttheme_default(core = list(bg_params = list(fill = "white"),
                                   fg_params = list(fontsize = 7.5)),
                       colhead = list(bg_params = list(fill = "white", col = "white"),
                                      fg_params = list(fontsize = 7.5)))
```

Now we can make a separate data.frame for each group and make a table grob with it. We'll had a black rectangle around each table as before, and we'll also add a line between the column headers and the rest of the table information.

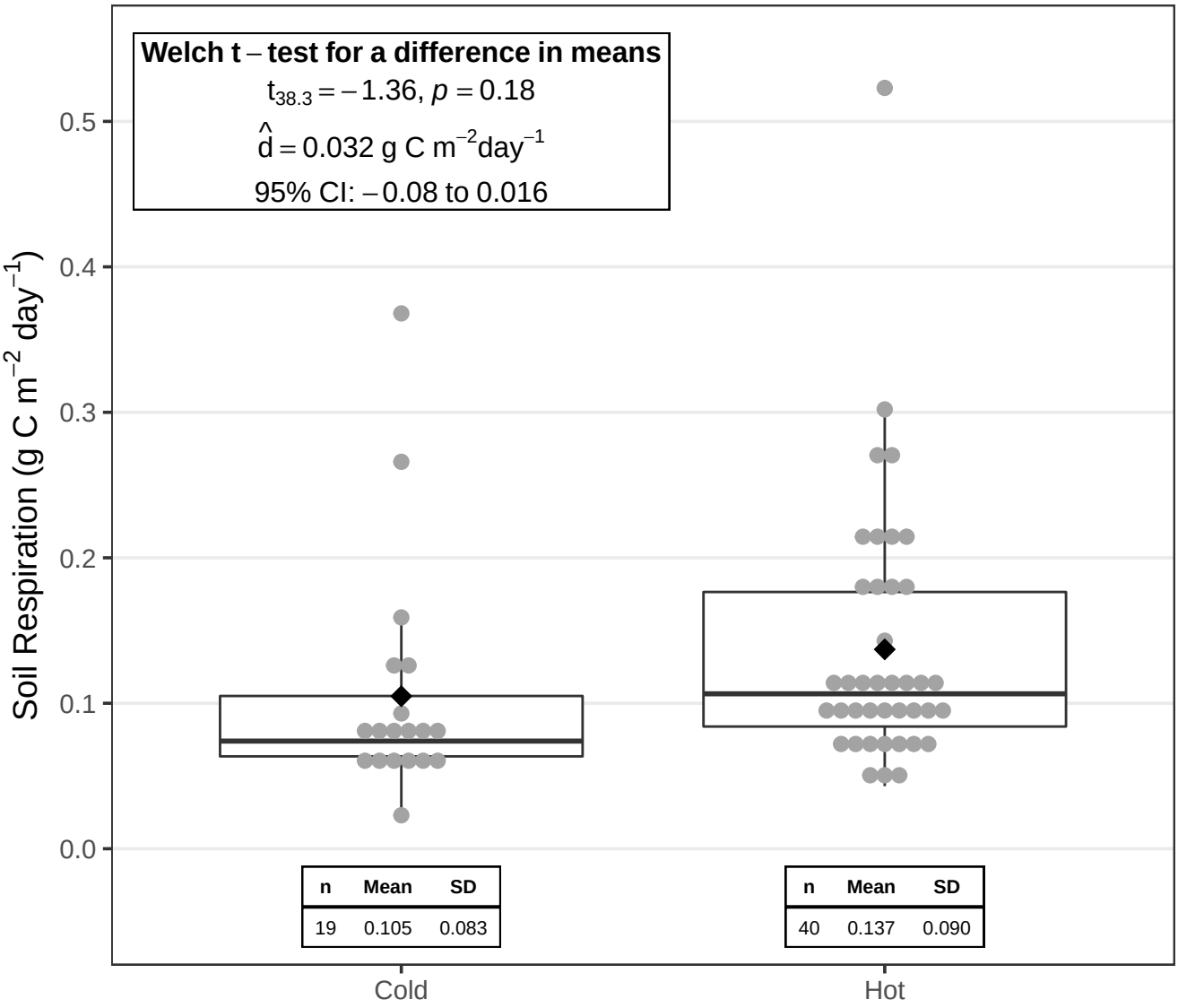
Notice that by default I lost the trailing zero on the standard deviation for one of the groups, and had to add it back on using `sprintf()`.

```
sumresp1 = sumresp[1, 2:4]
t2 = tableGrob(sumresp1, rows = NULL, theme = ttsum)

# Add rectangle around outside of table
t2 = gtable_add_grob(t2,
```

```
                  grobs = rectGrob(gp = gpar(fill = NA, lwd = 2)),
                  t = 2,
                  b = nrow(t2),
                  r = ncol(t2),
                  l = 1)

# Add line under column names
t2 = gtable_add_grob(t2,
                  grobs = rectGrob(gp = gpar(fill = NA, lwd = 2)),
                  t = 1,
                  r = ncol(t2),
                  l = 1)

sumresp2 = sumresp[2,2:4]

# Watch out - lose the trailing 0 on SD by default
sumresp2[3] = sprintf("%.3f", round(sumresp2[3], 3))

t3 = tableGrob(sumresp2, rows = NULL, theme = ttsum)
# Add rectangle around outside and line under column names
t3 = gtable_add_grob(t3,
                  grobs = rectGrob(gp = gpar(fill = NA, lwd = 2)),
                  t = 2,
                  b = nrow(t3),
                  r = ncol(t3),
                  l = 1)
t3 = gtable_add_grob(t3,
                  grobs = rectGrob(gp = gpar(fill = NA, lwd = 2)),
                  t = 1,
                  r = ncol(t3),
                  l = 1)
```

Now we can add these two graphical objects onto `g1res` to make our final figure, which we will call `g1fin`.

```
g1fin = g1res +
     annotation_custom(grob = t2,
                       xmin = .5, xmax = 1.5,
                       ymin = -.08, ymax = 0) +
     annotation_custom(grob = t3,
                       xmin = 1.5, xmax = 2.5,
                       ymin = -.08, ymax = 0)
g1fin
```



If you wanted to save the final figure `g1fin`, you could do so using `ggsave()`. In the example here we'll save it as a PNG file named `lab1figure`, and change the width and height of the final figure to be 10 inches by 7 inches, which is bigger than the plots shown above.

```
ggsave("lab1figure.png", plot = g1fin, width = 10, height = 7)
```