Summarising datasets, make graphs with error bars

Sébastien Guyader

2018 November 28

Load the required packages.

```
library(tidyverse)
library(grid)
```

Summarise a dataset and make a graph with error bars.

Load the ToothGrowth data (tooth growth (len) as function of supplement type (supp) and dose (dose)

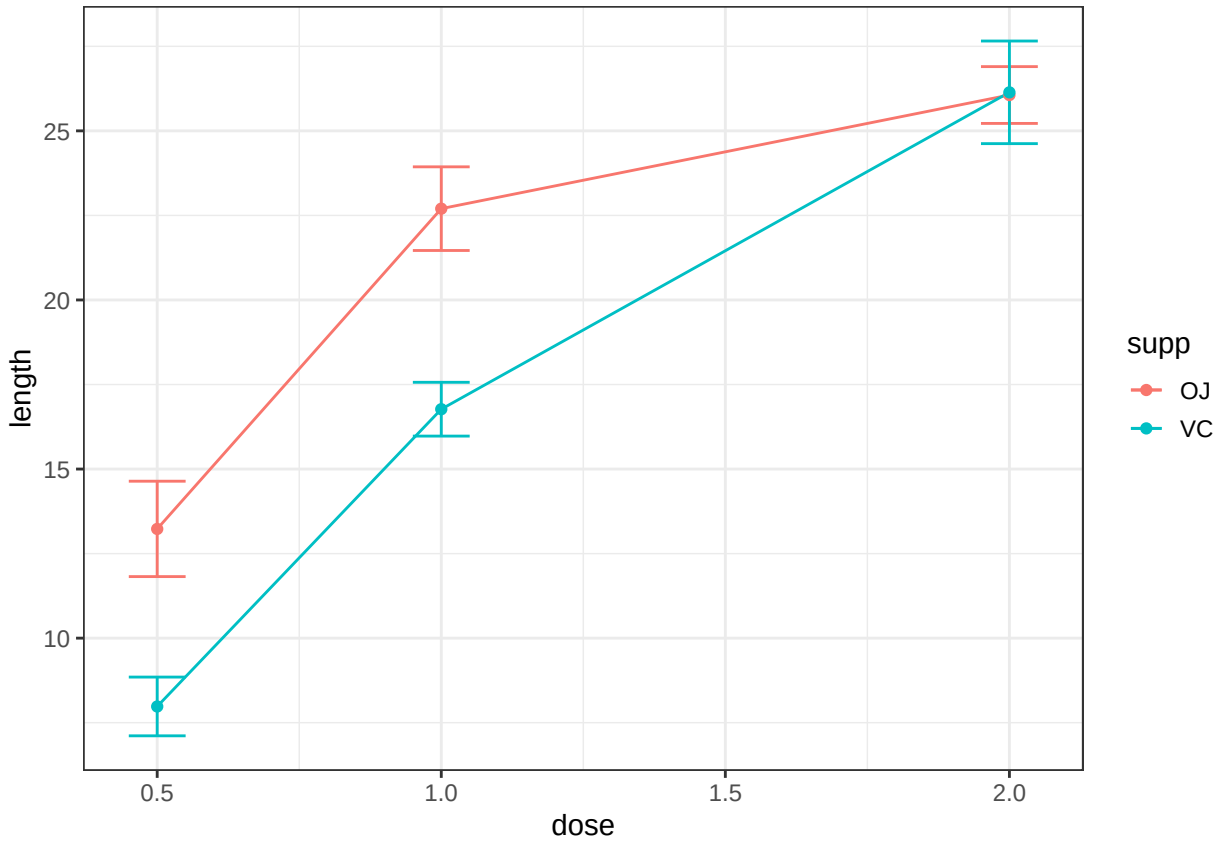
```
tg <- ToothGrowth
```

Summarize the data set grouped by supp and dose, with dplyr.

```
tg_summarised <- tg %>% group_by(supp, dose) %>% summarise(
  N=n(),
  length=mean(len),
  sd=sd(len),
  se=sd/sqrt(N),
  ci.student=qt(0.975,df=N-1)*se
)
```

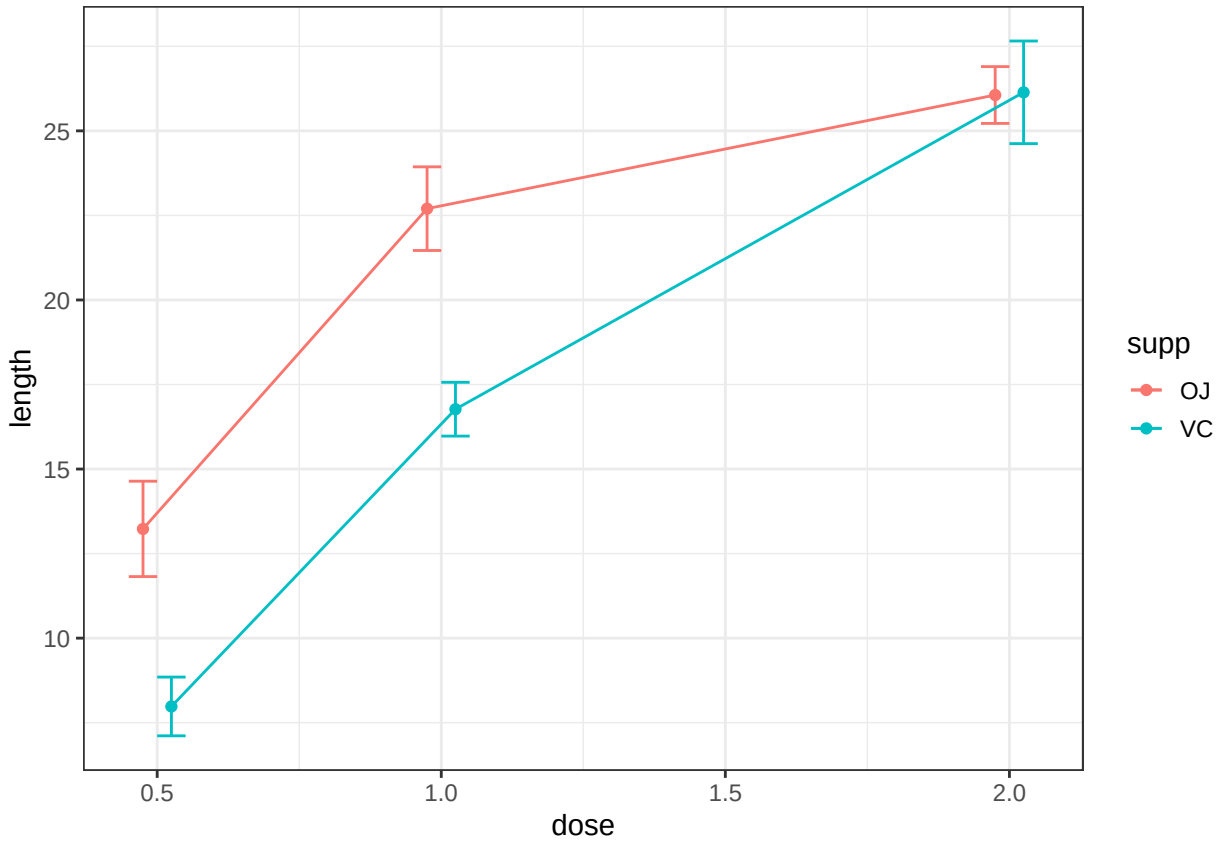
Generate plot in ggplot2, with error bars based on SEM.

```
ggplot(tg_summarised, aes(x=dose, y=length, colour=supp)) +
  geom_errorbar(aes(ymin=length-se, ymax=length+se), width=.1) +
  geom_line() +
  geom_point() +
  theme_bw() # use the custom theme "my_ggplot2_theme"
```



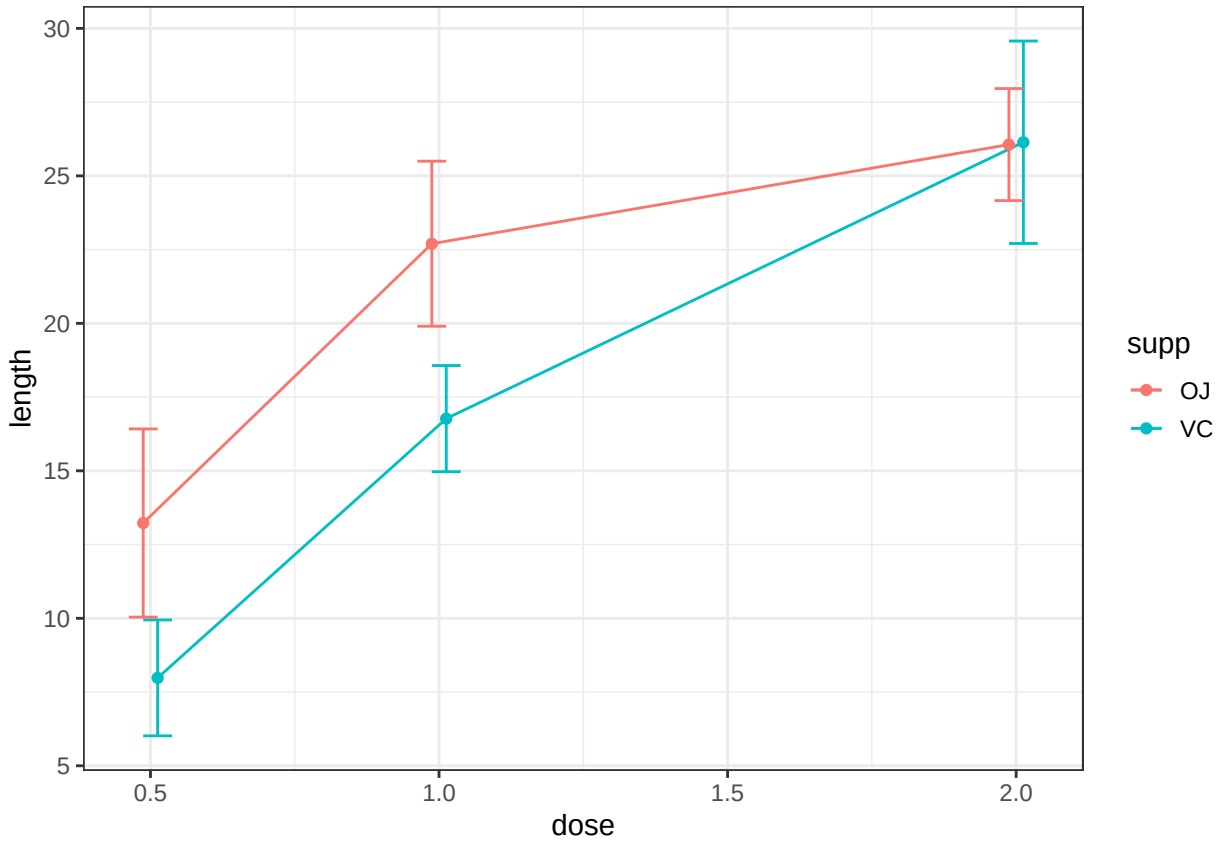
The errorbars overlapped, so use “position_dodge” to move them horizontally with position_dodge(0.1) to move them .05 to the left and right

```
ggplot(tg_summarised, aes(x=dose, y=length, colour=supp)) +
  geom_errorbar(aes(ymin=length-se, ymax=length+se), width=.1,
                position=position_dodge(0.1)) +
  geom_line(position=position_dodge(0.1)) +
  geom_point(position=position_dodge(0.1)) +
  theme_bw()
```



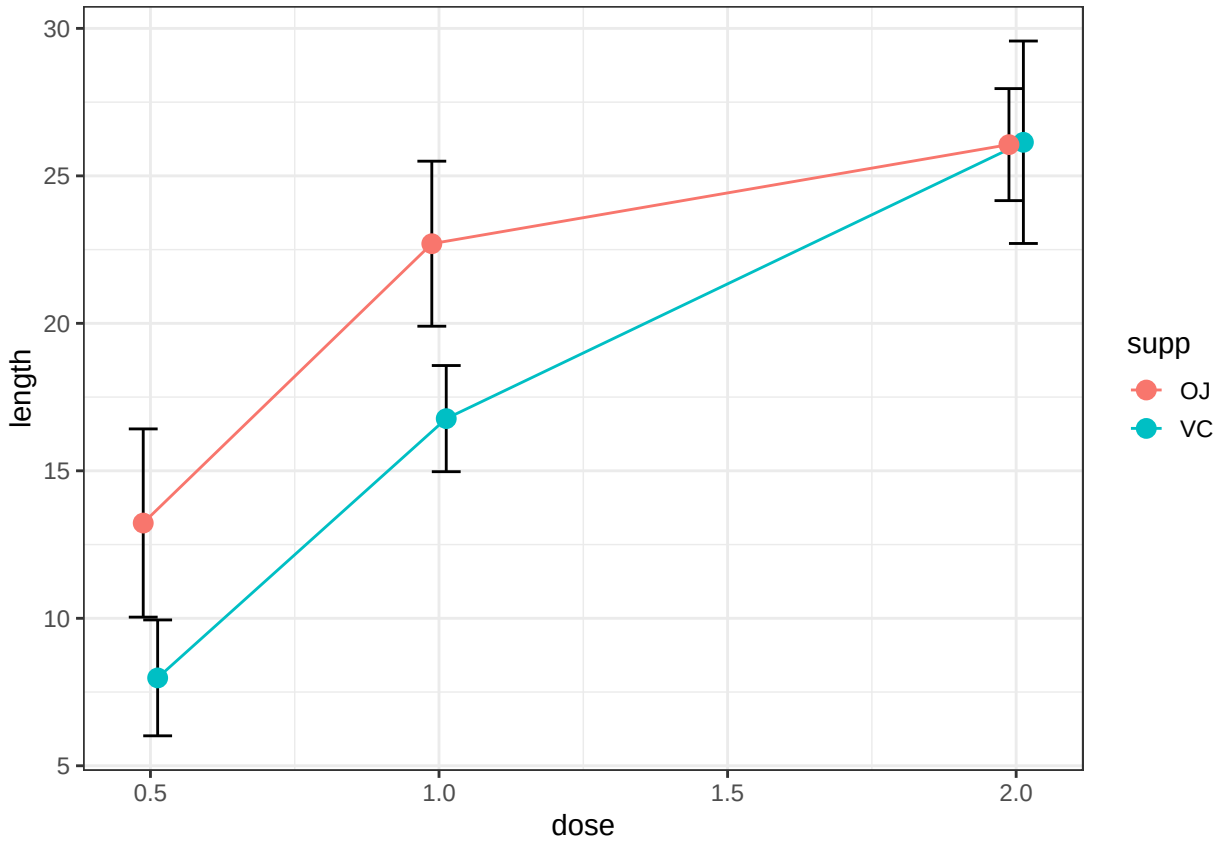
Use 95% confidence interval instead of SEM

```
ggplot(tg_summarised, aes(x=dose, y=length, colour=supp)) +
  geom_errorbar(aes(ymin=length-ci.student, ymax=length+ci.student),
                width=.1, position=position_dodge(0.05)) +
  geom_line(position=position_dodge(0.05)) +
  geom_point(position=position_dodge(0.05)) +
  theme_bw()
```



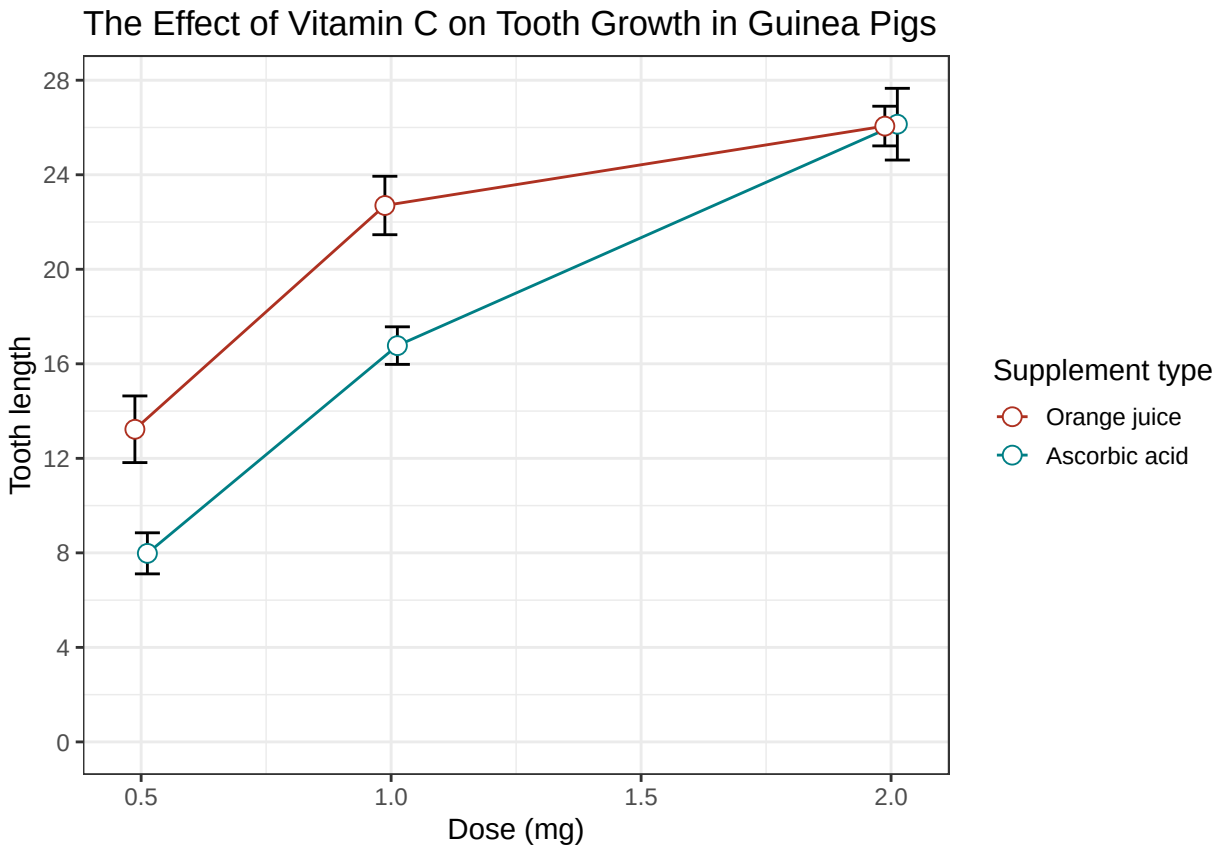
Black error bars - notice the mapping of 'group=supp' – without it, the error bars won't be dodged!

```
ggplot(tg_summarised, aes(x=dose, y=length, colour=supp, group=supp)) +
  geom_errorbar(aes(ymin=length-ci.student, ymax=length+ci.student),
                colour="black", width=.1, position=position_dodge(0.05)) +
  geom_line(position=position_dodge(0.05)) +
  geom_point(position=position_dodge(0.05), size=3) +
  theme_bw()
```



A finished graph with error bars representing the standard error of the mean might look like this. The points are drawn last so that the white fill goes on top of the lines and error bars.

```
ggplot(tg_summarised, aes(x=dose, y=length, colour=supp, group=supp)) +
  geom_errorbar(aes(ymin=length-se, ymax=length+se),
                colour="black", width=.1, position=position_dodge(0.05)) +
  geom_line(position=position_dodge(0.05)) +
  geom_point(position=position_dodge(0.05), size=3,
             shape=21, fill="white") + # 21 is filled circle
  xlab("Dose (mg)") +
  ylab("Tooth length") +
  scale_colour_hue(name="Supplement type", # Legend label
                   breaks=c("OJ", "VC"),
                   labels=c("Orange juice", "Ascorbic acid"),
                   l=40) +                    # Darker colors, lightness=40
  ggtitle("The Effect of Vitamin C on Tooth Growth in Guinea Pigs") +
  expand_limits(y=0) +                        # Expand y range
  scale_y_continuous(breaks=0:20*4) +         # Set tick every 4
  theme_bw()
```



Summarise a dataset within subject measurements (e.g. repeated measures) and plot the right means and error bars.

Create new data with subject values measured at 2 different times.

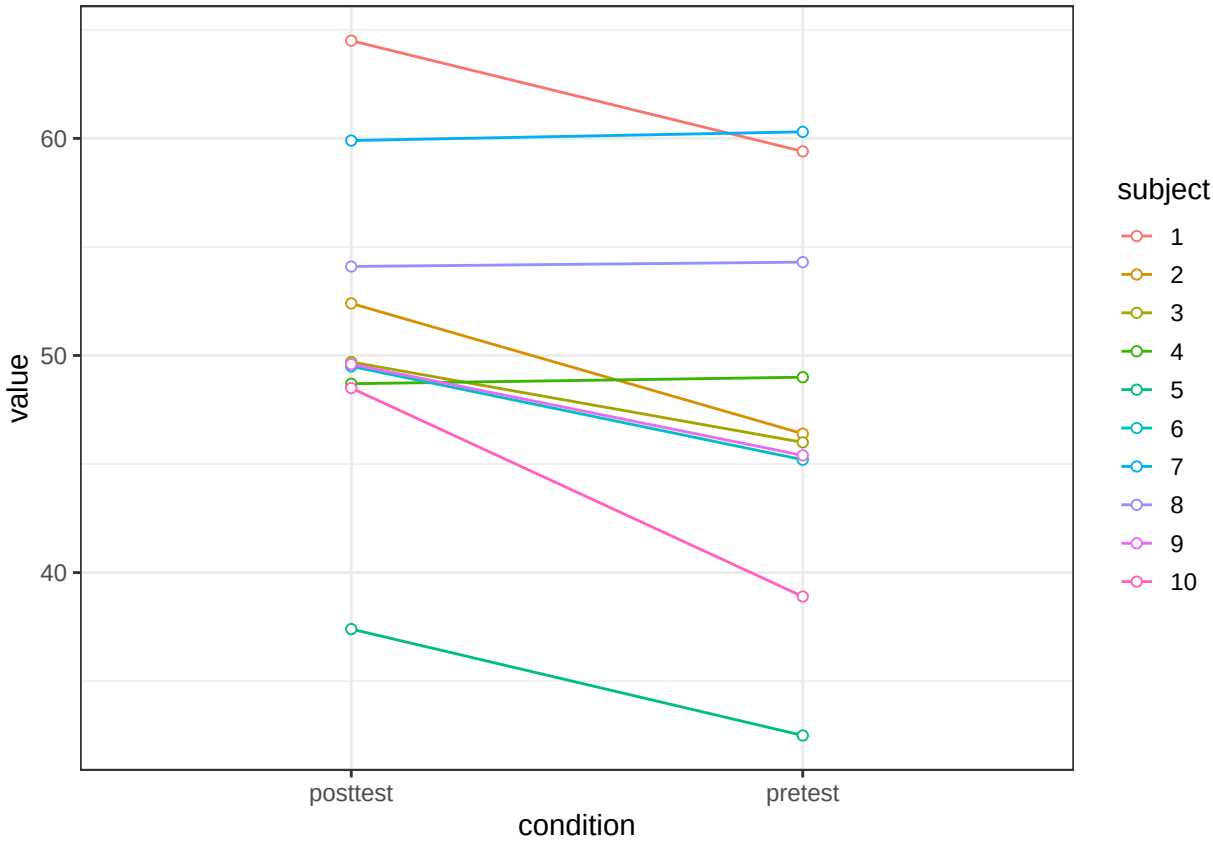
```
dfw <- read.table(header=TRUE, text='
 subject pretest posttest
       1    59.4     64.5
       2    46.4     52.4
       3    46.0     49.7
       4    49.0     48.7
       5    32.5     37.4
       6    45.2     49.5
       7    60.3     59.9
       8    54.3     54.1
       9    45.4     49.6
      10    38.9     48.5
 ')
```

Treat subject ID as a factor, and convert to long format with tidyr:gather()

```
dfw$subject <- factor(dfw$subject)

dfw_long <- gather(dfw, key=condition, value=value, pretest:posttest)
```

Plot the individuals

```
ymax <- max(dfw_long$value)
ymin <- min(dfw_long$value)

ggplot(dfw_long, aes(x=condition, y=value, colour=subject, group=subject)) +
  geom_line() + geom_point(shape=21, fill="white") +
  ylim(ymin,ymax) +
  theme_bw()
```



Use the custom function normDataWithin()

```
library(Rmisc)
```

```
## Loading required package: lattice
```

```
## Loading required package: plyr
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

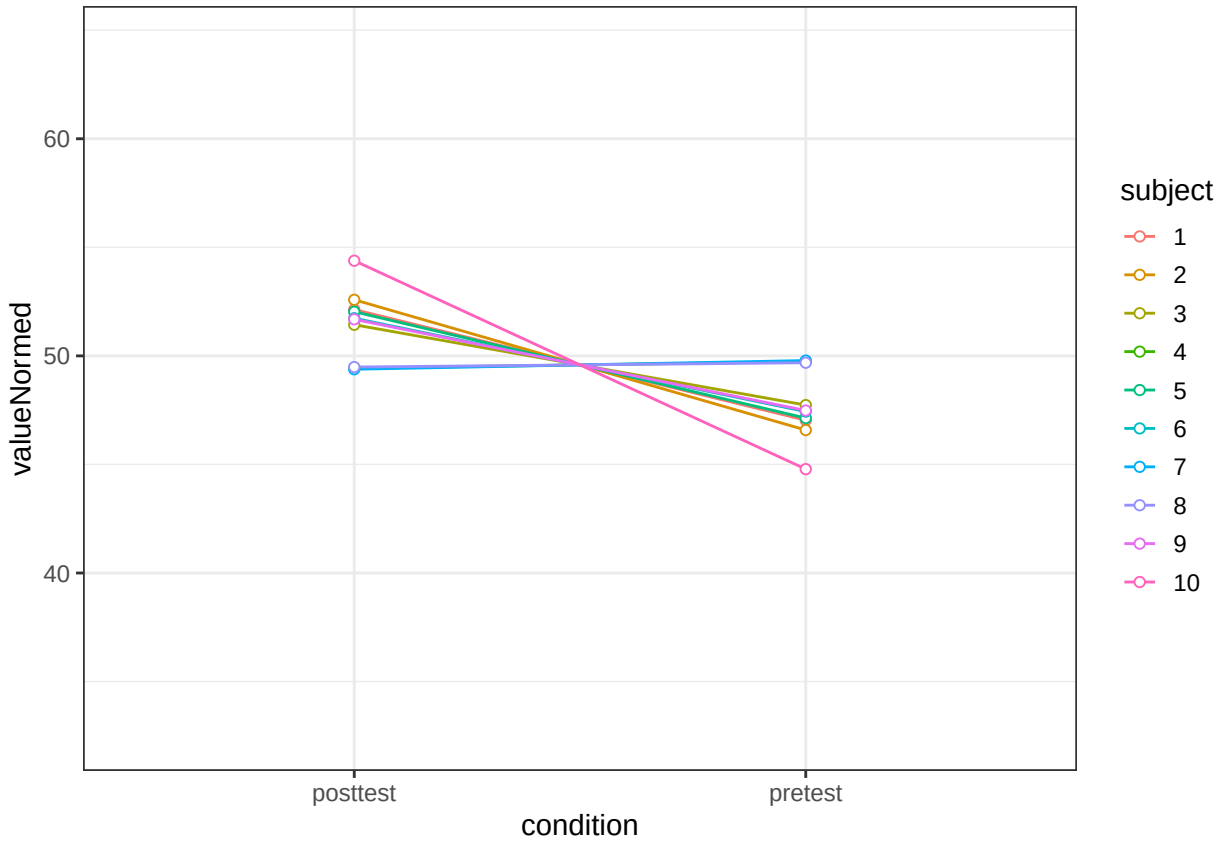
```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:purrr':
##
##     compact
```

```
dfwNorm_long <- normDataWithin(data=dfw_long, idvar="subject", measurevar="value")
```

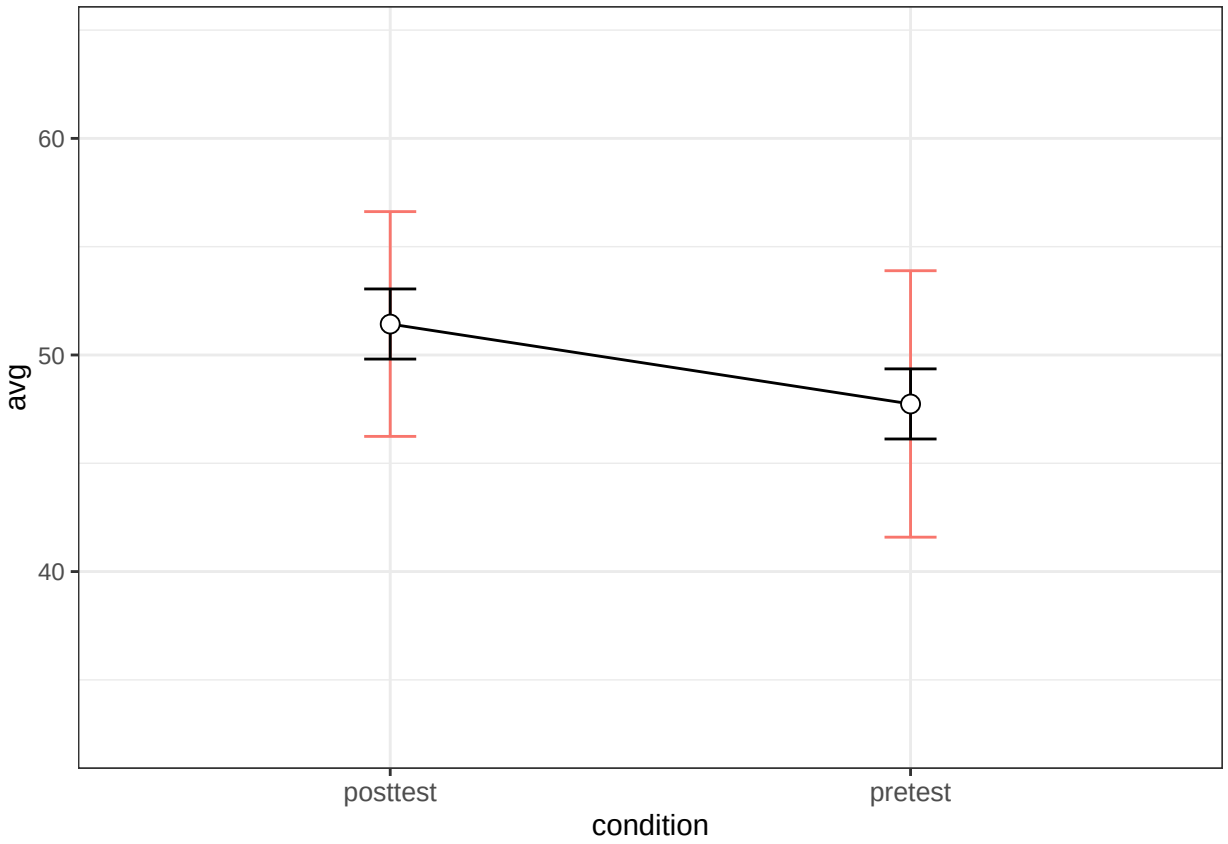
Plot the normed individuals

```
ggplot(dfwNorm_long, aes(x=condition, y=valueNormed,
                         colour=subject, group=subject)) +
  geom_line() + geom_point(shape=21, fill="white") +
  ylim(ymin,ymax) +
  theme_bw()
```




```
# get the number of levels in the within group
dfwNorm_long$nWithinGroups <- n_distinct(dfwNorm_long$condition)
```

```r
dfwNorm_long %>% group_by(condition) %>% dplyr::summarise(
  N=n(),
  nWithinGroups=mean(nWithinGroups),
  avg=mean(value),
  sd=sd(value),
  se=sd/sqrt(N),
  CI.student=qt(0.975,df=N-1)*se,
  sd_normed=sd(valueNormed)*sqrt(nWithinGroups/(nWithinGroups-1)),
  se_normed=sd_normed/sqrt(N),
  CI.student_normed=qt(0.975,df=N-1)*se_normed
) %>%
ggplot(aes(x=condition, y=avg, group=1)) +
  geom_line() +
  geom_errorbar(width=.1, aes(ymin=avg-CI.student,
                              ymax=avg+CI.student, color="red")) +
  geom_errorbar(width=.1, aes(ymin=avg-CI.student_normed,
                              ymax=avg+CI.student_normed)) +
  geom_point(shape=21, size=3, fill="white") +
  ylim(ymin,ymax) +
  theme_bw() +
  theme(legend.position = "none",                    # remove the legend
        axis.text.x = element_text(size=rel(1.1))  # increase the x axis
        )                                                  # labels size
```


60
50
40
avg
posttest
pretest
condition