ggplot for beginners

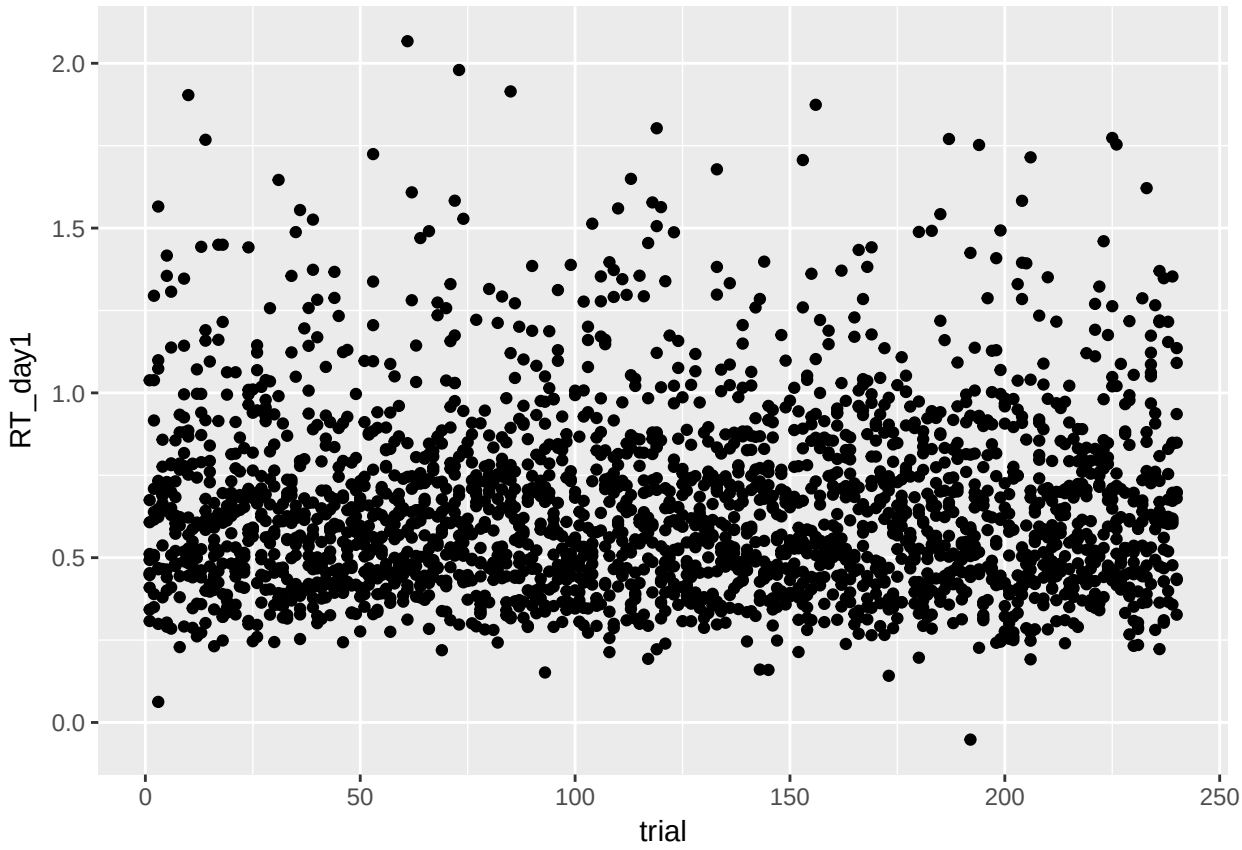
Last edit: 14.05.2018 - Chris edited some errors she found post PhD workshop

```
library(ggplot2) # we need to load the ggplot2 functions from our library of tools so that R/Rstudio ca

data_dir = "/Users/Chris/Bitbucket/2018_PhDcamp/scripts" # you can change this to the place where you h
# Now we can read the csv file - header=TRUE means that the read.csv function will take the column
# headings from your csv file and then use them for the names of your data frame
data <- read.csv(file.path(data_dir,"data","raw_data.csv"), header=TRUE)

# What if I want to change some column names, I hear you ask!!
# I have commented this out because our script uses the names from the excel file
# you could do this names(data)[1] <- "col1" - this names the first column of data to "col1" and so on
# you could also do colnames(data) <- c("subj_no" = "sub_no", "stim" = "stimuli") - so we tell R that w
# want to change "subj_no" to "sub_no" and so on - this way we dont have to remember which column the
# thing we want is

# Let's have a quick look at our data se we can see the column names etc
head(data)
```

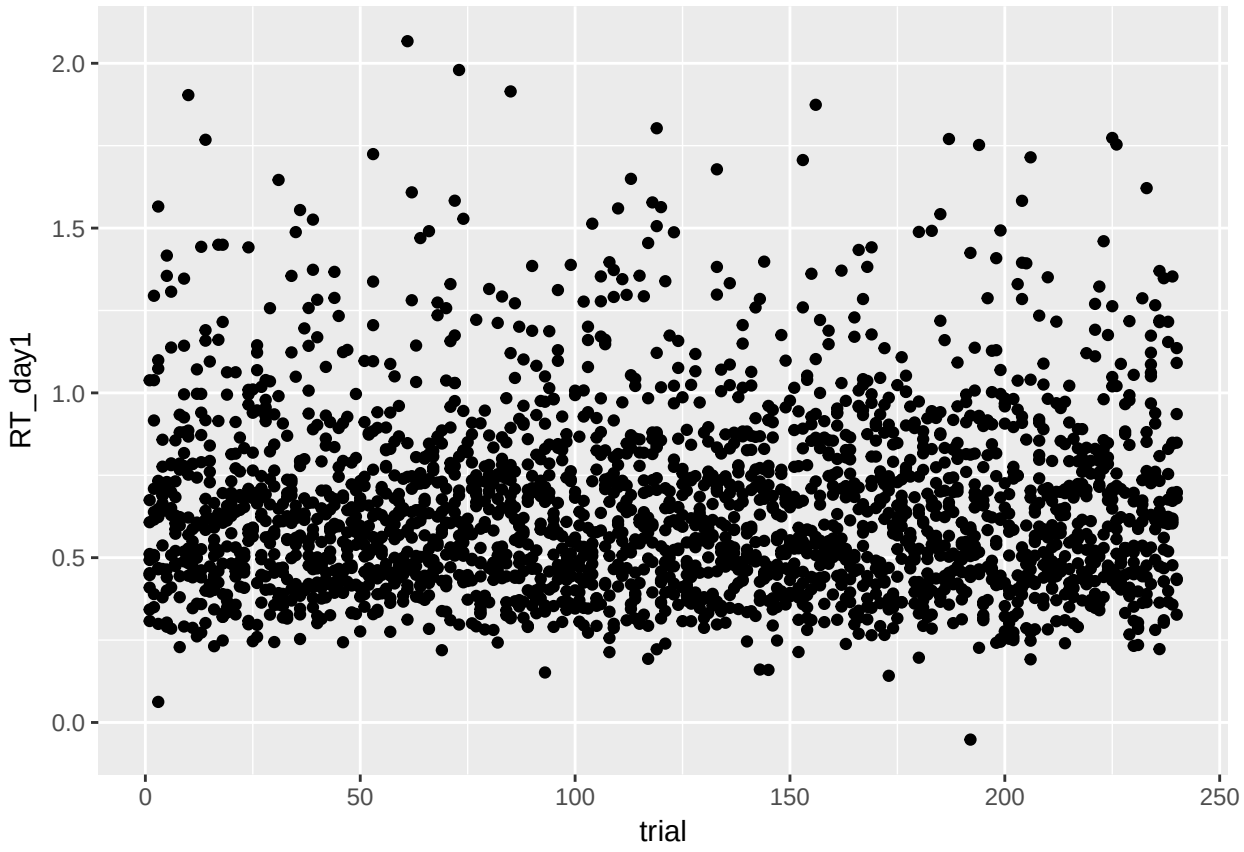
```
##   X subj_no age group stim trial difficulty resp corr conf   RT_day1
## 1 1      10  23     2    2     1          2    1    0    2 0.6076149
## 2 2      10  36     2    1     2          3    2    0    1 0.7090037
## 3 3      10  38     2    2     3          2    2    1    4 0.6232013
## 4 4      10  25     2    2     4          1    1    0    3 0.6880759
## 5 5      10  20     2    1     5          2    1    1    4 0.7296480
## 6 6      10  40     2    1     6          1    1    1    5 0.4098684
##    RT_day2
## 1 1.558945
## 2 1.719140
## 3 1.583572
## 4 1.686074
## 5 1.751758
## 6 1.246506
```

```
# The first thing you need to do is tell ggplot what data you want to plot
ggplot(data)
```

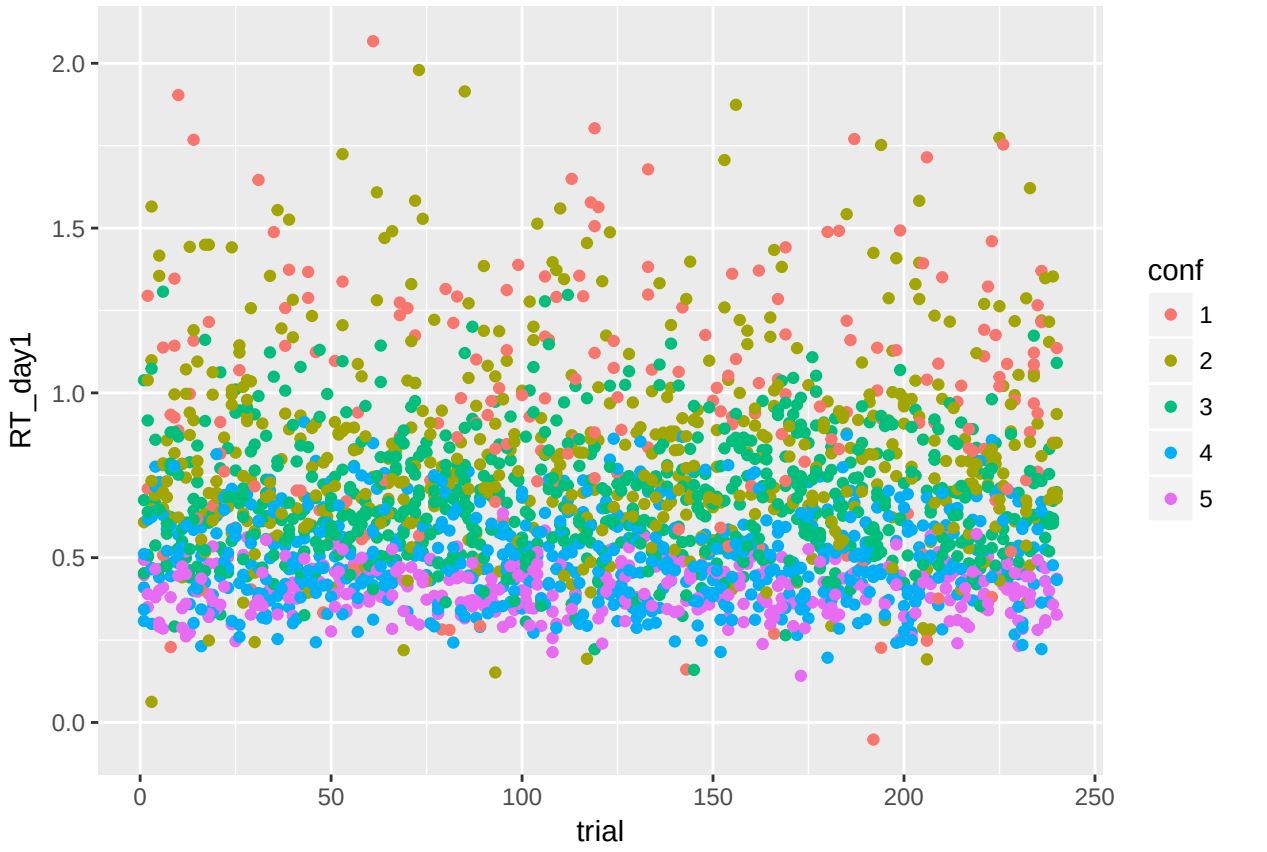
```
# Step1 - Tell ggplot what bits of the data you want to plot ggplot data x = trial, y = RT_daty1
ggplot(data, aes(x = trial, y = RT_day1)) + geom_point()
```



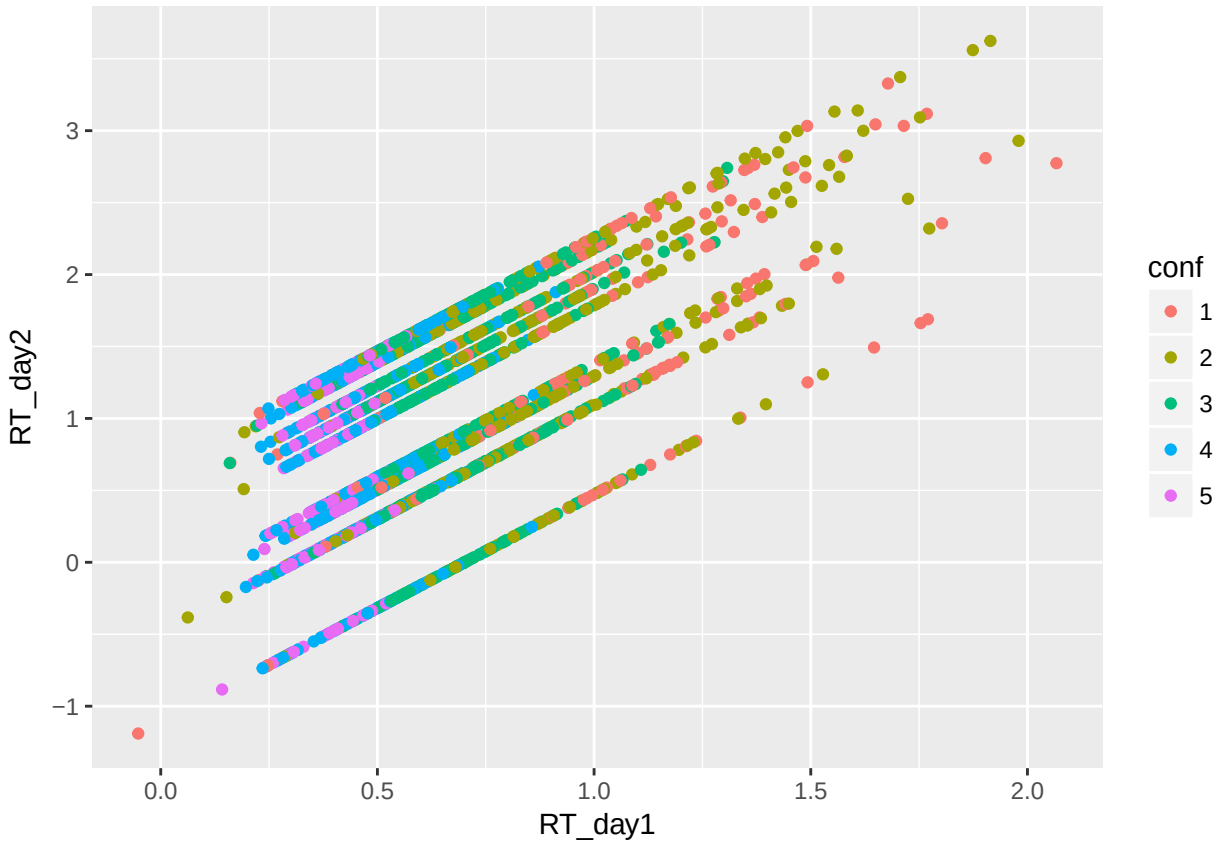

```
# Step 2 - Tell ggplot what you want to display it as points
ggplot(data, aes(x = trial, y = RT_day1)) + geom_point()
```



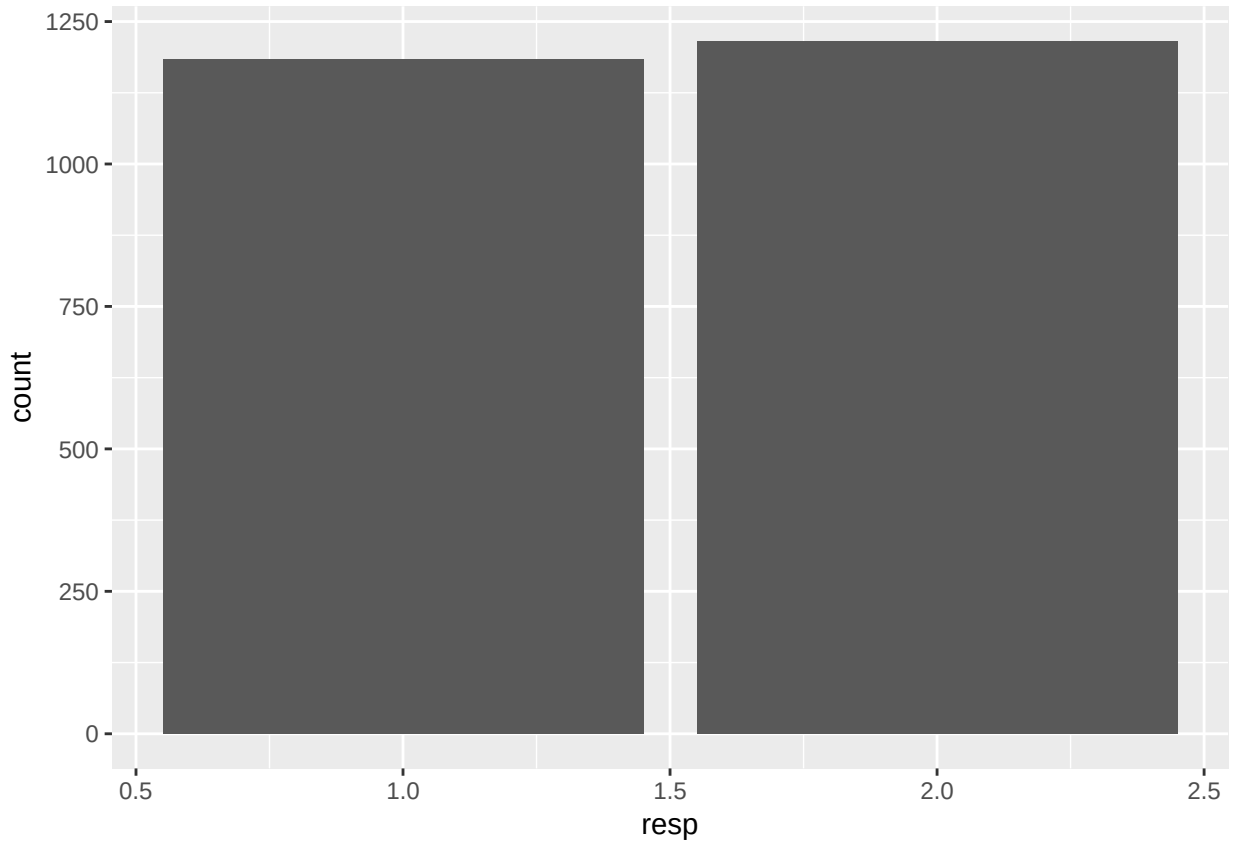

```
data[, 'conf'] <- as.factor(data[, 'conf'])

# Step 3 - Let's try adding some colour
data[,'conf'] <- as.factor(data[,'conf'])
# geom_point inherits everything from ggplot(data) part

ggplot(data, aes(x = trial, y = RT_day1)) + geom_point((aes(colour = conf)))
```



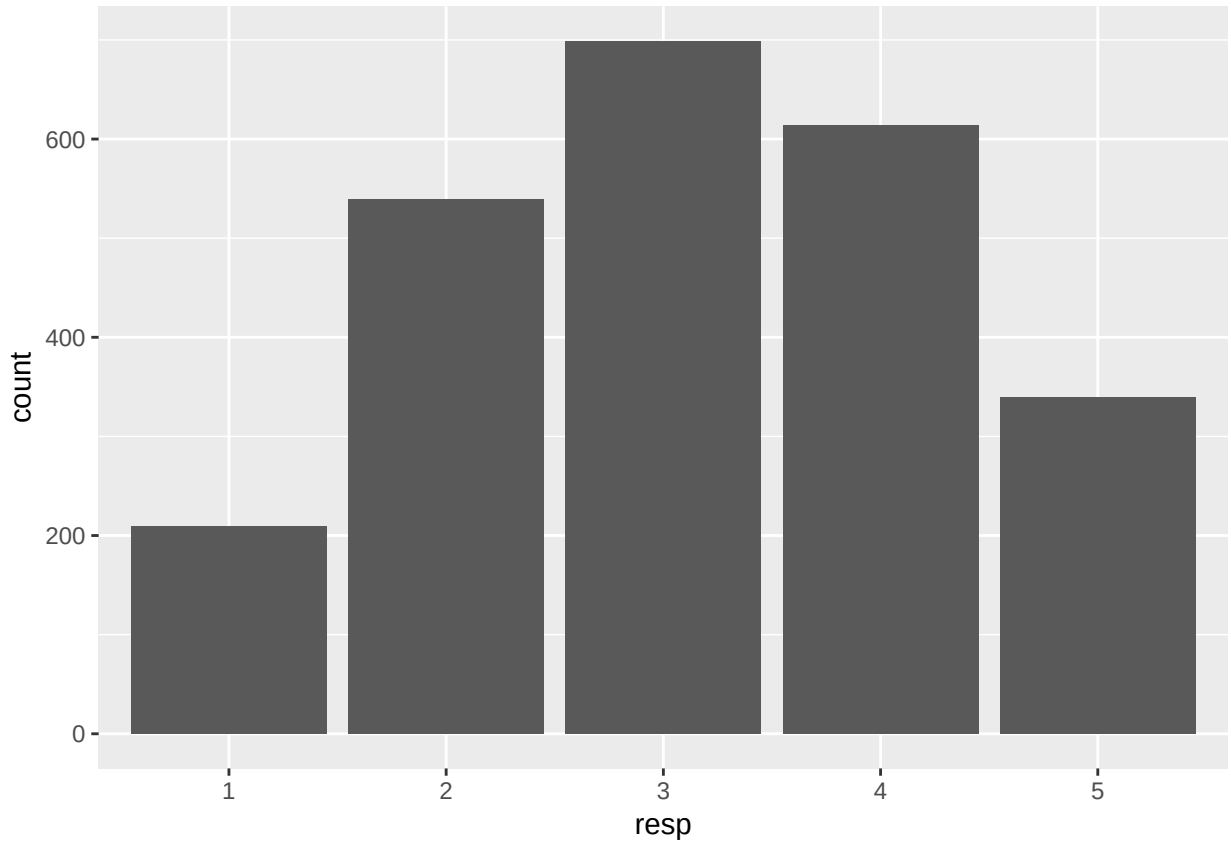

```
ggplot(data, aes(x = RT_day1, y = RT_day2)) + geom_point(aes(colour = conf))
```



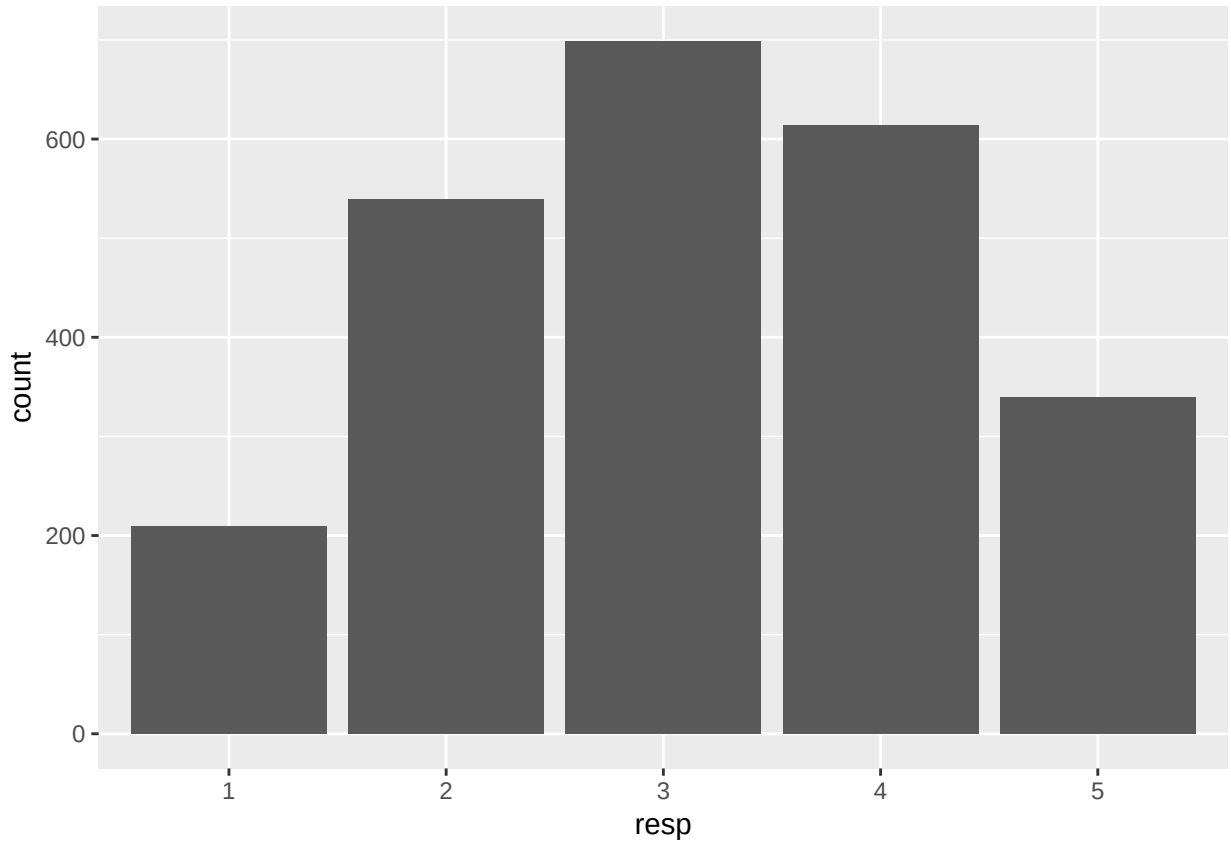

```
# What if I would like a bar plot?
# Imagine I want to know what the frequencies of responses according to confidence ratings look like
ggplot(data, aes(x = resp)) + geom_bar() # would just give me frequencies of two responses (faces vs ho
```



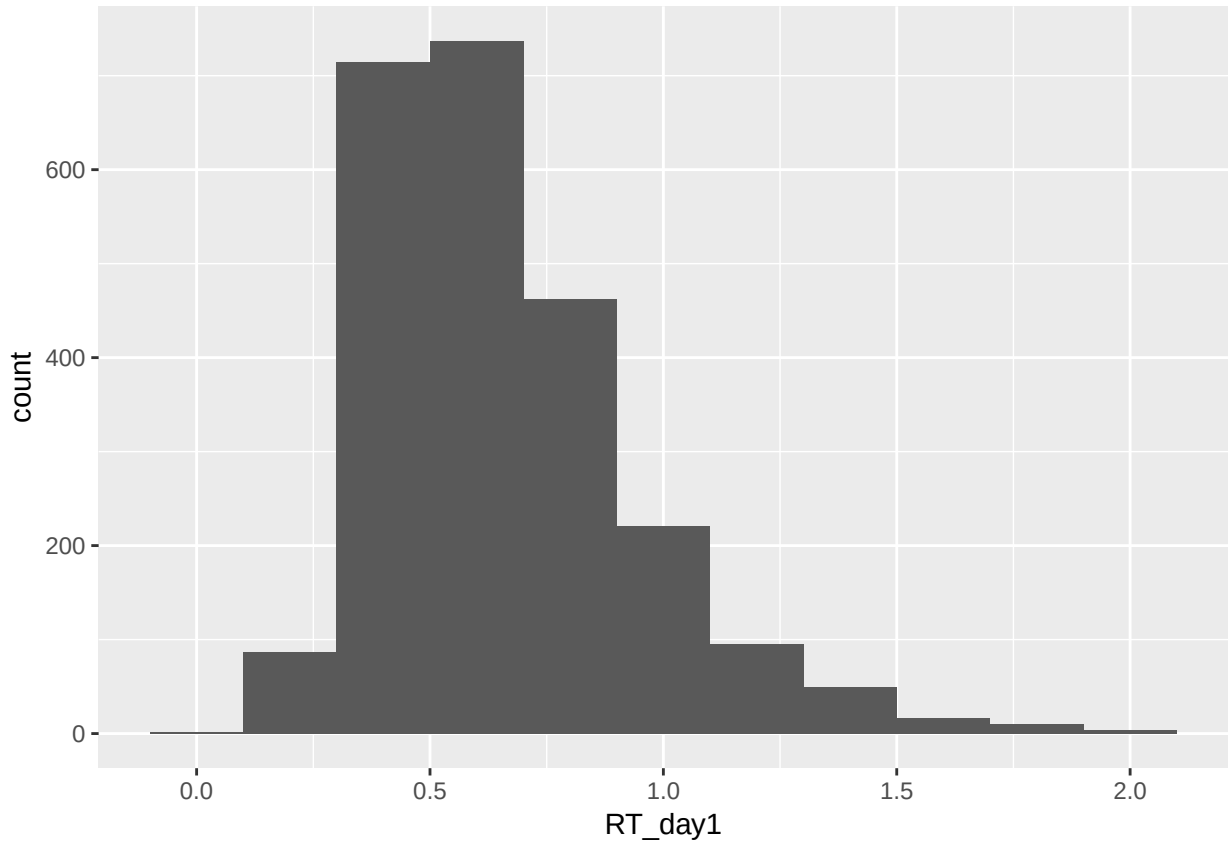

```
ggplot(data, aes(x = resp)) + geom_bar(aes(conf)) # gives me how many confidence ratings
```



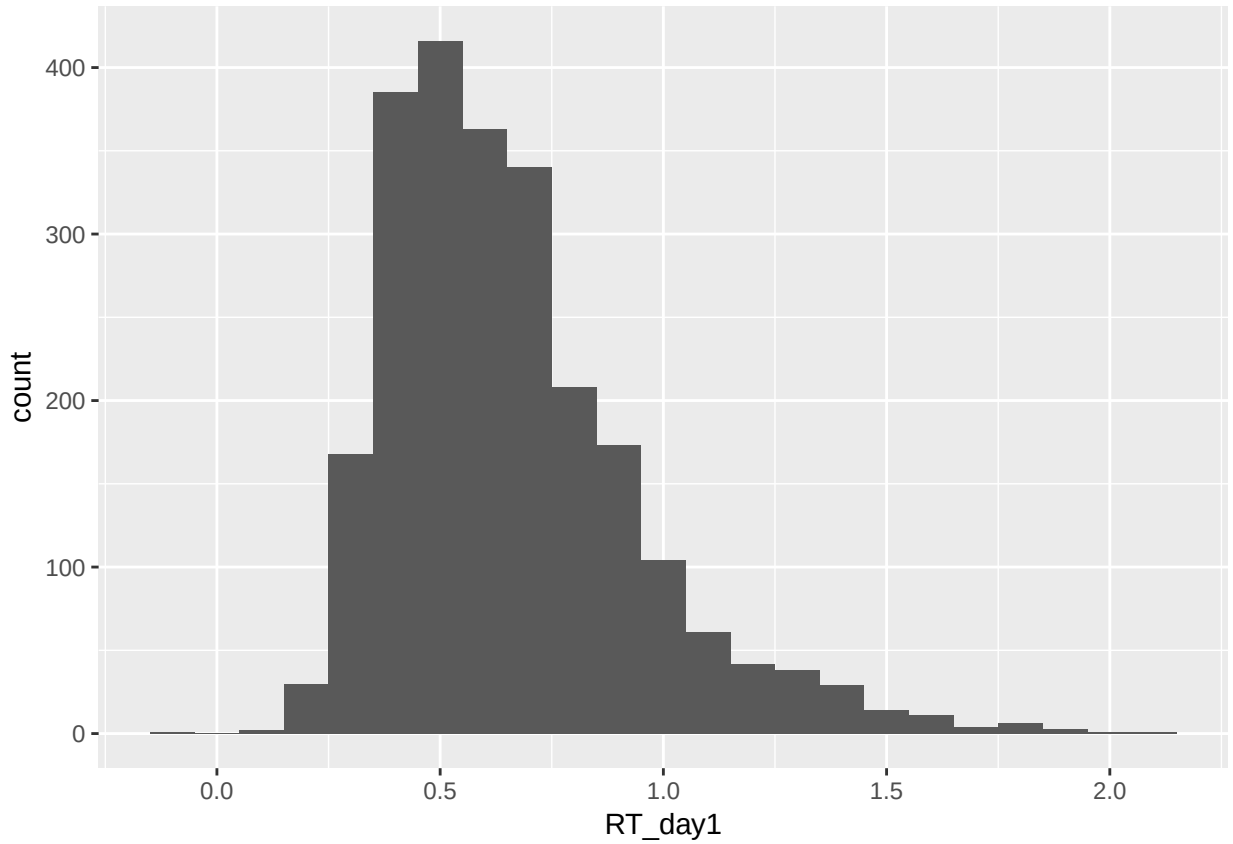

```
ggplot(data, aes(x = resp)) + geom_bar(aes(conf))
```



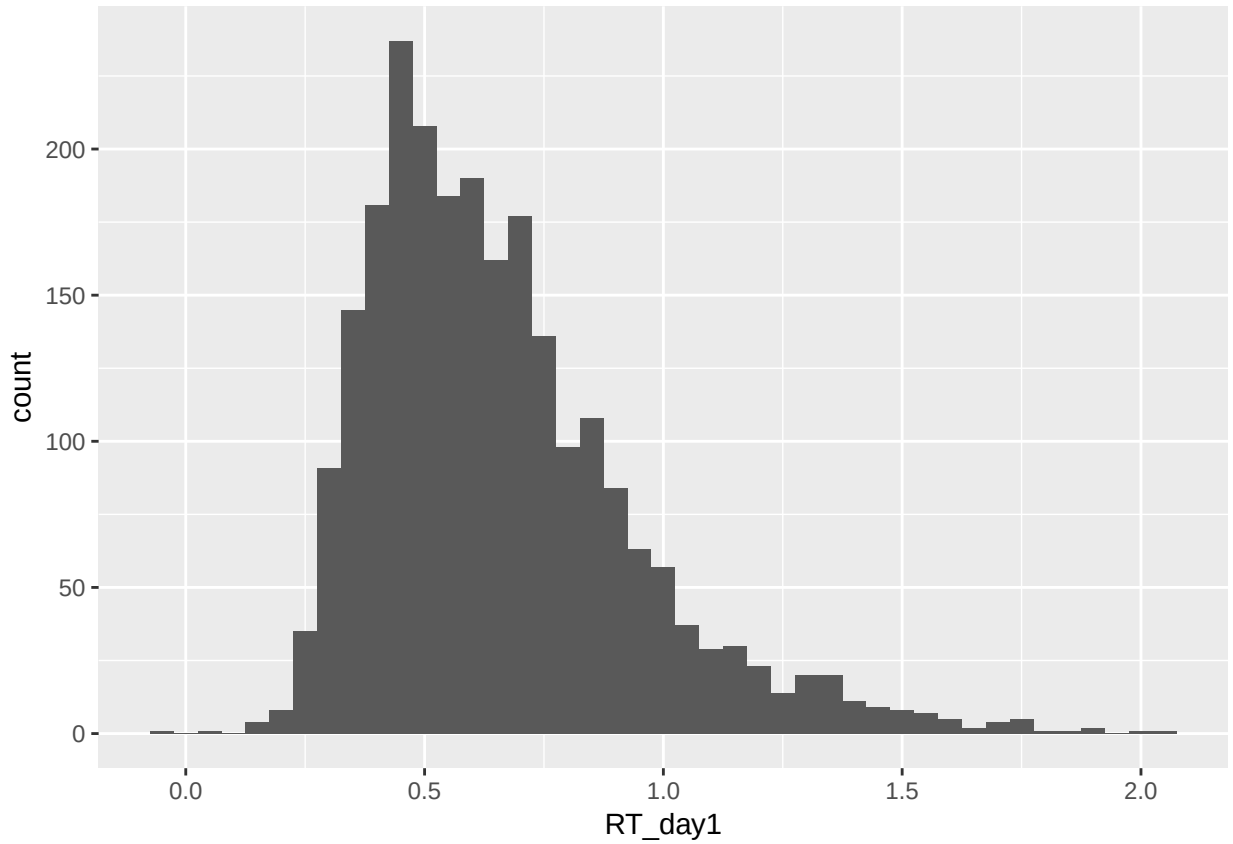

```
# We can also change how thin or thick the bars are - the thinness or thickness
# tells us how many data points we count (get frequencies) over. These are called "bins"
# Below we can change the bin size to different values using 'binwidth' inside the geom_histogram argum
ggplot(data, aes(x = RT_day1)) + geom_histogram(binwidth = 0.2)
```



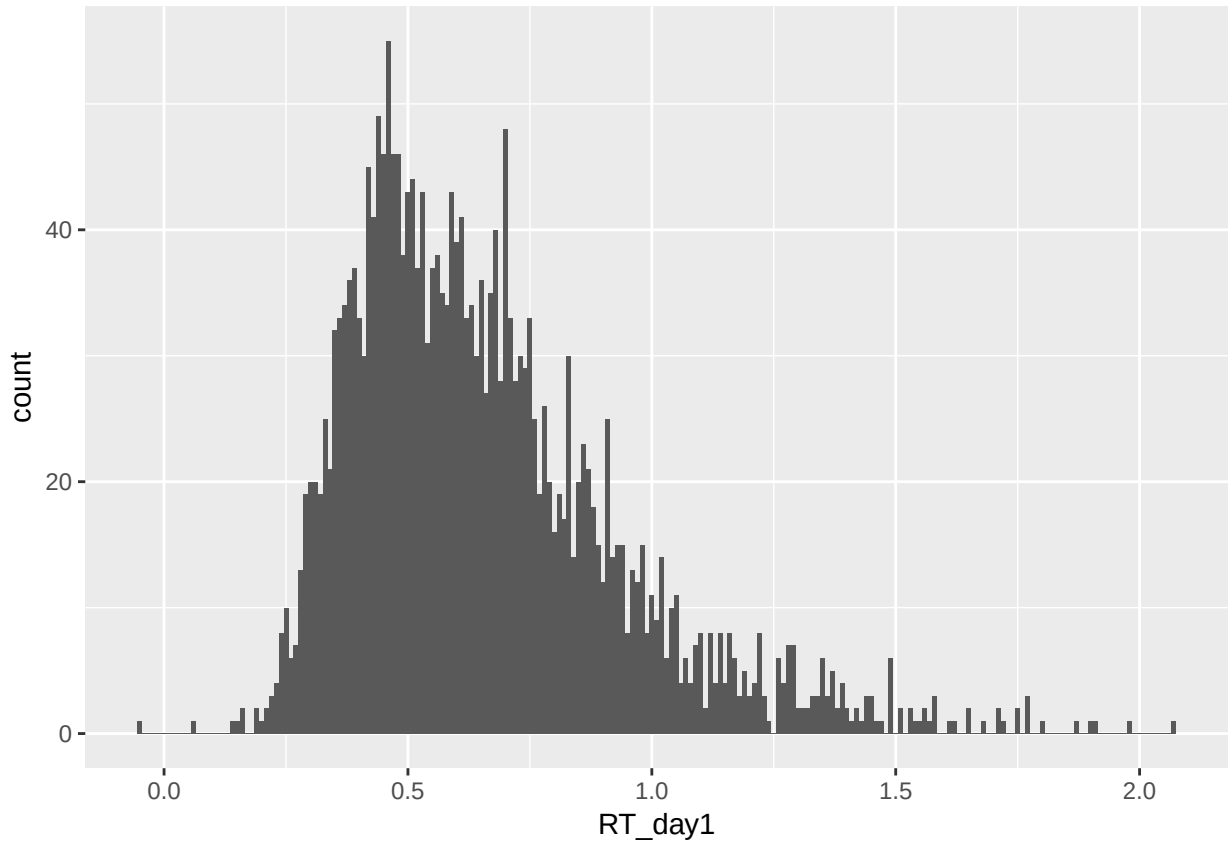

```
ggplot(data, aes(x = RT_day1)) + geom_histogram(binwidth = 0.1)
```



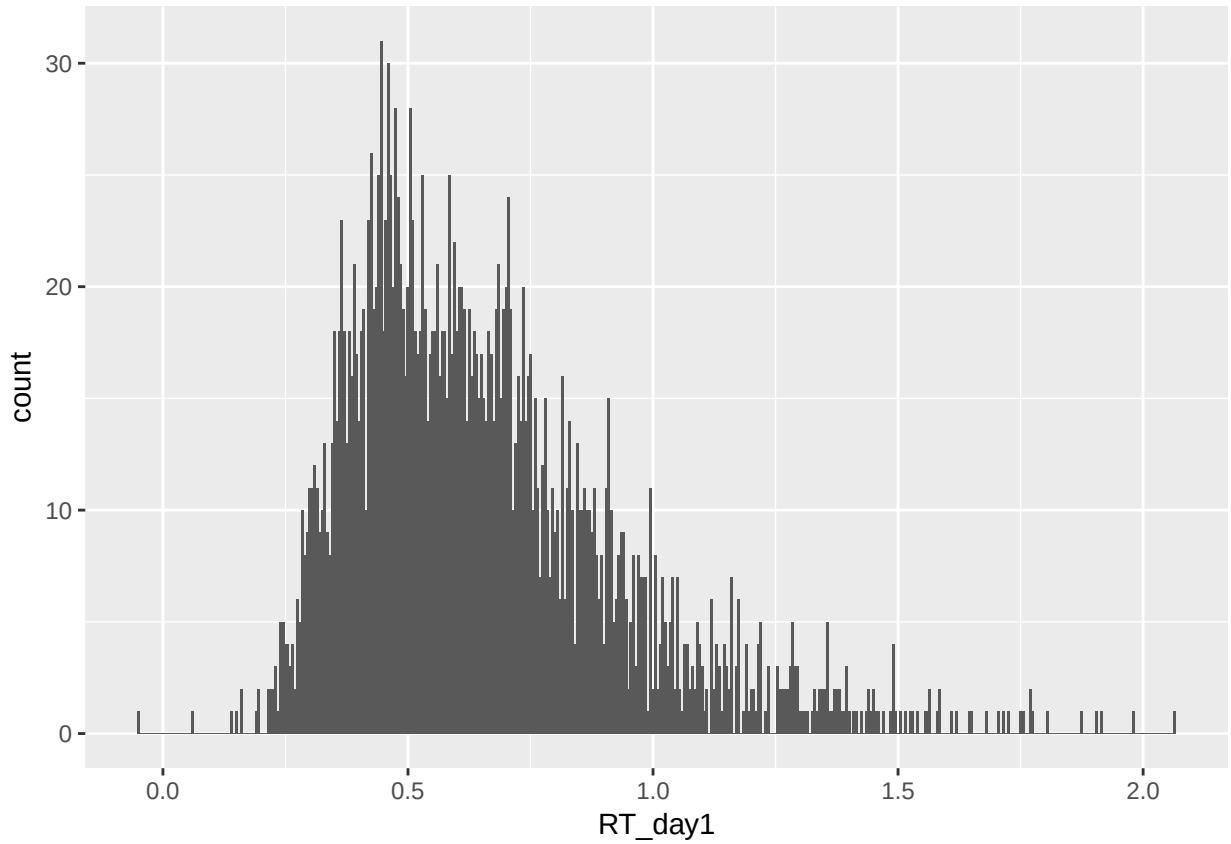

```
ggplot(data, aes(x = RT_day1)) + geom_histogram(binwidth = 0.05)
```



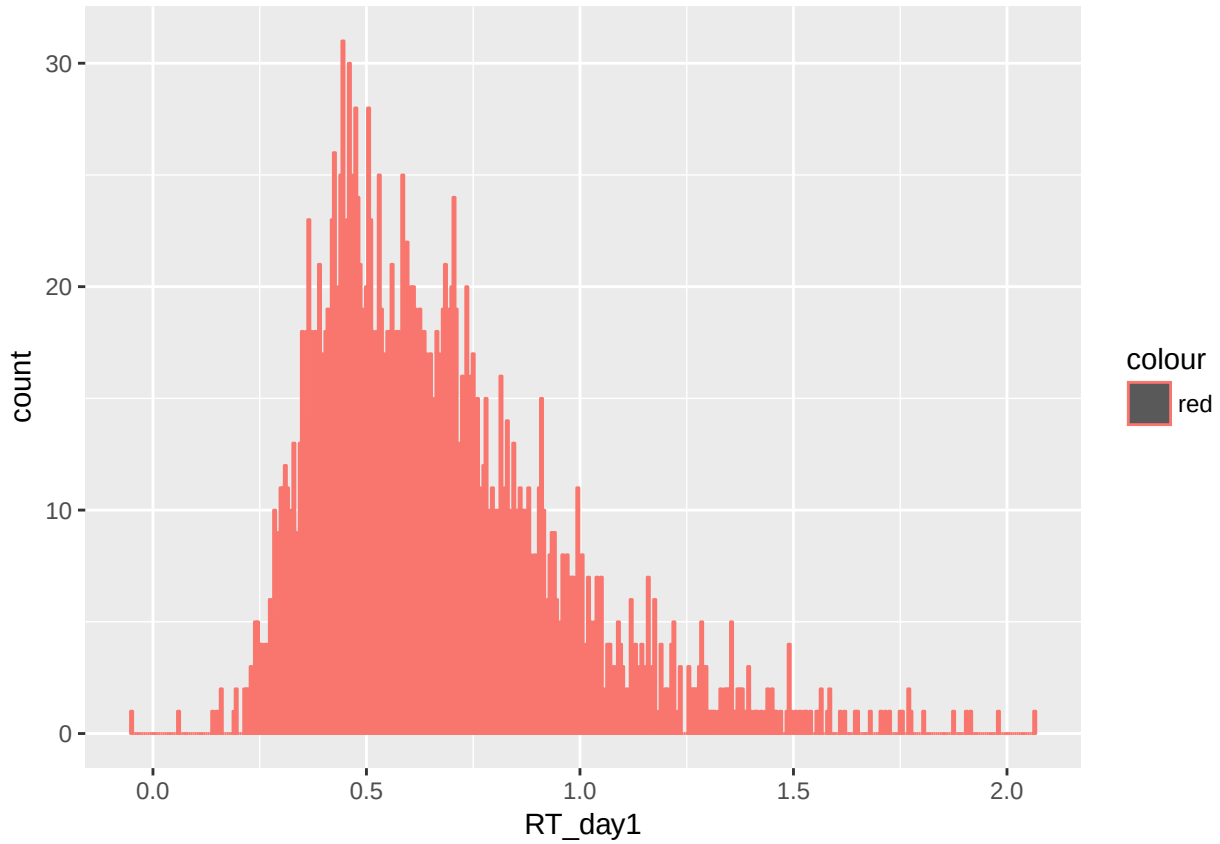

```
ggplot(data, aes(x = RT_day1)) + geom_histogram(binwidth = 0.01)
```



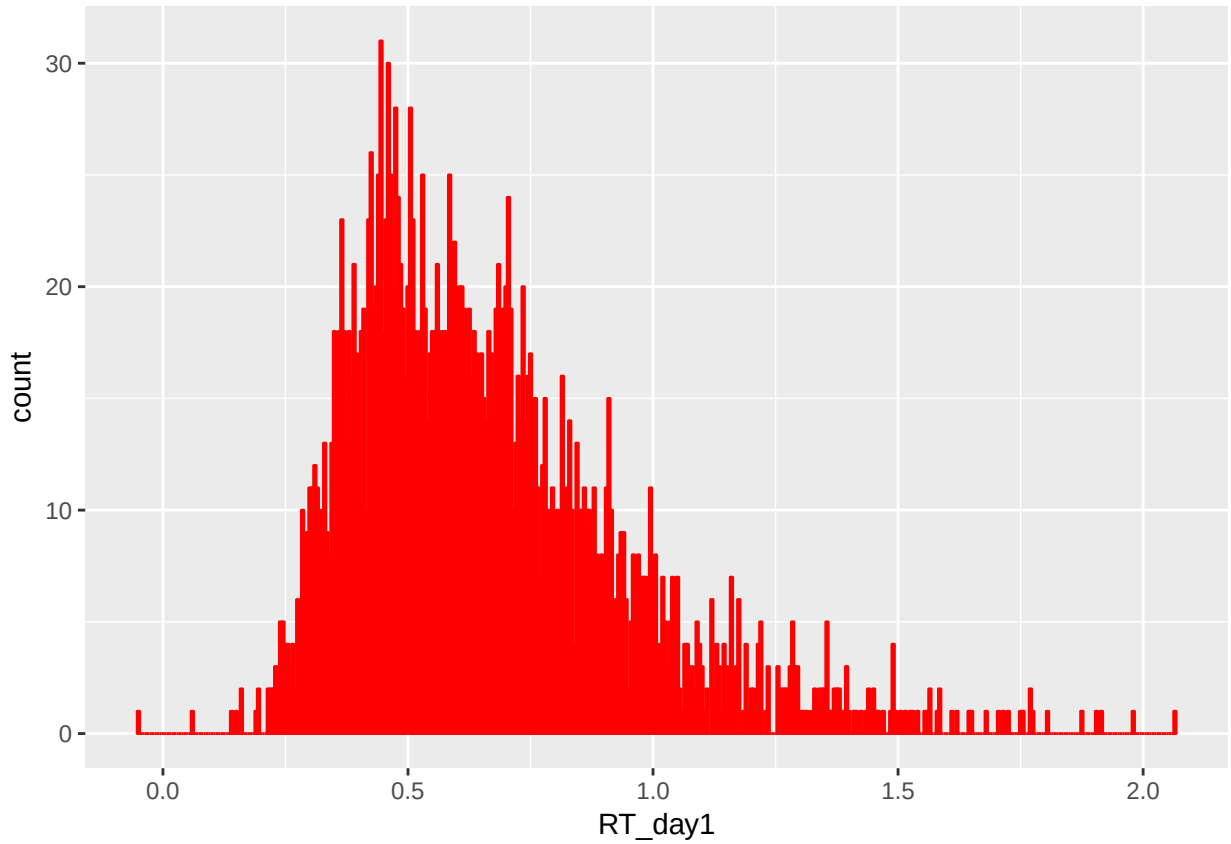

```
ggplot(data, aes(x = RT_day1)) + geom_histogram(binwidth = 0.005)
```



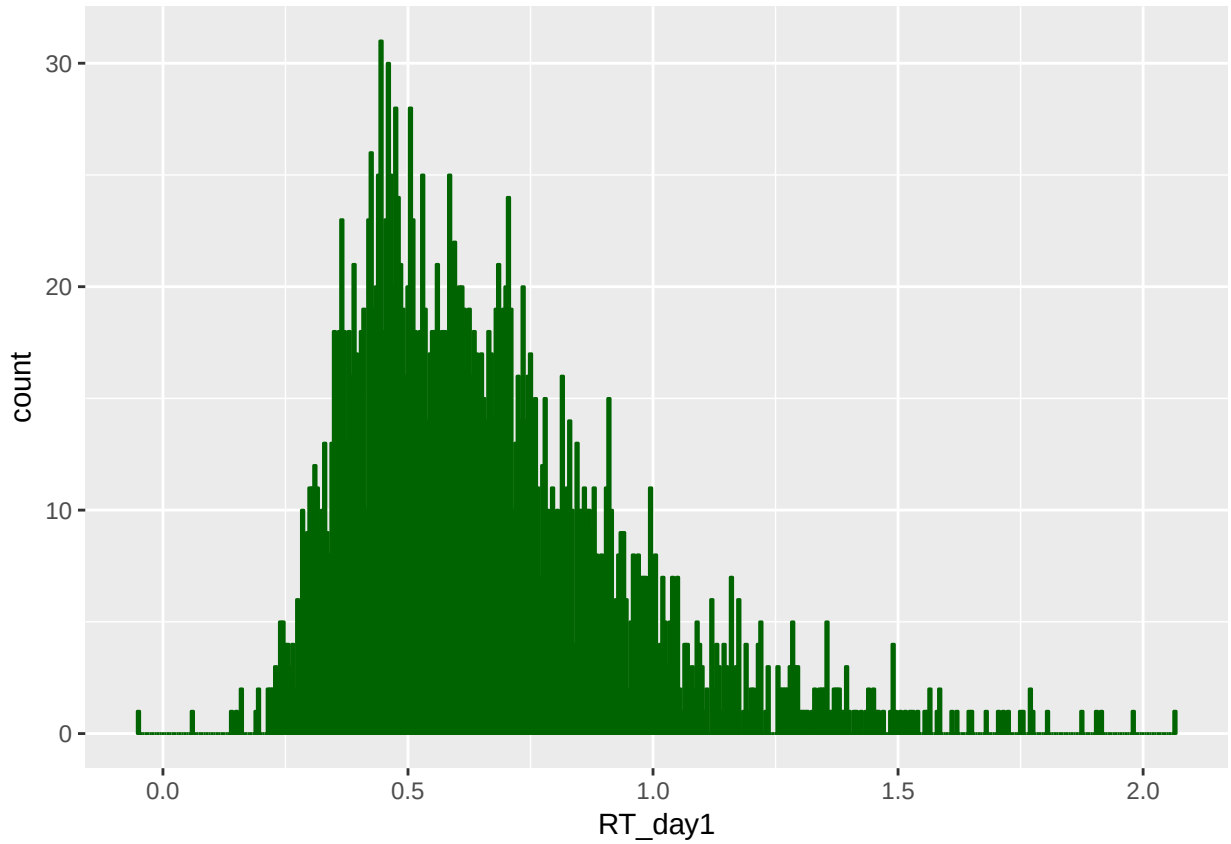

```
# We can also change colours here
ggplot(data, aes(x = RT_day1)) + geom_histogram(binwidth = 0.005, aes(colour="red"))
```



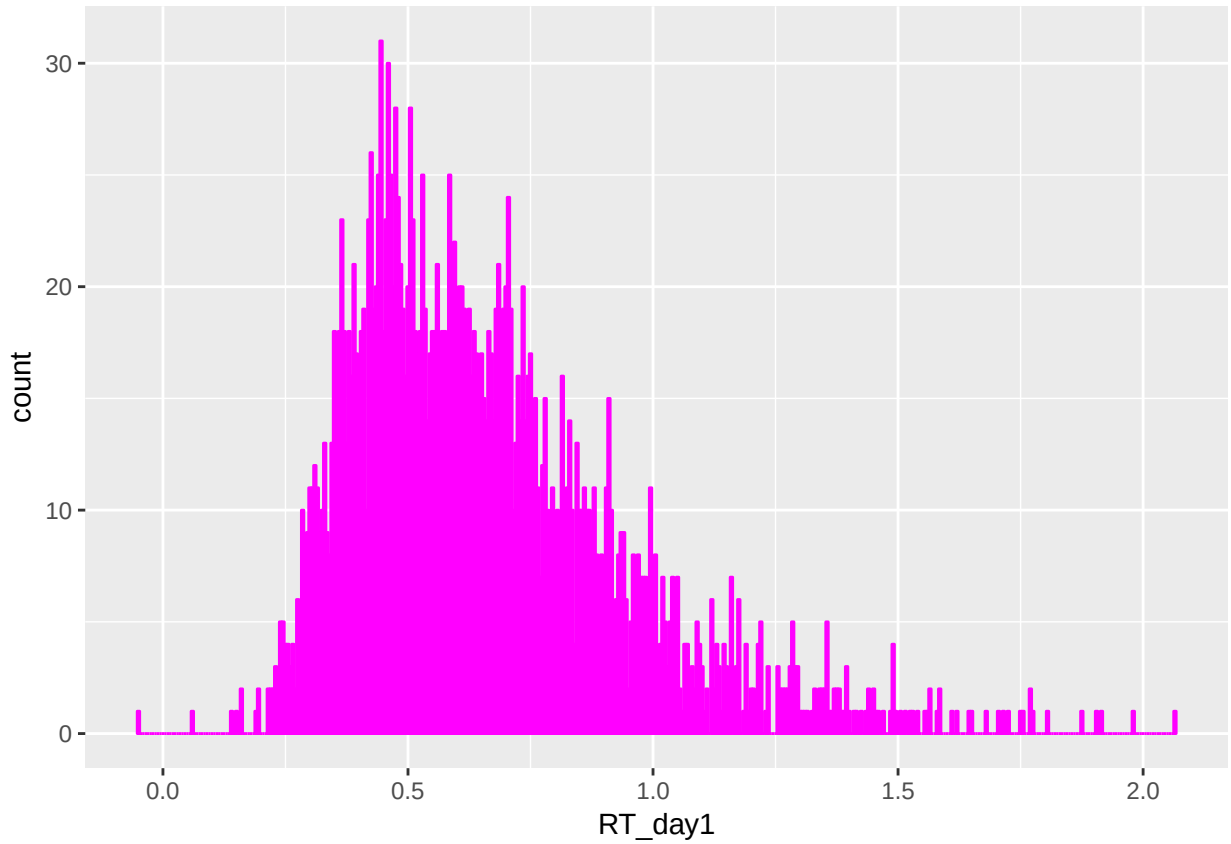

```
# And here, using aesthetic (aes) mapping
ggplot(data, aes(x = RT_day1)) + geom_histogram(binwidth = 0.005, colour="red")
```



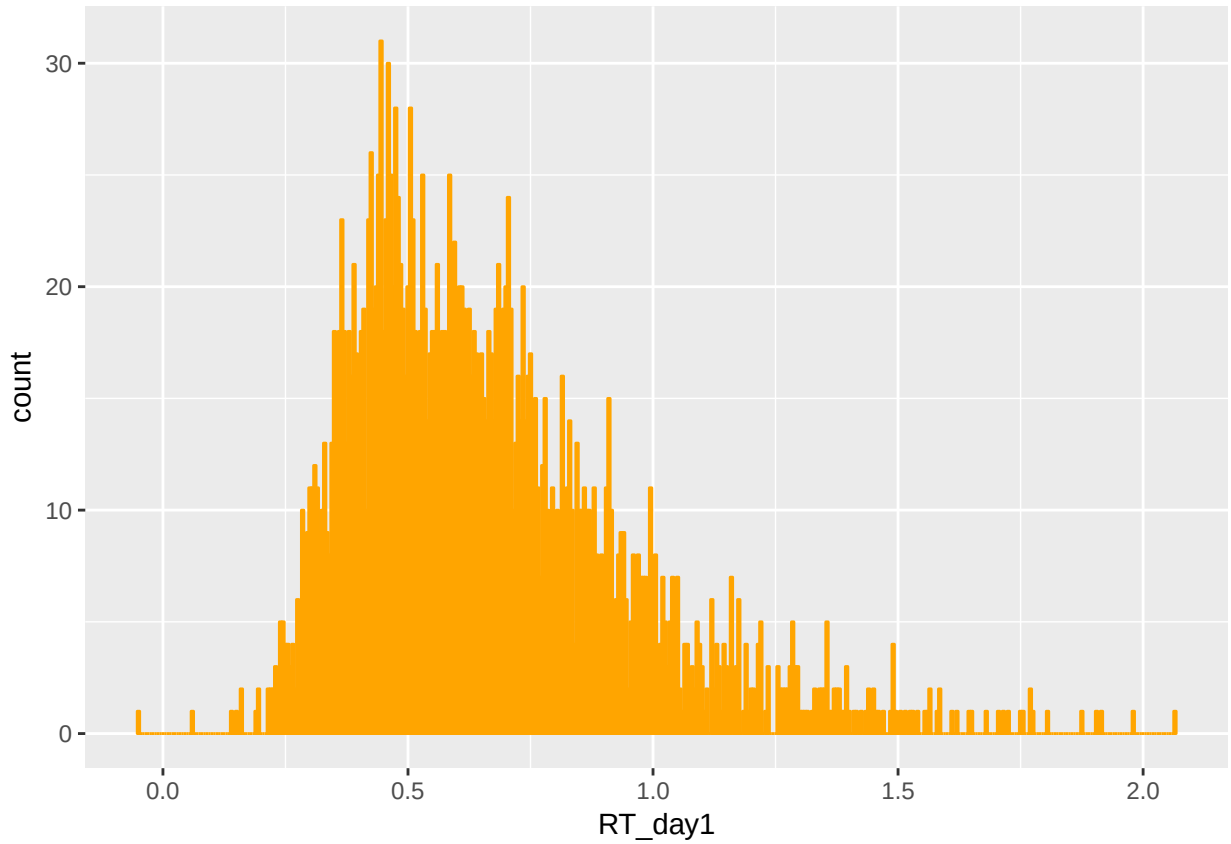

```
ggplot(data, aes(x = RT_day1)) + geom_histogram(binwidth = 0.005, colour="darkgreen")
```



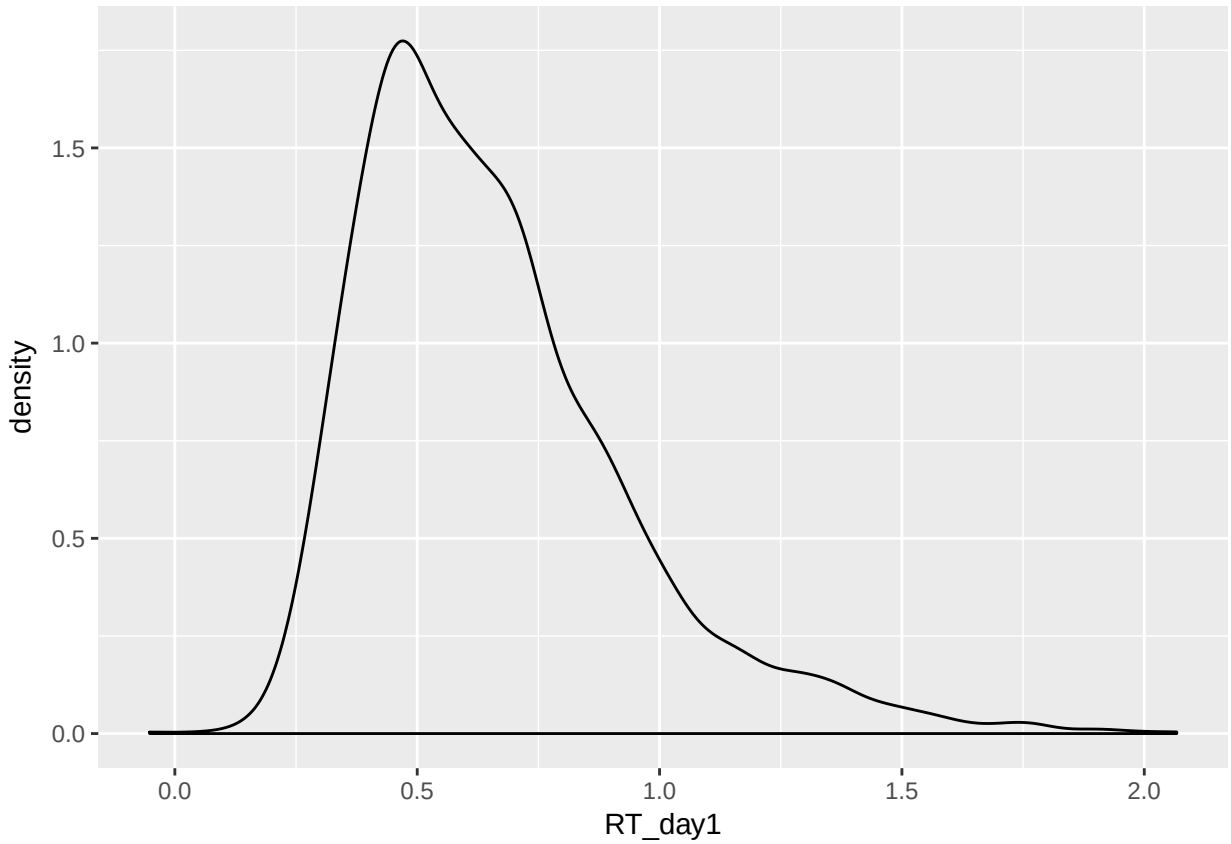

```
ggplot(data, aes(x = RT_day1)) + geom_histogram(binwidth = 0.005, colour="magenta")
```



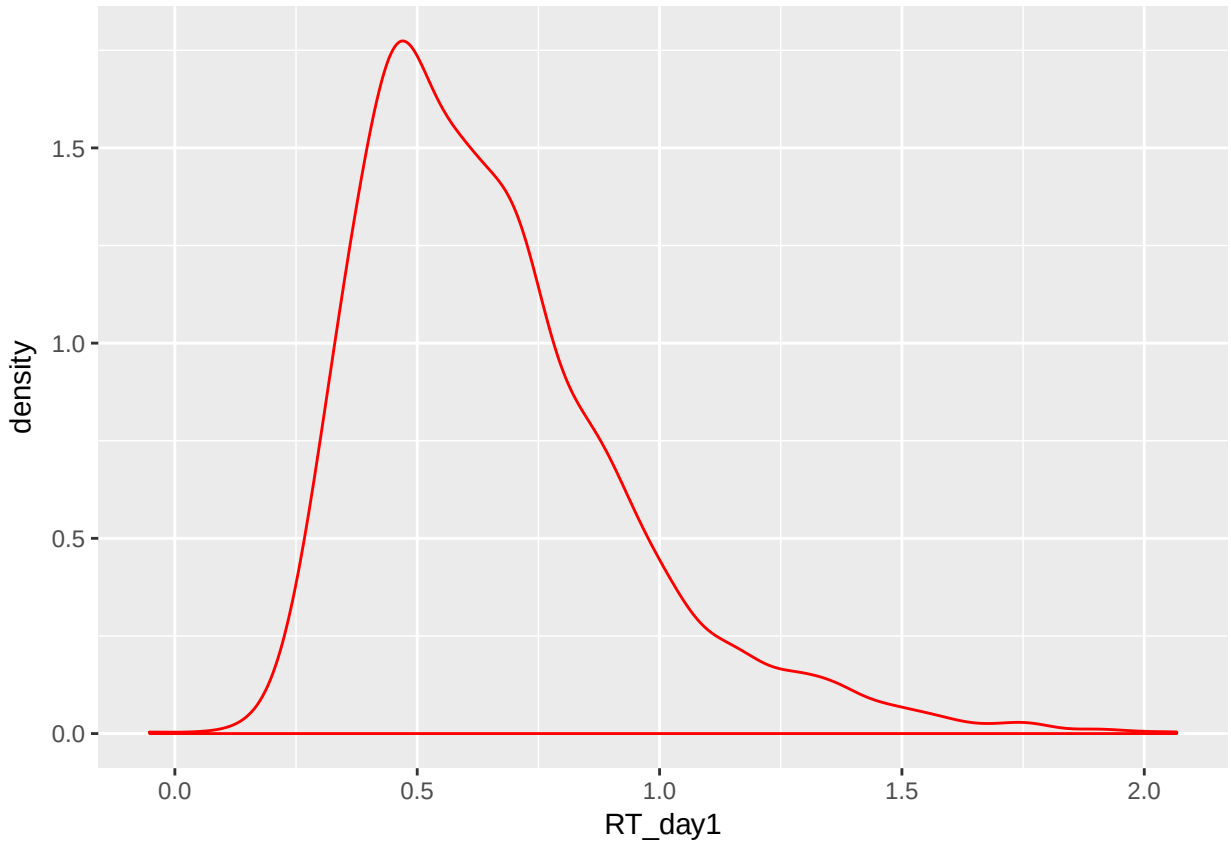

```
ggplot(data, aes(x = RT_day1)) + geom_histogram(binwidth = 0.005, colour="orange")
```



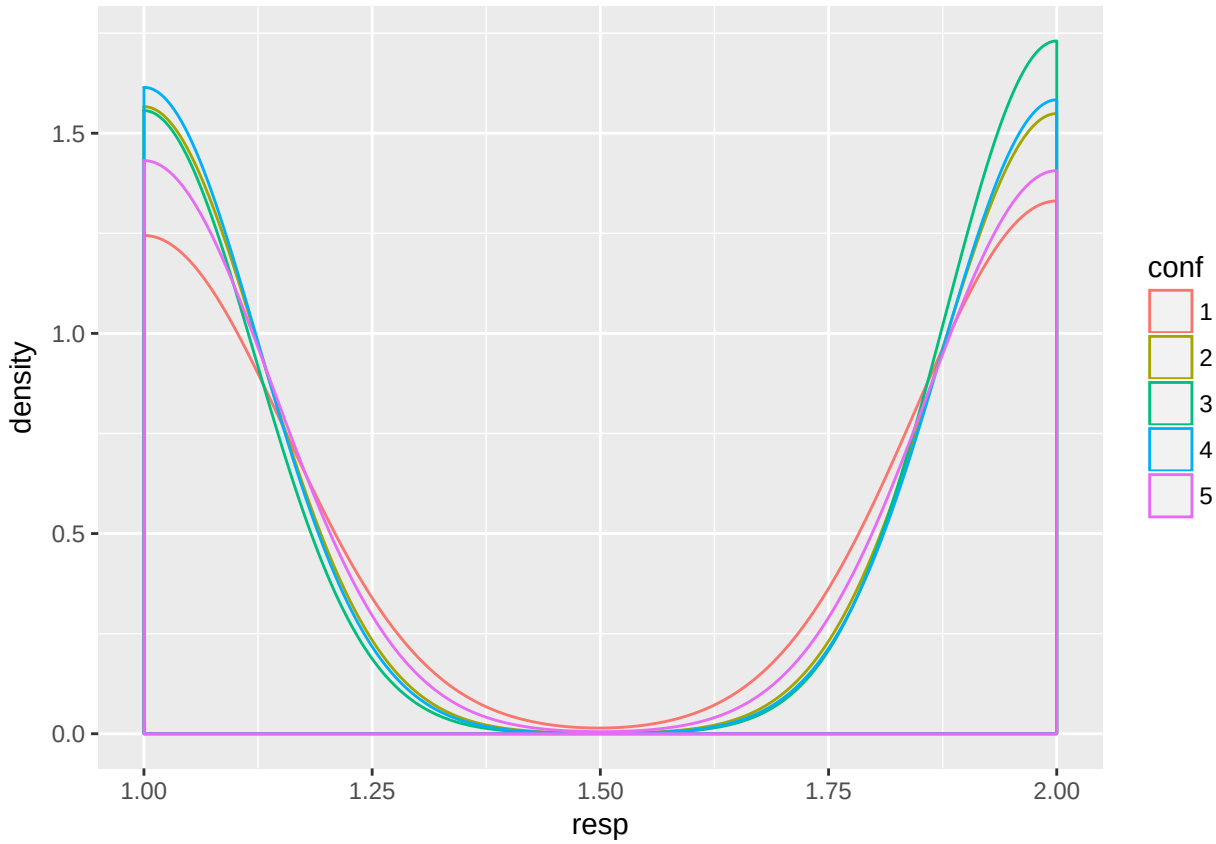

```
# We could also get a density plot of our RT's
ggplot(data, aes(x = RT_day1)) + geom_density()
```



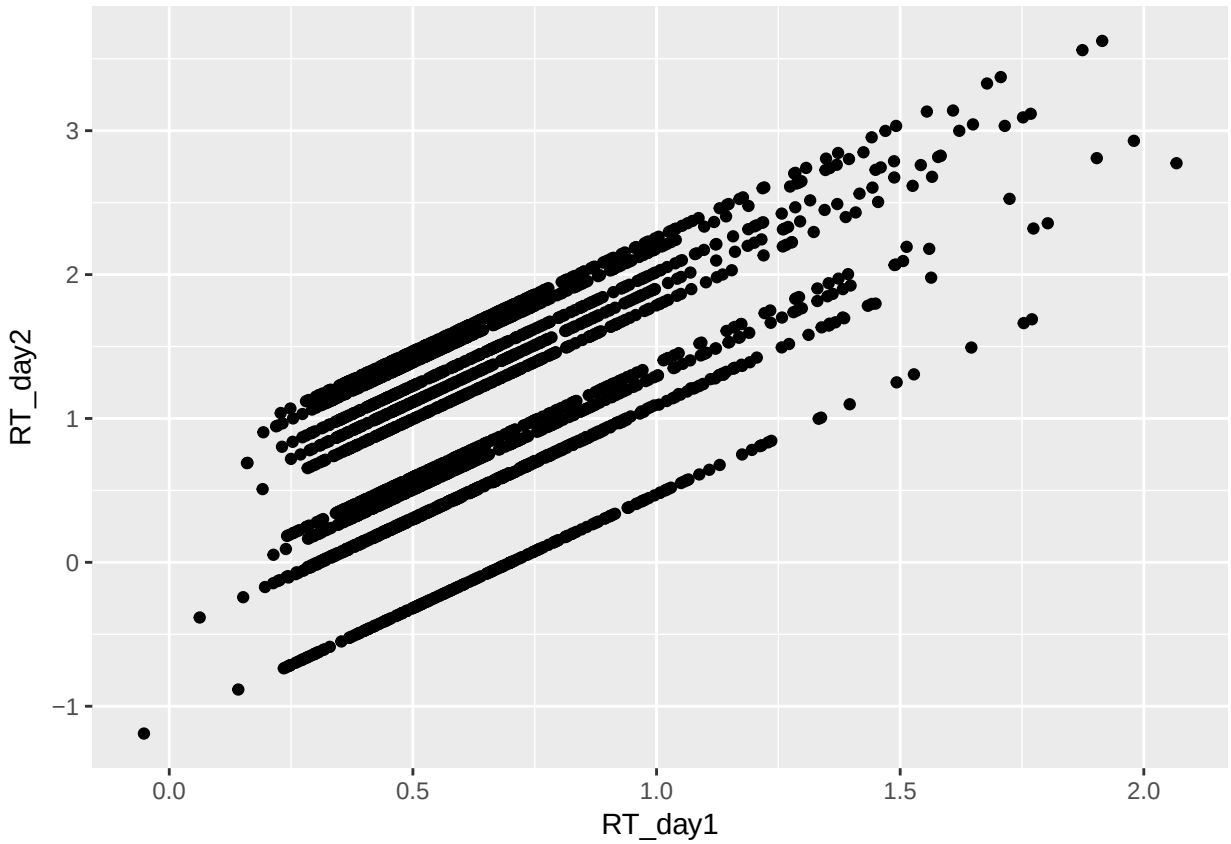

```
# We can also change the colours on this
ggplot(data, aes(x = RT_day1)) + geom_density(colour = "red")
```



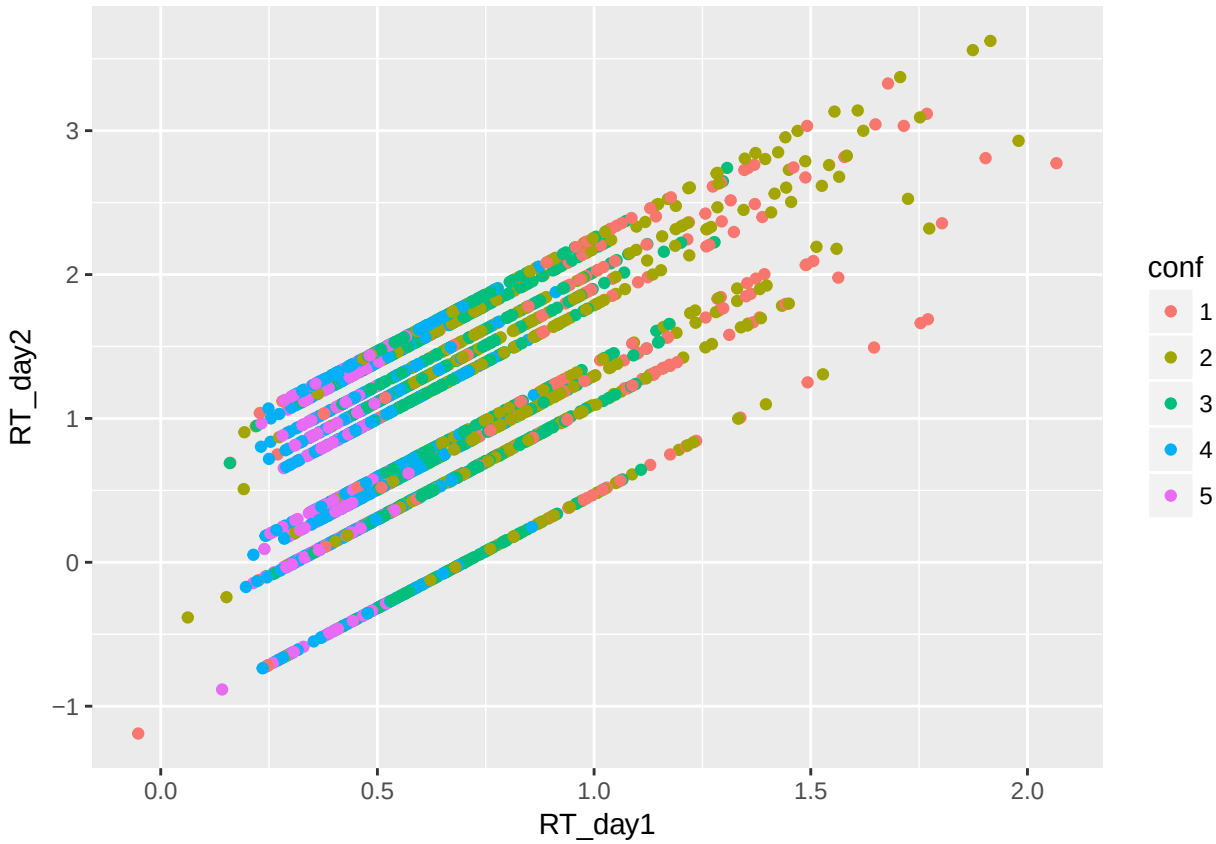

```
# We could get our responses (face vs house)
ggplot(data, aes(x = resp)) + geom_density(aes(colour = conf))
```



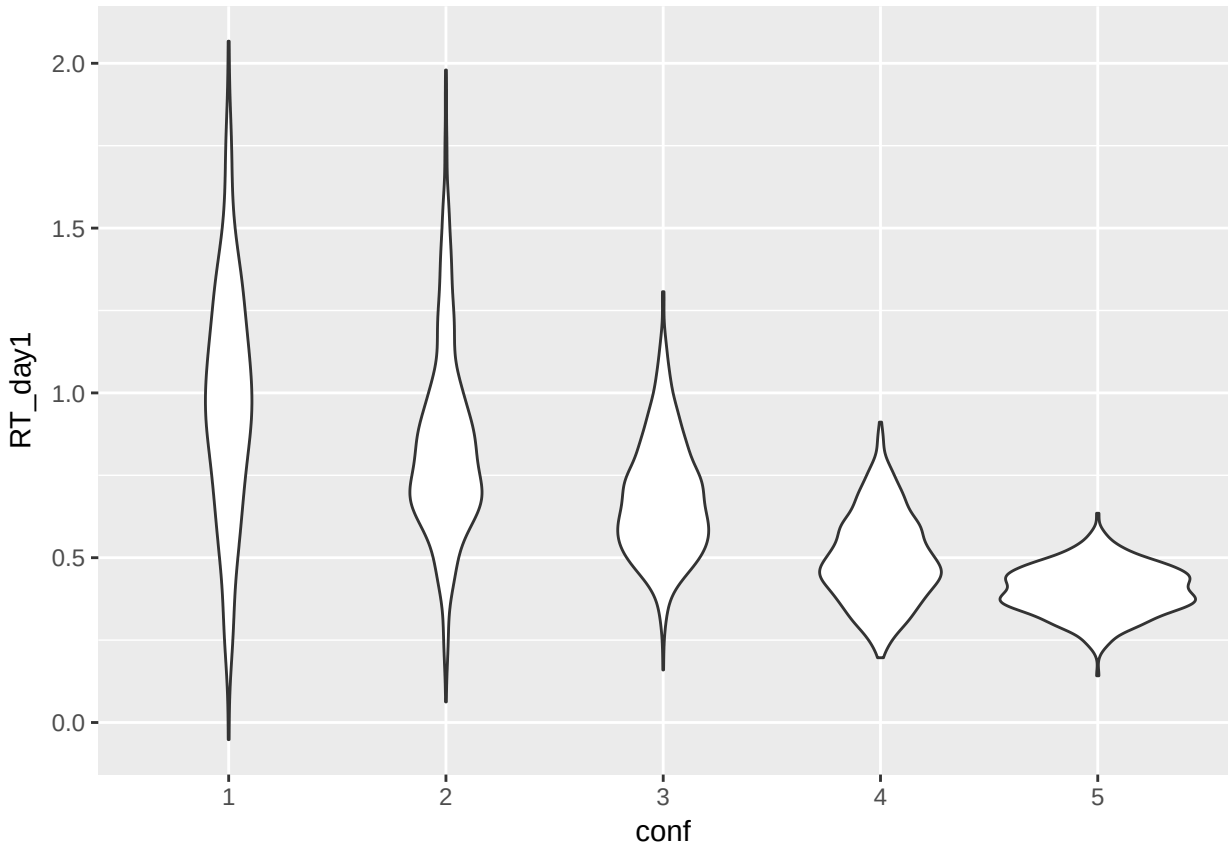

```
# What if we want a scatterplot?
scatter_RT <- ggplot(data, aes(x = RT_day1, y = RT_day2))  + geom_point()
scatter_RT
```



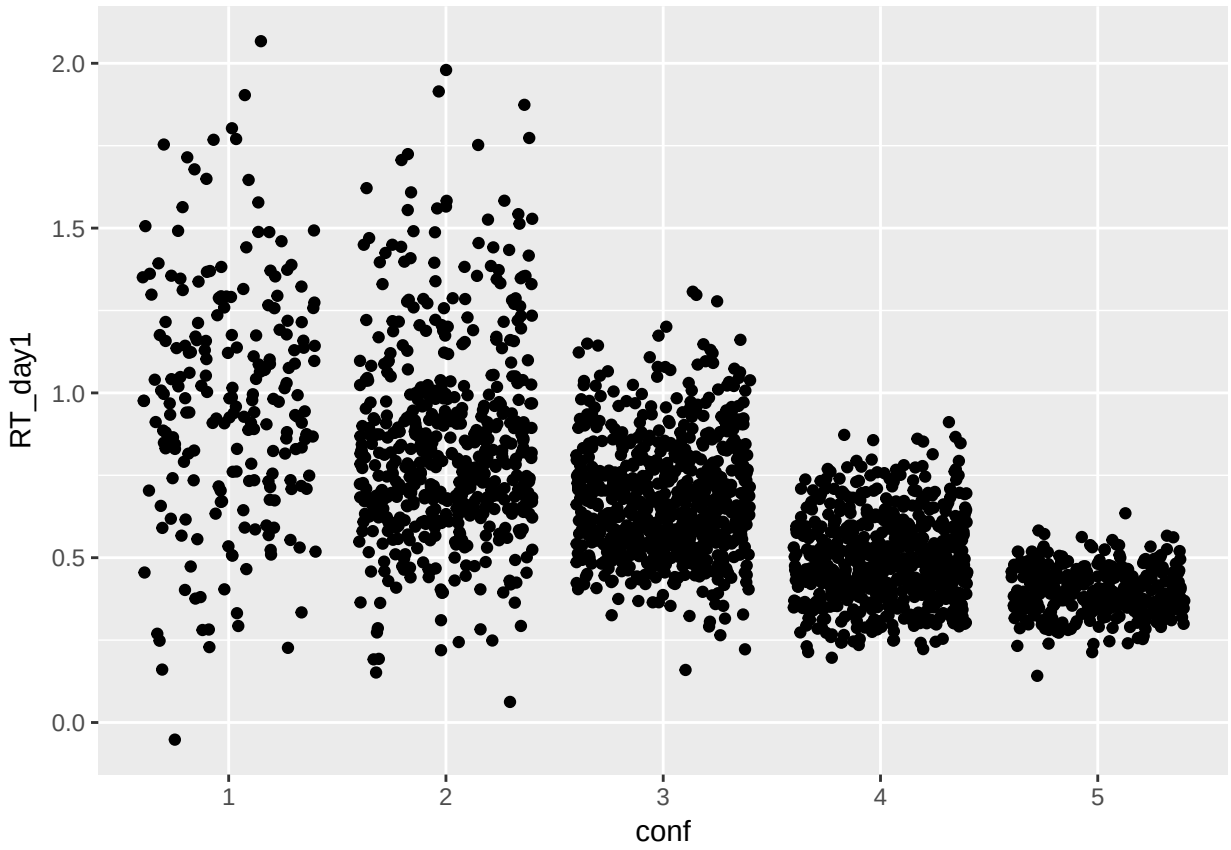

```
scatter_RTcol <- ggplot(data, aes(x = RT_day1, y = RT_day2)) + geom_point(aes(colour = conf))
scatter_RTcol
```



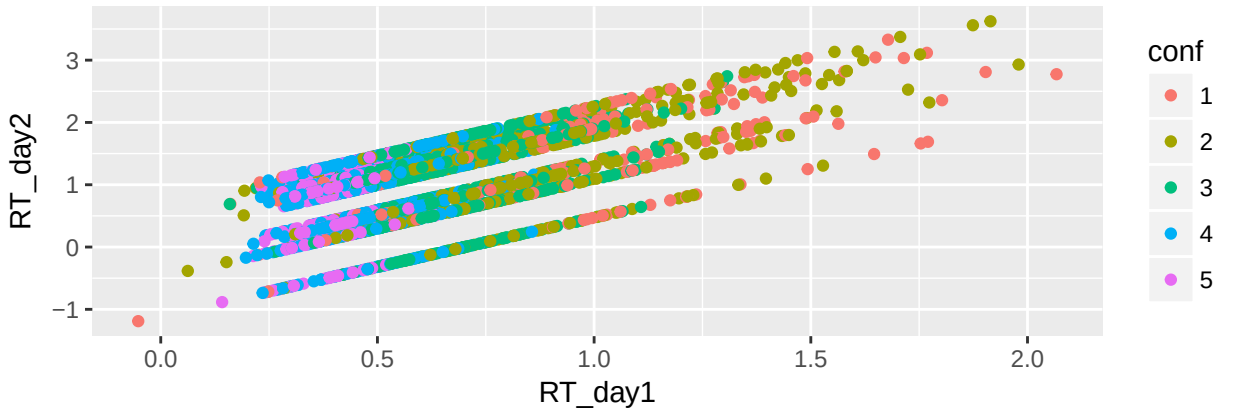

```
violin_RT  <- ggplot(data, aes(x = conf, y = RT_day1)) + geom_violin()
violin_RT
```

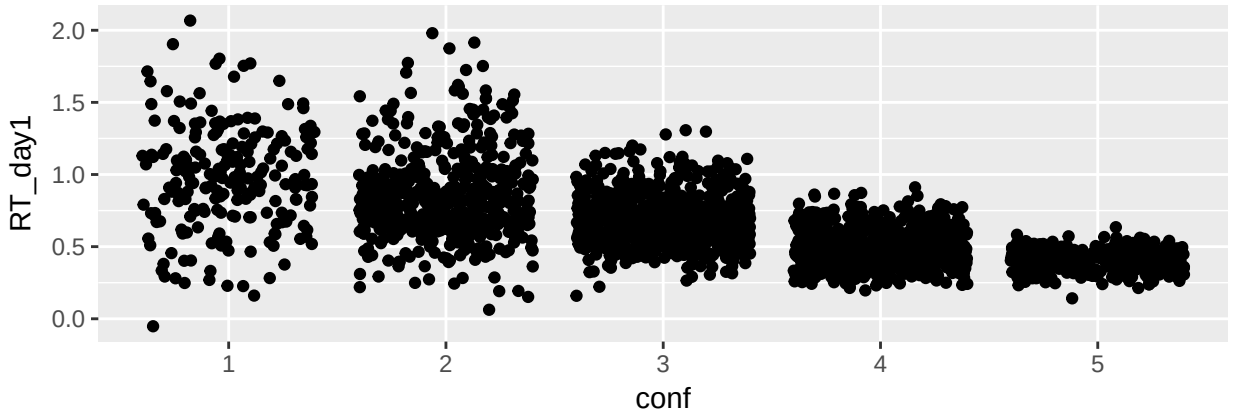



```
jitter_RTcol <- ggplot(data, aes(y = RT_day1, x = conf)) + geom_jitter()
jitter_RTcol
```



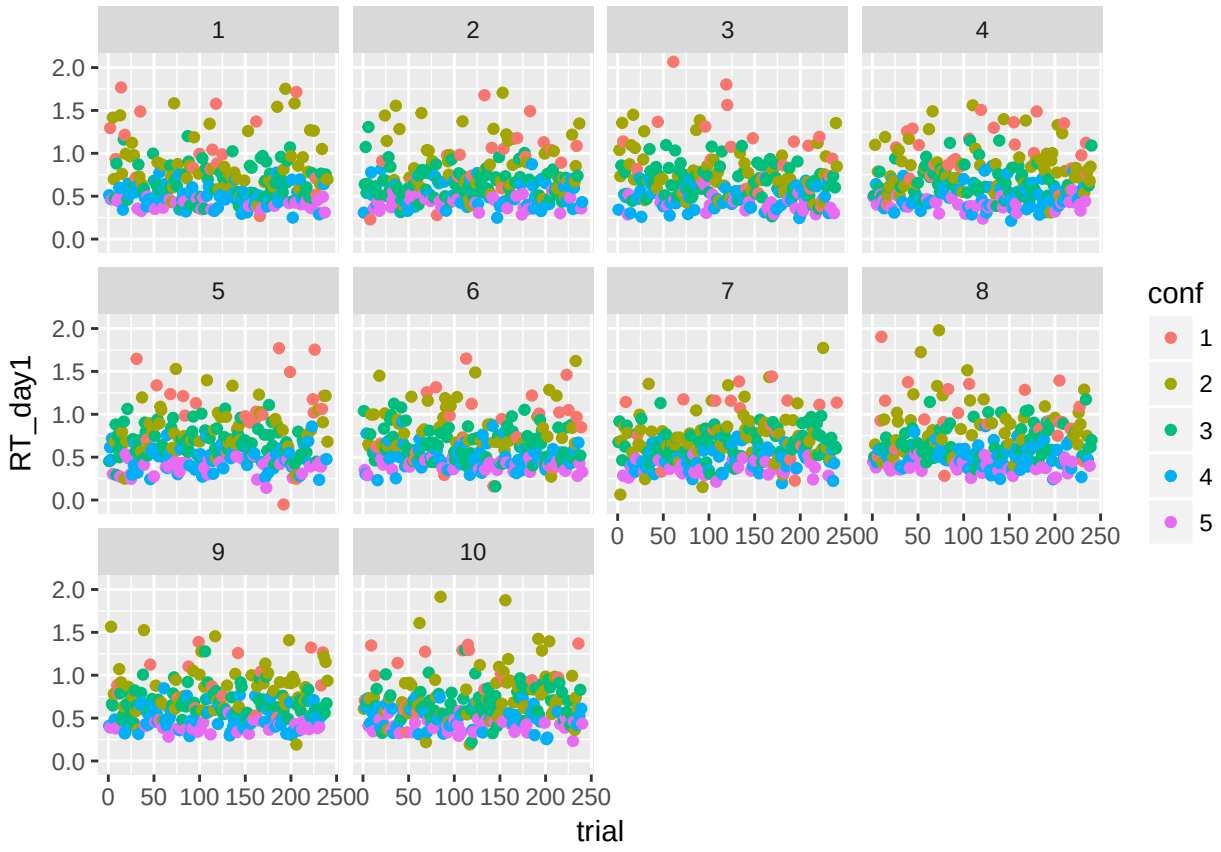

```
# I could put these two together in a grid
library(gridExtra)
grid.arrange(scatter_RTcol, jitter_RTcol, ncol = 1, nrow = 2)
```



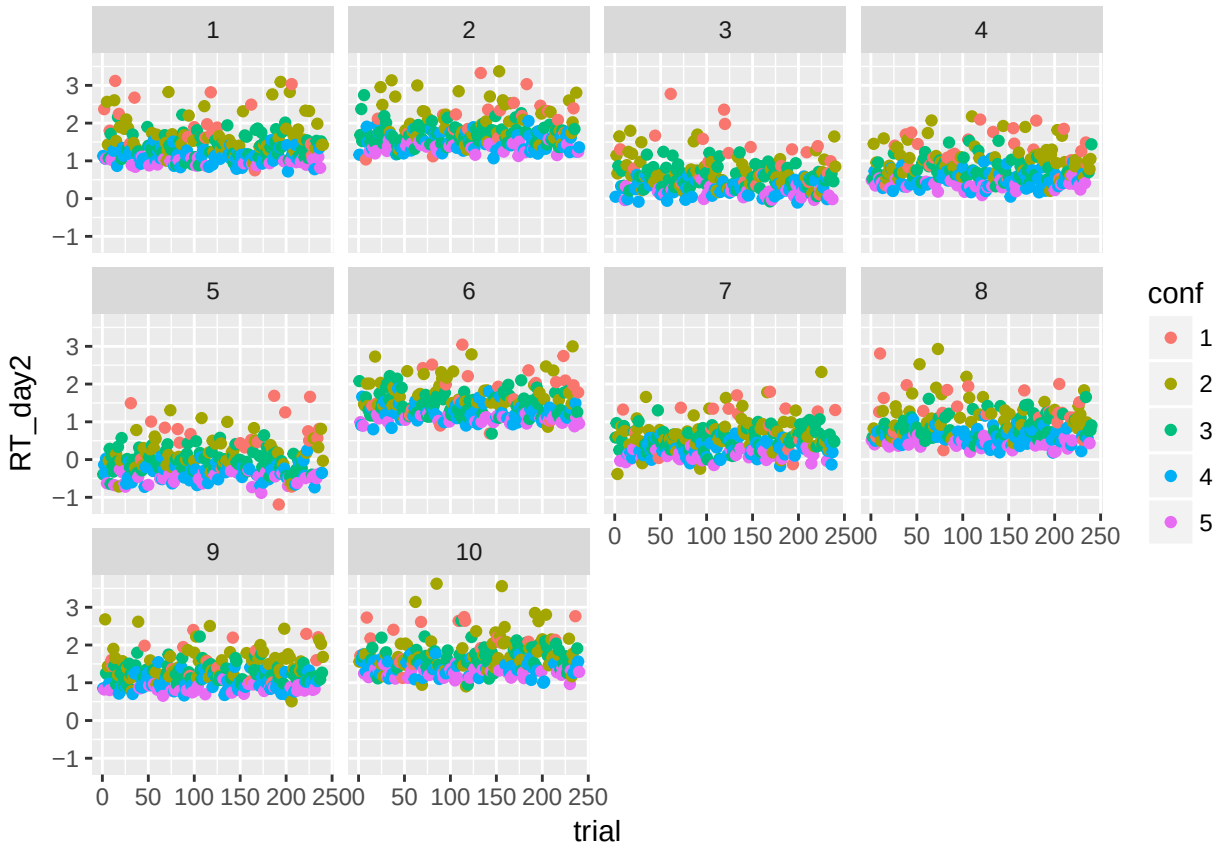



```
# To use for line and point colors, add
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73",
               "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
# grey, orange, blue, green, yellow, teal, red, pink
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73",
                "#F0E442", "#0072B2", "#D55E00", "#CC79A7")
# black, orange, blue, green, yellow, teal, red, pink
```

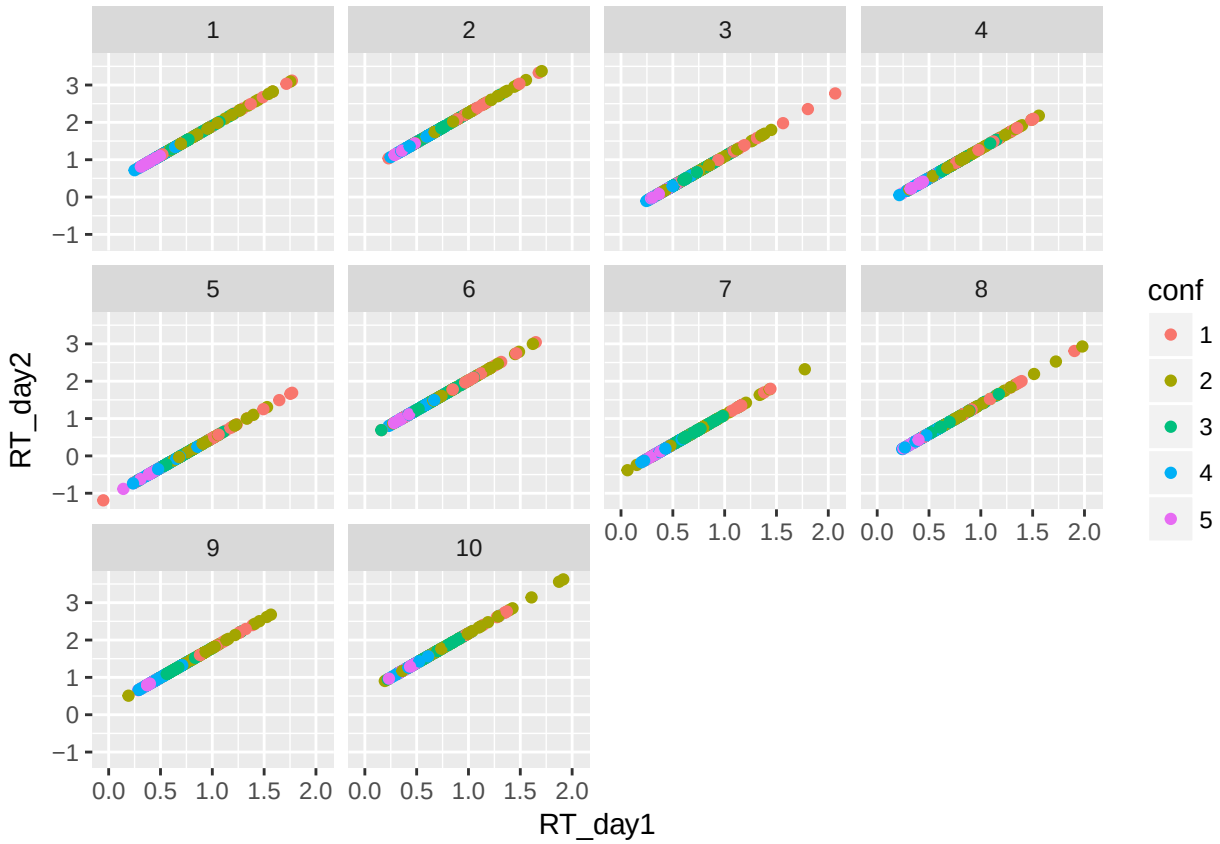
```
# What if I want subjects plotted separately? This section does the basics of what you have to add
# the only thing you need to add, is facet_wrap(~thing you want to group by) - in our case subject numb

# We could also do this - tell R what colour we want for the points
ggplot(data, aes(x = trial, y = RT_day1)) + geom_point(aes(colour = conf)) + facet_wrap(~subj_no)
```



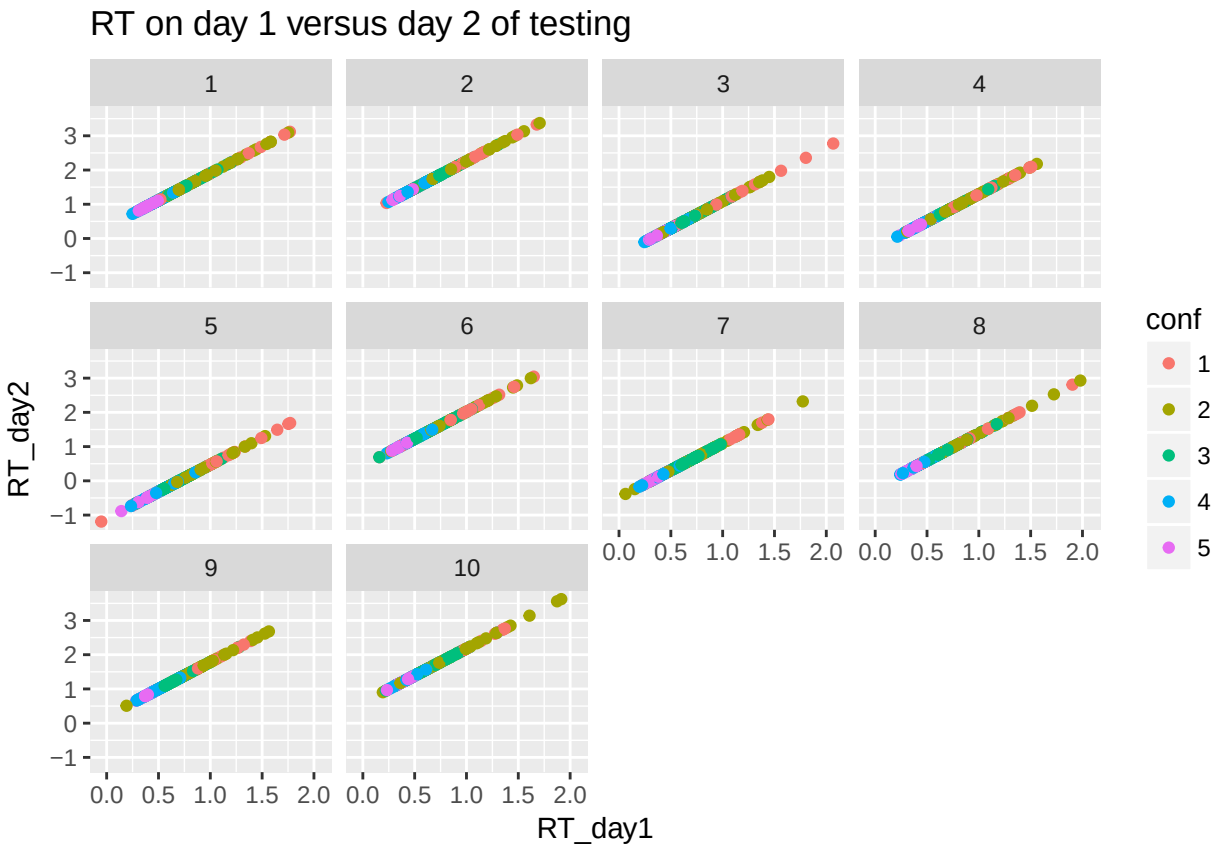

```
ggplot(data, aes(x = trial, y = RT_day2)) + geom_point(aes(colour = conf)) + facet_wrap(~subj_no)
```



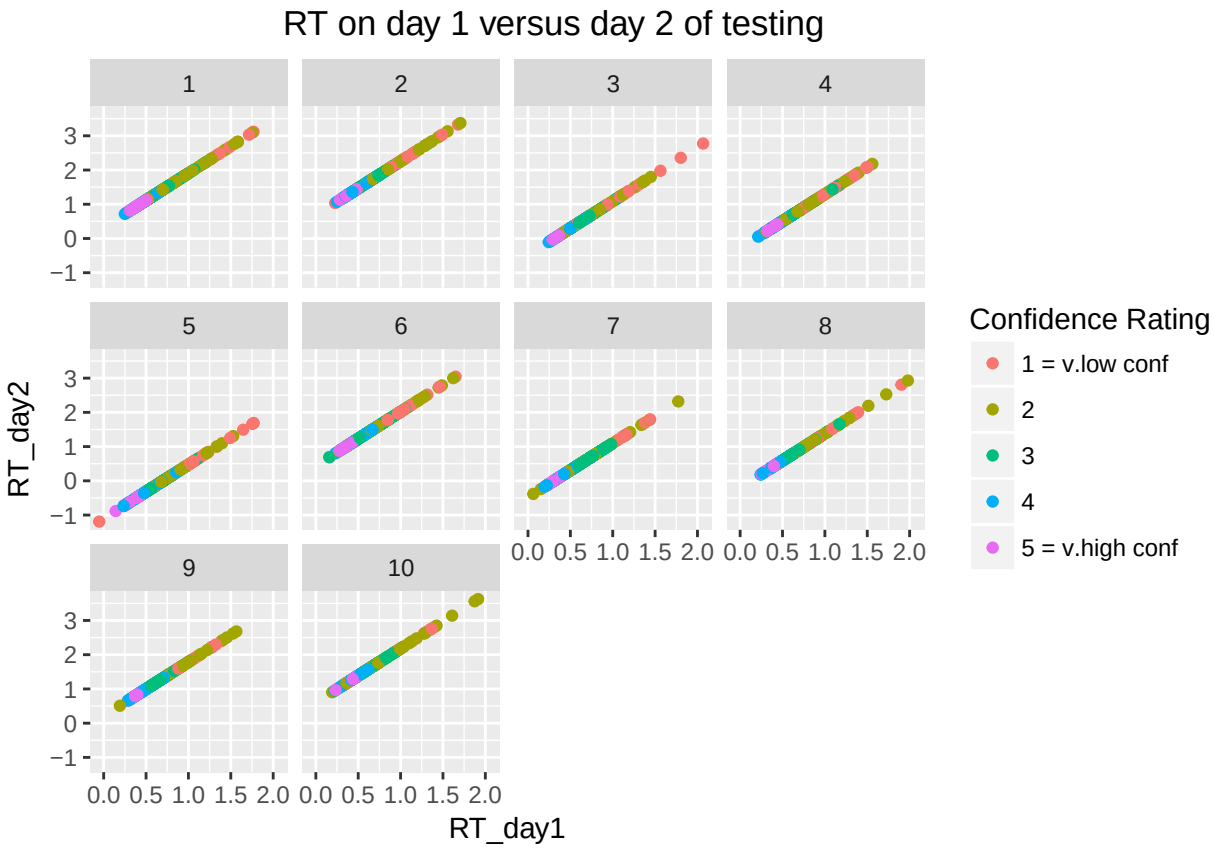

```
ggplot(data, aes(x = RT_day1, y = RT_day2)) + geom_point(aes(colour = conf)) + facet_wrap(~subj_no)
```



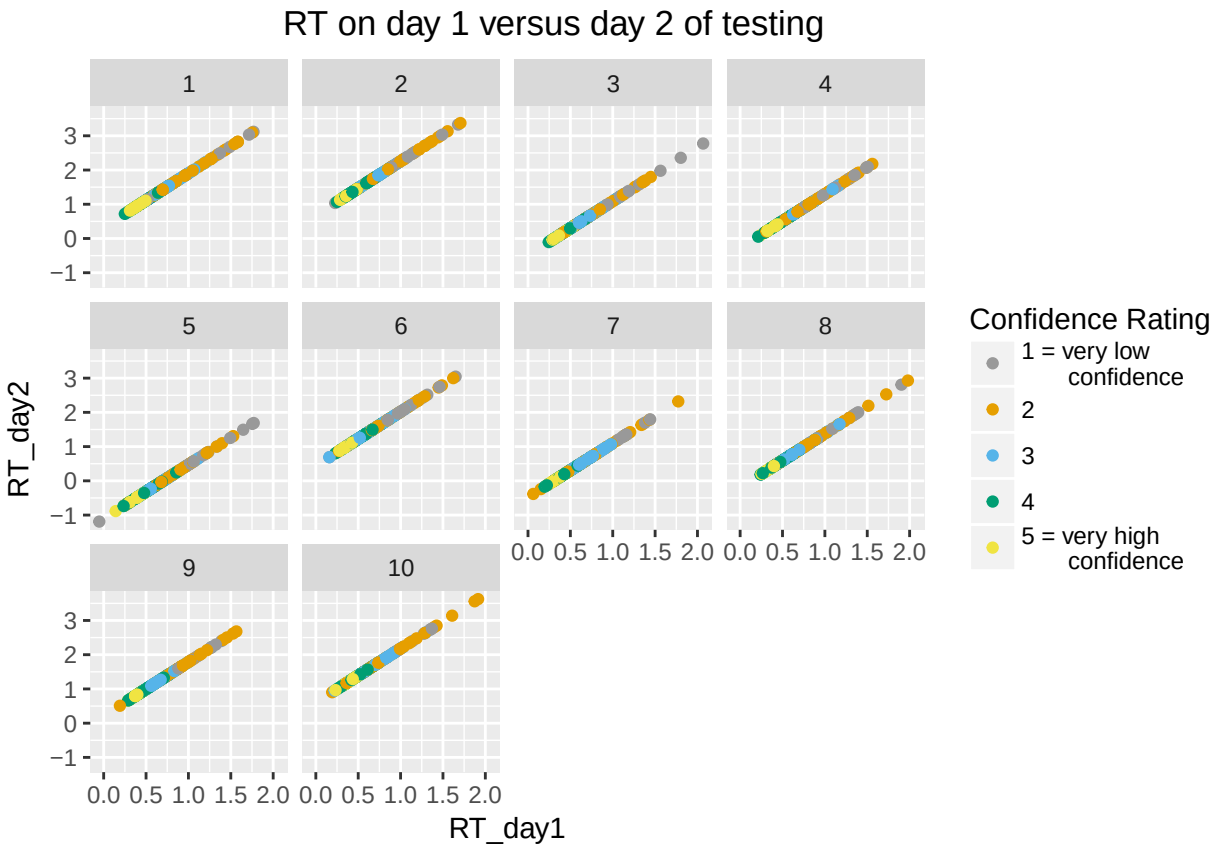

```
# We can add some colour, axes, and a title
ggplot(data, aes(x = RT_day1, y = RT_day2)) + geom_point(aes(colour = conf)) +
  facet_wrap(~subj_no) +
  xlab("RT_day1") + ylab("RT_day2") +
  ggtitle("RT on day 1 versus day 2 of testing")
```




```
# We can also add a legend
ggplot(data, aes(x = RT_day1, y = RT_day2)) + geom_point(aes(colour = conf)) +
  facet_wrap(~subj_no) +
  xlab("RT_day1") + ylab("RT_day2") +
  ggtitle("                    RT on day 1 versus day 2 of testing")  +
  scale_colour_discrete(name="Confidence Rating",
                        labels=c("1 = v.low conf", "2", "3","4","5 = v.high conf"))
```




```
# We can also add a legend and a colourblind palette or a special palette we added before
# MAKE SURE you have run (played) the 'set_colour_palette' section
# The thing that changes for the legend is we have something called 'values = cbPalette'
# This just tells ggplot that the legend (scale_colour_manual) should have values of whatever the colou
# We can give a bit more detail in the legend
ggplot(data, aes(x = RT_day1, y = RT_day2)) + geom_point(aes(colour = conf)) +
  facet_wrap(~subj_no) +
  xlab("RT_day1") + ylab("RT_day2") +
  ggtitle("                    RT on day 1 versus day 2 of testing")  +
  scale_colour_manual(values = cbPalette,
                      name="Confidence Rating",
                        labels=c("1 = very low \n       confidence", "2", "3","4","5 = very high \n
```




```
# What if I want to use different colours?
# change scale_fill_manual(values=cbbPalette) (do it above) change cb to cbb (because we have two diffe
# If you don't include the 'values = cbPalette' part, ggplot will choose colours for you.
```

```
# Changing the size of stuff in your plot to make it easier to read (esp for publications or powerpoint

# We can make it bigger - see the stuff that starts with 'theme' which allows us to change text size in
# Notice here I am ASSIGNING my plot to 'figure1' - this means the graph stores ALL of the information
# 'figure1' so when I want to look at the plot, I have to type 'figure1' again
figure1 <- ggplot(data, aes(x = RT_day1, y = RT_day2)) + geom_point(aes(colour = conf)) +
  facet_wrap(~subj_no) +
  xlab("RT_day1") + ylab("RT_day2") +
  ggtitle("                    RT on day 1 versus day 2 of testing")  +
  scale_colour_discrete(name="Confidence Rating",
                        labels=c("1 = very low \n       confidence", "2", "3","4","5 = very high \n
  theme(plot.title = element_text(size = 13, face = "italic")) +
  theme(axis.title.x = element_text(size=10)) + theme(axis.title.y = element_text(size=10)) +
  theme(legend.title = element_text(size=10)) + theme(legend.text = element_text(size=10))

# We can add a caption down the bottom (just keep adding stuff to the cake/plot called figure1 above)
figure1 <- figure1 + labs(caption = "Figure 1: data from the study that chris made up")

# I need to type 'figure1' to see the figure because we stored all
# of the information in figure1 variable
figure1 # it will appear below, so pretty!
```

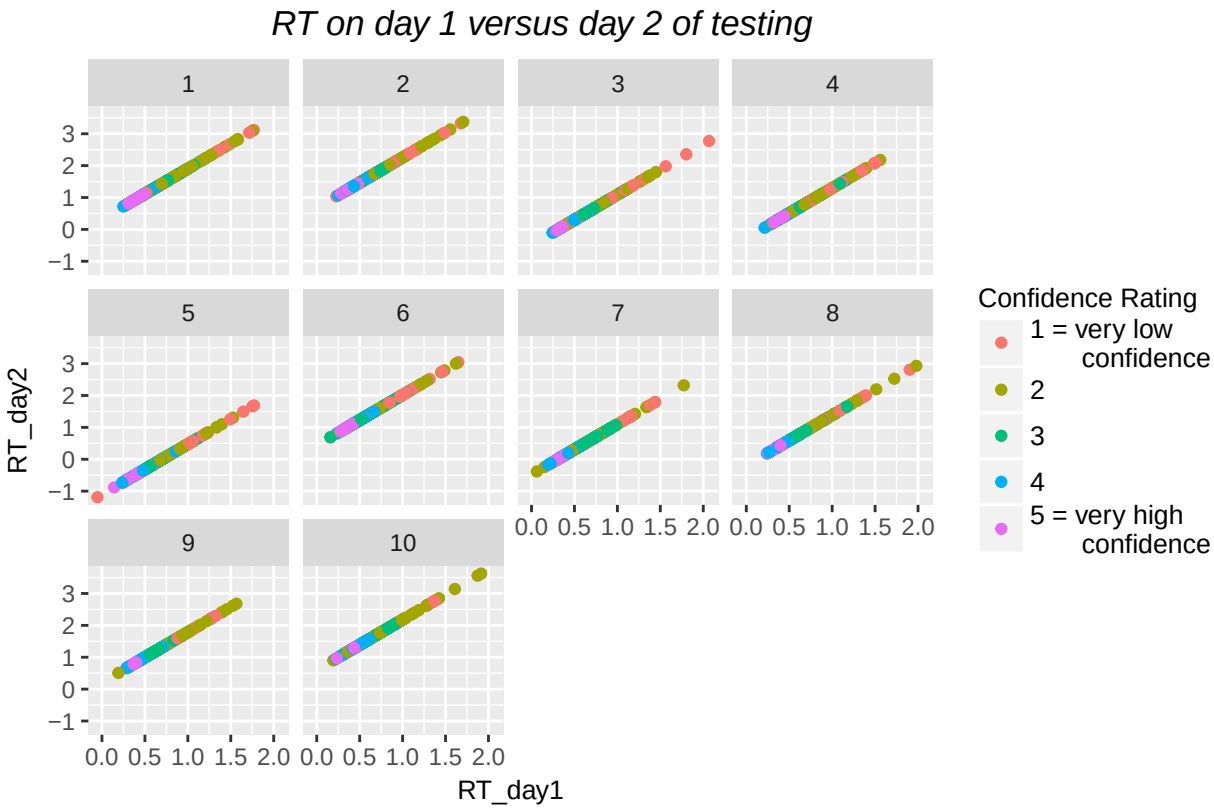


Figure 1: data from the study that chris made up

```
# I have commented this out so it doesn't stuff your script up, because you need to change some things
# as you are storing your script in a different file to mine

# Simple way of saving - change the path to where you want files to be
# We can automatically specify at the beginning of the script
# a place to store files, and you wont have to keep typing the path every time. We did this!
# we said data_dir = "~/Dropbox/GRIPS/PhD_camp/ggplot_workshop/" at the beginning this means we can als
# get a little fancy and do this
# in the section below that says:  filename =
# we can do-   file.path(data_dir,"figure1.jpg") - this creates a path to the file by taking the path i
# data_dir and combining it with a string ("figure1.jpg") to create the same path as the one above
# This is a good way to do this because we dont have to keep writing our whole path every time we save

# ---- uncomment the 4 lines below
ggsave(filename = file.path(data_dir,"figure1.jpg"),  # needs to be a full path
       plot = figure1,
       width = 10,
       height = 5)
```