Data Visualization in R - basic examples

## Learning Objectives

* Learn basic chart types using ggplot2 that are appropriate for common purposes
  + condition means (bar plot, line plot)
  + condition means with a grouping variable (barplot, line plot)
  + relation between 2 variables
  + relation between 2 variables with a grouping variable
  + *challenge section* - alternative methods to visualize variability
* Learn strategies for visualizing variability
* By going through examples, develop an beginner’s understanding of the “grammar of graphics” [(Wickham, 2007)](https://doi.org/10.1198/jcgs.2009.07098) used by ggplot2 (geom, coordinate system, aesthetic component mapping, statistical function, layers, …)

### Starting off notes

Today we are going to jump straight into plotting with the ggplot2 package. The ggplot2 package will most likely not seem intuitive at first, but after a few examples you might start to get the hang of it. Andy Field’s “discovr” tutorial #3 is highly recommended for getting more comfortable with ggplot2 capabilities.  
For learning more about ggplot2 you can go through the [official documentation (the cheat sheet is a good resource)](https://ggplot2.tidyverse.org/) and [examples in the R-Graph-Gallery site](https://www.r-graph-gallery.com/ggplot2-package.html).

This activity borrows the from the [PsyTeachR unit on data visualization](https://psyteachr.github.io/ug2-practical/visualisation-through-ggplot2.html) developed at the University of Glasgow [School of Psychology](https://www.gla.ac.uk/schools/psychology/) and [Institute of Neuroscience and Psychology](https://www.gla.ac.uk/researchinstitutes/neurosciencepsychology/).

## Step 1 - Get organized

* make a folder for today’s activity, with a new Rproj file
* make a “data” folder
  + Download the data file [mentalrotation.csv](../data/MentalRotationBehavioralData.csv) and place it in the new “data” folder
* make a “r\_docs” folder
  + start a new R Markdown doc and save it in your r\_docs folder

## Step 2 - Import data and check it out

* This data is from [Ganis and Kievit (2016)](https://doi.org/10.5334/jopd.ai), a replication of the [Shephard and Metzler (1971)](https://doi.org/10.1126/science.171.3972.701).
* In this study subjects had to mentally rotate a 3D shape and respond whether it was the same or different compared to a reference shape. The angle of rotation was manipulated (within subjects) at 0, 50, 100, and 150 degrees as well as the desired response (whether the shape was actually same or different).
* Each line in the MentalRotationBehavioralData.csv file represents 1 trial. The Time column is response time in milliseconds. Missed responses are coded as NA in the Time column and “[N/A]” in the ActualResponse column.
* **What to do first:** Make a new code chunk and use readr::read\_csv() to read in the data. Make sure that NA values are handled the way you want (click on the tibble in the Environment window pane to take a quick look).
* **What to do next:** make sure the columns that contain nominal vals are treated as nominal, using forcats::as\_factor() *take a look at the solution to see how*

Show/Hide Solution

#first import the data  
mrot\_tib <- readr::read\_csv("data/mentalrotation.csv", na = "NA")  
  
# now make sure the columns we want as factors are treated that way, using forcats::as\_factor() - we could let "Angle" be a Ratio variable   
mrot\_tib <- mrot\_tib %>% dplyr::mutate(  
 DesiredResponse = forcats::as\_factor(DesiredResponse),  
 ActualResponse = forcats::as\_factor(ActualResponse),  
 CorrectResponse = forcats::as\_factor(CorrectResponse),  
 Sex = forcats::as\_factor(Sex),  
)

* **Now skip some things you would normally do:** If you want to take a look at descriptives and distribution of response time, click the button:

Show/Hide Extra Stuff

#first import the data  
mrot\_tib <- readr::read\_csv("data/mentalrotation.csv", na = "NA")  
  
# now make sure the columns we want as nominal vals are treated as nominal, using forcats::as\_factor()  
mrot\_tib <- mrot\_tib %>% dplyr::mutate(  
 DesiredResponse = forcats::as\_factor(DesiredResponse),  
 ActualResponse = forcats::as\_factor(ActualResponse),  
 CorrectResponse = forcats::as\_factor(CorrectResponse),  
 Sex = forcats::as\_factor(Sex),  
)  
  
# first group by Participant and store in a new tibble  
mrot\_bysub <- mrot\_tib %>% drop\_na(Time) %>%   
 group\_by(Participant) %>%   
 dplyr::summarise(  
 sub\_meanRT = mean(Time)  
 )  
# then average across Participants  
mrot\_bysub %>% dplyr::summarise(  
 meanRT = mean(sub\_meanRT),  
 ci.low = ggplot2::mean\_cl\_normal(sub\_meanRT)$ymin,  
 ci.upp = ggplot2::mean\_cl\_normal(sub\_meanRT)$ymax,  
 median = median(sub\_meanRT),  
 sd = sd(sub\_meanRT),  
 cases = n() - sum(is.na(sub\_meanRT))  
) %>%   
 knitr::kable(caption = "Response Time Descriptives", digits = 3) %>%   
 kableExtra::kable\_styling(full\_width = FALSE)  
  
# now visualize the distribution  
p1 <- mrot\_bysub %>% drop\_na(sub\_meanRT) %>%  
 ggplot( aes(x=sub\_meanRT)) + geom\_histogram(binwidth=100) + theme\_classic() +  
 labs (title = "Response Time distribution")  
p2 <- mrot\_bysub %>% drop\_na(sub\_meanRT) %>%  
 ggplot( aes(y=sub\_meanRT)) + geom\_boxplot() + theme\_classic() +   
 labs (title = "Response Time box plot")  
p3 <- mrot\_bysub %>% drop\_na(sub\_meanRT) %>%  
 ggplot( aes(sample=sub\_meanRT)) + geom\_qq() + geom\_qq\_line() + theme\_classic() +  
 labs (title = "Response Time Q-Q")  
p1  
p2  
p3  
mrot\_bysub %>% {shapiro.test(.$sub\_meanRT)}

Response Time Descriptives

meanRT

ci.low

ci.upp

median

sd

cases

2734.102

2544.947

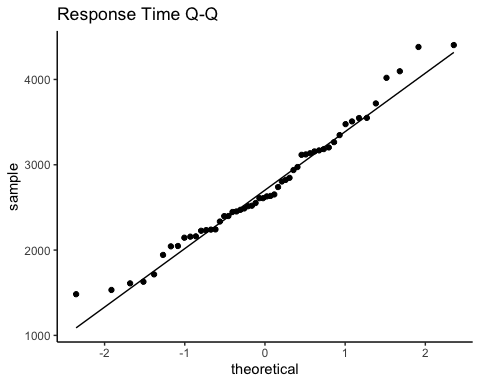
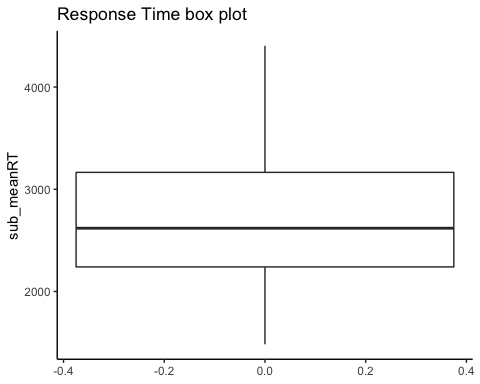
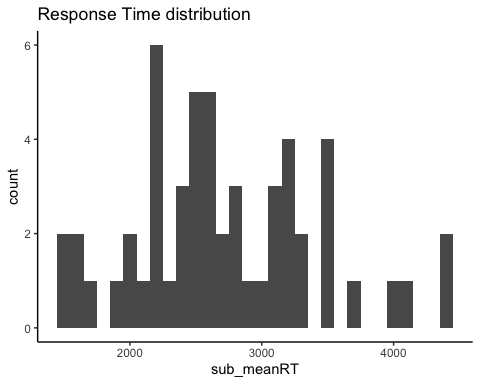
2923.256

2618.95

693.007

54

##   
## Shapiro-Wilk normality test  
##   
## data: .$sub\_meanRT  
## W = 0.97293, p-value = 0.2588

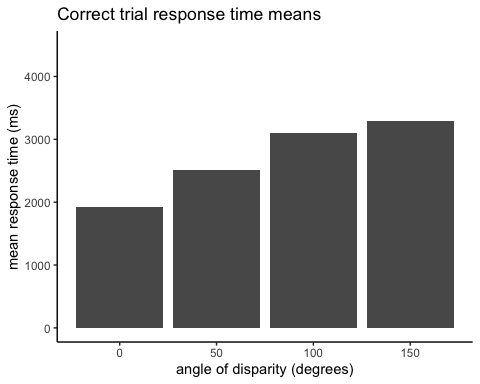


## Step 3 - make a bar plot of response time means by Angle condition

* The x-axis is the “Angle” condition (1,50,100,150), and the y-axis is the response time mean. For this exercise let’s first calculate means across trials for each subject, then calculate a sample mean across subjects, with sd based on the subject (by condition) means.
* We will only use response times from correct response trials (CorrectResponse==“Correct”)
* For most of these plots we recommend you reveal the solution immediately, then paste it into a code chunk in your own Rmd and check that it works (e.g., make sure the variable name matches for where you imported the data). Then you can look through the code and think about what each piece is for.

Show/Hide Solution

# A. first group data by Participant and Angle and store in a new tibble  
mrot\_bysub <- mrot\_tib %>% filter(CorrectResponse=="Correct") %>%   
 group\_by(Participant, Angle) %>%   
 dplyr::summarise(  
 sub\_meanRT = mean(Time)  
 )  
# B. then group data by Angle only to average across Participants  
mrot\_summary <- mrot\_bysub %>%   
 group\_by(Angle) %>%   
 dplyr::summarise(  
 meanRT = mean(sub\_meanRT),  
 ci.low = ggplot2::mean\_cl\_normal(sub\_meanRT)$ymin,  
 ci.upp = ggplot2::mean\_cl\_normal(sub\_meanRT)$ymax,  
 )  
# C. now we can make the plot  
p1 <- mrot\_summary %>%   
 ggplot(aes(x = Angle, y = meanRT)) +   
 geom\_bar(stat = "identity") +  
 coord\_cartesian(ylim = c(0, 4500)) +   
 theme\_classic() + labs(title="Correct trial response time means", y = "mean response time (ms)", x = "angle of disparity (degrees)")  
p1



##### A few key elements in the code:

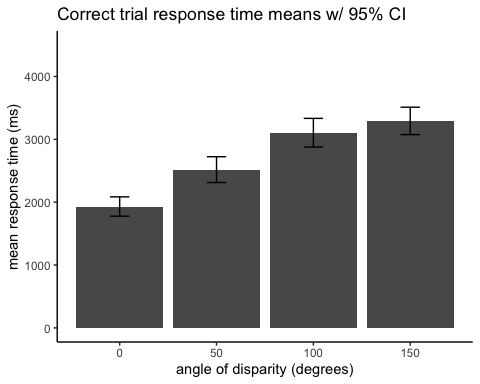
1. first we compute participant means for each Angle condition. the filter(CorrectResponse=="Correct") makes sure we only use trials where the response was correct.
2. the plotting command is all in the block of code containing the ggplot() function - mrot\_summary %>% sends the summary stats to the ggplot() functions.
3. ggplot(aes(x = Angle, y = meanRT)) sets the “Angle” column of the data to the x-axis and the “meanRT” column to the y-axis. These are called “aesthetics” (hence the aes() function) in ggplot terminology.
4. After the main ggplot() function we add “layers”. The main layer is the bars that are created by geom\_bar(). We specify stat = "identity" because the statistic we want to plot is already calculated in the mrot\_summary tibble the we sent as input to ggplot(). If instead we sent a tibble of participant-wise means, we could specify stat = "mean" to have ggplot compute the group means.
5. coord\_cartesian(ylim = c(0, 4500)) controls the range of the y-axis (we could add xlim = if we wanted to). labs() controls the labels. theme\_classic is one of many options for themes that specify color, font, positioning, etc.

#### Error bars

* Okay, so we can see the means, but there’s no information at all about variability, so let’s at least put error bars on the bar plot.
* Let’s set them to equal to the 95% confidence interval around the mean. The solution above already calculated ci.low and ci.upp, so just add a layer to the ggplot() code including this geom\_errorbar(aes(ymin=ci.low, ymax=ci.upp), width=10) - the ymin/ymax are values for the bars, and the width is in units of the x-axis (which is 50 units between levels because “Angle” is treated as an interval measure rather than as a factor).

Show/Hide Solution

# add ymin and ymax to the aesthetics list, which are needed for geom\_errorbar  
p1 <- mrot\_summary %>%   
 ggplot(aes(x = Angle, y = meanRT, ymin=ci.low, ymax=ci.upp)) +   
 geom\_bar(stat = "identity") +  
 geom\_errorbar(width=10) +  
 coord\_cartesian(ylim = c(0, 4500)) +   
 theme\_classic() + labs(title="Correct trial response time means w/ 95% CI", y = "mean response time (ms)", x = "angle of disparity (degrees)")  
p1

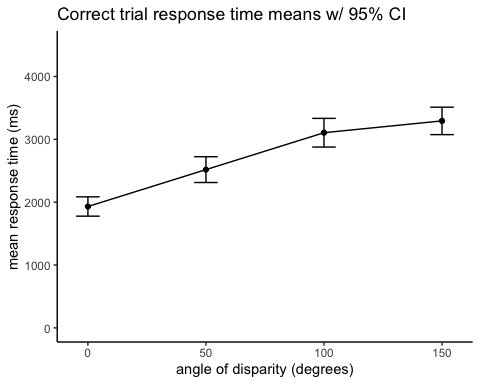


#### Line Plot

* In some cases you might prefer a point and line plot instead of bars, though it provides the same amount of information here. Let’s see how it looks as a line plot instead.
* Instead of geom\_bar use geom\_point to get a point at each mean.
* Now add another layer with geom\_line(aes(group=1),stat = "identity") to draw lines connecting the points (the group=1 part specifies there is only one group of lines).

Show/Hide Solution

p1 <- mrot\_summary %>%   
 ggplot(aes(x = Angle, y = meanRT, ymin=ci.low, ymax=ci.upp)) +   
 geom\_point(stat = "identity") +  
 geom\_line(aes(group=1),stat = "identity") +  
 geom\_errorbar(width=10) +  
 coord\_cartesian(ylim = c(0, 4500)) +   
 theme\_classic() + labs(title="Correct trial response time means w/ 95% CI", y = "mean response time (ms)", x = "angle of disparity (degrees)")  
p1

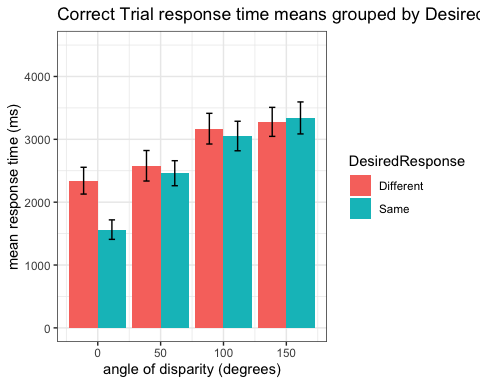


## Step 4 - group the bar plot by “DesiredResponse”

* The column “DesiredResponse” indicates whether the target shape was the same or different than the reference shape. Let’s treat the data like a 2x4 design and split each bar into two (one bar for “same” and one for “different”). We’ll color code the “DesiredResponse” values.
* to accomplish this, first we have to re-do the grouping to also group by “DesiredResponse” values
* then we add a new aesthetic called “fill” (referring to the bar fill color) to designate the colors by DesiredResponse values
* then we need to adjust the position of the bars so that they are not on top of each other - use position=position\_dodge() in the geoms
* Copy the Solution code below and then look at it piece by piece to understand it

Show/Hide Solution

# regroup data to include additional factor  
mrot\_bysub <- mrot\_tib %>% filter(CorrectResponse=="Correct") %>%  
 group\_by(Participant, Angle, DesiredResponse) %>%   
 dplyr::summarise(  
 sub\_meanRT = mean(Time)  
 )  
mrot\_summary <- mrot\_bysub %>%   
 group\_by(Angle,DesiredResponse) %>%   
 dplyr::summarise(  
 meanRT = mean(sub\_meanRT),  
 ci.low = ggplot2::mean\_cl\_normal(sub\_meanRT)$ymin,  
 ci.upp = ggplot2::mean\_cl\_normal(sub\_meanRT)$ymax,  
 )  
# add fill aesthetic as DesiredResponse  
# use position\_dodge to move the bars and errorbars horizontally so   
# they aren't on top of each other. A dodge value of .9\*(distance between levels) is generally a good dodge value.  
p1 <- mrot\_summary %>%   
 ggplot(aes(x = Angle, y = meanRT, ymin=ci.low, ymax=ci.upp, fill=DesiredResponse)) +   
 geom\_bar(stat = "identity", position=position\_dodge(45)) +  
 geom\_errorbar(width=10, position=position\_dodge(45)) +  
 coord\_cartesian(ylim = c(0, 4500)) +   
 theme\_bw() + labs(title="Correct Trial response time means grouped by DesiredResponse", y = "mean response time (ms)", x = "angle of disparity (degrees)")  
p1

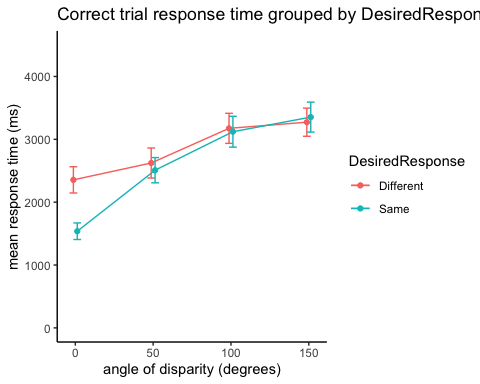


#### Line Plot, grouped by “DesiredResponse”

* Cool. Now that you’ve figured that out, try the same thing but as a line plot.
* Instead of fill, use the color aesthetic.
* Use a smaller dodge value (or no dodge value at all) now that you don’t have thick bars. If you specify a dodge value, you’ll need to do so for each of the three geom elements (geom\_point, geom\_line, and geom\_errorbar)

Show/Hide Solution

mrot\_bysub <- mrot\_tib %>% drop\_na(Time) %>%  
 group\_by(Participant, Angle, DesiredResponse) %>%   
 dplyr::summarise(  
 sub\_meanRT = mean(Time)  
 )  
mrot\_summary <- mrot\_bysub %>%   
 group\_by(Angle,DesiredResponse) %>%   
 dplyr::summarise(  
 meanRT = mean(sub\_meanRT),  
 ci.low = ggplot2::mean\_cl\_normal(sub\_meanRT)$ymin,  
 ci.upp = ggplot2::mean\_cl\_normal(sub\_meanRT)$ymax,  
 )  
p1 <- mrot\_summary %>%   
 ggplot(aes(x = Angle, y = meanRT, ymin=ci.low, ymax=ci.upp, color=DesiredResponse)) +   
 geom\_point(stat = "identity", position=position\_dodge(5)) +  
 geom\_line(aes(group=DesiredResponse), position=position\_dodge(5)) +  
 geom\_errorbar(width=10, position=position\_dodge(5)) +  
 coord\_cartesian(ylim = c(0, 4500)) +   
 theme\_classic() + labs(title="Correct trial response time grouped by DesiredResponse", y = "mean response time (ms)", x = "angle of disparity (degrees)")  
p1



## Step 5 - Visualize a relation between two variables

* Is reaction time related to accuracy?
* Let’s compute %correct for each subject, by counting the proportion of “Correct” values in the “CorrectResponse” column. We exclude trials with no response.

Show/Hide Solution

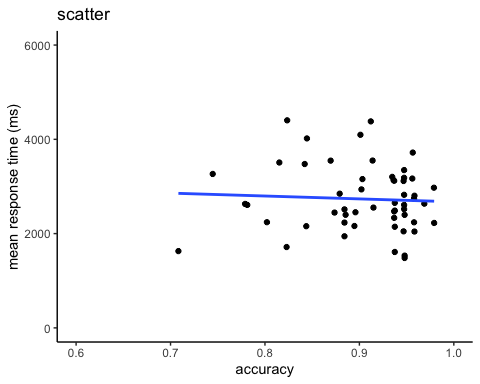
mrot\_bysub <- mrot\_tib %>% drop\_na() %>%   
 group\_by(Participant) %>%   
 dplyr::summarise(  
 sub\_meanRT = mean(Time),  
 sub\_accuracy = sum(CorrectResponse=="Correct")/n()  
 )

#### Scatter plot of response time by accuracy

* now we can specify the x and y axes, and use geom\_point() to plot a point for each subject with accuracy on the x-axis and response time on the y-axis
* we can add a regression line by adding a layer with stat\_smooth(method = "lm", formula = "y ~ x")

Show/Hide Solution

p1 <- mrot\_bysub %>%   
 ggplot(aes(x = sub\_accuracy, y = sub\_meanRT)) +   
 geom\_point(stat="identity") +  
 stat\_smooth(method = "lm", formula = "y ~ x", se = FALSE) + #intercept automatically included  
 coord\_cartesian(ylim = c(0, 6000), xlim = c(.6,1)) +   
 theme\_classic() + labs(title="scatter", y = "mean response time (ms)", x = "accuracy")  
p1



#### Scatter plot of response time by accuracy, grouped by DesiredResponse

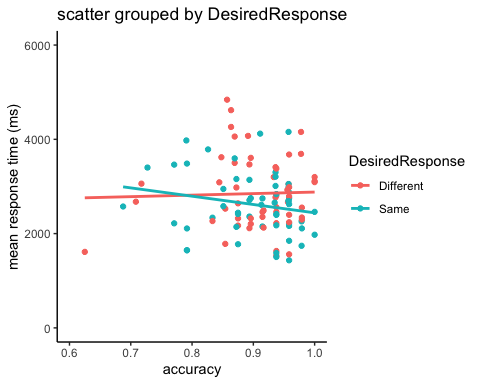
* What if we wanted to split up subject RT and accuracy by DesiredResponse (whether the shapes were same or different)?
* just (a) recompute the subject-level means, grouping by DesiredResponse, and (b) add a color mapping aesthetic color=DesiredResponse in the aes() part of the ggplot code

Show/Hide Solution

mrot\_bysub <- mrot\_tib %>% drop\_na() %>%   
 group\_by(Participant, DesiredResponse) %>%   
 dplyr::summarise(  
 sub\_meanRT = mean(Time),  
 sub\_accuracy = sum(CorrectResponse=="Correct")/n()  
 )

## `summarise()` has grouped output by 'Participant'. You can override using the `.groups` argument.

p1 <- mrot\_bysub %>%   
 ggplot(aes(x = sub\_accuracy, y = sub\_meanRT, color = DesiredResponse)) +   
 geom\_point(stat="identity") +  
 stat\_smooth(method = "lm", formula = "y ~ x", se = FALSE) + #intercept automatically included  
 coord\_cartesian(ylim = c(0, 6000), xlim = c(.6,1)) +   
 theme\_classic() + labs(title="scatter grouped by DesiredResponse", y = "mean response time (ms)", x = "accuracy")  
p1



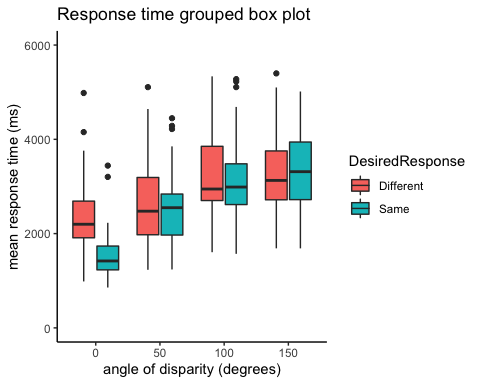
## Challenge: For your reference (or if you have extra time in lab) - here are some alternative approaches to visualizing variability:

#### 1. Use a box plot instead of plotting means:

* notice that we have to re-specify the x-axis (Angle) variable as a factor to get geom\_boxplot to work correctly mutate(Angle=forcats::as\_factor(Angle))
* notice that we pass in subject level means, instead of group means
* also notice that we have to expand the ylim range to see all outliers

Show/Hide Code and Plot

# regroup data   
mrot\_bysub <- mrot\_tib %>% mutate(Angle=forcats::as\_factor(Angle)) %>% drop\_na(Time) %>%   
 group\_by(Participant, Angle, DesiredResponse) %>%   
 dplyr::summarise(  
 sub\_meanRT = mean(Time)  
 )  
  
p1 <- mrot\_bysub %>%   
 ggplot(aes(x = Angle, y = sub\_meanRT, fill=DesiredResponse)) +   
 geom\_boxplot() +  
 coord\_cartesian(ylim = c(0, 6000)) +   
 theme\_classic() + labs(title="Response time grouped box plot", y = "mean response time (ms)", x = "angle of disparity (degrees)")  
p1

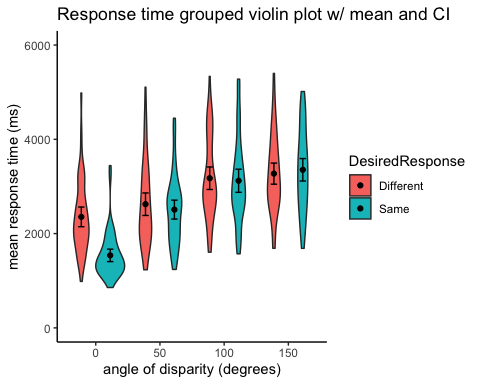


#### 2. Use a violin plot layer under plotted means

* again, we pass in subject level means
* again, we use an expanded ylim range
* for geom\_point() and geom\_errorbar this time we specify the group summarized data

Show/Hide Code and Plot

# group data   
mrot\_bysub <- mrot\_tib %>% mutate(Angle=forcats::as\_factor(Angle)) %>% drop\_na(Time) %>%   
 group\_by(Participant, Angle, DesiredResponse) %>%   
 dplyr::summarise(  
 sub\_meanRT = mean(Time)  
 )  
mrot\_summary <- mrot\_bysub %>%   
 group\_by(Angle,DesiredResponse) %>%   
 dplyr::summarise(  
 meanRT = mean(sub\_meanRT),  
 ci.low = ggplot2::mean\_cl\_normal(sub\_meanRT)$ymin,  
 ci.upp = ggplot2::mean\_cl\_normal(sub\_meanRT)$ymax,  
 )  
p1 <- mrot\_bysub %>%   
 ggplot(aes(x = Angle, y = sub\_meanRT, fill=DesiredResponse)) +   
 geom\_violin(position = position\_dodge(.9)) +  
 geom\_point(data=mrot\_summary, aes(x = Angle, y = meanRT), stat="identity", position = position\_dodge(.9)) +  
 geom\_errorbar(data=mrot\_summary, aes(x = Angle, y = meanRT, ymin=ci.low, ymax=ci.upp), stat="identity", width=.2, position\_dodge(.9)) +  
 coord\_cartesian(ylim = c(0, 6000)) +   
 theme\_classic() + labs(title="Response time grouped violin plot w/ mean and CI", y = "mean response time (ms)", x = "angle of disparity (degrees)")  
p1



#### 3. Show all the points: Use a 1D scatter with mean and error bars on top

* again, we pass in subject level means
* again, we use an expanded ylim range
* we do the mean and error bars just like in the previous example using group-summarized data

Show/Hide Code and Plot

# regroup data   
mrot\_bysub <- mrot\_tib %>% mutate(Angle=forcats::as\_factor(Angle)) %>% drop\_na(Time) %>%   
 group\_by(Participant, Angle, DesiredResponse) %>%   
 dplyr::summarise(  
 sub\_meanRT = mean(Time)  
 )  
p1 <- mrot\_bysub %>%   
 ggplot(aes(x = Angle, y = sub\_meanRT, color=DesiredResponse, shape=DesiredResponse)) +   
 geom\_jitter(position=position\_jitterdodge(.6)) +  
 geom\_point(data=mrot\_summary, aes(x = Angle, y = meanRT),color="black", stat="identity", position = position\_dodge(.75), show.legend = FALSE) +  
 geom\_errorbar(data=mrot\_summary, aes(x = Angle, y = meanRT, ymin=ci.low, ymax=ci.upp), color="black", stat="identity", width=.2, position\_dodge(.75), show.legend = FALSE) +  
 coord\_cartesian(ylim = c(0, 6000)) +   
 theme\_classic() + labs(title="Response time grouped 1D scatter plot", y = "mean response time (ms)", x = "angle of disparity (degrees)")  
p1

