Raincloud tutorial for repeated measures in R

Jordy van Langen

March 01, 2020

Background

This tutorial is inspired by work from Allen et al. (2019) and was initially created to contribute to a GitHub repository called 'open-visualizations' created by Jordy van Langen. The idea behind the 'open-visualizations' repository stems from the fact that (open) science - in general - lacks 'fully' transparent and robust visualizations, i.e., figures have always some form of 'hidden-data'. In line with the **open-science** naming, I came up with: **open-visualiziations**. To overcome the issue of figures that include 'hidden-data', I initially created a Jupyter Notebook written in the Python language. After posting a Tweet on Twitter, in which I stated that I was working on 'open-visualizations', Neuroscientist Micah Allen replied and advised me to check out his work on Raincloud. After performing two of their tutorials, in R and Python respectively, I thought of a way to combine the raincloud approach with my own work performed in Python. Shortly after posting another Tweet which included some figures that I produced in R, I received a huge amount of encouraging feedback which led me to writing this markdown document.

If you have any questions, suggestions for improvement or identify bugs, please open an issue in the GitHub repository open-visualizations.

If you use my repository for your research, please reference it.

R-version check

```
R.version$version.string
```

```
## [1] "R version 3.6.1 (2019-07-05)"
```

Package dependencies

Make sure you have the packages that are needed for this tutorial.

- Install **plyr** before **dplyr** - which is included in **tidyverse**. If you need functions from both **plyr** and **dplyr**, please load **plyr** first, then **dplyr**, otherwise error messages might occur (source: R console). **Rmisc** also depends on this package.
- Install **lattice** since **Rmisc** dependends on this package.
- Install **tidyverse** since it includes **ggplot2**, **dplyr**, and **readr**.
- Install **rmarkdown** to convert this .Rmd template into a variety of formats including HTML, MS Word, PDF, and Beamer (Only required if you do not work in Rstudio).
- Install **Rmisc** to perform some basic statistical computations (e.g., calculate mean, median, sd, se, ci).

- Install **cowplot** since it includes features that help with creating publication-quality figures, such as a set of themes, functions to align plots and arrange them into complex compound figures, and functions that make it easy to annotate plots and or mix plots with images. This package enables other themes next to the standard **ggplot2** *theme_()* functionality.

- Install **gghalves** from their GitHub repository since it features the newest options as opposed to the version on CRAN. To install from GitHub, the **devtools** package needs to be installed first. **gghalves** is a **ggplot2** extension for easy plotting of half-half geom combinations. Think half boxplot and half jitterplot, or half violinplot and half dotplot.

Package references

- **plyr** - Wickham, 2019
- **lattice** - Sarkar, 2019
- **tidyverse** - Wickham, Rstudio, 2019
- **rmarkdown** - Allaire & Xie, 2020
- **Rmisc** - Hope, 2013
- **cowplot** - Wilke, 2019
- **devtools** - Wickham & Hester, 2020
- **gghalves** - Tiedemann, 2020

Install packages

```
packages <- c("plyr", "lattice", "tidyverse", "rmarkdown","Rmisc", "cowplot")

if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(packages, rownames(installed.packages())))
}

if (!require(devtools)) {
    install.packages("devtools")
}
devtools::install_github('erocoar/gghalves')
```

Load packages

```
library("plyr")
library("lattice")
library("tidyverse")
library("rmarkdown")
library("Rmisc")
library("cowplot")
```

```r
library("devtools")
library("gghalves")


# width and height variables for saved plots
w = 6
h = 4

# Define limits of y-axis
y_lim_min = 4
y_lim_max = 7.5

# Make the figure folder if it doesn't exist yet
dir.create('../tutorial_R/figs_repmes/', showWarnings = FALSE)
```

For this tutorial, we make use of the package `gghalves`.

The main idea behind `gghalves` is that standard `geom's` aggregate data e.g.,`geom_boxplot`, `geom_violin` and `geom_dotplot` who all tend to be an approximation of symmetry. Given that the space to display information is limited, we can make better use of it by cutting the `geoms` in half and displaying additional `geoms` that e.g. give information about the sample size Tiedemann, 2020.

Figure 1

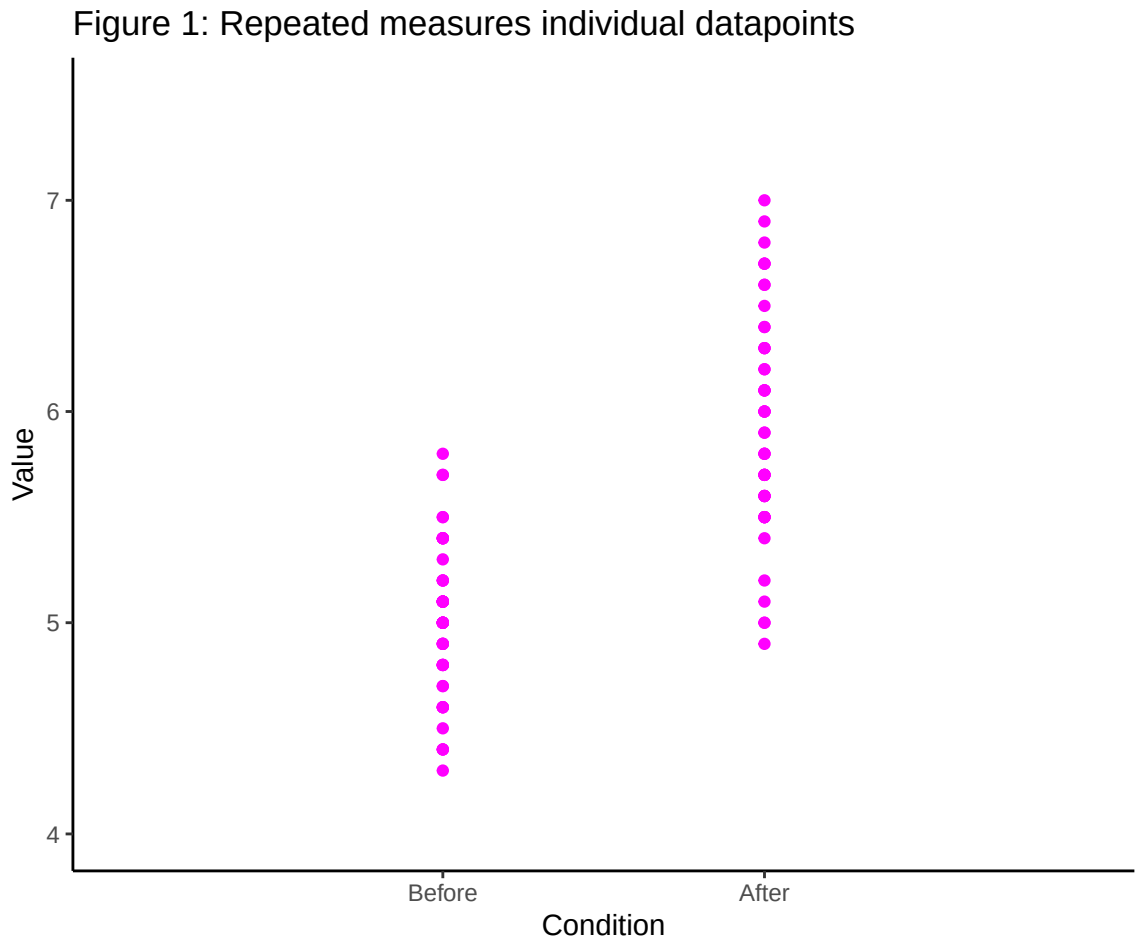
For this example, we make use of the *iris* dataset, which is freely available in R.

We manipulate the dataset by creating two variables (1) `before` and (2) `after`. Specifiy the variable `n` and create a dataframe `d` which includes the variables `y`, `x` and `id`.

```r
before = iris$Sepal.Length[1:50]
after = iris$Sepal.Length[51:100]
n <- length(before)
d <- data.frame(y = c(before, after),
                x = rep(c(1,2), each=n),
                id = factor(rep(1:n,2)))
```

Let's create a first very basic figure only showing the individual datapoints.

```r
f1 <- ggplot(data=d, aes(y=y)) +

   #Add geom_() objects
   geom_point(aes(x=x), color = "magenta", size = 1.5) +

   #Define additional settings
   scale_x_continuous(breaks=c(1,2), labels=c("Before", "After"), limits=c(0, 3)) +
   xlab("Condition") + ylab("Value") +
   ggtitle('Figure 1: Repeated measures individual datapoints') +
   theme_classic() +
   coord_cartesian(ylim=c(y_lim_min, y_lim_max))

f1
```

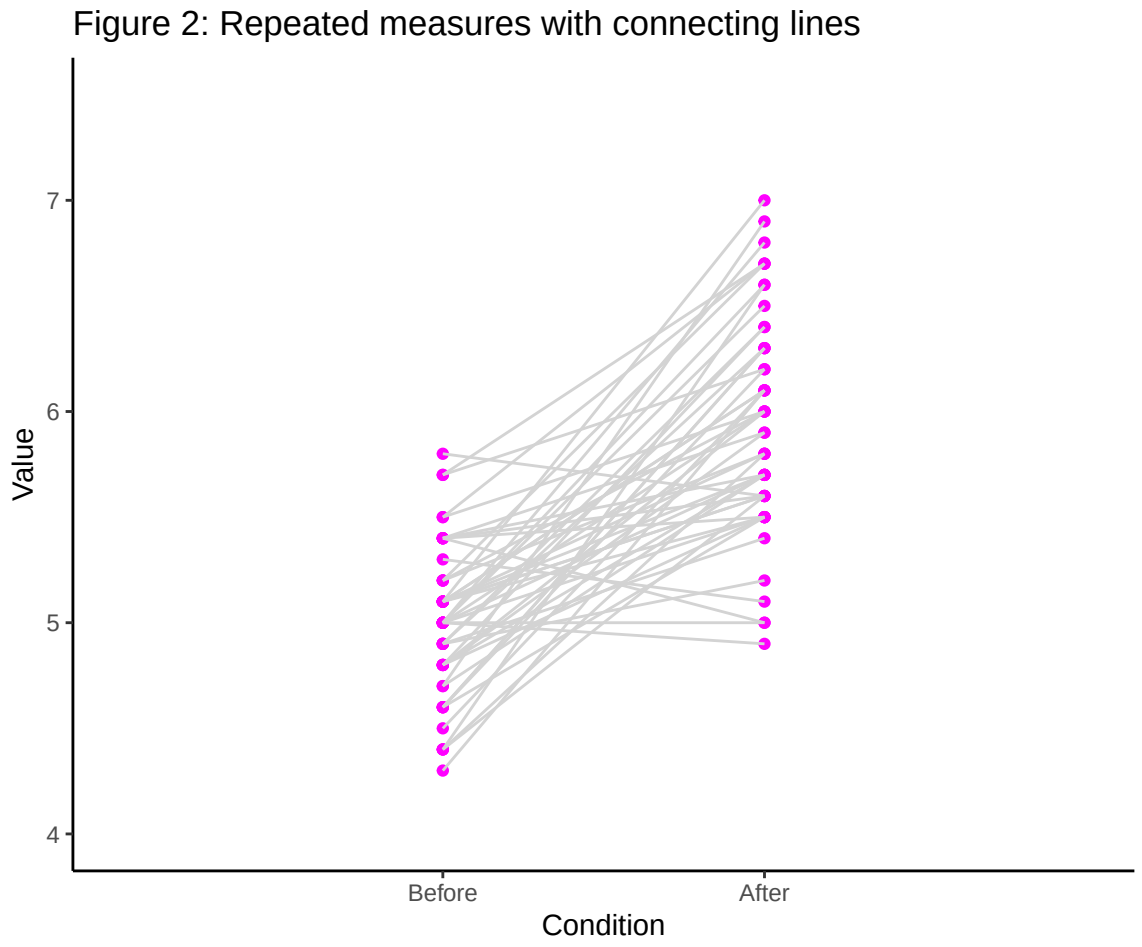



```
ggsave('../tutorial_R/figs_repmes/figure1.png', width = w, height = h)
```

Figure 2

Let's again create a simple figure, but now with the datapoints connected (i.e., intra-individual trends).

```
f2 <- ggplot(data=d, aes(y=y)) +

   #Add geom_() objects
   geom_point(aes(x=x), color = "magenta", size = 1.5) +
   geom_line(aes(x=x, group=id), color = 'lightgray') +

   #Define additional settings
   scale_x_continuous(breaks=c(1,2), labels=c("Before", "After"), limits=c(0, 3)) +
   xlab("Condition") + ylab("Value") +
   ggtitle('Figure 2: Repeated measures with connecting lines') +
   theme_classic()+
   coord_cartesian(ylim=c(y_lim_min, y_lim_max))

f2
```




```
ggsave('../tutorial_R/figs_repmes/figure2.png', width = w, height = h)
```

Figure 3

Let's add some jitter to avoid that datapoints overlap.

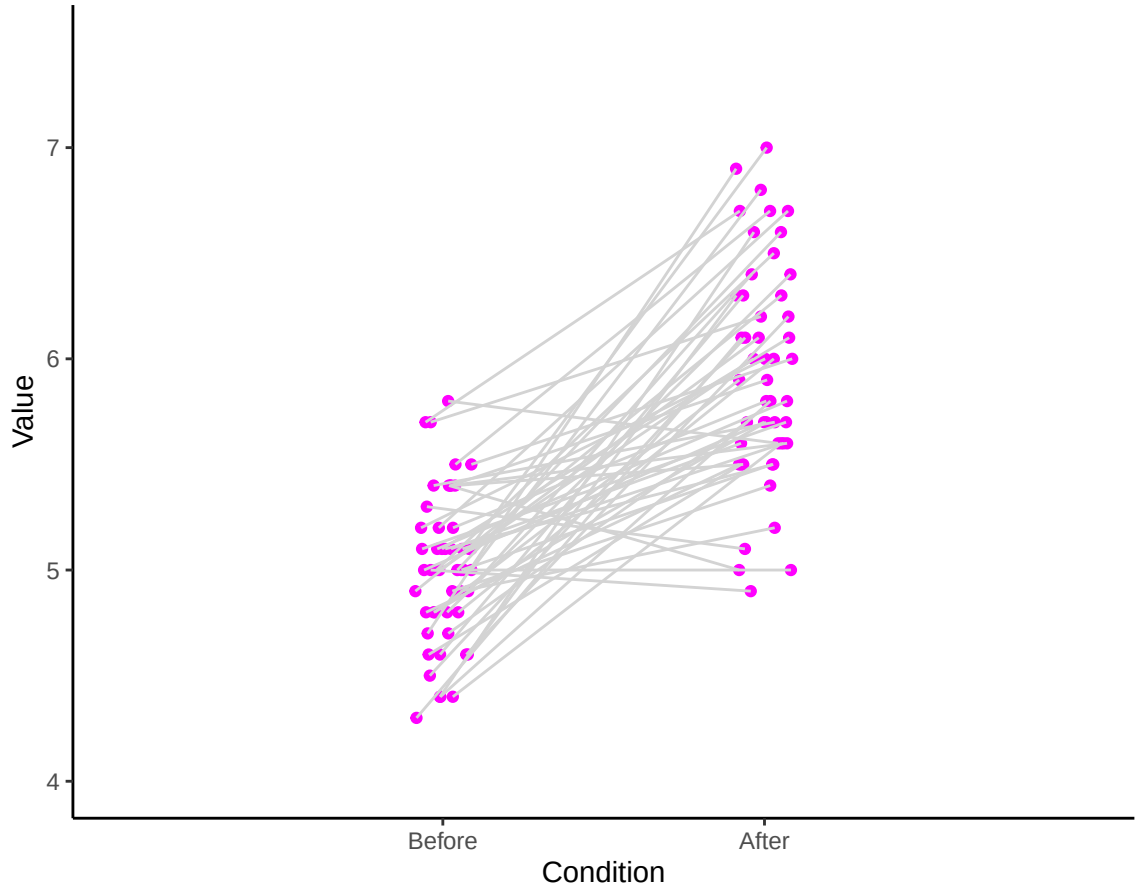
```
set.seed(321)
d$xj <- jitter(d$x, amount=.09)
```

Now we create the the figure again, including jitter.

```
f3 <- ggplot(data=d, aes(y=y)) +

   #Add geom_() objects
   geom_point(aes(x=xj), color = "magenta", size = 1.5) +
   geom_line(aes(x=xj, group=id), color = 'lightgray') +

   #Define additional settings
   scale_x_continuous(breaks=c(1,2), labels=c("Before", "After"), limits=c(0, 3)) +
   xlab("Condition") + ylab("Value") +
   ggtitle('Figure 3: Repeated measures with jitter and connections') +
```

```
   theme_classic()+
   coord_cartesian(ylim=c(y_lim_min, y_lim_max))

f3
```

Figure 3: Repeated measures with jitter and connections




```
ggsave('../tutorial_R/figs_repmes/figure3.png', width = w, height = h)
```

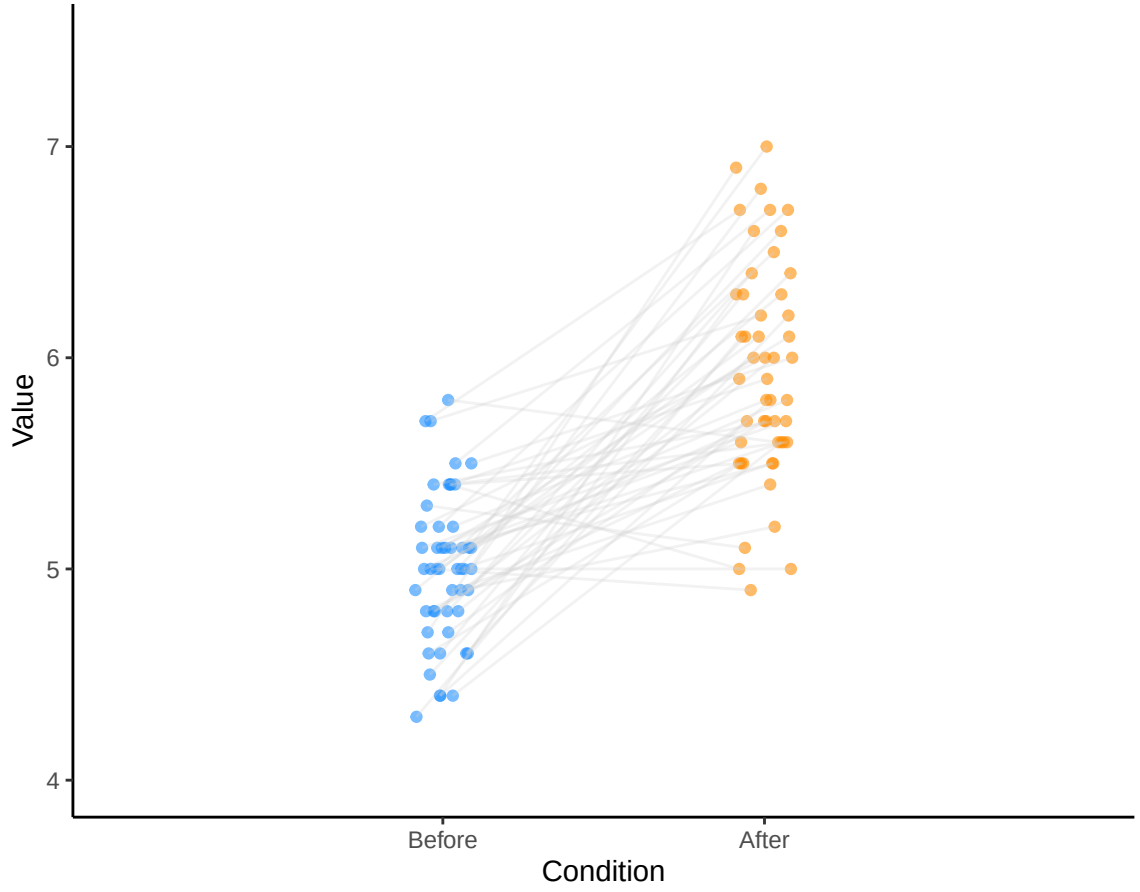
Figure 4

Produce another figure, but now defined by different colors for `Before` and `After`. Just for illustrative purposes, `alpha` was used in `geom_line`.

```
f4 <- ggplot(data=d, aes(y=y)) +

   #Add geom_() objects
   geom_point(data = d %>% filter(x=="1"), aes(x=xj), color = 'dodgerblue', size = 1.5,
              alpha = .6) +
   geom_point(data = d %>% filter(x=="2"), aes(x=xj), color = 'darkorange', size = 1.5,
              alpha = .6) +
   geom_line(aes(x=xj, group=id), color = 'lightgray', alpha = .3) +
```

```
   #Define additional settings
   scale_x_continuous(breaks=c(1,2), labels=c("Before", "After"), limits=c(0, 3)) +
   xlab("Condition") + ylab("Value") +
   ggtitle('Figure 4: Repeated measures with jittered datapoints and connections') +
   theme_classic()+
   coord_cartesian(ylim=c(y_lim_min, y_lim_max))

f4
```

Figure 4: Repeated measures with jittered datapoints and connections




```
ggsave('../tutorial_R/figs_repmes/figure4.png', width = w, height = h)
```

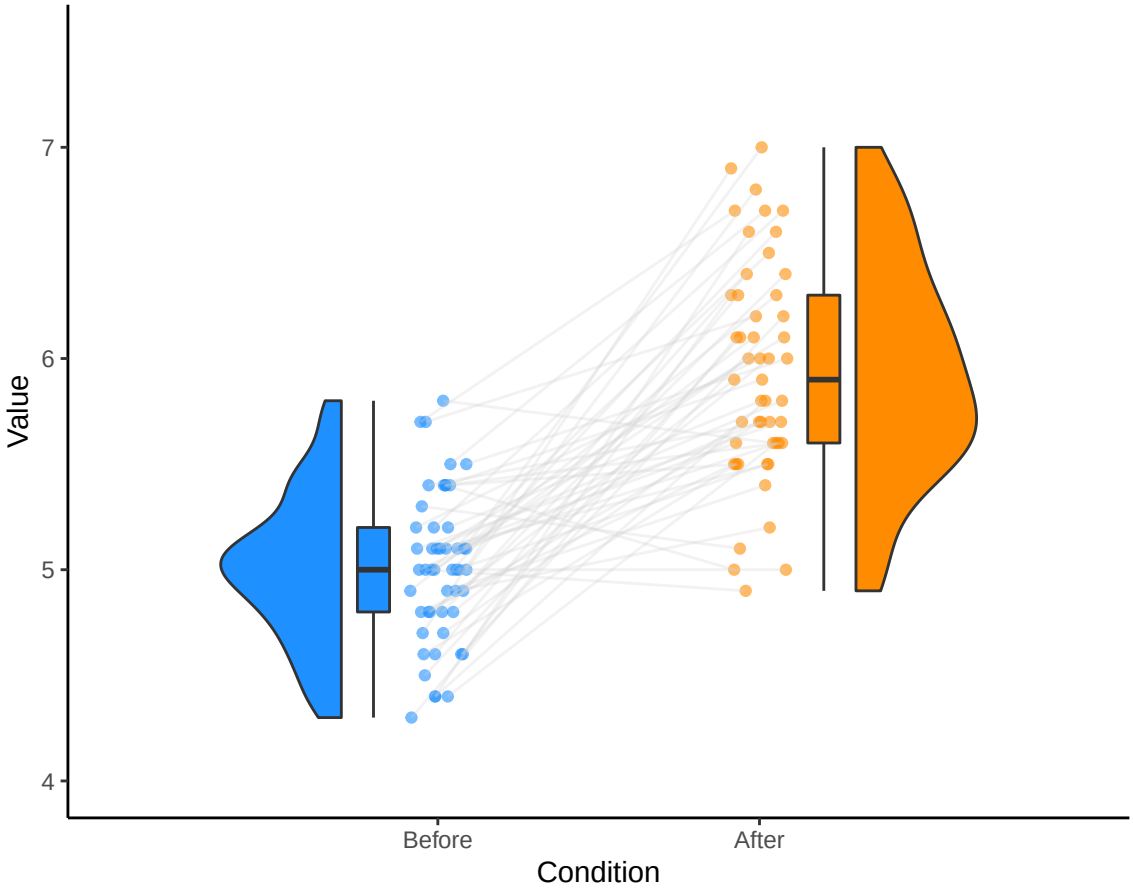
Figure 5

Let's add box- and violinplots to create `raincloud` like plots.

```
f5 <- ggplot(data = d, aes(y = y)) +

   #Add geom_() objects
   geom_point(data = d %>% filter(x =="1"), aes(x = xj), color = 'dodgerblue', size = 1.5,
              alpha = .6) +
   geom_point(data = d %>% filter(x =="2"), aes(x = xj), color = 'darkorange', size = 1.5,
```

```
              alpha = .6) +
  geom_line(aes(x = xj, group = id), color = 'lightgray', alpha = .3) +

  geom_half_boxplot(
    data = d %>% filter(x=="1"), aes(x=x, y = y), position = position_nudge(x = -.25),
    side = "r",outlier.shape = NA, center = TRUE, errorbar.draw = FALSE, width = .2,
    fill = 'dodgerblue') +

  geom_half_boxplot(
    data = d %>% filter(x=="2"), aes(x=x, y = y), position = position_nudge(x = .15),
    side = "r",outlier.shape = NA, center = TRUE, errorbar.draw = FALSE, width = .2,
    fill = 'darkorange') +

  geom_half_violin(
    data = d %>% filter(x=="1"),aes(x = x, y = y), position = position_nudge(x = -.3),
    side = "l", fill = 'dodgerblue') +

  geom_half_violin(
    data = d %>% filter(x=="2"),aes(x = x, y = y), position = position_nudge(x = .3),
    side = "r", fill = "darkorange") +


  #Define additional settings
  scale_x_continuous(breaks=c(1,2), labels=c("Before", "After"), limits=c(0, 3)) +
  xlab("Condition") + ylab("Value") +
  ggtitle('Figure 5: Repeated measures with box- and violin plots') +
  theme_classic()+
  coord_cartesian(ylim=c(y_lim_min, y_lim_max))

f5
```

Figure 5: Repeated measures with box− and violin plots




```
ggsave('../tutorial_R/figs_repmes/figure5.png', width = w, height = h)
```

Note

A potential downside of the current approach is that it uses a lot of filtering and a lot of `geoms`.

Add descriptive statistics

- Create a dataframe including:
  - Mean
  - Median
  - Standard deviation
  - Standard error
  - Confidence interval (95 %)

```
score_mean_1 <- mean(d$y[1:50])
score_mean_2 <- mean(d$y[51:100])
score_median1 <- median(d$y[1:50])
score_median2 <- median(d$y[51:100])
score_sd_1 <- sd(d$y[1:50])
```

```
score_sd_2 <- sd(d$y[51:100])
score_se_1 <- score_sd_1/sqrt(n) #-> adjust your n
score_se_2 <- score_sd_2/sqrt(n) #-> adjust your n
score_ci_1 <- CI(d$y[1:50], ci = 0.95)
score_ci_2 <- CI(d$y[51:100], ci = 0.95)

#Create data frame with 2 rows and 7 columns containing the descriptives
group <- c("x", "z")
N <- c(50, 50)
score_mean <- c(score_mean_1, score_mean_2)
score_median <- c(score_median1, score_median2)
sd <- c(score_sd_1, score_sd_2)
se <- c(score_se_1, score_se_2)
ci <- c((score_ci_1[1] - score_ci_1[3]), (score_ci_2[1] - score_ci_2[3]))

#Create the dataframe
summary_df <- data.frame(group, N, score_mean, score_median, sd, se, ci)
```

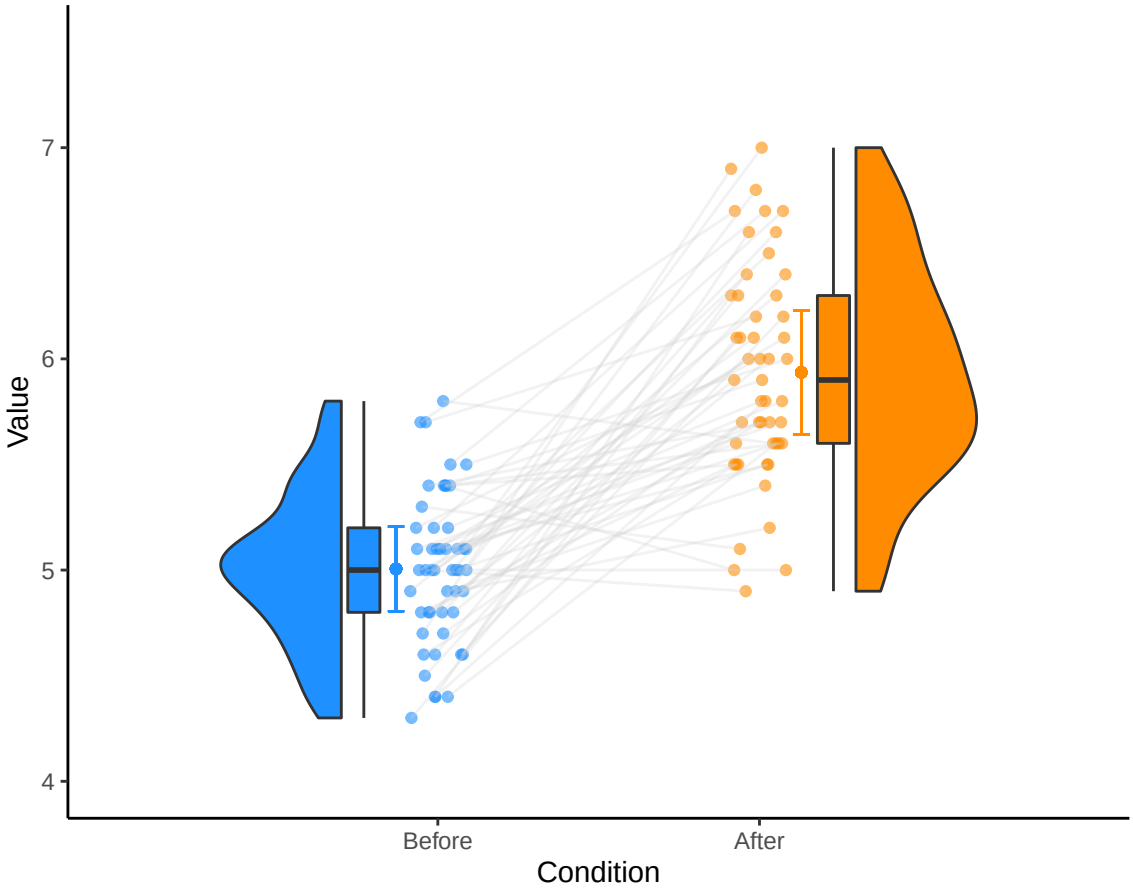
Figure 6

Let's add the items calculated in `summary_df` to the figure

```
f6 <- ggplot(data = d, aes(y = y)) +

   #Add geom_() objects
   geom_point(data = d %>% filter(x =="1"), aes(x = xj), color = 'dodgerblue', size = 1.5,
              alpha = .6) +
   geom_point(data = d %>% filter(x =="2"), aes(x = xj), color = 'darkorange', size = 1.5,
              alpha = .6) +
   geom_line(aes(x = xj, group = id), color = 'lightgray', alpha = .3) +

   geom_half_boxplot(
     data = d %>% filter(x=="1"), aes(x=x, y = y), position = position_nudge(x = -.28),
     side = "r",outlier.shape = NA, center = TRUE, errorbar.draw = FALSE, width = .2,
     fill = 'dodgerblue') +

   geom_half_boxplot(
     data = d %>% filter(x=="2"), aes(x=x, y = y), position = position_nudge(x = .18),
     side = "r",outlier.shape = NA, center = TRUE, errorbar.draw = FALSE, width = .2,
     fill = 'darkorange') +

   geom_half_violin(
     data = d %>% filter(x=="1"),aes(x = x, y = y), position = position_nudge(x = -.3),
     side = "l", fill = 'dodgerblue') +

   geom_half_violin(
     data = d %>% filter(x=="2"),aes(x = x, y = y), position = position_nudge(x = .3),
     side = "r", fill = "darkorange") +

   geom_point(data = d %>% filter(x=="1"), aes(x = x, y = score_mean[1]),
     position = position_nudge(x = -.13), color = "dodgerblue", alpha = .6, size = 1.5) +
```

```
  geom_errorbar(data = d %>% filter(x=="1"), aes(x = x, y = score_mean[1],
    ymin = score_mean[1]-ci[1], ymax = score_mean[1]+ci[1]),
    position = position_nudge(-.13),
    color = "dodgerblue", width = 0.05, size = 0.4, alpha = .5) +

  geom_point(data = d %>% filter(x=="2"), aes(x = x, y = score_mean[2]),
   position = position_nudge(x = .13), color = "darkorange", alpha = .6, size = 1.5)+

  geom_errorbar(data = d %>% filter(x=="2"), aes(x = x, y = score_mean[2],
   ymin = score_mean[2]-ci[2],
   ymax = score_mean[2]+ci[2]), position = position_nudge(.13), color = "darkorange",
   width = 0.05, size = 0.4, alpha = .5) +


  #Define additional settings
  scale_x_continuous(breaks=c(1,2), labels=c("Before", "After"), limits=c(0, 3)) +
  xlab("Condition") + ylab("Value") +
  ggtitle('Figure 6: Repeated measures with box- and violin plots') +
  theme_classic()+
  coord_cartesian(ylim=c(y_lim_min, y_lim_max))

f6
```

Figure 6: Repeated measures with box− and violin plots



```
ggsave('../tutorial_R/figs_repmes/figure6.png', width = w, height = h)
```

Figure 7

Optionally you could add a line between the two means. Define the x-coordinates where the line needs to be drawn. Note these coordinates can be calculated as 1 - position_nudge() for the first variable and 2 + position_nudge() for the second variable, which in our case is 1 - .13 = .87 and 2 + .13 = 2.13.

```
x_tick_means <- c(.87, 2.13)

f7 <- ggplot(data = d, aes(y = y)) +

   #Add geom_() objects
   geom_point(data = d %>% filter(x =="1"), aes(x = xj), color = 'dodgerblue', size = 1.5,
              alpha = .6) +
   geom_point(data = d %>% filter(x =="2"), aes(x = xj), color = 'darkorange', size = 1.5,
              alpha = .6) +
   geom_line(aes(x = xj, group = id), color = 'lightgray', alpha = .3) +

   geom_half_boxplot(
     data = d %>% filter(x=="1"), aes(x=x, y = y), position = position_nudge(x = -.28),
     side = "r",outlier.shape = NA, center = TRUE, errorbar.draw = FALSE, width = .2,
     fill = 'dodgerblue') +

   geom_half_boxplot(
     data = d %>% filter(x=="2"), aes(x=x, y = y), position = position_nudge(x = .18),
     side = "r",outlier.shape = NA, center = TRUE, errorbar.draw = FALSE, width = .2,
     fill = 'darkorange') +

   geom_half_violin(
     data = d %>% filter(x=="1"),aes(x = x, y = y), position = position_nudge(x = -.3),
     side = "l", fill = 'dodgerblue') +

   geom_half_violin(
     data = d %>% filter(x=="2"),aes(x = x, y = y), position = position_nudge(x = .3),
     side = "r", fill = "darkorange") +

   geom_point(data = d %>% filter(x=="1"), aes(x = x, y = score_mean[1]),
     position = position_nudge(x = -.13), color = "dodgerblue", alpha = .6, size = 1.5) +

   geom_errorbar(data = d %>% filter(x=="1"), aes(x = x, y = score_mean[1],
     ymin = score_mean[1]-ci[1], ymax = score_mean[1]+ci[1]),
     position = position_nudge(-.13),
     color = "dodgerblue", width = 0.05, size = 0.4, alpha = .6) +

   geom_point(data = d %>% filter(x=="2"), aes(x = x, y = score_mean[2]),
    position = position_nudge(x = .13), color = "darkorange", alpha = .6, size = 1.5)+

   geom_errorbar(data = d %>% filter(x=="2"), aes(x = x, y = score_mean[2],
    ymin = score_mean[2]-ci[2],
    ymax = score_mean[2]+ci[2]), position = position_nudge(.13), color = "darkorange",
    width = 0.05, size = 0.4, alpha = .6) +
```

```
  #Add a line connecting the two means
   geom_line(data = summary_df, aes(x = x_tick_means, y = score_mean), color = 'gray',
    size = 1) +

   #Define additional settings
   scale_x_continuous(breaks=c(1,2), labels=c("Before", "After"), limits=c(0, 3)) +
   xlab("Condition") + ylab("Value") +
   ggtitle('Figure 7: Repeated measures with box- and violin plots and means + ci ') +
   theme_classic()+
   coord_cartesian(ylim=c(y_lim_min, y_lim_max))

f7
```


```
ggsave('../tutorial_R/figs_repmes/figure7.png', width = w, height = h)
```

2 x 2 repeated measures rainclouds in a #butterfly fashion

As a last step, let's create these figures for a 2 x 2 repeated measures study.

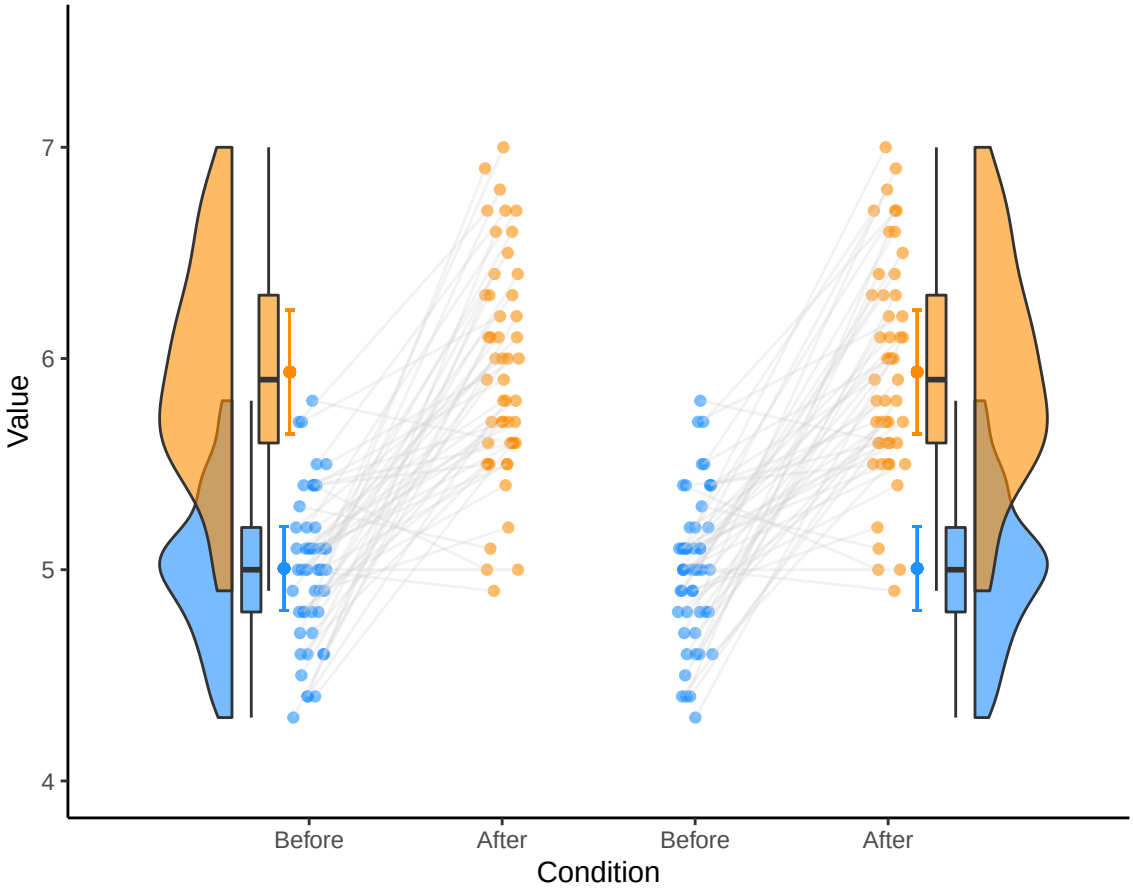
Define an additional variable `z` which has two categories `3` and `4` and create a second jittered variable.

```r
before = iris$Sepal.Length[1:50]
after = iris$Sepal.Length[51:100]
n <- length(before)
d <- data.frame(y = c(before, after),
                x = rep(c(1,2), each=n),
                z = rep(c(3,4), each=n),
                id = factor(rep(1:n,2)))

set.seed(321)
d$xj <- jitter(d$x, amount = .09)
d$xj_2 <- jitter(d$z, amount = .09)
```

```r
f8 <- ggplot(data = d, aes(y = y)) +

   #Add geom_() objects
   geom_point(data = d %>% filter(x =="1"), aes(x = xj), color = 'dodgerblue', size = 1.5,
              alpha = .6) +
   geom_point(data = d %>% filter(x =="2"), aes(x = xj), color = 'darkorange', size = 1.5,
              alpha = .6) +
   geom_point(data = d %>% filter(z =="3"), aes(x = xj_2), color = 'dodgerblue', size = 1.5,
              alpha = .6) +
   geom_point(data = d %>% filter(z =="4"), aes(x = xj_2), color = 'darkorange', size = 1.5,
              alpha = .6) +
   geom_line(aes(x = xj, group = id), color = 'lightgray', alpha = .3) +
   geom_line(aes(x = xj_2, group = id), color = 'lightgray', alpha = .3) +

   geom_half_boxplot(
     data = d %>% filter(x=="1"), aes(x=x, y = y), position = position_nudge(x = -.35),
     side = "r",outlier.shape = NA, center = TRUE, errorbar.draw = FALSE, width = .2,
     fill = 'dodgerblue', alpha = .6) +

   geom_half_boxplot(
     data = d %>% filter(x=="2"), aes(x=x, y = y), position = position_nudge(x = -1.16),
     side = "l",outlier.shape = NA, center = TRUE, errorbar.draw = FALSE, width = .2,
     fill = 'darkorange', alpha = .6) +

   geom_half_boxplot(
     data = d %>% filter(z=="3"), aes(x=z, y = y), position = position_nudge(x = 1.3),
     side = "r",outlier.shape = NA, center = TRUE, errorbar.draw = FALSE, width = .2,
     fill = 'dodgerblue', alpha = .6) +

   geom_half_boxplot(
     data = d %>% filter(z=="4"), aes(x=z, y = y), position = position_nudge(x = .2),
     side = "r",outlier.shape = NA, center = TRUE, errorbar.draw = FALSE, width = .2,
     fill = 'darkorange', alpha = .6) +

   geom_half_violin(
     data = d %>% filter(x=="1"),aes(x = x, y = y), position = position_nudge(x = -.40),
     side = "l", fill = 'dodgerblue', alpha = .6) +

   geom_half_violin(
     data = d %>% filter(x=="2"),aes(x = x, y = y), position = position_nudge(x = -1.40),
     side = "l", fill = "darkorange", alpha = .6) +
```

```
  geom_half_violin(
    data = d %>% filter(z=="3"),aes(x = z, y = y), position = position_nudge(x = 1.45),
    side = "r", fill = 'dodgerblue', alpha = .6) +

  geom_half_violin(
    data = d %>% filter(z=="4"),aes(x = z, y = y), position = position_nudge(x = .45),
    side = "r", fill = "darkorange", alpha = .6) +

  geom_point(data = d %>% filter(x=="1"), aes(x = x, y = score_mean[1]),
    position = position_nudge(x = -.13), color = "dodgerblue", alpha = .6, size = 1.5) +

  geom_errorbar(data = d %>% filter(x=="1"), aes(x = x, y = score_mean[1],
    ymin = score_mean[1]-ci[1], ymax = score_mean[1]+ci[1]),
    position = position_nudge(-.13),
    color = "dodgerblue", width = 0.05, size = 0.4, alpha = .6) +

  geom_point(data = d %>% filter(x=="2"), aes(x = x, y = score_mean[2]),
   position = position_nudge(x = -1.1), color = "darkorange", alpha = .6, size = 1.5)+

  geom_errorbar(data = d %>% filter(x=="2"), aes(x = x, y = score_mean[2],
   ymin = score_mean[2]-ci[2],
   ymax = score_mean[2]+ci[2]), position = position_nudge(x = -1.1), color = "darkorange",
   width = 0.05, size = 0.4, alpha = .6) +

  geom_point(data = d %>% filter(z=="3"), aes(x = z, y = score_mean[1]),
   position = position_nudge(x = 1.15), color = "dodgerblue", alpha = .5) +

  geom_errorbar(data = d %>% filter(z=="3"), aes(x = z, y = score_mean[1],
   ymin = score_mean[1]-ci[1],
   ymax = score_mean[1]+ci[1]), position = position_nudge(1.15),
   color = "dodgerblue", width = 0.05, size = 0.4, alpha = .5)+

  geom_point(data = d %>% filter(z=="4"), aes(x = z, y = score_mean[2]),
   position = position_nudge(x = .15), color = "darkorange", alpha = .5)+

  geom_errorbar(data = d %>% filter(z=="4"), aes(x = z, y = score_mean[2],
   ymin = score_mean[2]-ci[2], ymax = score_mean[2]+ci[2]), position = position_nudge(.15),
   color = "darkorange", width = 0.05, size = 0.4, alpha = .5)+


  #Define additional settings
  scale_x_continuous(breaks=c(1,2,3,4), labels=c("Before", "After","Before", "After"),
                     limits=c(0, 5))+
  xlab("Condition") + ylab("Value") +
  ggtitle('Figure 8: 2 x 2 Repeated measures with box- and violin plots') +
  theme_classic()+
  coord_cartesian(ylim=c(y_lim_min, y_lim_max))

f8
```

Figure 8: 2 x 2 Repeated measures with box− and violin plots




```
ggsave('../tutorial_R/figs_repmes/figure8.png', width = w, height = h)
```

Figure 9

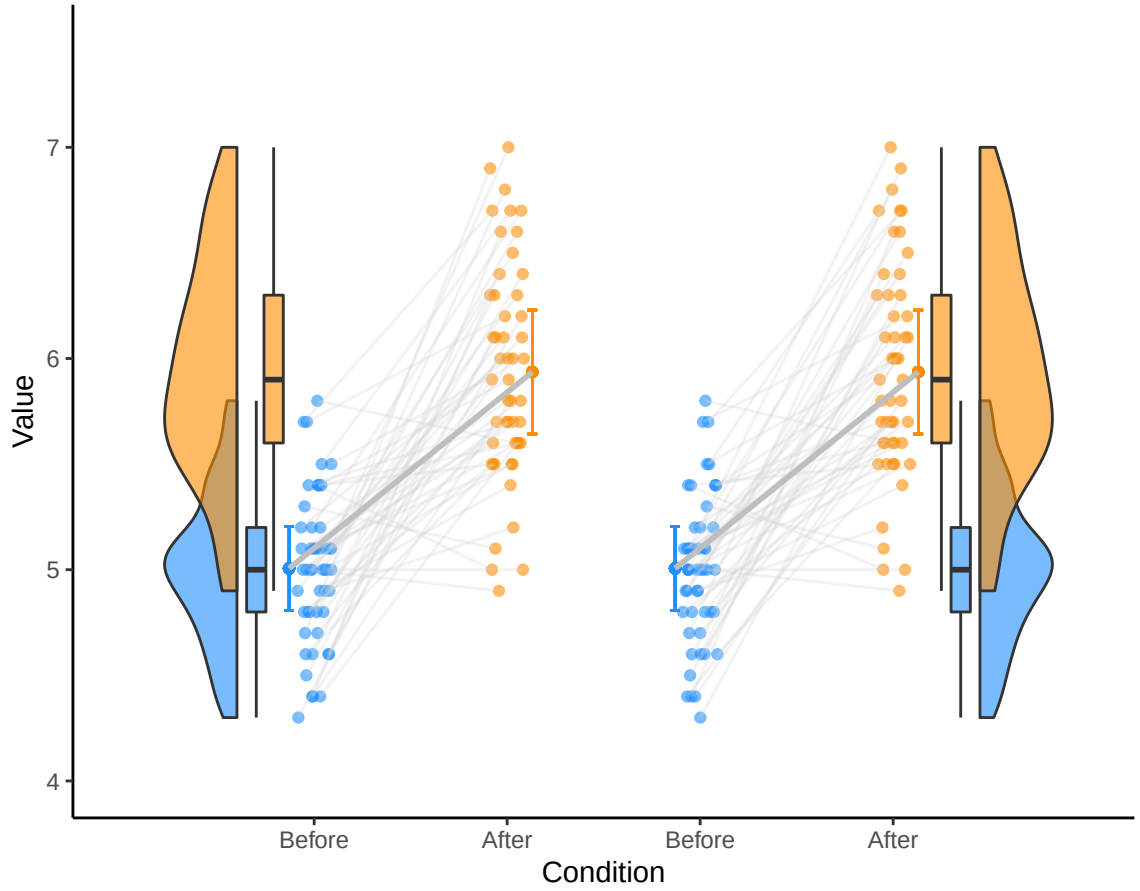
Finally, let's add lines that connect the means of each group.

```
#First we must again define the x-coordinates of the means.
x_tick_means_x <- c(.87, 2.13) #same as above
x_tick_means_z <- c(2.87, 4.13) #just add 2 for each tick

f9 <- ggplot(data = d, aes(y = y)) +

   #Add geom_() objects
   geom_point(data = d %>% filter(x =="1"), aes(x = xj), color = 'dodgerblue', size = 1.5,
              alpha = .6) +
   geom_point(data = d %>% filter(x =="2"), aes(x = xj), color = 'darkorange', size = 1.5,
              alpha = .6) +
   geom_point(data = d %>% filter(z =="3"), aes(x = xj_2), color = 'dodgerblue', size = 1.5,
              alpha = .6) +
   geom_point(data = d %>% filter(z =="4"), aes(x = xj_2), color = 'darkorange', size = 1.5,
              alpha = .6) +
   geom_line(aes(x = xj, group = id), color = 'lightgray', alpha = .3) +
```

```
  geom_line(aes(x = xj_2, group = id), color = 'lightgray', alpha = .3) +

  geom_half_boxplot(
    data = d %>% filter(x=="1"), aes(x=x, y = y), position = position_nudge(x = -.35),
    side = "r",outlier.shape = NA, center = TRUE, errorbar.draw = FALSE, width = .2,
    fill = 'dodgerblue', alpha = .6) +

  geom_half_boxplot(
    data = d %>% filter(x=="2"), aes(x=x, y = y), position = position_nudge(x = -1.16),
    side = "l",outlier.shape = NA, center = TRUE, errorbar.draw = FALSE, width = .2,
    fill = 'darkorange', alpha = .6) +

  geom_half_boxplot(
    data = d %>% filter(z=="3"), aes(x=z, y = y), position = position_nudge(x = 1.3),
    side = "r",outlier.shape = NA, center = TRUE, errorbar.draw = FALSE, width = .2,
    fill = 'dodgerblue', alpha = .6) +

  geom_half_boxplot(
    data = d %>% filter(z=="4"), aes(x=z, y = y), position = position_nudge(x = .2),
    side = "r",outlier.shape = NA, center = TRUE, errorbar.draw = FALSE, width = .2,
    fill = 'darkorange', alpha = .6) +

  geom_half_violin(
    data = d %>% filter(x=="1"),aes(x = x, y = y), position = position_nudge(x = -.40),
    side = "l", fill = 'dodgerblue', alpha = .6) +

  geom_half_violin(
    data = d %>% filter(x=="2"),aes(x = x, y = y), position = position_nudge(x = -1.40),
    side = "l", fill = "darkorange", alpha = .6) +

  geom_half_violin(
    data = d %>% filter(z=="3"),aes(x = z, y = y), position = position_nudge(x = 1.45),
    side = "r", fill = 'dodgerblue', alpha = .6) +

  geom_half_violin(
    data = d %>% filter(z=="4"),aes(x = z, y = y), position = position_nudge(x = .45),
    side = "r", fill = "darkorange", alpha = .6) +

  geom_point(data = d %>% filter(x=="1"), aes(x = x, y = score_mean[1]),
    position = position_nudge(x = -.13), color = "dodgerblue", alpha = .6, size = 1.5) +

  geom_errorbar(data = d %>% filter(x=="1"), aes(x = x, y = score_mean[1],
    ymin = score_mean[1]-ci[1], ymax = score_mean[1]+ci[1]),
    position = position_nudge(-.13),
    color = "dodgerblue", width = 0.05, size = 0.4, alpha = .6) +

  geom_point(data = d %>% filter(x=="2"), aes(x = x, y = score_mean[2]),
   position = position_nudge(x = .13), color = "darkorange", alpha = .6, size = 1.5)+

  geom_errorbar(data = d %>% filter(x=="2"), aes(x = x, y = score_mean[2],
   ymin = score_mean[2]-ci[2],
   ymax = score_mean[2]+ci[2]), position = position_nudge(x = .13), color = "darkorange",
   width = 0.05, size = 0.4, alpha = .6) +
```

```
  geom_point(data = d %>% filter(z=="3"), aes(x = z, y = score_mean[1]),
   position = position_nudge(x = -.13), color = "dodgerblue", alpha = .5) +

  geom_errorbar(data = d %>% filter(z=="3"), aes(x = z, y = score_mean[1],
   ymin = score_mean[1]-ci[1],
   ymax = score_mean[1]+ci[1]), position = position_nudge(-.13),
   color = "dodgerblue", width = 0.05, size = 0.4, alpha = .5)+

  geom_point(data = d %>% filter(z=="4"), aes(x = z, y = score_mean[2]),
   position = position_nudge(x = .13), color = "darkorange", alpha = .5)+

  geom_errorbar(data = d %>% filter(z=="4"), aes(x = z, y = score_mean[2],
   ymin = score_mean[2]-ci[2], ymax = score_mean[2]+ci[2]),
   position = position_nudge(.13),
   color = "darkorange", width = 0.05, size = 0.4, alpha = .5)+

  #Add lines connecting the two means
  geom_line(data = summary_df, aes(x = x_tick_means_x, y = score_mean),
            color = 'gray', size = 1) +
  geom_line(data = summary_df, aes(x = x_tick_means_z, y = score_mean),
            color = 'gray', size = 1) +

  #Define additional settings
  scale_x_continuous(breaks=c(1,2,3,4), labels=c("Before", "After","Before", "After"),
                     limits=c(0, 5))+
  xlab("Condition") + ylab("Value") +
  ggtitle('Figure 9: 2 x 2 Repeated measures with box- and violin plots') +
  theme_classic()+
  coord_cartesian(ylim=c(y_lim_min, y_lim_max))

f9
```

Figure 9: 2 x 2 Repeated measures with box− and violin plots




```
ggsave('../tutorial_R/figs_repmes/figure9.png', width = w, height = h)
```

General remarks / tips

- To be more flexible in assigning labels to your figures, the **ggtext** package by Wilke, 2020 might be worthwile .
- If you would like to be flexible in plotting multiple figures next to each other, check-out the **patchwork** package by Pedersen, 2020.
- If you want to save your figures in a high-quality manner (> GB) for e.g., publications, you could save your figure with a `.tiff` extension and add `dpi=` as used in the following line of code:

```
ggsave("figure.tiff", height=h, width=w, units='in', dpi=600)
```

- If for some reason your code does not work due to e.g., an update in a package, you could use the following line of code to unload and load that package again.

```
if("package_name" %in% (.packages())){
  detach('package:package_name', unload=TRUE)}

library("package_name")
```

You have reached the end of this document.

I hope you'll be able to use this tutorial to create more `open-visualizations` for your research!

If you use this tutorial, please cite it in your work.

open-visualizations for repeated measures in R by **Jordy van Langen**