

Crossover Trial Further Simulation and Plotting

Eamonn O'Brien

24 July, 2019

Contents

Function to simulate crossover trial dataset	3
Generate data and analyse	4
Simulation function for crossover design, preparing to vary an input parameter	5
Simulate crossover power and vary between person SD	6
Simulate crossover power and vary within person SD	7
Simulate a ABBA data set with known effects that has good power	8
Summary stats	10
Use a package to plot AB sequence then BA sequence, should be mirror image ideally	11
My code to plot treatment comparisons	13
Plot AB sequence then BA sequence, should be mirror image ideally	15
My code to plot all data in a useful way	16
Computing Environment	18

Contents

List of Figures

List of Tables

Function to simulate crossover trial dataset

```

ABBA <- function(n = 10, sdW = 4, sdB = 1, beta = c(8, 1, 0,
0), alpha = 0.05) {

  # generate data
  Patient <- as.factor(rep(1:(2 * n), rep(2, 2 * n)))
  Treatment <- c(rep(c("Treatment1", "Treatment2"), n), rep(c("Treatment2",
    "Treatment1"), n))
  Order <- rep(c("First", "Second"), 2 * n)
  Data <- data.frame(Patient, Treatment, Order)
  FMat <- model.matrix(~Treatment * Order, data = Data)
  RMat <- model.matrix(~0 + Patient, data = Data)
  Response <- FMat %*% beta + RMat %*% rnorm(2 * n, 0, sdB) +
    rnorm(4 * n, 0, sdW) # beta here
  Data$Response <- Response

  return(data.frame(Data))
  # analyse Fit <- lme(Response~ Treatment * Order, random=~1 |
  # Patient, data=Data, na.action='na.omit' ) Est <-
  # fixed.effects(Fit)[2] Ste <- sqrt(vcov(Fit)[2,2]) prod(Est
  # + c(-1,1) * qnorm(1-alpha/2) * Ste) > 0
}

```

Generate data and analyse

```
mydata <- ABBA(n = 10, sdW = 2, sdB = 2, beta = c(8, 1, 0, 0),
  alpha = 0.05)

require(nlme)
f <- lme(Response ~ Treatment * Order, random = ~1 | Patient,
  data = mydata, na.action = "na.omit")
anova(f)
```

	numDF	denDF	F-value	p-value
(Intercept)	1	19	421.0340	<.0001
Treatment	1	17	0.5352	0.4744
Order	1	17	0.9612	0.3406
Treatment:Order	1	17	0.4587	0.5073

```
summary(f)$tTable
```

	Value	Std.Error	DF
(Intercept)	8.023633	0.7468591	19
TreatmentTreatment2	1.030554	1.0562182	17
OrderSecond	1.182608	1.0562182	17
TreatmentTreatment2:OrderSecond	-1.166955	1.7229741	17

	t-value
(Intercept)	10.7431685
TreatmentTreatment2	0.9757021
OrderSecond	1.1196628
TreatmentTreatment2:OrderSecond	-0.6772910

	p-value
(Intercept)	0.000000001638637
TreatmentTreatment2	0.342892860343766
OrderSecond	0.278427969007660
TreatmentTreatment2:OrderSecond	0.507334457267561

Simulation function for crossover design, preparing to vary an input parameter

```

C.power <- function(a, b, c, d, e, f, g, n.sims = sims) {

  Treatment <- rep(NA, n.sims)
  Order <- rep(NA, n.sims)
  Interaction <- rep(NA, n.sims)

  for (s in 1:n.sims) {

    fake <- ABBA(n = a, sdW = b, sdB = c, beta = c(d, e,
      f, g), alpha = 0.05)

    possibleError <- tryCatch(f1 <- lme(Response ~ Treatment *
      Order, random = ~1 | Patient, data = fake, na.action = "na.omit"),
      error = function(e) e)

    ### http://stackoverflow.com/questions/8093914
    ### /skip-to-next-value-of-loop-upon-error-in-r-trycatch

    if (!inherits(possibleError, "error")) {

      modelint <- possibleError

      z <- as.matrix(summary(modelint)$tTable)
      Treatment[s] <- z[2, 5][[1]]
      Order[s] <- z[3, 5][[1]]
      Interaction[s] <- z[4, 5][[1]]
    }
  }

  trt <- mean(Treatment < 0.05)
  ord <- mean(Order < 0.05)
  int <- mean(Interaction < 0.05)

  c(trt, ord, int)
}

# stand alone, single example that shows power for main
# effect, order effect and interaction effect
C.power(a = 20, b = 1, c = 1, d = 8, e = 1, f = 0, g = 0, n.sims = 100)

```

```
[1] 0.60 0.08 0.10
```

Simulate crossover power and vary between person SD

```
## execute the functions numerous times assess changing
## between person SD

sims <- 200

J <- c(2, 2.5, 3, 4) # here we vary

k <- length(J)

dnam = list(betweenSD = J, Estimate = c("trt effect power", "order power",
    "interaction power"))

pwpr <- array(NA, dim = sapply(dnam, length), dimnames = dnam)

system.time(for (i in 1:k) {
  pwpr[i, ] <- C.power(a = 20, b = 2, c = J[i], d = 8, e = 4,
    f = 0, g = 0, n.sims = sims)
})

      user  system elapsed
13.90    0.00   13.91

print(pwpr, digits = 4)
```

	Estimate			
betweenSD	trt effect power	order power	interaction power	
2	0.990	0.050	0.055	
2.5	0.975	0.035	0.030	
3	0.920	0.035	0.030	
4	0.785	0.030	0.040	

Simulate crossover power and vary within person SD

```

sims <- 200

J <- c(2, 2.5, 3, 4) # here we vary

k <- length(J)

dnam = list(betweenSD = J, Estimate = c("trt effect power", "order power",
    "interaction power"))

pwpr <- array(NA, dim = sapply(dnam, length), dimnames = dnam)

system.time(for (i in 1:k) {
  pwpr[i, ] <- C.power(a = 20, b = J[i], c = 4, d = 8, e = 4,
    f = 1, g = 0, n.sims = sims)
})

```

```

      user  system elapsed
13.72    0.00    13.72

```

```
print(pwpr, digits = 4)
```

	Estimate			
betweenSD	trt effect power	order power	interaction power	
2		0.820	0.110	0.035
2.5		0.795	0.070	0.030
3		0.615	0.065	0.055
4		0.515	0.075	0.045

Simulate a ABBA data set with known effects that has good power

```

# require(ggpubr)
require(ggplot2)

# state some parameters so we can pick them up later on in
# plot titles
sb <- 2
sw <- 2
beta = c(8, 2, 1, 0)

# check the power, it is not likely the main effect will be
# missed, we even have good power for the order effect.
C.power(a = 100, b = sb, c = sw, d = beta[1], e = beta[2], f = beta[3],
        g = beta[4], n.sims = 1000)

[1] 1.000 0.704 0.044

# so lets create the data
mydata <- ABBA(n = 100, sdW = 2, sdB = 2, beta = c(8, 2, 1, 0),
              alpha = 0.05)

# analysis
tryCatch(f1 <- lme(Response ~ Treatment * Order, random = ~1 |
                  Patient, data = mydata, na.action = "na.omit"), error = function(e) e)

summary(f1)

```

Linear mixed-effects model fit by REML

Data: mydata

	AIC	BIC	logLik
	1908.478	1932.367	-948.2392

Random effects:

Formula: ~1 | Patient

(Intercept) Residual

StdDev: 2.283018 1.813877

Fixed effects: Response ~ Treatment * Order

	Value	Std.Error	DF
(Intercept)	7.945616	0.2915874	199
TreatmentTreatment2	1.987840	0.4123668	197
OrderSecond	1.088590	0.4123668	197
TreatmentTreatment2:OrderSecond	0.123714	0.7406616	197

	t-value	p-value
(Intercept)	27.249520	0.0000
TreatmentTreatment2	4.820562	0.0000
OrderSecond	2.639859	0.0090
TreatmentTreatment2:OrderSecond	0.167031	0.8675

Correlation:

	(Intr)	TrtmT2	OrdrSc
TreatmentTreatment2	-0.707		
OrderSecond	-0.707	0.807	
TreatmentTreatment2:OrderSecond	0.635	-0.898	-0.898

Standardized Within-Group Residuals:

Min	Q1	Med	Q3	Max
-2.43567647	-0.53233676	0.03385682	0.54524738	1.89156778

Number of Observations: 400

Number of Groups: 200

`intervals(f1)`

Approximate 95% confidence intervals

Fixed effects:

	lower	est.
(Intercept)	7.3706182	7.9456158
TreatmentTreatment2	1.1746198	1.9878398
OrderSecond	0.2753701	1.0885900
TreatmentTreatment2:OrderSecond	-1.3369295	0.1237137

	upper
(Intercept)	8.520613
TreatmentTreatment2	2.801060
OrderSecond	1.901810
TreatmentTreatment2:OrderSecond	1.584357

`attr("label")`

`[1] "Fixed effects:"`

Random Effects:

Level: Patient

	lower	est.	upper
<code>sd((Intercept))</code>	1.998192	2.283018	2.608443

Within-group standard error:

	lower	est.	upper
	1.643741	1.813877	2.001623

useful concatenation for plotting

`mydata$temp <- paste(mydata$Treatment, mydata$Order, sep = ".")`

Summary stats

```
options(digits = 3)
library("tidyverse")
group_by(mydata, temp) %>% summarise(count = n(), mean = mean(Response,
  na.rm = TRUE), SD = sd(Response, na.rm = TRUE), median = median(Response,
  na.rm = TRUE), IQR = IQR(Response, na.rm = TRUE))
```

```
# A tibble: 4 x 6
  temp          count mean    SD median  IQR
<chr>         <int> <dbl> <dbl> <dbl> <dbl>
1 Treatment1.First    100  7.95  2.98   8.11  4.61
2 Treatment1.Second   100  9.03  2.86   9.04  3.65
3 Treatment2.First    100  9.93  2.80   9.92  4.26
4 Treatment2.Second   100 11.1   3.02  11.4   3.94
```

I tried this package, it was only ok

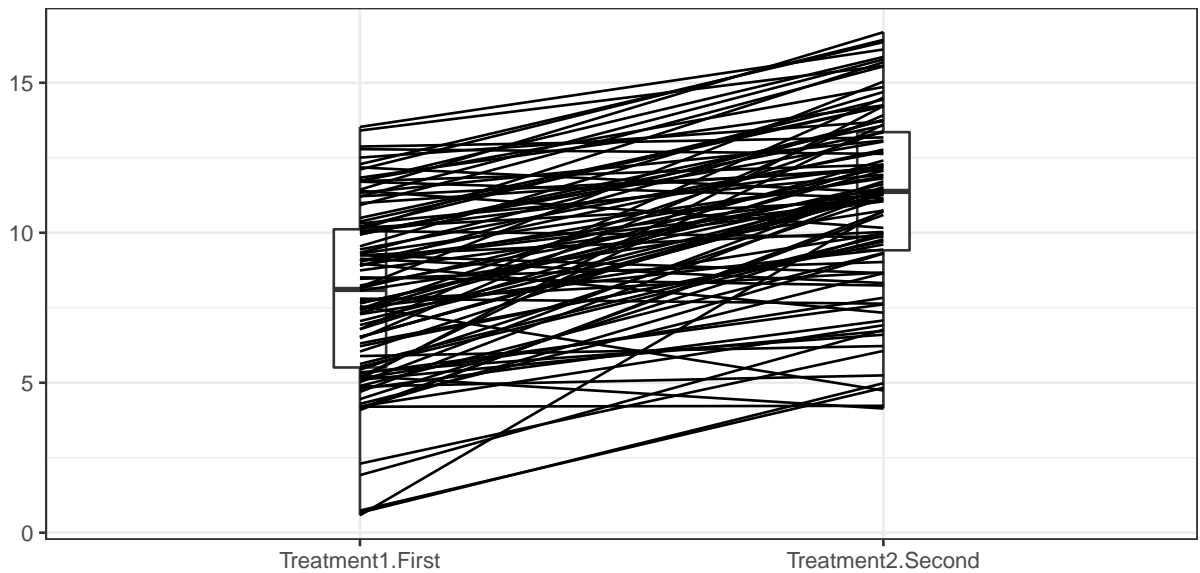
```
# B <- mydata[mydata$temp %in% 'Treatment1.First' |
# mydata$temp %in% 'Treatment2.Second',] B1 <-
# ggpubr::ggpaired(B, x='Treatment', y = 'Response', color =
# 'Treatment', line.color = 'gray', line.size = 0.4, palette
# = 'npg', , title='First period comparison') +
# stat_compare_means(paired = TRUE) C <- mydata[mydata$temp
# %in% 'Treatment2.First' | mydata$temp %in%
# 'Treatment1.Second',] C1 <- ggpubr::ggpaired(C,
# x='Treatment', y = 'Response', color = 'Treatment',
# line.color = 'gray', line.size = 0.4, palette = 'npg' ,
# title='Second period comparison')+
# stat_compare_means(paired = TRUE)

# print(B1);print(C1)
```

Use a package to plot AB sequence then BA sequence, should be mirror image ideally

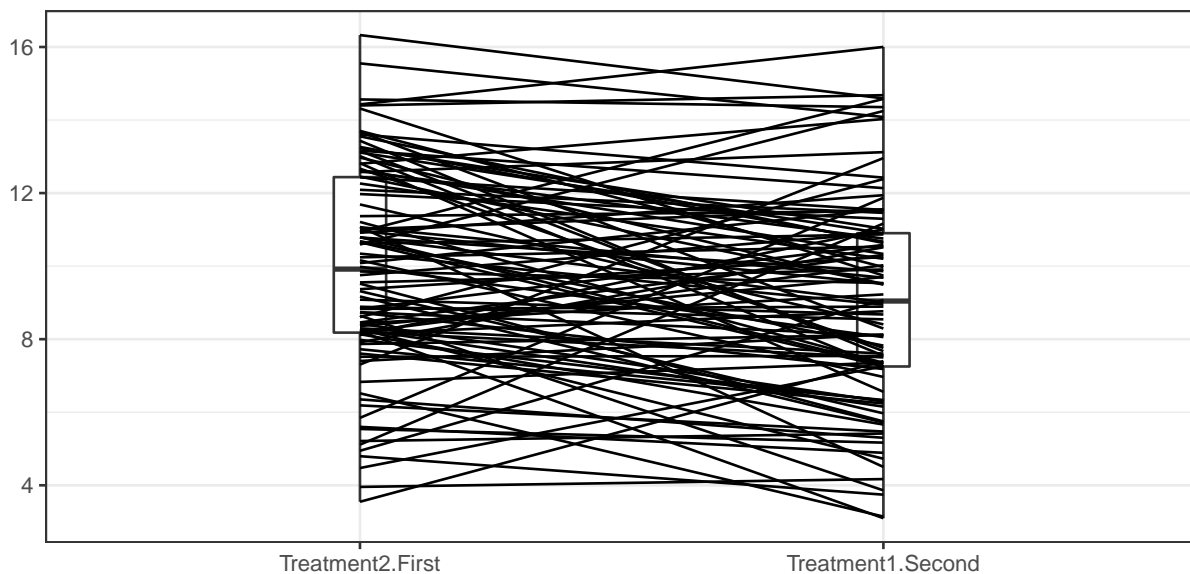
```
require("PairedData")
Treatment1.First <- subset(mydata, temp == "Treatment1.First",
  Response, drop = TRUE)
Treatment2.Second <- subset(mydata, temp == "Treatment2.Second",
  Response, drop = TRUE)

pd <- paired(Treatment1.First, Treatment2.Second)
plot(pd, type = "profile") + theme_bw()
```



```
cat("\n\n\\pagebreak\n")
```

```
Treatment2.First <- subset(mydata, temp == "Treatment2.First",  
  Response, drop = TRUE)  
Treatment1.Second <- subset(mydata, temp == "Treatment1.Second",  
  Response, drop = TRUE)  
  
pd <- paired(Treatment2.First, Treatment1.Second)  
plot(pd, type = "profile") + theme_bw()
```



My code to plot treatment comparisons

```

# http://www.surefoss.org/visualisation/visualizing-small-scale-paired-data-combining-boxplots-stripchart

# main effect 4 and an order effect of 1, no interaction so
# second period gets a boost of +1

co.plot <- function(mydata) {

  par(mfrow = c(2, 2))

  # creat a concatenated variable
  mydata$temp <- paste(mydata$Treatment, mydata$Order, sep = ".")

  # pull out AB data
  post <- as.vector(mydata[grepl("Treatment1.First", mydata$temp),
    4])
  pre <- as.vector(mydata[grepl("Treatment2.Second", mydata$temp),
    4])
  s <- seq(length(pre))
  par(bty = "l")

  # boxplot AB
  boxplot(post, pre, main = "Treatment1 -> Treatment2 (AB)",
    xlab = "Order", ylab = "Response", names = c("Period 1 (TRT1)",
      "Period 2 (TRT2)"), col = c("lightgreen", "lightblue"))

  # Add raw data with jitter
  stripchart(list(post, pre), vertical = T, pch = 16, method = "jitter",
    cex = 0.5, add = T)
  # join lines, this would be better if jittered points were
  # joined
  segments(rep(0.95, length(pre))[s], post[s], rep(2, length(pre))[s],
    pre[s], col = 1, lwd = 0.5)

  # Wilcoxon signed rank test
  res <- wilcox.test(post, pre, paired = T, conf.int = T)
  stripchart(post - pre, vertical = T, pch = 16, method = "jitter",
    main = "Median of all differences TRT1 - TRT2", ylab = "Difference:Treatment1-Treatment2",
    xlab = "Median of all differences +/-95%CI")
  points(1, res$estimate, col = "red", pch = 16, cex = 2)
  arrows(1, res$conf.int[1], 1, res$conf.int[2], col = "red",
    code = 3, lwd = 3, angle = 90)
  abline(h = 0, lty = 2) #Zero-effect line

  # repeat for BA sequence

```

```

pre <- as.vector(mydata[grepl("Treatment2.First", mydata$temp),
  4])
post <- as.vector(mydata[grepl("Treatment1.Second", mydata$temp),
  4])
s <- seq(length(pre))
par(bty = "l")

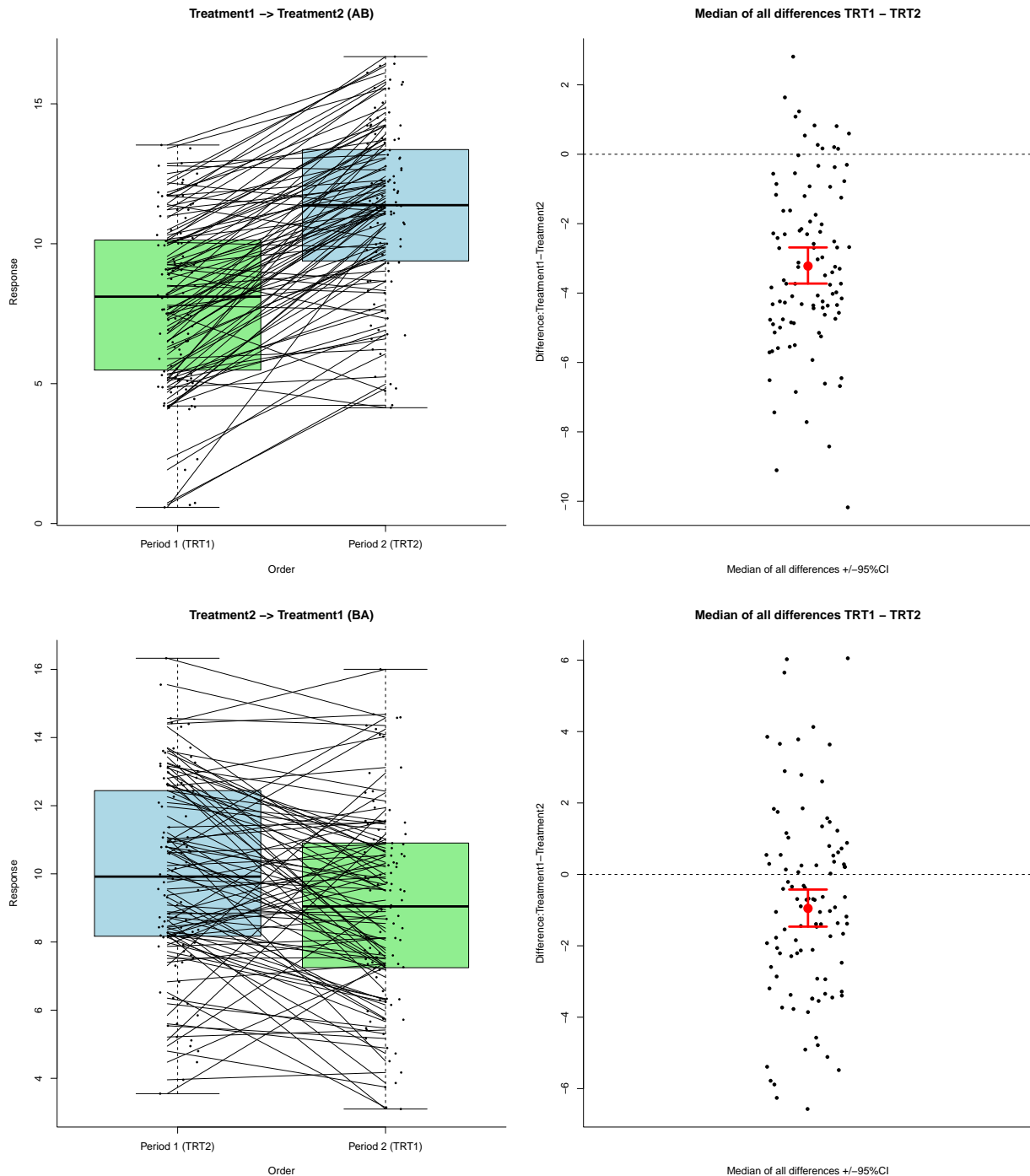
boxplot(pre, post, main = "Treatment2 -> Treatment1 (BA)",
  xlab = "Order", ylab = "Response", names = c("Period 1 (TRT2)",
    "Period 2 (TRT1)", col = c("lightblue", "lightgreen"))
stripchart(list(pre, post), vertical = T, pch = 16, method = "jitter",
  cex = 0.5, add = T)
segments(rep(0.95, length(pre))[s], pre[s], rep(2, length(pre))[s],
  post[s], col = 1, lwd = 0.5)

res <- wilcox.test(post, pre, paired = T, conf.int = T)
stripchart(post - pre, vertical = T, pch = 16, method = "jitter",
  main = "Median of all differences TRT1 - TRT2", ylab = "Difference:Treatment1-Treatment2",
  xlab = "Median of all differences +/-95%CI")
points(1, res$estimate, col = "red", pch = 16, cex = 2)
arrows(1, res$conf.int[1], 1, res$conf.int[2], col = "red",
  code = 3, lwd = 3, angle = 90)
abline(h = 0, lty = 2) # Zero-effect line
par(mfrow = c(1, 1))
}

```

Plot AB sequence then BA sequence, should be mirror image ideally

```
co.plot(mydata)
```



My code to plot all data in a useful way

```

d1 <- mydata
require(reshape)
d1$grp <- paste(d1$Treatment, d1$Order, sep=" ")
d1 <- rename(d1, c(Response="count"))
d1 <- rename(d1, c(grp="spray"))
d1 <- d1[order(d1$spray),]

attach(d1)
sprayTypes <- unique(spray)

y <- as.numeric(as.character(d1[,4]))

plot(y, as.factor(d1[,5]) , ylim=c(min(y),max(y)) , xlim=c(1, 5) , xaxt="n",
      main=paste("\nTruth: Intercept ",beta[1]," Treatment (main) effect ",beta[2]," order effect ",
                  "\n Between subjects SD",p2(sb)," within subjects SD", p2(sw) ),
      xlab="", ylab="Outcome measure", frame.plot =F , col="white")
##if not white I get a nasty boxplot

axis(1, at=1:4, labels=F )

text(x=1:length(sprayTypes)*1.05, par("usr")[3]-0.08 , labels = sprayTypes,
      srt = 45, pos = 2, xpd = TRUE,
      cex.axis=.5 )

##make colours
chk <- as.character(d1$spray)
x <- as.data.frame(table(chk))
freq <- x[,2]
value <- max(dput(1:dim(x)[1]))

```

1:4

```

IR <- value+1
clr <- rainbow(IR)
clr <- c("blue","red")
wR <- (2*IR-1:IR)/(2*IR)
##colour made
for (i in 1 : length(sprayTypes)){
  y <- count[spray == sprayTypes[i]]
  n <- sum(spray == sprayTypes[i])
  points(jitter(rep(i, n), amount = .1), y, pch = 16, cex = .5,
         col=ifelse(i<3,clr[1],clr[2])

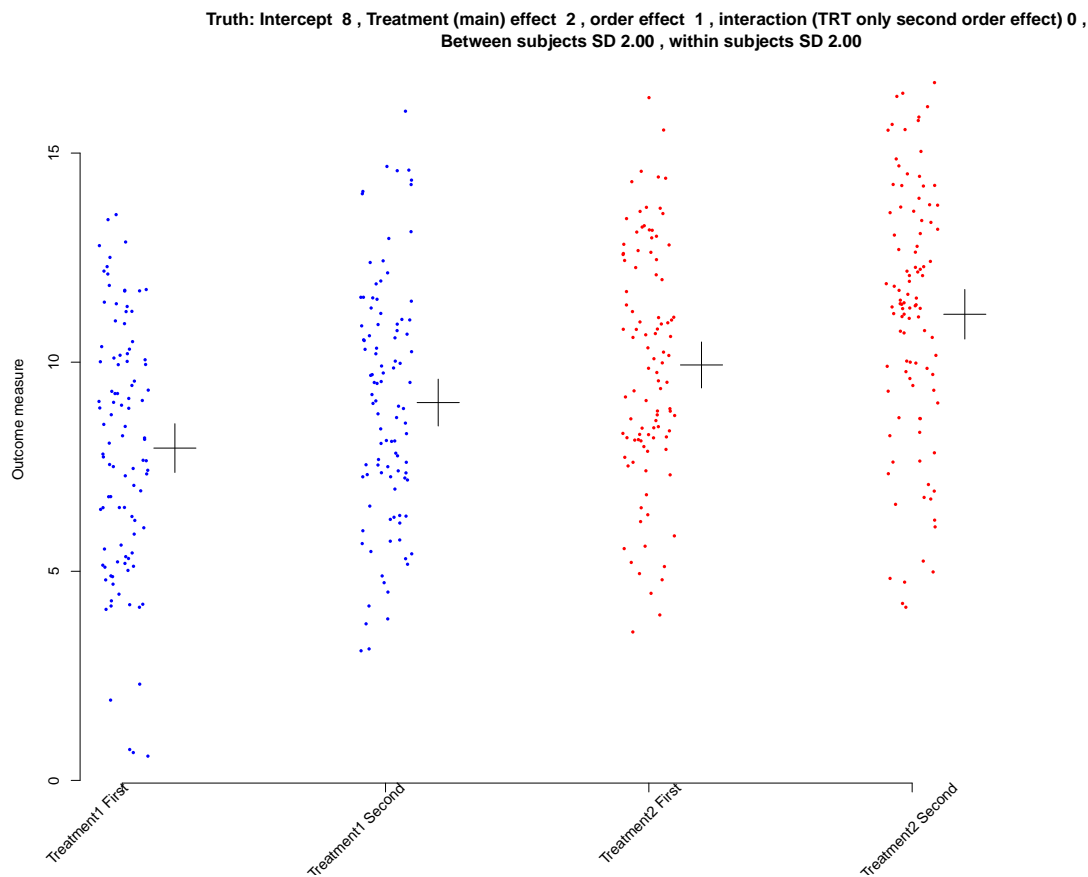
  )

  col=ifelse(Treatment=="Treatment1",clr[1],clr[2])

  lines(i + c(.12, .28), rep(mean(y), 2), lwd = 1, col="black")
  lines(rep(i + .2, 2),
        mean(y) + c(-1.96, 1.96) * sd(y) / sqrt(n), lwd = 1, col="black"

  )
}

```

This is a useful plot to look at all the data together. Perhaps lines joining the pairs might be an improvement to show between and within person variation. The crosses are the arithmetic means and 95% CIs. The first left cross is an estimate of the intercept. We can see the treatment effect, the red data is greater than blue. The order effect is apparent too, second period is higher than first. We are expecting this (we know the truth). We did power this study, so there is a high probability the main effect should manifest and to a lesser probability also the order. We know in truth there is no interaction.

Computing Environment

sessionInfo()

```
R version 3.6.1 (2019-07-05)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 17134)

Matrix products: default

locale:
[1] LC_COLLATE=English_United Kingdom.1252
[2] LC_CTYPE=English_United Kingdom.1252
[3] LC_MONETARY=English_United Kingdom.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United Kingdom.1252

attached base packages:
[1] stats      graphics  grDevices  utils      datasets
[6] methods    base

other attached packages:
[1] reshape_0.8.8      PairedData_1.1.1 lattice_0.20-38
[4] mvtnorm_1.0-11     gld_2.5          MASS_7.3-51.4
[7] forcats_0.4.0      stringr_1.4.0    dplyr_0.8.3
[10] purrr_0.3.2        readr_1.3.1      tidyr_0.8.3
[13] tibble_2.1.3        tidyverse_1.2.1  ggplot2_3.2.0
[16] nlme_3.1-140       knitr_1.23

loaded via a namespace (and not attached):
[1] tidyselect_0.2.5  xfun_0.8          haven_2.1.1
[4] colorspace_1.4-1  generics_0.0.2    vctrs_0.2.0
[7] htmltools_0.3.6   yaml_2.2.0        utf8_1.1.4
[10] rlang_0.4.0       e1071_1.7-2       pillar_1.4.2
[13] glue_1.3.1        withr_2.1.2       modelr_0.1.4
[16] readxl_1.3.1      plyr_1.8.4        munsell_0.5.0
[19] gtable_0.3.0      cellranger_1.1.0  rvest_0.3.4
[22] evaluate_0.14     labeling_0.3       lmom_2.8
[25] class_7.3-15      fansi_0.4.0        broom_0.5.2
[28] Rcpp_1.0.1        backports_1.1.4    scales_1.0.0
[31] formatR_1.7       jsonlite_1.6       hms_0.5.0
[34] digest_0.6.20     stringi_1.4.3      grid_3.6.1
[37] cli_1.1.0         tools_3.6.1        magrittr_1.5
[40] lazyeval_0.2.2    crayon_1.3.4       pkgconfig_2.0.2
[43] zeallot_0.1.0     xml2_1.2.0         lubridate_1.7.4
[46] assertthat_0.2.1  rmarkdown_1.14     httr_1.4.0
[49] rstudioapi_0.10   R6_2.4.0           compiler_3.6.1
```

This took 76.16 seconds to execute.