THE TWO-PERIOD CROSS-OVER CLINICAL TRIAL

Eamonn O'Brien

25 July, 2019

Contents

Hardcode in the data from Hills and Armitage Br. J. clin. Pharmac. (1979), 8, 7-20 3
Period and treatment summary stats, agree with paper 6
Fit a random effects model (not checking for order nor interaction) and linear regression model . . 7
Function to analyse using permutation approach to duplicate Stephen Senn talk 10
Execute the simulations 12
Plot the distributions 13
Permutation p values 14
Summary statisitcs 15
Computing Environment 16

Contents

List of Figures

List of Tables

Hardcode in the data from Hills and Armitage Br. J. clin. Pharmac. (1979), 8, 7-20

```
patient1 <- c(1, 3, 4, 6, 7, 9, 11, 13, 16, 18, 19, 21, 22, 24,
    25, 27, 28)
y1 <- c(8, 14, 8, 9, 11, 3, 6, 0, 13, 10, 7, 13, 8, 7, 9, 10,
    2)
treatment1 <- rep("Treatment1", length(patient1))
order <- rep("First", length(treatment1))


treatment2 <- rep("Treatment2", length(treatment1))
order2 <- rep("Second", length(treatment1))
y2 <- c(5, 10, 0, 7, 6, 5, 0, 0, 12, 2, 5, 13, 10, 7, 0, 6, 2)


patient2 <- c(2, 5, 8, 10, 12, 14, 15, 17, 20, 23, 26, 29)
y1a <- c(12, 6, 13, 8, 8, 4, 8, 2, 8, 9, 7, 7)
treatment2a <- rep("Treatment2", length(patient2))
order3 <- rep("First", length(patient2))


y1b <- c(11, 8, 9, 8, 9, 8, 14, 4, 13, 7, 10, 6)
order4 <- rep("Second", length(patient2))
treatment3a <- rep("Treatment1", length(patient2))

grp1 <- cbind(as.numeric(patient1), (as.character(treatment1)),
    (as.character(order)), as.numeric(y1))

grp2 <- cbind(as.numeric(patient1), (as.character(treatment2)),
    (as.character(order2)), as.numeric(y2))

grp3 <- cbind(as.numeric(patient2), (as.character(treatment2a)),
    (as.character(order3)), as.numeric(y1a))

grp4 <- cbind(as.numeric(patient2), (as.character(treatment3a)),
    (as.character(order4)), as.numeric(y1b))


all <- as.data.frame(rbind(grp1, grp2, grp3, grp4))

all <- plyr::arrange(all, V1, V2)

names(all) <- c("Patient", "Treatment", "Order", "Response")

all$Patient <- as.numeric(as.character(all$Patient))
all$Response <- as.numeric(as.character(all$Response))

all <- plyr::arrange(all, Patient, Treatment)


knitr::kable(all)
```

Patient	Treatment	Order	Response
1	Treatment1	First	8
1	Treatment2	Second	5
2	Treatment1	Second	11
2	Treatment2	First	12
3	Treatment1	First	14
3	Treatment2	Second	10
4	Treatment1	First	8
4	Treatment2	Second	0
5	Treatment1	Second	8
5	Treatment2	First	6
6	Treatment1	First	9
6	Treatment2	Second	7
7	Treatment1	First	11
7	Treatment2	Second	6
8	Treatment1	Second	9
8	Treatment2	First	13
9	Treatment1	First	3
9	Treatment2	Second	5
10	Treatment1	Second	8
10	Treatment2	First	8
11	Treatment1	First	6
11	Treatment2	Second	0
12	Treatment1	Second	9
12	Treatment2	First	8
13	Treatment1	First	0
13	Treatment2	Second	0
14	Treatment1	Second	8
14	Treatment2	First	4
15	Treatment1	Second	14
15	Treatment2	First	8
16	Treatment1	First	13
16	Treatment2	Second	12
17	Treatment1	Second	4
17	Treatment2	First	2
18	Treatment1	First	10
18	Treatment2	Second	2
19	Treatment1	First	7
19	Treatment2	Second	5
20	Treatment1	Second	13
20	Treatment2	First	8
21	Treatment1	First	13
21	Treatment2	Second	13
22	Treatment1	First	8
22	Treatment2	Second	10
23	Treatment1	Second	7
23	Treatment2	First	9
24	Treatment1	First	7
24	Treatment2	Second	7
25	Treatment1	First	9
25	Treatment2	Second	0
26	Treatment1	Second	10
26	Treatment2	First	7

Patient	Treatment	Order	Response
27	Treatment1	First	10
27	Treatment2	Second	6
28	Treatment1	First	2
28	Treatment2	Second	2
29	Treatment1	Second	6
29	Treatment2	First	7

Period and treatment summary stats, agree with paper

```
# with(all, tapply(Response, list(Treatment, Order), mean))

require(tidyverse)
all %>% group_by(Treatment, Order) %>% summarise_each(funs(n = length(!is.na(.)),
    mean, sd, se = sd(.)/sqrt(n())), Response)
```

```
# A tibble: 4 x 6
# Groups:   Treatment [2]
  Treatment  Order      n  mean    sd    se
  <fct>      <fct>  <int> <dbl> <dbl> <dbl>
1 Treatment1 First     17  8.12  3.84 0.931
2 Treatment1 Second    12  8.92  2.81 0.811
3 Treatment2 First     12  7.67  2.99 0.865
4 Treatment2 Second    17  5.29  4.25 1.03
```

```
# calc difference in treatements and summarise
w <- spread(select(all, -c(Order)), Treatment, Response)

w <- w %>% select(Treatment1, Treatment2) %>% mutate(Response = Treatment1 -
    Treatment2)  #%>% head()

w %>% summarise(mean = mean(Response), sd = sd(Response), n = length(!is.na(Response)),
    se = sd/sqrt(n))
```

```
      mean       sd  n        se
1 2.172414 3.317367 29 0.6160197
```

Fit a random effects model (not checking for order nor interaction) and linear regression model

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

```
            numDF denDF   F-value p-value
(Intercept)     1    28 146.48262  <.0001
Treatment       1    28  12.43644  0.0015
```

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

```
                       Value Std.Error DF   t-value
(Intercept)          8.448276 0.6818213 28 12.390749
TreatmentTreatment2 -2.172414 0.6160197 28 -3.526533
                                  p-value
(Intercept)         0.0000000000006969003
TreatmentTreatment2 0.0014712573664130472
```

```
intervals(f)
```

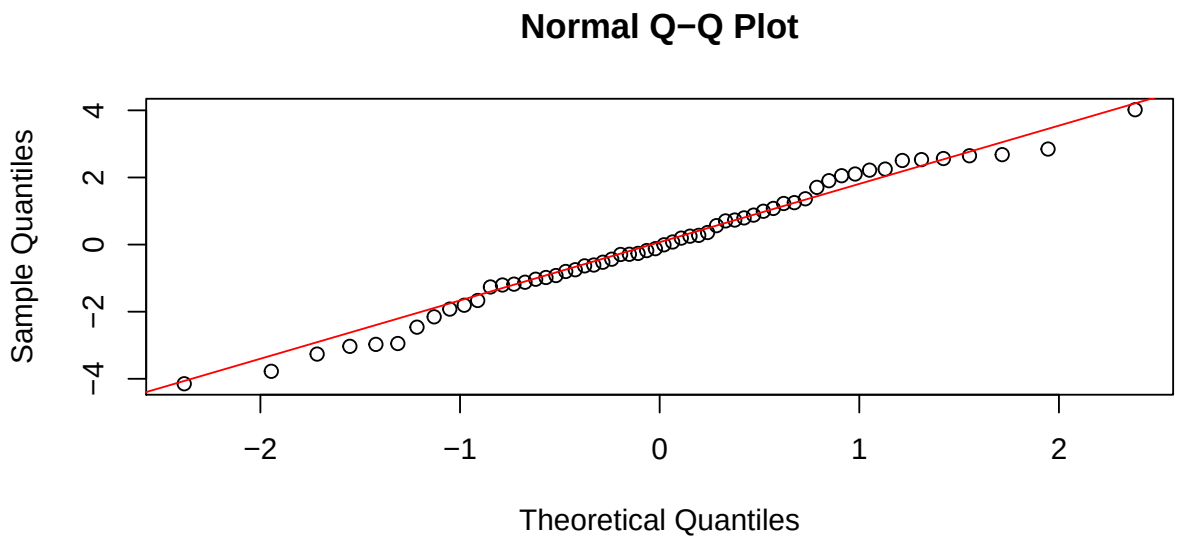
```
Approximate 95% confidence intervals

 Fixed effects:
                       lower      est.      upper
(Intercept)          7.051628  8.448276  9.8449234
TreatmentTreatment2 -3.434273 -2.172414 -0.9105547
attr(,"label")
[1] "Fixed effects:"

 Random Effects:
  Level: Patient
                   lower     est.   upper
sd((Intercept)) 1.963652 2.824724 4.06338

 Within-group standard error:
   lower     est.    upper
1.805238 2.345733 3.048054
```

```
qqnorm(resid(f), main = "Normal Q-Q Plot")
qqline(resid(f), col = "red")
```




```
# collect the treatment effect estimate to make inferences
# later
z <- as.matrix(summary(f)$tTable)
Treatment <- z[2, 1][[1]]


f <- lm(Response ~ Treatment, data = all, na.action = "na.omit")
summary(f)
```

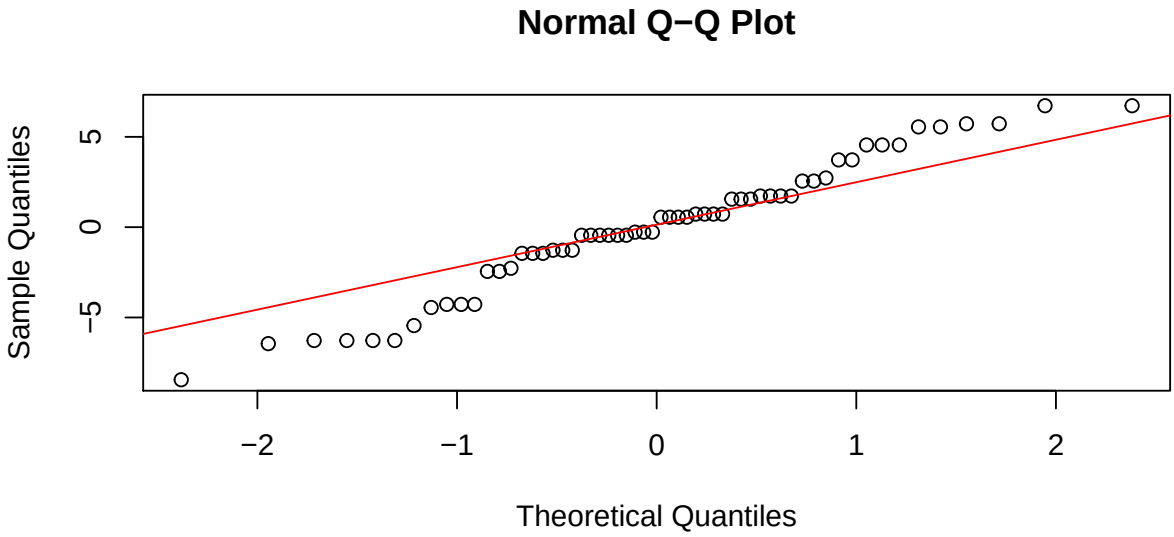
```
Call:
lm(formula = Response ~ Treatment, data = all, na.action = "na.omit")

Residuals:
    Min      1Q  Median      3Q     Max
-8.4483 -1.4483  0.1379  1.7241  6.7241

Coefficients:
                    Estimate Std. Error t value
(Intercept)           8.4483     0.6818  12.391
TreatmentTreatment2  -2.1724     0.9642  -2.253
                            Pr(>|t|)
(Intercept)         <0.0000000000000002 ***
TreatmentTreatment2              0.0282 *
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.672 on 56 degrees of freedom
Multiple R-squared:  0.08311,   Adjusted R-squared:  0.06674
F-statistic: 5.076 on 1 and 56 DF,  p-value: 0.0282
```

```
qqnorm(resid(f), main = "Normal Q-Q Plot")
qqline(resid(f), col = "red")
```


Normal Q−Q Plot
Sample Quantiles
5
0
−5
−2
−1
0
1
2
Theoretical Quantiles

Function to analyse using permutation approach to duplicate Stephen Senn talk

```
Dq1 <- all
library(data.table)
Dq1 <- as.data.table(all)

# function to get permuted distribution of treatment effect

perm.dist <- function(block = "yes", n.sims = 10000) {

    # set up an array to store parameter estimates
    estArray <- array(NA, dim = c(n.sims, 4))

    for (s in 1:n.sims) {

        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        # permute

        if (block == "yes") {

            # permute within person
            permz <- Dq1[, `:=`(y, sample(Response)), by = Patient]

        } else {

            # no blocking
            permz <- Dq1[, `:=`(y, sample(Response))]

        }

        # ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        # analysis

        # respecting blocking
        possibleError <- tryCatch(f1 <- lme(y ~ Treatment, random = ~1 |
            Patient, data = permz, method = "REML"), error = function(e) e)

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

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

            modelint <- possibleError
            z <- as.matrix(summary(modelint)$tTable)

        }


        # ignoring blocking
        possibleError2 <- tryCatch(f0 <- lm(y ~ Treatment, data = permz),
            error = function(e) e)

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

            modelint1 <- possibleError2
```

```
            zz <- as.matrix(summary(modelint1)$coefficients)


        }


        estArray[s, 1] <- z[2, 1][[1]]  # collect trt effect estimate
        estArray[s, 2] <- vcov(modelint)[2, 2]  # collect variance of trt effect estimate


        estArray[s, 3] <- zz[2, 1][[1]]  # collect trt effect estimate
        estArray[s, 4] <- vcov(modelint1)[2, 2]  # collect variance of trt effect estimate

    }

    list(estArray = estArray)
}
```

Execute the simulations

```
block <- perm.dist(block = "yes", n.sims = 10000)

no.block <- perm.dist(block = "no", n.sims = 10000)
```

Plot the distributions

```
plot(density(block))
plot(density(no.block))
```

Permutation p values

```
# see Senn 34.09mins right panel youtube, good match!
sum(abs(block$estArray[, 1]) >= abs(Treatment))/10000  # Senn 0.0014
```

[1] 0.0014

```
sum(abs(no.block$estArray[, 1]) >= abs(Treatment))/10000  # Senn 0.034
```

[1] 0.0288

Summary statisitcs

```
apply(block$estArray, 2, summary)
```

```
                [,1]      [,2]         [,3]      [,4]
Min.    -2.448275862 0.3091558 -2.448275862 0.8945983
1st Qu. -0.517241379 0.5214880 -0.517241379 1.0007644
Median   0.034482759 0.5384745  0.034482759 1.0092577
Mean    -0.006193103 0.5292576 -0.006193103 1.0046492
3rd Qu.  0.517241379 0.5459486  0.517241379 1.0129947
Max.     2.586206897 0.5479870  2.586206897 1.0140139
```

```
apply(block$estArray, 2, var)
```

```
[1] 0.5256265744 0.0006348833 0.5256265744 0.0001587210
```

```
apply(no.block$estArray, 2, summary)
```

```
                [,1]      [,2]         [,3]      [,4]
Min.    -3.75862069 0.3471208 -3.75862069 0.7617632
1st Qu. -0.72413793 0.8626212 -0.72413793 0.9909122
Median  -0.03448276 0.9710379 -0.03448276 1.0063700
Mean    -0.03222759 0.9197404 -0.03222759 0.9960086
3rd Qu.  0.65517241 1.0063700  0.65517241 1.0123153
Max.     3.41379310 1.0140139  3.41379310 1.0140139
```

```
apply(no.block$estArray, 2, var)
```

```
[1] 1.0085490390 0.0126192077 1.0085490390 0.0006195791
```

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] data.table_1.12.2 nlme_3.1-140      forcats_0.4.0
 [4] stringr_1.4.0     dplyr_0.8.3       purrr_0.3.2
 [7] readr_1.3.1       tidyr_0.8.3       tibble_2.1.3
[10] ggplot2_3.2.0     tidyverse_1.2.1   knitr_1.23

loaded via a namespace (and not attached):
 [1] tidyselect_0.2.5 xfun_0.8         haven_2.1.1
 [4] lattice_0.20-38  colorspace_1.4-1 generics_0.0.2
 [7] vctrs_0.2.0      htmltools_0.3.6  yaml_2.2.0
[10] utf8_1.1.4       rlang_0.4.0      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    fansi_0.4.0      highr_0.8
[25] broom_0.5.2      Rcpp_1.0.1       scales_1.0.0
[28] backports_1.1.4  formatR_1.7      jsonlite_1.6
[31] hms_0.5.0        digest_0.6.20    stringi_1.4.3
[34] grid_3.6.1       cli_1.1.0        tools_3.6.1
[37] magrittr_1.5     lazyeval_0.2.2   crayon_1.3.4
[40] pkgconfig_2.0.2  zeallot_0.1.0    xml2_1.2.0
[43] lubridate_1.7.4  assertthat_0.2.1 rmarkdown_1.14
[46] httr_1.4.0       rstudioapi_0.10  R6_2.4.0
[49] compiler_3.6.1
```

This took 655.85 seconds to execute.