Performing GPC in a paired design

Brice Ozenne

February 20, 2026

This vignette describes how to use Generalized Pairwise comparisons (GPC) in a paired design. This for instance corresponds to the Diabetic Retinopathy Study (DRS) contained in the survival ® package where 197 patients had one of their eye randomized to laser treatment while the other did not receive any treatment:

```
data(diabetic, package = "survival")
head(diabetic)
```

```
   id laser age    eye trt risk  time status
1   5 argon  28   left   0    9 46.23      0
2   5 argon  28  right   1    9 46.23      0
3  14 xenon  12   left   1    8 42.50      0
4  14 xenon  12  right   0    6 31.30      1
5  16 xenon   9   left   1   11 42.27      0
6  16 xenon   9  right   0   11 42.27      0
```

The outcome was time to blindness (visual acuity drop below a certain threshold). In the real study `status` equal to 0 mixes death and censoring (due to drop-out or end of study) but this complication will be neglected here for simplicity.

We will replicate some of the analyzes presented in Matsouaka (2022). In this paper they split the dataset into juvenile and adult patients:

```
diabetic$juvenile <- diabetic$age <= 19
library(LMMstar)
summarize(age ~ juvenile, data = diabetic[!duplicated(diabetic$id),])
```

```
  juvenile observed missing     mean       sd min q1 median    q3 max
1    FALSE       83       0 35.30120 11.242054  20 25     34 45.00  58
2     TRUE      114       0 10.21053  4.713892   1  7     10 13.75  19
```

and we will focus on the juvenile patients:

```
diabeticJ <- diabetic[diabetic$juvenile,]
```

1 Wald methods (Gehan scoring rule)

To mimic the methodology underlying the results presented in Table 1 of Matsouaka (2022), we perform GPC stratified by patient using the Gehan scoring rule:

```
library(BuyseTest)
e.BTjuv <- BuyseTest(trt ~ tte(time,status) + strata(id, match = TRUE),
                     data = diabeticJ, trace = FALSE,
                     scoring.rule =  "Gehan")
model.tables(e.BTjuv, percentage = FALSE)
```

	endpoint	total	favorable	unfavorable	neutral	uninf	Delta	lower.ci	upper.ci	p.value
1	time	114	39	21	3	51	0.1578947	0.02591623	0.2844633	0.01922741

Indeed this scoring rule does not involve any extra-modeling, only evaluating the patient specific net benefit and averaging them:

```
mean(coef(e.BTjuv, strata = TRUE))
```

```
[1] 0.1578947
```

Matsouaka (2022) propose to estimate the standard error as:

```
N <- nobs(e.BTjuv)["pairs"]
pw <- coef(e.BTjuv, statistic = "favorable")
pl <- coef(e.BTjuv, statistic = "unfavorable")
sqrt((pw + pl - (pw - pl)^2)/N)
```

```
      time
0.06631828
```

which matches what `BuyseTest` output:

```
confint(e.BTjuv)
```

	estimate	se	lower.ci	upper.ci	null	p.value
time	0.1578947	0.06631828	0.02591623	0.2844633	0	0.01922741

By default `confint` uses a hyperbolic tangent to compute confidence intervals (CIs), which will then differ from the 'Wald' discussed in Matsouaka (2022). These 'untransformed Wald' CIs can be retrieved by setting the argument `transform` to `FALSE`:

```
confint(e.BTjuv, transform = FALSE)
```

	estimate	se	lower.ci	upper.ci	null	p.value
time	0.1578947	0.06631828	0.02791329	0.2878762	0	0.01727214

Note: naively one may think to estimate the standard error as:

```
sqrt(var(coef(e.BTjuv, strata = TRUE))/N)
```

```
     pairs
0.06661108
```

This is equivalent (in large samples to the previous formula). Indeed:

$$\mathbb{P}\left[X > Y\right] + \mathbb{P}\left[Y > X\right] - (\mathbb{P}\left[X > Y\right] - \mathbb{P}\left[Y > X\right])^2$$
$$=\mathbb{P}\left[X > Y\right] + \mathbb{P}\left[Y > X\right] - \mathbb{P}\left[X > Y\right]^- \mathbb{P}\left[Y > X\right]^2 + 2\mathbb{P}\left[X > Y\right]\mathbb{P}\left[Y > X\right]$$
$$=\mathbb{P}\left[X > Y\right](1 - \mathbb{P}\left[X > Y\right]) + \mathbb{P}\left[Y > X\right](1 - \mathbb{P}\left[Y > X\right]) + 2\mathbb{P}\left[X > Y\right]\mathbb{P}\left[Y > X\right]$$
$$=\mathbb{P}\left[X > Y\right](1 - \mathbb{P}\left[X > Y\right]) + \mathbb{P}\left[Y > X\right](1 - \mathbb{P}\left[Y > X\right])$$
$$\quad - 2(0 - \mathbb{P}\left[X > Y\right]\mathbb{P}\left[Y > X\right] - \mathbb{P}\left[X > Y\right]\mathbb{P}\left[Y > X\right] + \mathbb{P}\left[X > Y\right]\mathbb{P}\left[Y > X\right]$$
$$=\mathbb{V}ar\left[\mathbb{1}_{X>Y}\right] + \mathbb{V}ar\left[\mathbb{1}_{X<Y}\right] - 2\mathbb{C}ov\left(\mathbb{1}_{X>Y}, \mathbb{1}_{X<Y}\right)$$
$$=\mathbb{V}ar\left[\mathbb{1}_{X>Y} - \mathbb{1}_{X<Y}\right]$$

There is only a factor `N/(N-1)` difference between the two:

```
sqrt(var(coef(e.BTjuv, strata = TRUE))/N) * sqrt((N-1)/N)
```

```
     pairs
0.06631828
```

2 MOVER method (Gehan scoring rule)

The method recommended by Matsouaka (2022) is the MOVER approach, which has been developed for a binary scoring rule (e.g. Gehan). An experimental function with the same name has been implemented in the BuyseTest package:

```
BuyseTest:::mover(e.BTjuv)
```

```
  estimate      lower      upper     pvalue
0.15789474 0.02540421 0.28317729 0.01967878
```

leading to the same results as those of the table 1 in the original article, up to rounding.

3 Wald methods (Peron scoring rule)

To better account for censoring one could use the Peron scoring rule where the survival is estimated across all subjects within a treatment group. One has to specify the survival model, otherwise, the BuyseTest function will estimate a treatmnet and strata specific survival curve which is impossible to perform here. The `model.tte` argument can be used to specify such survival model:

```
library(prodlim)
e.BTjuv2 <- BuyseTest(trt ~ tte(time,status) + strata(id, match = TRUE),
                      data = diabeticJ, trace = FALSE,
                      model.tte = prodlim(Hist(time,status)~ trt, data = diabeticJ))
model.tables(e.BTjuv2, percentage = FALSE)
```

```
  endpoint total favorable unfavorable neutral    uninf    Delta   lower.ci  upper.ci
1     time   114  47.36525    24.29552       3 39.33923 0.202366 0.05045454 0.3451254
      p.value
1 0.009329589
```

Ignoring the uncertainty of the survival model, the standard would be:

```
c(sqrt(var(coef(e.BTjuv2, strata = TRUE))/N),
  sqrt(var(coef(e.BTjuv2, strata = TRUE))/N) * sqrt((N-1)/N)
  )
```

```
     pairs       pairs
0.06595510 0.06566518
```

depending on whether a small sample correction is used or not. This matches the output of `BuyseTest` when ignoring the uncertainty of the survival model:

```
model.tte <- prodlim(Hist(time,status)~ trt, data = diabeticJ)
attr(model.tte, "iidNuisance") <- FALSE
confint(BuyseTest(trt ~ tte(time,status) + strata(id, match = TRUE),
                  data = diabeticJ, trace = FALSE,
                  model.tte = model.tte))
```

```
     estimate         se   lower.ci  upper.ci null     p.value
time 0.202366 0.06566518 0.07088227 0.3269375    0 0.002726979
```

⚠ Because the pairwise win and loss score are no more binary, the previous formula no more simplifies into the formula presented in Matsouaka (2022):

```
pw.peron <- coef(e.BTjuv2, statistic = "favorable")
pl.peron <- coef(e.BTjuv2, statistic = "unfavorable")
sqrt((pw.peron + pl.peron - (pw.peron - pl.peron)^2)/N)
```

```
     time
0.07179718
```

To account for the uncertainty of the survival model, `BuyseTest` outputs a slightly higher standard error:

```
confint(e.BTjuv2)
```

```
        estimate         se   lower.ci  upper.ci null     p.value
time 0.202366 0.07569815 0.05045454 0.3451254    0 0.009329589
```

This is achieved by considering two sources of uncertainty:

- average of a finite number of pairs:

```
pw.peronS <- coef(e.BTjuv2, statistic = "favorable", strata = TRUE)
pl.peronS <- coef(e.BTjuv2, statistic = "unfavorable", strata = TRUE)
Hterm1 <- (pw.peronS - pl.peronS) - (pw.peron - pl.peron)
```

- propage the uncertainty of the survival model to the net benefit. Because each pair appear twice (control and treatment) the impact of removing a pair on the net benefit is stored in the control and the treated is set to 0:

```
Hterm2.obs <- e.BTjuv2@iidNuisance$favorable - e.BTjuv2@iidNuisance$unfavorable
Hterm2.pair <- Hterm2.obs[diabeticJ$trt==0]
table(Hterm2.obs[diabeticJ$trt==1])
```

```
  0
114
```

After rescaling the terms by a factor N (number of pairs, to account for the pooling) we retrieve the uncertainty output by `BuyseTest`:

```
c(average = sqrt(crossprod((Hterm1/N))),
  nuisance = sqrt(crossprod((Hterm2.pair/N))),
  all = sqrt(crossprod((Hterm1/N + Hterm2.pair/N))))
```

```
   average    nuisance         all
0.06566518 0.02084622 0.07569815
```

4 More general cross-over designs

Another type of paired design is a cross-over design where each patient may repeatedly experience each treatment. As an example, we will consider the PROFIL trial whose dataset is available in the BuyseTest package:

```
data(profil, package = "BuyseTest")
profil <- profil[order(profil$id),]
profil[profil$id==1 & profil$period==1,]
```
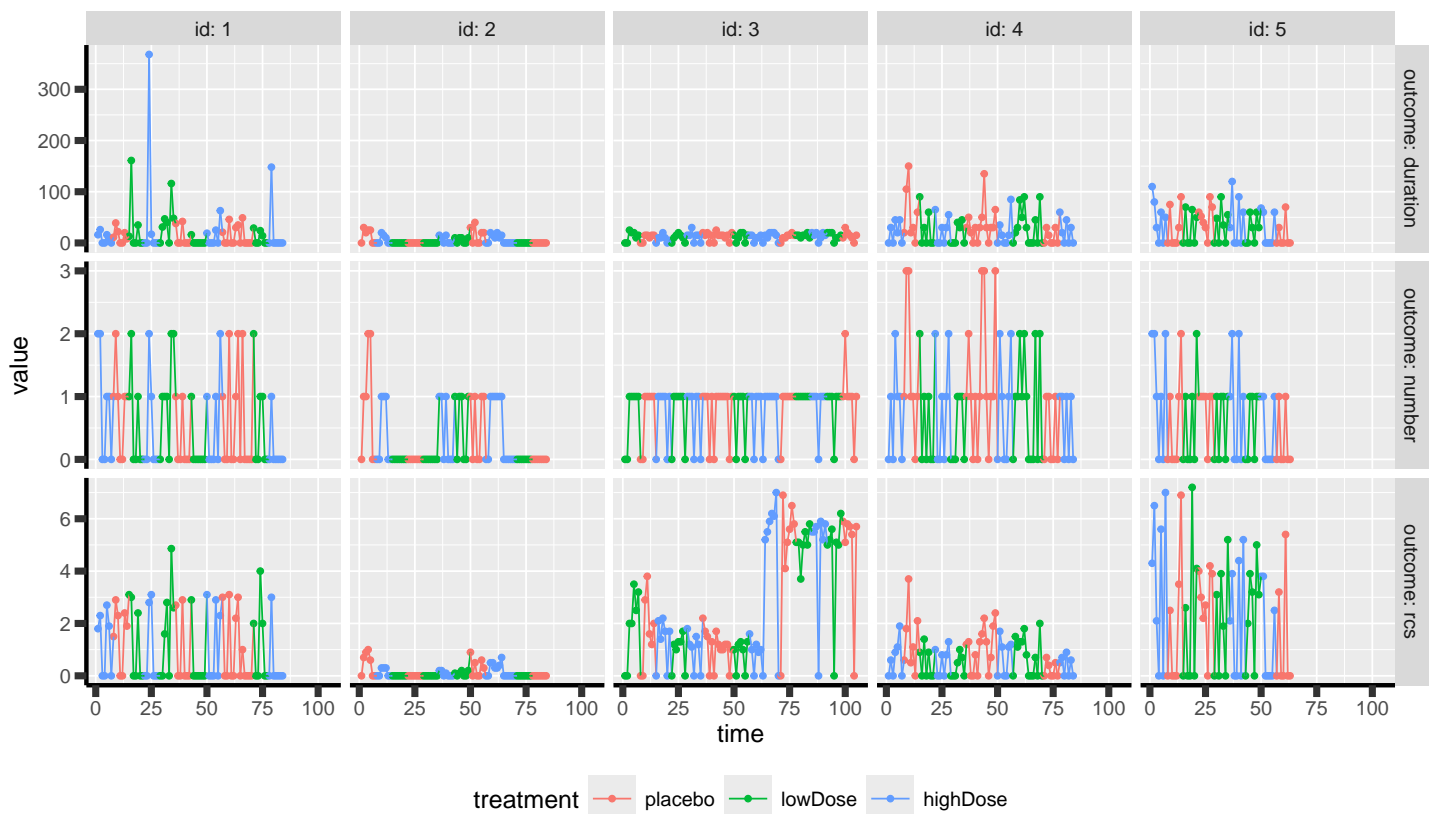
	id	age	male	period	time	treatment	rcs	number	duration
1	1	23	0	1	1	highDose	1.8	2	16
2	1	23	0	1	2	highDose	2.3	2	26
3	1	23	0	1	3	highDose	0.0	0	0
4	1	23	0	1	4	highDose	0.0	0	0
5	1	23	0	1	5	highDose	2.7	1	16
6	1	23	0	1	6	highDose	1.9	1	10
7	1	23	0	1	7	highDose	0.0	0	0
8	1	23	0	1	8	placebo	1.5	1	11
9	1	23	0	1	9	placebo	2.9	2	39
10	1	23	0	1	10	placebo	2.3	1	22
11	1	23	0	1	11	placebo	0.0	0	0
12	1	23	0	1	12	placebo	0.0	0	0
13	1	23	0	1	13	placebo	2.4	1	20
14	1	23	0	1	14	placebo	1.9	1	8
15	1	23	0	1	15	lowDose	3.1	1	13
16	1	23	0	1	16	lowDose	3.0	2	161
17	1	23	0	1	17	lowDose	0.0	0	0
18	1	23	0	1	18	lowDose	0.0	0	0
19	1	23	0	1	19	lowDose	2.4	1	35
20	1	23	0	1	20	lowDose	0.0	0	0
21	1	23	0	1	21	lowDose	0.0	0	0

The software output displays the information of the first patient relative to the first period (out of 4) during which the patient is sequentially assigned one of three treatments and her outcomes (rcs, number, and duration) are monitored daily. To obtain a graphical display of the outcomes over time we first reshape the data:

```
profil.L <- reshape(profil, direction = "long", idvar = c("id","time"),
                    varying = c("rcs","number","duration"), v.names = "value",
                    timevar = "outcome", times = c("rcs","number","duration"))
```

and make a spaghetti plot for the first 5 patients:

```
ggRCS <- ggplot(profil.L[profil.L$id %in% 1:5,],
                aes(x = time, group = id, color = treatment, y = value))
ggRCS <- ggRCS + geom_point() + geom_line()
ggRCS <- ggRCS + facet_grid(outcome~id, labeller = label_both, scales = "free_y")
ggRCS <- ggRCS + labs(y="")
ggRCS
```

4.1 With 2 treatment groups

Since `BuyseTest` can only handle two treatment group, we restrict the dataset to placebo and low dose:

```
lowProfil <- profil[profil$treatment %in% c("placebo","lowDose"),]
lowProfil$treatment <- droplevels(lowProfil$treatment)
```

We will use the following hierarchy and threshold of clinical relevance:

```
fff <- treatment ~ cont(rcs, threshold = 1.45, operator = "<0") + cont(number, threshold =
    0.35, operator = "<0") + cont(duration, threshold = 3, operator = "<0")
```

One could either run a separate GPC for each patient:

```
ls.lowGPC <- by(lowProfil, lowProfil$id, BuyseTest, formula = fff, trace = FALSE)
df.lowGPC <- data.frame(do.call(rbind,lapply(ls.lowGPC, coef)),
                        do.call(rbind,lapply(ls.lowGPC, nobs)),
                        Buyse = as.numeric(NA), CMH = as.numeric(NA))
df.lowGPC$Buyse <- with(df.lowGPC, pairs/sum(pairs))
df.lowGPC$CMH <- with(df.lowGPC, (pairs/(placebo+lowDose))/sum(pairs/(placebo+lowDose)))
head(df.lowGPC)
```

	rcs_t1.45	number_t0.35	duration_t3	placebo	lowDose	pairs	Buyse	CMH
1	-0.016581633	-0.015306122	-0.021683673	28	28	784	0.04694049	0.0364368
2	0.000000000	0.153061224	0.183673469	28	28	784	0.04694049	0.0364368
3	-0.001632653	0.003265306	-0.008979592	35	35	1225	0.07334451	0.0455460
4	0.117346939	0.225765306	0.154336735	28	28	784	0.04694049	0.0364368

7

```
5 -0.043083900 -0.047619048 -0.029478458        21        21    441 0.02640402 0.0273276
6  0.102040816  0.092970522  0.077097506        21        21    441 0.02640402 0.0273276
```

and pool the patient-specific Net Treatment Benefits. Different weighting scheme are possible, e.g. same weight for all patients, weight dependent on the number of blocks experienced by the patient:

```
rbind(average = colMeans(df.lowGPC[,1:3]),
      Buyse = apply(df.lowGPC[,1:3], 2, weighted.mean, w = df.lowGPC$Buyse),
      CMH = apply(df.lowGPC[,1:3], 2, weighted.mean, w = df.lowGPC$CMH))
```

```
          rcs_t1.45 number_t0.35 duration_t3
average  0.02741742   0.02755903  0.03397497
Buyse    0.01628547   0.03730092  0.04215064
CMH      0.02145018   0.03266743  0.03869945
```

This can be replicated using a single call to BuyseTest specifying a strata in the formula:

```
lowGPC <- BuyseTest(update(fff, . ~ . + strata(id, match = TRUE)),
                    data = lowProfil, trace = FALSE)
confint(lowGPC)
```

```
              estimate         se     lower.ci   upper.ci null   p.value
rcs_t1.45    0.02741742 0.02047690 -0.01273920 0.06748574    0 0.1808076
number_t0.35 0.02755903 0.03139999 -0.03401048 0.08892015    0 0.3803606
duration_t3  0.03397497 0.02801978 -0.02099009 0.08873527    0 0.2256649
```

By default, equal weights are given to each patient but other weighting schemes can be used by specifying the pool.strata argument:

```
lowGPC_Buyse <- BuyseTest(update(fff, . ~ . + strata(id, match = TRUE)),
                          data = lowProfil, pool.strata = "Buyse", trace = FALSE)
confint(lowGPC_Buyse)
```

```
              estimate         se      lower.ci   upper.ci null    p.value
rcs_t1.45    0.01628547 0.01771680 -0.018444486 0.05097618    0 0.35807018
number_t0.35 0.03730092 0.02668638 -0.015057839 0.08945568    0 0.16257765
duration_t3  0.04215064 0.02400508 -0.004957166 0.08907178    0 0.07946061
```

```
lowGPC_CMH <- BuyseTest(update(fff, . ~ . + strata(id, match = TRUE)),
                        data = lowProfil, pool.strata = "CMH", trace = FALSE)
confint(lowGPC_CMH)
```

```
              estimate         se    lower.ci   upper.ci null   p.value
rcs_t1.45    0.02145018 0.01855984 -0.01493878 0.05778241    0 0.2479363
number_t0.35 0.03266743 0.02784903 -0.02195881 0.08709921    0 0.2411232
duration_t3  0.03869945 0.02491698 -0.01019050 0.08740482    0 0.1207617
```

⚠ in all approaches, all pairwise comparisons are performed within each patient, not only within-block comparisons.

In term of uncertainty quantification, it is important to specify `match=TRUE` when using a single call so `BuyseTest` does not treat each line of the dataset as an independent replicate. An intuitive way to evaluate the standard error of the pooled estimator is to use the across subject variability:

```
sqrt(apply(df.lowGPC[,1:3],2,var)/NROW(df.lowGPC))
```

```
 rcs_t1.45 number_t0.35  duration_t3
0.02075177   0.03182148   0.02839590
```

This is, up to a factor `N/(N-1)` exactly what the single call approach returns. Actually we can retrieve this value by modifying the default options:

```
BuyseTest.options(order.Hprojection = 2)
lowGPC2 <- BuyseTest(update(fff, . ~ . + strata(id, match = TRUE)),
                 data = lowProfil, trace = FALSE)
confint(lowGPC2)
```

```
              estimate         se    lower.ci   upper.ci null   p.value
rcs_t1.45     0.02741742 0.02075177 -0.01327825 0.06802241    0 0.1866527
number_t0.35  0.02755903 0.03182148 -0.03483625 0.08974030    0 0.3867027
duration_t3   0.03397497 0.02839590 -0.02172779 0.08946745    0 0.2318709
```

Similarly when using other weighting scheme. For instance we can retrieve the results of the Buyse pooling scheme doing:

```
df.lowGPC_center <- sweep(df.lowGPC[,1:3], MARGIN = 2, FUN = "-", STATS = coef(lowGPC_Buyse))
df.lowGPC_Wcenter <- sweep(df.lowGPC_center, MARGIN = 1, FUN = "*", STATS = df.lowGPC$Buyse)
sqrt(colSums(df.lowGPC_Wcenter^2))
```

```
rcs_t1.45 number_t0.35  duration_t3
0.01771680   0.02668638   0.02400508
```

4.2 With 3 or more treatment groups (WORK IN PROGRESS!)

Handling more than two treatment groups is still an area of development for the BuyseTest package. Several approaches have been proposed in the litterature and here we focus on one that aim at handling the non-transitivity issues that comes with Wilcoxon-like tests (Lumley and Gillen, 2016). This approach compare the treatment-specific distribution to a pooled distribution over all treatment groups (Thangavelu and Brunner, 2007):

```
CasinoTest(update(fff, . ~ . + strata(id)), data = profil,
            method.inference = "none", type = "unweighted")
## do not trust inference since it would require accounting for matching
```

	endpoint	treatment	estimate	se	lower.ci	upper.ci	null	p.value
placebo: rcs	rcs	placebo	-0.012193158	NA	NA	NA	NA	NA
lowDose: rcs	rcs	lowDose	0.008046335	NA	NA	NA	NA	NA
highDose: rcs	rcs	highDose	0.004146823	NA	NA	NA	NA	NA
placebo: number	number	placebo	-0.020434169	NA	NA	NA	NA	NA
lowDose: number	number	lowDose	0.011597293	NA	NA	NA	NA	NA
highDose: number	number	highDose	0.008836876	NA	NA	NA	NA	NA
placebo: duration	duration	placebo	-0.024148505	NA	NA	NA	NA	NA
lowDose: duration	duration	lowDose	0.013175906	NA	NA	NA	NA	NA
highDose: duration	duration	highDose	0.010972598	NA	NA	NA	NA	NA

Its main drawback is that the assessment of say `placebo` vs. `lowDose` is now influenced by `highDose`.

References

Lumley, T. and Gillen, D. L. (2016). Characterising transitive two-sample tests. *Statistics & Probability Letters*, 109:118–123.

Matsouaka, R. A. (2022). Robust statistical inference for matched win statistics. *Statistical Methods in Medical Research*, 31(8):1423–1438.

Thangavelu, K. and Brunner, E. (2007). Wilcoxon–mann–whitney test for stratified samples and efron's paradox dice. *Journal of Statistical Planning and Inference*, 137(3):720–737.