


# Performing GPC in a paired design

Brice Ozenne

May 13, 2025

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),
                     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)
```

```
pairs
0.06631828
```

which matches what BuyseTest output:

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

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

By default `confint` uses a hyperbolic tangent unless the argument `transform` is set to `FALSE`.

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:

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

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)

In fact the results of table 1 were obtained using MOVER approach. They can be retrieved with a dedicated function processing the output of the `BuyseTest` function:

```
runMover <- function(object, level = 0.95, p.value = TRUE, type = "Wilson", tol = 1e-6){

  ## ** normalize user input
  q.low <- qnorm((1-level)/2)
  q.high <- qnorm(1-(1-level)/2)
  type <- match.arg(type, c("Wilson", "Agresti-Coull"))

  ## ** normalize user input
  N <- nobs(object)["pairs"]
  pi_w <- coef(object, statistic = "favorable")
  pi_l <- coef(object, statistic = "unfavorable")

  if(level == 1){
    DeltaL <- -1
    DeltaU <- 1
  }else{
    Ntilde <- N + q.high
    pitilde_w <- (N*pi_w+0.5*q.high^2)/Ntilde
    pitilde_l <- (N*pi_l+0.5*q.high^2)/Ntilde

    if(any(c(pi_w, pi_l)<tol) || any(c(pi_w, pi_l)>1-tol)){
      cor_wl <- 0
    }else{
      cor_wl <- - pi_w * pi_l / sqrt(pi_w * (1 - pi_w) * pi_l * (1 - pi_l))
    }

    ## Wilson
    if(type == "Wilson"){
      L_w <- pitilde_w + 0.5/Ntilde * q.low * sqrt(q.low^2 + 4*N*pi_w*(1-pi_w))
      U_w <- pitilde_w + 0.5/Ntilde * q.high * sqrt(q.high^2 + 4*N*pi_w*(1-pi_w))
      L_l <- pitilde_l + 0.5/Ntilde * q.low * sqrt(q.low^2 + 4*N*pi_l*(1-pi_l))
      U_l <- pitilde_l + 0.5/Ntilde * q.high * sqrt(q.high^2 + 4*N*pi_l*(1-pi_l))
    }else if(type == "Agresti-Coull"){
      L_w <- pitilde_w + q.low * sqrt(pitilde_w*(1-pitilde_w)/Ntilde)
      U_w <- pitilde_w + q.high * sqrt(pitilde_w*(1-pitilde_w)/Ntilde)
      L_l <- pitilde_l + q.low * sqrt(pitilde_l*(1-pitilde_l)/Ntilde)
      U_l <- pitilde_l + q.high * sqrt(pitilde_l*(1-pitilde_l)/Ntilde)
    }

    DeltaL <- (pi_w-pi_l) - sqrt((pi_w - L_w)^2 + (U_l - pi_l)^2 - 2 * cor_wl * (pi_w - L_w) * (U_l - pi_l))
    DeltaU <- (pi_w-pi_l) + sqrt((U_w - pi_w)^2 + (pi_l - L_l)^2 - 2 * cor_wl * (U_w - pi_w) * (pi_l - L_l))
  }

  ## p-value
  if(p.value){
    if(abs(DeltaL)<tol | abs(DeltaU)<tol){ ## CI touch 0
      p <- 1-level
    }else if(DeltaL>0 && DeltaU>0){ ## CI do not overlap 0: p>level
      res.search <- uniroot(f = function(mylevel){runMover(object, level = mylevel, p.value = FALSE)["lower"]},
        lower = level, upper = 1)
    }else if(DeltaL<0 && DeltaU<0){ ## CI do not overlap 0: p>level
      res.search <- uniroot(f = function(mylevel){runMover(object, level = mylevel, p.value = FALSE)["upper"]},
        lower = level, upper = 1)
    }else if(DeltaL<0 && DeltaU>0){ ## CI overlaps 0: p<level
      res.search <- uniroot(f = function(mylevel){
        tempo <- runMover(object, level = mylevel, p.value = FALSE)[c("lower", "upper")]
        return(tempo[which.min(abs(tempo))])
      }, lower = 0, upper = level)
    }
    p.value <- 1-res.search$root
  }else{
    p.value <- NA
  }

  ## export
  out <- c(estimate = pi_w-pi_l, lower = unname(DeltaL), upper = unname(DeltaU), pvalue = p.value)
  return(out)
}
```

```
runMover(e.BTjuv, level = 0.95)
```

```
estimate      lower      upper      pvalue
0.15789474 0.02525513 0.28805176 0.02048651
```

### 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 treatment 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),
                     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	p.value
1	time	114	47.36525	24.29552	3	39.33923	0.202366	0.05045454	0.3451254	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),
                  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)
```

	pairs
	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)
```

- propagate 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
```

## References

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