Predictive Value of Left Ventricular Untwist on Intradialytic Blood Pressure and Outcome in Patients with mildly reduced and Preserved Ejection Fraction

Code & Data Supplement

Nidhal Bouchadha, Fabian Scheipl

2025-03-03

Setup

```
library(mgcv)
```

```
## Loading required package: nlme

## This is mgcv 1.9-1. For overview type 'help("mgcv-package")'.
```

```
library(MASS)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::collapse() masks nlme::collapse()
## x dplyr::filter()   masks stats::filter()
## x dplyr::lag()      masks stats::lag()
## x dplyr::select()   masks MASS::select()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(tidyfun)
```

```
## Loading required package: tf
##
## Attaching package: 'tf'
##
## The following objects are masked from 'package:stats':
##
##     sd, var
##
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```r
library(refund)
library(patchwork)
```

```
##
## Attaching package: 'patchwork'
##
## The following object is masked from 'package:MASS':
##
##     area
```

```r
library(ggplot2)
pacman::p_load(pacman, rio)
pacman::p_load(pacman, psych)
library(table1)
```

```
##
## Attaching package: 'table1'
##
## The following objects are masked from 'package:base':
##
##     units, units<-
```

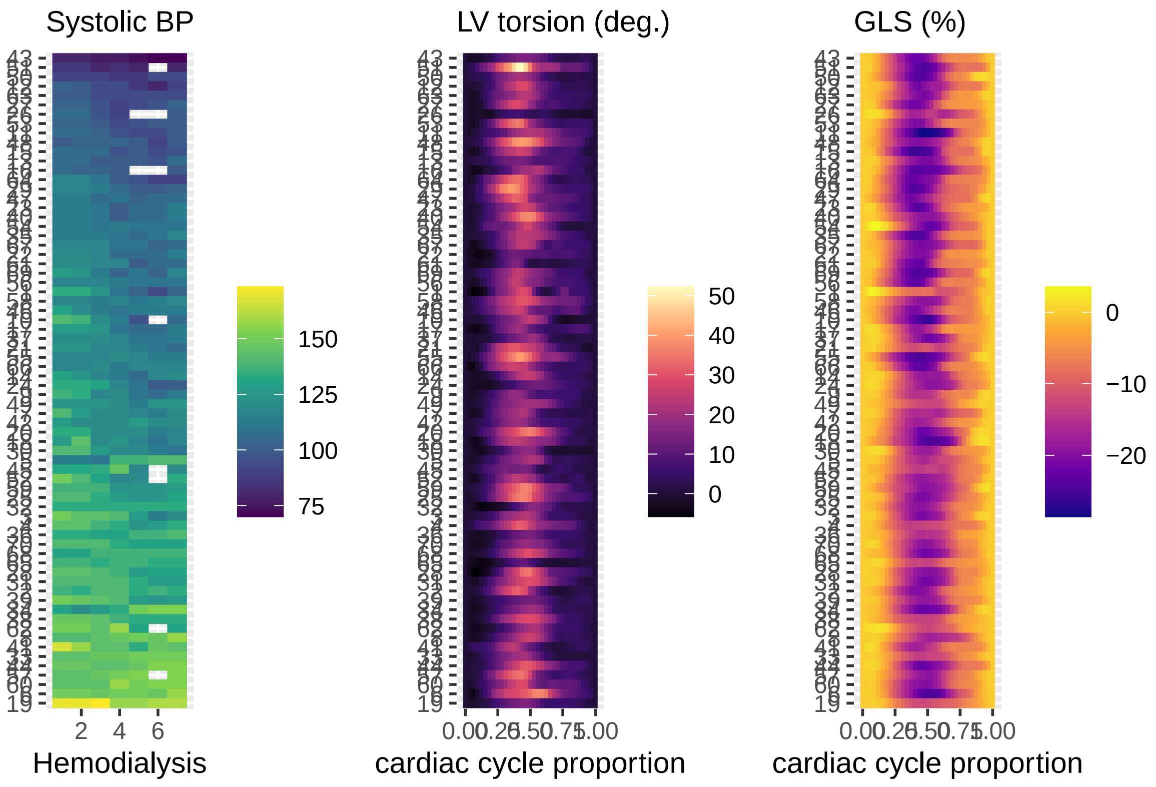
```r
library(mgcViz)
```

```
## Loading required package: qgam
## Registered S3 method overwritten by 'mgcViz':
##   method from
##   +.gg   GGally
##
## Attaching package: 'mgcViz'
##
## The following objects are masked from 'package:stats':
##
##     qqline, qqnorm, qqplot
```

```r
library(rgl)

data <- readRDS("data-anonymized.rds")
data_rep2 <- readRDS("data-rep-anonymized.rds")
```

Raw data

```r
p1 <- gglasagna(data, tf = pas_overall, order = tf_fmean(pas_overall)) +
  scale_fill_viridis_c("") +
  scale_color_viridis_c("") +
  labs(subtitle = "Systolic BP",
       caption = "patients sorted by mean SBP",
       x = "Hemodialysis")
p2 <- gglasagna(data, tf = LV_torsion, order = tf_fmean(pas_overall)) +
  scale_fill_viridis_c("", option = "A") +
  scale_color_viridis_c("", option = "A") +
  labs(subtitle = "LV torsion (deg.)",
       caption = "", x = "cardiac cycle proportion")
p3 <- gglasagna(data, tf = gls, order = tf_fmean(pas_overall)) +
  scale_fill_viridis_c("", option = "C") +
  scale_color_viridis_c("", option = "C") +
  labs(subtitle = "GLS (%)",  caption = "",
       x = "cardiac cycle proportion")
p1 + p2 + p3
```



ıts sorted by mean SBP

Mean Functions & Descriptive Statistics

```
tf_ind <- which(map(data, is_tf) |> unlist())
table1::table1(~ age + factor(SEXE) + factor(HTA) + factor(DBT) + dialaysishistory + weight + uf_mean +
                 tapse + masse + ee + LV_torsion_max + LV_torsion_max_time +
                 gls_max + gls_max_time + follow_up + factor(death),
               data[,-tf_ind],
               render.continuous = c(.="Mean (SD)", .="Median [Q1-Q3]")) |>
  table1::t1kable(longtable = TRUE)
```

	Overall
	(N=70)
age	
Mean (SD)	50.4 (14.9)
Median [Q1-Q3]	51.5 [41.3-61.8]
factor(SEXE)	
0	29 (41.4%)
1	41 (58.6%)
factor(HTA)	

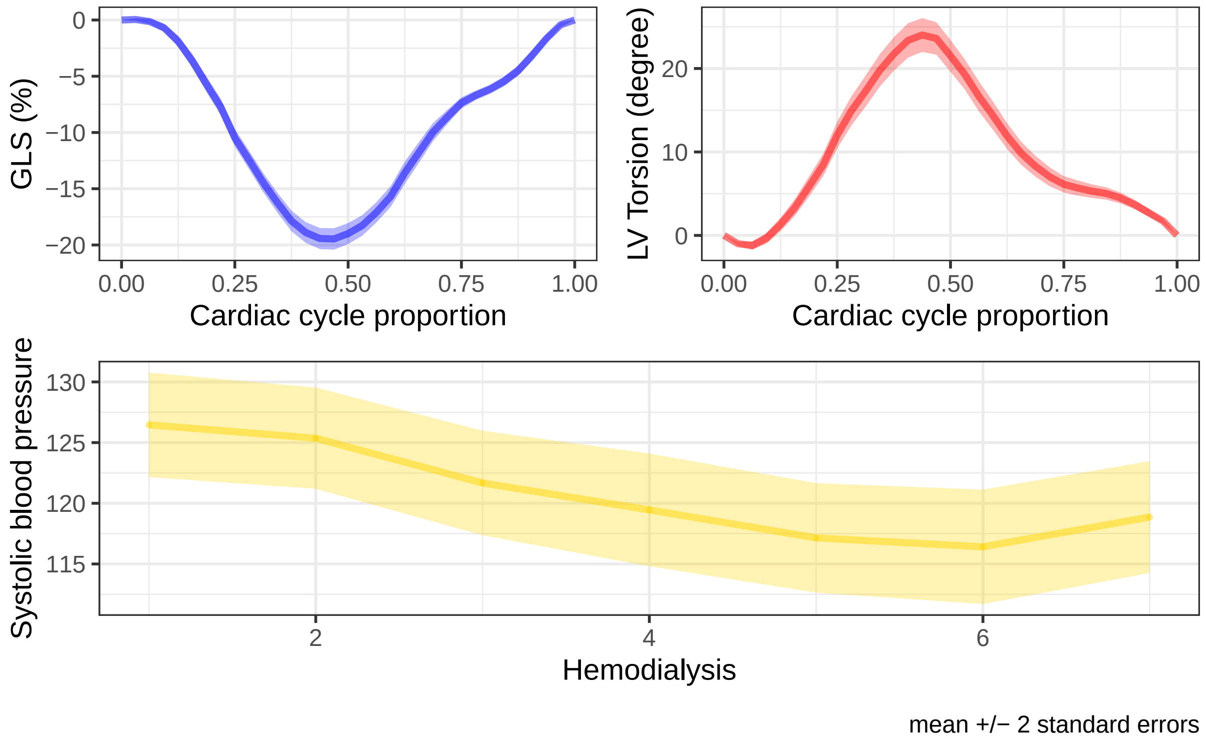
0	24 (34.3%)
1	46 (65.7%)
factor(DBT)	
0	63 (90.0%)
1	7 (10.0%)
dialaysishistory	
Mean (SD)	7.84 (5.90)
Median [Q1-Q3]	6.00 [4.00-10.8]
weight	
Mean (SD)	69.9 (13.2)
Median [Q1-Q3]	70.0 [61.0-79.0]
Missing	1 (1.4%)
uf_mean	
Mean (SD)	2350 (663)
Median [Q1-Q3]	2330 [1830-2910]
lvef	
Mean (SD)	59.5 (6.14)
Median [Q1-Q3]	59.0 [55.0-64.0]
ivc	
Mean (SD)	16.3 (3.15)
Median [Q1-Q3]	16.0 [15.0-18.0]
tapse	
Mean (SD)	23.9 (3.64)
Median [Q1-Q3]	23.9 [21.3-26.8]
masse	
Mean (SD)	111 (39.0)
Median [Q1-Q3]	108 [84.3-136]
ee	
Mean (SD)	10.6 (4.62)
Median [Q1-Q3]	9.52 [8.18-12.5]
Missing	1 (1.4%)
LV_torsion_max	
Mean (SD)	25.5 (8.40)
Median [Q1-Q3]	24.6 [19.8-30.6]
LV_torsion_max_time	
Mean (SD)	0.453 (0.0504)
Median [Q1-Q3]	0.438 [0.406-0.500]
gls_max	
Mean (SD)	-20.0 (3.83)
Median [Q1-Q3]	-20.7 [-22.5–17.9]
gls_max_time	
Mean (SD)	0.463 (0.0498)
Median [Q1-Q3]	0.469 [0.438-0.500]
follow_up	
Mean (SD)	14.5 (8.31)
Median [Q1-Q3]	9.23 [7.60-24.6]
factor(death)	
0	64 (91.4%)
1	6 (8.6%)

```r
data_mean <- data |>
  dplyr::select(id, LV_torsion, gls, rvfws, pas_overall) %>%
  dplyr::summarize(across(-id, c(mean = mean, sd = sd), na.rm = TRUE))
```

```
## Warning: There was 1 warning in `dplyr::summarize()`.
## i In argument: `across(-id, c(mean = mean, sd = sd), na.rm = TRUE)`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

```r
p1 <- ggplot(data_mean) +
  geom_spaghetti(aes(y = gls_mean), color = "blue", linewidth = 1.25) +
  geom_errorband(
    aes(
      ymax = gls_mean + 2 * gls_sd / sqrt(70),
      ymin = gls_mean - 2 * gls_sd / sqrt(70)
    ),
    fill = "blue"
  ) +
  theme_bw() +
  theme(legend.position = "none") +
  labs(x = "Cardiac cycle proportion", y = "GLS (%)")

p2 <- ggplot(data_mean) +
  geom_spaghetti(aes(y = LV_torsion_mean), color = "red", linewidth = 1.25) +
  geom_errorband(
    aes(
      ymax = LV_torsion_mean + 2 * LV_torsion_sd / sqrt(70),
      ymin = LV_torsion_mean - 2 * LV_torsion_sd / sqrt(70)
    ),
    fill = "red"
  ) +
  theme_bw() +
  theme(legend.position = "none") +
  labs(x = "Cardiac cycle proportion", y = "LV Torsion (degree)")

p3 <- ggplot(data_mean) +
  geom_meatballs(aes(y = pas_overall_mean), color = "gold", linewidth = 1.25) +
  geom_errorband(
    aes(
      ymax = pas_overall_mean + 2 * pas_overall_sd / sqrt(70),
      ymin = pas_overall_mean - 2 * pas_overall_sd / sqrt(70)
    ),
    fill = "gold"
  ) +
  theme_bw() +
  theme(legend.position = "none") +
  labs(x = "Hemodialysis", y = "Systolic blood pressure")
```

```
patchwork<-(p1|p2)/p3

patchwork+plot_annotation(
  title = "",
  subtitle = "",
  caption = "mean +/- 2 standard errors"
)
```



Blood Pressure Model

```
torsion_lr <- as.matrix(data_rep2$LV_torsion)
gls_lr <- as.matrix(data_rep2$gls)
rv_lr <- as.matrix(data_rep2$rvfws)
```

```
## Warning: i `interpolate = FALSE` & no values present for some `j`
## x `NA`s created.
```

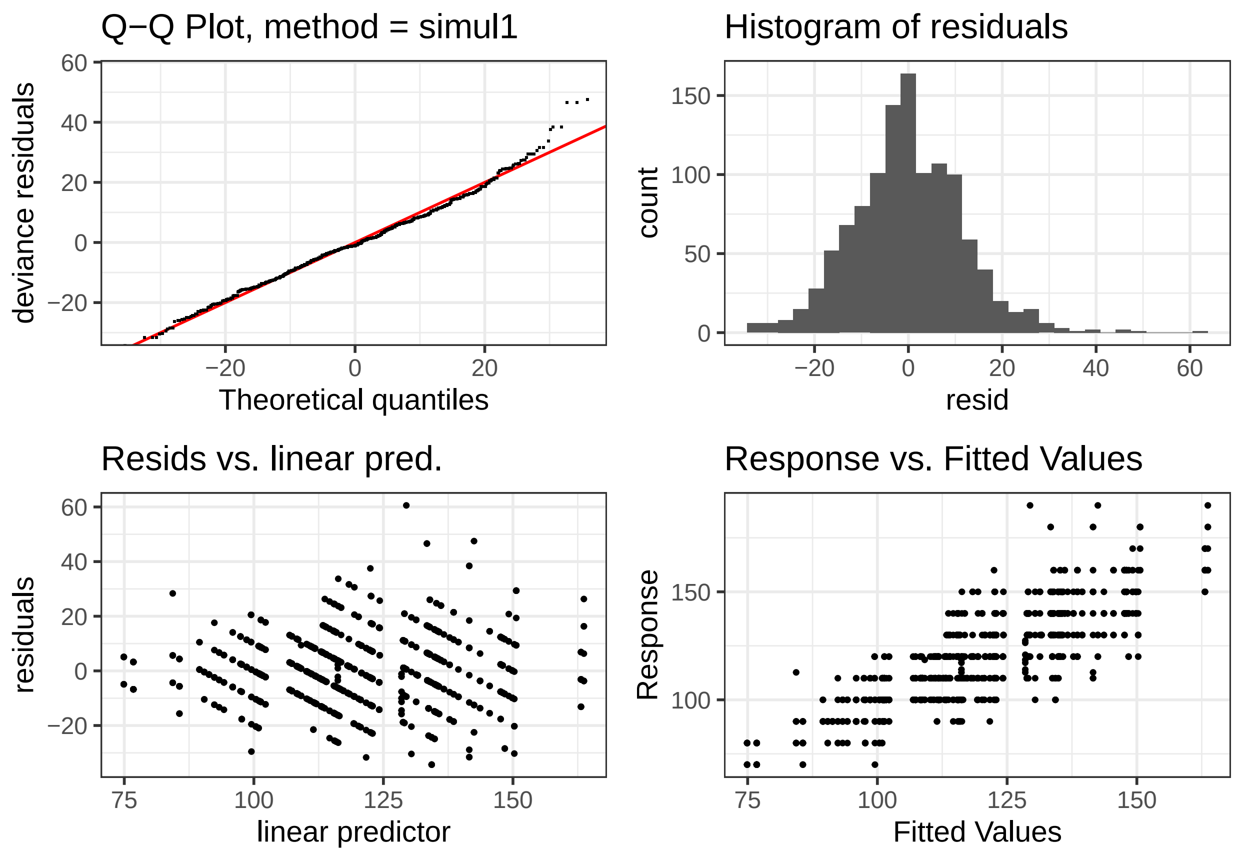
```
data_rep2_fit <- pfr(
  BP ~ uf +
      masse +
      SEXE +
      age +
      HTA +
      DBT +
      ivc +
```

```r
      dialaysishistory +
      lf(gls_lr) +
      lf(torsion_lr) +
      lf(rv_lr) +
      re(id),
  method = "REML",
  data = data_rep2
)
summary(data_rep2_fit)
```

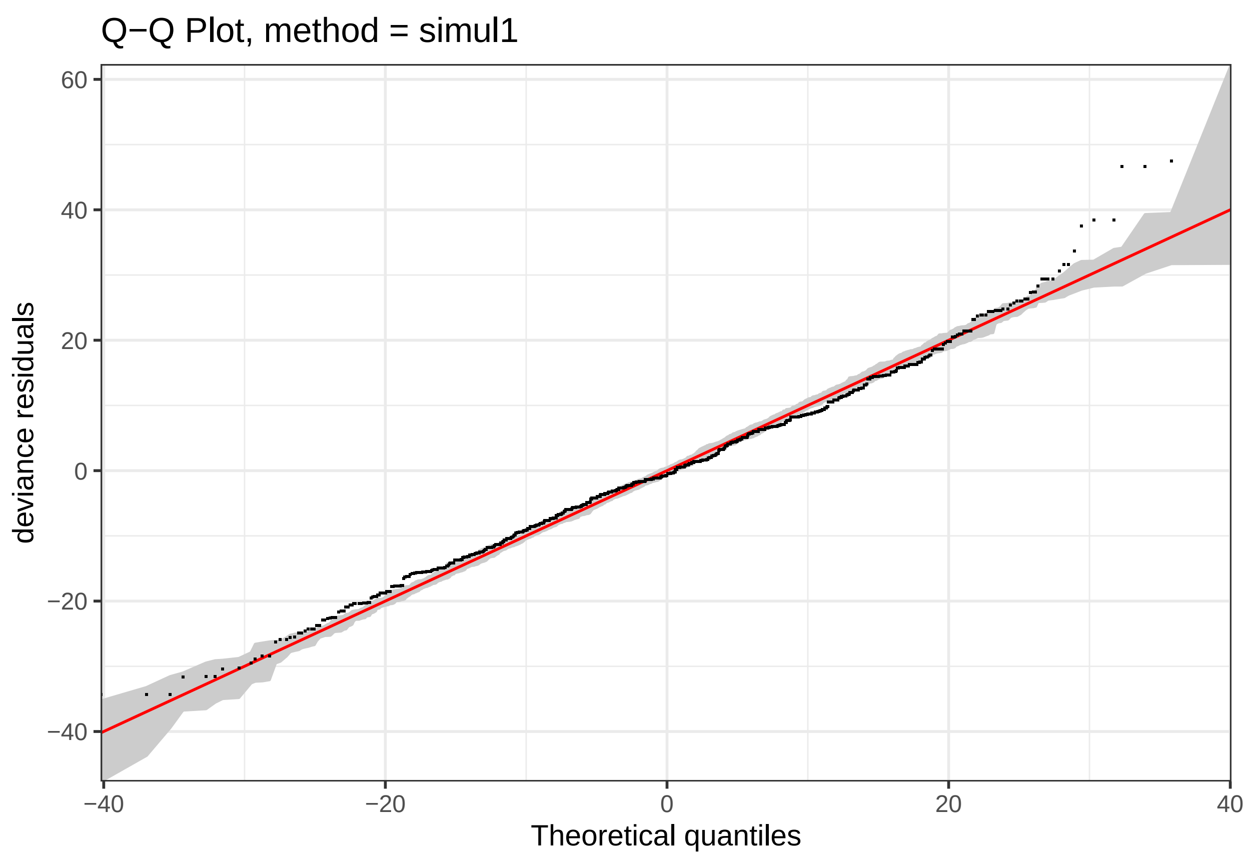
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## BP ~ uf + masse + SEXE + age + HTA + DBT + ivc + dialaysishistory +
##     s(x = gls_lr.tmat, by = L.gls_lr) + s(x = torsion_lr.tmat,
##     by = L.torsion_lr) + s(x = rv_lr.tmat, by = L.rv_lr) + s(x = id,
##     bs = "re")
##
## Parametric coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1.085e+02  2.272e+01   4.778 2.01e-06 ***
## uf               -1.847e-03  6.535e-04  -2.826  0.00479 **
## masse             1.209e-01  7.242e-02   1.670  0.09530 .
## SEXE              1.400e+00  4.708e+00   0.297  0.76622
## age              -3.033e-01  1.497e-01  -2.026  0.04296 *
## HTA               5.850e+00  4.374e+00   1.337  0.18135
## DBT               9.704e-01  8.687e+00   0.112  0.91108
## ivc               1.330e+00  6.352e-01   2.094  0.03653 *
## dialaysishistory -4.477e-01  4.199e-01  -1.066  0.28653
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                                  edf Ref.df      F p-value
## s(gls_lr.tmat):L.gls_lr          2.418  2.429  2.439  0.0984 .
## s(torsion_lr.tmat):L.torsion_lr  2.001  2.001  3.382  0.0343 *
## s(rv_lr.tmat):L.rv_lr            2.000  2.000  0.998  0.3690
## s(id)                           39.023 41.000 23.730  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.684   Deviance explained = 69.9%
## -REML = 4540.2  Scale est. = 144.26    n = 1143
```

```r
g <- getViz(data_rep2_fit)
check(g)
```

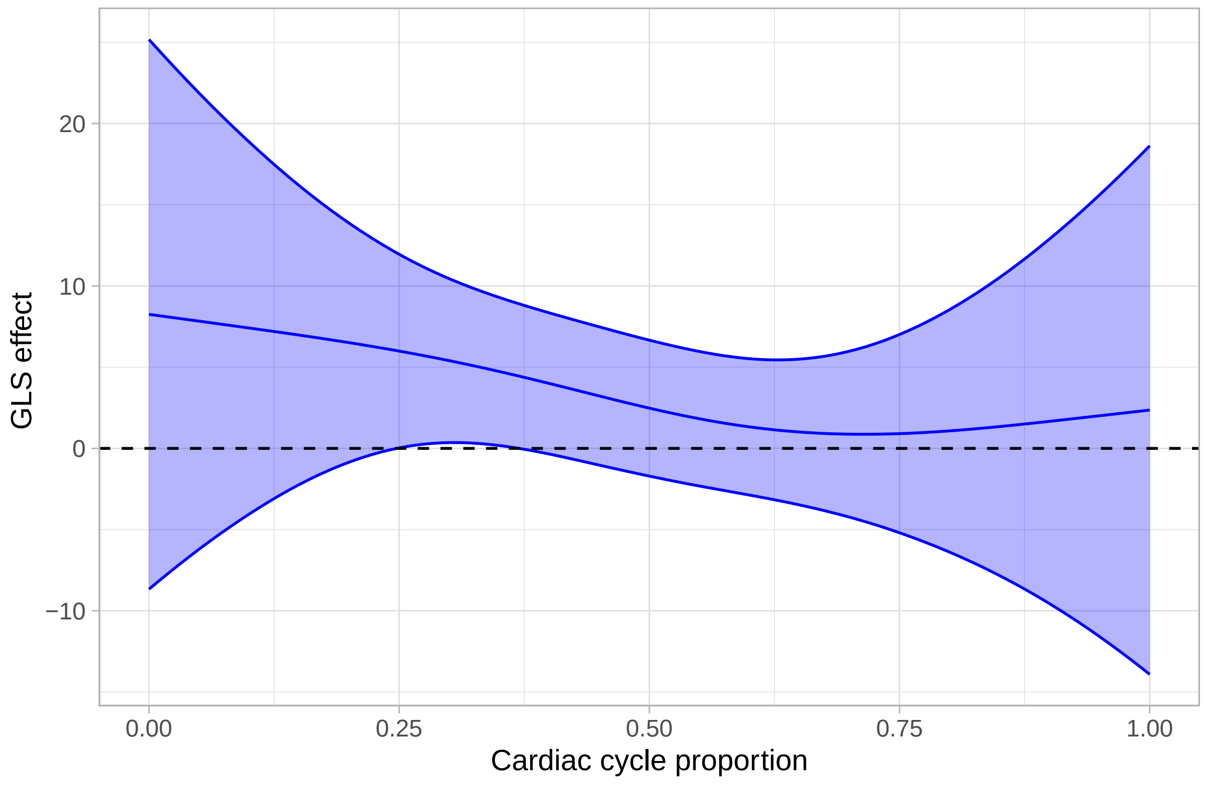
```
##
## Method: REML   Optimizer: outer newton
## full convergence after 9 iterations.
## Gradient range [-0.0002218889,0.0002453307]
## (score 4540.21 & scale 144.2627).
## Hessian positive definite, eigenvalue range [0.0001001315,564.6973].
## Model rank =  94 / 94
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                                    k'   edf k-index p-value
## s(gls_lr.tmat):L.gls_lr         10.00  2.42      NA      NA
## s(torsion_lr.tmat):L.torsion_lr 10.00  2.00      NA      NA
## s(rv_lr.tmat):L.rv_lr           10.00  2.00      NA      NA
## s(id)                           55.00 39.02      NA      NA

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
qq.gamViz(g, level = .9, CI = "quantile")
```

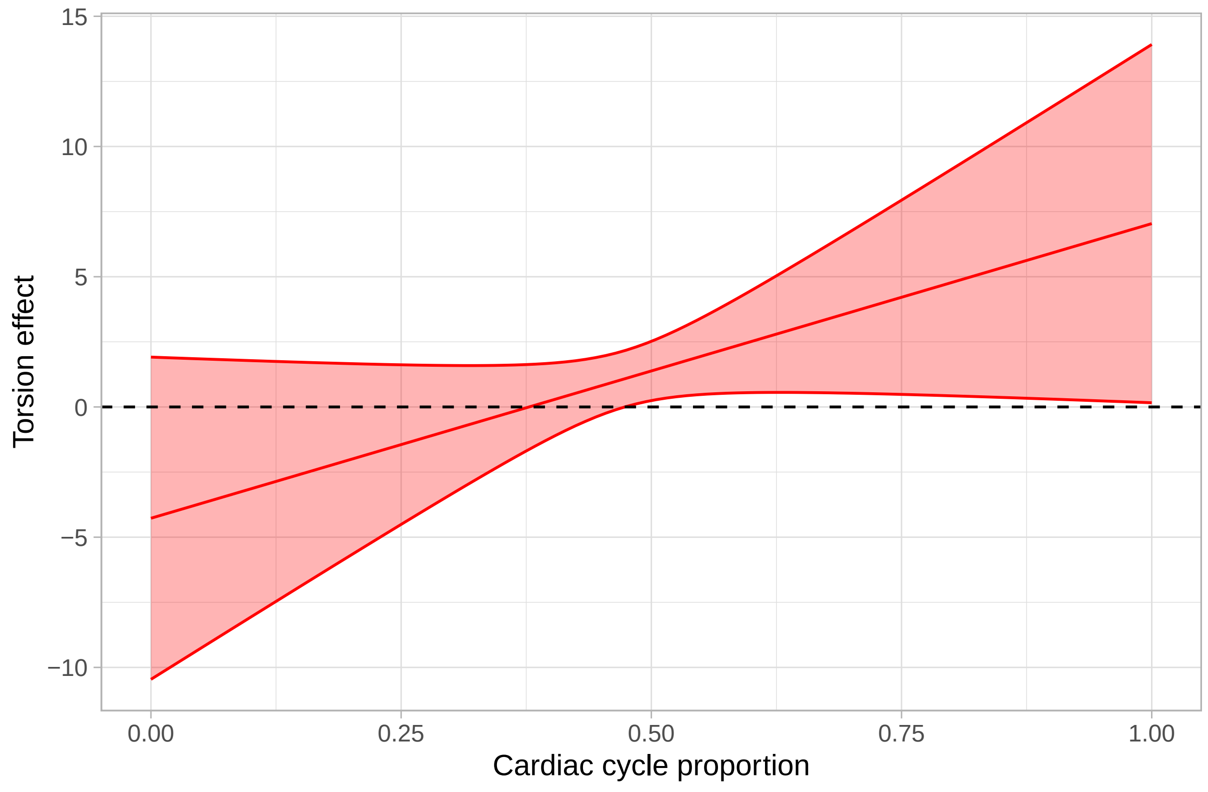


```
plot(sm(g, 1)) +
  l_fitLine(colour = "blue") +
  l_ciLine(colour = "blue", linetype = 1) +
  l_ciPoly(fill = "blue", alpha = 0.3) +
  theme_light() +
  geom_hline(yintercept = 0, linetype = 2) +
  labs(x = "Cardiac cycle proportion", y = "GLS effect ") +
  plot_annotation(caption = "pointwise 95% CIs")
```



pointwise 95% CIs

```r
plot(sm(g, 2)) +
  l_fitLine(colour = "red") +
  l_ciLine(colour = "red", linetype = 1) +
  l_ciPoly(fill = "red", alpha = 0.3) +
  theme_light() +
  geom_hline(yintercept = 0, linetype = 2) +
  labs(x = "Cardiac cycle proportion", y = "Torsion effect") +
  plot_annotation(caption = "pointwise 95% CIs")
```



pointwise 95% CIs

Bootstrap analysis (adapted from https://doi.org/10.1002/sim.10194)

```
NJ <- nrow(data_rep2)
uid <- unique(data_rep2$id)
nid <- nrow(data_rep2)

lX_sm_gls <- as.matrix(data_rep2$gls)
lX_sm_tor <- as.matrix(data_rep2$LV_torsion)
lX_sm_rv <- as.matrix(data_rep2$rvfws * -1)
```

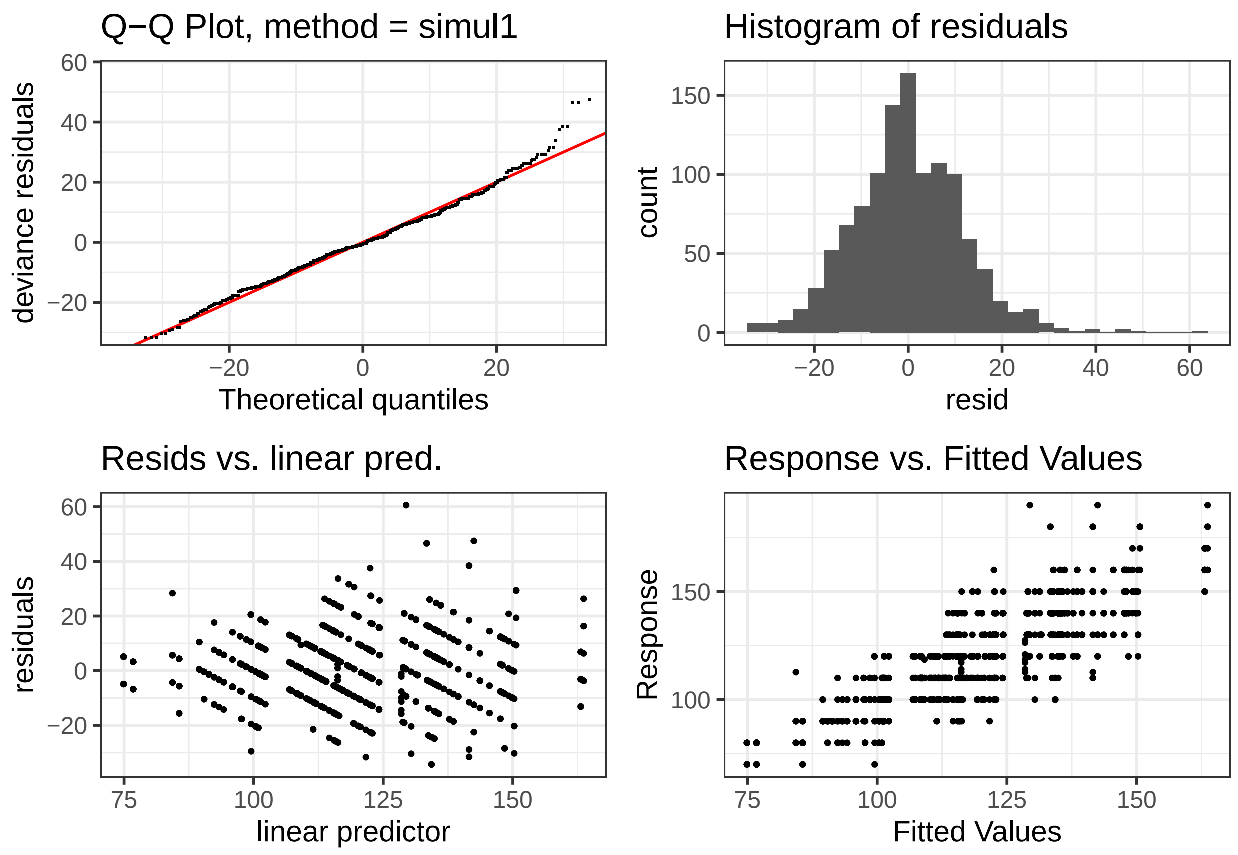
```
## Warning: i `interpolate = FALSE` & no values present for some `j`
## x `NA`s created.
```

```
#### create matrices for "cardiac cycle" and numeric integration via Riemann integration
sind <- seq(0, 1, len = 33)
smat <- matrix(1, nid, 1) %x% matrix(sind, 1, 33) # just a repetition of sind on all the rows
lmat <- matrix(1 / 33, nid, 33)

## merge PA data into the dataframe
data_rep2$smat <- I(smat) # used for fitting FGLM (functional domain matrix)

data_rep2$lX_lmat_gls <- I(lX_sm_gls * lmat)
data_rep2$lX_lmat_tor <- I(lX_sm_tor * lmat)
data_rep2$lX_lmat_rv <- I(lX_sm_rv * lmat) # used for fitting FGLM (numeric integration times functiona
rm(lmat, smat, nid, NJ, lX_sm_gls, lX_sm_tor, lX_sm_rv)
```

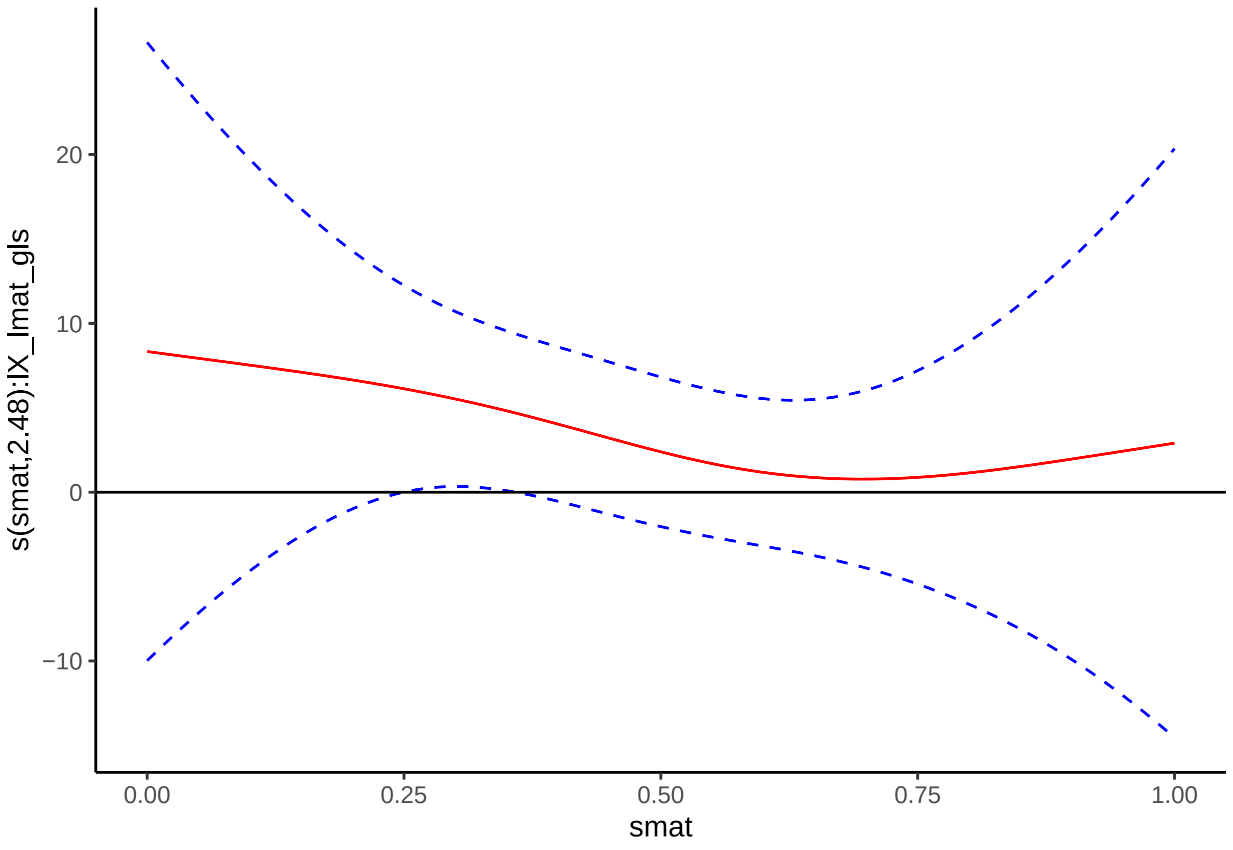
```r
## create dataframe used for extracting predicted coefficient
vars_covar <- c(
  "uf",
  "SEXE",
  "age",
  "HTA",
  "DBT",
  "ivc",
  "dialaysishistory",
  "id",
  "masse"
)

ns_pred <- 1000
sind_pred <- seq(0, 1, len = ns_pred)
df_pred <- data.frame(
  lX_lmat_gls = 1,
  lX_lmat_tor = 1,
  lX_lmat_rv = 1,
  smat = sind_pred,
  data_rep2[1, vars_covar]
)

fit_SOFR_uw <- gam(
  BP ~
    uf +
      masse +
      SEXE +
      age +
      HTA +
      DBT +
      ivc +
      dialaysishistory +
      s(smat, by = lX_lmat_gls) +
      s(smat, by = lX_lmat_tor) +
      s(smat, by = lX_lmat_rv) +
      s(id, bs = "re"),
  data = data_rep2,
  method = "REML"
)

## get the estimated coefficient plus intercept term: \beta_0 + \gamma(s)   for s \in [0,1]
est_uw <- predict(fit_SOFR_uw, newdata = df_pred, type = 'terms', se.fit = TRUE)


g <- getViz(fit_SOFR_uw)

check(g) #  is it OK?
```

```
##
## Method: REML   Optimizer: outer newton
## full convergence after 8 iterations.
## Gradient range [-0.0139819,0.0219834]
## (score 4540.158 & scale 144.2626).
## Hessian positive definite, eigenvalue range [0.0002285919,564.6718].
## Model rank =  94 / 94
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                         k'   edf k-index p-value
## s(smat):lX_lmat_gls 10.00  2.48      NA      NA
## s(smat):lX_lmat_tor 10.00  2.00      NA      NA
## s(smat):lX_lmat_rv  10.00  2.00      NA      NA
## s(id)               55.00 38.96      NA      NA
```

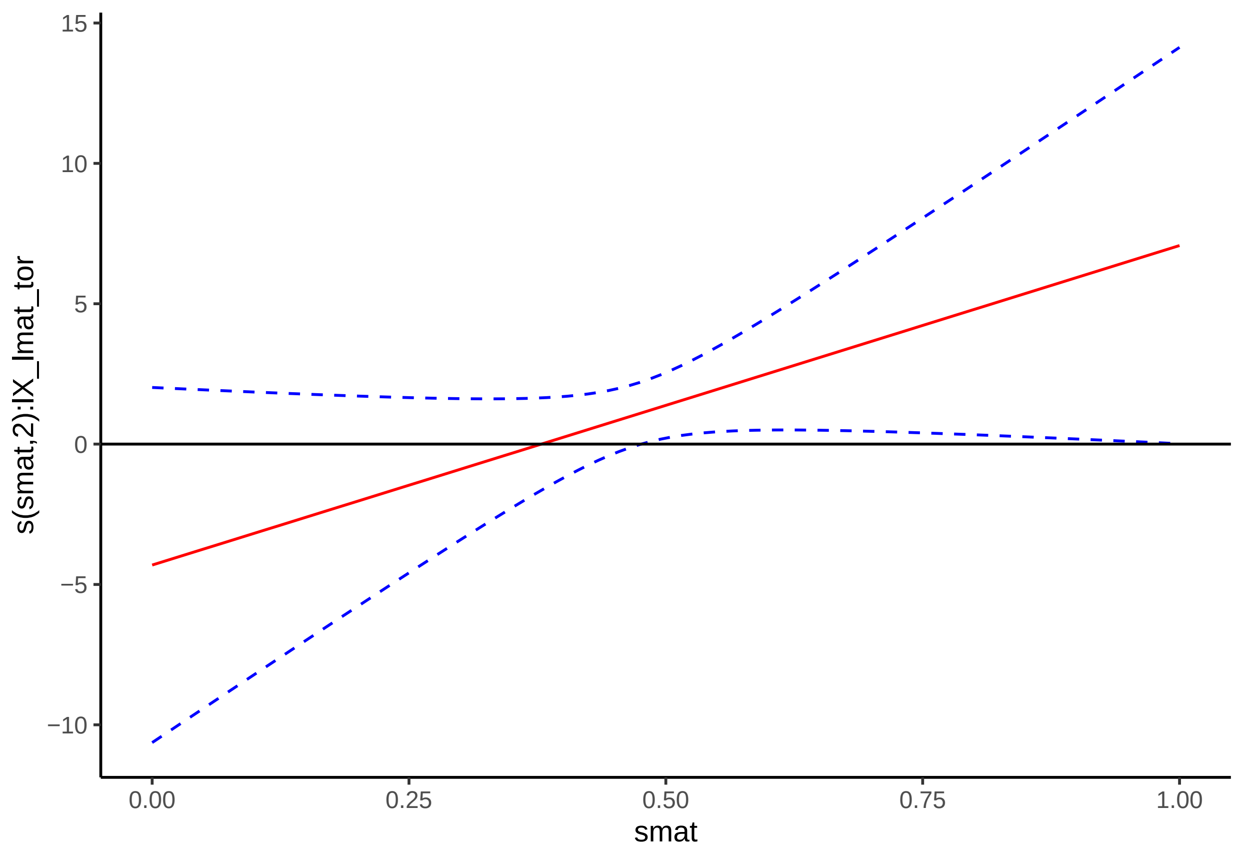
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



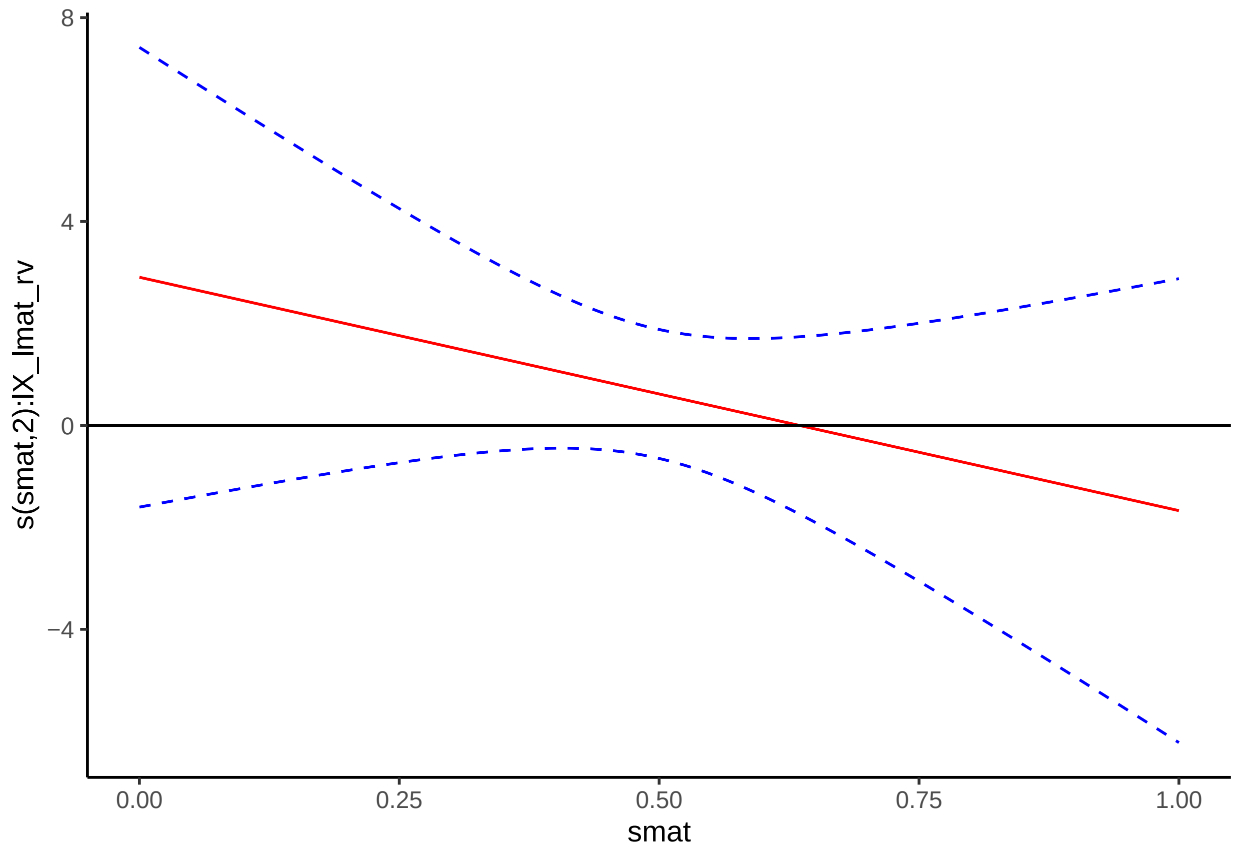
```
plot(sm(g, 1)) +
  l_fitLine(colour = "red") +
  l_ciLine(mul = 2, colour = "blue", linetype = 2) +
  theme_classic() +
  geom_hline(yintercept = 0)
```



```r
plot(sm(g, 2)) +
  l_fitLine(colour = "red") +
  l_ciLine(mul = 2, colour = "blue", linetype = 2) +
  theme_classic() +
  geom_hline(yintercept = 0)
```



```r
plot(sm(g, 3)) +
  l_fitLine(colour = "red") +
  l_ciLine(mul = 2, colour = "blue", linetype = 2) +
  theme_classic() +
  geom_hline(yintercept = 0)
```



```
#same model as PFR above
```

```
## get  bootstrap SEs
nboot <- 500
coef_mat_boot_gls <- matrix(NA, nboot, ns_pred)
coef_mat_boot_tor <- matrix(NA, nboot, ns_pred)
# pb_boot <- txtProgressBar(min=0, max=nboot,style=3)
inq_q_mat <- matrix(NA, nboot, dim(data_rep2)[1])

for(q in 1:nboot){

        inx_q <- sample(1:dim(data_rep2)[1], size=dim(data_rep2)[1], replace=TRUE)
        inq_q_mat[q,] <- inx_q

       data_rep2_q <- data_rep2[inx_q,]
        fit_gnq <- gam(fit_SOFR_uw$formula,method="REML", data=data_rep2_q)
        coef_mat_boot_gls[q,] <- predict(fit_gnq, newdata=df_pred, type="iterms")[,"s(smat):lX_lmat_gls
        # setTxtProgressBar(pb_boot, value=q)
         coef_mat_boot_tor[q,] <- predict(fit_gnq, newdata=df_pred, type="iterms")[,"s(smat):lX_lmat_to
  if (!(q%%10)) cat(q, "  ")
}
```

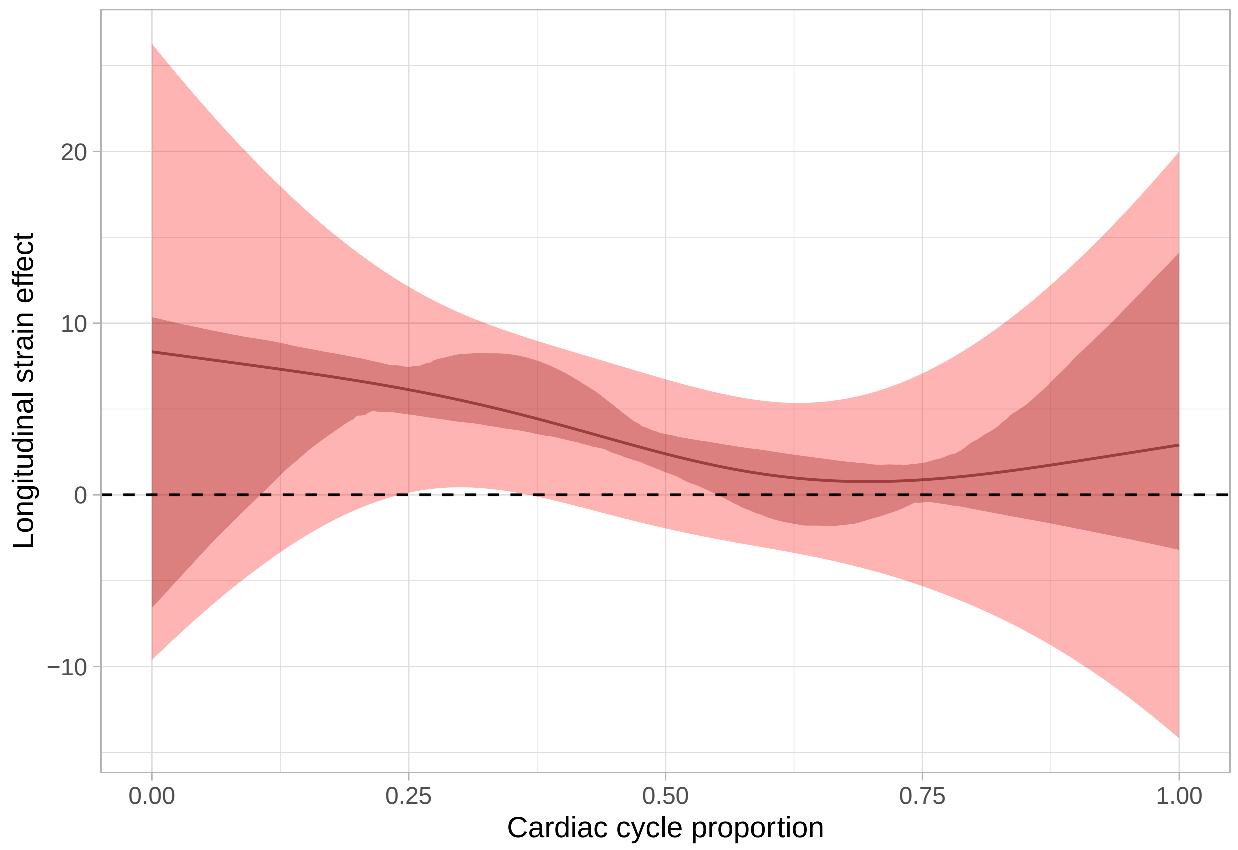
```
## 10   20   30   40   50   60   70   80   90   100   110   120   130   140   150   160   170   180   1
```

```
se_boot_gls_gamma <- apply(coef_mat_boot_gls, 2, sd, na.rm=TRUE)
se_boot_tor_gamma <- apply(coef_mat_boot_tor, 2, sd, na.rm=TRUE)
```

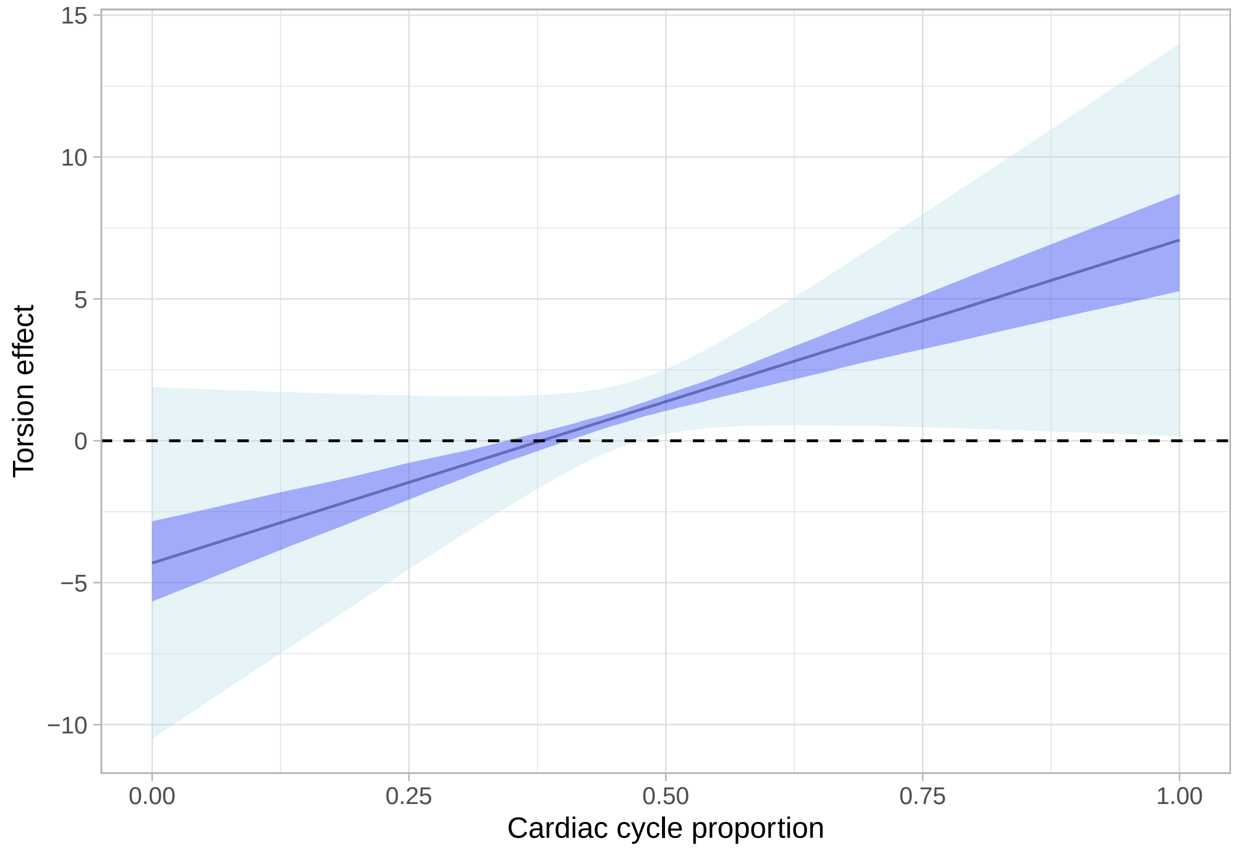
```r
# using quantiles to build CI
quantile_gls_up<-vector()
quantile_gls_lo<-vector()


for (i in 1:dim(coef_mat_boot_gls)[2]){
 quantile_gls_up[i]<-quantile(coef_mat_boot_gls[,i],probs = 0.975)
 quantile_gls_lo[i]<-quantile(coef_mat_boot_gls[,i],probs = 0.025)
}


quantile_tor_up<-vector()
quantile_tor_lo<-vector()

for (i in 1:dim(coef_mat_boot_tor)[2]){
 quantile_tor_up[i]<-quantile(coef_mat_boot_tor[,i],probs = 0.975)
 quantile_tor_lo[i]<-quantile(coef_mat_boot_tor[,i],probs = 0.025)
}
```

```r
# Non-parametric Bootstrap of the Max Absolute Statistic
# adapted from "Functional Data Analysis with R" (https://doi.org/10.1201/9781003278726)
#Find the max statistic
dvec_gls <- apply(
  coef_mat_boot_gls,
  1,
  function(x) max(abs(x - est_uw$fit[, 9]) / est_uw$se.fit[, 9])
)
dvec_tor <- apply(
  coef_mat_boot_tor,
  1,
  function(x) max(abs(x - est_uw$fit[, 10]) / est_uw$se.fit[, 10])
)

Z_global_gls <- quantile(dvec_gls, 0.95)
Z_global_tor <- quantile(dvec_tor, 0.95)

## point estimates of the functional coefficient and various SE estimates,
## combine into tibble using tidyfun

df_plot <- tibble(id = 1)

df_plot$gls_estimate <- tfd(est_uw$fit[, 9], seq(0, 1, length.out = 1000))
df_plot$torsion_estimate <- tfd(est_uw$fit[, 10], seq(0, 1, length.out = 1000))
df_plot$gls_se <- tfd(est_uw$se.fit[, 9], seq(0, 1, length.out = 1000))
df_plot$torsion_se <- tfd(est_uw$se.fit[, 10], seq(0, 1, length.out = 1000))
df_plot$gls_se_boot <- tfd(se_boot_gls_gamma, seq(0, 1, length.out = 1000))
df_plot$torsion_se_boot <- tfd(se_boot_tor_gamma, seq(0, 1, length.out = 1000))
df_plot$quantile_gls_up <- tfd(quantile_gls_up, seq(0, 1, length.out = 1000))
df_plot$quantile_gls_lo <- tfd(quantile_gls_lo, seq(0, 1, length.out = 1000))
df_plot$quantile_tor_up <- tfd(quantile_tor_up, seq(0, 1, length.out = 1000))
df_plot$quantile_tor_lo <- tfd(quantile_tor_lo, seq(0, 1, length.out = 1000))
```

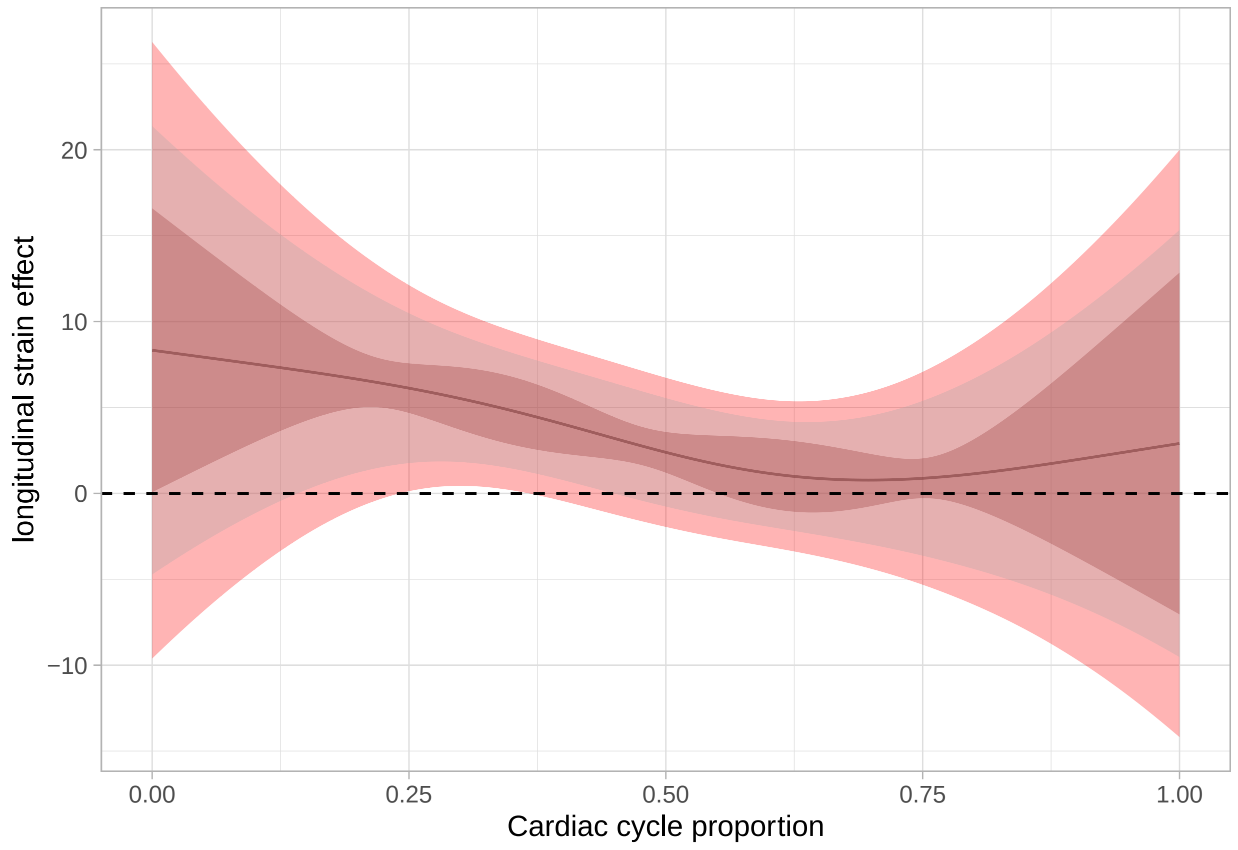
```r
# plot CI bands according to quantiles
ggplot(df_plot) +
  geom_spaghetti(aes(y = gls_estimate)) +
  geom_errorband(
    aes(
      ymax = gls_estimate + qnorm(0.975) * gls_se,
      ymin = gls_estimate - qnorm(0.975) * gls_se
    ),
    fill = "red"
  ) +
  geom_errorband(
    aes(ymax = quantile_gls_up, ymin = quantile_gls_lo),
    fill = "darkred"
  ) +
  geom_hline(yintercept = 0, linetype = 2) +
  xlab("Cardiac cycle proportion") +
  ylab("Longitudinal strain effect") +
  theme_light()
```



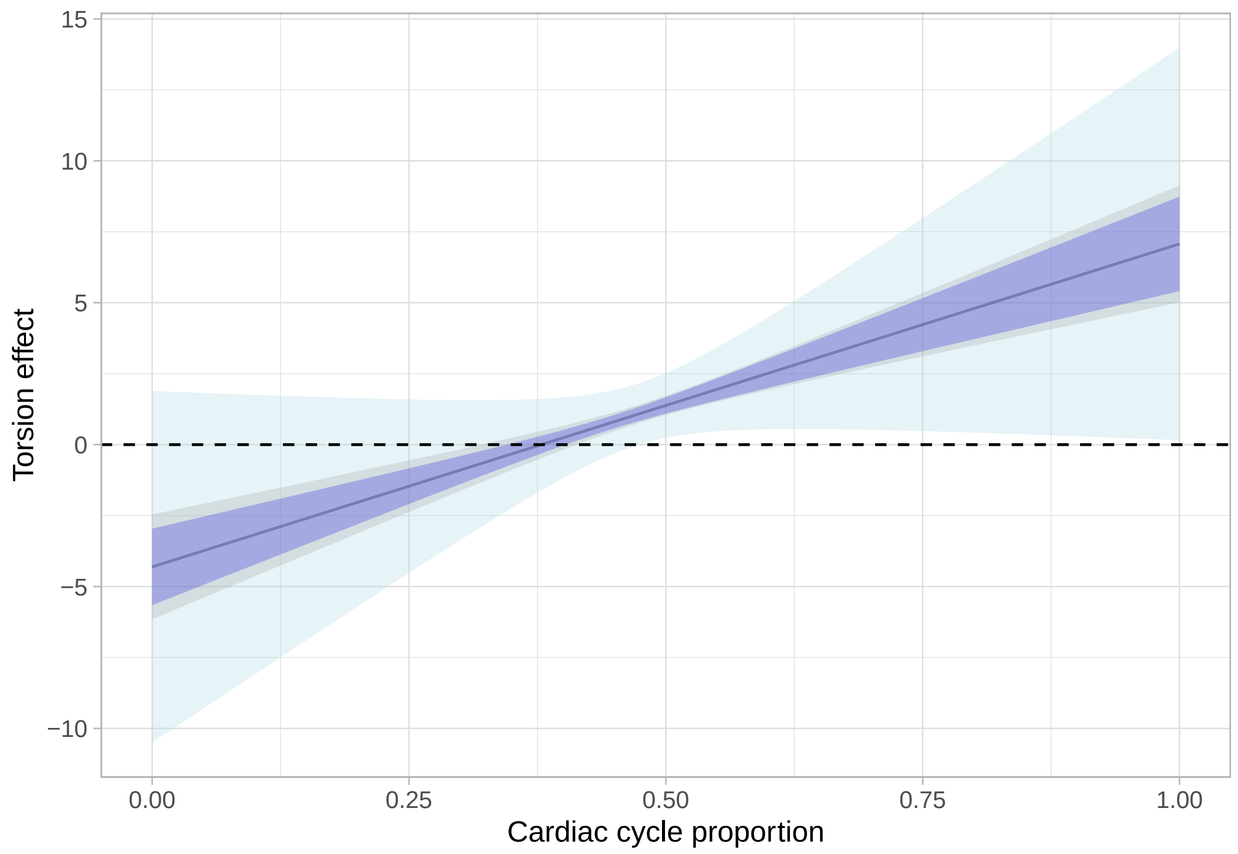
```
ggplot(df_plot) +
  geom_spaghetti(aes(y = torsion_estimate)) +
  geom_errorband(
    aes(
      ymax = torsion_estimate + qnorm(0.975) * torsion_se,
      ymin = torsion_estimate - qnorm(0.975) * torsion_se
    ),
    fill = "lightblue"
  ) +
  geom_errorband(
    aes(ymax = quantile_tor_up, ymin = quantile_tor_lo),
    fill = "blue"
  ) +

  geom_hline(yintercept = 0, linetype = 2) +
  xlab("Cardiac cycle proportion") +
  ylab("Torsion effect") +
  theme_light()
```



```
# with bootstrap CI other quantile method
ggplot(df_plot) +
  geom_spaghetti(aes(y = gls_estimate)) +
  geom_errorband(
    aes(
      ymax = gls_estimate + qnorm(0.975) * gls_se,
      ymin = gls_estimate - qnorm(0.975) * gls_se
    ),
    fill = "red"
  ) +
  geom_errorband(
    aes(
      ymax = gls_estimate + qnorm(0.975) * gls_se_boot,
      ymin = gls_estimate - qnorm(0.975) * gls_se_boot
    ),
    fill = "darkred"
  ) +
  geom_errorband(
    aes(
      ymax = gls_estimate + Z_global_gls * gls_se,
      ymin = gls_estimate - Z_global_gls * gls_se
    ),
    fill = "darkgrey"
  ) +
  geom_hline(yintercept = 0, linetype = 2) +
  xlab("Cardiac cycle proportion") +
  ylab("longitudinal strain effect") +
  theme_light()
```



```
ggplot(df_plot) +
  geom_spaghetti(aes(y = torsion_estimate)) +
  geom_errorband(
    aes(
      ymax = torsion_estimate + qnorm(0.975) * torsion_se,
      ymin = torsion_estimate - qnorm(0.975) * torsion_se
    ),
    fill = "lightblue"
  ) +

  geom_errorband(
    aes(
      ymax = torsion_estimate + qnorm(0.975) * torsion_se_boot,
      ymin = torsion_estimate - qnorm(0.975) * torsion_se_boot,
    ),
    fill = "blue"
  ) +
  geom_errorband(
    aes(
      ymax = torsion_estimate + Z_global_tor * torsion_se,
      ymin = torsion_estimate - Z_global_tor * torsion_se
    ),
    fill = "darkgrey"
  ) +
  geom_hline(yintercept = 0, linetype = 2) +
  xlab("Cardiac cycle proportion") +
  ylab("Torsion effect") +
  theme_light()
```



Mortality Model

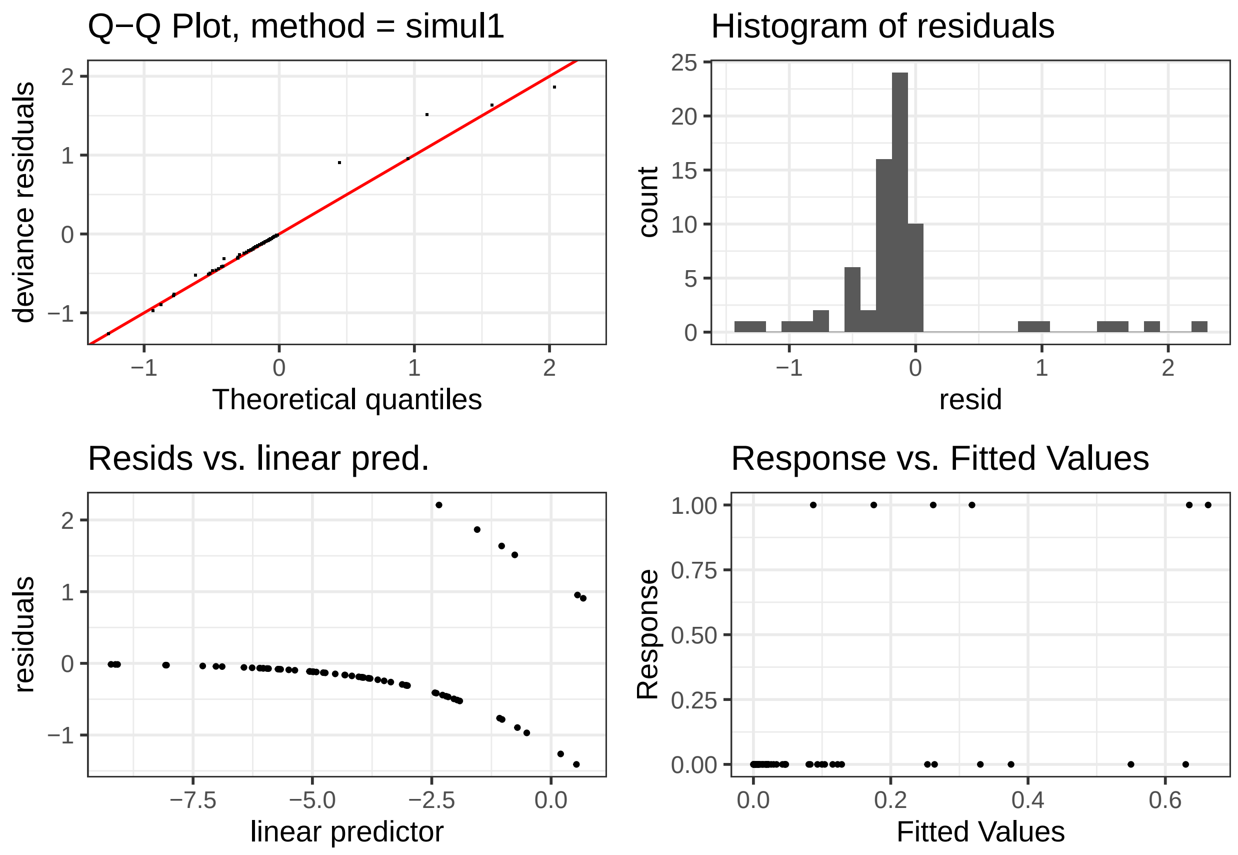
```
# predicting death

gls_mat<-as.matrix(data$gls)
torsion_mat<-as.matrix(data$LV_torsion)
death_fit<-pfr(death~lf(gls_mat)+lf(torsion_mat),method="REML",family=binomial(), data=data)
summary(death_fit)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## death ~ s(x = gls_mat.tmat, by = L.gls_mat) + s(x = torsion_mat.tmat,
##     by = L.torsion_mat)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -13.1        4.7  -2.787  0.00532 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                                 edf Ref.df Chi.sq p-value
## s(gls_mat.tmat):L.gls_mat         2.47  2.777   7.83  0.0341 *
## s(torsion_mat.tmat):L.torsion_mat 2.00  2.000   4.52  0.1044
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.228   Deviance explained = 39.9%
## -REML = 13.356  Scale est. = 1         n = 70
```

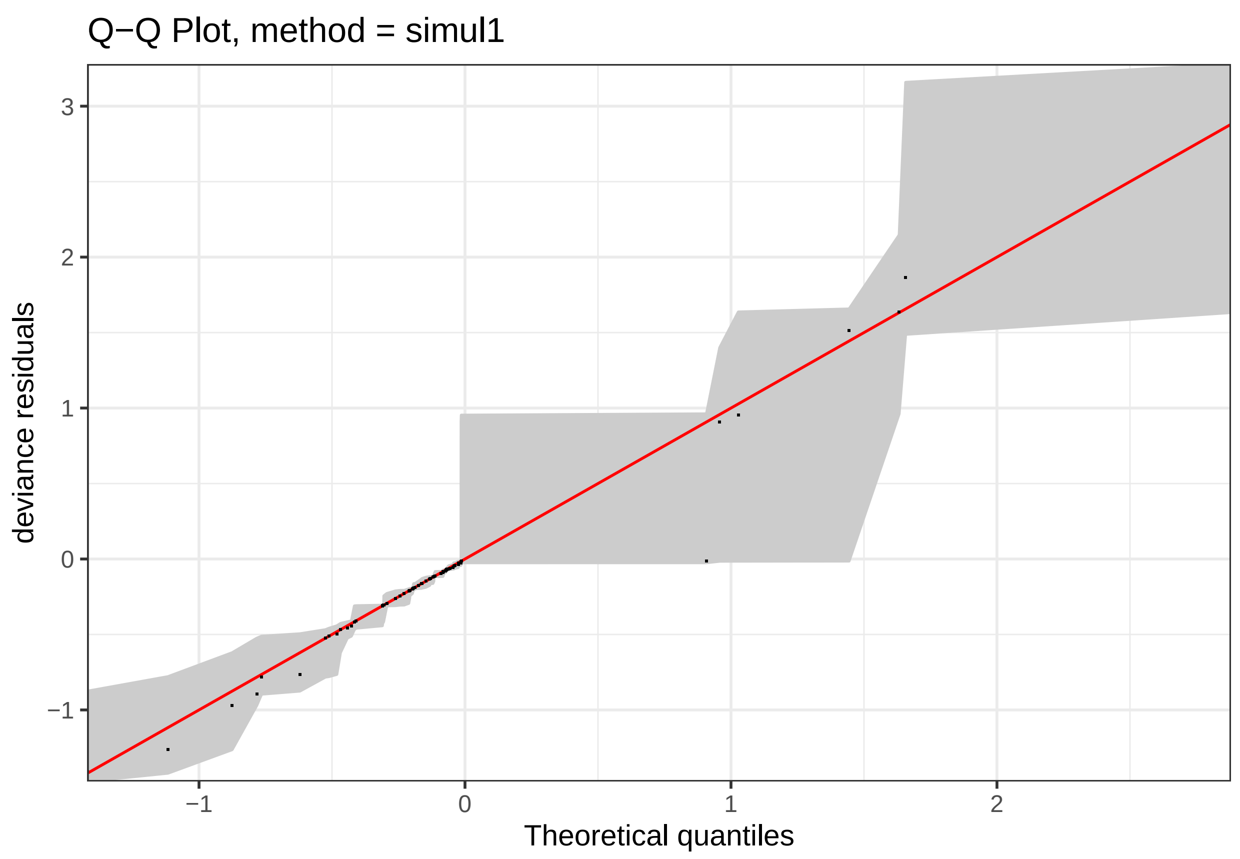
```r
death_plot<-getViz(death_fit)
check(death_plot)
```

```
##
## Method: REML   Optimizer: outer newton
## full convergence after 9 iterations.
## Gradient range [-3.442994e-06,-1.498625e-08]
## (score 13.35583 & scale 1).
## Hessian positive definite, eigenvalue range [3.442972e-06,0.02827983].
## Model rank =  21 / 21
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                                      k'   edf k-index p-value
## s(gls_mat.tmat):L.gls_mat         10.00  2.47      NA      NA
## s(torsion_mat.tmat):L.torsion_mat 10.00  2.00      NA      NA
```

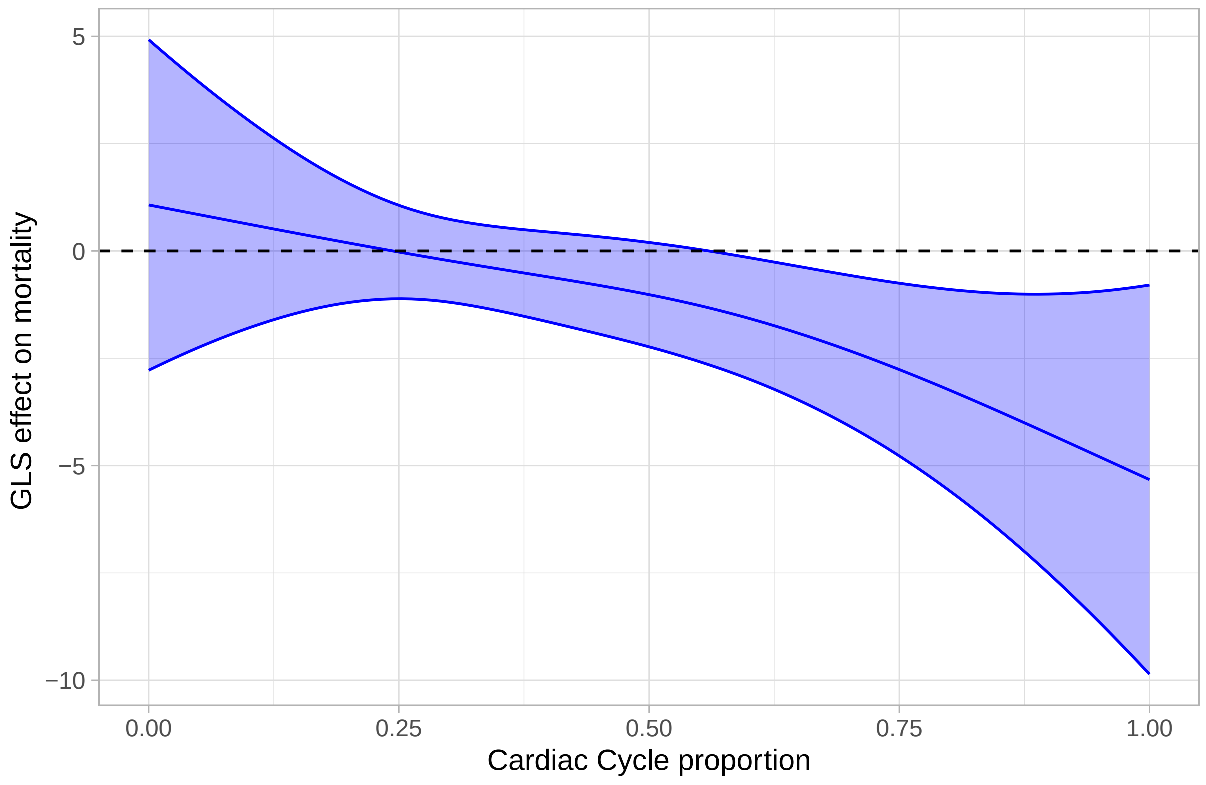
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```r
qq.gamViz(death_plot, level = .9,CI = "quantile")
```

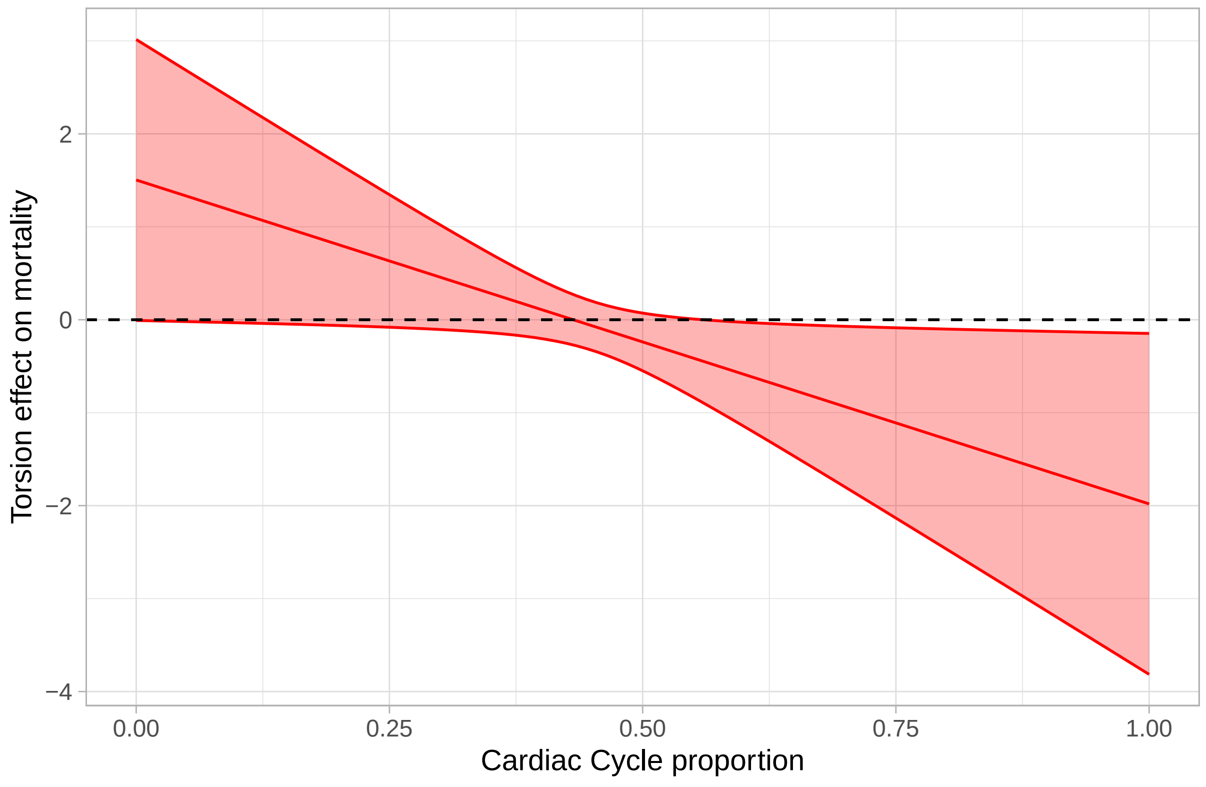


```
plot(sm(death_plot,1))+l_fitLine(colour = "blue") +
    l_ciLine( colour = "blue", linetype = 1) +l_ciPoly(fill="blue",alpha=0.3)+ theme_light()+geom_hline
```



pointwise 95% CIs

```
plot(sm(death_plot,2))+l_fitLine(colour = "red") +
    l_ciLine( colour = "red", linetype = 1)+l_ciPoly(fill="red",alpha=0.3) + theme_light()+geom_hline(yi
```



pointwise 95% CIs

System Info

```
sessionInfo()
```

```
## R version 4.4.2 (2024-10-31)
## Platform: x86_64-pc-linux-gnu
## Running under: Linux Mint 20
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/liblapack.so.3;  LAPACK version 3.9.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=de_DE.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=de_DE.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C
##
## time zone: Europe/Berlin
## tzcode source: system (glibc)
##
## attached base packages:
```

```
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] rgl_1.3.1        mgcViz_0.1.11   qgam_1.3.4      table1_1.4.3
##  [5] psych_2.4.6.26  rio_1.2.3       pacman_0.5.1    patchwork_1.3.0
##  [9] refund_0.1-38   tidyfun_0.0.98  tf_0.3.5        lubridate_1.9.4
## [13] forcats_1.0.0   stringr_1.5.1   dplyr_1.1.4     purrr_1.0.4
## [17] readr_2.1.5     tidyr_1.3.1     tibble_3.2.1    ggplot2_3.5.1
## [21] tidyverse_2.0.0 MASS_7.3-64     mgcv_1.9-1      nlme_3.1-167
##
## loaded via a namespace (and not attached):
##   [1] Rdpack_2.6.2       mnormt_2.1.1       bitops_1.0-9
##   [4] gridExtra_2.3      rlang_1.1.5        magrittr_2.0.3
##   [7] matrixStats_1.4.1  compiler_4.4.2     systemfonts_1.2.1
##  [10] vctrs_0.6.5        pkgconfig_2.0.3    fastmap_1.2.0
##  [13] backports_1.5.0    labeling_0.4.3     magic_1.6-1
##  [16] promises_1.3.0     deSolve_1.40       rmarkdown_2.29
##  [19] tzdb_0.4.0         pracma_2.4.4       nloptr_2.1.1
##  [22] tinytex_0.54       xfun_0.50          jsonlite_1.8.9
##  [25] later_1.3.2        parallel_4.4.2     cluster_2.1.8
##  [28] R6_2.6.0           stringi_1.8.4      RColorBrewer_1.1-3
##  [31] GGally_2.2.1       extrafontdb_1.0    boot_1.3-31
##  [34] rainbow_3.8        Rcpp_1.0.14        iterators_1.0.14
##  [37] knitr_1.49         zoo_1.8-12         base64enc_0.1-3
##  [40] hdrcde_3.4         extrafont_0.19     httpuv_1.6.15
##  [43] Matrix_1.7-1       splines_4.4.2      timechange_0.3.0
##  [46] tidyselect_1.2.1   viridis_0.6.5      rstudioapi_0.17.1
##  [49] abind_1.4-8        yaml_2.3.10        miniUI_0.1.1.1
##  [52] doParallel_1.0.17  codetools_0.2-19   RLRsim_3.1-8
##  [55] lattice_0.22-5     plyr_1.8.9         shiny_1.9.1
##  [58] ks_1.14.3          withr_3.0.2        pbs_1.1
##  [61] evaluate_1.0.3     ggstats_0.7.0      grpreg_3.5.0
##  [64] xml2_1.3.6         mclust_6.1.1       pillar_1.10.1
##  [67] fda_6.2.0          KernSmooth_2.23-26 checkmate_2.3.2
##  [70] foreach_1.5.2      reformulas_0.4.0   pcaPP_2.0-5
##  [73] generics_0.1.3     RCurl_1.98-1.16    hms_1.1.3
##  [76] munsell_0.5.1      scales_1.3.0       minqa_1.2.8
##  [79] xtable_1.8-4       gamm4_0.2-6        glue_1.8.0
##  [82] tools_4.4.2        fds_1.8            lme4_1.1-36
##  [85] mvtnorm_1.3-3      grid_4.4.2         Rttf2pt1_1.3.12
##  [88] rbibutils_2.3      colorspace_2.1-1   Formula_1.2-5
##  [91] cli_3.6.4          kableExtra_1.4.0   viridisLite_0.4.2
##  [94] svglite_2.1.3      gtable_0.3.6       digest_0.6.37
##  [97] htmlwidgets_1.6.4  farver_2.1.2       htmltools_0.5.8.1
## [100] lifecycle_1.0.4    mime_0.12
```