Using generalized additive models to analyze biomedical non-linear longitudinal data

Beyond repeated measures ANOVA and Linear Mixed Models

SUPPLEMENTARY MATERIALS: APPENDIX

Ariel I. Mundo

Department of Biomedical Engineering, University of Arkansas, Fayetteville, AR, USA

John R. Tipton

Department of Mathematical Sciences, University of Arkansas, Fayetteville, AR, USA

Timothy J. Muldoon*

Department of Biomedical Engineering, University of Arkansas, Fayetteville, AR, USA

tmuldoon@uark.edu

A APPENDIX

This section presents the code used to generate figures, models and simulated data from the main manuscript.

A.1 Compound symmetry and independent errors in linear and quadratic responses

This section simulates linear and quadratic data in the same manner as in Section 3.5 in the main manuscript. The linear simulations using Figure A.1 show in panels A and D the simulated mean responses and individual data points. Panels C and G show a visual interpretation of "correlation" in the responses: In panel C, subjects that have a value of the random error ε either above or below the mean group response are more likely to have other observations that follow the same trajectory, thereby demonstrating correlation in the response. In panel G,because the errors are independent, there is no expectation that responses are likely to follow a similar pattern. Panels D and H show the predictions from the rm-ANOVA model.

The following code produces a more comprehensive exploration of Figure 1 in the main manuscript.

```r
##########Section for calculations###########


## Example with linear response

#This function simulates data using a linear or quadratic mean response
   and each with correlated
#or uncorrelated errors. Each group has a different slope/concavity.
example <- function(n_time = 6, #number of time points
                    fun_type = "linear", #type of response
                    error_type = "correlated") {

  if (!(fun_type %in% c("linear", "quadratic")))
    stop('fun_type must be either "linear", or "quadratic"')
  if (!(error_type %in% c("correlated", "independent")))
    stop('fun_type must be either "correlated", or "independent"')


  x <- seq(1,6, length.out = n_time)

  #Create mean response matrix: linear or quadratic
  mu <- matrix(0, length(x), 2)
  # linear response
  if (fun_type == "linear") {
    mu[, 1] <- - (0.25*x)+2
    mu[, 2] <- 0.25*x+2
  } else {
    # quadratic response (non-linear)

    mu[, 1] <-  -(0.25 * x^2) +1.5*x-1.25
    mu[, 2] <- (0.25 * x^2) -1.5*x+1.25
  }

  #create an array where individual observations per each time point for
     each group are to be stored. Currently using 10 observations per
     timepoint
  y <- array(0, dim = c(length(x), 2, 10))
  #Create array to store the "errors" for each group at each timepoint.
     The "errors" are the
  #between-group variability in the response.
  errors <- array(0, dim = c(length(x), 2, 10))
  #create an array where 10 observations per each time point for each
     group are to be stored

  #The following cycles create independent or correlated responses. To
     each value of mu (mean response per group) a randomly generated error
      (correlated or uncorrelated) is added and thus the individual
     response is created.
  if (error_type == "independent") {
    ## independent errors
    for (i in 1:2) {
      for (j in 1:10) {
        errors[, i, j] <- rnorm(6, 0, 0.25)
        y[, i, j] <- mu[, i] + errors[, i, j]
      }
    }
  } else {
    for (i in 1:2) {     # number of treatments
      for (j in 1:10) {  # number of subjects
        # compound symmetry errors: variance covariance matrix
        errors[, i, j] <- rmvn(1, rep(0, length(x)), 0.1 * diag(6) + 0.25
           * matrix(1, 6, 6))
        y[, i, j] <- mu[, i] + errors[, i, j]
      }
    }
  }


  ## subject random effects

  ## visualizing the difference between independent errors and compound
     symmetry
  ## why do we need to account for this -- overly confident inference

#labeling y and errors
  dimnames(y) <- list(time = x,
                      treatment = 1:2,
                      subject = 1:10)

  dimnames(errors) <- list(time = x,
                           treatment = 1:2,
                           subject = 1:10)

  #labeling the mean response
  dimnames(mu) <- list(time = x,
                       treatment = 1:2)

  #convert y, mu and errors to  dataframes with time, treatment and
     subject columns
  dat <- as.data.frame.table(y,
                             responseName = "y")
  dat_errors <- as.data.frame.table(errors,
                                    responseName = "errors")
  dat_mu <- as.data.frame.table(mu,
                                responseName = "mu")

  #join the dataframes to show mean response and errors per subject
  dat <- left_join(dat, dat_errors,
                   by = c("time", "treatment", "subject"))
  dat <- left_join(dat, dat_mu,
                   by = c("time", "treatment"))
  #add time
  dat$time <- as.numeric(as.character(dat$time))
  #label subjects per group
  dat <- dat %>%
    mutate(subject = factor(paste(subject,
                                  treatment,
                                  sep = "-")))


  ## repeated measures ANOVA

  fit_anova <- lm(y ~ time + treatment + time * treatment, data = dat)

#LMEM: time and treatment interaction model, compound symmetry
  fit_lme <- lme(y ~ treatment + time + treatment:time,
                 data = dat,
                 random = ~ 1 | subject,
                 correlation = corCompSymm(form = ~ 1 | subject)
  )

  #create a prediction frame where the model can be used for plotting
     purposes
  pred_dat <- expand.grid(
    treatment = factor(1:2),
    time = unique(dat$time)
  )

  #add model predictions to the dataframe that has the simulated data
  dat$pred_anova <- predict(fit_anova)
  dat$pred_lmem <- predict(fit_lme)

  #return everything in a list
  return(list(
    dat = dat,
    pred_dat = pred_dat,
    fit_anova=fit_anova,
    fit_lme = fit_lme
  ))
}
##################Section for plotting#################################

#This function will create the plots for either a "linear" or "quadratic"
   response
```

```r
plot_example <- function(sim_dat) {
  ## Plot the simulated data (scatterplot)

  p1 <- sim_dat$dat %>%
    ggplot(aes(x = time,
               y = y,
               group = treatment,
               color = treatment)
           ) +
    geom_point(show.legend=FALSE) +
    labs(y='response')+
    geom_line(aes(x = time,
                  y = mu,
                  color = treatment),
              show.legend=FALSE) +
    theme_classic() +
    theme(plot.title = element_text(size = 30,
                                    face = "bold"),
        text=element_text(size=30))+
    thm

  #plot the simulated data with trajectories per each subject
  p2 <- sim_dat$dat %>%
    ggplot(aes(x = time,
               y = y,
               group = subject,
               color = treatment)
           ) +
    geom_line(aes(size = "Subjects"),
              show.legend = FALSE) +
    # facet_wrap(~ treatment) +
    geom_line(aes(x = time,
                  y = mu,
                  color = treatment,
                  size = "Simulated Truth"),
              lty = 1,show.legend = FALSE) +
    labs(y='response')+
    scale_size_manual(name = "Type", values=c("Subjects" = 0.5, "Simulated
        Truth" = 3)) +
    theme_classic()+
     theme(plot.title = element_text(size = 30,
                                   face = "bold"),
     text=element_text(size=30))+
    thm

  #plot the errors
   p3 <- sim_dat$dat %>%
    ggplot(aes(x = time,
               y = errors,
               group = subject,
               color = treatment)) +
    geom_line(show.legend=FALSE) +
     labs(y='errors')+
     theme_classic()+
     theme(plot.title = element_text(size = 30,
                                    face = "bold"),
        text=element_text(size=30))+
    thm

   #plot the model predictions for rm-ANOVA
  p4 <- ggplot(sim_dat$dat,
               aes(x = time,
                   y = y,
                   color = treatment)) +
    geom_point(show.legend=FALSE)+
    labs(y='response')+
    geom_line(aes(y = predict(sim_dat$fit_anova),
                  group = subject, size = "Subjects"),show.legend = FALSE)
                      +
    geom_line(data = sim_dat$pred_dat,
              aes(y = predict(sim_dat$fit_anova,
                              level = 0,
                              newdata = sim_dat$pred_dat),
                  size = "Population"),
              show.legend=FALSE) +
    guides(color = guide_legend(override.aes = list(size = 2)))+
    scale_size_manual(name = "Predictions",
                      values=c("Subjects" = 0.5, "Population" = 3)) +
    theme_classic() +
    theme(plot.title = element_text(size = 30,
                                    face = "bold"),
        text=element_text(size=30))+
    thm



   #plot the LMEM predictions
  p5 <- ggplot(sim_dat$dat,
               aes(x = time,
                   y = y,
                   color = treatment)) +
    geom_point()+
    labs(y='response')+
    geom_line(aes(y = predict(sim_dat$fit_lme),
                  group = subject, size = "Subjects")) +
    geom_line(data = sim_dat$pred_dat,
              aes(y = predict(sim_dat$fit_lme,
                              level = 0,
                              newdata = sim_dat$pred_dat),
                  size = "Population")) +
    guides(color = guide_legend(override.aes = list(size = 2)))+
    scale_size_manual(name = "Predictions",
                      values=c("Subjects" = 0.5, "Population" = 3)) +
    theme_classic() +
    theme(plot.title = element_text(size = 30,
                                    face = "bold"),
        text=element_text(size=30))+
    thm
  return((p1+p3+p2+p4+p5)+plot_layout(nrow=1)+plot_annotation(tag_levels =
      'A'))


}

txt<-18

#Store each plot in a separate object
A1<-plot_example(example(fun_type = "linear", error_type = "correlated"))

B1<-plot_example(example(fun_type = "linear", error_type = "independent"))

C1<-plot_example(example(fun_type = "quadratic", error_type = "correlated"
   ))

D1<-plot_example(example(fun_type = "quadratic", error_type = "independent
   "))
```

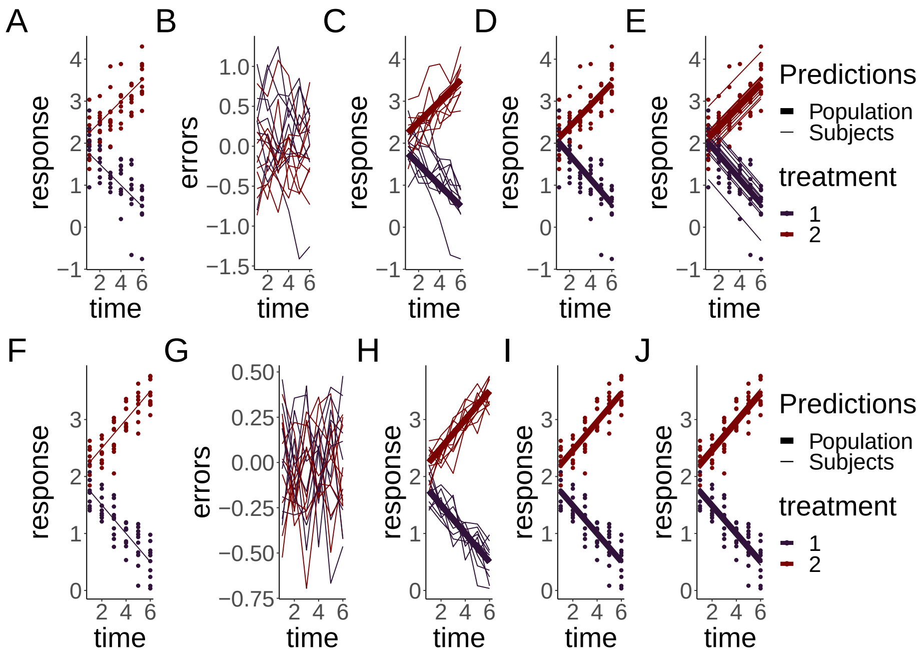


Figure A.1: Simulated linear responses from two groups with correlated (top row) or independent (bottom row) errors using a rm-ANOVA model and a LMEM. A, F:Simulated data with known mean response and individual responses (points) showing the dispersion of the data. B,G: Generated errors showing the difference in the behavior of correlated and independent errors. C,H: Simulated data with thin lines representing individual trajectories. D,I: Estimations from the rm-ANOVA model for the mean group response. E, J: Estimations from the LMEM for the mean group response and individual responses (thin lines). In all panels, thick lines are the predicted mean response per group, thin lines are the random effects for each subject and points represent the original raw data. Both rm-ANOVA and the LMEM are able to capture the trend of the data.

For the quadratic response case, Figure A.2 shows the simulated responses using compound symmetry and independent errors.

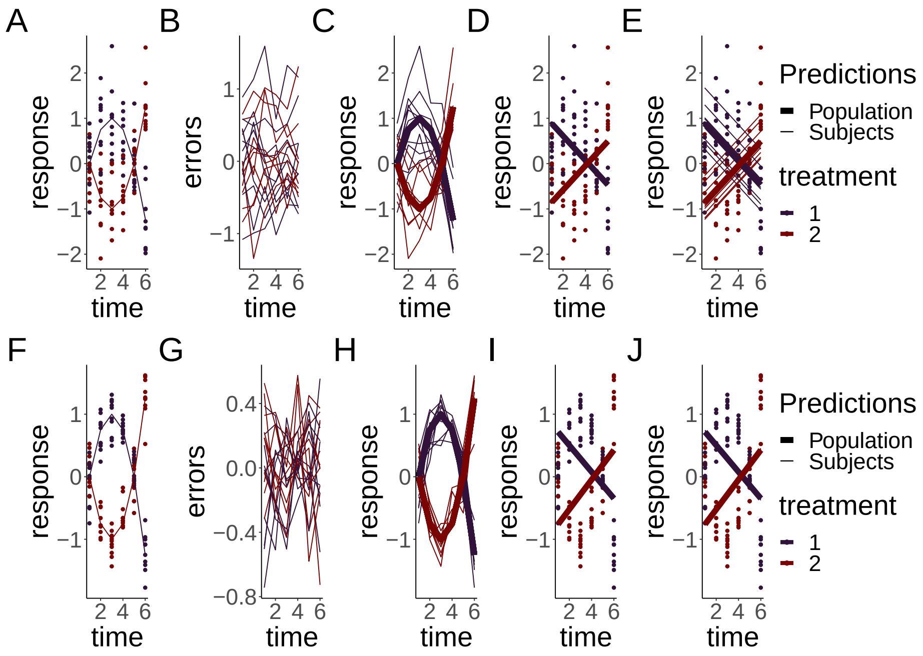


Figure A.2: Simulated quadratic responses from two groups with correlated (top row) or independent (bottom row) errors using a rm-ANOVA model and a LMEM. A, F:Simulated data with known mean response and individual responses (points) showing the dispersion of the data. B,G: Generated errors showing the difference in the behavior of correlated and independent errors. C,H: Simulated data with thin lines representing individual trajectories. D,I: Estimations from the rm-ANOVA model for the mean group response. E, J: Estimations from the LMEM for the mean group response and individual responses (thin lines). In all panels, thick lines are the predicted mean response per group, thin lines are the random effects for each subject and points represent the original raw data. Both rm-ANOVA and the LMEM are not able to capture the changes in each group over time.

A.2 Basis functions and GAMs

This code produces Figure 2 from the main manuscript. Briefly, a non-linear (quadratic) response is simulated a gam model is fitted and the basis are extracted in order to explain how the smooth is constructed. The code for data simulation is used again here for the sake of keeping the same structure, although the data can be simulated in a more simple fashion.

```r
#generate the response: the same initial procedure from the previous
   section to simulate
#the response
set.seed(1)
n_time = 6
 x <- seq(1,6, length.out = n_time)
 mu <- matrix(0, length(x), 2)
 mu[, 1] <-  -(0.25 * x^2) +1.5*x-1.25 #mean response
 mu[, 2] <- (0.25 * x^2) -1.5*x+1.25 #mean response
 y <- array(0, dim = c(length(x), 2, 10))
 errors <- array(0, dim = c(length(x), 2, 10))
 for (i in 1:2) {     # number of treatments
     for (j in 1:10) {  # number of subjects
         # compound symmetry errors
         errors[, i, j] <- rmvn(1, rep(0, length(x)), 0.1 * diag(6) + 0.25
             * matrix(1, 6, 6))
         y[, i, j] <- mu[, i] + errors[, i, j]
     }
 }

 #label each table
  dimnames(y) <- list(time = x, treatment = 1:2, subject = 1:10)
 dimnames(errors) <- list(time = x, treatment = 1:2, subject = 1:10)
 dimnames(mu) <- list(time = x, treatment = 1:2)

 #Convert to dataframes with subject, time and group columns
 dat <- as.data.frame.table(y, responseName = "y")
 dat_errors <- as.data.frame.table(errors, responseName = "errors")
 dat_mu <- as.data.frame.table(mu, responseName = "mu")
 dat <- left_join(dat, dat_errors, by = c("time", "treatment", "subject"))
 dat <- left_join(dat, dat_mu, by = c("time", "treatment"))
 dat$time <- as.numeric(as.character(dat$time))

 #label subject per group
 dat <- dat %>%
     mutate(subject = factor(paste(subject, treatment, sep = "-")))

 #extract  "Group 1" to fit the GAM
  dat<-subset(dat,treatment==1)
 #keep just the response and timepoint columns
   dat<-dat[,c('y','time')]

   #GAM model of time, 5 knots
gm<-gam(y~s(time,k=5),data=dat)

#model_matrix (also known as) 'design matrix'
#will contain the smooths used to create  model 'gm'
model_matrix<-as.data.frame(predict(gm,type='lpmatrix'))
```

```r
time<-c(1:6)

basis<-model_matrix[1:6,] #extracting basis (because the values are
   repeated after every 6 rows)
#basis<-model_matrix[1:6,-1] #extracting basis
colnames(basis)[colnames(basis)=="(Intercept)"]<-"s(time).0"
basis<-basis %>% #pivoting to long format
  pivot_longer(
    cols=starts_with("s")
  )%>%
  arrange(name) #ordering

#length of dataframe to be created: number of knots by number of
   timepoints (minus 1 for the intercept that we won't plot)
ln<-6*(length(coef(gm)))

basis_plot<-data.frame(Basis=integer(ln),
                       value_orig=double(ln),
                       time=integer(ln),
                       cof=double(ln)
)

basis_plot$time<-rep(time) #pasting timepoints
basis_plot$Basis<-factor(rep(c(1:5),each=6)) #pasting basis number values
basis_plot$value_orig<-basis$value #pasting basis values
basis_plot$cof<-rep(coef(gm)[1:5],each=6) #pasting coefficients
basis_plot<-basis_plot%>%
  mutate(mod_val=value_orig*cof) #the create the predicted values the
     bases need to be
#multiplied by the coefficients

#creating labeller to change the labels in the basis plots

basis_names<-c(
  `1`="Intercept",
  `2`="1",
  `3`="2",
  `4`="3",
  `5`="4"
)

#calculating the final smooth by aggregating the basis functions

smooth<-basis_plot%>%
  group_by(time)%>%
  summarize(smooth=sum(mod_val))


#original basis
sz<-1
p11<-ggplot(basis_plot,
            aes(x=time,
                y=value_orig,
                colour=as.factor(Basis)
                )
            )+
  geom_line(size=sz,
            show.legend=FALSE)+
  geom_point(size=sz+1,
             show.legend = FALSE)+
  labs(y='Basis functions')+
  facet_wrap(~Basis,
             labeller = as_labeller(basis_names)
             )+
  theme_classic()+
  thm


#penalized basis
p12<-ggplot(basis_plot,
            aes(x=time,
                y=mod_val,
                colour=as.factor(Basis)
                )
            )+
  geom_line(show.legend = FALSE,
            size=sz)+
  geom_point(show.legend = FALSE,
             size=sz+1)+
  labs(y='Penalized \n basis functions')+
  scale_y_continuous(breaks=seq(-1,1,1))+
  facet_wrap(~Basis,
             labeller=as_labeller(basis_names)
             )+
  theme_classic()+
  thm

#heatmap of the coefficients
x_labels<-c("Intercept","1","2","3","4")
p13<-ggplot(basis_plot,
            aes(x=Basis,
                y=Basis))+
  geom_tile(aes(fill = cof),
            colour = "black") +
    scale_fill_gradient(low = "white",
                        high = "#B50A2AFF")+ #color picked from KikiMedium
  labs(x='Basis',
       y='Basis')+
  scale_x_discrete(labels=x_labels)+
  geom_text(aes(label=round(cof,2)),
            size=7,
            show.legend = FALSE)+
  theme_classic()+
  theme(legend.title = element_blank())

#plotting simulated datapoints and smooth term
```

```
p14<-ggplot(data=dat,
            aes(x=time,y=y))+
  geom_point(size=sz+1)+
  labs(y='Simulated \n response')+
  geom_line(data=smooth,
            aes(x=time,
                y=smooth),
            color="#6C581DFF",
            size=sz+1)+
  theme_classic()


#Combining all
b_plot<-p11+p13+p12+p14+plot_annotation(tag_levels='A')&
  theme(
     text=element_text(size=18)
     )
```

A.3 Longitudinal biomedical data simulation and GAMs

This section describes how to fit GAMs to longitudinal data using simulated data. First, data is simulated according to Section 6, where reported data of oxygen saturation (StO_2) in tumors under either chemotherapy or saline control is used as a starting point to generate individual responses in each group.

```
set.seed(1)
#Dataframe that contains the original reported trends
dat<-tibble(StO2=c(4,27,3,2,0.5,7,4,50,45,56),
            Day=rep(c(0,2,5,7,10),times=2),
            Group=as.factor(rep(c("Control","Treatment"),each=5))
            )


## plot the mean response
f1<-ggplot(dat,
           aes(x = Day,
               y = StO2,
               color = Group)) +
    geom_line(size=1,
              show.legend = FALSE)+
    geom_point(show.legend = FALSE,
               size=1.5,
               alpha=0.5)+
  labs(y=expression(paste(StO[2],
                           ' (real)')))+
  theme_classic()+
  thm+
    scale_x_continuous(breaks=c(0,5,10))+
    scale_y_continuous(breaks=c(0,40))+
  plot_layout(tag_level = 'new')+
  theme(
    plot.background = element_rect(fill = "transparent",
                                   color = NA),
    axis.text=element_text(size=14)
  )
```

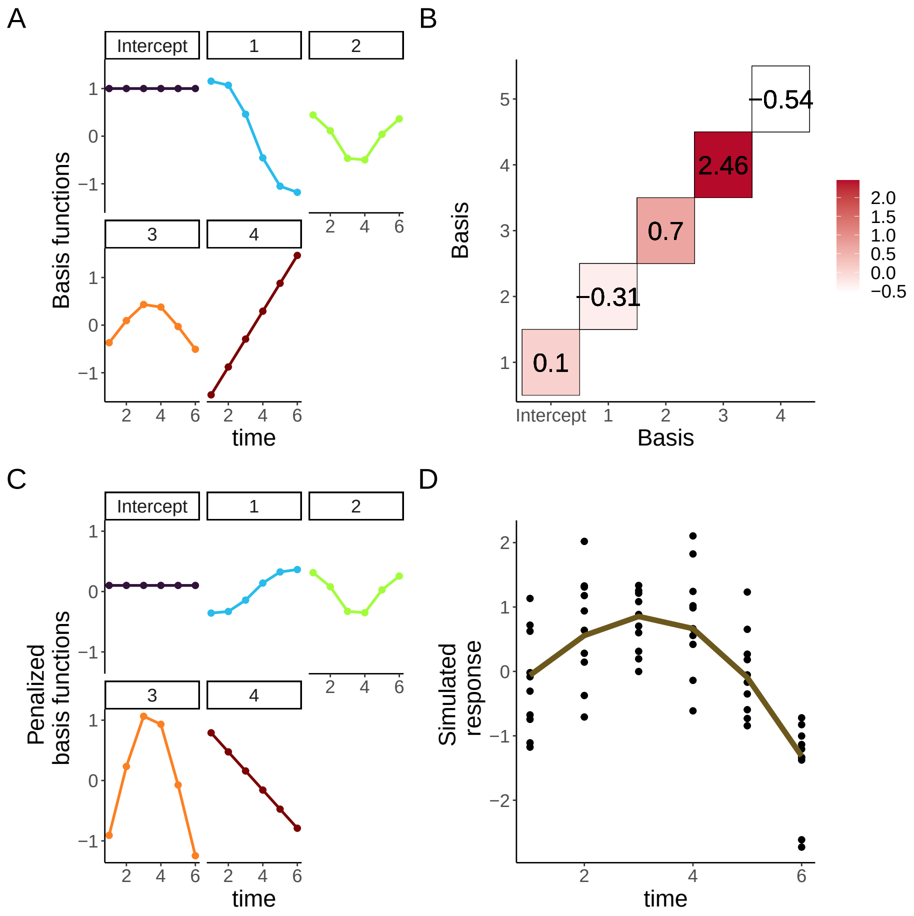


Figure A.3: Basis functions for a single smoother for time with five knots. A: Basis functions for a single smoother for time for the simulated data of Group 1 from Figure 2. B: Matrix of basis function weights. Each basis function is multiplied by a coefficient which can be positive or negative. The coefficient determines the overall effect of each basis in the final smoother. C: Weighted basis functions. Each of the four basis functions of panel A has been weighted by the corresponding coefficient shown in Panel B. Note the corresponding increase (or decrease) in magnitude of each weighted basis function. D: Smoother for time and original data points. The smoother (line) is the result of the sum of each weighted basis function at each time point, with simulated values for the group shown as points.

```r
#This function simulates data for the tumor data using default parameters
   of 10 observations per time point,and Standard deviation (sd) of 5%.
#Because physiologically StO2 cannot go below 0%, data is  generated with
   a cutoff value of 0.0001 (the "StO2_sim")

simulate_data <- function(dat, n = 10, sd = 5) {
    dat_sim <- dat %>%
        slice(rep(1:n(), each = n)) %>%
        group_by(Group, Day) %>%
        mutate(
               StO2_sim = pmax(rnorm(n, StO2, sd), 0.0001),
               subject=rep(1:10),
               subject=factor(paste(subject, Group, sep = "-"))
               ) %>%
        ungroup()

    return(dat_sim)
}


#subject = factor(paste(subject, treatment, sep = "-")))
n <- 10 #number of observations
sd <- 10 #approximate sd from paper
df <- 6
dat_sim <- simulate_data(dat, n, sd)

#plotting simulated data
f2<-ggplot(dat_sim,
           aes(x = Day,
               y = StO2_sim,
               color = Group)) +
    geom_point(show.legend=FALSE,
               size=1.5,
               alpha=0.5)+
    stat_summary(aes(y = StO2_sim,
                     group=Group),
                 fun=mean, geom="line",
                 size=1,
                 show.legend = FALSE)+
  labs(y=expression(atop(StO[2],
                          '(simulated)')))+
  theme_classic()+
  theme(
    axis.text=element_text(size=22)
  )+
  thm+
    scale_x_continuous(breaks=c(0,2,5,7,10))
```

A.4 A basic Workflow for GAMs

This section shows a basic workflow to fit a series of increasingly complex GAMs to the simulated data from the previous section. Graphical and parameter diagnostics for goodness of fit are discussed, as well as model comparison via AIC (Aikake Information Criterion).

A.4.1 First model

The first model fitted to the data is one that only accounts for different smooths by day. The model syntax specifies that `gam_00` is the object that will contain all the model information, and that the model attempts to explain changes in `StO2_sim` (simulated StO_2) using a smooth per `Day`. The model will use 5 knots (`k=5`) for the smooth. The smooth is constructed by default using thin plate regression splines. The smoothing parameter estimation method used is the restricted maximum likelihood (`REML`).

```
gam_00<-gam(StO2_sim ~ s(Day, k = 5),
            method='REML',
            data   = dat_sim)
```

To obtain model diagnostics, two methodologies are to be used: 1) graphical diagnostics, and 2) a model check. In the first case, the functions `appraise` and `draw` from the package *gratia* can be used to obtain a single output with all the graphical diagnostics. For model check, the functions `gam.check` and `summary` from *mgcv* provide detailed information about the model fit and its parameters.

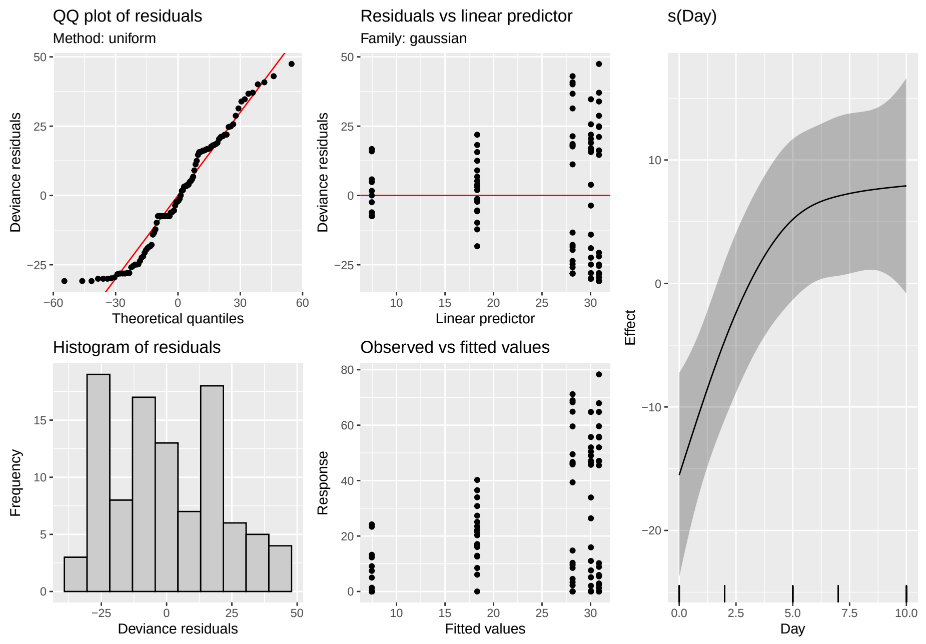


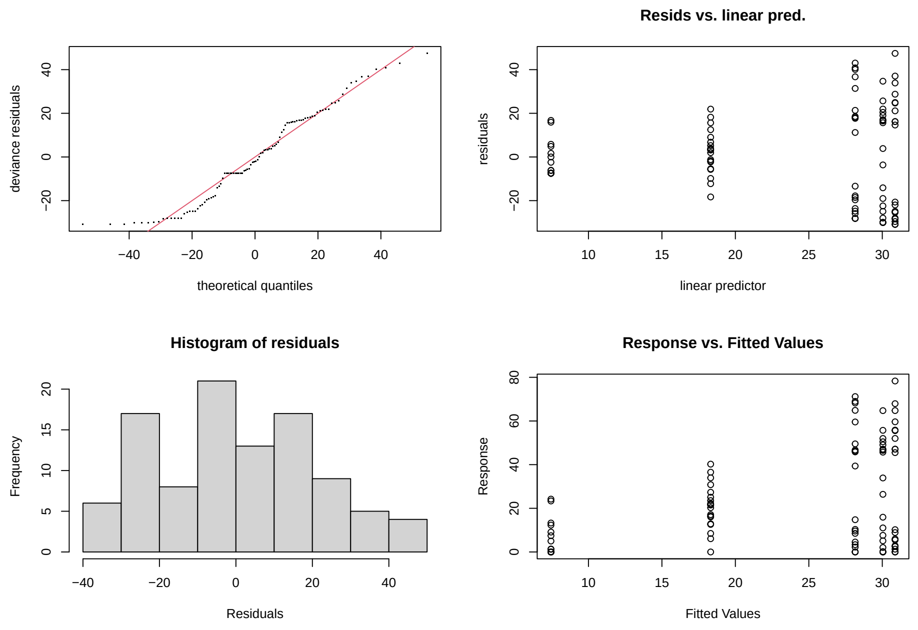
Figure A.4: Graphical diagnostics for the first GAM model. Left: Graphical diagnostics provided by the function `appraise` from the package *gratia*. Right: Fitted smooth for the model, provided by the function `draw`.

A.4.1.1 Graphical diagnostics From the output of the function `appraise` in Figure A.4, the major indicators of concern about the model are the QQ plot of residuals and the histogram of residuals. The QQ plot shows that the errors are not reasonably located along the 45° line (which indicates normality), as there are multiple points that deviate from the trend, specially in the tails. The histogram also shows that the variation (residuals) is not following the assumption of a normal distribution.

The `draw` function permits to plot the smooths as `ggplot2` objects, which eases subsequent manipulation, if desired. Because model `gam_00` specifies only one smooth for the time covariate (Day), the plot only contains only one smooth. Note that the smooth shows an almost linear profile.

A.4.1.2 Model check Special attention must be paid to the 'k-index' from `gam.check`. This parameter indicates if the basis dimension of the smooth is adequate, i.e., it checks that the basis used to create the smooth are adequate to capture the trends in the data. If the model is not adequately capturing the trens in the data, this is indicated by a low k-index value (<1).

```
gam.check(gam_00)
```



```
##
## Method: REML   Optimizer: outer newton
## full convergence after 5 iterations.
## Gradient range [-0.0003727881,-6.621452e-07]
## (score 444.0118 & scale 450.6638).
## Hessian positive definite, eigenvalue range [0.3881695,49.00676].
## Model rank =  5 / 5
##
## 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(Day) 4.00 2.11    0.36  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(gam_00)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
```

```
## StO2_sim ~ s(Day, k = 5)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   22.967      2.123   10.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##          edf Ref.df     F  p-value
## s(Day) 2.114  2.565 7.633 0.000517 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.153   Deviance explained = 17.2%
## -REML = 444.01  Scale est. = 450.66    n = 100
```

From the output, it can be seen that the k-index is 0.36, which indicates that the model is not capturing the variability in the data. The `edf` (effective degrees of freedom) is an indicator of the complexity of the smooth. Here the complexity of the smooth is comparable to that of a 4th degree polynomial.

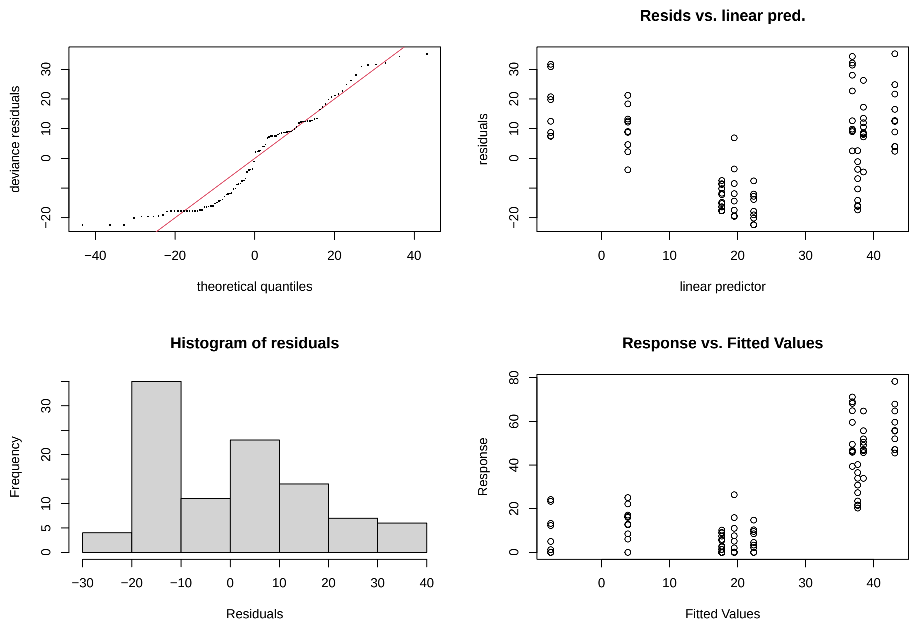
From the `summary` function, information about the assumed distribution of the errors (Gaussian in this case) and the link function can be obtained. The link function is 'identity' as the model does not make any transformation on the predictors. The 'significance of smooth terms' *p-value* indicates if each smooth is adding significance to the model. Here, the *p-value* is low but we have seen that there are issues with the model from the previous outputs. Finally, the 'deviance explained' indicates how much of the data the model is able to capture, which in this case corresponds to ≈ 17%.

A.4.2 Second model

The major flaw of `gam_00` is that this model is not taking into account the fact that the data is nested in groups. The next iteration is a model where a different smooth of time (Day) is assigned for each group using `by=Group` in the model syntax.

```
gam_01<-gam(StO2_sim ~ s(Day, by=Group,k = 5),
            method='REML',
            data  = dat_sim)

gam.check(gam_01)
```



```
##
## Method: REML   Optimizer: outer newton
## full convergence after 7 iterations.
## Gradient range [-5.51754e-05,2.671715e-06]
## (score 423.3916 & scale 280.8777).
## Hessian positive definite, eigenvalue range [0.3162258,48.5557].
## Model rank =  9 / 9
##
## 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(Day):GroupControl    4.00 3.39    0.43  <2e-16 ***
## s(Day):GroupTreatment 4.00 3.23    0.43  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(gam_01)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## StO2_sim ~ s(Day, by = Group, k = 5)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   22.967      1.676    13.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Approximate significance of smooth terms:
##                          edf Ref.df      F p-value
## s(Day):GroupControl    3.392  3.794  3.817  0.0304 *
## s(Day):GroupTreatment 3.229  3.682 21.174  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.472   Deviance explained = 50.8%
## -REML = 423.39  Scale est. = 280.88    n = 100
```

Diagnostics for this model indicate that the k-index is still below 1 (0.43 from `gam.check`), and that the residuals are still not following a normal distribution (Figure A.5). Moreover, the smooths (plotted via the `draw()` function) appear with a fairly linear profile, which indicates they are still not capturing the trends observed in the data. From `summary()`, the deviance explained by the model is ≈ 51%.

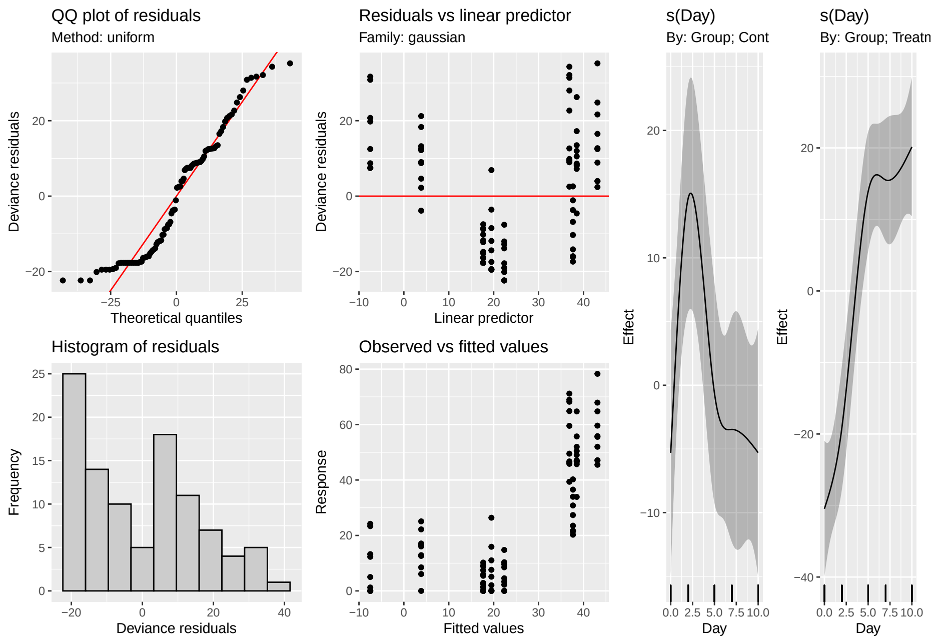


Figure A.5: Graphical diagnostics for the second GAM model. Left: Graphical diagnostics provided by the function `appraise` from the package *gratia*. Right: Fitted smooth for the model, provided by the function `draw`.

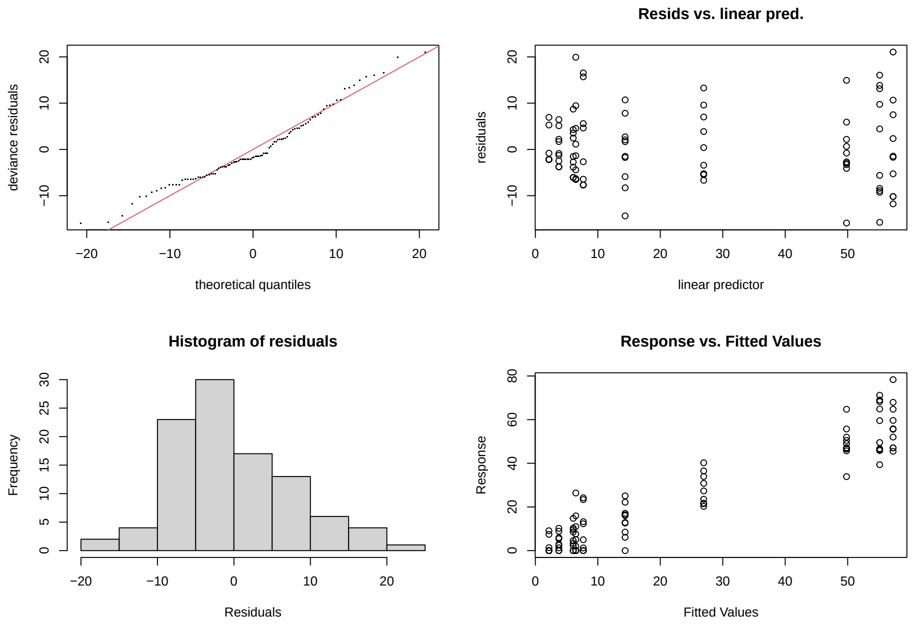
A.4.3 Third model

Model `gam_00` was built for didactic purposes to cover the simplest case, but it does not account for the nesting of the data by Group, which is apparent from the type of smooth fitted, the model diagnostics, and, the low variance explained by the model. On the other hand, `gam_01` takes into account the nesting within each group and provides better variance explanation, but as indicated in Section 6, in order to differentiate between each group a parametric term needs to be added to the model for the interaction of *Day* and *Group*.

```
#GAM for StO2

m1 <- gam(StO2_sim ~ Group+s(Day, by = Group, k = 5),
          method='REML',
          data  = dat_sim)

gam.check(m1)
```



```
##
## Method: REML   Optimizer: outer newton
## full convergence after 10 iterations.
## Gradient range [-8.164307e-08,1.500338e-08]
## (score 355.8554 & scale 64.53344).
## Hessian positive definite, eigenvalue range [1.174841,48.08834].
## Model rank =  10 / 10
##
## 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(Day):GroupControl   4.00 3.87    1.02    0.59
## s(Day):GroupTreatment 4.00 3.88    1.02    0.54
```

```
summary(m1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## StO2_sim ~ Group + s(Day, by = Group, k = 5)
##
## Parametric coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)       9.084      1.136   7.996 4.09e-12 ***
## GroupTreatment   27.766      1.607  17.282  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
```

```
##                              edf  Ref.df      F  p-value
## s(Day):GroupControl    3.873   3.990  17.57   <2e-16  ***
## s(Day):GroupTreatment  3.879   3.991  89.33   <2e-16  ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.879    Deviance explained = 88.9%
## -REML = 355.86   Scale est. = 64.533    n = 100
```

The resulting model is `m1`, which is the model fitted in the main manuscript. By using `appraise()` and `draw` on this model (Figure A.6) we see that the trend on the QQ plot has improved, the histogram of the residuals appears to be reasonably distributed, and the smooths are capturing the trend of the data within each group. From `gam.check`, the k-index is now at an acceptable value (≈ 1.02), and `summary` now indicates that the model is able to capture 89% of the variance in the data.

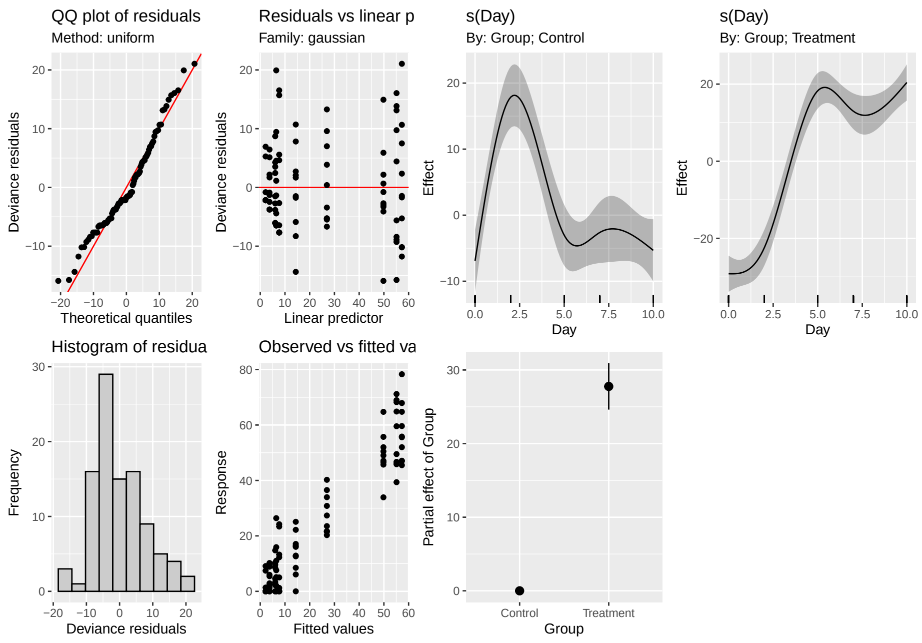


Figure A.6: Graphical diagnostics for the final GAM model. Left: Graphical diagnostics provided by the function `appraise` from the package *gratia*. Right: Fitted smooths for the model, provided by the function `draw`.

A.4.4 Comparing models via AIC

One final comparison that can be made for model selection involves the use of the Aikake Information Criterion (AIC). This metric is used to estimate information loss, which we want to minimize with an appropriate model. Therefore, when 2 or more models are compared, the model with lower AIC is preferred. In R, the comparison is done using the `AIC` function.

```
AIC(gam_00,gam_01,m1)
```

```
##                df       AIC
## gam_00   4.564893 900.8257
## gam_01   9.476137 858.6051
## m1      10.980983 712.2067
```

The output in this case is expected: model m1 has a lower AIC (712.46) whereas the initial two models have higher AICs (900 and 858). The AIC should not be considered as the only estimator of model quality, instead to be used as complimentary information to the graphical diagnostics and model checks described above.

A.4.4.1 Pairwise comparisons of smooth confidence intervals The estimation of significant differences between each treatment group can be achieved via pairwise comparisons of the smooth confidence intervals as described in section 6.3. In this case, the "design matrix" is used to estimate the pairwise comparisons (see main manuscript for details and associated references). Briefly, the "design matrix" (also known as the "Xp matrix") from the selected model (m1) is used to calculate a 95% confidence interval of the difference between the smooth terms for each group. This approach allows to estimate the time intervals where a significant difference exists between the groups (confidence interval above or below 0). **All pairwise comparisons in this paper have been centered at the response scale to ease interpretation** .

```
##Pairwise comparisons
pdat <- expand.grid(Day = seq(0, 10, length = 400),
                    Group = c('Control', 'Treatment'))

##matrix that contains the basis functions evaluated at the points in pdat
    xp <- predict(m1, newdata = pdat, type = 'lpmatrix')


#Find columns in xp where the name contains "Control"
    c1 <- grepl('Control', colnames(xp))

#Find columns in xp where the name contains 'Treatment'
    c2 <- grepl('Treatment', colnames(xp))

#Find rows in pdat that correspond to either 'Control' or 'Treatment'
    r1 <- with(pdat, Group == 'Control')
    r2 <- with(pdat, Group == 'Treatment')

# In xp: find the rows that correspond to Control or Treatment, those that
    do not match will be
    #set to zero. Then, substract the values from the rows corresponding
       to 'Control' from those that correspond
    #to 'Treatment'
    X <- xp[r1, ] - xp[r2, ]

    ## remove columns that do not contain name 'Control' or 'Treatment'
    X[, ! (c1 | c2)] <- 0
    ## zero out the parametric cols, those that do not contain in the
       characters 's('
    #X[, !grepl('^s\\(', colnames(xp))] <- 0

    #Multiply matrix by model coefficients. X has (p,n) (rows, columns)
       and the coefficient matrix has
    #dimensions (n,1). The resulting matrix has dimensions (p,1)
    dif <- X %*% coef(m1)

    #comp<-test %*% coef(gam1)[3:10]

#Calculate standard error for the computed differences using the variance-
   covariance matrix
    #of the model
    se <- sqrt(rowSums((X %*% vcov(m1, unconditional = FALSE)) * X))
    crit <- qt(0.05/2, df.residual(m1), lower.tail = FALSE)
    #upper  limits
    upr <- dif + (crit * se)
    #lower limits
    lwr <- dif - (crit * se)
    #put all components in a dataframe for plotting
    comp1<-data.frame(pair = paste('Control', 'Treatment', sep = '-'),
               diff = dif,
               se = se,
               upper = upr,
               lower = lwr)



#add time point sequence
comp_StO2 <- cbind(Day = seq(0, 10, length = 400),
                   rbind(comp1))

#use function from the pairwise comparison plot in the manuscript to get
   the shaded regions

    my_list<-pairwise_limits(comp_StO2)
  rib_col<-'#EDD03AFF' #color for the ribbon
#plot the difference
c1<-ggplot(comp_StO2, aes(x = Day, y = diff, group = pair)) +
  #shaded region
  annotate("rect",
              xmin =my_list$init1, xmax =my_list$final1,ymin=-Inf,ymax=
                 Inf,
              fill='#30123BFF',
              alpha = 0.5,
              ) +
  annotate("text",
           x=1.5,
           y=-10,
           label="Control",size=10
         )+
  #shaded region
  annotate("rect",
           xmin =my_list$init2, xmax =my_list$final2,ymin=-Inf,ymax=Inf,
           fill='#7A0403FF',
           alpha = 0.5
         ) +
  annotate("text",
           x=6,
           y=-10,
           label="Treatment",
           size=10
         )+
  #ribbon for difference confidence interval
  geom_ribbon(aes(ymin = lower, ymax = upper),
              alpha = 0.5,
              fill=rib_col) +
    geom_line(color='black',size=1) +
    geom_line(data=comp_StO2,aes(y=0),size=0.5)+
    facet_wrap(~ pair) +
    theme_classic()+
    labs(x = 'Days', y = expression(paste('Difference in StO'[2] )))+
    scale_x_continuous(breaks=c(0,2,5,7,10))+
    theme(
        text=element_text(size=18),
        legend.title=element_blank()
    )
```

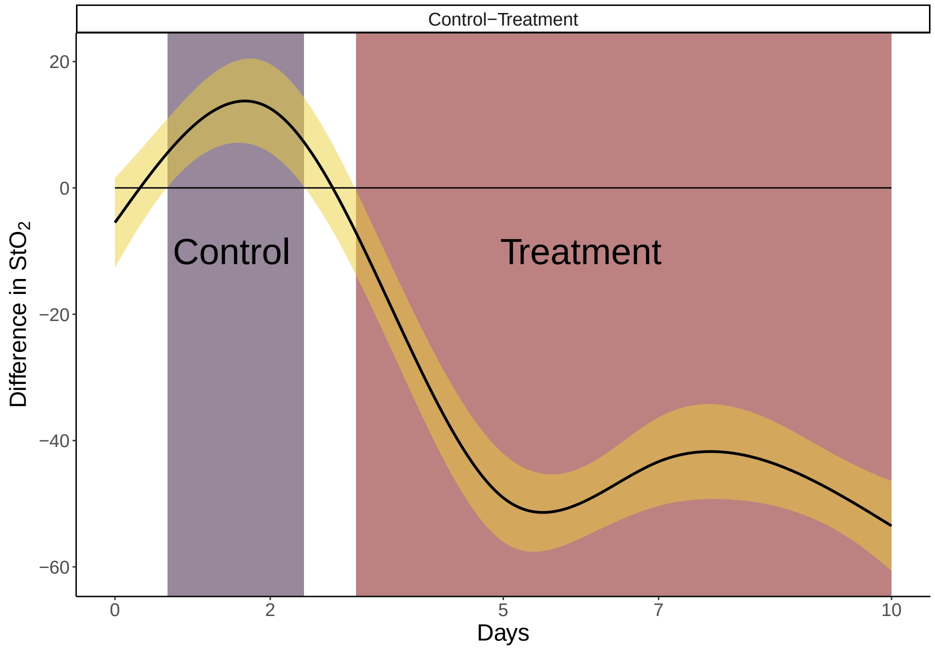


Figure A.7: Smooth pairwise comparisons for model `m1` using a 95% confidence interval for the difference between smooths. The comparison is centered at the response scale. Shaded regions indicate time intervals where each treatment group has a non-zero effect.

Of notice, a convenient wrapper for the function described above exists in the package `gratia`. In this package, `difference_smooths` is a function that makes the comparisons and produces Figure A.7 when is used on a fitted model. The function syntax and an example can be found at:

https://cran.r-project.org/web/packages/gratia/gratia.pdf

Keep in mind that this function **does not** center the pairwise comparison at the response scale, so it has to be shifted in order to be compared to the raw data.

A.5 GAM and Linear model plots and Missing data

This section covers the code used to generate Figure 3, where the simulated data, fit of the "final" GAM (`m1`), linear model and GAM on data with missing observations are presented. Note that panel A in Figure 3 and the inset are generated in the code chunk where the data is simulated in Section A.3, and are called later to build the figure.

A.5.1 GAM and Linear model plots

This code chunk creates panels B and D in Figure 3. Note that this code uses the final GAM from the previous section (`m1`), so the simulated data and the model should be generated before running this section.

```
#linear model
lm1<-lm(StO2_sim ~ Day + Group + Day * Group, data = dat_sim)


#creates a dataframe using the length of the covariates for the GAM
gam_predict <- expand_grid(Group = factor(c("Control", "Treatment")),
                         Day = seq(0, 10, by = 0.1),
                         subject=factor(rep(1:10)))

#creates a dataframe using the length of the covariates for rm-ANOVA
lm_predict<-expand_grid(Group = factor(c("Control", "Treatment")),
                         Day = c(0:10),
                        subject=factor(rep(1:10)),
                          )
lm_predict$subject<-factor(paste(lm_predict$subject, lm_predict$Group, sep
    = "-"))

#adds the predictions to the grid and creates a confidence interval for
   GAM
gam_predict<-gam_predict%>%
    mutate(fit = predict(m1,gam_predict,se.fit = TRUE,type='response')$fit
        ,
           se.fit = predict(m1, gam_predict,se.fit = TRUE,type='response')
              $se.fit)

#using lm
lm_predict<-lm_predict%>%
    mutate(fit = predict(lm1,lm_predict,se.fit = TRUE,type='response')$fit
        ,
           se.fit = predict(lm1, lm_predict,se.fit = TRUE,type='response')
              $se.fit)

#plot smooths and confidence interval for GAM
f3<-ggplot(data=dat_sim, aes(x=Day, y=StO2_sim, group=Group)) +
    geom_point(aes(color=Group),size=1.5,alpha=0.5,show.legend = FALSE)+
  geom_ribbon(aes( x=Day,ymin=(fit - 2*se.fit),
                  ymax=(fit + 2*se.fit),
                  fill=Group
                  ),
              alpha=0.3,
              data=gam_predict,
            show.legend=FALSE,
                inherit.aes=FALSE) +
  geom_line(aes(y=fit,
                color=Group),
              size=1,data=gam_predict,
              show.legend = FALSE)+
  #facet_wrap(~Group)+
  labs(y=expression(atop(StO[2],'complete')))+
    scale_x_continuous(breaks=c(0,2,5,7,10))+
      theme_classic()+
  theme(
    axis.text=element_text(size=22)
  )+
      thm+
  thm1

#plot linear fit for rm-ANOVA
f4<-ggplot(data=dat_sim, aes(x=Day, y=StO2_sim, group=Group)) +
    geom_point(aes(color=Group),size=1.5,alpha=0.5,show.legend = FALSE)+
  geom_ribbon(aes( x=Day,ymin=(fit - 2*se.fit),
                   ymax=(fit + 2*se.fit),fill=Group),
              alpha=0.3,
              data=lm_predict,
              show.legend = FALSE,
                inherit.aes=FALSE) +
  geom_line(aes(y=fit,
                color=Group),
              size=1,data=lm_predict,
              show.legend = FALSE)+
  #facet_wrap(~Group)+
  labs(y=expression(paste('StO'[2],' (simulated)')))+
    scale_x_continuous(breaks=c(0,2,5,7,10))+
      theme_classic()+
  theme(
    axis.text=element_text(size=22)
  )+
      thm+
  thm1



#posthoc comparisons for the linear model
#library(multcomp)


#summary(glht(lm1, linfct = mcp(Group = 'Tukey')))
#summary(glht(lm1, linfct=mcp(Group="Tukey", interaction_average=TRUE)))
```

A.6 Working with Missing data in GAMs

This code chunk first randomly deletes 40% of the total observations in the original simulated data, and then an interaction GAM is fitted to the remaining data. Model diagnostics are presented, and an object that stores the fitted smooths is saved to be called in the final code chunk to build the figure.

```
#missing data
#create a sequence of 40 random numbers between 1 and 100, these numbers
   will
#correspond to the row numbers to be randomly erased from the original
   dataset

missing <- sample(1:100, 40)
```

```
#create a new dataframe from the simulated data with 40 rows randomly
   removed, keep the missing values as NA

ind <- which(dat_sim$StO2_sim %in% sample(dat_sim$StO2_sim, 40))

#create a new dataframe, remove the StO2 column
dat_missing <- dat_sim[,-1]

#add NAs at the ind positions
dat_missing$StO2_sim[ind]<-NA

#Count the number of remaining observations per day (original dataset had
   10 per group per day)
dat_missing %>%
    group_by(Day,Group) %>%
    filter(!is.na(StO2_sim))%>%
  count(Day)


#the same model used for the full dataset
mod_m1 <- gam(StO2_sim ~ Group+s(Day,by=Group,k=5), data  = dat_missing,
   family=scat)
#appraise the model
appraise(mod_m1)


m_predict <- expand_grid(Group = factor(c("Control", "Treatment")),
                         Day = seq(0, 10, by = 0.1))

#adds the predictions to the grid and creates a confidence interval
m_predict<-m_predict%>%
    mutate(fit = predict(mod_m1,m_predict,se.fit = TRUE,type='response')$
       fit,
           se.fit = predict(mod_m1, m_predict,se.fit = TRUE,type='response
              ')$se.fit)


f6<-ggplot(data=dat_missing, aes(x=Day, y=StO2_sim, group=Group)) +
    geom_point(aes(color=Group),size=1.5,alpha=0.5,show.legend = FALSE)+
  geom_ribbon(aes( x=Day,ymin=(fit - 2*se.fit),
                   ymax=(fit + 2*se.fit),
                   fill=Group
                   ),
              alpha=0.3,
              data=m_predict,
            show.legend=FALSE,
                inherit.aes=FALSE) +
  geom_line(aes(y=fit,
                color=Group),
              size=1,data=m_predict,
              show.legend = TRUE)+
  #facet_wrap(~Group)+
  labs(y=expression(atop(StO[2],'missing')))+
    scale_x_continuous(breaks=c(0,2,5,7,10))+
      theme_classic()+
  theme(
    axis.text=element_text(size=22)
  )+
      thm+
  thm1
```

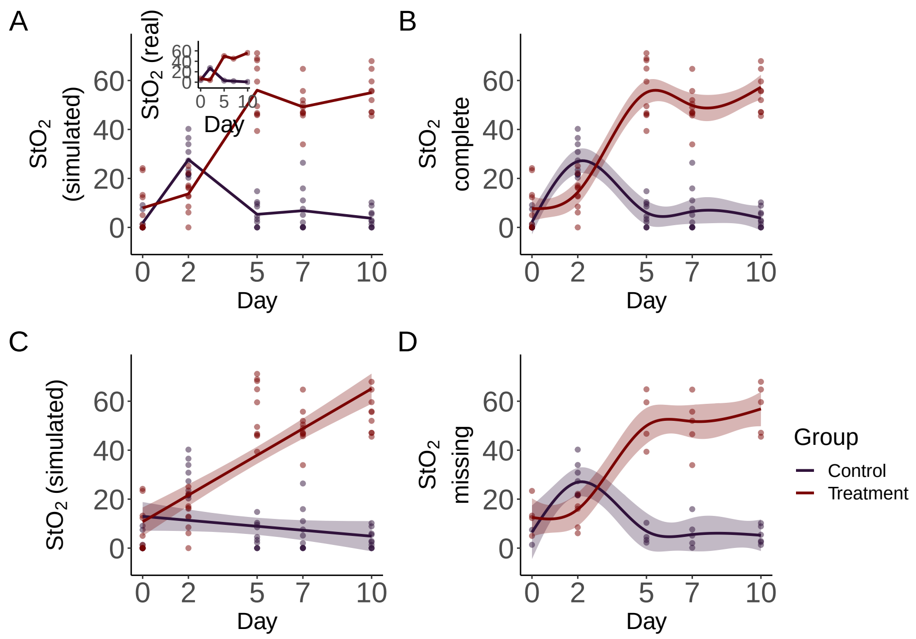


Figure A.8: Simulated data and smooths for oxygen saturation in tumors. A: Simulated data that follows previously reported trends (inset) in tumors under chemotherapy (Treatment) or saline (Control) treatment. Simulated data is from a normal distribution with standard deviation of 10% with 10 observations per time point. Lines indicate mean oxygen saturation B: Smooths from the GAM model for the full simulated data with interaction of Group and Treatment. Lines represent trends for each group, shaded regions are 95% confidence intervals. C: The rm-ANOVA model for the simulated data, which does not capture the changes in each group over time. D: Smooths for the GAM model for the simulated data with 40% of its observations missing. Lines represent trends for each group, shaded regions are 95% empirical Bayesian confidence intervals.

A.7 Pairwise comparisons in GAMs: full and missing data cases

The next code chunk reproduces Figure 4. Here pairwise comparisons are made for the full and missing datasets.

```
##Pairwise comparisons

pdat <- expand.grid(Day = seq(0, 10, length = 400),
                    Group = c('Control', 'Treatment'))

#this function takes the model, grid and groups to be compared using the
   lpmatrix
#originally developed by G. Simpson:
#https://fromthebottomoftheheap.net/2017/10/10/difference-splines-i/

smooth_diff <- function(model, newdata, g1, g2, alpha = 0.05,
                        unconditional = FALSE) {
```

```
    xp <- predict(model, newdata = newdata, type = 'lpmatrix')
    #Find columns in xp where the name contains "Control" and "Treatment"
    col1 <- grepl(g1, colnames(xp))
    col2 <- grepl(g2, colnames(xp))
    #Find rows in xp that correspond to each treatment
    row1 <- with(newdata, Group == g1)
    row2 <- with(newdata, Group == g2)
    ## difference rows of xp for data from comparison
    X <- xp[row1, ] - xp[row2, ]
    ## zero out cols of X related to splines for other lochs
    X[, ! (col1 | col2)] <- 0

    ## zero out the parametric cols
    #This line has been commented to keep the comparison at the response
       level,
    #otherwise it gives the marginal change between smooths
    #X[, !grepl('^s\\(', colnames(xp))] <- 0
    dif <- X %*% coef(model)
    #get standard error, critical value and boundaries
    se <- sqrt(rowSums((X %*% vcov(model, unconditional = unconditional))
       * X))
    crit <- qt(alpha/2, df.residual(model), lower.tail = FALSE)
    upr <- dif + (crit * se)
    lwr <- dif - (crit * se)
    data.frame(pair = paste(g1, g2, sep = '-'),
               diff = dif,
               se = se,
               upper = upr,
               lower = lwr)
}

#use the function to calculate the difference in smooths
comp1<-smooth_diff(m1,pdat,'Control','Treatment')

#Create a dataframe with time, comparisons and labels for regions where
   difference exists
comp_StO2_full <- cbind(Day = seq(0, 10, length = 400),
                   rbind(comp1)) %>%
  mutate(interval=case_when(
    upper>0 & lower<0~"no-diff",
    upper<0~"less",
    lower>0~"greater"
  ))

pairwise_limits<-function(dataframe){
    #extract values where the lower limit of the ribbon is greater than
       zero
    #this is the region where the control group effect is greater
    v1<-dataframe%>%
        filter(lower>0)%>%
        select(Day)
    #get day  initial value
    init1=v1$Day[[1]]
    #get day final value
    final1=v1$Day[[nrow(v1)]]

    #extract values where the value of the upper limit of the ribbon is
       lower than zero
    #this corresponds to the region where the treatment group effect is
       greater
    v2<-comp_StO2_full%>%
        filter(upper<0)%>%
        select(Day)

    init2=v2$Day[[1]]
    final2=v2$Day[[nrow(v2)]]
    #store values
    my_list<-list(init1=init1,
                  final1=final1,
                  init2=init2,
                  final2=final2)
return(my_list)
}

my_list<-pairwise_limits(comp_StO2_full)
rib_col<-'#EDD03AFF'

c1<-ggplot(comp_StO2_full, aes(x = Day, y = diff, group = pair)) +
    annotate("rect",
                xmin =my_list$init1, xmax =my_list$final1,ymin=-Inf,ymax=
                   Inf,
                fill='#30123BFF',
                alpha = 0.5,
                ) +
  annotate("text",
             x=1.5,
             y=-18,
             label="Control>Treatment",
           size=8,
           angle=90
           )+
    annotate("rect",
             xmin =my_list$init2, xmax =my_list$final2,ymin=-Inf,ymax=Inf,
             fill='#7A0403FF',
             alpha = 0.5,
    ) +
  annotate("text",
             x=6,
             y=-18,
             label="Treatment>Control",
             size=8,
           angle=90
           )+
    geom_ribbon(aes(ymin = lower, ymax = upper),
                alpha = 0.5,
                fill=rib_col) +
    geom_line(data=comp_StO2_full,aes(y=0),size=0.5)+
    geom_line(color='black',size=1) +
    facet_wrap(~ pair) +
    theme_classic()+
    labs(x = 'Days', y = expression(paste('Difference in StO'[2] )))+
    scale_x_continuous(breaks=c(0,2,5,7,10))+
    theme(
        text=element_text(size=18),
        legend.title=element_blank()
    )


###for missing data
comp2<-smooth_diff(mod_m1,pdat,'Control','Treatment')
comp_StO2_missing <- cbind(Day = seq(0, 10, length = 400),
                   rbind(comp2))

missing_plot<-ggplot(comp_StO2_missing, aes(x = Day, y = diff, group =
   pair)) +
    geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +
    geom_line(color='black',size=1) +
    facet_wrap(~ pair) +
    labs(x = 'Days',
         y = expression(paste('Difference in StO'[2],'\n (missing data)'
                              )))+
  scale_x_continuous(breaks=c(0,2,5,7,10))+
  theme_classic()+
  theme(
     text=element_text(size=18),
     legend.title=element_blank()
     )

my_list<-pairwise_limits(comp_StO2_missing)

c2<-ggplot(comp_StO2_missing, aes(x = Day, y = diff, group = pair)) +
    annotate("rect",
             xmin =my_list$init1, xmax =my_list$final1,ymin=-Inf,ymax=Inf,
             fill='#30123BFF',
             alpha = 0.5,
    ) +
  annotate("text",
             x=1.5,
             y=-18,
             label="Control>Treatment",
           size=8
           )+
    annotate("rect",
             xmin =my_list$init2, xmax =my_list$final2,ymin=-Inf,ymax=Inf,
             fill='#7A0403FF',
             alpha = 0.5,
    ) +
  annotate("text",
             x=6,
             y=-18,
             label="Treatment>Control",
              size=8)+
    geom_ribbon(aes(ymin = lower, ymax = upper),
                alpha = 0.5,
                fill=rib_col) +
    geom_line(data=comp_StO2_missing,aes(y=0),size=0.5)+
    geom_line(color='black',size=1) +
    facet_wrap(~ pair) +
    theme_classic()+
    labs(x = 'Days', y = expression(paste('Difference in StO'[2] )))+
    scale_x_continuous(breaks=c(0,2,5,7,10))+
    theme(
        text=element_text(size=18),
        legend.title=element_blank()
    )

pair_comp<-c1+c2
```

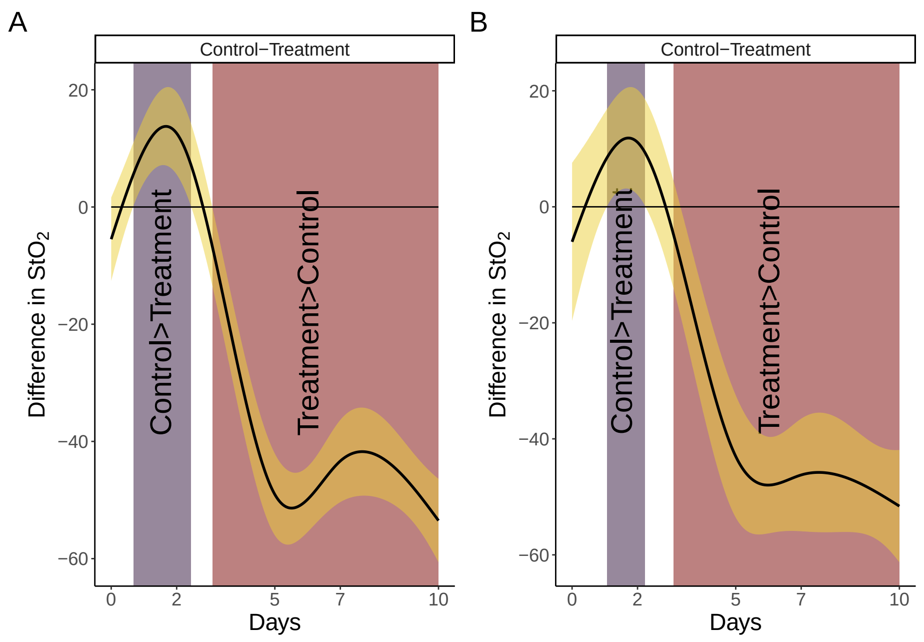


Figure A.9: Pairwise comparisons for smooth terms. A: Pairwise comparisons for the full dataset. B: Pairwise comparisons for the dataset with missing observations. Significant differences exist where the 95% empirical Bayesian credible interval does not cover 0. In both cases the effect of treatment is significant after day 3.