Modeling mean

For all the models discussed below assume the error to be normally distributed. Hence, checking the normality of the residuals is important, at least in the case of the semiparametric models.

1 Baseline Measurement

There are two ways to use the baseline data in the model: as a predictor or as a response itself. As response, there are two ways, 1) Calculate the changes from baseline and set baseline values to 0 for all the subjects, or 2) the baseline data as a response itself and do a joint model. As a predictor, we can use baseline data as a covariate. It often depends on the scientific question. It is good to write down what the specified model leads to in terms of interpretation. That would help to decide.

For example, in the lead dataset, we can consider the baseline data as a response since this dataset records continuous change over time. However, in the seizure counts dataset, the baseline records some sort of initial status of the patient which may be included as a predictor.

In general, the most appropriate approach depends on the study question. It is always important to think about the interpretation of the regression coefficients and decide which interpretation is correct from your point of view. **In randomized studies, the baseline numbers should not be statistically significant in the two groups. If their difference is statistically significant, we should adjust for the baseline in most cases.**

- Retain it as part of the outcome vector and make no assumptions about group differences in the mean response at baseline. $y_{i,j,g} = \alpha + \theta g + \beta_j + \gamma_j * g + \epsilon_{i,j}$ or $y_{i,j,g} = \alpha + \theta g + t_{i,j,g}\beta + \gamma t_{i,j,g} * g + \epsilon_{i,j}$
- Retain it as part of the outcome vector and assume the group means are equal at baseline, as might be appropriate in a randomized trial. $y_{i,j,g} = \alpha + \beta_j + \gamma_j * g + \epsilon_{i,j}$ or $y_{i,j,g} = \alpha + t_{i,j,g}\beta + \gamma t_{i,j,g} * g + \epsilon_{i,j}$
- Subtract the baseline response from all of the remaining post-baseline responses, and analyze the differences from baseline. $(y_{i,j,g} - y_{i,1,g}) = \beta_j + \gamma_j * g + \epsilon_{i,j}$ or $(y_{i,j,g} - y_{i,1,g}) = t_{i,j,g}\beta + \gamma t_{i,j,g} * g + \epsilon_{i,j}$
- Use baseline value as a covariate in the analysis of the post-baseline responses. $y_{i,j,g} = \alpha + \theta g + \beta_j + \gamma_j * g + \kappa y_{i,1,g} + \epsilon_{i,j}$ or $y_{i,j,g} = \alpha + \theta g + t_{i,j,g}\beta + \gamma t_{i,j,g} * g + \kappa y_{i,1,g} + \epsilon_{i,j}$

2 Nonlinear effect of time

Here we focus on the model $y_{i,j} = f(\mathbf{x}_{i,j}) + \epsilon_{i,j}$ for j-th observation for i-th subject for $j = 1, \ldots, n_i$. So far we have discussed the case, $f(\mathbf{x}_{i,j}) = \mathbf{x}_{i,j}\boldsymbol{\beta}$ i.e. the linear model. Now in many applications, we may only have the observation times, then $t_{i,j}$'s as the only predictors. In that case, the above model reduces to $y_{i,j} = f(t_{i,j}) + \epsilon_{i,j}$. The error model for $\epsilon_{i,j}$ can be based on what we discussed last time for gridded and non-gridded data. But here, we primarily focus on the models for 'f'. In earlier cases, we had $f(t_{i,j}) = \beta_0 + t_{i,j}\beta_1$.

In general, accommodation of nonlinear effect only requires writing the model like a linear model after some adjustment. Once we do that, we can use any standard package for fitting linear models for non-linear models.

2.1 Time as a factor

When the observation times are fixed for all the subjects, one may fit a model with time as a factor that estimates the effect of different observation times separately as,

$$y_{i,j} = \alpha + \beta_j + \epsilon_{i,j}.$$

For identifiability, we usually set the constraints $\beta_1 = 0$ or $\sum_{j=1}^{n} \beta_j$. R sets $\beta_1 = 0$ which is also called Dummy Coding. There is no particular time trend assumed in this model. Hence, these estimates can be used to test for any particular pattern that is of interest. By plotting the data, one may find distinctive patterns in the mean over time. To properly analyze such patterns, orthogonal polynomial contrast is used on β_j's. It is a form of trend analysis where it looks for the linear, quadratic, or cubic trends in the ordinal categorical variable (i.e. time or different levels of doses).

The most standard packages only allow the levels to be equally spaced. However, it is not necessary. Specifically, orthogonal polynomial contrasts will get us the $\boldsymbol{\ell}_i$'s which can be used to test $H_0 : \boldsymbol{\ell}_i^T\boldsymbol{\beta} = 0$ if i-th order polynomial effect is significant or not.

The levels of the ordinal factor will determine the contrast vectors (or coefficients). In the case of equally spaced levels, the contrasts are fixed for any problem.

```
data_new$Week <- as.factor(data_new$Week)
contr.poly(4)
contrasts(data_new$Week) = contr.poly(4)
summary(lm(measurement ~ Week, data=data_new))
```

R package `emeans` also has some ways to evaluate polynomial contrasts, but again for equally-spaced factors and requires output from `lme4::lmer`.

```
res <- lmer(measurement ~ factor(Week) + (1|Subject), data = data_new)
coef(summary(res))

library(emmeans)
library(ggplot2)
```

```
res.emm <- emmeans::emmeans(res, "Week")

# To check the polynomial contrasts, to see if there was a linear or quadratic change over time:
emmeans::contrast(res.emm, 'poly') %>%
  broom::tidy() %>%
  head(3)
```

More details on this package can be found in `https://cran.r-project.org/web/packages/emmeans/vignettes/comparisons.html`. There is another package called `contrast` (`https://cran.r-project.org/web/packages/contrast/vignettes/contrast.html`). But `emmeans` is probably more useful.

2.1.1 Contrast coding

There are several other contrasts, that are routinely used to analyze several types of behaviors over time. First see that under the above model, $E(y_{i,0}) = \alpha$ and $E(y_{i,j}) = \alpha + \beta_j$ for $j > 0$.

Several contrasts can be found at `https://stats.oarc.ucla.edu/r/library/r-library-contrast-coding-systems-for-categorical-variables/`. These are all for single-group analysis. For cross-group comparison, AUC-based contrasts are more useful.

In a two-group scenario, the model becomes,

$$y_{i,j,g} = \alpha + \theta g + \beta_j + \gamma_j * g + \epsilon_{i,j},$$

with $\beta_1 = \gamma_1 = 0$ for identifiability. For $g = 0$, $y_{i,j,0} = \alpha + \beta_j + \epsilon_{i,j}$, and $g = 1$, we have $y_{i,j,1} = (\alpha + \theta) + (\beta_j + \gamma_j) + \epsilon_{i,j}$.

Hence, to test the group effect on time, we need to test $H_0 : \gamma_j = 0$ for all $j = 2, 3, \ldots, n$ for n timepoints. Thus there are $n - 1$ parameters to be tested. This leads to making the test with $n-1$ degrees of freedom. In general, with G groups, this test will have $(G-1)(n-1)$ parameters to be tested and thus will be of $(G-1)(n-1)$ degrees of freedom. As the degrees of freedom or the number of parameters increases, the test will become less sensitive. Thus contrast-based tests are used.

Why constrast-based testing? Specifically, a contrast combines all the coefficients into one single coefficient and thus it has only one degree of freedom. Also, a contrast can be formulated based on the inference motivation.

The AUC (area under the curve) minus baseline contrast from note will not work for R output as R does not estimate group-specific effects. We have derived this contrast.

In a similar way, we can derive the contrast to test for equality of the difference between the average response at occasions 2 through n and the baseline value in the two groups.

There is no direct package function to run this contrast. Hence, we apply this contrast in the following way.

```
res1 <- lmer(measurement ~ factor(Week)*Treatment + (1|Subject), data = data_new)
```

```
res2 <- lme(measurement ~ factor(Week)*Treatment, random = ~1|Subject, data = data_new)

###############We can use either of the two formats
res <- res1
## set up contrast (linear comb. of coefficients)
ct <- c(0,0,0,0,0,1,1,1/2)   #This is the contrast for AUCMB_1-AUCMB_0

m <- sum(fixef(res) * ct)  ## mean of contrast
v <- t(ct) %*% vcov(res) %*% ct   ## variance of contrast
stder <- sqrt(as.numeric(v))      ## standard error
tstat <- m/stder      ## t statistic

##########For large sample##################
2*pnorm(abs(tstat), lower.tail=FALSE)
##############For small sample###############
error_df = nrow(data_new) - length(fixef(res))
2*pt(abs(tstat), df=error_df, lower.tail = FALSE)
```

2.2 Polynomial model

Once test for the trends, we may consider fitting models incorporating polynomial effects of time in the time. For example, if the above polynomial contrast-based approach tells us that the quadratic trend is significant. We can then fit the following model,

$$y_{i,j} = \beta_0 + \beta_1 t_{i,j} + \beta_2 t_{i,j}^2 + \epsilon_{i,j}.$$

Note that the above model can be fitted using linear model fitting softwares with predictors $\{t_{i,j}, t_{i,j}^2\}$.

```
library(nlme)
res <- lme(measurement ~ Week + Week^2, random = ~1|V1, data = data_new)
res.AR1 <- update(res, correlation = corAR1())
res.ARMA11 <- update(res, corr = corARMA(p = 1, q = 1))
```

2.3 Spline model

A more general model for nonlinear f can be implemented using splines. In the class of spline models, B-splines are the most popular basis functions. B-splines are controlled by knots, and they form basis functions with local support. From the theory of nonparametric function estimations under functional ridge penalty, cubic splines come as a natural solution with knots at the observation points. However, we often set the knots as some equidistant points in the domain for simplicity. For longitudinal analysis, observation times do not need to be equidistant or fixed to use spline models. However, in case the observation times are fixed, one may set the knots to be those points.

2.3.1 Cubic splines

Consider knot-points $t_{-2} = t_{-1} = t_0 = t_1 = A < t_2 < \cdots < B = t_M = t_{M+1} = t_{M+2} = t_{M+3}$, where $t_{1:M}$ are equidistant with $\delta = (t_2 - t_1)$. For $j = 1, 2, \ldots, (M+3)$, cubic B-splines $b_{3,j}$ are defined as

$$\mathrm{b}_{3,j}(x) = \begin{cases} \frac{1}{6\delta^3}\{(x - t_{j-2})\}^3 & \text{if } t_{j-2} \le x < t_{j-1}, \\ \frac{1}{6\delta^3}\{\delta^3 + 3(x - t_{j-1})\delta^2 + 3(x - t_{j-1})^2\delta - 3(x - t_{j-1})^3\} & \text{if } t_{j-1} \le x < t_j, \\ \frac{1}{6\delta^3}\{\delta^3 + 3(t_{j+1} - x)\delta^2 + 3(t_{j+1} - x)^2\delta - 3(t_{j+1} - x)^3\} & \text{if } t_j \le x < t_{j+1}, \\ \frac{1}{6\delta^3}\{(t_{j+2} - x)\}^3 & \text{if } t_{j+1} \le x < t_{j+2}, \\ 0 & \text{otherwise.} \end{cases}$$

Cubic B-splines are also called B-splines with order 4 (3+1). Hence, quadratic splines will be B-splines with order 3 (2+1), etc. How do we relate $f(x)$ with B-splines? - $f(x) = \sum_{j=1}^{M+3} \theta_j b_{3,j}(x)$. We, in general, need to select M via cross validation.

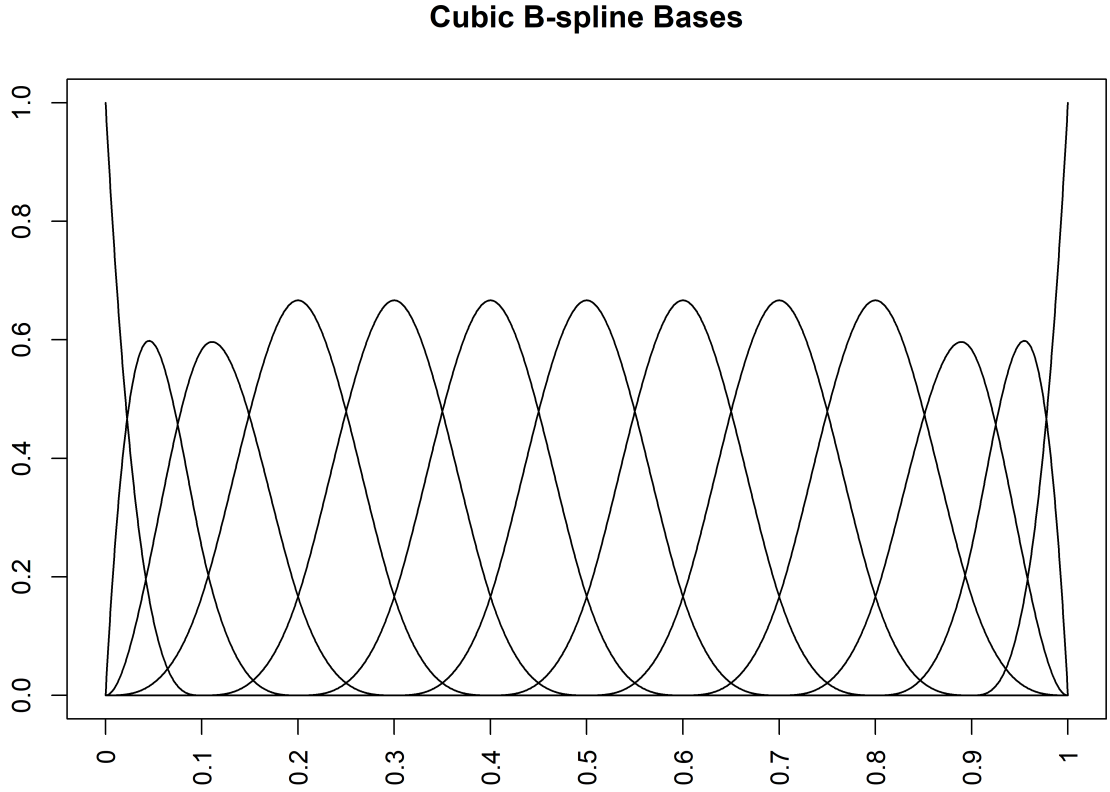

Figure 1: Plot of 13 cubic B-splines on $[0, 1]$ defined using 10 knot points that divide $[0, 1]$ into $M = 10$ equal sub-intervals.

The domain of the data needs to be pre-specified to use B-splines. Hence, it is important to rescale the time within $[0, 1]$.

```
library(nlme)
M = 4
data_new$Week = data_new$Week/max(data_new$Week)
X <- bs(data_new$Week, df=M, degree = 3)
res <- lme(measurement ~ X, random = ~1|V1, data = data_new)
res.AR1 <- update(res, correlation = corAR1())
res.ARMA11 <- update(res, corr = corARMA(p = 1, q = 1))
```

One can again use AIC or out-of-sample prediction to select M. R package `mgcv` can also be used. However, this package can handle even more general models as discussed next.

2.4 Visualization of spline bases

B-splines have been defined in different ways in the literature. Here, we can visualize the basis functions produced by two packages `fda` and `splines`.

```
vec <- (1:100)/100

X <- fda::bsplineS(vec, breaks=seq(0,1,length=10), norder = 3)

plot(vec, X[,1], type='l', col =1, ylim=range(X))
points(vec, X[,2], type='l', col =2)
points(vec, X[,3], type='l', col =3)

vec <- (1:100)/100

X <- splines::bs(vec, df=11, degree = 3)

plot(vec, X[,1], type='l', col =1, ylim=range(X))
points(vec, X[,2], type='l', col =2)
points(vec, X[,3], type='l', col =3)
```

2.5 Generalized (mixed) additive model

The above model can be further generalized to incorporate other predictors. Say $\mathbf{x}_{i,j} = (x_{i,j,1}, \ldots, x_{i,j,P})$ with one these P covariates is time. Here, $x_{i,j,p}$'s need to be in continuous scale for all p. A generalized additive model sets,

$$y_{i,j} = \sum_{p=1}^{P} f_p(x_{i,j,p}) + \epsilon_{i,j}$$

One can again consider polynomial bases or B-splines to model the f_p's. The error $\epsilon_{i,j}$ can again incorporate random effects and/or AR/ARMA or other covariance structures. The main difference between $\sum_{p=1}^{P} f_p(x_{i,j,p})$ and $f(\mathbf{x}_i)$ is that the latter model incorporates possible interactions among predictors as well.

R package `mgcv` is an excellent resource for fitting this class of models. It allows GAM models to have interactions and also factors as predictors such as treatment indicator or disease state. `https://r.qcbs.ca/workshop08/book-en/index.html` has some good tutorials on this package.

GAM also has mixed model extensions. `gamm` of `mgcv` is developed for this purpose.

Illustration model:

Say we want to build a nonlinear version of the following linear mixed model,

```
res <- lme(measurement ~ Week, random = ~1+Week|V1, data = data_new)
```

which fits $y_{i,j} = \beta_0 + t_{i,j}\beta_1 + \eta_{i,0} + t_{i,j}\eta_{i,1} + \delta_{i,j},$. Now a nonlinear extension will be $y_{i,j} = f(t_{i,j}) + g_i(t_{i,j}) + \delta_{i,j}$. If we model the unknown functions using cubic B-splines, we have $f(t) = \sum_{j=1}^{M+3} \theta_j b_{3,j}(t) = \mathbf{b}_t^T \boldsymbol{\theta}$ where $\mathbf{b}_t = \{b_{3,j}(t)\}_{j=1}^{M+3}$ and $\boldsymbol{\theta} = \{\theta_j\}_{j=1}^{M+3}$ and similarly, $g_i(t) = \sum_{j=1}^{M+3} \eta_{i,j} b_{3,j}(t) = \mathbf{b}_t^T \boldsymbol{\eta}_i$.

Hence, the nonlinear reduces to $y_{i,j} = \mathbf{b}_{t_{i,j}}^T \boldsymbol{\theta} + \mathbf{b}_{t_{i,j}}^T \boldsymbol{\eta}_i + \delta_{i,j}$. Hence, it gets a linear-model representation for a given M. One can set different M for f and g_i. $\mathbf{b}_{t_{i,j}}$ is called the basis vector evaluated at $t_{i,j}$. For simplicity, they are kept the same. One way to fit the above model as

```
library(splines)
M <- 3
X <- bs(data_new$Week, df=M, degree = 3)

####Multivariate error
##Generalized least square with unstructured covariance############
fm1 <- gls(measurement ~ X, data_new,
           correlation = corSymm(form = ~ 1 | Subject))

###Linear mixed model with random intercept
res <- lme(measurement ~ X, random = ~1|V1, data = data_new)

###Linear mixed model with random slope and intercept
res <- lme(measurement ~ X, random = ~1+Week|V1, data = data_new)

####Nonlinear random effect
res <- lme(measurement ~ X, random = ~X|V1, data = data_new)
```

Hence, it is possible to fit any nonlinear model using functions designed for linear models. Nonlinear models just expand the predictor space. Thus, several theoretical insights of nonlinear models often come from linear models. However, several parameters need to be tuned, like the

number of bases M. For that reason, `mgcv` is very useful as it does these selections automatically and often puts penalties to shrink the coefficients to avoid overfitting.

```
gam_model <- gamm(measurement ~ s(Week, k = 4) +
                    s(V1, bs = 're') +
                    s(V1, Week, bs = 're'),
                  data = data_new, method = 'REML')
```

Here `re` lets the function know that these are random effects.

Dr Simon Wood is the author of `mgcv` which is based `nlme` and he has also developed `gamm4` which is based on `lme4`. The notations of `gamm4` are similar to `lme4`. And `mgcv` shares similarities with `nlme` while fitting mixed models using `gamm`.

`https://reseau-mexico.fr/sites/reseau-mexico.fr/files/igam.pdf` is a book, written by Dr Woods on mixed modeling with special emphasis on GAM along with its implementations. The book is written with a lot of intuitions (Generalized Additive Models: an introduction with R).