Parametric mean curves

Alexander McLain

Contents

The datasets can be found at http://www.biostat.jhsph.edu/~fdominic/teaching/LDA/lda.html under “Data Sets”.

1 Example one: HIV data

First, let’s load the packages we’ll be using, read in the data and look at the variables.

```
library(nlme)
library(tidyverse)
HIV_data <- read.delim("hivstudy.txt", sep = "", header = FALSE)
names(HIV_data)
```

```
## [1] "V1" "V2" "V3" "V4"
```

```
names(HIV_data) <- c("ID", "Month", "CD4", "Group")
head(HIV_data)
```

```
##   ID Month CD4 Group
## 1  1     0 658     1
## 2  1     2 543     1
## 3  1     4 520     1
## 4  1     6 563     1
## 5  1     8 389     1
## 6  1    10 371     1
```

```
str(HIV_data)
```

```
## 'data.frame':    720 obs. of  4 variables:
##  $ ID   : int  1 1 1 1 1 1 2 2 2 2 ...
##  $ Month: int  0 2 4 6 8 10 0 2 4 6 ...
##  $ CD4  : int  658 543 520 563 389 371 500 419 431 285 ...
##  $ Group: int  1 1 1 1 1 1 1 1 1 1 ...
```

I’m going to change the `Group` variable so that it’s a `factor` not a numeric variable.

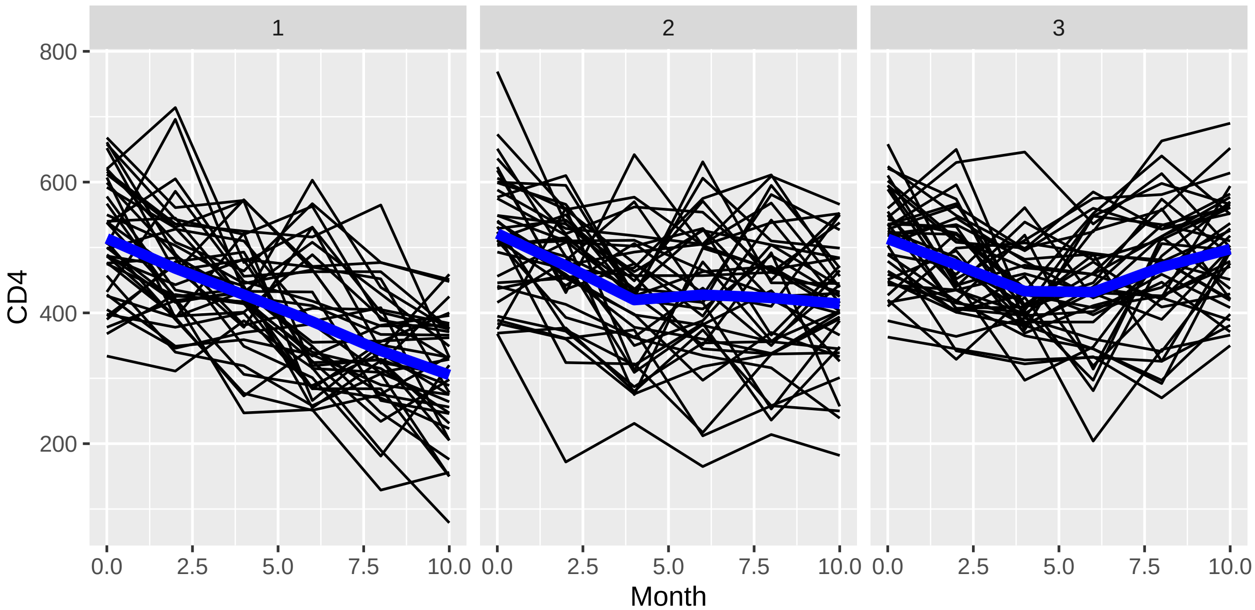
```
HIV_data <- HIV_data %>% mutate( Group = relevel( as.factor(Group) , "1") )
head( HIV_data$Group )
```

```
## [1] 1 1 1 1 1 1
```

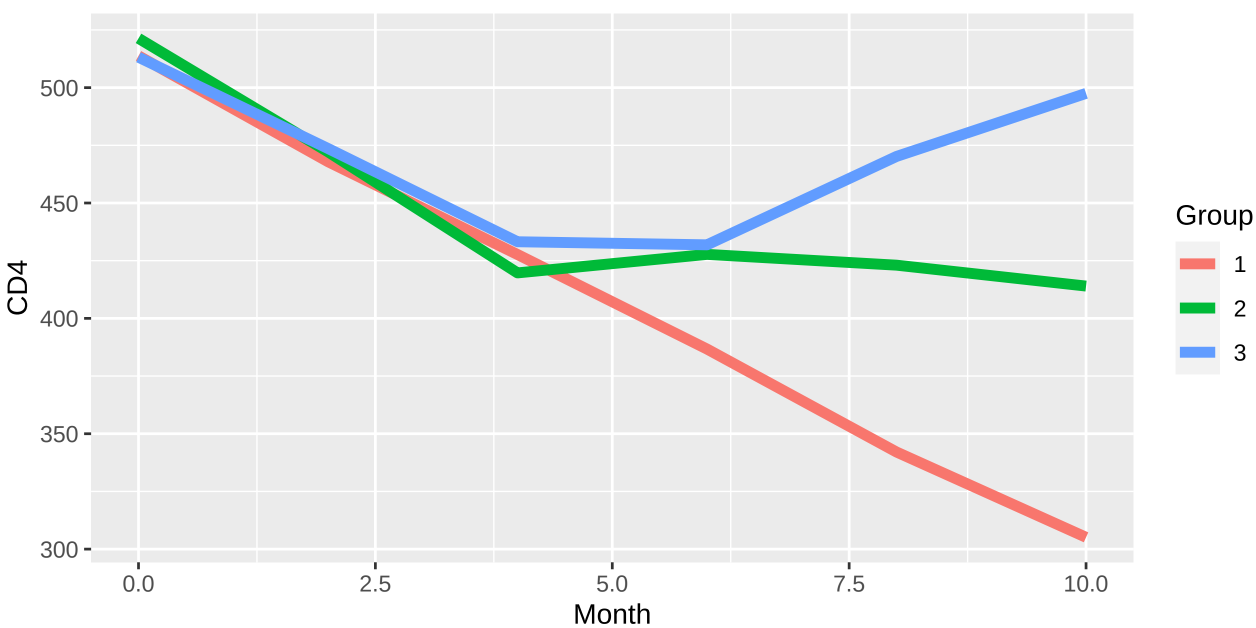
```
## Levels: 1 2 3
```

Now, let's get two plots of the data.

```r
## Spagetti plot by group with mean lines
p <- ggplot(data = HIV_data, aes(x = Month, y = CD4, group = ID))
p + geom_line() +
  stat_summary(aes(group = 1), geom = "line", fun = mean,
               color = "blue", size = 2) +
  facet_grid(. ~ Group)
```




```r
## Mean lines by group on the same plot.
p <- ggplot(data = HIV_data, aes(x = Month, y = CD4, group = ID))
p + stat_summary(aes(group = Group, color = Group), geom = "line",
                 fun = mean, size = 2)
```



It appears that there might not be any effect before 4 months. Then some effect after. Let's test this out. We'll also want to compare this with some other models. To do that, when we're comparing different mean models we'll use `method = "ML"`.

First, we'll create a spline variable.

```
HIV_data <- HIV_data %>% mutate( Month_4 = Month - 4 ) %>%
  mutate( Month_4 = replace(Month_4, Month_4 < 0, 0))
head(HIV_data)
```

```
##   ID Month CD4 Group Month_4
## 1  1     0 658     1       0
## 2  1     2 543     1       0
## 3  1     4 520     1       0
## 4  1     6 563     1       2
## 5  1     8 389     1       4
## 6  1    10 371     1       6
```

Now we can see if there is an effect of month after month 4.

```
formula_sp <- CD4 ~ Group + Month + Month_4 + Group*Month_4
cor_fun <- corCompSymm( form = ~ 1|ID )
### Fit the model
gls_sp <- gls( formula_sp, data = HIV_data , correlation = cor_fun,
               method = "ML")
```

We'll compare this to a profile analysis.

```
formula_pro <- CD4 ~ Group + as.factor(Month) + Group*as.factor(Month)
### Fit the model
gls_prof <- gls( formula_pro, data = HIV_data , correlation = cor_fun,
                 method = "ML")
```

And a quadratic analysis

```
HIV_data <- HIV_data %>% mutate( Month2 = (Month - 5)^2 )
formula_quad <- CD4 ~ Group + Month + Month2 + Group*Month + Group*Month2
### Fit the model
gls_quad <- gls( formula_quad, data = HIV_data , correlation = cor_fun,
                 method = "ML")
```

Now let's compare the models

```
anova(gls_sp, gls_prof, gls_quad)
```

```
##          Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## gls_sp       1  9 8212.918 8254.131 -4097.459
## gls_prof     2 20 8229.641 8321.226 -4094.820 1 vs 2 5.277154  0.9170
## gls_quad     3 11 8221.509 8271.881 -4099.755 2 vs 3 9.868473  0.3612
```

Now we'll refit the spline model using the standard `method = "REML"`.

```
gls_sp <- gls( formula_sp, data = HIV_data , correlation = cor_fun,
               method = "REML")
summary(gls_sp)
```

```
## Generalized least squares fit by REML
##   Model: formula_sp
##   Data: HIV_data
##        AIC      BIC    logLik
```

```
##   8181.371 8222.497 -4081.686
##
## Correlation Structure: Compound symmetry
##  Formula: ~1 | ID
##  Parameter estimate(s):
##       Rho
## 0.5367877
##
## Coefficients:
##                   Value Std.Error   t-value p-value
## (Intercept)    515.5966 12.223443  42.18096  0.0000
## Group2           2.2667 16.316339   0.13892  0.8896
## Group3          -0.0604 16.316339  -0.00370  0.9970
## Month          -22.7814  1.794427 -12.69562  0.0000
## Month_4          2.7572  2.992581   0.92134  0.3572
## Group2:Month_4  18.4313  2.394905   7.69603  0.0000
## Group3:Month_4  31.3906  2.394905  13.10725  0.0000
##
##  Correlation:
##                (Intr) Group2 Group3 Month  Mnth_4 G2:M_4
## Group2         -0.667
## Group3         -0.667  0.500
## Month          -0.330  0.000  0.000
## Month_4         0.116  0.117  0.117 -0.824
## Group2:Month_4  0.196 -0.294 -0.147  0.000 -0.400
## Group3:Month_4  0.196 -0.147 -0.294  0.000 -0.400  0.500
##
## Standardized residuals:
##         Min          Q1         Med          Q3         Max
## -3.37340049 -0.66645710  0.01721225  0.69782814  2.82112298
##
## Residual standard error: 89.02013
## Degrees of freedom: 720 total; 713 residual
```

2 Example two: Exercise data

First, let's load in the data and look at the variables.

```
Exer_data_wide <- read.delim("weightloss.txt", sep = "", header = FALSE)
names(Exer_data_wide)
```

```
## [1] "V1" "V2" "V3" "V4" "V5" "V6" "V7"
```

```
names(Exer_data_wide) <- c("ID", "Weight0", "Weight3", "Weight6", "Weight9", "Weight12",
                           "Program")
head(Exer_data_wide)
```

```
##   ID Weight0 Weight3 Weight6 Weight9 Weight12 Program
## 1  1   289.5   272.4   280.1   298.7    277.6       1
## 2  2   241.9   233.5   232.2   245.4    228.8       1
## 3  3   235.5   219.5   220.3   255.2    238.5       1
## 4  4   255.4   247.2   260.1   263.6    269.0       1
## 5  5   237.8   215.5   233.0   247.8    235.0       1
## 6  6   260.5   235.4   237.1   228.1    256.1       1
```

```
str(Exer_data_wide)
```

```
## 'data.frame':    100 obs. of  7 variables:
##  $ ID      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Weight0 : num  290 242 236 255 238 ...
##  $ Weight3 : num  272 234 220 247 216 ...
##  $ Weight6 : num  280 232 220 260 233 ...
##  $ Weight9 : num  299 245 255 264 248 ...
##  $ Weight12: num  278 229 238 269 235 ...
##  $ Program : int  1 1 1 1 1 1 1 1 1 1 ...
```

```
Exer_data_wide <- Exer_data_wide %>% mutate( Program = relevel( as.factor(Program) , "1") )
head( Exer_data_wide$Program )
```

```
## [1] 1 1 1 1 1 1
## Levels: 1 2 3
```

Now, we'll change this from wide to long.

```
Exer_data <- Exer_data_wide %>% pivot_longer(cols = starts_with("Weight"),
                                             names_to = "Month",
                                             names_prefix = "Weight",
                                             values_to = "Weight",
                                             values_drop_na = TRUE)
head( Exer_data )
```

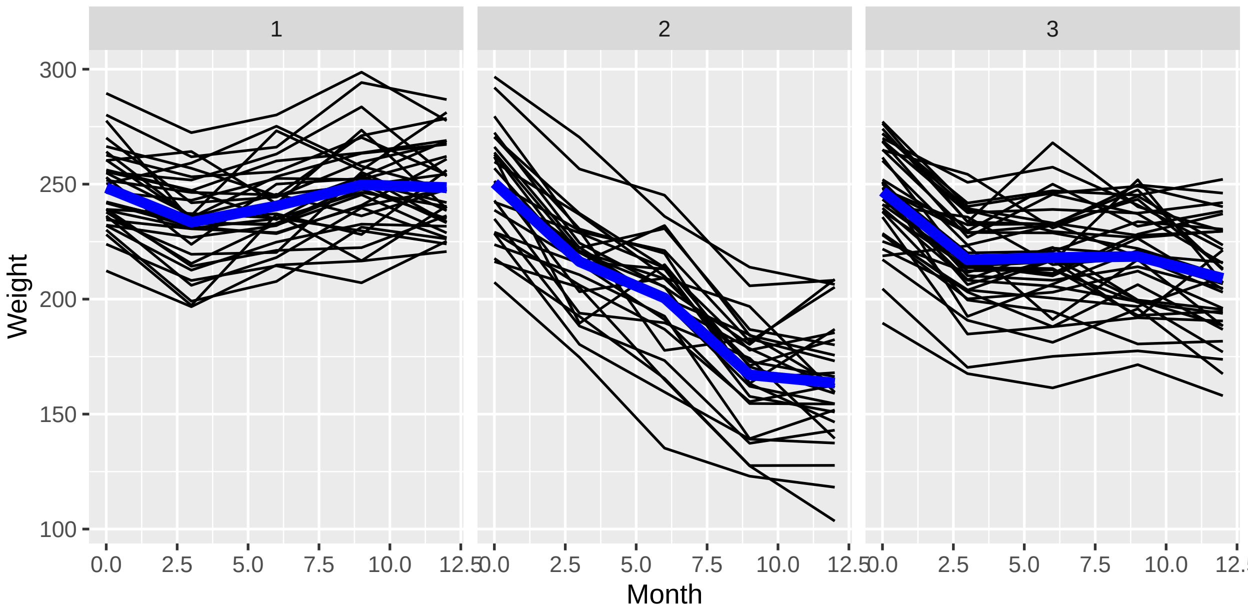
```
## # A tibble: 6 x 4
##      ID Program Month Weight
##   <int> <fct>   <chr>  <dbl>
## 1     1 1       0       290.
## 2     1 1       3       272.
## 3     1 1       6       280.
## 4     1 1       9       299.
## 5     1 1       12      278.
## 6     2 1       0       242.
```

```
Exer_data <- Exer_data %>% mutate( Month = as.numeric( Month ))
head( Exer_data )
```

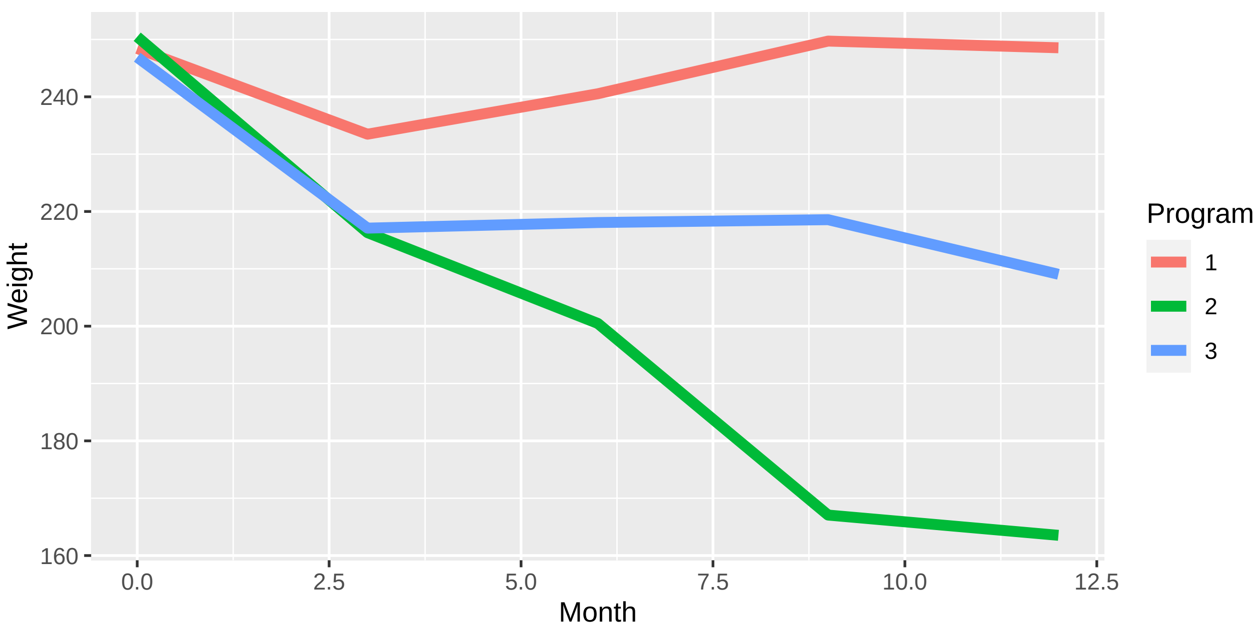
```
## # A tibble: 6 x 4
##      ID Program Month Weight
##   <int> <fct>   <dbl>  <dbl>
## 1     1 1           0   290.
## 2     1 1           3   272.
## 3     1 1           6   280.
## 4     1 1           9   299.
## 5     1 1          12   278.
## 6     2 1           0   242.
```

Now, let's get two plots of the data.

```
## Spagetti plot by group with mean lines
p <- ggplot(data = Exer_data, aes(x = Month, y = Weight, group = ID))
p + geom_line() +
  stat_summary(aes(group = 1), geom = "line", fun = mean,
               color = "blue", size = 2) +
  facet_grid(. ~ Program)
```




```
## Mean lines by group on the same plot.
p + stat_summary(aes(group = Program, color = Program), geom = "line",
                 fun = mean, size = 2)
```



It appears that a linear model may be appropriate here. To make things interesting, we'll compare it with a quadratic, log linear, square root and profile analysis models. Again, since we're comparing different mean models we'll use `method = "ML"`.

```
Exer_data <- Exer_data %>% mutate(Month2 = (Month - 6)^2,
                                  log_Month = log( Month + 1),
                                  sq_Month = sqrt(Month) )

#Linear
formula_linear <- Weight ~ Program + Month + Program*Month
gls_linear <- gls( formula_linear, data = Exer_data , correlation = cor_fun,
```

```r
               method = "ML")

#Quadratic
formula_quad <- Weight ~ Program + Month + Month2 + Program*Month + Program*Month2
gls_quad <- gls( formula_quad, data = Exer_data , correlation = cor_fun,
                      method = "ML")

#Log-linear
formula_log_linear <- Weight ~ Program + Month + log_Month +
                               Program*Month + Program*log_Month
gls_log_linear <- gls( formula_log_linear, data = Exer_data , correlation = cor_fun,
               method = "ML")

#Square-root
formula_sqrt <- Weight ~ Program + Month + sq_Month +
                         Program*Month + Program*sq_Month
gls_log_sqrt <- gls( formula_sqrt, data = Exer_data , correlation = cor_fun,
               method = "ML")

#Profile analysis
formula_profile <- Weight ~ Program + as.factor(Month) + Program*as.factor(Month)
gls_profile <- gls( formula_profile, data = Exer_data , correlation = cor_fun,
               method = "ML")
```

Now let's compare the models

```r
anova(gls_linear, gls_quad, gls_log_linear, gls_log_sqrt, gls_profile)
```

```
##                Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## gls_linear         1  8 4202.886 4236.603 -2093.443
## gls_quad           2 11 4135.680 4182.041 -2056.840 1 vs 2 73.20588  <.0001
## gls_log_linear     3 11 4094.930 4141.291 -2036.465
## gls_log_sqrt       4 11 4089.427 4135.787 -2033.713
## gls_profile        5 17 4027.610 4099.259 -1996.805 4 vs 5 73.81643  <.0001
```

Now we'll refit the profile model using the standard `method = "REML"`.

```r
gls_profile <- gls( formula_profile, data = Exer_data , correlation = cor_fun)
anova(gls_profile)
```

```
## Denom. DF: 485
##                          numDF   F-value p-value
## (Intercept)                  1 13004.735  <.0001
## Program                      2    39.935  <.0001
## as.factor(Month)             4   228.301  <.0001
## Program:as.factor(Month)     8   121.610  <.0001
```

There is an interaction. Now, to see what's going on we could look directly at the coefficients or use the `emmeans` package (see previous examples).

3 Example Three

The Six Cities Study of Air Pollution and Health example (see the first R notes for details).

```r
Six_cities <- read.csv("Six_cities.csv", header = TRUE)
head(Six_cities,8)
```

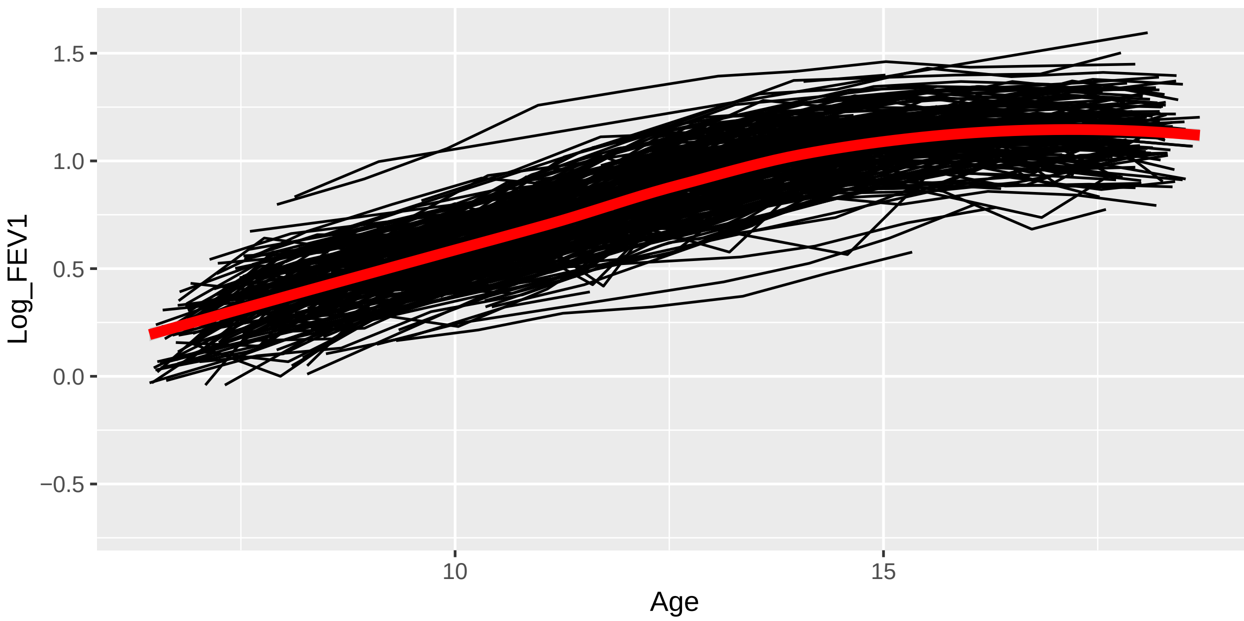
```
##   ID Height     Age INI_Height INI_Age Log_FEV1
## 1  1   1.20  9.3415       1.20  9.3415  0.21511
## 2  1   1.28 10.3929       1.20  9.3415  0.37156
## 3  1   1.33 11.4524       1.20  9.3415  0.48858
## 4  1   1.42 12.4600       1.20  9.3415  0.75142
## 5  1   1.48 13.4182       1.20  9.3415  0.83291
## 6  1   1.50 15.4743       1.20  9.3415  0.89200
## 7  1   1.52 16.3723       1.20  9.3415  0.87129
## 8  2   1.13  6.5873       1.13  6.5873  0.30748
```

Here, we don't have a treatment vs control type situation. We interested in the impact of age and height.

First, let's look at the data by Age and by Height.

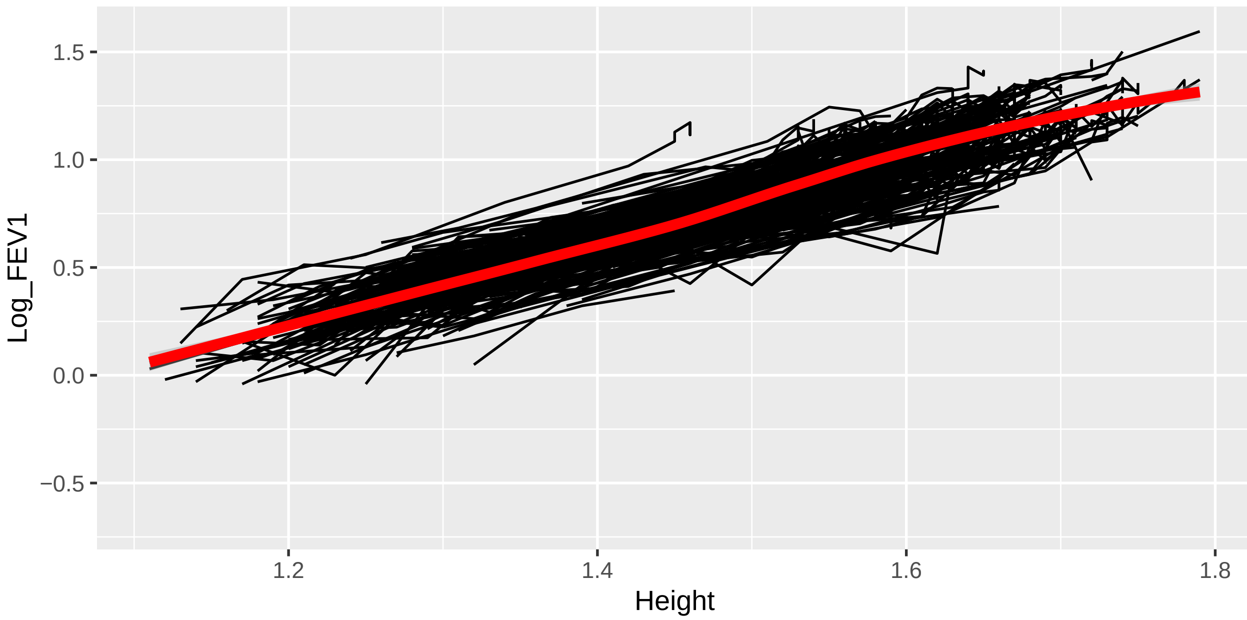
```
p <- ggplot(Six_cities, aes(x = Age, y = Log_FEV1, group = ID))
p + geom_line() + geom_smooth(aes(group = 1), method = "loess",
                              color = "red", size = 2)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
p <- ggplot(Six_cities, aes(x = Height, y = Log_FEV1, group = ID))
p + geom_line() + geom_smooth(aes(group = 1), method = "loess",
                              color = "red", size = 2)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Recall that this is unbalanced data, so our choices for correlation structures is limited. Here, we'll consider compound symmetric and exponential (with and without nugget effect). For the exponential structure we'll try age and log-transformed age. We cannot use height since some heights are equal which would imply a correlation = 1.

For the mean, we'll consider the log-transformed age variable (it's clear height has a linear impact).

```
Six_cities <- Six_cities %>% mutate( log_Age = log( Age ) ,
                                     INI_log_Age = log( INI_Age ) )
formula_six <- Log_FEV1 ~ Height + INI_log_Age
cor_fun <- corCompSymm(form = ~1|ID)
gls_CS <- gls( formula_six, data = Six_cities, correlation = cor_fun,
                      method = "REML")

cor_fun <- corExp(form = ~Age|ID, nugget = FALSE)
gls_exp_A <- gls( formula_six, data = Six_cities, correlation = cor_fun,
                      method = "REML")

cor_fun <- corExp(form = ~log_Age|ID, nugget = FALSE)
gls_exp_LA <- gls( formula_six, data = Six_cities, correlation = cor_fun,
                      method = "REML")

cor_fun <- corExp(form = ~Age|ID, nugget = TRUE)
gls_exp_A_nug <- gls( formula_six, data = Six_cities, correlation = cor_fun,
                      method = "REML")

cor_fun <- corExp(form = ~log_Age|ID, nugget = TRUE)
gls_exp_LA_nug <- gls( formula_six, data = Six_cities, correlation = cor_fun,
                      method = "REML")

anova(gls_CS, gls_exp_A, gls_exp_A_nug, gls_exp_LA, gls_exp_LA_nug)
```

```
##                 Model df       AIC       BIC   logLik   Test   L.Ratio p-value
## gls_CS              1  5 -4266.777 -4238.796 2138.389
## gls_exp_A           2  5 -4395.793 -4367.811 2202.897
```

```
## gls_exp_A_nug      3  6 -4515.192 -4481.614 2263.596 2 vs 3 121.39913  <.0001
## gls_exp_LA         4  5 -4435.261 -4407.279 2222.630 3 vs 4  81.93165  <.0001
## gls_exp_LA_nug     5  6 -4542.304 -4508.725 2277.152 4 vs 5 109.04285  <.0001
```

The best model appears to be the model with the expnential correlation structure as a function of log-transformed age with a nugget effect.

Now, let's consider some different mean models:

```r
cor_fun <- corExp(form = ~log_Age|ID, nugget = TRUE)

formula_six    <- Log_FEV1 ~ Height + INI_log_Age
gls_H_ILA <- gls( formula_six, data = Six_cities, correlation = cor_fun,
                      method = "ML")

formula_six    <- Log_FEV1 ~ Height + log_Age
gls_H_LA <- gls( formula_six, data = Six_cities, correlation = cor_fun,
                      method = "ML")

formula_six    <- Log_FEV1 ~ INI_Height + log_Age
gls_IH_LA <- gls( formula_six, data = Six_cities, correlation = cor_fun,
                      method = "ML")
anova(gls_H_ILA, gls_H_LA, gls_IH_LA)
```

```
##           Model df       AIC       BIC   logLik
## gls_H_ILA     1  6 -4561.133 -4527.546 2286.567
## gls_H_LA      2  6 -4659.311 -4625.724 2335.656
## gls_IH_LA     3  6 -3949.773 -3916.185 1980.886
```

Now, let's refit our final model using `method = "REML"` and get the estimates.

```r
cor_fun <- corExp(form = ~log_Age|ID, nugget = TRUE)

formula_six    <- Log_FEV1 ~ Height + log_Age
gls_H_LA <- gls( formula_six, data = Six_cities, correlation = cor_fun,
                      method = "REML")
summary(gls_H_LA)
```

```
## Generalized least squares fit by REML
##   Model: formula_six
##   Data: Six_cities
##         AIC       BIC   logLik
##   -4639.257 -4605.678 2325.628
##
## Correlation Structure: Exponential spatial correlation
##  Formula: ~log_Age | ID
##  Parameter estimate(s):
##     range    nugget
## 1.5094968 0.1422051
##
## Coefficients:
##                  Value  Std.Error   t-value p-value
## (Intercept) -2.1814997 0.02732472 -79.83613       0
## Height       1.5710146 0.04597970  34.16757       0
## log_Age      0.2582108 0.02456255  10.51238       0
##
##  Correlation:
```

```
##         (Intr) Height
## Height  -0.445
## log_Age  0.060 -0.916
##
## Standardized residuals:
##         Min          Q1         Med          Q3         Max
## -8.14302213 -0.60591086  0.05016381  0.65856746  2.95756375
##
## Residual standard error: 0.1227448
## Degrees of freedom: 1994 total; 1991 residual
```

Fitting smoothing splines is beyond the scope of this class, but if you're interested this can be done with mixed models (which we'll discuss later) using the `gamm` function (which stands for Generalized Additive Mixed Models) in the `mgcv` package.