Modeling response profiles

Alexander McLain

Contents

1 Modeling response profiles **1**

2 Changing the referent level **1**

3 Estimating least squared means **5**

4 Example Two **8**
4.1 Likelihood Ratio Test 11
4.2 Time effect 13

1 Modeling response profiles

First, let's load in the led data and turn it from wide to long.

```r
library(tidyverse)
wide_lead <- read.csv("wide_lead.csv", header = TRUE, na.strings = "",
                      stringsAsFactors = FALSE)
long_lead <- pivot_longer(wide_lead, cols = starts_with("PB"), names_to = "time",
                          names_prefix = "PB", values_to = "PB",
                          values_drop_na = TRUE)
head(long_lead,10)
```

```
## # A tibble: 10 x 4
##       ID TRT   time     PB
##    <int> <chr> <chr> <dbl>
##  1     1 P     1      30.8
##  2     1 P     2      26.9
##  3     1 P     3      25.8
##  4     1 P     4      23.8
##  5     2 A     1      26.5
##  6     2 A     2      14.8
##  7     2 A     3      19.5
##  8     2 A     4      21
##  9     3 A     1      25.8
## 10     3 A     2      23
```

Based on our previous modeling we're going to use an unstructed covariance matrix by TRT with the **gls** function.

```r
library(nlme)
```

```
##
## Attaching package: 'nlme'
## The following object is masked from 'package:dplyr':
##
##     collapse
```

2 Changing the referent level

So far we haven't done anything with reformatting variables. We've just put them into the model (no questions asked). Today, we'll talk more specifically about reformatting variables and how to use different ways in R.

First, recall what our variables are:

```
str(long_lead)
```

```
## tibble [400 x 4] (S3: tbl_df/tbl/data.frame)
##  $ ID  : int [1:400] 1 1 1 1 2 2 2 2 3 3 ...
##  $ TRT : chr [1:400] "P" "P" "P" "P" ...
##  $ time: chr [1:400] "1" "2" "3" "4" ...
##  $ PB  : num [1:400] 30.8 26.9 25.8 23.8 26.5 14.8 19.5 21 25.8 23 ...
```

The two variables we might want to reformat are **TRT** and **time**. To do this we'll turn them into factors

```
long_lead <- long_lead %>% mutate(TRT = as.factor(TRT), time = as.factor(time) )
str(long_lead)
```

```
## tibble [400 x 4] (S3: tbl_df/tbl/data.frame)
##  $ ID  : int [1:400] 1 1 1 1 2 2 2 2 3 3 ...
##  $ TRT : Factor w/ 2 levels "A","P": 2 2 2 2 1 1 1 1 1 1 ...
##  $ time: Factor w/ 4 levels "1","2","3","4": 1 2 3 4 1 2 3 4 1 2 ...
##  $ PB  : num [1:400] 30.8 26.9 25.8 23.8 26.5 14.8 19.5 21 25.8 23 ...
```

```
head(long_lead$TRT)
```

```
## [1] P P P P A A
## Levels: A P
```

It doesn't really matter, but let's say I want P to the the reference not A. This can be done using the **relevel** function.

```
long_lead <- long_lead %>% mutate(TRT = relevel(as.factor(TRT), ref = "P") )
head(long_lead$TRT)
```

```
## [1] P P P P A A
## Levels: P A
```

Let's check out the referent group for time.

```
head(long_lead$time)
```

```
## [1] 1 2 3 4 1 2
## Levels: 1 2 3 4
```

Having timepoint 1 as the referent is what we want, but we'll also consider other options below.

```
# Set the formula, correlation and variance
formula2<- PB ~ TRT + time + TRT*time
cor_fun <- corSymm(form = ~ 1|ID)
var_fun <- varIdent(form = ~ 1|time*TRT)
# Run the model
lm_t1 <- gls(model = formula2, data = long_lead, correlation = cor_fun,
```

```r
                    weights = var_fun)
summary(lm_t1)
```

```
## Generalized least squares fit by REML
##   Model: formula2
##   Data: long_lead
##        AIC      BIC    logLik
##   2386.174 2473.541 -1171.087
##
## Correlation Structure: General
##  Formula: ~1 | ID
##  Parameter estimate(s):
##  Correlation:
##   1     2     3
## 2 0.683
## 3 0.689 0.818
## 4 0.667 0.674 0.733
## Variance function:
##  Structure: Different standard deviations per stratum
##  Formula: ~1 | time * TRT
##  Parameter estimates:
##      1*P      2*P      3*P      4*P      1*A      2*A      3*A      4*A
## 1.000000 1.088081 1.127808 1.120538 1.391910 2.122834 2.245122 2.572223
##
## Coefficients:
##                Value Std.Error   t-value p-value
## (Intercept)   26.272 0.6184949  42.47731  0.0000
## TRTA           0.268 1.0600311   0.25282  0.8005
## time2         -1.612 0.5162843  -3.12231  0.0019
## time3         -2.202 0.5240495  -4.20189  0.0000
## time4         -2.626 0.5390985  -4.87109  0.0000
## TRTA:time2   -11.406 1.0893585 -10.47038  0.0000
## TRTA:time3    -8.824 1.1387475  -7.74886  0.0000
## TRTA:time4    -3.152 1.3169647  -2.39338  0.0172
##
##  Correlation:
##            (Intr) TRTA   time2  time3  time4  TRTA:2 TRTA:3
## TRTA       -0.583
## time2      -0.307  0.179
## time3      -0.263  0.154  0.682
## time4      -0.289  0.169  0.453  0.544
## TRTA:time2  0.146 -0.058 -0.474 -0.323 -0.215
## TRTA:time3  0.121 -0.002 -0.314 -0.460 -0.250  0.661
## TRTA:time4  0.118  0.055 -0.186 -0.223 -0.409  0.410  0.518
##
## Standardized residuals:
##        Min         Q1        Med         Q3        Max
## -2.0703660 -0.7307083 -0.1597097  0.5598924  4.0105434
##
## Residual standard error: 4.373419
## Degrees of freedom: 400 total; 392 residual
```

```r
anova(lm_t1)
```

```
## Denom. DF: 392
##             numDF   F-value p-value
## (Intercept)     1 2911.8966  <.0001
## TRT             1    2.7953  0.0953
## time            3   34.5528  <.0001
## TRT:time        3   39.2936  <.0001
```

Now let's switch the reference level to timepoint 2

```r
long_lead <- long_lead %>% mutate(time = relevel(as.factor(time), ref = "2"))
# Run the model
lm_t2 <- gls(model = formula2, data = long_lead, correlation = cor_fun,
                   weights = var_fun)
summary(lm_t2)
```

```
## Generalized least squares fit by REML
##   Model: formula2
##   Data: long_lead
##        AIC      BIC    logLik
##   2386.174 2473.541 -1171.087
##
## Correlation Structure: General
##  Formula: ~1 | ID
##  Parameter estimate(s):
##  Correlation:
##   1     2     3
## 2 0.683
## 3 0.689 0.818
## 4 0.667 0.674 0.733
## Variance function:
##  Structure: Different standard deviations per stratum
##  Formula: ~1 | time * TRT
##  Parameter estimates:
##      1*P      2*P      3*P      4*P      1*A      2*A      3*A      4*A
## 1.000000 1.088073 1.127802 1.120537 1.391919 2.122828 2.245125 2.572218
##
## Coefficients:
##                Value Std.Error  t-value p-value
## (Intercept)  24.660 0.6729715 36.64345  0.0000
## TRTA        -11.138 1.4753884 -7.54920  0.0000
## time1         1.612 0.5162788  3.12234  0.0019
## time3        -0.590 0.4146268 -1.42297  0.1555
## time4        -1.014 0.5521975 -1.83630  0.0671
## TRTA:time1   11.406 1.0893503 10.47046  0.0000
## TRTA:time3    2.582 0.9181442  2.81219  0.0052
## TRTA:time4    8.254 1.3213565  6.24661  0.0000
##
##  Correlation:
##            (Intr) TRTA   time1  time3  time4  TRTA:1 TRTA:3
## TRTA       -0.456
## time1      -0.485  0.221
## time3      -0.248  0.113  0.383
## time4      -0.373  0.170  0.493  0.478
## TRTA:time1  0.230 -0.697 -0.474 -0.181 -0.233
## TRTA:time3  0.112 -0.223 -0.173 -0.452 -0.216  0.366
## TRTA:time4  0.156 -0.234 -0.206 -0.200 -0.418  0.416  0.458
##
## Standardized residuals:
##        Min         Q1        Med         Q3        Max
## -2.0703552 -0.7306997 -0.1597099  0.5598886  4.0105225
##
## Residual standard error: 4.373446
## Degrees of freedom: 400 total; 392 residual
```

```r
anova(lm_t2)
```

```
## Denom. DF: 392
##             numDF   F-value p-value
## (Intercept)     1 2911.8522  <.0001
## TRT             1    2.7955  0.0953
## time            3   34.5533  <.0001
## TRT:time        3   39.2940  <.0001
```

Clearly, there is an interaction.

3 Estimating least squared means

To estimate the means at each treatment by timepoint we'll use the `emmeans` package. This package will take a fitted model (like the ones we produced above) and estimate the means for any combination of variables. First let's load in the package (install if you haven't already).

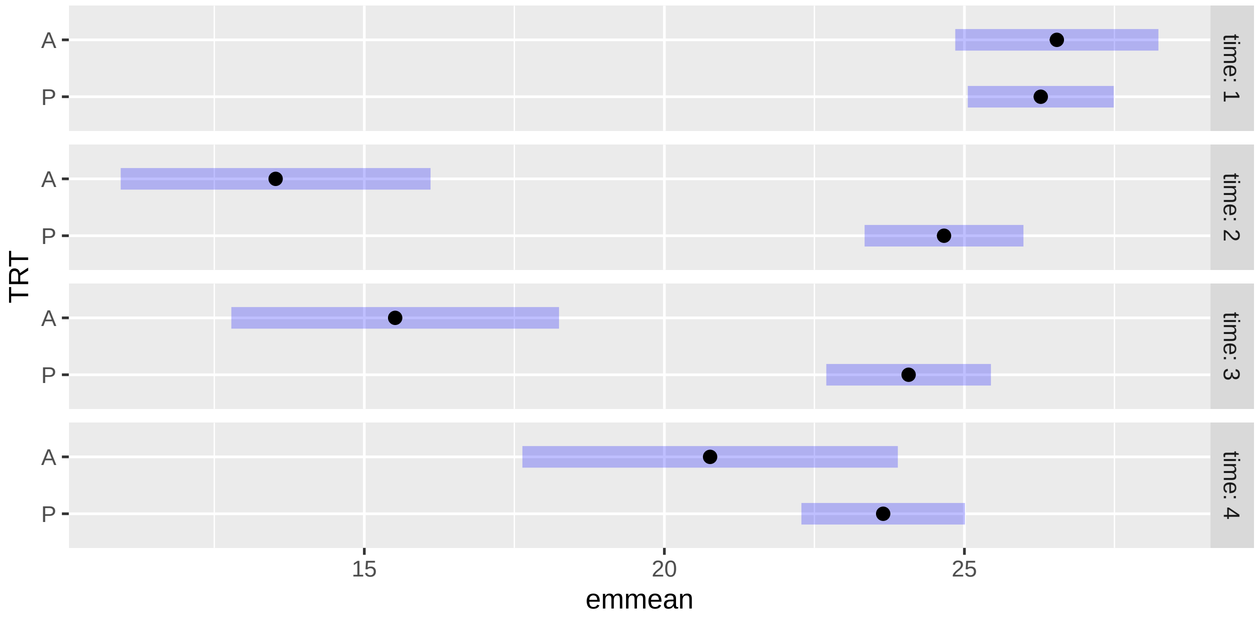
```r
library(emmeans)
?emmeans
args(emmeans)
```

```
## function (object, specs, by = NULL, fac.reduce = function(coefs) apply(coefs,
##     2, mean), contr, options = get_emm_option("emmeans"), weights,
##     offset, trend, ..., tran)
## NULL
```

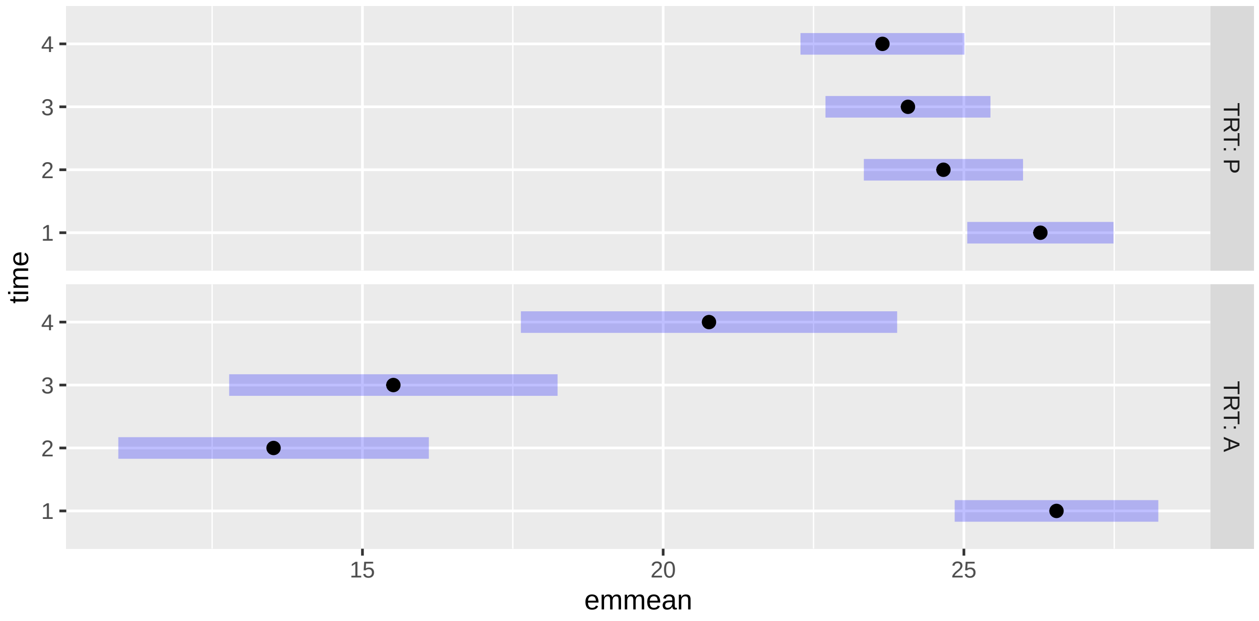
Also, I'm going to change back to having timepoint 1 as the referent (since this is more natural). So, i'm going to change the referent level back to timepoint 1 **and** i'm going to use the model where timepoint 1 was the referent level `lm_t1`.

Now we're going to estimate the means and plot them. I'll do this two different ways. The first way will plot them by time point. The second way will plot them by treatment.

```r
long_lead <- long_lead %>% mutate(time = relevel(as.factor(time), ref = "1"))
est_means_time <- emmeans(lm_t1, specs = ~TRT|time, mode = "df.error")
plot(est_means_time)
```

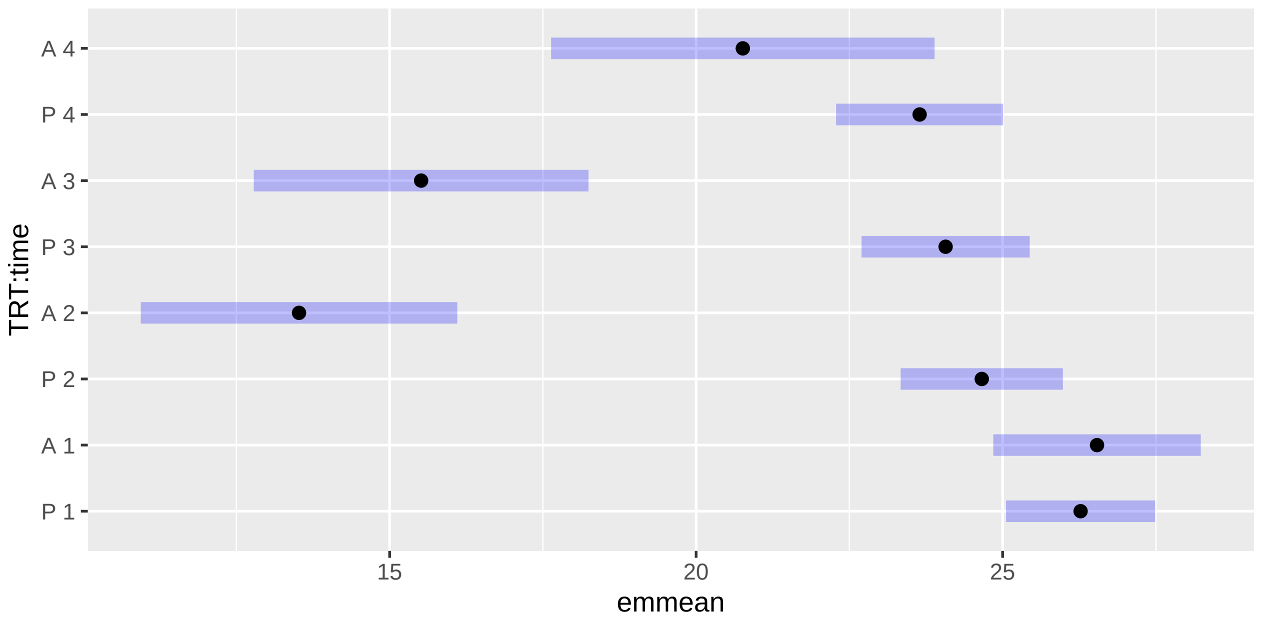


```
est_means_TRT <- emmeans(lm_t1, specs = ~time|TRT, mode = "df.error")
plot(est_means_TRT)
```



Besides plotting the means, we want to use this package to get some pairwise comparisons while controling for multiple comparisons. To do this we'll want to run the `emeans` function a little differently. Below we use `TRT*time` so it's not *by* either variable.

```
est_means <- emmeans(lm_t1, specs = ~TRT*time, mode = "df.error")
plot(est_means)
```



Now we'll use some other functions in the **emmeans** package to get some more summaries of the data.

First we'll look at all pairwise comparisons. Note the last line about using the "Tukey" method for multiple comparisons. This is the perfered method of controlling for multiple comparisons when doing all pairwise comparisons.

```
pairs(est_means)
```

```
##  contrast  estimate    SE  df t.ratio p.value
##  P 1 - A 1   -0.268 1.060 378 -0.253  1.0000
##  P 1 - P 2    1.612 0.516 378  3.122  0.0403
##  P 1 - A 2   12.750 1.451 378  8.785  <.0001
##  P 1 - P 3    2.202 0.524 378  4.202  0.0009
##  P 1 - A 3   10.758 1.520 378  7.077  <.0001
##  P 1 - P 4    2.626 0.539 378  4.871  <.0001
##  P 1 - A 4    5.510 1.707 378  3.228  0.0292
##  A 1 - P 2    1.880 1.093 378  1.720  0.6739
##  A 1 - A 2   13.018 0.959 378 13.571  <.0001
##  A 1 - P 3    2.470 1.108 378  2.229  0.3366
##  A 1 - A 3   11.026 1.011 378 10.906  <.0001
##  A 1 - P 4    2.894 1.105 378  2.619  0.1525
##  A 1 - A 4    5.778 1.202 378  4.809  0.0001
##  P 2 - A 2   11.138 1.475 378  7.549  <.0001
##  P 2 - P 3    0.590 0.415 378  1.423  0.8463
##  P 2 - A 3    9.146 1.543 378  5.927  <.0001
##  P 2 - P 4    1.014 0.552 378  1.836  0.5956
##  P 2 - A 4    3.898 1.727 378  2.257  0.3207
##  A 2 - P 3  -10.548 1.487 378 -7.095  <.0001
##  A 2 - A 3   -1.992 0.819 378 -2.432  0.2291
##  A 2 - P 4  -10.124 1.485 378 -6.819  <.0001
##  A 2 - A 4   -7.240 1.200 378 -6.031  <.0001
##  P 3 - A 3    8.556 1.554 378  5.506  <.0001
##  P 3 - P 4    0.424 0.508 378  0.835  0.9910
##  P 3 - A 4    3.308 1.737 378  1.904  0.5488
##  A 3 - P 4   -8.132 1.552 378 -5.240  <.0001
##  A 3 - A 4   -5.248 1.104 378 -4.752  0.0001
##  P 4 - A 4    2.884 1.735 378  1.662  0.7118
##
## Degrees-of-freedom method: df.error
## P value adjustment: tukey method for comparing a family of 8 estimates
```

Now we'll look at some contrasts of interest. The first will be give us the difference in differences (the same as the coefficients above) when comparing each value to the referent value, which here is 1. All of these use the `Sidak` procedure to control for multiple comparisons.

```r
Trt_v_cnt_ref_time <- contrast(est_means, interaction = c(TRT = "trt.vs.ctrl"),
                               adjust = "sidak")
Trt_v_cnt_ref_time
```

```
##  TRT_trt.vs.ctrl time_trt.vs.ctrl estimate   SE  df t.ratio p.value
##  A - P           2 - 1              -11.41 1.09 378 -10.470 <.0001
##  A - P           3 - 1               -8.82 1.14 378  -7.749 <.0001
##  A - P           4 - 1               -3.15 1.32 378  -2.393 0.0507
##
## Degrees-of-freedom method: df.error
## P value adjustment: sidak method for 3 tests
```

```r
Trt_v_cnt_pairwise_times <- contrast(est_means, interaction =
                                  c(TRT = "trt.vs.ctrl", time = "pairwise"),
                                adjust = "sidak")
Trt_v_cnt_pairwise_times
```

```
##  TRT_trt.vs.ctrl time_pairwise estimate    SE  df t.ratio p.value
##  A - P           1 - 2            11.41 1.089 378 10.470  <.0001
##  A - P           1 - 3             8.82 1.139 378  7.749  <.0001
##  A - P           1 - 4             3.15 1.317 378  2.393  0.0988
##  A - P           2 - 3            -2.58 0.918 378 -2.812  0.0307
##  A - P           2 - 4            -8.25 1.321 378 -6.247  <.0001
##  A - P           3 - 4            -5.67 1.216 378 -4.666  <.0001
##
## Degrees-of-freedom method: df.error
## P value adjustment: sidak method for 6 tests
```

Lastly, we'll get some confidence intervals for the last comparisons

```r
confint(Trt_v_cnt_pairwise_times)
```

```
##  TRT_trt.vs.ctrl time_pairwise estimate    SE  df lower.CL upper.CL
##  A - P           1 - 2            11.41 1.089 378    8.525   14.287
##  A - P           1 - 3             8.82 1.139 378    5.812   11.836
##  A - P           1 - 4             3.15 1.317 378   -0.331    6.635
##  A - P           2 - 3            -2.58 0.918 378   -5.010   -0.154
##  A - P           2 - 4            -8.25 1.321 378  -11.749   -4.759
##  A - P           3 - 4            -5.67 1.216 378   -8.887   -2.457
##
## Degrees-of-freedom method: df.error
## Confidence level used: 0.95
## Conf-level adjustment: sidak method for 6 estimates
```

There are lot of possibilities with the `emmeans` package.

4 Example Two

The data are from a study of exercise therapies, where 37 patients were assigned to one of two weightlifting programs. In the first program (treatment 1), the number of repetitions was increased as subjects became stronger. In the second program (treatment 2), the number of repetitions was fixed but the amount of weight was increased as subjects became stronger. Measures of strength were taken at baseline (day 0), and on days 2, 4, 6, 8, 10, and 12.

Variable List: ID, PROGRAM (1=Repetitions Increase, 2=Weights Increase), Response at Time 1, Response at Time 2, Response at Time 3, Response at Time 4, Response at Time 5, Response at Time 6, Response at Time 7.

First we'll load the data in.

```
wide_exer <- read.csv("Exercise_wide.csv", header = TRUE, na.strings = "",
                      stringsAsFactors = FALSE)
str(wide_exer)
```

```
## 'data.frame':    37 obs. of  9 variables:
##  $ ID  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Prog: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ S1  : int  79 83 81 81 80 76 81 77 84 74 ...
##  $ S2  : chr  "." "83" "83" "81" ...
##  $ S3  : chr  "79" "85" "82" "81" ...
##  $ S4  : chr  "80" "85" "82" "82" ...
##  $ S5  : chr  "80" "86" "83" "82" ...
##  $ S6  : chr  "78" "87" "83" "83" ...
##  $ S7  : chr  "80" "87" "82" "81" ...
```

```
wide_exer <- read.csv("Exercise_wide.csv", header = TRUE, na.strings = ".",
                      stringsAsFactors = FALSE)
str(wide_exer)
```

```
## 'data.frame':    37 obs. of  9 variables:
##  $ ID  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Prog: int  1 1 1 1 1 1 1 1 1 1 ...
##  $ S1  : int  79 83 81 81 80 76 81 77 84 74 ...
##  $ S2  : int  NA 83 83 81 81 76 84 78 85 75 ...
##  $ S3  : int  79 85 82 81 82 76 83 79 87 78 ...
##  $ S4  : int  80 85 82 82 82 76 83 79 89 78 ...
##  $ S5  : int  80 86 83 82 82 76 85 81 NA 79 ...
##  $ S6  : int  78 87 83 83 NA 76 85 82 NA 78 ...
##  $ S7  : int  80 87 82 81 86 75 85 81 86 78 ...
```

```
head(wide_exer)
```

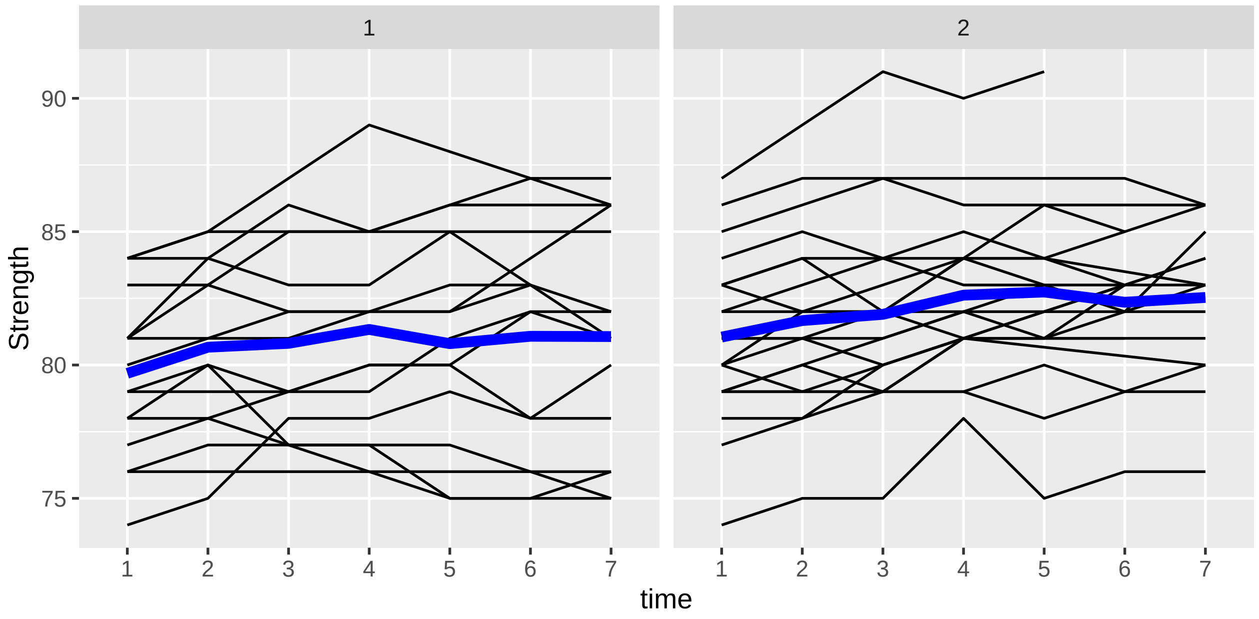
```
##   ID Prog S1 S2 S3 S4 S5 S6 S7
## 1  1    1 79 NA 79 80 80 78 80
## 2  2    1 83 83 85 85 86 87 87
## 3  3    1 81 83 82 82 83 83 82
## 4  4    1 81 81 81 82 82 83 81
## 5  5    1 80 81 82 82 82 NA 86
## 6  6    1 76 76 76 76 76 76 75
```

```
long_exer <- pivot_longer(wide_exer, cols = starts_with("S"), names_to = "time",
                          names_prefix = "S", values_to = "Strength",
                          values_drop_na = TRUE)
head(long_exer,10)
```

```
## # A tibble: 10 x 4
##       ID  Prog time  Strength
##    <int> <int> <chr>    <int>
##  1     1     1 1           79
##  2     1     1 3           79
##  3     1     1 4           80
##  4     1     1 5           80
##  5     1     1 6           78
##  6     1     1 7           80
##  7     2     1 1           83
##  8     2     1 2           83
##  9     2     1 3           85
## 10     2     1 4           85
```

Now, let's look at a plot of the data by group with mean lines.

```
p <- ggplot(data = long_exer, aes(x = time, y = Strength, group = ID))
p + geom_line() +
  stat_summary(aes(group = 1), geom = "line", fun = mean,
               color = "blue", size = 2) +
  facet_grid(. ~ Prog)
```



From the figure it appears that the homogeneous assumption should suffice. Later, we'll compare compound symmetric and unstructured correlation matrices. Please try others if you're interested.

First, I'm going to change Prog and time to factors. Then I'll set the formula and correlation (don't need to set the variance since we're using homogeneous).

```
long_exer <- long_exer %>% mutate( Prog = as.factor(Prog), time = as.factor(time) )
formula_exer <- Strength ~ Prog + time + Prog*time
cor_fun <- corCompSymm(form = ~ 1|ID)
# Run the model
lm_inter <- gls(model = formula_exer, data = long_exer, correlation = cor_fun)
# First look at the type III tests
anova(lm_inter)
```

```
## Denom. DF: 225
##             numDF   F-value p-value
## (Intercept)     1 22812.147  <.0001
## Prog            1     1.513  0.2200
## time            6    11.009  <.0001
## Prog:time       6     0.231  0.9661
```

There doesn't appear to be any interaction. Let's fit it without an interaction.

```
formula_exer_noint <- Strength ~ Prog + time
cor_fun <- corCompSymm(form = ~ 1|ID)
# Run the model
lm_no_inter <- gls(model = formula_exer_noint, data = long_exer, correlation = cor_fun)
# First look at the type III tests
anova(lm_no_inter)
```

```
## Denom. DF: 231
##             numDF   F-value p-value
## (Intercept)     1 22808.302  <.0001
## Prog            1     1.513    0.22
## time            6    11.276  <.0001
```

4.1 Likelihood Ratio Test

Now let's see what fits better compound symmetric or unstructured correlation matrices. To do this we'll run a **likelihood ratio test**. We'll discuss these formally later in the class.

```
cor_fun <- corSymm(form = ~ 1|ID)
# Run the model
lm_no_inter_symm <- gls(model = formula_exer_noint, data = long_exer, correlation = cor_fun)

# Look at the covariance matrices
getVarCov(lm_no_inter_symm)
```

```
## Marginal variance covariance matrix
##         [,1]    [,2]    [,3]    [,4]    [,5]    [,6]
## [1,] 11.2160 10.8540 10.3270  9.7030  9.4239  8.7603
## [2,] 10.8540 11.2160 10.4360  9.5210  9.5001  8.7236
## [3,] 10.3270 10.4360 11.2160 10.5980 10.5390  9.8375
## [4,]  9.7030  9.5210 10.5980 11.2160 10.4300  9.8658
## [5,]  9.4239  9.5001 10.5390 10.4300 11.2160 10.4820
## [6,]  8.7603  8.7236  9.8375  9.8658 10.4820 11.2160
##   Standard Deviations: 3.349 3.349 3.349 3.349 3.349 3.349
```

```
getVarCov(lm_no_inter)
```

```
## Marginal variance covariance matrix
##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
## [1,] 11.808 10.608 10.608 10.608 10.608 10.608
## [2,] 10.608 11.808 10.608 10.608 10.608 10.608
## [3,] 10.608 10.608 11.808 10.608 10.608 10.608
## [4,] 10.608 10.608 10.608 11.808 10.608 10.608
## [5,] 10.608 10.608 10.608 10.608 11.808 10.608
## [6,] 10.608 10.608 10.608 10.608 10.608 11.808
##   Standard Deviations: 3.4363 3.4363 3.4363 3.4363 3.4363 3.4363
```

```
# Now we'll test for differences (note: this also shows AIC and BIC).
anova(lm_no_inter,lm_no_inter_symm)
```

```
##                    Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## lm_no_inter            1 10 888.3146 922.7388 -434.1573
## lm_no_inter_symm       2 30 834.7948 938.0673 -387.3974 1 vs 2 93.51989  <.0001
```

A **significant** result indicates that that the **bigger** model (i.e., the one with larger degrees of freedom) should be used. Here, this means that we should use the unstructured correlation matrix.

Let's also check against an exponential covariance matrix with a nugget. To run this model, we have to create a **continuous** (or numeric) version of the time variable. We'll do that first and call it `time_c`.

```r
long_exer <- long_exer %>% mutate( time_c = as.numeric(time) )
cor_fun <- corExp(form = ~ time_c|ID, nugget = TRUE)
# Run the model
lm_no_inter_exp <- gls(model = formula_exer_noint, data = long_exer, correlation = cor_fun)

# Look at the covariance matrices.
getVarCov(lm_no_inter_exp)
```

```
## Marginal variance covariance matrix
##         [,1]    [,2]   [,3]    [,4]    [,5]    [,6]
## [1,] 11.6240 10.6590 10.284  9.9222  9.5732  9.2364
## [2,] 10.6590 11.6240 11.047 10.6590 10.2840  9.9222
## [3,] 10.2840 11.0470 11.624 11.0470 10.6590 10.2840
## [4,]  9.9222 10.6590 11.047 11.6240 11.0470 10.6590
## [5,]  9.5732 10.2840 10.659 11.0470 11.6240 11.0470
## [6,]  9.2364  9.9222 10.284 10.6590 11.0470 11.6240
##   Standard Deviations: 3.4094 3.4094 3.4094 3.4094 3.4094 3.4094
```

```r
getVarCov(lm_no_inter_symm)
```

```
## Marginal variance covariance matrix
##         [,1]    [,2]    [,3]    [,4]    [,5]    [,6]
## [1,] 11.2160 10.8540 10.3270  9.7030  9.4239  8.7603
## [2,] 10.8540 11.2160 10.4360  9.5210  9.5001  8.7236
## [3,] 10.3270 10.4360 11.2160 10.5980 10.5390  9.8375
## [4,]  9.7030  9.5210 10.5980 11.2160 10.4300  9.8658
## [5,]  9.4239  9.5001 10.5390 10.4300 11.2160 10.4820
## [6,]  8.7603  8.7236  9.8375  9.8658 10.4820 11.2160
##   Standard Deviations: 3.349 3.349 3.349 3.349 3.349 3.349
```

```r
# Now we'll test for differences (including the compound symmetric).
anova(lm_no_inter_exp,lm_no_inter_symm)
```

```
##                    Model df      AIC      BIC    logLik   Test  L.Ratio p-value
## lm_no_inter_exp        1 11 815.3404 853.2070 -396.6702
## lm_no_inter_symm       2 30 834.7948 938.0673 -387.3974 1 vs 2 18.54564  0.4863
```

Here, the result is **not significant**. This means we should choose the **smaller** model, which is the one with an exponential correlation structure. *(Note: you do `anova(lm_no_inter_exp,lm_no_inter_symm,lm_no_inter)` to get the model fit statistics of all 3 models at once, but the likelihood ratio tests will not always make sense.)*

Let's take a look at our final model.

```r
summary(lm_no_inter_exp)
```

```
## Generalized least squares fit by REML
##   Model: formula_exer_noint
##   Data: long_exer
##        AIC      BIC    logLik
##   815.3404 853.207 -396.6702
##
## Correlation Structure: Exponential spatial correlation
##  Formula: ~time_c | ID
##  Parameter estimate(s):
##       range      nugget
## 27.92711657  0.01496366
##
## Coefficients:
##                 Value Std.Error  t-value p-value
## (Intercept) 79.58964 0.8273655 96.19647  0.0000
## Prog2        1.53254 1.0722602  1.42926  0.1543
## time2        0.73702 0.1779619  4.14144  0.0000
## time3        1.00236 0.2295001  4.36757  0.0000
## time4        1.46771 0.2700665  5.43463  0.0000
## time5        1.54972 0.3062633  5.06010  0.0000
## time6        1.62815 0.3405533  4.78091  0.0000
## time7        1.79201 0.3704463  4.83743  0.0000
##
##  Correlation:
##       (Intr) Prog2  time2  time3  time4  time5  time6
## Prog2 -0.736
## time2 -0.106  0.000
## time3 -0.137  0.000  0.639
## time4 -0.162  0.000  0.535  0.754
## time5 -0.181  0.000  0.465  0.654  0.805
## time6 -0.198  0.002  0.413  0.580  0.713  0.827
## time7 -0.218  0.011  0.374  0.525  0.645  0.743  0.834
##
## Standardized residuals:
##         Min          Q1         Med          Q3         Max
## -2.25021510 -0.62531372 -0.03652747  0.59078599  2.60322698
##
## Residual standard error: 3.409408
## Degrees of freedom: 239 total; 231 residual
```

4.2 Time effect

Let's take a closer look at the `time` effect.

```
est_means <- emmeans(lm_no_inter, specs = ~time, mode = "df.error")
est_means
```

```
##  time emmean    SE  df lower.CL upper.CL
##  1      80.4 0.570 229     79.2     81.5
##  2      81.1 0.571 229     80.0     82.2
##  3      81.4 0.571 229     80.3     82.5
##  4      81.9 0.571 229     80.7     83.0
##  5      81.9 0.573 229     80.8     83.0
##  6      82.0 0.577 229     80.9     83.1
##  7      82.1 0.577 229     81.0     83.2
##
## Results are averaged over the levels of: Prog
## Degrees-of-freedom method: df.error
## Confidence level used: 0.95
```

```r
contrast(est_means, "trt.vs.ctrl")
```

```
##  contrast estimate    SE  df t.ratio p.value
##  2 - 1       0.726 0.257 229 2.826   0.0269
##  3 - 1       1.019 0.257 229 3.968   0.0006
##  4 - 1       1.485 0.257 229 5.784   <.0001
##  5 - 1       1.546 0.261 229 5.915   <.0001
##  6 - 1       1.636 0.272 229 6.023   <.0001
##  7 - 1       1.721 0.272 229 6.335   <.0001
##
## Results are averaged over the levels of: Prog
## Degrees-of-freedom method: df.error
## P value adjustment: dunnettx method for 6 tests
```

```r
contrast(est_means, "trt.vs.ctrl", ref = "7")
```

```
##  contrast estimate    SE  df t.ratio p.value
##  1 - 7     -1.7212 0.272 229 -6.335  <.0001
##  2 - 7     -0.9954 0.274 229 -3.636  0.0020
##  3 - 7     -0.7022 0.273 229 -2.572  0.0539
##  4 - 7     -0.2359 0.274 229 -0.862  0.8453
##  5 - 7     -0.1753 0.278 229 -0.631  0.9330
##  6 - 7     -0.0848 0.287 229 -0.295  0.9925
##
## Results are averaged over the levels of: Prog
## Degrees-of-freedom method: df.error
## P value adjustment: dunnettx method for 6 tests
```