# Example 6a: Modeling Change over Time Using Polynomial Trends
## *(complete data, syntax, and output available for STATA, R, and SAS electronically)*

These data for these example models come from Hoffman (2015) chapter 6. We will be examining change in response time (RT) in milliseconds over six practice sessions (balanced time) to a measure of processing speed in a sample of 101 older adults. We will examine the extent to which individual differences in change in RT can be described by polynomial slopes (current Example 6a), piecewise slopes (Example 6b), "truly" nonlinear exponential and latent basis models (Example 6c), and the use of log-transformed time to approximate a (truly nonlinear) exponential model (Example 6d).

## STATA Syntax for Data Import and Manipulation:

```
// Define working directory for file location
   cd "C:\Dropbox\24_PSQF6271\PSQF6271_Example6"

// Import Example 6 six-occasion long-format data from excel
   clear // clear memory in case a dataset is already open
   import excel "Example6_Data.xlsx", firstrow case(preserve) sheet("Example6") clear

// Center time predictor for polynomial time models (also need to make quadratic version)
   gen time=session-1
   gen timesq=time*time
   label variable time   "time: Session (0=1)"
   label variable timesq "timesq: Quadratic Session (0=1)"

// Create session dummy codes for testing means model absolute fit in REML
   gen s1=0
   gen s2=0
   gen s3=0
   gen s4=0
   gen s5=0
   gen s6=0
   replace s1=1 if session==1
   replace s2=1 if session==2
   replace s3=1 if session==3
   replace s4=1 if session==4
   replace s5=1 if session==5
   replace s6=1 if session==6

global LastTime = 5  // Save time value at last occasion for use later
```

## R Syntax for Data Import and Manipulation (after loading 4 custom functions and packages *readxl*, *TeachingDemos*, *ggplot2*, *nlme*, *multcomp*, *emmeans*, *lmerTest*, and *performance*):

```
# Set working directory (to import and export files to)
setwd("C:/Dropbox/24_PSQF6271/PSQF6271_Example6")

# Import Example 6 six-occasion long-format data from excel -- path = file name
Example6 = read_excel(path="Example6_Data.xlsx", sheet="Example6")
# Convert to data frame to use for analysis
Example6 = as.data.frame(Example6)
# Sort by person and occasion (needed for correct R or V matrix)
Example6 = Example6[order(Example6$PersonID, Example6$session), ]

# Center time predictor for polynomial time models
Example6$time=Example6$session-1

# Create session dummy codes for testing means model absolute fit in REML
Example6$s1=0
Example6$s2=0
Example6$s3=0
Example6$s4=0
Example6$s5=0
Example6$s6=0
```

```
Example6$s1[which(Example6$session==1)]=1
Example6$s2[which(Example6$session==2)]=1
Example6$s3[which(Example6$session==3)]=1
Example6$s4[which(Example6$session==4)]=1
Example6$s5[which(Example6$session==5)]=1
Example6$s6[which(Example6$session==6)]=1

# Save number of occasions per person for use later
Ntimes = 6
# Save total number of observations for use later
Ntotal = 606
```

---

## Model 0: Saturated Means, Unstructured Variance Model (the answer key best baseline model)

**This model provides the ANSWER KEY** for both the model for the means (via saturated means) and the model for the variance (via an unstructured **R** matrix of all possible variances and covariances). This model is only possible to estimate directly (without rounding occasions) in balanced data. The predicted outcome from the (saturated) fixed effects is given by: $\widehat{rt}_{ti} = \beta_0 + \beta_1(s2_{ti}) + \beta_2(s3_{ti}) + \beta_3(s4_{ti}) + \beta_4(s5_{ti}) + \beta_5(s6_{ti})$, in which the $s_{ti}$ predictors are binary indicators for each session, but the unstructured model for the variance cannot be easily summarized by scalar notation. Note that the variance and covariates estimates given in the unstructured R matrix differ slightly across programs.

```
display "STATA Ch 6: 0: Saturated Means, Unstructured Variance Model -- TOTAL ANSWER KEY"
mixed rt i.session, || PersonID: , noconstant reml nolog difficult  ///
      residuals(unstructured,t(session)) dfmethod(satterthwaite) dftable(pvalue)
display "-2LL = " e(ll)*-2      // Print -2LL for model
estat wcorrelation, covariance // R matrix
estat wcorrelation             // RCORR matrix
contrast i.session, small      // Omnibus F-test for session
margins  i.session             // Means per session
marginsplot, xdimension(session) name(predicted_session, replace)
graph export "STATA Saturated Means Plot.png", replace
margins  i.session, pwcompare(pveffects) df(100) // Mean diffs by session

print("R Ch 6: 0: Saturated Means, Unstructured Variance Model in GLS -- TOTAL ANSWER KEY")
SatUN = gls(data=Example6, method="REML", model=rt~1+factor(session),
            correlation=corSymm(form=~as.numeric(session)|PersonID), # Unstructured correlations
            weights=varIdent(form=~1|session))                       # Heterogeneous variances
print("Show results with -2LL but incorrect residual DDF")
-2*logLik(SatUN); summary(SatUN)

'log Lik.' 8229.788 (df=27) → -2LL for model

Coefficients:
                      Value Std.Error   t-value p-value  fixed effect
(Intercept)       1961.89337 54.679749 35.879707       0  beta0
factor(session)2 -146.72098 29.820968 -4.920061       0  beta1
factor(session)3 -211.85872 31.366023 -6.754402       0  beta2
factor(session)4 -244.09691 33.642973 -7.255509       0  beta3
factor(session)5 -254.71764 35.845925 -7.105902       0  beta4
factor(session)6 -289.75736 32.700282 -8.861005       0  beta5

print("Show R and RCORR matrices for first person")
SatUN_R = getVarCov(SatUN, individual="101", type="marginal"); SatUN_R
corMatrix(SatUN$modelStruct$corStruct)[[Ntimes]]

Marginal variance covariance matrix
       [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
[1,] 301980 235650 217990 202600 192150 195350
[2,] 235650 259140 230210 213220 202080 193260
[3,] 217990 230210 233360 205200 196910 188600
[4,] 202600 213220 205200 217540 193670 185320
[5,] 192150 202080 196910 193670 212090 187840
[6,] 195350 193260 188600 185320 187840 196730
```

This marginal unstructured **R matrix** (**R = V here**) allows all the variances and covariances to be estimated.

THIS IS THE PATTERN we are trying to duplicate with our **model for the variance**.

```
         [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
[1,] 1.0000000 0.8423879 0.8211591 0.7904659 0.7592441 0.8014915
[2,] 0.8423879 1.0000000 0.9361376 0.8980528 0.8619918 0.8559406
[3,] 0.8211591 0.9361376 1.0000000 0.9107555 0.8851084 0.880219 4
[4,] 0.7904659 0.8980528 0.9107555 1.0000000 0.9016388 0.8957986
[5,] 0.7592441 0.8619918 0.8851084 0.9016388 1.0000000 0.9195627
[6,] 0.8014915 0.8559406 0.8802194 0.8957986 0.9195627 1.0000000
```

```
print("F-test p-value using Satterthwaite DDF")
joint_tests(object=ref_grid(SatUN), adjust="none", mode="satterthwaite")
```

```
 model term df1   df2 F.ratio p.value
 session      5 98.84  16.720  <.0001
```

```
print("Wave means and pairwise mean differences with Satterthwaite DDF")
emmeans(ref_grid(SatUN), pairwise~session, adjust="none", mode="satterthwaite")
```

```
session emmean   SE    df lower.CL upper.CL  b=beta
      1   1962 54.7  98.2     1853     2070   b0
      2   1815 50.7 102.0     1715     1916   b0+b1
      3   1750 48.1 101.6     1655     1845   b0+b2
      4   1718 46.4 101.4     1626     1810   b0+b3
      5   1707 45.8 101.3     1616     1798   b0+b4
      6   1672 44.1 100.8     1585     1760   b0+b5
```

The **saturated means model** allows all session means to be estimated.

THIS IS THE PATTERN we are trying to reproduce with our **model for the means**.

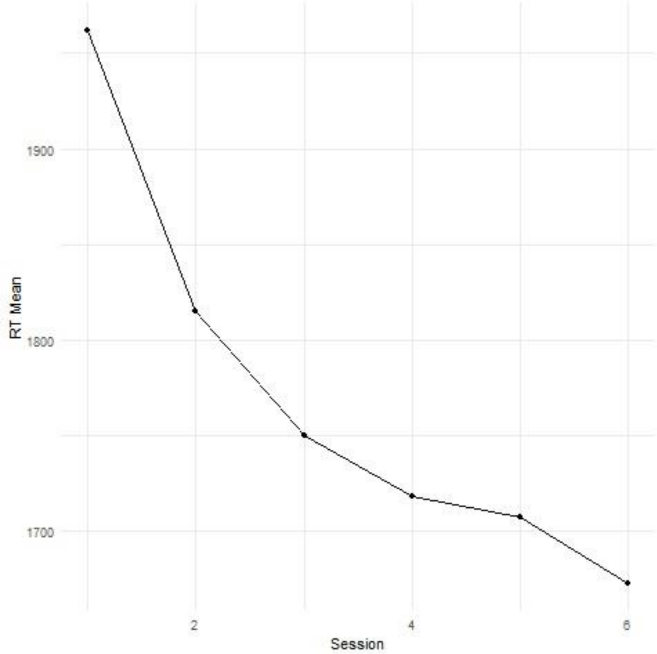
```
contrast            estimate   SE    df t.ratio p.value   b=beta
 session1 - session2    146.7 29.8  99.5   4.920  <.0001  (b0) - (b0+b1) = -b1
 session1 - session3    211.9 31.4 100.1   6.754  <.0001  (b0) - (b0+b2) = -b2
 session1 - session4    244.1 33.6  99.9   7.256  <.0001  (b0) - (b0+b3) = -b3
 session1 - session5    254.7 35.8 100.3   7.106  <.0001  (b0) - (b0+b4) = -b4
 session1 - session6    289.8 32.7  99.6   8.861  <.0001  (b0) - (b0+b5) = -b5
 session2 - session3     65.1 17.8 100.2   3.655  0.0004  (b0+b1) - (b0+b2) = b1 - b2
 session2 - session4     97.4 22.3  99.2   4.367  <.0001  (b0+b1) - (b0+b3) = b1 - b3
 session2 - session5    108.0 25.8 100.4   4.191  0.0001  (b0+b1) - (b0+b4) = b1 - b4
 session2 - session6    143.0 26.2  98.7   5.459  <.0001  (b0+b1) - (b0+b5) = b1 - b5
 session3 - session4     32.2 20.0  98.2   1.610  0.1106  (b0+b2) - (b0+b3) = b2 - b3
 session3 - session5     42.9 22.6  99.6   1.896  0.0609  (b0+b2) - (b0+b4) = b2 - b4
 session3 - session6     77.9 22.9  97.3   3.404  0.0010  (b0+b2) - (b0+b5) = b2 - b5
 session4 - session5     10.6 20.5  98.7   0.519  0.6049  (b0+b3) - (b0+b4) = b3 - b4
 session4 - session6     45.7 20.8  95.9   2.197  0.0304  (b0+b3) - (b0+b3) = b3 - b5
 session5 - session6     35.0 18.1  95.9   1.934  0.0560  (b0+b4) - (b0+b5) = b4 - b5
```

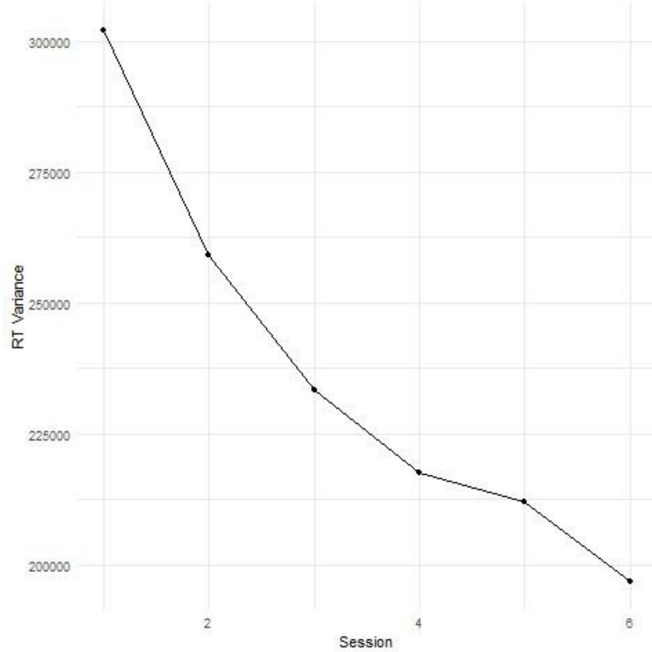
By what other name do you know this “total answer key” model (i.e., when it is estimated using least squares)?

By what other name do you know this “total answer key” model (i.e., when it is estimated in SEM)?

```
png(file="R Saturated Means Plot.png")  # open file
SatMeans = emmeans(object=SatUN, specs="session")  # Save saturated means to new object
SatMeans = as.data.frame(summary(SatMeans))        # Convert to data frame for plot
ggplot(data=SatMeans, aes(x=session, y=emmean)) + geom_line() + geom_point() + theme_minimal() +
labs(x="Session", y="RT Mean")
dev.off()  # close file

png(file="R Saturated Variances Plot.png")  # open file
SatVar = as.matrix(diag(SatUN_R)); SatVar = as.data.frame(SatVar) # Save saturated means to new
object
SatMeans = cbind(SatMeans, SatVar)                                # Concatenate to saturated means
to get wave
ggplot(data=SatMeans, aes(x=session, y=V1)) + geom_line() + geom_point() + theme_minimal() +
labs(x="Session", y="RT Variance")
dev.off()  # close file
```




---

## Model 1b: Empty Means, Random Intercept Model (the worst baseline longitudinal model)

Level 1: $y_{ti} = \beta_{0i} + e_{ti}$

Level 2: Intercept: $\beta_{0i} = \gamma_{00} + U_{0i}$

```
display "STATA Ch 6: 1b: Empty Means, Random Intercept Model"
mixed rt , || PersonID: , reml nolog  ///
      residuals(independent,t(session)) dfmethod(satterthwaite) dftable(pvalue)
matrix EmptyRI = r(table)       // Save results for computations below

-----------------------------------------------------------------
          rt |      Coef.   Std. Err.          DF        t    P>|t|
-------------+---------------------------------------------------
       _cons |   1770.701   45.42063        100.0     38.98   0.000
-----------------------------------------------------------------

------------------------------------------------------------------------------
  Random-effects Parameters  |   Estimate   Std. Err.     [95% Conf. Interval]
-----------------------------+------------------------------------------------
PersonID: Identity           |
                  var(_cons) |     200883   29471.23      150683.2    267806.8
-----------------------------+------------------------------------------------
               var(Residual) |   44899.96    2825.63      39689.76    50794.13
------------------------------------------------------------------------------
LR test vs. linear model: chibar2(01) = 691.74        Prob >= chibar2 = 0.0000

display "-2LL = " e(ll)*-2      // Print -2LL for model
-2LL = 8536.8609

estat icc                       // Intraclass correlation
Intraclass correlation
------------------------------------------------------------------------------
                       Level |        ICC   Std. Err.     [95% Conf. Interval]
-----------------------------+------------------------------------------------
                    PersonID |   .8173187   .0239727      .7655942    .8597208
------------------------------------------------------------------------------
matrix list EmptyRI             // Show saved results
```

```
EmptyRI[9,3]
                 rt:   lns1_1_1:    lnsig_e:
              _cons       _cons       _cons
     b   1770.7014    6.105239   5.3560961
    se   45.420625    .0733542   .03146583
     t   38.984523   83.229573   170.21942
pvalue   2.763e-62           0           0
    ll   1680.5882   5.9614674   5.2944242
    ul   1860.8147   6.2490106    5.417768
    df         100           .           .
  crit   1.9839715    1.959964    1.959964
 eform           0           0           0

// Variances are stored as log of SD instead
global EmptyResVar = exp(EmptyRI[1,3])^2 // Save as L1 residual variance
display $EmptyResVar                     // Show saved value to make sure it worked
44899.964

print("R Ch 6: 1b: Empty Means, Random Intercept Model")
EmptyRI = lmer(data=Example6, REML=TRUE, control=lmerControl(optimizer="Nelder_Mead"),
               formula=rt~1+(1|PersonID))
print("Show results with -2LL using Satterthwaite DDF")
llikAIC(EmptyRI, chkREML=FALSE); summary(EmptyRI, ddf="Satterthwaite")
print("Show unconditional ICC"); icc(EmptyRI)
print("LRT for random intercept variance"); ranova(EmptyRI, reduce.term=TRUE)
```

---

## Model 2a: Fixed Linear Time, Random Intercept Model

$$\text{Level 1: } y_{ti} = \beta_{0i} + \beta_{1i}\left(\text{Time}_{ti}\right) + e_{ti}$$

$$\text{Level 2: Intercept: } \beta_{0i} = \gamma_{00} + U_{0i}$$

$$\text{Linear Time: } \beta_{1i} = \gamma_{10}$$

```
display "STATA Ch 6: 2a: Fixed Linear Time, Random Intercept Model"
mixed rt c.time, || PersonID: , reml nolog  ///
      residuals(independent,t(session)) dfmethod(satterthwaite) dftable(pvalue)
estimates store FitFixLin     // Save for LRT
matrix FixLin = r(table)      // Save results for computations below
display "-2LL = " e(ll)*-2    // Print -2LL for model

// Get predicted mean per occasion from value of time predictor
lincom _cons*1 + c.time*0     // Intercept at Session=1 Time=0
lincom _cons*1 + c.time*1     // Intercept at Session=2 Time=1
lincom _cons*1 + c.time*2     // Intercept at Session=3 Time=2
lincom _cons*1 + c.time*3     // Intercept at Session=4 Time=3
lincom _cons*1 + c.time*4     // Intercept at Session=5 Time=4
lincom _cons*1 + c.time*5     // Intercept at Session=6 Time=5
margins, at(c.time=(0(1)$LastTime)) vsquish  // Intercept at each occasion (start(by)end)

matrix list FixLin                           // Show saved results
global FixLinResVar = exp(FixLin[1,4])^2     // Save as L1 residual variance for pseudo-R2
display "Pseudo-R2 for L1 Residual Variance = " 1-($FixLinResVar/$EmptyResVar)


print("R Ch 6 2a: Fixed Linear Time, Random Intercept Model")
FixLin = lmer(data=Example6, REML=TRUE, control=lmerControl(optimizer="Nelder_Mead"),
              formula=rt~1+time+(1|PersonID))
print("Show results with -2LL using Satterthwaite DDF")
llikAIC(FixLin, chkREML=FALSE); summary(FixLin, ddf="Satterthwaite")

      AIC       BIC    logLik  deviance  df.resid
 8422.688  8440.316 -4207.344  8414.688   602.000  → deviance = -2LL for model
```

```
Random effects:
 Groups   Name        Variance Std.Dev.
 PersonID (Intercept) 202423   449.9
 Residual              35662   188.8

Fixed effects:
            Estimate Std. Error       df t value Pr(>|t|)
(Intercept) 1899.631     46.788  112.515   40.60   <2e-16
time         -51.572      4.492  504.000  -11.48   <2e-16
```

```
print("Get predicted mean per occasion from value of time predictor")
print("Intercept at Session=1 Time=0"); contest1D(FixLin, ddf="Satterthwaite", L=c(1,0))
print("Intercept at Session=2 Time=1"); contest1D(FixLin, ddf="Satterthwaite", L=c(1,1))
print("Intercept at Session=3 Time=2"); contest1D(FixLin, ddf="Satterthwaite", L=c(1,2))
print("Intercept at Session=4 Time=3"); contest1D(FixLin, ddf="Satterthwaite", L=c(1,3))
print("Intercept at Session=5 Time=4"); contest1D(FixLin, ddf="Satterthwaite", L=c(1,4))
print("Intercept at Session=6 Time=5"); contest1D(FixLin, ddf="Satterthwaite", L=c(1,5))
print("Use custom function to get predicted means over time instead"); PredTimeMeans(FixLin)
```

```
[1] "Predicted outcome means using predict on fake cases"
  time      fit   se.fit
1    0 1899.631 46.78824
2    1 1848.059 45.91769
3    2 1796.487 45.47616
4    3 1744.916 45.47616
5    4 1693.344 45.91769
6    5 1641.772 46.78824
```

```
print("Psuedo-R2 for variance accounted for by fixed linear time")
pseudoRSquaredinator(smallerModel=EmptyRI, largerModel=FixLin)
```

```
Pseudo R2 Estimates
R2 Random.(Intercept):  -0.00767
R2 L1.Residual.Variance: 0.20575
```

---

## Model 2b: Random Linear Time Model

Level 1: $y_{ti} = \beta_{0i} + \beta_{1i}\left(\text{Time}_{ti}\right) + e_{ti}$

Level 2: Intercept: $\beta_{0i} = \gamma_{00} + U_{0i}$

Linear Time: $\beta_{1i} = \gamma_{10} + U_{1i}$

```
display "STATA Ch 6: 2b: Random Linear Time Model"
mixed rt c.time, || PersonID: time, reml nolog covariance(unstructured) ///
      residuals(independent,t(session)) dfmethod(satterthwaite) dftable(pvalue)
estimates store FitRandLin    // Save for LRT
matrix RandLin = r(table)     // Save results for computations below
```

```
------------------------------------------------------------------
          rt |      Coef.   Std. Err.           DF       t    P>|t|
-------------+----------------------------------------------------
        time |  -51.57185   6.156722         100.0    -8.38   0.000
       _cons |   1899.631    51.4998         100.0    36.89   0.000
------------------------------------------------------------------
------------------------------------------------------------------------------
  Random-effects Parameters  |   Estimate   Std. Err.     [95% Conf. Interval]
-----------------------------+------------------------------------------------
PersonID: Unstructured       |
                   var(time) |   2233.833   552.9239      1375.178    3628.626
                  var(_cons) |     253258   37897.26      188881.9    339575.3
             cov(time,_cons) |  -12700.79   3621.977     -19799.74   -5601.848
-----------------------------+------------------------------------------------
               var(Residual) |   27905.42   1963.419      24310.74    32031.62
------------------------------------------------------------------------------
LR test vs. linear model: chi2(3) = 830.20                Prob > chi2 = 0.0000
```

```
display "-2LL = " e(ll)*-2                          // Print -2LL for model
-2LL = 8372.1025

estat recovariance, relevel(PersonID) correlation  // GCORR matrix
             |      time      _cons
-------------+----------------------
        time |         1
       _cons | -.5339786          1

estat wcorrelation, covariance                      // V matrix
(printed in scientific notation; see R version below)

estat wcorrelation                                  // VCORR matrix
         obs |      1        2        3        4        5        6
-------------+-------------------------------------------------------
           1 |  1.000
           2 |  0.893    1.000
           3 |  0.878    0.884    1.000
           4 |  0.855    0.868    0.875    1.000
           5 |  0.823    0.843    0.859    0.868    1.000
           6 |  0.781    0.809    0.833    0.852    0.864    1.000
```

The marginal **VCORR** matrix now predicts that the correlation differs by occasion (in a specific-time-dependent pattern).

```
lrtest FitRandLin FitFixLin  // LRT for random linear time slope variance and covariance

Likelihood-ratio test                                 LR chi2(2)  =      42.59
(Assumption: FitFixLin nested in FitRandLin)          Prob > chi2 =     0.0000

matrix list RandLin                          // Show saved results
global RandLinResVar = exp(RandLin[1,6])^2   // Save as L1 residual variance for pseudo-R2

RandLin[9,6]
                rt:          rt:    lns1_1_1:    lns1_1_2:  atr1_1_1_2:     lnsig_e:
              time        _cons        _cons        _cons        _cons        _cons
     b  -51.571855    1899.6311     3.855737    6.2210821   -.59569429    5.1182881
    se   6.1567222      51.4998    .12376126    .07481946    .13617811    .03517988
     t  -8.3765117    36.886184    31.154635    83.147914   -4.3743762    145.48908
pvalue   3.497e-13    4.892e-60    4.39e-213            0    .00001218            0
    ll  -63.786616    1797.4569    3.6131694    6.0744386   -.86259848    5.0493368
    ul  -39.357094    2001.8052    4.0983046    6.3677255   -.32879009    5.1872394
    df         100          100            .            .            .            .
  crit   1.9839715    1.9839715     1.959964     1.959964     1.959964     1.959964
 eform           0            0            0            0            0            0

print("R Ch 6 2b: Random Linear Time Model")
RandLin = lmer(data=Example6, REML=TRUE, control=lmerControl(optimizer="Nelder_Mead"),
               formula=rt~1+time+(1+time|PersonID))
print("Show results with -2LL using Satterthwaite DDF")
llikAIC(RandLin, chkREML=FALSE); summary(RandLin, ddf="Satterthwaite")
print("LRT for random linear time slope variance & covariance"); ranova(RandLin, reduce.term=TRUE)
print("Use custom function to get predicted V matrix"); PrintV(RandLin)

Show First Block Diagonal of V matrix
         1        2        3        4        5        6
1 281163.5 240557.2 227856.3 215155.5 202454.6 189753.7
2 240557.2 257995.6 219623.1 209156.1 198689.1 188222.0
3 227856.3 219623.1 239295.4 203156.8 194923.6 186690.4
4 215155.5 209156.1 203156.8 225062.8 191158.0 185158.7
5 202454.6 198689.1 194923.6 191158.0 215298.0 183627.0
6 189753.7 188222.0 186690.4 185158.7 183627.0 210000.8
```

The marginal **V** matrix now predicts that **variance differs quadratically over time** (as a function of $\text{time}^2$).

## How the V matrix variances and covariances get created in a random linear time model:

$$\mathbf{V}_i \text{ matrix: Variance}\left[y_{time}\right] = \tau^2_{U_0} + \left[(\text{Session}-1)^2 \tau^2_{U_1}\right] + \left[2(\text{Session}-1)\tau_{U_{01}}\right] + \sigma^2_e$$

$$\mathbf{V}_i \text{ matrix: Covariance}\left[y_A, y_B\right] = \tau^2_{U_0} + \left[(A+B)\tau_{U_{01}}\right] + \left[(AB)\tau^2_{U_1}\right]$$

## Model 3a: Fixed Quadratic, Random Linear Time Model

Level 1: $y_{ti} = \beta_{0i} + \beta_{1i}\left(\text{Time}_{ti}\right) + \beta_{2i}\left(\text{Time}_{ti}\right)^2 + e_{ti}$

Level 2: Intercept: $\beta_{0i} = \gamma_{00} + U_{0i}$

Linear Time: $\beta_{1i} = \gamma_{10} + U_{1i}$

Quadratic Time: $\beta_{2i} = \gamma_{20}$

**Predicted intercept at any occasion:**

$= \gamma_{00} + \gamma_{10}(\text{Time}_{ti}) + \gamma_{20}(\text{Time}_{ti})^2$

**Instantaneous linear time slope at any occasion:**

$= \gamma_{10} + 2\gamma_{20}(\text{Time}_{ti})$

Because twice the quadratic slope is how the linear slope changes per unit time, the value for time used in predicting the model-implied linear slope per session gets multiplied by 2.

```
display "STATA Ch 6: 3a: Fixed Quadratic, Random Linear Time Model"
mixed rt c.time c.time#c.time, || PersonID: time, reml nolog covariance(unstructured) ///
      residuals(independent,t(session)) dfmethod(satterthwaite) dftable(pvalue)
estimates store FitFixQuad     // Save for LRT
matrix FixQuad = r(table)      // Save results for computations below
display "-2LL = " e(ll)*-2                         // Print -2LL for model
estat recovariance, relevel(PersonID) correlation // GCORR matrix

// Get predicted mean per occasion from values of time predictors
lincom _cons*1 + c.time*0 + c.time#c.time*0    // Intercept at Session=1 Time=0
lincom _cons*1 + c.time*1 + c.time#c.time*1    // Intercept at Session=2 Time=1
lincom _cons*1 + c.time*2 + c.time#c.time*4    // Intercept at Session=3 Time=2
lincom _cons*1 + c.time*3 + c.time#c.time*9    // Intercept at Session=4 Time=3
lincom _cons*1 + c.time*4 + c.time#c.time*16   // Intercept at Session=5 Time=4
lincom _cons*1 + c.time*5 + c.time#c.time*25   // Intercept at Session=6 Time=5
// Get predicted instantaneous linear slope per occasion from 2*value of time predictor
lincom c.time*1 + c.time#c.time*0,  small   // Linear Slope at Session=1 Time=0
lincom c.time*1 + c.time#c.time*2,  small   // Linear Slope at Session=2 Time=1
lincom c.time*1 + c.time#c.time*4,  small   // Linear Slope at Session=3 Time=2
lincom c.time*1 + c.time#c.time*6,  small   // Linear Slope at Session=4 Time=3
lincom c.time*1 + c.time#c.time*8,  small   // Linear Slope at Session=5 Time=4
lincom c.time*1 + c.time#c.time*10, small   // Linear Slope at Session=6 Time=5

matrix list FixQuad            // Show saved results
// Variances are stored as log of SD instead
global FixQuadResVar = exp(FixQuad[1,7])^2     // Save as L1 residual variance
display "Pseudo-R2 for L1 Residual Variance  = " 1-($FixQuadResVar/$RandLinResVar)
```

```
print("R Ch 6 3a: Fixed Quadratic, Random Linear Time Model")
FixQuad = lmer(data=Example6, REML=TRUE, control=lmerControl(optimizer="Nelder_Mead"),
               formula=rt~1+time+I(time^2)+(1+time|PersonID))
print("Show results with -2LL using Satterthwaite DDF")
llikAIC(FixQuad, chkREML=FALSE); summary(FixQuad, ddf="Satterthwaite")
```

```
      AIC       BIC    logLik  deviance  df.resid
 8355.477  8386.325 -4170.739  8341.477   599.000

Random effects:
 Groups   Name        Variance Std.Dev. Corr
 PersonID (Intercept) 254163   504.1
          time          2333    48.3    -0.53
 Residual              26176   161.8

Fixed effects:
            Estimate Std. Error      df t value Pr(>|t|)
(Intercept) 1945.850     52.243 105.879  37.246  < 2e-16
time        -120.900     14.541 501.796  -8.314 8.74e-16
I(time^2)     13.866      2.635 403.000   5.263 2.31e-07
```

```
print("Get predicted mean per occasion from values of time predictors")
print("Intercept at Session=1 Time=0"); contest1D(FixQuad, ddf="Satterthwaite", L=c(1,0,0))
print("Intercept at Session=2 Time=1"); contest1D(FixQuad, ddf="Satterthwaite", L=c(1,1,1))
print("Intercept at Session=3 Time=2"); contest1D(FixQuad, ddf="Satterthwaite", L=c(1,2,4))
print("Intercept at Session=4 Time=3"); contest1D(FixQuad, ddf="Satterthwaite", L=c(1,3,9))
print("Intercept at Session=5 Time=4"); contest1D(FixQuad, ddf="Satterthwaite", L=c(1,4,16))
print("Intercept at Session=6 Time=5"); contest1D(FixQuad, ddf="Satterthwaite", L=c(1,5,25))

print("Get predicted instantaneous linear slope per occasion from 2*value of time predictor")
print("Linear Slope at Session=1 Time=0"); contest1D(FixQuad, ddf="Satterthwaite", L=c(0,1,0))
print("Linear Slope at Session=2 Time=1"); contest1D(FixQuad, ddf="Satterthwaite", L=c(0,1,2))
print("Linear Slope at Session=3 Time=2"); contest1D(FixQuad, ddf="Satterthwaite", L=c(0,1,4))
print("Linear Slope at Session=4 Time=3"); contest1D(FixQuad, ddf="Satterthwaite", L=c(0,1,6))
print("Linear Slope at Session=5 Time=4"); contest1D(FixQuad, ddf="Satterthwaite", L=c(0,1,8))
print("Linear Slope at Session=6 Time=5"); contest1D(FixQuad, ddf="Satterthwaite", L=c(0,1,10))
```

**Estimates (from SAS for neater presentation)**

| Label | Estimate | Standard Error | DF | t Value | Pr > \|t\| | **g = gamma fixed effect** |
|---|---|---|---|---|---|---|
| Intercept at Session=1 Time=0 | 1945.85 | 52.2433 | 106 | 37.25 | <.0001 | g00(1)+ g10(0)+ g20(0) |
| Intercept at Session=2 Time=1 | 1838.82 | 48.6084 | 100 | 37.83 | <.0001 | g00(1)+ g10(1)+ g20(1) |
| Intercept at Session=3 Time=2 | 1759.51 | 46.8223 | 105 | 37.58 | <.0001 | g00(1)+ g10(2)+ g20(4) |
| Intercept at Session=4 Time=3 | 1707.94 | 45.2925 | 105 | 37.71 | <.0001 | g00(1)+ g10(3)+ g20(9) |
| Intercept at Session=5 Time=4 | 1684.10 | 44.0458 | 100 | 38.24 | <.0001 | g00(1)+ g10(4)+ g20(16) |
| Intercept at Session=6 Time=5 | 1687.99 | 44.9976 | 108 | 37.51 | <.0001 | g00(1)+ g10(5)+ g20(25) |
| Linear Slope at Session=1 Time=0 | -120.90 | 14.5415 | 502 | -8.31 | <.0001 | g10(1) + 2*g20(0) |
| Linear Slope at Session=2 Time=1 | -93.1687 | 10.0191 | 419 | -9.30 | <.0001 | g10(1) + 2*g20(2) |
| Linear Slope at Session=3 Time=2 | -65.4375 | 6.6968 | 139 | -9.77 | <.0001 | g10(1) + 2*g20(4) |
| Linear Slope at Session=4 Time=3 | -37.7062 | 6.6968 | 139 | -5.63 | <.0001 | g10(1) + 2*g20(6) |
| Linear Slope at Session=5 Time=4 | -9.9750 | 10.0191 | 419 | -1.00 | 0.3200 | g10(1) + 2*g20(8) |
| Linear Slope at Session=6 Time=5 | 17.7562 | 14.5415 | 502 | 1.22 | 0.2226 | g10(1) + 2*g20(10) |

```
print("Psuedo-R2 for variance accounted for by fixed quadratic time")
pseudoRSquaredinator(smallerModel=RandLin, largerModel=FixQuad)
```

```
Pseudo R2 Estimates
R2 Random.(Intercept): -0.00357
R2 Random.time: -0.04424
R2 L1.Residual.Variance: 0.06198
```

---

## Model 3b: Random Quadratic Time Model

Level 1: $y_{ti} = \beta_{0i} + \beta_{1i}(\text{Time}_{ti}) + \beta_{2i}(\text{Time}_{ti})^2 + e_{ti}$

Level 2: Intercept: $\beta_{0i} = \gamma_{00} + U_{0i}$

Linear Time: $\beta_{1i} = \gamma_{10} + U_{1i}$

Quadratic Time: $\beta_{2i} = \gamma_{20} + U_{2i}$

**Predicted intercept at any occasion:**

$= \gamma_{00} + \gamma_{10}(\text{Time}_{ti}) + \gamma_{20}(\text{Time}_{ti})^2$

**Instantaneous linear time slope at any occasion:**

$= \gamma_{10} + 2\gamma_{20}(\text{Time}_{ti})$

```
display "STATA Ch 6: 3b: Random Quadratic Time Model"
mixed rt c.time c.time#c.time, || PersonID: time timesq, ///
      reml nolog difficult covariance(unstructured)       ///
      residuals(independent,t(session)) dfmethod(satterthwaite) dftable(pvalue)
estimates store FitRandQuad     // Save for LRT
matrix RandQuad = r(table)      // Save results for computations below
```

```
------------------------------------------------------------------
           rt |      Coef.   Std. Err.          DF       t    P>|t|
--------------+---------------------------------------------------
         time |  -120.8999   20.04752        100.0   -6.03   0.000
c.time#c.time |   13.86561    3.41541        100.0    4.06   0.000
        _cons |    1945.85   53.84993        100.0   36.13   0.000
------------------------------------------------------------------
```

```
------------------------------------------------------------------------------
  Random-effects Parameters  |   Estimate   Std. Err.     [95% Conf. Interval]
-----------------------------+------------------------------------------------
PersonID: Unstructured       |
                   var(time) |   25839.79    5864.68      16561.42    40316.27
                 var(timesq) |   634.4658    172.375      372.5197    1080.605
                  var(_cons) |   276207.8   41445.49      205831.3    370646.8
            cov(time,timesq) |   -3903.29   982.6245     -5829.199   -1977.382
             cov(time,_cons) |  -35734.05   11947.78     -59151.26   -12316.83
           cov(timesq,_cons) |   3901.973   1950.274      79.50683     7724.44
-----------------------------+------------------------------------------------
               var(Residual) |   20298.19   1649.118      17310.19    23801.96
------------------------------------------------------------------------------
LR test vs. linear model: chi2(6) = 890.51                Prob > chi2 = 0.0000

display "-2LL = " e(ll)*-2                        // Print -2LL for model
-2LL = 8302.7457

estat recovariance, relevel(PersonID) correlation // GCORR matrix

Random-effects correlation matrix for level PersonID
             |      time     timesq      _cons
-------------+---------------------------------
        time |         1
      timesq | -.9640116          1
       _cons | -.4229799   .2947557          1

estat wcorrelation, covariance                    // V matrix

(printed in scientific notation; see R version below)

estat wcorrelation                                 // VCORR matrix
         obs |       1       2       3       4       5       6
-------------+------------------------------------------------
           1 |  1.000
           2 |  0.895   1.000
           3 |  0.833   0.900   1.000
           4 |  0.789   0.876   0.906   1.000
           5 |  0.781   0.864   0.894   0.902   1.000
           6 |  0.799   0.851   0.863   0.869   0.885   1.000
```

The marginal **VCORR** matrix now predicts that the correlation differs across occasions in a more complex time-dependent pattern than for random linear time.

```
lrtest FitRandQuad FitFixQuad   // LRT for random quadratic time slope variances and covariances

Likelihood-ratio test                                 LR chi2(3)  =     38.73
(Assumption: FitFixQuad nested in FitRandQuad)        Prob > chi2 =    0.0000

matrix list RandQuad                      // Show saved results
```

RandQuad[9,10]

| | rt: time | rt: c.time# c.time | rt: _cons | lns1_1_1: _cons | lns1_1_2: _cons | lns1_1_3: _cons | atr1_1_1_2: _cons | atr1_1_1_3: _cons | atr1_1_2_3: _cons | lnsig_e: _cons |
|---|---|---|---|---|---|---|---|---|---|---|
| b | -120.89992 | 13.865613 | 1945.8498 | 5.0798354 | 3.2263917 | 6.2644543 | -1.9997732 | -.45131573 | .30376654 | 4.9591434 |
| se | 20.047524 | 3.4154096 | 53.849926 | .11348157 | .13584264 | .07502594 | .18927919 | .12818523 | .13982446 | .04062229 |
| t | -6.0306661 | 4.059722 | 36.134679 | 44.76353 | 23.75095 | 83.497178 | -10.565203 | -3.5208091 | 2.1724849 | 122.07938 |
| pvalue | 2.761e-08 | .00009768 | 3.324e-59 | 0 | 1.07e-124 | 0 | 4.320e-26 | .00043023 | .02981911 | 0 |
| ll | -160.67364 | 7.0895381 | 1839.0131 | 4.8574157 | 2.960145 | 6.1174062 | -2.3707536 | -.70255416 | .02971562 | 4.8795252 |
| ul | -81.126206 | 20.641689 | 2052.6865 | 5.3022552 | 3.4926384 | 6.4115024 | -1.6287928 | -.20007729 | .57781745 | 5.0387617 |
| df | 100 | 100 | 100 | . | . | . | . | . | . | . |
| crit | 1.9839715 | 1.9839715 | 1.9839715 | 1.959964 | 1.959964 | 1.959964 | 1.959964 | 1.959964 | 1.959964 | 1.959964 |
| eform | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

```
// Save fixed effects and variances for computations below
global FixInt  = RandQuad[1,3]          // Save fixed intercept
global FixLin  = RandQuad[1,1]          // Save fixed linear time slope
global FixQuad = RandQuad[1,2]          // Save fixed quadratic time slope
global IntVar  = exp(RandQuad[1,6])^2   // Save L2 random intercept variance
global LinVar  = exp(RandQuad[1,4])^2   // Save L2 random linear time slope variance
global QuadVar = exp(RandQuad[1,5])^2   // Save L2 random quadratic time slope variance
```

```
// Check if correct saved variances
display $IntVar
display $LinVar
display $QuadVar
```

95% Random Effect Confidence Intervals that describe the *predicted* range of *individual* random effects:

$$\text{Random Effect 95\% CI} = \text{fixed effect} \pm \left(1.96*\sqrt{\text{Random Variance}}\right)$$

$$\text{Intercept 95\% CI} = \gamma_{00} \pm \left(1.96*\sqrt{\tau^2_{U_0}}\right) \rightarrow 1{,}945.9 \pm \left(1.96*\sqrt{276{,}209}\right) = 916 \text{ to } 2{,}976$$

$$\text{Linear Time Slope 95\% CI} = \gamma_{10} \pm \left(1.96*\sqrt{\tau^2_{U_1}}\right) \rightarrow -120.9 \pm \left(1.96*\sqrt{25{,}840}\right) = -436 \text{ to } 194$$

$$\text{Quadratic Time Slope 95\% CI} = \gamma_{20} \pm \left(1.96*\sqrt{\tau^2_{U_2}}\right) \rightarrow 13.9 \pm \left(1.96*\sqrt{634}\right) = -36 \text{ to } 63$$

```
display "STATA 95% Random Intercept CI"
display "Lower = " $FixInt - 1.96*sqrt($IntVar)
display "Upper = " $FixInt + 1.96*sqrt($IntVar)
Lower = 915.76255
Upper = 2975.937

display "STATA 95% Random Linear Time Slope CI"
display "Lower = " $FixLin - 1.96*sqrt($LinVar)
display "Upper = " $FixLin + 1.96*sqrt($LinVar)
Lower = -435.96522
Upper = 194.16537

display "STATA 95% Random Quadratic Time Slope CI"
display "Lower = " $FixQuad - 1.96*sqrt($QuadVar)
display "Upper = " $FixQuad + 1.96*sqrt($QuadVar)
Lower = -35.504052
Upper = 63.235279

print("R Ch 6 3b: Random Quadratic Time Model")
RandQuad = lmer(data=Example6, REML=TRUE, control=lmerControl(optimizer="Nelder_Mead"),
                formula=rt~1+time+I(time^2)+(1+time+I(time^2)|PersonID))
print("Show results with -2LL using Satterthwaite DDF")
llikAIC(RandQuad, chkREML=FALSE); summary(RandQuad, ddf="Satterthwaite")
print("LRT for random quadratic time slope variance and covariances")
ranova(RandQuad, reduce.term=TRUE)

print("Use custom function to get predicted V matrix"); PrintV(RandLin)

Show First Block Diagonal of V matrix
         1        2        3        4        5        6
1 296504.4 244374.6 220346.6 204122.5 195702.0 195085.4
2 244374.6 251508.7 219312.5 208680.7 199315.1 191215.6
3 220346.6 219312.5 235843.0 209043.5 199808.5 187840.0
4 204122.5 208680.7 209043.5 225508.9 197182.4 184958.6
5 195702.0 199315.1 199808.5 197182.4 211734.9 182571.3
6 195085.4 191215.6 187840.0 184958.6 182571.3 200976.4
```

The marginal **V** matrix now predicts the variances to differ across occasions in a **quartic pattern (time$^4$)**.

**How the V matrix variances and covariances get calculated in a random quadratic time model:**

Predicted Variance at Time *T*:

$$\text{Var}(y_T) = \sigma^2_e + \tau^2_{U_0} + 2*T*\tau_{U_{01}} + T^2*\tau^2_{U_1} + 2*T^2*\tau_{U_{02}} + 2*T^3*\tau_{U_{12}} + T^4*\tau^2_{U_2}$$

Predicted Covariance between Time *A* and *B*:

$$\text{Cov}(y_A, y_B) = \tau^2_{U_0} + (A+B)*\tau_{U_{01}} + (AB)*\tau^2_{U_1} + (A^2+B^2)*\tau_{U_{02}} + (AB^2)+(A^2B)*\tau_{U_{12}} + (A^2B^2)*\tau^2_{U_2}$$

```
# Get ingredients for 95% random effect confidence intervals
# Print each object first to see which row and column values to extract
as.data.frame(fixef(RandQuad)); as.data.frame(VarCorr(RandQuad))
# Save fixed effects and variances for computations below
FixInt  = as.data.frame(fixef(RandQuad))[1,1]    # Save fixed intercept
FixLin  = as.data.frame(fixef(RandQuad))[2,1]    # Save fixed linear time slope
FixQuad = as.data.frame(fixef(RandQuad))[3,1]    # Save fixed quadratic time slope
IntVar  = as.data.frame(VarCorr(RandQuad))[1,4] # Save L2 random intercept variance
LinVar  = as.data.frame(VarCorr(RandQuad))[2,4] # Save L2 random linear time slope variance
QuadVar = as.data.frame(VarCorr(RandQuad))[3,4] # Save L2 random quadratic time slope variance

print("R 95% Random Intercept Confidence Interval")
print("Lower = "); FixInt - 1.96*sqrt(IntVar)
print("Upper = "); FixInt + 1.96*sqrt(IntVar)
print("R 95% Random Linear Time Slope Confidence Interval")
print("Lower = "); FixLin - 1.96*sqrt(LinVar)
print("Upper = "); FixLin + 1.96*sqrt(LinVar)
print("R 95% Random Quadratic Time Slope Confidence Interval")
print("Lower = "); FixQuad - 1.96*sqrt(QuadVar)
print("Upper = "); FixQuad + 1.96*sqrt(QuadVar)
```

---

## Bonus: Is there any residual correlation left unmodeled in the level-1 R matrix?

```
display "STATA: Test AR1 Residual Correlation in Random Quadratic Time Model"
mixed rt c.time c.time#c.time, || PersonID: time timesq, ///
      reml nolog difficult covariance(unstructured)       ///
      residuals(ar1,t(session)) dfmethod(satterthwaite) dftable(pvalue)
estimates store FitRandQuadAR1      // Save for LRT
display "-2LL = " e(ll)*-2          // Print -2LL for model
lrtest FitRandQuadAR1 FitFixQuad    // LRT for AR1 residual correlation

print("R: Test AR1 Residual Correlation in Random Quadratic Time Model in LME")
RandQuadAR1 = lme(data=Example6, method="REML", fixed=rt~1+time+I(time^2),
                  random=~1+time+I(time^2)|PersonID,
                  correlation=(corAR1(form=~as.numeric(time)|PersonID)))
print("Show results using incorrect DDF"); summary(RandQuadAR1)
```

```
Random effects:
            StdDev   Corr
(Intercept) 521.6618 (Intr) time
time        155.1276 -0.433
I(time^2)    23.4018  0.323 -0.979
Residual    156.6327

Correlation Structure: AR(1)
      Phi
0.1710807

Fixed effects:  rt ~ 1 + time + I(time^2)
                Value Std.Error  DF  t-value p-value
(Intercept) 1948.7409  53.97380 503 36.10531  0.0000
time        -122.0833  20.51037 503 -5.95227  0.0000
I(time^2)     13.8734   3.47491 503  3.99246  0.0001
```

```
print("Show V matrix for first person")
getVarCov(RandQuadAR1, individual="101", type="marginal")
```

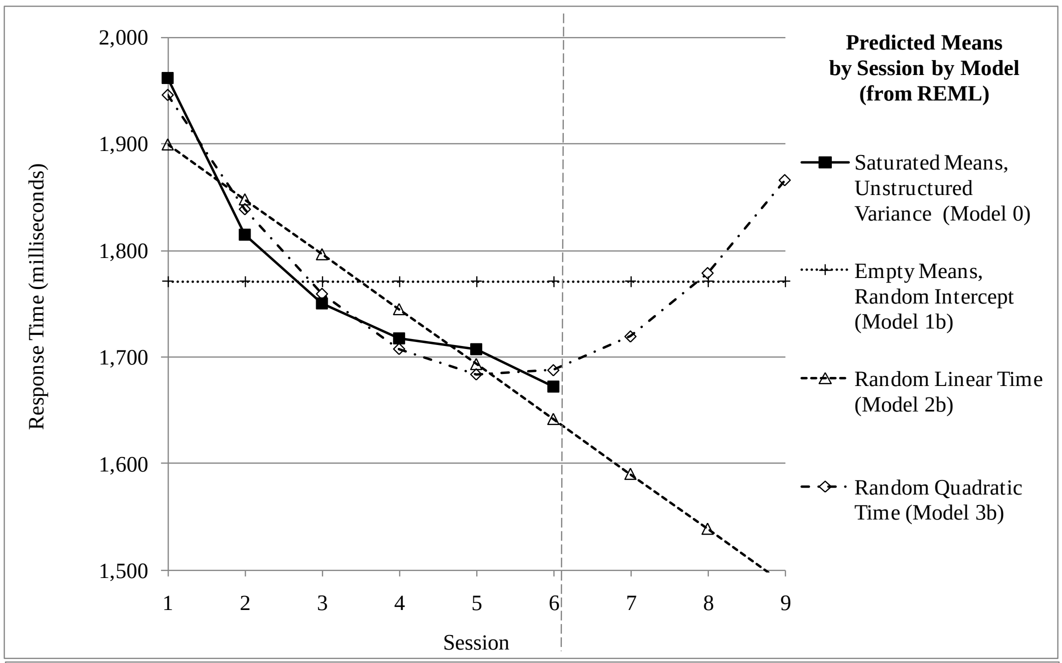
```
Marginal variance covariance matrix
       1      2      3      4      5      6
1 296660 245260 218610 202710 195200 195660
2 245260 252040 220010 206720 198170 191990
3 218610 220010 236330 210020 198610 188140
4 202710 206720 210020 226590 198910 184500
5 195200 198170 198610 198910 213040 183470
```

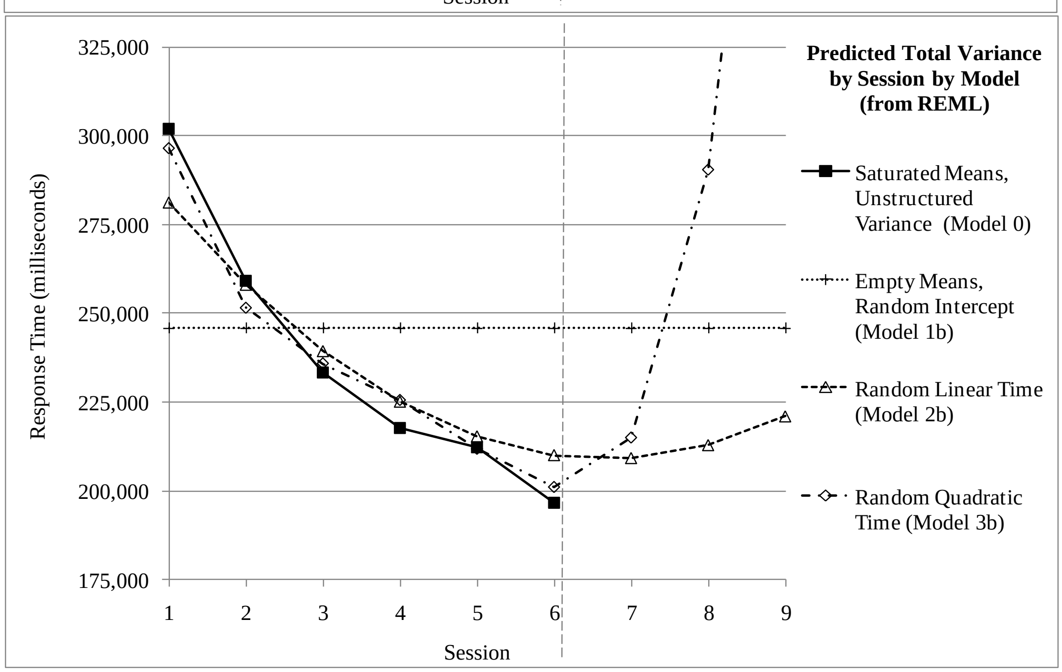
```
6 195660 191990 188140 184500 183470 199030

print("LRT for AR1 residual correlation in R matrix")
# Have to re-run random quadratic time model using lme to get LRT
RandQuadlme = lme(data=Example6, method="REML", rt~1+time+I(time^2),
                  random=~1+time+I(time^2)|PersonID)
anova(RandQuadAR1,RandQuadlme) # anova does LRT using LME versions

            Model df      AIC      BIC    logLik   Test  L.Ratio p-value
RandQuadAR1     1 11 8322.670 8371.091 -4150.335
RandQuadlme     2 10 8322.746 8366.765 -4151.373 1 vs 2 2.075513  0.1497
```

**Given that the AR1 R matrix did not improve model fit, I'd say this is as good as it gets for random quadratic time. So how did we do? Let's visually compare model predictions in terms of per-occasion means (top) and variances (bottom)…**



## Bonus Material: Testing Absolute Fit of Each Side of the Model When Using REML

As shown as Model 0, the saturated means, unstructured variance model is the best-fitting model for each side (means and variances). However, when using REML, we cannot do a model comparison against our random quadratic model, because models cannot differ in their fixed effects for the −2LL (LRT) to be valid. Instead, we can test the absolute fit for each side of the model separately as shown next (This will be included in the second edition of my textbook, in progress).

**The absolute fit of the quadratic time model for the means can be tested by mimicking a saturated means model using the *same random quadratic time slopes* (i.e., holding the model for the variance constant):**

```
display "STATA: Test Absolute Fit of Quadratic Time Means Model"
display "Using Random Quadratic Variance Model"
display "Add 3 session dummy codes to saturate the means model"
mixed rt c.time c.time#c.time c.s1 c.s2 c.s3, || PersonID: time timesq, ///
      reml nolog difficult covariance(unstructured)                    ///
      residuals(independent,t(session)) dfmethod(satterthwaite) dftable(pvalue)

------------------------------------------------------------
           rt |      Coef.   Std. Err.   DF        t    P>|t|
--------------+---------------------------------------------
         time |   74.84574    138.822  312.5     0.54   0.590
c.time#c.time |   -12.2095   17.37596  323.0    -0.70   0.483
           s1 |   358.7487   267.0632  300.0     1.34   0.180
           s2 |   149.3914   147.2164  300.0     1.01   0.311
           s3 |   46.03645   62.77327  300.0     0.73   0.464
        _cons |   1603.145   271.7788  323.4     5.90   0.000
------------------------------------------------------------
// Wald test for fixed quadratic time vs saturated means
test (c.s1=0)(c.s2=0)(c.s3=0), small

       F(  3,300.00) =     3.02
            Prob > F =     0.0299
```

Because there are now **6 fixed effects for the 6 means**, this model is equivalent to **saturated means** (even if the linear and quadratic fixed slopes are largely uninterpretable). In an SEM context, this model would be specified by letting three of the observed occasions' intercepts be estimated (as trajectory discrepancies).

The multivariate Wald test using TEST indicates that the 3 extra session contrasts improved model fit (bad news here).

```
print("R: Test Absolute Fit of Quadratic Time Means Model")
print("Using Random Quadratic Variance Model")
print("Add 3 session dummy codes to saturate the means model")
QuadMean = lmer(data=Example6, REML=TRUE, control=lmerControl(optimizer="Nelder_Mead"),
                formula=rt~1+time+I(time^2)+s1+s2+s3+(1+time+I(time^2)|PersonID))
print("Show results using Satterthwaite DDF"); summary(QuadMean, ddf="Satterthwaite")
print("Wald test for fixed quadratic time vs saturated means")
contestMD(QuadMean, ddf="Satterthwaite",
          L=rbind(c(0,0,0,1,0,0),c(0,0,0,0,1,0),c(0,0,0,0,0,1)))

  Sum Sq   Mean Sq NumDF DenDF F value     Pr(>F)
1 180509 60169.66     3   300 3.02368 0.02994353
```

**Btw, in this quadratic model, it doesn't matter which three dummy codes are added as fixed effects…**

```
display "STATA: Test Absolute Fit of Quadratic Time Means Model"
display "Using Random Quadratic Variance Model"
display "Add different 3 session dummy codes to saturate the means model"
mixed rt c.time c.time#c.time c.s4 c.s5 c.s6, || PersonID: time timesq, ///
      reml nolog difficult covariance(unstructured)                    ///
      residuals(independent,t(session)) dfmethod(satterthwaite) dftable(pvalue)
--------------------------------------------------------------------
           rt |      Coef.   Std. Err.           DF        t    P>|t|
--------------+-----------------------------------------------------
         time |  -187.5126   39.23477         399.7    -4.78   0.000
c.time#c.time |   40.79162   17.37596         323.0     2.35   0.019
           s4 |   -48.6837   62.77327         300.0    -0.78   0.439
           s5 |  -157.3332   147.2164         300.0    -1.07   0.286
           s6 |  -371.9849   267.0632         300.0    -1.39   0.165
        _cons |   1961.893    54.1756         102.4    36.21   0.000
--------------------------------------------------------------------
```

```
// Wald test for fixed quadratic time vs equivalent saturated means
test (c.s4=0)(c.s5=0)(c.s6=0), small

       F(  3,300.00) =    3.02
            Prob > F =    0.0299
```

```
print("R: Test Absolute Fit of Quadratic Time Means Model")
print("Using Random Quadratic Variance Model")
print("Add different 3 session dummy codes to saturate the means model")
QuadMean2 = lmer(data=Example6, REML=TRUE, control=lmerControl(optimizer="Nelder_Mead"),
                 formula=rt~1+time+I(time^2)+s4+s5+s6+(1+time+I(time^2)|PersonID))
print("Show results using Satterthwaite DDF"); summary(QuadMean2, ddf="Satterthwaite")

Fixed effects:
            Estimate Std. Error      df t value   Pr(>|t|)
(Intercept)  1961.89      54.18  102.44  36.214    < 2e-16
time         -187.51      39.23  399.74  -4.779 0.00000248
I(time^2)      40.79      17.38  323.03   2.348     0.0195
s4            -48.68      62.77  300.00  -0.776     0.4386
s5           -157.33     147.22  300.00  -1.069     0.2861
s6           -371.98     267.06  300.00  -1.393     0.1647

print("Wald test for fixed quadratic time vs equivalent saturated means")
contestMD(QuadMean2, ddf="Satterthwaite",
          L=rbind(c(0,0,0,1,0,0),c(0,0,0,0,1,0),c(0,0,0,0,0,1)))

  Sum Sq  Mean Sq NumDF    DenDF F value      Pr(>F)
1 180509 60169.66     3 299.9998 3.02368 0.02994353
```

**The absolute fit of the random quadratic time model for the variance can be tested against a UN variance model using the *same fixed quadratic time slopes* (i.e., holding the model for the means constant):**

```
display "STATA: Test Absolute Fit of Quadratic Time Variance Model"
display "Using Fixed Quadratic Means Model"
display "Change to Unstructured R matrix as variance model answer key"
quietly mixed rt c.time c.time#c.time, || PersonID: , noconstant reml nolog difficult ///
              residuals(unstructured,t(session)) dfmethod(satterthwaite) dftable(pvalue)
estimates store FitFixQuadUN     // Save for LRT
display "-2LL = " e(ll)*-2       // Print -2LL for model
lrtest FitFixQuadUN FitRandQuad  // LRT for random quadratic time vs unstructured variance model

Likelihood-ratio test                                 LR chi2(14) =     35.76
(Assumption: FitRandQuad nested in FitFixQuadUN)      Prob > chi2 =    0.0011
```

```
print("R: Testing Absolute Fit of Quadratic Time Variance Model")
print("Using Random Quadratic Variance Model")
print("Change to Unstructured R matrix as variance model answer key in GLS")
FixQuadUN = gls(data=Example6, method="REML", model=rt~1+time+I(time^2),
                correlation=corSymm(form=~as.numeric(session)|PersonID),  # Unstructured corrs
                weights=varIdent(form=~1|session))                        # Heterogeneous variances

print("LRT for random quadratic time vs unstructured variance model")
# Have to re-run random quadratic time model using LME to get LRT
RandQuadlme = lme(data=Example6, method="REML", rt~1+time+I(time^2),
                  random=~1+time+I(time^2)|PersonID)
anova(FixQuadUN,RandQuadlme) # anova does LRT using LME versions

            Model df      AIC      BIC    logLik   Test  L.Ratio p-value
FixQuadUN       1 24 8314.988 8420.634 -4133.494
RandQuadlme     2 10 8322.746 8366.765 -4151.373 1 vs 2 35.75799  0.0011
```

So what do we conclude about the absolute fit of the random quadratic model for the variance?

For a sample results section, please see the end of Hoffman (2015) chapter 6.