Example 6d: Modeling Change over Time Using Log Time to Approximate Exponential Trends
(complete data, syntax, and output available for 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. Previously in Example6c we used SAS NLMIXED to estimate a truly nonlinear model with an exponential curve, in which REML is not available. This Example 6d builds on an empty means, random intercept model (as shown in Example 6a) to illustrate a strategy by which to mimic the same shape that uses linear and quadratic effects of natural-log-transformed time instead, such that we can use REML in MIXED. At the end we will compare the fit of the "best" version of each family of models (polynomial, piecewise, exponential, and latent basis).

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

// Log-transform time as new predictor
   gen logtime=log(session)
   gen logtimesq=log(session)*log(session)
   label variable logtime   "logtime: Natural Log Session"
   label variable logtimesq "logtimesq: Quadratic Natural Log Session"

// Create session dummy codes for testing means model absolute fit in REML
   gen s1=0
   gen s2=0
   gen s3=0
   gen s4=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
```

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

```
# 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), ]

# Log-transform time as new predictor
Example6$logtime=log(Example6$session)

# 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$s1[which(Example6$session==1)]=1
Example6$s2[which(Example6$session==2)]=1
Example6$s3[which(Example6$session==3)]=1
Example6$s4[which(Example6$session==4)]=1

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

Model 7a. Fixed Linear Log Time, Random Intercept Model

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

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

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

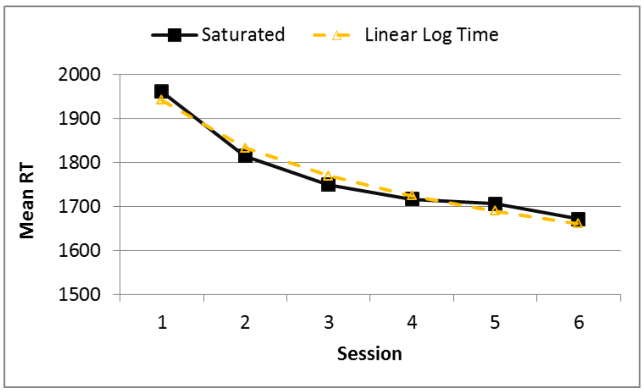
```
display "STATA: 7a: Fixed Linear Log Time, Random Intercept Model"
mixed rt c.logtime, || PersonID: , reml nolog  ///
        residuals(independent,t(session)) dfmethod(satterthwaite) dftable(pvalue)
estimates store FitFixLinLog            // Save for LRT
matrix FixLinLog = r(table)             // Save results for computations below

------------------------------------------------------------------
          rt |      Coef.   Std. Err.          DF       t    P>|t|
-------------+----------------------------------------------------
     logtime |  -156.7164   12.41602        504.0   -12.62   0.000
       _cons |   1942.547   47.41722        118.6    40.97   0.000
------------------------------------------------------------------
------------------------------------------------------------------------------
  Random-effects Parameters  |   Estimate   Std. Err.     [95% Conf. Interval]
-----------------------------+------------------------------------------------
PersonID: Identity           |
                  var(_cons) |   202669.1   29469.65        152411      269500
-----------------------------+------------------------------------------------
               var(Residual) |   34183.44   2153.354      30213.09    38675.54
------------------------------------------------------------------------------
LR test vs. linear model: chibar2(01) = 805.80        Prob >= chibar2 = 0.0000

display "-2LL = " e(ll)*-2              // Print -2LL for model
-2LL = 8391.2737

// Get conditional means per occasion from values of time predictor
lincom _cons*1 + c.logtime*0            // Intercept at Session=1
lincom _cons*1 + c.logtime*0.6931       // Intercept at Session=2
lincom _cons*1 + c.logtime*1.0986       // Intercept at Session=3
lincom _cons*1 + c.logtime*1.3863       // Intercept at Session=4
lincom _cons*1 + c.logtime*1.6094       // Intercept at Session=5
lincom _cons*1 + c.logtime*1.7918       // Intercept at Session=6
```

Estimates (from SAS for better organization)

Label	Estimate	Standard Error	DF	t Value	Pr > |t|	gamma00(1)+gamma10(logtime)
Intercept at Session=1	1942.55	47.4172	119	40.97	<.0001	
Intercept at Session=2	1833.93	45.6960	102	40.13	<.0001	
Intercept at Session=3	1770.38	45.4206	100	38.98	<.0001	
Intercept at Session=4	1725.29	45.5629	101	37.87	<.0001	
Intercept at Session=5	1690.33	45.8648	104	36.85	<.0001	
Intercept at Session=6	1661.74	46.2336	107	35.94	<.0001	

```
matrix list FixLinLog                                    // Show saved results
FixLinLog[9,4]
                rt:         rt:    lns1_1_1:    lnsig_e:
           logtime        _cons        _cons        _cons
     b  -156.71635    1942.5475     6.109665    5.2197483
    se   12.416016    47.417221    .07270385    .03149703
     t  -12.622113    40.967131    84.034954     165.7219
pvalue   6.255e-32    7.647e-72            0            0
    ll  -181.10988    1848.6533    5.9671681    5.1580153
    ul  -132.32283    2036.4417    6.2521619    5.2814814
    df         504    118.58694            .            .
  crit    1.964682    1.9801707     1.959964     1.959964
 eform           0            0            0            0
```

```
// Variances are stored as log of SD instead
global FixLinLogResVar = exp(FixLinLog[1,4])^2      // Save as L1 residual variance for pseudo-R2
display "Pseudo-R2 for L1 Residual Variance = " 1-($FixLinLogResVar/$EmptyResVar)
Pseudo-R2 for L1 Residual Variance = .23867553

print("R 7a: Fixed Linear Log Time, Random Intercept Model")
FixLinLog = lmer(data=Example6, REML=TRUE, control=lmerControl(optimizer="Nelder_Mead"),
                 formula=rt~1+logtime+(1|PersonID))
print("Show results with -2LL using Satterthwaite DDF")
llikAIC(FixLinLog, chkREML=FALSE); summary(FixLinLog, ddf="Satterthwaite")

print("Get conditional mean per occasion from value of time predictor")
print("Intercept at Session=1"); contest1D(FixLinLog, ddf="Satterthwaite", L=c(1,0))
print("Intercept at Session=2"); contest1D(FixLinLog, ddf="Satterthwaite", L=c(1,0.6931))
print("Intercept at Session=3"); contest1D(FixLinLog, ddf="Satterthwaite", L=c(1,1.0986))
print("Intercept at Session=4"); contest1D(FixLinLog, ddf="Satterthwaite", L=c(1,1.3863))
print("Intercept at Session=5"); contest1D(FixLinLog, ddf="Satterthwaite", L=c(1,1.6094))
print("Intercept at Session=6"); contest1D(FixLinLog, ddf="Satterthwaite", L=c(1,1.7918))

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



The fixed linear effect of log time mimics an exponential curve (and appears to fit pretty well to the saturated means).

As shown below, the fixed linear slope for log time accounted for 24% of the level-1 residual variance.

Model 7b. Random Linear Log Time

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

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

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

```
display "STATA: 7b: Random Linear Log Time Model"
mixed rt c.logtime,  || PersonID: logtime, reml nolog covariance(unstructured)  ///
      residuals(independent,t(session)) dfmethod(satterthwaite) dftable(pvalue)
estimates store FitRandLinLog                        // Save for LRT
matrix RandLinLog = r(table)                         // Save results for computations below
display "-2LL = " e(ll)*-2                           // Print -2LL for model
estat recovariance, relevel(PersonID) correlation // GCORR matrix
estat wcorrelation, covariance                       // V matrix
estat wcorrelation                                   // VCORR matrix
lrtest FitRandLinLog FitFixLinLog  // LRT for random linear log time slope variance and covariance
matrix list RandLinLog                               // Show saved results
global RandLinLogResVar = exp(RandLinLog[1,6])^2  // Save as L1 residual variance for pseudo-R2

print("R 7b: Random Linear Log Time Model")
RandLinLog = lmer(data=Example6, REML=TRUE, control=lmerControl(optimizer="Nelder_Mead"),
                  formula=rt~1+logtime+(1+logtime|PersonID))
print("Show results with -2LL using Satterthwaite DDF")
llikAIC(RandLinLog, chkREML=FALSE); summary(RandLinLog, ddf="Satterthwaite")
```

```
$AICtab
      AIC       BIC    logLik  deviance  df.resid 
 8335.429  8361.870 -4161.714  8323.429   600.000 

Random effects:
 Groups   Name        Variance Std.Dev. Corr 
 PersonID (Intercept) 274253   523.7         
          logtime      23101   152.0    -0.56
 Residual              24120   155.3         

Fixed effects:
            Estimate Std. Error      df t value Pr(>|t|)    
(Intercept)  1942.55      53.72  100.00  36.160  < 2e-16
logtime      -156.72      18.37  100.00  -8.531 1.62e-13
```

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

```
         1        2        3        4        5        6
1 298373.0 243378.7 225318.6 212504.8 202565.7 194444.8
2 243378.7 247724.2 212036.2 203828.9 197462.8 192261.3
3 225318.6 212036.2 228386.9 198753.8 194477.8 190984.1
4 212504.8 203828.9 198753.8 219273.4 192360.0 190077.9
5 202565.7 197462.8 194477.8 192360.0 214837.6 189375.0
6 194444.8 192261.3 190984.1 190077.9 189375.0 212921.1
```

The marginal **V** matrix now predicts that **variance changes quadratically over time** (as a function of $logtime^2$).

```
print("LRT for random linear logtime slope variance and covariances")
ranova(RandLinLog, reduce.term=TRUE)
```

```
                                npar  logLik    AIC    LRT Df Pr(>Chisq)
<none>                             6 -4161.7 8335.4                     
logtime in (1 + logtime | PersonID)    4 -4195.6 8399.3 67.845  2  1.852e-15
```

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

```
display "STATA: Test Absolute Fit of the Means Model"
display "(Using Random Linear Log Time Variance Model"
display "Add 4 session dummy codes to saturate the means model"
mixed rt c.logtime c.s1 c.s2 c.s3 c.s4, || PersonID: logtime,  ///
      reml nolog covariance(unstructured)                      ///
      residuals(independent,t(session)) dfmethod(satterthwaite) dftable(pvalue)
```

```
------------------------------------------------------------------
          rt |      Coef.   Std. Err.          DF        t    P>|t|
-------------+----------------------------------------------------
     logtime |  -192.1864   120.4285        418.3    -1.60    0.111
          s1 |  -54.59442   204.0479        400.0    -0.27    0.789
          s2 |  -68.10195   121.8316        400.0    -0.56    0.576
          s3 |  -55.31481   74.35333        400.0    -0.74    0.457
          s4 |  -32.26443    42.0226        400.0    -0.77    0.443
       _cons |   2016.488   210.0352        449.2     9.60    0.000
------------------------------------------------------------------
```

```
test (c.s1=0)(c.s2=0)(c.s3=0)(c.s4=0), small  // Wald test for fixed linear log time vs sat means
       F(  4,400.00) =     1.67
            Prob > F =     0.1554
```

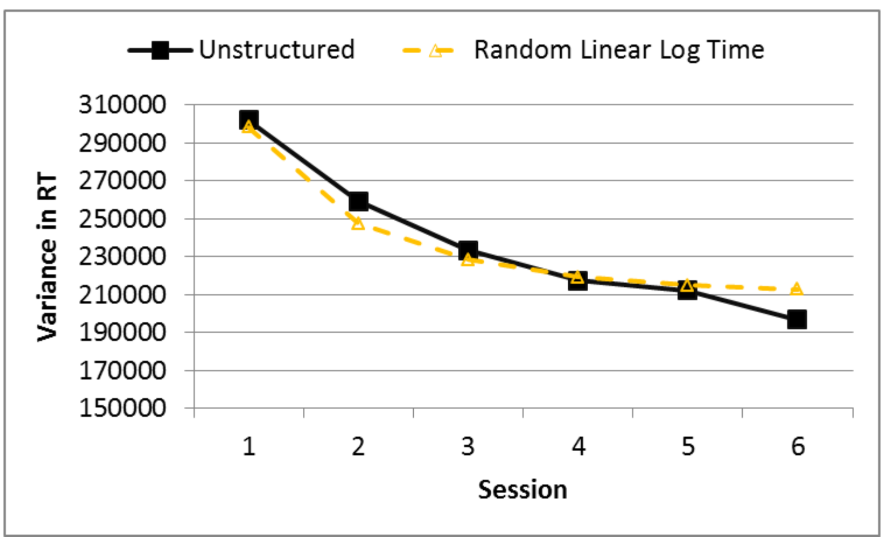
```
print("R: Test Absolute Fit of Linear Log Time Means Model")
print("Using Random Linear Log Time Variance Model")
print("Add 4 session dummy codes to second piece to saturate the means model")
LinLogMean = lmer(data=Example6, REML=TRUE, control=lmerControl(optimizer="Nelder_Mead"),
                  formula=rt~1+logtime+s1+s2+s3+s4+(1+logtime|PersonID))
print("Show results with -2LL using Satterthwaite DDF")
llikAIC(LinLogMean, chkREML=FALSE); summary(LinLogMean, ddf="Satterthwaite")
```

```
print("Wald test for fixed two-piece slopes vs saturated means")
contestMD(LinLogMean, ddf="Satterthwaite",
          L=rbind(c(0,0,1,0,0,0),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 160353.3 40088.33     4 400.0001 1.673085 0.1553707
```

Because there are now **6 fixed effects for the 6 means**, this model is equivalent to **saturated means** (even if the linear log time fixed slope is largely uninterpretable). In an SEM context, this model would be specified by letting four of the observed occasions' intercepts be estimated (as discrepancies). The multivariate Wald test indicates that the 4 extra session contrasts did not improve model fit (which is good news here).

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

```
display "STATA: Test Absolute Fit of the Variance Model"
display "(Using Fixed Linear Log Time Means Model"
mixed rt c.logtime, || PersonID: , noconstant reml nolog difficult  ///
         residuals(unstructured,t(session)) dfmethod(satterthwaite) dftable(pvalue)
estimates store FitFixLinLogUN      // Save for LRT
lrtest FitFixLinLogUN FitRandLinLog // LRT for random linear log time vs unstruct variance model

print("R: Test Absolute Fit of Linear Log Time Variance Model")
print("Using Fixed Linear Log Time Means Model)")
print("Change to Unstructured R matrix as variance model answer key in GLS")
LinLogUN = gls(data=Example6, method="REML", model=rt~1+logtime,
               correlation=corSymm(form=~as.numeric(session)|PersonID),  # Unstructured corrs
               weights=varIdent(form=~1|session))                        # Heterogeneous variances

print("LRT for random linear logtime vs unstructured variance model")
# Have to re-run random quadratic log time model using LME to get LRT
RandLinLoglme = lme(data=Example6, method="REML", rt~1+logtime, random=~1+logtime|PersonID)
anova(LinLogUN,RandLinLoglme) # anova does LRT using LME versions

              Model df      AIC      BIC    logLik   Test  L.Ratio p-value
LinLogUN          1 23 8311.419 8412.701 -4132.710
RandLinLoglme     2  6 8335.429 8361.850 -4161.714 1 vs 2 58.00971  <.0001
```



The random linear slope of log time approximates most of the observed variances from the unstructured model, but not well enough according to the LRT (and the same is likely true of the covariances, not shown here).

Model 7c. Fixed Quadratic, Random Linear Log Time

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

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

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

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

Predicted intercept at any occasion:

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

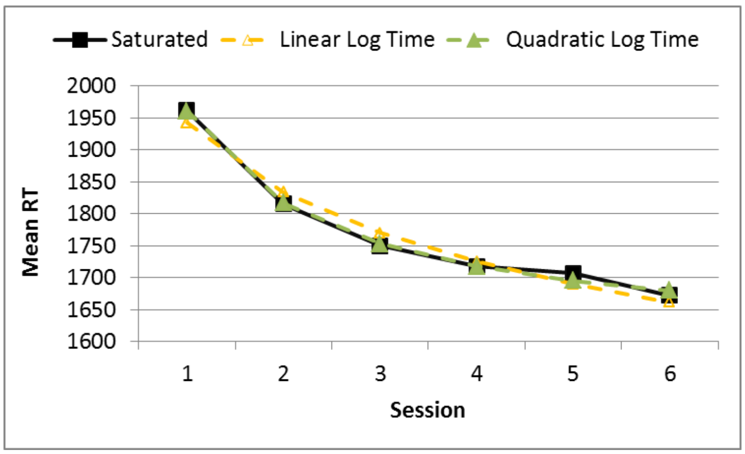
Instantaneous linear time slope at any occasion:

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

```
display "STATA: 7c: Fixed Quadratic, Random Linear Log Time Model"
mixed rt c.logtime c.logtime#c.logtime, || PersonID: logtime, reml nolog cov(unstructured) ///
      residuals(independent,t(session)) dfmethod(satterthwaite) dftable(pvalue)
estimates store FitFixQuadLog                        // Save for LRT
matrix FixQuadLog = r(table)                         // Save results for computations below

------------------------------------------------------------------------
                 rt |      Coef.   Std. Err.          DF       t    P>|t|
--------------------+---------------------------------------------------
            logtime |  -240.6102   39.45926        502.0   -6.10   0.000
c.logtime#c.logtime |   46.91462   19.52878        403.0    2.40   0.017
              _cons |   1960.964   54.26537        104.1   36.14   0.000
------------------------------------------------------------------------
----------------------------------------------------------------------------
  Random-effects Parameters  |   Estimate   Std. Err.     [95% Conf. Interval]
-----------------------------+----------------------------------------------
PersonID: Unstructured       |
                var(logtime) |   23229.18   4880.984      15387.81    35066.38
                  var(_cons) |   274453.4   41239.34      204440.6    368442.7
          cov(logtime,_cons) |   -44681.8   11492.79     -67207.26   -22156.34
-----------------------------+----------------------------------------------
               var(Residual) |   23838.87   1679.377      20764.48    27368.45
----------------------------------------------------------------------------
LR test vs. linear model: chi2(3) = 876.51                Prob > chi2 = 0.0000

display "-2LL = " e(ll)*-2                           // Print -2LL for model
-2LL = 8309.904

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

             |   logtime      _cons
-------------+----------------------
     logtime |         1
       _cons | -.5596022          1

// Get conditional means per occasion from values of time predictors
lincom _cons*1 + c.logtime*0      + c.logtime#c.logtime*0      // Intercept at Session=1
lincom _cons*1 + c.logtime*0.6931 + c.logtime#c.logtime*0.4805 // Intercept at Session=2
lincom _cons*1 + c.logtime*1.0986 + c.logtime#c.logtime*1.2069 // Intercept at Session=3
lincom _cons*1 + c.logtime*1.3863 + c.logtime#c.logtime*1.9218 // Intercept at Session=4
lincom _cons*1 + c.logtime*1.6094 + c.logtime#c.logtime*2.5903 // Intercept at Session=5
lincom _cons*1 + c.logtime*1.7918 + c.logtime#c.logtime*3.2104 // Intercept at Session=6

// Get instantaneous linear slope per occasion from 2*value of time predictor
lincom c.logtime*1 + c.logtime#c.logtime*0     , small    // Linear Slope at Session=1
lincom c.logtime*1 + c.logtime#c.logtime*1.3863, small    // Linear Slope at Session=2
lincom c.logtime*1 + c.logtime#c.logtime*2.1972, small    // Linear Slope at Session=3
lincom c.logtime*1 + c.logtime#c.logtime*2.7726, small    // Linear Slope at Session=4
lincom c.logtime*1 + c.logtime#c.logtime*3.2189, small    // Linear Slope at Session=5
lincom c.logtime*1 + c.logtime#c.logtime*3.5835, small    // Linear Slope at Session=6
```

```
                              Estimates (from SAS for better organization)
                                               Standard
Label                        Estimate           Error       DF     t Value     Pr > |t|   g = gamma, logt = logtime
Intercept at Session=1        1960.96         54.2653      104       36.14       <.0001   g00(1)+ g10(logt)+ g20(logt^2)
Intercept at Session=2        1816.74         48.1936      105       37.70       <.0001
Intercept at Session=3        1753.25         45.9683      105       38.14       <.0001
Intercept at Session=4        1717.57         44.6261      101       38.49       <.0001
Intercept at Session=5        1695.25         44.2821      100       38.28       <.0001
Intercept at Session=6        1680.45         44.9706      106       37.37       <.0001

Linear Slope at Session=1     -240.61         39.4593      502       -6.10       <.0001   g10(1) + 2*g20(logt)
Linear Slope at Session=2     -175.57         19.9776      139       -8.79       <.0001
Linear Slope at Session=3     -137.53         20.0322      140       -6.87       <.0001
Linear Slope at Session=4     -110.53         26.5904      338       -4.16       <.0001
Linear Slope at Session=5    -89.5967         33.4381      472       -2.68       0.0076
Linear Slope at Session=6    -72.4917         39.5812      502       -1.83       0.0676
```

```
matrix list FixQuadLog                                    // Show saved results
// Variances are stored as log of SD instead
global FixQuadLogResVar = exp(FixQuadLog[1,7])^2          // Save as L1 residual variance for pseudo-R2
display "Pseudo-R2 for L1 Residual Variance = " 1-($FixQuadLogResVar/$RandLinLogResVar)
Pseudo-R2 for L1 Residual Variance = .01167206
```



The quadratic effect of log time appears to make the predicted line a little more bendy, such that it fits the saturated means slightly (but significantly) better.

```
print("R 7c: Fixed Quadratic, Random Linear Log Time Model")
FixQuadLog = lmer(data=Example6, REML=TRUE, control=lmerControl(optimizer="Nelder_Mead"),
                  formula=rt~1+logtime+I(logtime^2)+(1+logtime|PersonID))
print("Show results with -2LL using Satterthwaite DDF")
llikAIC(FixQuadLog, chkREML=FALSE); summary(FixQuadLog, ddf="Satterthwaite")

print("Get conditional mean per occasion from values of time predictors")
print("Intercept at Session=1"); contest1D(FixQuadLog, ddf="Satterthwaite", L=c(1,0,0))
print("Intercept at Session=2"); contest1D(FixQuadLog, ddf="Satterthwaite", L=c(1,0.6931,0.4805))
print("Intercept at Session=3"); contest1D(FixQuadLog, ddf="Satterthwaite", L=c(1,1.0986,1.2069))
print("Intercept at Session=4"); contest1D(FixQuadLog, ddf="Satterthwaite", L=c(1,1.3863,1.9218))
print("Intercept at Session=5"); contest1D(FixQuadLog, ddf="Satterthwaite", L=c(1,1.6094,2.5903))
print("Intercept at Session=6"); contest1D(FixQuadLog, ddf="Satterthwaite", L=c(1,1.7918,3.2104))

print("Get instantaneous linear slope per occasion from 2*value of time predictor")
print("Linear Slope at Session=1 Time=0"); contest1D(FixQuadLog, ddf="Satterthwaite", L=c(0,1,0))
print("Linear Slope at Session=2 Time=1"); contest1D(FixQuadLog, ddf="Satterthwaite", L=c(0,1,1.3863))
print("Linear Slope at Session=3 Time=2"); contest1D(FixQuadLog, ddf="Satterthwaite", L=c(0,1,2.1972))
print("Linear Slope at Session=4 Time=3"); contest1D(FixQuadLog, ddf="Satterthwaite", L=c(0,1,2.7726))
print("Linear Slope at Session=5 Time=4"); contest1D(FixQuadLog, ddf="Satterthwaite", L=c(0,1,3.2189))
print("Linear Slope at Session=6 Time=5"); contest1D(FixQuadLog, ddf="Satterthwaite", L=c(0,1,3.5835))

print("Psuedo-R2 for variance accounted for by fixed quadratic logtime")
pseudoRSquaredinator(smallerModel=RandLinLog, largerModel=FixQuadLog)
```

Model 7d. Random Quadratic Log Time

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

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

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

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

Predicted intercept at any occasion:
$= \gamma_{00} + \gamma_{10}(\text{LogTime}_{ti}) + \gamma_{20}(\text{LogTime}_{ti})^2$

Instantaneous linear time slope at any occasion:
$= \gamma_{10} + 2\gamma_{20}(\text{LogTime}_{ti})$

```
display "STATA: 7d: Random Quadratic Log Time Model"
mixed rt c.logtime c.logtime#c.logtime, || PersonID: logtime logtimesq,  ///
      reml nolog difficult covariance(unstructured)                      ///
      residuals(independent,t(session)) dfmethod(satterthwaite) dftable(pvalue)
estimates store FitRandQuadLog                    // Save for LRT
matrix RandQuadLog = r(table)                     // Save results for computations below

------------------------------------------------------------------------------
                  rt |      Coef.   Std. Err.          DF        t    P>|t|
---------------------+--------------------------------------------------------
             logtime |  -240.6102   54.14334        100.0    -4.44    0.000
 c.logtime#c.logtime |   46.91463    26.4888        100.0     1.77    0.080
               _cons |   1960.964   54.65208        100.0    35.88    0.000
------------------------------------------------------------------------------
------------------------------------------------------------------------------
  Random-effects Parameters  |   Estimate   Std. Err.     [95% Conf. Interval]
-----------------------------+------------------------------------------------
PersonID: Unstructured       |
                var(logtime) |   199199.5   42610.93      130980.6    302948.9
               var(logtim~q) |    43024.7   10274.46      26943.11    68704.95
                  var(_cons) |   285072.4   42686.93      212566.9    382309.1
        cov(logtime,logtim~q) |  -86829.94   20319.62     -126655.7   -47004.22
          cov(logtime,_cons) |  -80402.61   31908.46       -142942   -17863.18
        cov(logtim~q,_cons) |   15958.66   14908.06     -13260.59    45177.91
-----------------------------+------------------------------------------------
               var(Residual) |   17231.53   1399.967      14694.96    20205.95
------------------------------------------------------------------------------
LR test vs. linear model: chi2(6) = 920.29                Prob > chi2 = 0.0000

display "-2LL = " e(ll)*-2                        // Print -2LL for model
-2LL = 8266.1215

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

             |   logtime  logtimesq      _cons
-------------+---------------------------------
     logtime |         1
   logtimesq | -.9379216          1
       _cons | -.3374026   .1440987          1

estat wcorrelation, covariance                    // V matrix
estat wcorrelation                                // VCORR matrix
```

The marginal **V** matrix (from SAS) now predicts the variances to change in a **quartic pattern (logtime⁴)**.

Estimated V Matrix for PersonID 101

Row	Col1	Col2	Col3	Col4	Col5	Col6
1	302304	237009	216002	204280	197007	192244
2	237009	253981	226105	213856	201669	189971
3	216002	226105	236995	209779	198899	187976
4	204280	213856	209779	219785	194472	186263
5	197007	201669	198899	194472	206863	184764
6	192244	189971	187976	186263	184764	200661

Estimated V Correlation Matrix for PersonID 101

Row	Col1	Col2	Col3	Col4	Col5	Col6
1	1.0000	0.8553	0.8070	0.7925	0.7878	0.7805
2	0.8553	1.0000	0.9216	0.9052	0.8798	0.8415
3	0.8070	0.9216	1.0000	0.9192	0.8983	0.8620
4	0.7925	0.9052	0.9192	1.0000	0.9120	0.8869
5	0.7878	0.8798	0.8983	0.9120	1.0000	0.9069
6	0.7805	0.8415	0.8620	0.8869	0.9069	1.0000

```
lrtest FitRandQuadLog FitFixQuadLog   // LRT for random quad logtime slope variance and covariances

Likelihood-ratio test                                 LR chi2(3)   =     43.78
(Assumption: FitFixQuadLog nested in FitRandQuadLog)  Prob > chi2 =    0.0000

matrix list RandQuadLog                // Show saved results
// Save fixed effects and variances for computations below
global FixInt  = RandQuadLog[1,3]          // Save fixed intercept
global FixLin  = RandQuadLog[1,1]          // Save fixed linear logtime slope
global FixQuad = RandQuadLog[1,2]          // Save fixed quadratic logtime slope
global IntVar  = exp(RandQuadLog[1,6])^2   // Save L2 random intercept variance
global LinVar  = exp(RandQuadLog[1,4])^2   // Save L2 random linear logtime slope variance
global QuadVar = exp(RandQuadLog[1,5])^2   // Save L2 random quadratic logtime slope variance
// Check if correct saved variances
display $IntVar
display $LinVar
display $QuadVar
display "STATA 95% Random Intercept CI"
display "Lower = " $FixInt - 1.96*sqrt($IntVar)
display "Upper = " $FixInt + 1.96*sqrt($IntVar)
display "STATA 95% Random Linear LogTime Slope CI"
display "Lower = " $FixLin - 1.96*sqrt($LinVar)
display "Upper = " $FixLin + 1.96*sqrt($LinVar)
display "STATA 95% Random Quadratic LogTime Slope CI"
display "Lower = " $FixQuad - 1.96*sqrt($QuadVar)
display "Upper = " $FixQuad + 1.96*sqrt($QuadVar)
```

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

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

$$\text{Intercept 95\% CI} = \gamma_{00} \pm \left(1.96 * \sqrt{\tau^2_{U_0}}\right) \rightarrow 1{,}960.96 \pm \left(1.96 * \sqrt{285{,}072}\right) = 914 \text{ to } 3{,}007$$

$$\text{Linear LogTime Slope 95\% CI} = \gamma_{10} \pm \left(1.96 * \sqrt{\tau^2_{U_1}}\right) \rightarrow -240.61 \pm \left(1.96 * \sqrt{199{,}200}\right) = -1115 \text{ to } 634$$

$$\text{Quadratic LogTime Slope 95\% CI} = \gamma_{20} \pm \left(1.96 * \sqrt{\tau^2_{U_2}}\right) \rightarrow 46.91 \pm \left(1.96 * \sqrt{43025}\right) = -360 \text{ to } 453$$

```
print("R 7d: Random Quadratic Log Time Model -- reports convergence problem")
RandQuadLog = lmer(data=Example6, REML=TRUE, control=lmerControl(optimizer="Nelder_Mead"),
                   formula=rt~1+logtime+I(logtime^2)+(1+logtime+I(logtime^2)|PersonID))
print("Show results with -2LL using Satterthwaite DDF")
llikAIC(RandQuadLog, chkREML=FALSE); summary(RandQuadLog, ddf="Satterthwaite")
```

```
$AICtab
      AIC       BIC    logLik  deviance  df.resid
 8350.850  8394.918 -4165.425  8330.850   596.000

Random effects:
 Groups   Name         Variance Std.Dev. Corr
 PersonID (Intercept)  259082.0 509.00
          logtime         759.5  27.56   -1.00
          I(logtime^2)   4424.8  66.52   -0.36  0.36
 Residual               25945.5 161.08

Fixed effects:
             Estimate Std. Error     df t value Pr(>|t|)
(Intercept)   1960.96      53.03  99.68  36.975  < 2e-16
logtime       -240.61      38.10 394.87  -6.315 7.29e-10
I(logtime^2)    46.91      21.42 373.97   2.190   0.0291
```

The −2LL value is not the same as what STATA and SAS came up with, and the output notes a convergence problem. The random slope correlation = −1.00 below is likely part of the problem (which was −.94 in STATA and SAS).

```
print("Use custom function to get predicted V matrix"); PrintV(RandQuadLog) # Not shown b/c wrong
```

```
print("LRT for random quadratic time slope variances and covariances")
ranova(RandQuadLog, reduce.term=TRUE)
                                                          npar  logLik    AIC      LRT Df Pr(>Chisq)
<none>                                                      10 -4165.4 8350.8
logtime in (1 + logtime + I(logtime^2) | PersonID)          7 -4165.7 8345.3   0.4767  3      0.924
I(logtime^2) in (1 + logtime + I(logtime^2) | PersonID)     7 -4155.0 8323.9 -20.9456  3      1.000

# 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(RandQuadLog)); as.data.frame(VarCorr(RandQuadLog))
# Save fixed effects and variances for computations below
FixInt  = as.data.frame(fixef(RandQuadLog))[1,1]   # Save fixed intercept
FixLin  = as.data.frame(fixef(RandQuadLog))[2,1]   # Save fixed linear logtime slope
FixQuad = as.data.frame(fixef(RandQuadLog))[3,1]   # Save fixed quadratic logtime slope
IntVar  = as.data.frame(VarCorr(RandQuadLog))[1,4] # Save L2 random intercept variance
LinVar  = as.data.frame(VarCorr(RandQuadLog))[2,4] # Save L2 random linear logtime slope variance
QuadVar = as.data.frame(VarCorr(RandQuadLog))[3,4] # Save L2 random quad logtime 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)
```

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

```
display "STATA: Test Absolute Fit of Quadratic Log Time Means Model"
display "Using Random Quadratic Log Time Variance Model"
display "Add 3 session dummy codes to saturate the means model"
mixed rt c.logtime c.logtime#c.logtime c.s1 c.s2 c.s3, || PersonID: logtime logtimesq,  ///
      reml nolog difficult covariance(unstructured)                                     ///
      residuals(independent,t(session)) dfmethod(satterthwaite) dftable(pvalue)
test (c.s1=0)(c.s2=0)(c.s3=0), small  // Wald test for fixed quadratic log time vs saturated means

       F(  3,300.00) =    0.44
            Prob > F =    0.7259

print("R: Test Absolute Fit of Quadratic Log Time Means Model")
print("Using Random Quadratic Log Time Variance Model")
print("Add 3 session dummy codes to second piece to saturate the means model")
QuadLogMean = lmer(data=Example6, REML=TRUE, control=lmerControl(optimizer="Nelder_Mead"),
                   formula=rt~1+logtime+I(logtime^2)+s1+s2+s3+(1+logtime+I(logtime^2)|PersonID))
print("Show results using Satterthwaite DDF"); summary(QuadLogMean, ddf="Satterthwaite")
print("Wald test for fixed quadratic log time vs saturated means")
contestMD(QuadLogMean, 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 22774.41 7591.472     3 299.9902 0.4381059 0.7258993
```

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 discrepancies). The multivariate Wald test indicates that the 3 extra session contrasts did not improve model fit (which is good news here).

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

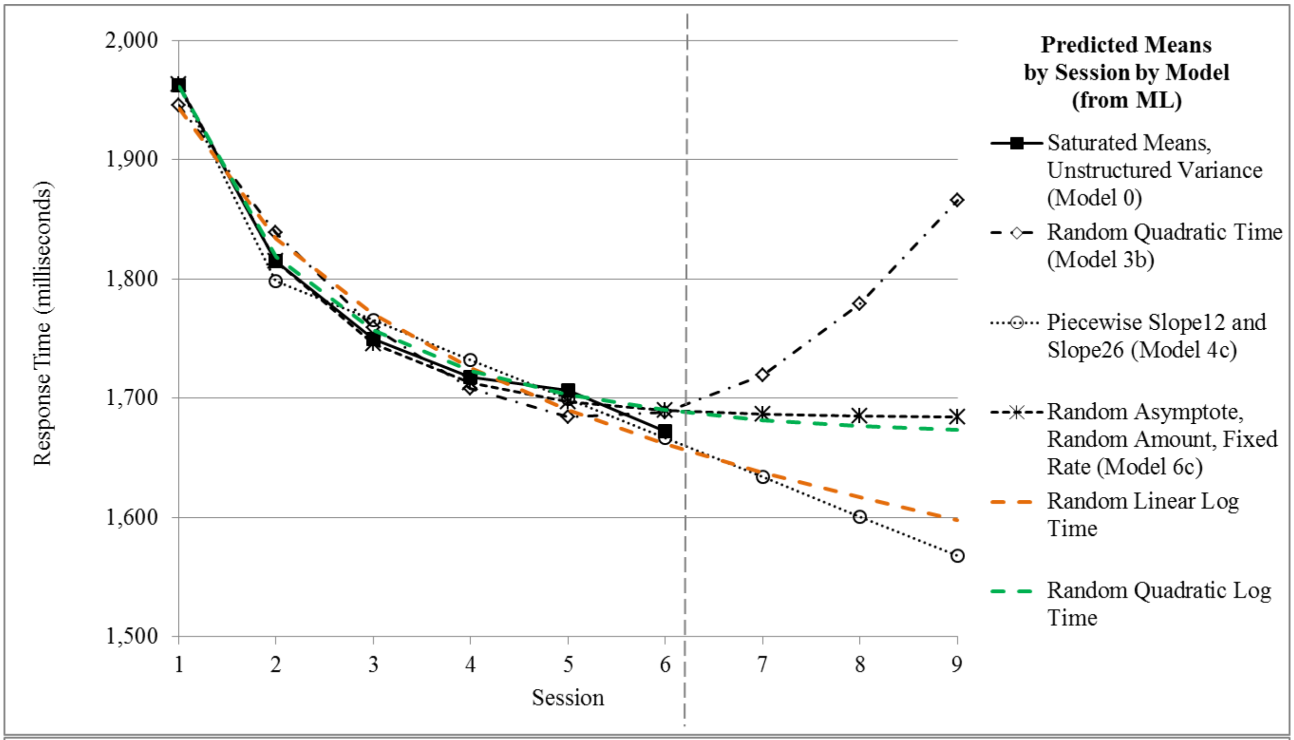
```
display "STATA: Test Absolute Fit of Quadratic Log Time Variance Model"
display "Using Fixed Quadratic Log Time Means Model"
display "Change to Unstructured R matrix as variance model answer key"
mixed rt c.logtime c.logtime#c.logtime, || PersonID: , noconstant reml nolog difficult  ///
      residuals(unstructured,t(session))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

Likelihood-ratio test                                 LR chi2(14) =     11.98
(Assumption: FitRandQuadLog nested in FitFixQuadLo~N) Prob > chi2 =    0.6075  Hooray!

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

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

               Model df      AIC      BIC    logLik   Test  L.Ratio p-value
QuadLogUN          1 24 8302.137 8407.783 -4127.068
RandQuadLoglme     2 10 8286.122 8330.141 -4133.061 1 vs 2 11.98459  0.6075  Hooray!
```

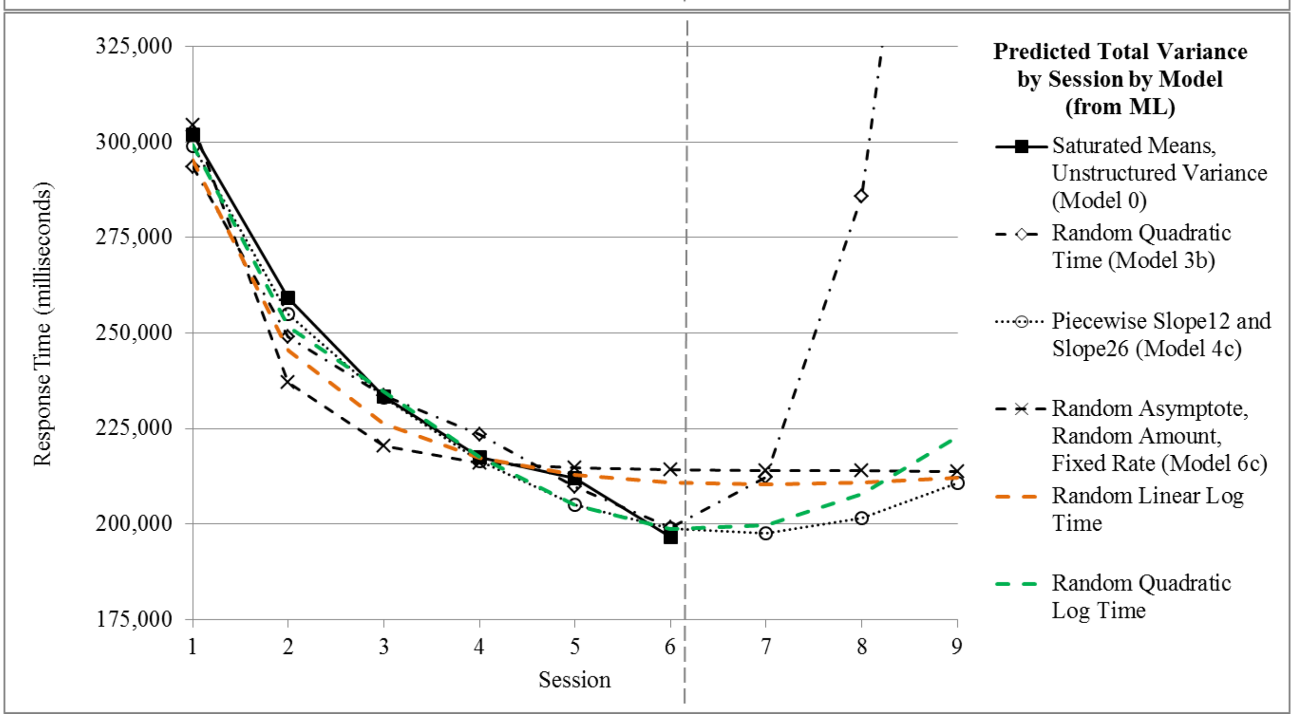
At last, the grand finale comparison: Tests of absolute fit of each side of the model for the "best" version of each model as presented in Examples 6a–6d, as well as tests of absolute fit of both sides of the model combined (see example syntax and output online for the latter)....

Summary of Tests of Absolute Model Fit For Each Side of the Model		Means Side Test			Variance Side Test		
		F	DF	p-value	LRT	DF	p-value
3b	Polynomial: Random Quadratic Time	3.02	3	0.030	35.76	14	0.001
4c	**Piecewise: Random Slope12, Random Slope26**	1.58	3	**0.195**	15.77	14	**0.327**
6c	Negative Exponential: Random Asymptote, Random Amount, Fixed Rate	0.29	3	**0.831**	46.50	17	0.000
7c	**Random Linear Log Time**	**1.67**	**4**	**0.155**	58.01	17	0.000
7d	**Random Quadratic Log Time**	**0.44**	**3**	**0.726**	**11.98**	**14**	**0.608**

Model		Total # Parameters	ML -2LL	ML AIC	ML BIC
7c	Random Linear Log Time	6	8340.49	8352.5	8368.2
6c	Negative Exponential: Random Asymptote, Random Amount, Fixed Rate	7	8327.30	8341.3	8359.6
3b	Polynomial: Random Quadratic Time	10	8321.77	8341.8	8367.9
4c	Piecewise: Random Slope12, Random Slope26	10	8298.95	8318.9	8345.1
7d	**Random Quadratic Log Time**	10	8291.51	**8311.5**	**8337.7**
8	**Latent Basis**	10	8326.19	8346.2	8372.3
0	Least Parsimonious Baseline: Saturated Means, Unstructured Variance	27	**8278.09**	8332.1	8402.7
0	Absolute Fit worse than Saturated Means, Unstructured Variance Model?	df	Chi-Square	p-value	
7c	Random Linear Log Time	21	62.41	.000	
6c	Negative Exponential: Random Asymptote, Random Amount, Fixed Rate	20	49.21	.000	
3b	Polynomial: Random Quadratic Time	17	43.68	.000	
4c	Piecewise: Random Slope12, Random Slope26	17	20.86	.233	
7d	**Random Quadratic Log Time**	**17**	**13.42**	**.708**	
8	**Latent Basis**	**17**	**48.10**	**.000**	

Predicted means/variances all in ML for comparability





I think random quadratic log time wins—and that's going in chapter 6 when I revise my book!

For sample results sections that go with Example 6, please see the end of Chapter 6.