Exercise 16 - Repeated measures analysis with mixed models

Zoltan Kekecs

27 november 2018

Contents

1 Abstract

This exercise is focused on the use of linear mixed models in case of repeated measures designes, and how to re-structure data from a 'wide format' to a 'long format' to be useful for linear mixed model analysis

The latest version of this document and the code the document refers to can be found in the GitHub repository of the class at: https://github.com/kekecsz/PSYP13_Data_analysis_class-2018

2 Data management and descriptive statistics

2.1 Loading packages

You will need the following packages for this exercise.

```
library(psych)  # for describe
library(reshape2)  # for melt function
library(ggplot2)  # for ggplot
library(cAIC4)  # for cAIC
library(r2glmm)  # for r2beta
```

2.2 Custom functions

This is a function to extract standardized beta coefficients from linear mixed models.

This function was adapted from: https://stackoverflow.com/questions/25142901/standardized-coefficients-for-lmer-model

```
stdCoef.merMod <- function(object) {
    sdy <- sd(getME(object, "y"))
    sdx <- apply(getME(object, "X"), 2, sd)
    sc <- fixef(object) * sdx/sdy
    se.fixef <- coef(summary(object))[, "Std. Error"]
    se <- se.fixef * sdx/sdy
    return(data.frame(stdcoef = sc, stdse = se))
}
```

2.3 Load wound healing data

In this exercise we will work with simulated data about wound healing over time after a surgical procedure. We know that psychological factors, esepcailly stress, can influence recovery after surgery, and the rate of wound healing. Let's say that we have a theory that wound it is important for hospitalized patients to have a connection with the outside world. So we may think that patients who have a window close to their hospital beds may have a better mood and thus, would show a faster recovery after surgery. This hypothesis is tested in a simple study looking at whether the distance of the patients' bed from the closest window would predict rate of wound healing. Distance is measured in meters, and wound healing is measured by rating the wound using a standardized wound rating measure taking into account the size of the wound, its inflammation and scarring. A physician rates the wound each day for seven days in the afternoon at the same time of the day. We will use this variable as our outcome measure.

Let's say that our hypothesis extends to the role of sunlight in this context, where we suppose that the more sunlight a patient gets the better their recovery would be. To test this hypothesis, our model will take into account whether the bed of the patient is in the north wing or the south wing of the hospital (since the hospital is in the northern hemisphere, we can assume that patients in the south wing would get more sunlight overall during their hospital stay).

In the code below we load the dataset from GitHub, then, we identify ID (participant ID) and location (south or north wing) as factors, which will help R handle these variables.

```
data_wound = read.csv("https://raw.githubusercontent.com/kekecsz/PSYP13_Data_analysis_class-2018/master/

# asign ID and location as factors
data_wound$ID = factor(data_wound$ID)
data_wound$location = factor(data_wound$location)
```

Lets inspect the layout of this data frame using the View() function. Notice that each row contains all the data collected from the same participant, and the wound rating data for each day are stored in variables 'day_1', 'day_2', ..., 'day_7' respectively.

```
View(data_wound)
```

2.4 Check the dataset

As always, you should start by checking the dataset for coding errors or data that does not make sense, by eyeballing the data through the data view tool, checking descriptive statistics and through data visualization.

```
# descriptives
describe(data_wound)
table(data_wound[, "location"])

# histograms
hist(data_wound$day_1)
hist(data_wound$day_2)
hist(data_wound$day_3)
hist(data_wound$day_4)
hist(data_wound$day_5)
hist(data_wound$day_6)
hist(data_wound$day_7)
hist(data_wound$distance_window)
```

3 Repeated measures analysis using linear mixed models

3.1 Exploring clustering in the data

Now lets look at the relatiohsip of the repeated measures of wound healing. (First, we save the variable names of the repeated measures into an object called repeated_variables so that we can easily refer to these variable names later in our functions.) We can explore the correlation between the variables with the cor() function. Notice that the repeated measures data points are highly correlated. This shows that the different observations of wound rating are not independent from each other. This is normal, since the wound rating and the initial size of the incision and the wound healing rate depends on the patient. So this is clustered data. Just like the data in the previous exercise was clustered in classes, here, the data is clustered within participants.

```
# designate which are the repeated varibales
repeated_variables = c("day_1", "day_2", "day_3", "day_4", "day_5",
    "day_6", "day_7")

# correlation of repeated variables
cor(data_wound[, repeated_variables])
```

```
##           day_1     day_2     day_3     day_4     day_5     day_6
## day_1 1.0000000 0.8505812 0.6565436 0.5469131 0.4647985 0.3849832
## day_2 0.8505812 1.0000000 0.8360082 0.7686539 0.5932054 0.4679627
## day_3 0.6565436 0.8360082 1.0000000 0.8618242 0.7075322 0.6016984
```

```
## day_4 0.5469131 0.7686539 0.8618242 1.0000000 0.8542087 0.7178904
## day_5 0.4647985 0.5932054 0.7075322 0.8542087 1.0000000 0.8898712
## day_6 0.3849832 0.4679627 0.6016984 0.7178904 0.8898712 1.0000000
## day_7 0.2708845 0.3461029 0.4406317 0.6015019 0.7978323 0.9043339
##           day_7
## day_1 0.2708845
## day_2 0.3461029
## day_3 0.4406317
## day_4 0.6015019
## day_5 0.7978323
## day_6 0.9043339
## day_7 1.0000000
```

3.2 Reshape dataframe

Because of this clustering, we can basically treat this data similarly to the bullying dataset. However, first we need to re-structure the dataset to a format that will be interpretable to the linear mixed effect regression (lmer()) function.

At this point, the dataframe contains 7 observations of wound rating from the same participant in one row (one for each day of the week while data was collected). This format is called the **wide format**.

For lmer() to be able to interpret the data correctly, we will have to restructure the dataframe so that each row contains a single observation. This would mean that each participant would have 7 rows instead of 1. Data in the variables ID, distance_window, and location would be duplicated in each of the rows of the same participant, and there would be only a single column for wound rating in this new dataframe. This format of the data is usually referred to as the **long format**.

We can do this using the melt() function from the reshape2 package. In the melt function we specify the wide format dataframe, then at measure.vars= we specify which variable names contain the repeated measures. (Remember that we have already saved these variable names to an object called repeated_variables, so we can simply refer to this object now instead of having to type in all of the variable names again.) At variable.name = we can give a varialbe name to the time variable. We will simply call this 'time' now. At value.name = we can specify what should be the variable name of the variable contaning the wound rating data. We will call this 'wound_rating'.

(As always, we create a new object where we will store the data with the new shape, and leave the raw data unchanged. The new object is called data_wound_long.)

```
data_wound_long = melt(data_wound, measure.vars = repeated_variables,
    variable.name = "time", value.name = "wound_rating")
```

We can also clarify the new dataframe a little bit by ordering the rows so that each observation from the same participant follow each other.

Also, notice that our 'time' variable now contains the names of the repeated measures variables from the long format ('day_1', 'day_2' etc.). We will simplify this by simply using numbers 1-7 here. (the easiest to do this is by using the as.numeric() function, which, if applied to factors, will convert the factor levels one by one to consecutive numbers).

```
# order data frame by participant ID(not necessary, just
# makes the dataframe look more intuitive)
data_wound_long = data_wound_long[order(data_wound_long[, "ID"]),
    ]

# change the time variable to a numerical vector
data_wound_long$time = as.numeric(data_wound_long$time)
```

Let's explore how this new datafrmae looks like.

```
View(data_wound_long)
```

3.3 Building the linear mixed model

Now that we have our dataframe in an approriate formate, we can build our prediction model. The outcome will be wound rating, and the fixed effect predictors will be day after surgery, distance of the bed from the window, and south or north location (these information are stored in the variables time, distance_window, and location).

Since our outcome is clustered within participants, the random effect predictor will be participant ID. As in the previous exercise, we will fit two models, one will be a random intercept model, the other a random slope model.

Note that the **random intercept model** means that we suspect that the each participant is different in their overall wound rating (or baseline wound rating), but that the effect of the fixed effect predictors (time, distance from window, and location) is the same for each participant. On the other hand, the **random slope model** not only baseline wound rating will be different across participants, but also that the fixed effects will be different from participant to participant as well.

Note that here we have 3 different fixed effect predictors, so we can specify in the random slope model, which of these predictors we suspect that will be influenced by the random effect (participant). By specifying the random effect term as + (time|ID) we allow for the effect of time to be different across participants (basically saying that the rate of wound healing can be different from person to person), but restrict the model to predict the same effect for the other two fixed predictors: distance_window and location.

(We could allow for the random slope of distance_window and location as well by adding + (time|ID) + (distance_window|ID) + (location|ID) if you want the random effects to be uncorrelated, or + (time + distance_window + location|ID) if you want all random effects to be correlated). Now let's stick to a simpler model where we only allow for the random slope of time.

```
mod_rep_int = lmer(wound_rating ~ time + distance_window + location +
    (1 | ID), data = data_wound_long)
mod_rep_slope = lmer(wound_rating ~ time + distance_window +
    location + (time | ID), data = data_wound_long)
```

3.4 Comparing models

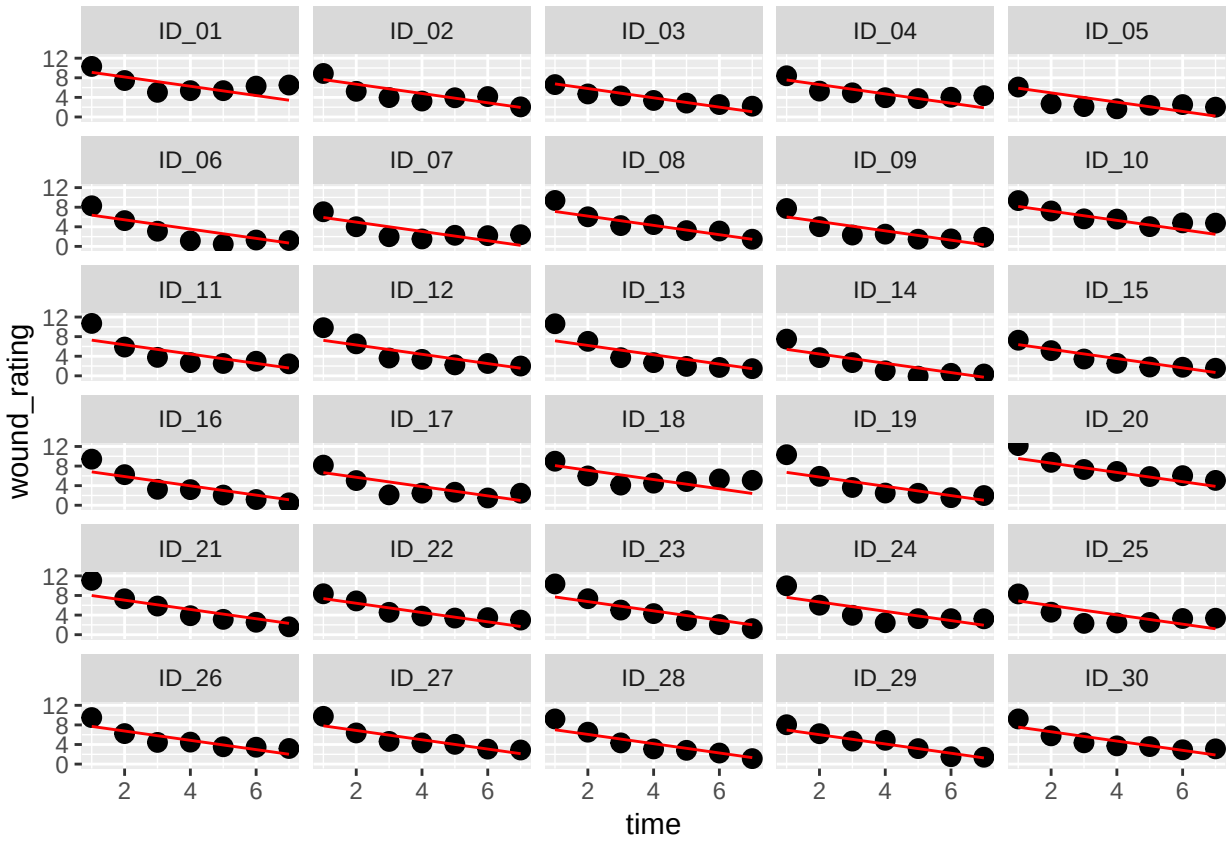
Now let's compare the model predictions of the different random effect models to see which one fits the data better.

First, let's visualize the predictions. For this we will have to save the predicted values into new variables, then, we can visualize the predicted values and the actual observations for each participant separately for both the random intercept and the random slope model.

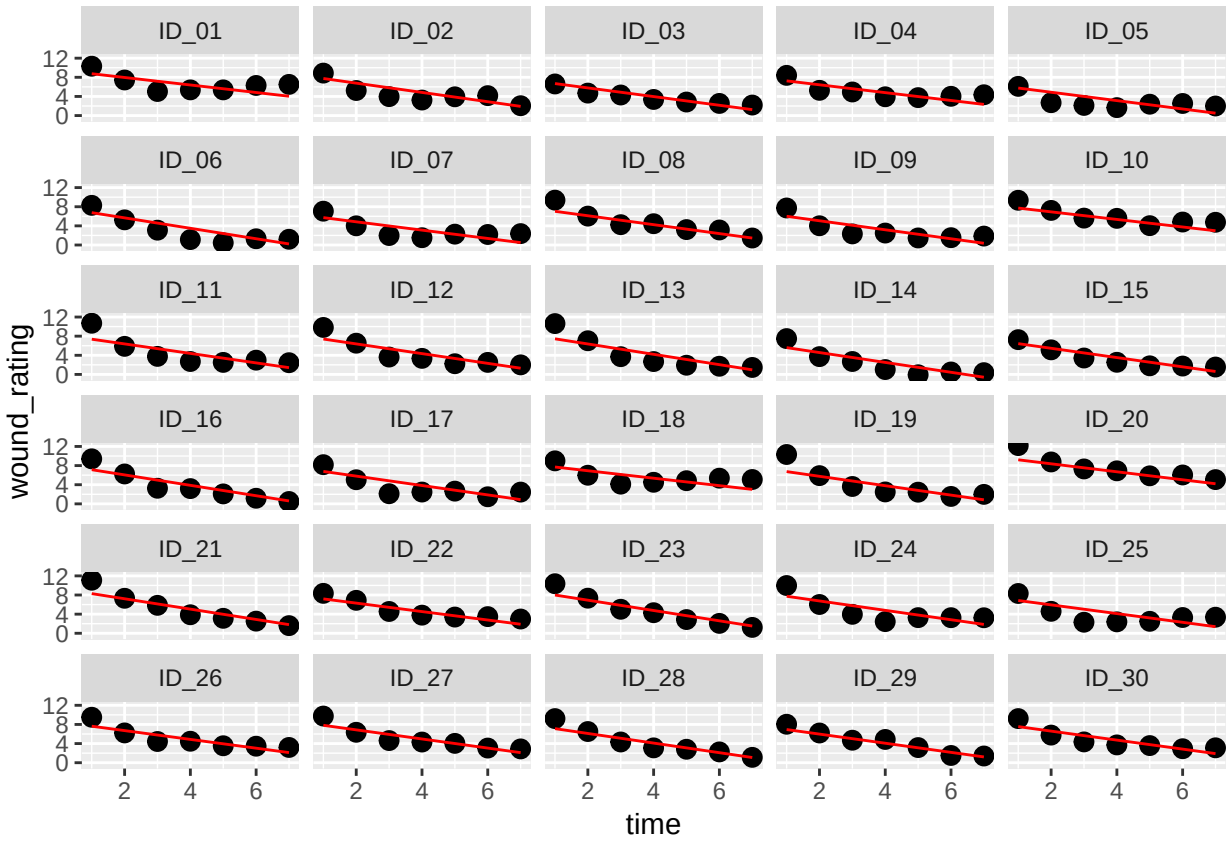
(We create a new copy of the data object so that our long format data can remain unharmed.)

```
data_wound_long_withpreds = data_wound_long
data_wound_long_withpreds$pred_int = predict(mod_rep_int)
data_wound_long_withpreds$pred_slope = predict(mod_rep_slope)

# random intercept model
ggplot(data_wound_long_withpreds, aes(y = wound_rating, x = time,
    group = ID)) + geom_point(size = 3) + geom_line(color = "red",
    aes(y = pred_int, x = time)) + facet_wrap(~ID, ncol = 5)
```




```
# random slope and intercept model
ggplot(data_wound_long_withpreds, aes(y = wound_rating, x = time,
    group = ID)) + geom_point(size = 3) + geom_line(color = "red",
    aes(y = pred_slope, x = time)) + facet_wrap(~ID, ncol = 5)
```



The difference between the predictions of the two models is unremarkable.

Furthermore, we can compare the cAIC of the two models and use the likelihood ratio test with the anova() function to get further information about the model fit of the two models in comparison to wach other.

```
cAIC(mod_rep_int)$caic
```

```
## [1] 757.0182
```

```
cAIC(mod_rep_slope)$caic
```

```
## [1] 757.3522
```

```
anova(mod_rep_int, mod_rep_slope)
```

```
## refitting model(s) with ML (instead of REML)
```

```
## Data: data_wound_long
## Models:
## mod_rep_int: wound_rating ~ time + distance_window + location + (1 | ID)
## mod_rep_slope: wound_rating ~ time + distance_window + location + (time | ID)
##                Df    AIC    BIC  logLik deviance Chisq Chi Df Pr(>Chisq)
## mod_rep_int     6 773.40 793.49 -380.70   761.40
## mod_rep_slope   8 774.82 801.60 -379.41   758.82 2.583      2     0.2749
```

None of these methods indicate a significant difference between the prediction efficiency of the models. So in this particular sample thre is not too much benefit for assuming a different slope of time for each participant. But this does not necesseraly mean that there is no point of using it in another sample. Previous studies and theory needs to be evaluated as well.

For now, without any prior knowledge from the literature, we can continue using the random intercept model.

3.5 Adding the quadratic term of time to the model

While exploring the plots we might notice that there is a non-linear relationship between time and wound rating. It seems that wounds improve fast in the first few days, and the healing is slower in the days after that.

Let's add the quadratic term of time to the model random intercept model tp account for this non-linear relationship.

```
mod_rep_int_quad = lmer(wound_rating ~ time + I(time^2) + distance_window +
    location + (1 | ID), data = data_wound_long)
```

And add the predictions to the new dataframe containing the other predicted values as well.

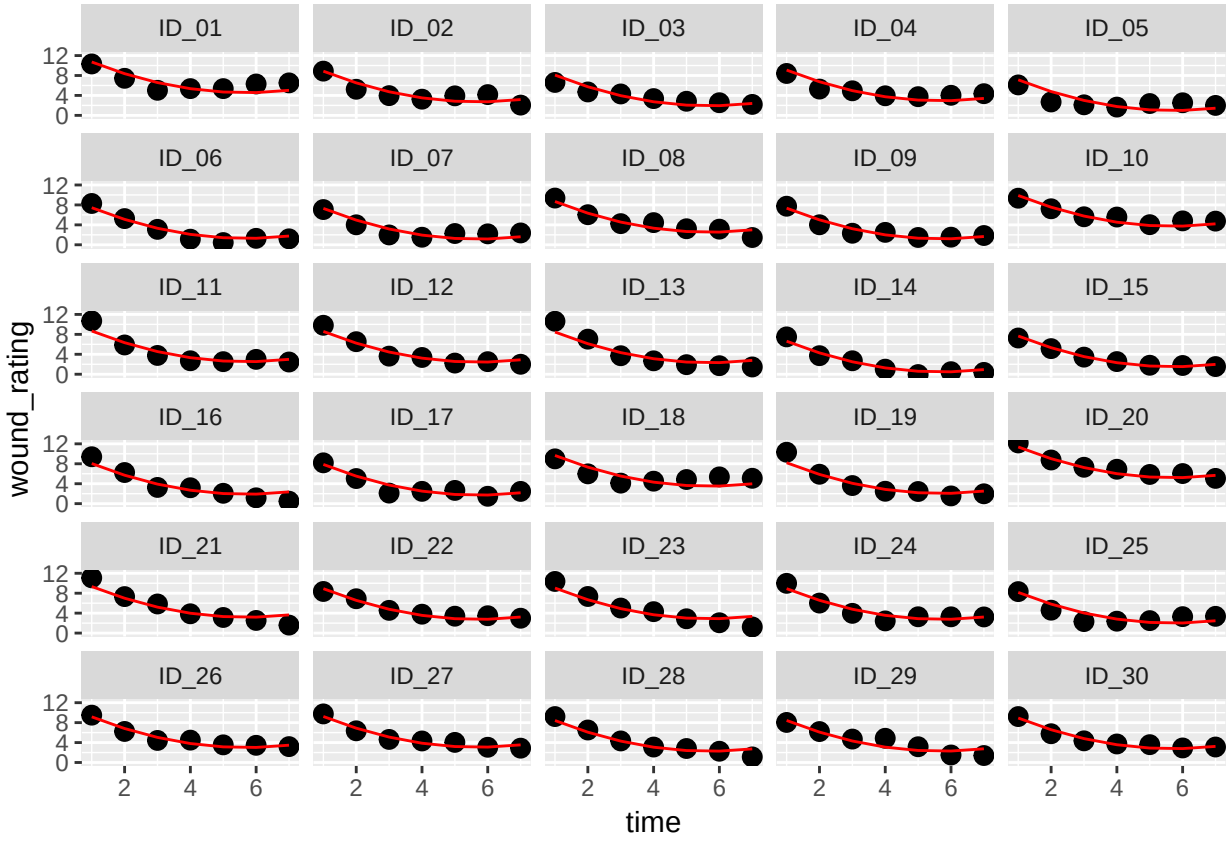
```
data_wound_long_withpreds$pred_int_quad = predict(mod_rep_int_quad)
```

Now we can compare the model fit of the random intercept model containing the quadratic effect of time with the random intercept model without it. As usual, we use visualization, cAIC and the likelihood ratio test.

```
data_wound_long_withpreds$pred_int_quad = predict(mod_rep_int_quad)

plot_quad = ggplot(data_wound_long_withpreds, aes(y = wound_rating,
    x = time, group = ID)) + geom_point(size = 3) + geom_line(color = "red",
    aes(y = pred_int_quad, x = time)) + facet_wrap(~ID, ncol = 5)
```

```
plot_quad
```



```
cAIC(mod_rep_int)$caic
```

```
## [1] 757.0182
```

```
cAIC(mod_rep_int_quad)$caic
```

```
## [1] 583.2994
```

```
anova(mod_rep_int, mod_rep_int_quad)
```

```
## refitting model(s) with ML (instead of REML)
```

```
## Data: data_wound_long
## Models:
## mod_rep_int: wound_rating ~ time + distance_window + location + (1 | ID)
## mod_rep_int_quad: wound_rating ~ time + I(time^2) + distance_window + location +
## mod_rep_int_quad:     (1 | ID)
##                   Df    AIC    BIC  logLik deviance  Chisq Chi Df
## mod_rep_int        6 773.40 793.49 -380.70   761.40
## mod_rep_int_quad   7 621.08 644.51 -303.54   607.08 154.32      1
##                   Pr(>Chisq)
## mod_rep_int
## mod_rep_int_quad  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results indicate that a model taking into account the nonlinear relationship of time and wound rating produces a significantly better fit to the observed data than a model only allowing for a linear trend of time and wound healing.

The fit seems reasonable, so we stop here and decide that this will be our final model.

Since we entered time's quadratic term into the model, we can expect problems with multicollinearity. As seen in the exercise on model diagnostics, we can avoid this problem by centering the variable time, this way removing the correlation of time and time^2.

Let's do this now and refit our model with the centered time and its quadratic term as predictors.

```
data_wound_long_centered_time = data_wound_long
data_wound_long_centered_time$time_centered = data_wound_long_centered_time$time -
    mean(data_wound_long_centered_time$time)


mod_rep_int_quad = lmer(wound_rating ~ time_centered + I(time_centered^2) +
    distance_window + location + (1 | ID), data = data_wound_long_centered_time)
```

Now we can request the reportable results the same way we did in the previous exercise.

```
# Marginal R squared
r2beta(mod_rep_int_quad, method = "nsj", data = data_wound_long_centered_time)
```

```
##                Effect   Rsq upper.CL lower.CL
## 1               Model 0.763    0.805    0.717
## 2      time_centered 0.700    0.752    0.643
## 3 I(time_centered^2) 0.377    0.470    0.284
## 4    distance_window 0.130    0.220    0.058
## 5 locationsouth_wing 0.103    0.189    0.039
```

```
# Conditional AIC
cAIC(mod_rep_int_quad)$caic
```

```
## [1] 583.2994
```

```
# Model coefficients
summary(mod_rep_int_quad)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula:
## wound_rating ~ time_centered + I(time_centered^2) + distance_window +
##     location + (1 | ID)
##    Data: data_wound_long_centered_time
##
## REML criterion at convergence: 625.7
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.34290 -0.61819  0.02384  0.61744  2.40352
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  ID       (Intercept) 0.7380   0.8591
##  Residual             0.8126   0.9015
## Number of obs: 210, groups:  ID, 30
##
## Fixed effects:
##                     Estimate Std. Error t value
## (Intercept)          4.95676    0.50887   9.741
## time_centered       -0.94826    0.03110 -30.487
## I(time_centered^2)   0.27908    0.01796  15.541
## distance_window     -0.09016    0.03174  -2.840
## locationsouth_wing -0.84297    0.33787  -2.495
##
## Correlation of Fixed Effects:
##             (Intr) tm_cnt I(_^2) dstnc_
## time_centrd  0.000
## I(tm_cnt^2) -0.141  0.000
## distnc_wndw -0.872  0.000  0.000
## lctnsth_wng -0.288  0.000  0.000 -0.049
```

```
# Confidence intervals for the coefficients
confint(mod_rep_int_quad)
```

```
## Computing profile confidence intervals ...
```

```
##                          2.5 %      97.5 %
## .sig01              0.6048206  1.10514198
## .sigma              0.8112449  0.99766032
## (Intercept)         3.9797744  5.93373628
## time_centered      -1.0092087 -0.88731515
## I(time_centered^2)  0.2438917  0.31426699
## distance_window    -0.1511234 -0.02919934
## locationsouth_wing -1.4918557 -0.19408605
```

```
# standardized Betas
stdCoef.merMod(mod_rep_int_quad)
```

```
##                       stdcoef      stdse
## (Intercept)         0.0000000 0.00000000
```

```
## time_centered      -0.7486918 0.02455740
## I(time_centered^2)  0.3816481 0.02455740
## distance_window    -0.1894277 0.06669033
## locationsouth_wing -0.1663901 0.06669033
```

As always, you will need to run model diagnostics before reporting your final results. The next exercis will conver this topic.