

# Kevert modellek - Ismetelt mereses elemzesek

*Zoltan Kekecs*

*19 november 2019*

## Contents

<b>1</b>	<b>Absztrakt</b>	<b>2</b>
<b>2</b>	<b>Adatmenedzsment es leiro statisztikak</b>	<b>2</b>
2.1	Package-ek betoltese . . . . .	2
2.2	Sajat funkcio . . . . .	2
2.3	Sebgyogyulas adat betoltese . . . . .	2
2.4	Adatellenorzes . . . . .	3
<b>3</b>	<b>Repeated measures analysis using linear mixed models</b>	<b>4</b>
3.1	Exploring clustering in the data . . . . .	4
3.2	Reshape dataframe . . . . .	5
3.3	Building the linear mixed model . . . . .	6
3.4	Comparing models . . . . .	6
3.5	Adding the quadratic term of days to the model . . . . .	9

# 1 Absztrakt

Ez a gyakorlat az ismertetett meréses elemzésekkel foglalkozik. Amikor az elemzésünkben ugyan attól a vizsgálati személytől több adat is szerepel ugyan abból a változóból, ismertetett meréses elemzést végzünk (pl. a személy kezelése előtti és utáni depresszió szintje).

## 2 Adatmenedzsment és leíró statisztikák

### 2.1 Package-ek betöltése

A következő package-ekre lesz szükség a gyakorláthoz:

```
library(psych) # for describe
library(tidyverse) # for tidy code and ggplot
library(cAIC4) # for cAIC
library(r2glmm) # for r2beta
```

### 2.2 Saját funkció

Ezzel a funkcióval kinyerhetjük a standardizált Beta együtthatót a kevert modellekből. Ez a funkció innen lett átveve: <https://stackoverflow.com/questions/25142901/standardized-coefficients-for-lmer-model>

```
stdCoef.lmerMod <- 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 Sebgyógyulás adat betöltése

A gyakorlat során a sebgyógyulás adatbázissal fogunk dolgozni. Ez egy szimulált adatbázis, ami a műtet során ejtett bemetszések gyógyulását vizsgáljuk annak függvényében, hogy a betegek agya milyen közel van az ablakhoz, és hogy mennyi napfény éri őket a felépülés időszak alatt. Mondjuk, hogy az elméletünk, hogy a kórházi betegeknek szükségük van a külvilággal való kapcsolatra ahhoz, hogy gyorsan felépüljenek. Egy ablak, ami a szabadba nyílik, megteremtheti ezt a kapcsolatot a külvilággal, ezért a kutatásunk azt vizsgálja, hogy befolyásolja-e a sebgyógyulás mértékét az, hogy a személynek milyen közel van az agya a legközelebbi ablakhoz. Az elmélet egy változata azt állítja, hogy az ablak nem csak a külvilággal való szorosabb kapcsolat megteremtésén keresztül vezet gyorsabb gyógyuláshoz, hanem azon keresztül is, hogy több napfényt enged a szobába, és az elmélet szerint a napfény is jótékony hatással van a gyógyulásra.

Változók az adatbázisban:

- ID: azonosító kód
- day\_1, day\_2, ..., day\_7: A műtet utáni 1-7. napon egy orvos megvizsgálta a seb bemetszési sebeit, és értékelte azokat egy standardizált seb-állapot értékekkel. Minél nagyobb ez az érték, annál nagyobb vagy rosszabb állapotú a seb (pl. gyulladt). Minden személynek mind a 7 napra külön seb-állapot értékek tartoznak.
- distance\_window: A személy agyához legközelebbi ablak távolsága az agytól méterben.
- location: A kórházi szárny, ahol a beteg agya van. Kettszintű faktor változó: szintjei "north wing" és "south wing" (a "south wing"-ben több napfény éri a betegeket, ez a változó azért fontos).

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

```
# assign ID and location as factors
data_wound = data_wound %>% mutate(ID = factor(ID), location = factor(location))
```

## 2.4 Adatellenorzes

Vizsgáljuk meg az adattáblát a View(), describe(), és table() funkciók segítségével. Fontos, hogy az adattáblában jelenleg minden adata mit egy adott személytől gyűjtöttek egy sorban található. Vizualizálhatjuk is az adatokat.

```
View(data_wound)

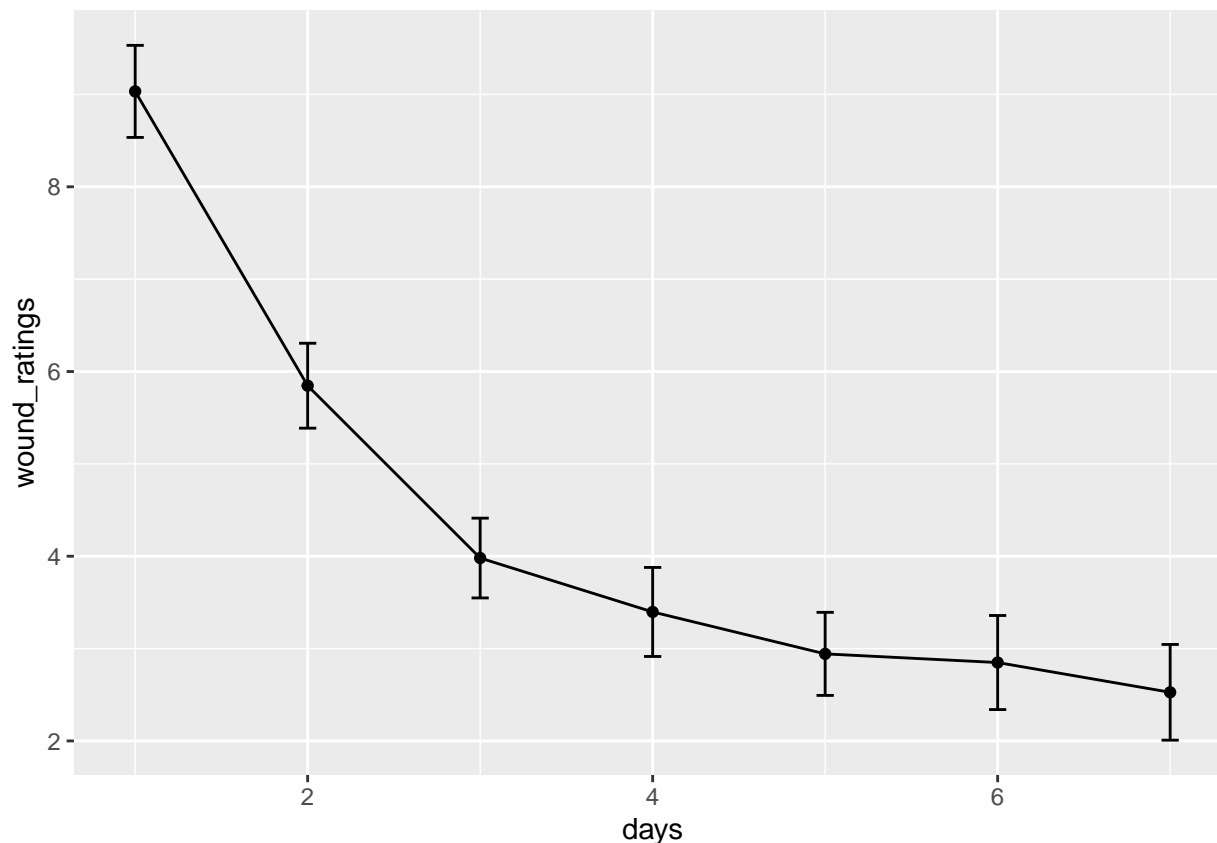
# descriptives
describe(data_wound)
table(data_wound$location)
```

Az alábbi ábra elkészítéséhez először a day1-day7 változókban külön-külön kiszámoljuk az átlagokat és a standard hibát, majd a standard hibát megszorozva 1.96-al megkapjuk a konfidencia intervallumot. Vegül mindezt egy új adat objektumba tesszük, és a geom\_errorbar, geom\_point, és geom\_line segítségével vizualizáljuk. Látható hogy a seb állapot értékek egyre csökken ahogyan telnek a napok.

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

# explore change over time
wound_ratings = describe(data_wound[, repeated_variables])$mean
repeated_CIs = describe(data_wound[, repeated_variables])$se *
  1.96
days = as.numeric(as.factor(repeated_variables))
days = as.data.frame(days)
data_for_plot = cbind(days, wound_ratings, repeated_CIs)

data_for_plot %>% ggplot() + aes(x = days, y = wound_ratings) +
  geom_errorbar(aes(ymin = wound_ratings - repeated_CIs, ymax = wound_ratings +
    repeated_CIs), width = 0.1) + geom_point() + geom_line()
```



### 3 Repeated measures analysis using linear mixed models

#### 3.1 Exploring clustering in the data

Now let's look at the relationship 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.

```
# correlation of repeated variables
```

```
cor(data_wound[, repeated_variables])
```

```
##      day_1    day_2    day_3    day_4    day_5    day_6
## day_1 1.000000 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 `gather()` function from the `tidyr` package. In this function we specify the variable name in which will index the repeated observations (in our case we will call this “days”), and the variable name in which we want to gather all the observations made on the same variable (in our case we will call this “wound rating”). Finally, we will have to specify the variable names in which the data is currently located in the wide format. (By saying: `day_1:day_7` we say that the variables are the column names between the `day_1` and `day_7` columns). Using the `arrange()` function we sort the data table based on the column “ID”. This is not important, but it is easier to see the logic behind a long format file if we look at this sorted dataset.

(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 = data_wound %>% gather(key = days, value = wound_rating,
  day_1:day_7) %>% arrange(ID)
```

```
data_wound_long
```

```
## # A tibble: 210 x 5
##   ID    distance_window location    days  wound_rating
##   <fct>          <dbl> <fct>    <chr>      <dbl>
## 1 ID_01          6.18 north_wing day_1        10.3
## 2 ID_01          6.18 north_wing day_2         7.44
## 3 ID_01          6.18 north_wing day_3         5.03
## 4 ID_01          6.18 north_wing day_4         5.37
## 5 ID_01          6.18 north_wing day_5         5.37
## 6 ID_01          6.18 north_wing day_6         6.3
## 7 ID_01          6.18 north_wing day_7         6.52
## 8 ID_02          7.21 north_wing day_1         8.88
## 9 ID_02          7.21 north_wing day_2         5.24
## 10 ID_02         7.21 north_wing day_3         3.96
## # ... with 200 more rows
```

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 'days' variable now contains the names of the repeated measures variables from the wide format ('day\_1', 'day\_2' etc.). We will simplify this by simply using numbers 1-7 here to make them a numerical variable which is easier to deal with statistically. The easiest to do this is by using the `mutate()` and `recode()` functions.

```
# change the days variable to a numerical vector
data_wound_long = data_wound_long %>% mutate(days = recode(days,
  day_1 = 1, day_2 = 2, day_3 = 3, day_4 = 4, day_5 = 5, day_6 = 6,
  day_7 = 7))
```

Let's explore how this new dataframe looks like.

```
View(data_wound_long)
```

### 3.3 Building the linear mixed model

Now that we have our dataframe in an appropriate format, 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 `days`, `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 (`days`, `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 `+(days|ID)` we allow for the effect of `days` 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 `+(days|ID) + (distance_window|ID) + (location|ID)` if you want the random effects to be uncorrelated, or `+(days + 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 `days`.

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

```
## Warning in checkConv(attr("opt", "derivs"), opt$par, ctrl =
## control(checkConv, : Model failed to converge with max|grad| = 0.00253673
## (tol = 0.002, component 1)
```

### 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.

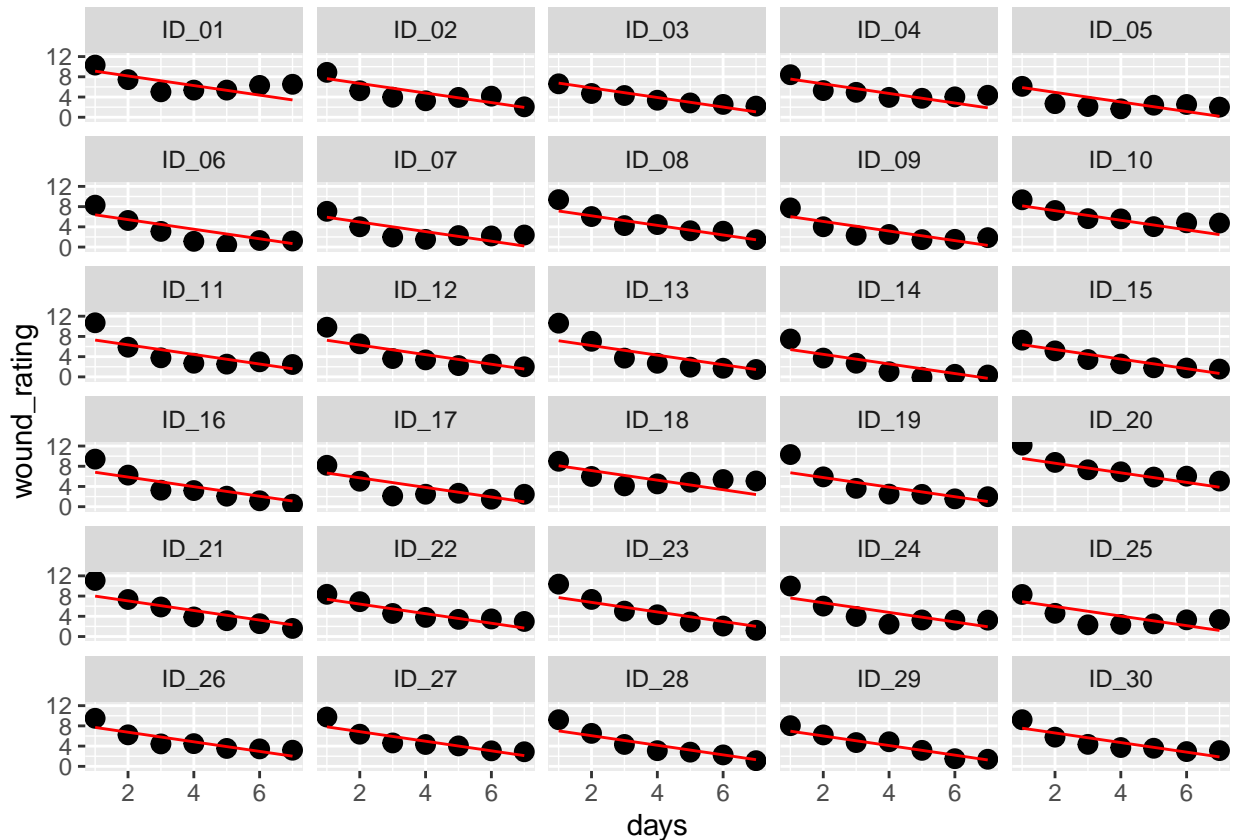
First, let's visualize the predictions. For this we will have to save the predicted values into new variables, then, we can visualize the predicted values and the actual observations for each participant separately for both the random intercept and the random slope model.

(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)
```

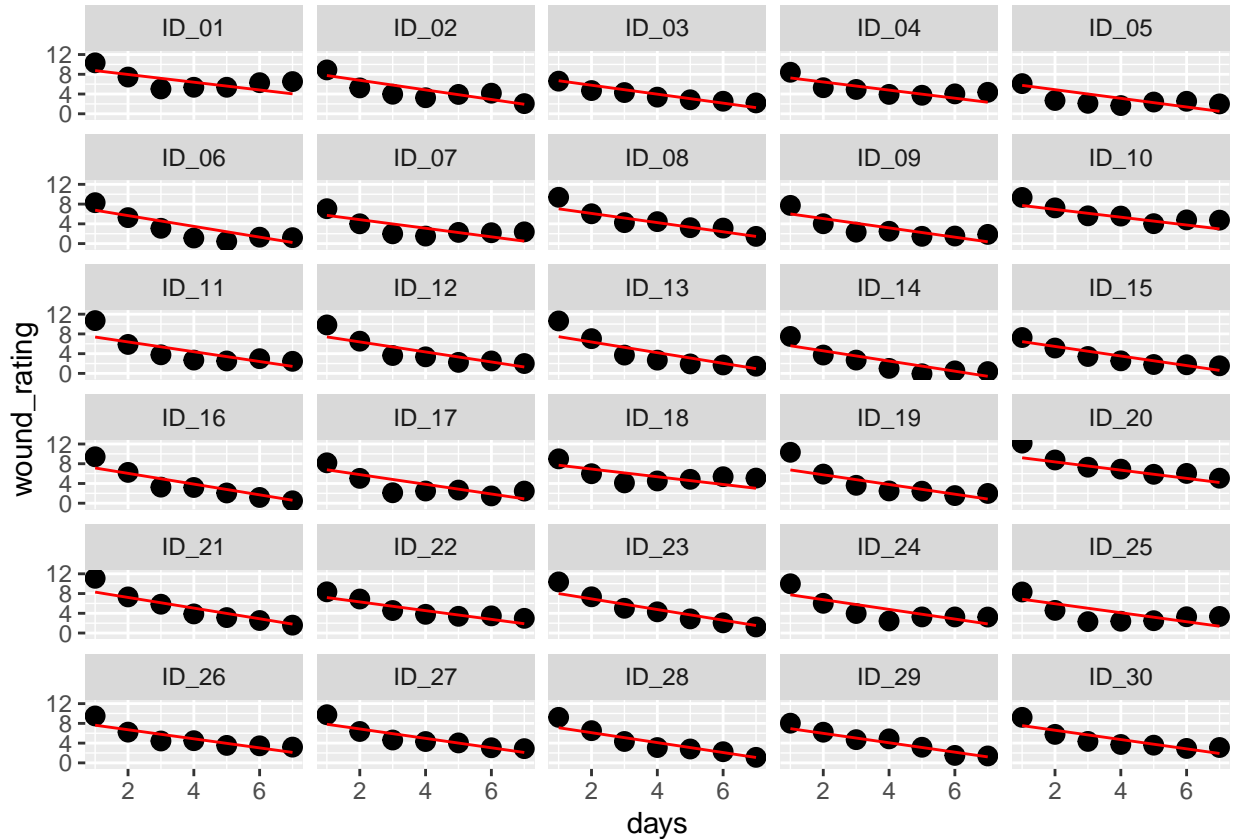
*# random intercept model*

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



*# random slope and intercept model*

```
ggplot(data_wound_long_withpreds, aes(y = wound_rating, x = days,
  group = ID)) + geom_point(size = 3) + geom_line(color = "red",
  aes(y = pred_slope, x = days)) + 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 each other.

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

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

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

```
## [1] 757.3564
```

```
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 ~ days + distance_window + location + (1 | ID)
```

```
## mod_rep_slope: wound_rating ~ days + distance_window + location + (days | 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 there is not too much benefit for assuming a different slope of days for each participant. But this does not necessarily 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 days to the model

While exploring the plots we might notice that there is a non-linear relationship between days 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 days to the model random intercept model to account for this non-linear relationship.

```
mod_rep_int_quad = lmer(wound_rating ~ days + I(days^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.

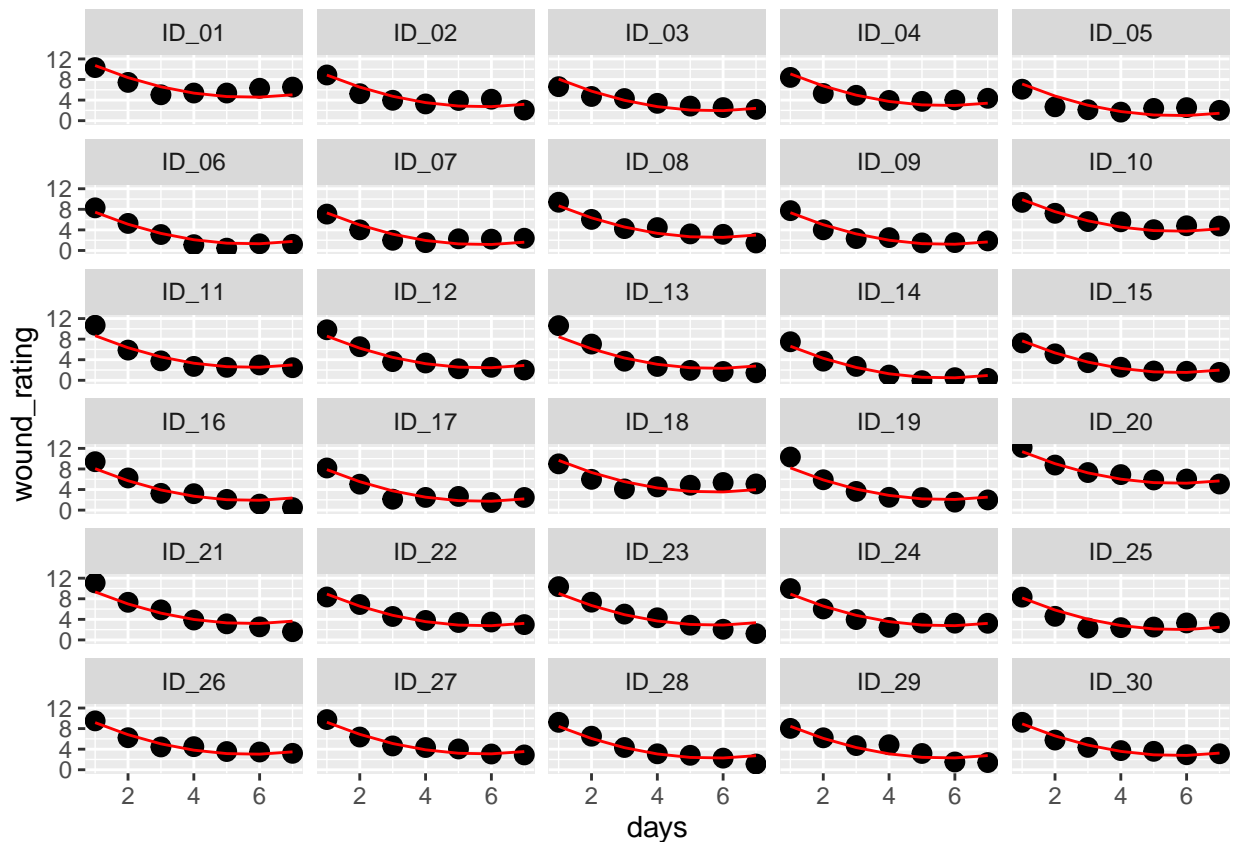
```
data_wound_long_withpreds$pred_int_quad = predict(mod_rep_int_quad)
```

Now we can compare the model fit of the random intercept model containing the quadratic effect of days 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 = days, group = ID)) + geom_point(size = 3) + geom_line(color = "red",  
  aes(y = pred_int_quad, x = days)) + 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 ~ days + distance_window + location + (1 | ID)
## mod_rep_int_quad: wound_rating ~ days + I(days^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 days and wound rating produces a significantly better fit to the observed data than a model only allowing for a linear trend of days and wound healing.

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

Since we entered days'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 days, this way removing the correlation of days and days<sup>2</sup>.

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

```

data_wound_long_centered_days = data_wound_long
data_wound_long_centered_days$days_centered = data_wound_long_centered_days$days -
  mean(data_wound_long_centered_days$days)

mod_rep_int_quad = lmer(wound_rating ~ days_centered + I(days_centered^2) +
  distance_window + location + (1 | ID), data = data_wound_long_centered_days)

```

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_days)

##           Effect    Rsq upper.CL lower.CL
## 1           Model 0.763    0.805    0.717
## 2    days_centered 0.700    0.752    0.643
## 3 I(days_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 ~ days_centered + I(days_centered^2) + distance_window +
## location + (1 | ID)
## Data: data_wound_long_centered_days
##
## 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
## days_centered -0.94826    0.03110 -30.487
## I(days_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) dys_cn I(_^2) dstnc_
## days_centrd  0.000
## I(dys_cn^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
## days_centered -1.0092087 -0.88731515
## I(days_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 0.0000000
```

```
## days_centered      -0.7486918  0.02455740
## I(days_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 exercise will cover this topic.

---

### Practice

Load the surgical pain dataset.

This dataset contains information about pain after surgery, and a number of variables which may be connected to surgical pain.

Variables:

- ID: participant ID
- pain1, pain2, pain3, pain4: In this dataset pain was measured on four consecutive days after surgery using a 0-10 visual analog scale.
- sex: sex reported by the participant
- STAI\_trait: score on the State Trait Anxiety Inventory - t form, which reflects trait anxiety
- pain\_cat: pain catastrophising
- cortisol\_serum; cortisol\_saliva: cortisol is a stress hormone. This was measured on the day of surgery (after surgery) at the same time when the first pain rating was taken. Cortisol was measured from blood and also from the saliva.
- mindfulness: trait mindfulness of the participant measured by a mindfulness questionnaire
- weight: weight in kilograms
- IQ: IQ score on an IQ test one week before surgery
- household\_income: household income in USD

Practice tasks:

1. Load the dataset (a .csv file) from this link: <https://tinyurl.com/data-pain1>.
  2. Convert the data to long format (you can use the `gather()` function or the `melt()` function to do this) so that each observation/repeated measure is in a different row.
  3. Build a linear mixed model to predict as much of the variance in postoperative pain as possible. (You can use any variable as a fixed predictor that makes sense for you to predict postoperative pain.) Since data is clustered within participants, include the random effect of participant ID in the model.
  4. Experiment with both random intercept and random slope models, and compare them using cAIC.
  5. Build a random intercept and a random slope model where the only fixed predictor is time (days after surgery). Visualize the regression lines produced by these two models for each participant separately, and compare their fit to the observations. Is there a significant gain in allowing for random slope of time?
  6. Answer the same question using cAIC.
  7. What is the marginal  $R^2$  for the random intercept model? Based on the confidence interval, is this model better than the null model at predicting pain?
-