Exercise 17 - Model diagnostics of mixed models

Zoltan Kekecs

27 november 2018

Contents

1 Abstract

The aim of this exercise is to learn how to perform model diagnostics and checking assumptions of linear mixed models.

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 pairs.panels
library(ggplot2)  # for ggplot
library(reshape2)  # for melt function
library(influence.ME)  # for influence (this will also load the lme4 package)
library(lattice)  # for qqmath
```

2.2 Wound healing data

In this exercise we will work with the wound healing data we used in the previous exercise.

[*The following description is the same as in the previous exercise*]

This is 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).

We will run the model diagnostics on the model we considered final in the last exercise. In this model we predicted wound rating as an outcome using the fixed effect predictors of time, time^2, distance_window, and location, and we allowed for a random intercept for each participant, but did not include a random slope.

So here we start with reproducing the same dataframe and model what we had last time.

2.3 Loading and managing data

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/

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

```
    "day_6", "day_7")

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

We create a new data object where the data is restructured to a 'long format'.

```
data_wound_long = melt(data_wound, measure.vars = repeated_variables,
    variable.name = "time", value.name = "wound_rating")
data_wound_long = data_wound_long[order(data_wound_long[, "ID"]),
    ]
data_wound_long$time = as.numeric(data_wound_long$time)
```

And we center the variable 'time' to avoid problems with multicollinearity.

```
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)
```

We also save the sqared time as a new variable. In effect this is one of our fixed effect predictors, and this will save us time later during the model diagnostics. We will call this new variable time_centered_2.

```
data_wound_long_centered_time$time_centered_2 = data_wound_long_centered_time$time_centered^2
```

2.4 building the model

Now we build the final model from the previous exercise. We also create a copy of our data object and save the residuals in a variable we call resid.

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


data_wound_long_with_resid = data_wound_long_centered_time
data_wound_long_with_resid$resid = residuals(mod_rep_int_quad)
```

3 Model diagnostics for linear mixed models

3.1 Assumptions of linear mixed models

Remember that the assumptions for linear models were the following:

- **Normality**: The residuals of the model must be normally distributed
- **Linearity**: Ther relationship between the outcome variable and the predictor(s) must be linear
- **Homoscedasticity**: The variance of the residuals are similar at all values of the predictor(s)
- **No Multicollinearity**: None of the predictors can be linearly determined by the other predictor(s)

The same assumptions need to be statisfied in the case of linear mixed models as well. For example, we assume that we have modelled the dependency structure of random effects correctly, that the within-unit residual errors follow normal distributions, and have constant variance. We also assume that random effects (the intercepts and slopes if any) are normally distributed centered around 0.

How to effectively and consistantly check these assumptions is still being worked out by researchers and statisticians. But there are already some trend emerging about how can we do model diagnostics. Because these techniques are constantly evolving, you should check new developments on this topic when you use mixed models a few years from now.

3.2 Influential outliers

First, we need to check our data for influential outliers which have a large impack on our model.

You can check for influential observations, using the influence() function from package influence.ME. Here, the logic is that the model is refit with each observation excluded one-by-one, and the model coefficients are re-calculated for all of these models with the exclusion of a single observation. If one of the observations were influential, we would see a large difference between the coefficients in models with and without that particular observation.

You can get the model coefficients for these leave-one-out models using the influence() function on the model object, and specifying obs = T.

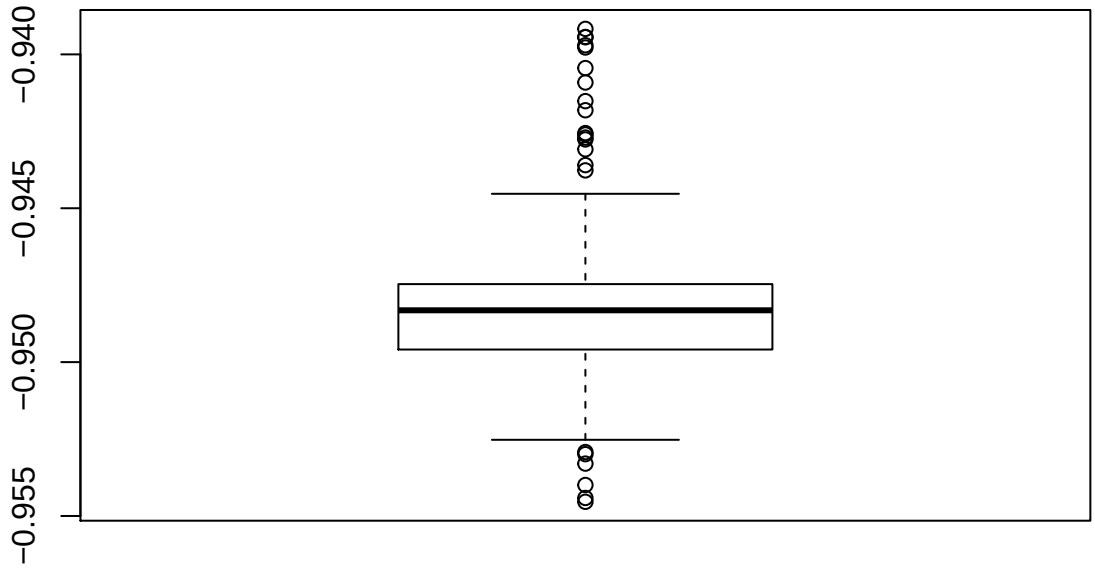
The same can be done to identify influential clusters/units (in this case, units are participants, but in the previous exercise using the bullying data, clusters would be the school classes). Here, instead of obs = T, we need to specify the clustering (random effect) predictor after group =. In our case the variable is named 'ID'.

```
influence_observation = influence(mod_rep_int_quad, obs = T)$alt.fixed  # this can take a minute or so
influence_group = influence(mod_rep_int_quad, group = "ID")$alt.fixed
```

You can explore these leave-one-out coefficients if you plot them on a boxplot. If there is a problem, you should see outliers that are considerably off the average coefficient value.

The plot below shows the leave-one-out coefficients for the predictor 'time'.

```
boxplot(influence_observation[, "time_centered"])
```



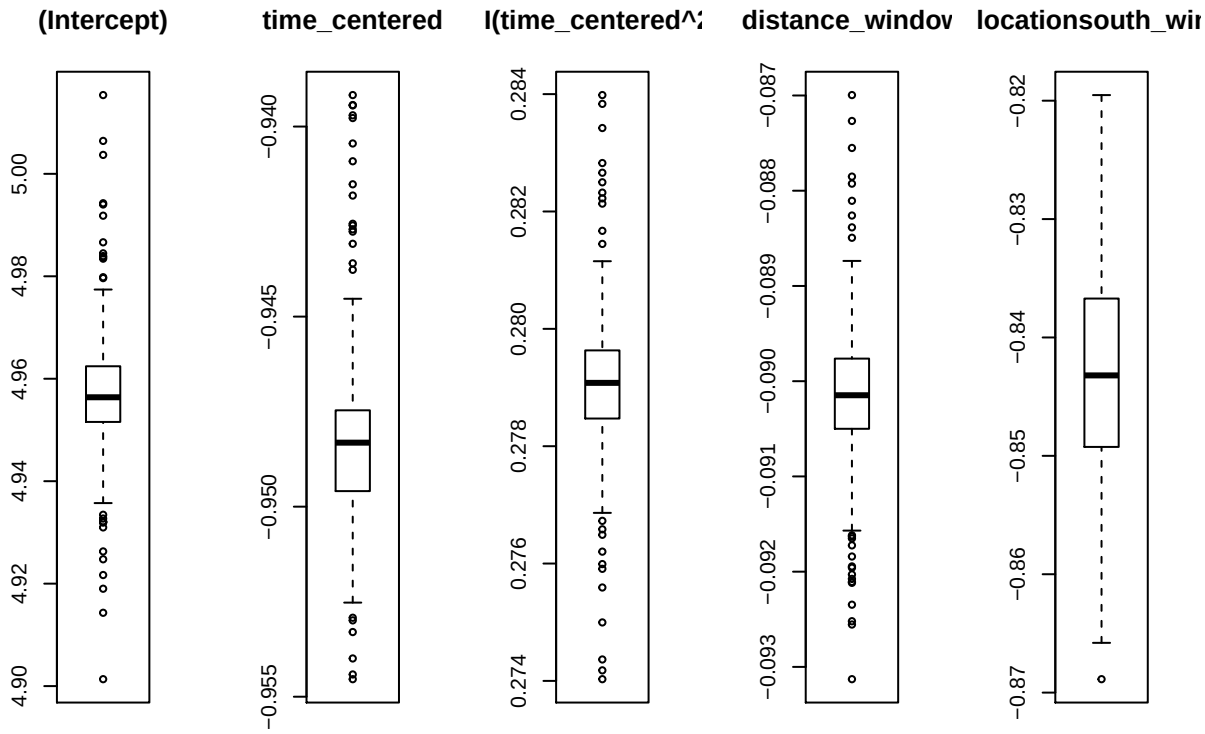
You should check these coefficients for all predictirs. You can do this by using the same code over and over again just changing the variable name.

The code below uses a simple for-loop to display the boxplot for the coefficients of all fixed effect predictors. The par(mfrow=c()) function is used to place multiple plots in the same plot space side-by-side.

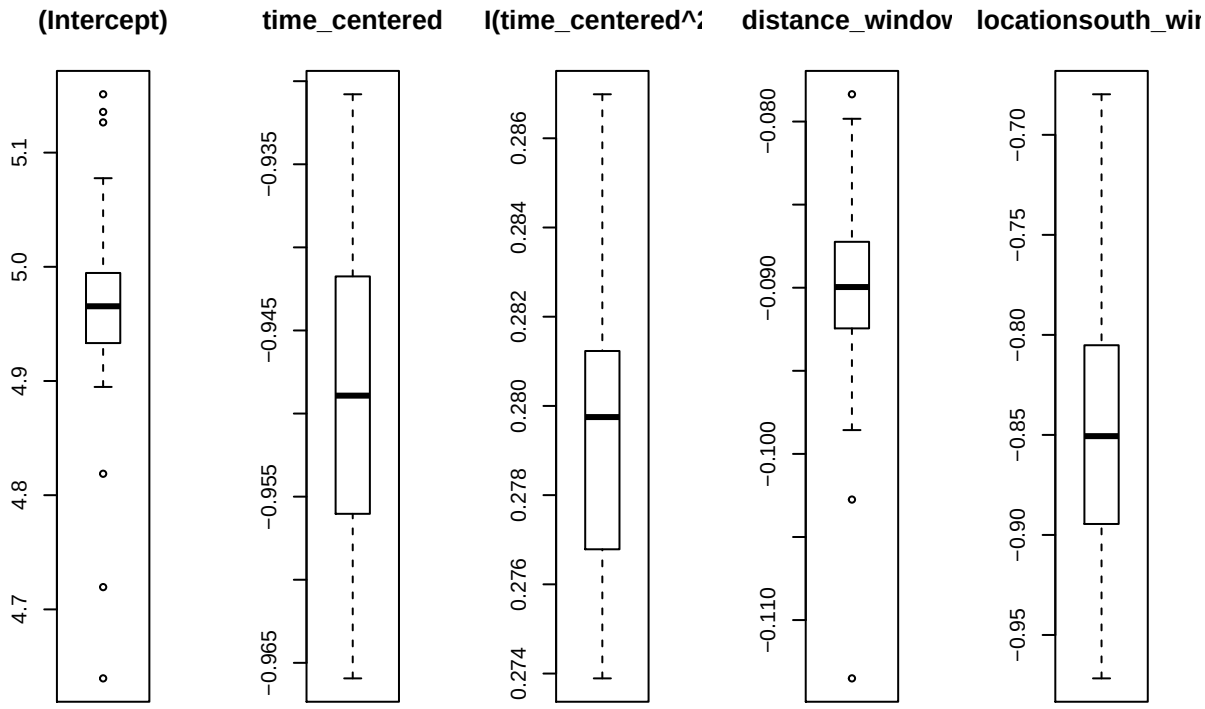
Google for loops and mfrow if you are interested in how they work.

```
pred_names = colnames(influence_group)

par(mfrow = c(1, length(pred_names)))
for (i in 1:length(pred_names)) {
    boxplot(influence_observation[, pred_names[i]], main = pred_names[i])
}
```




```
for (i in 1:length(pred_names)) {
    boxplot(influence_group[, pred_names[i]], main = pred_names[i])
}
```



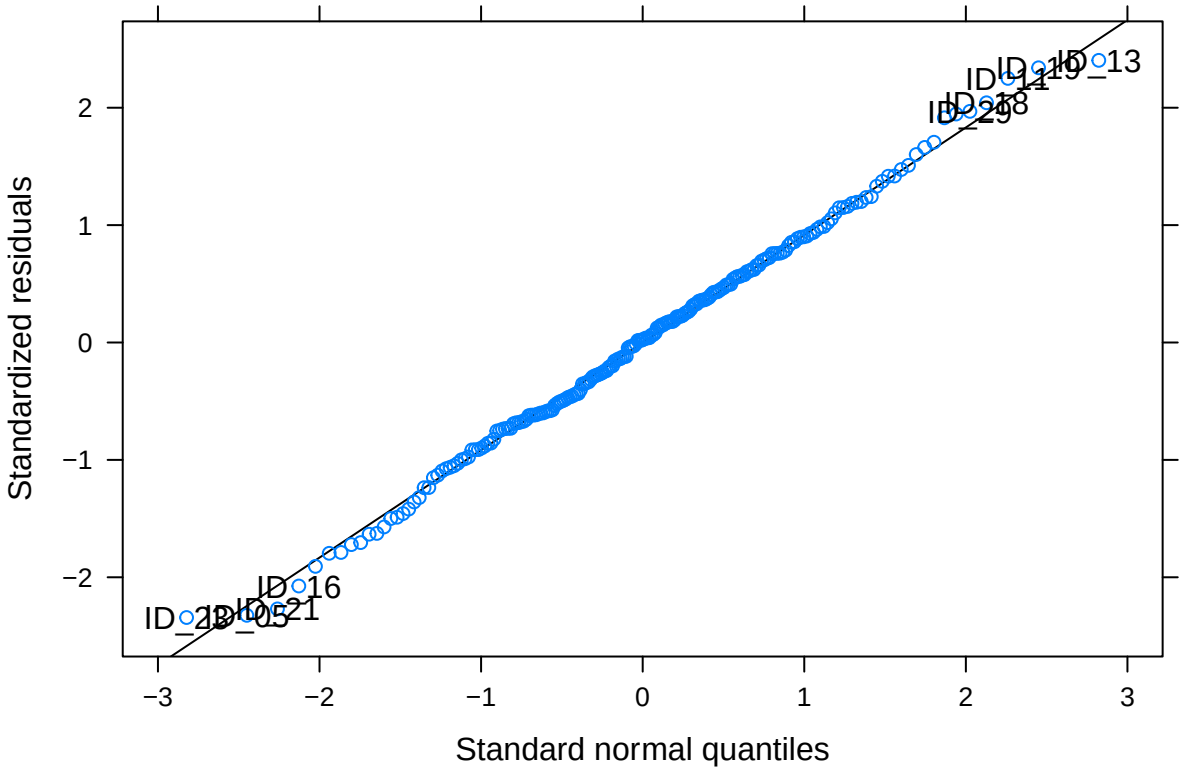

```
par(mfrow = c(1, 1))
```

These plots do not indicate extreme influential cases.

3.3 Normality

You can check if the residuals on the observation level are normally distributed using the qqmath() or the qqnorm() function.

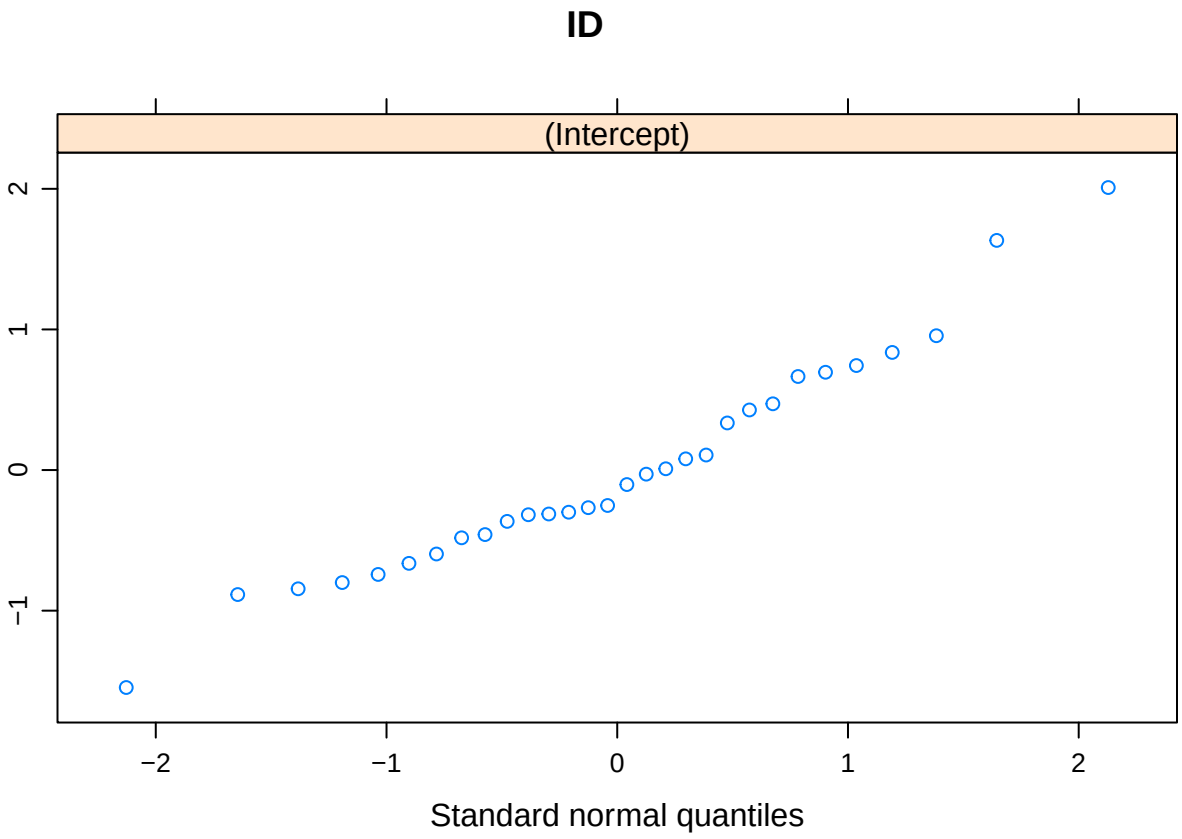
```
qqmath(mod_rep_int_quad, id = 0.05)
```



You can use the same function to draw the QQplot for the random effects. In this case, we only have one random effect, random intercept. The points on the plot should roughly fit on a straight line.

```
qqmath(ranef(mod_rep_int_quad))
```

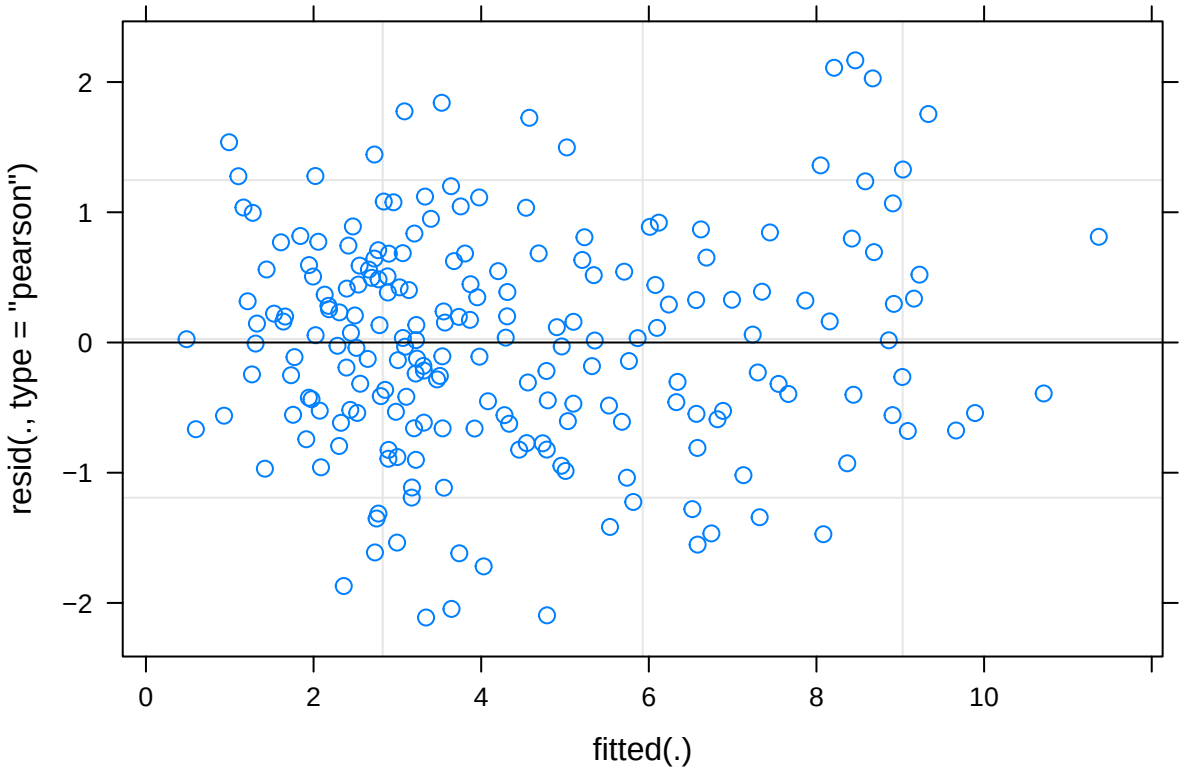
```
## $ID
```



3.4 Linearity

The linearity of the relationship of the fixed effect predictors and the ourcome can be explored by plotting the scatterplot of the standardized residuals and the predicted values.

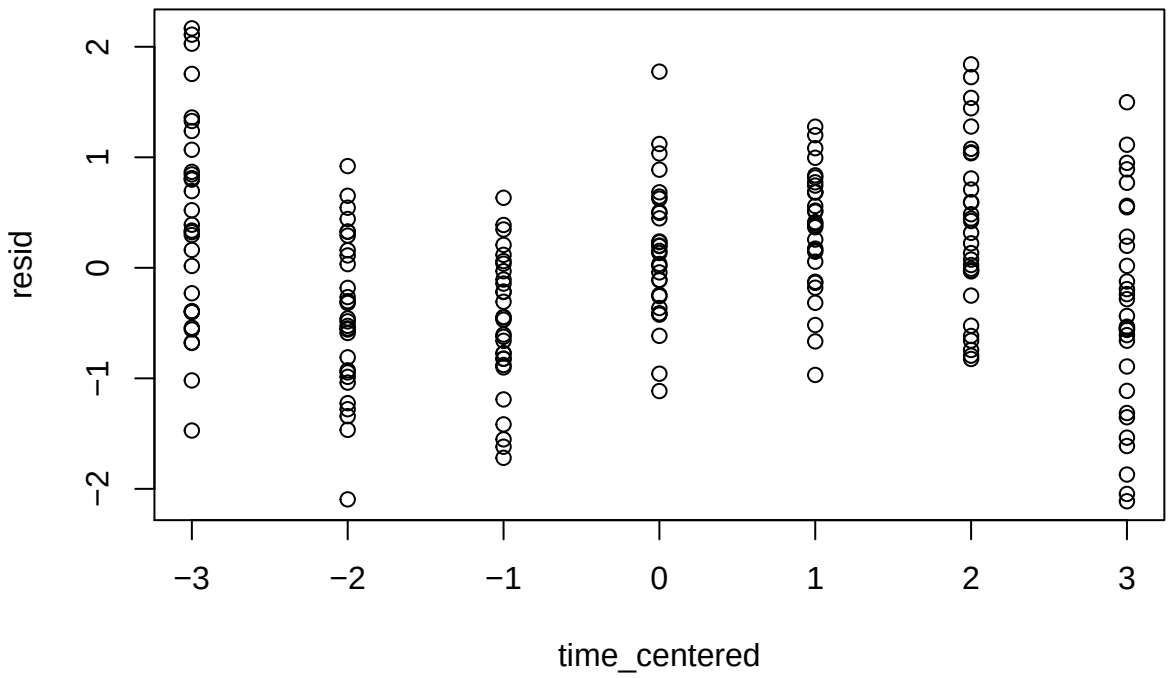
```
plot(mod_rep_int_quad, arg = "pearson")
```



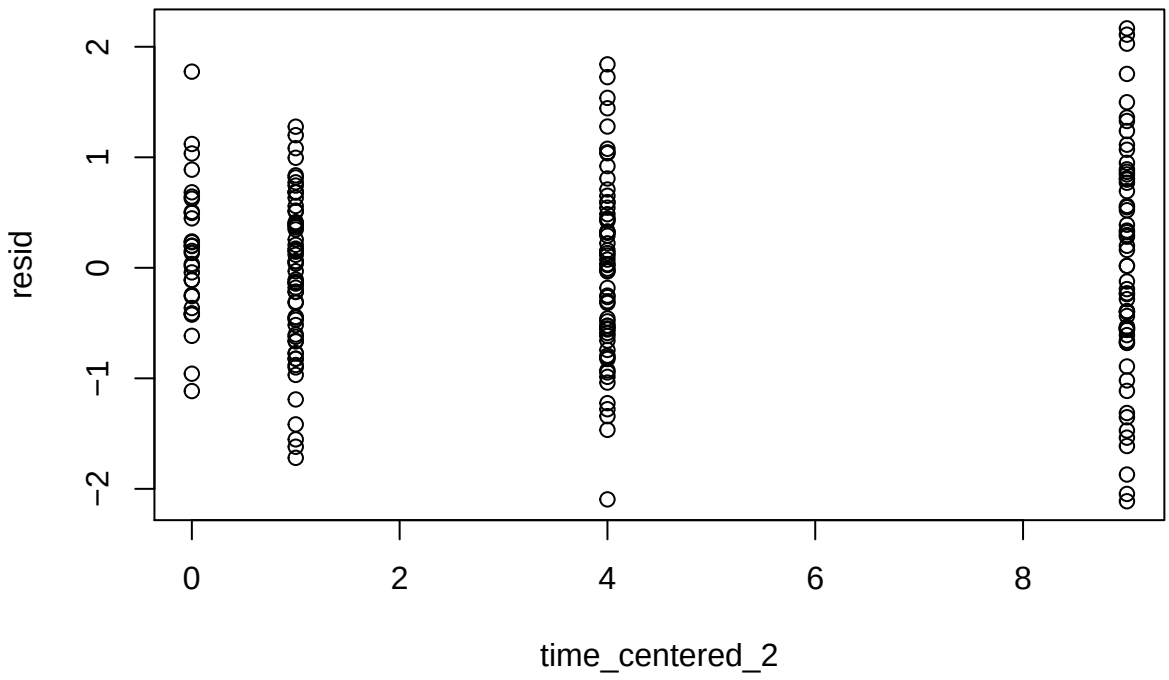
You should also look at the scatterplot of the residuals and the fixed predictors separately.

Here you can notice that time_centered still has a nonlinear relationship with the residuals. Maybe we could decrease this by adding the cubic term of time to the model as well adding + I(time_centered^3) to the model. If this was a real paper, this might be done as an exploratory analysis or this might be discussed in the limitations or the future directions for research sections.

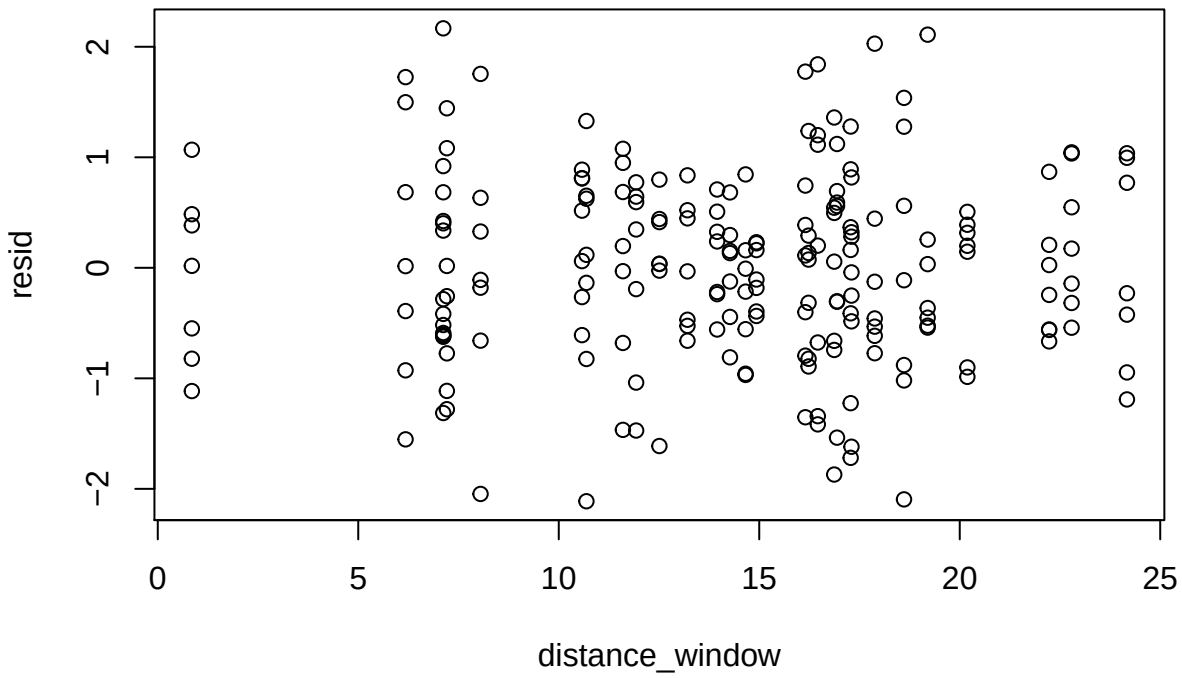
```
plot(resid ~ time_centered, data = data_wound_long_with_resid)
```



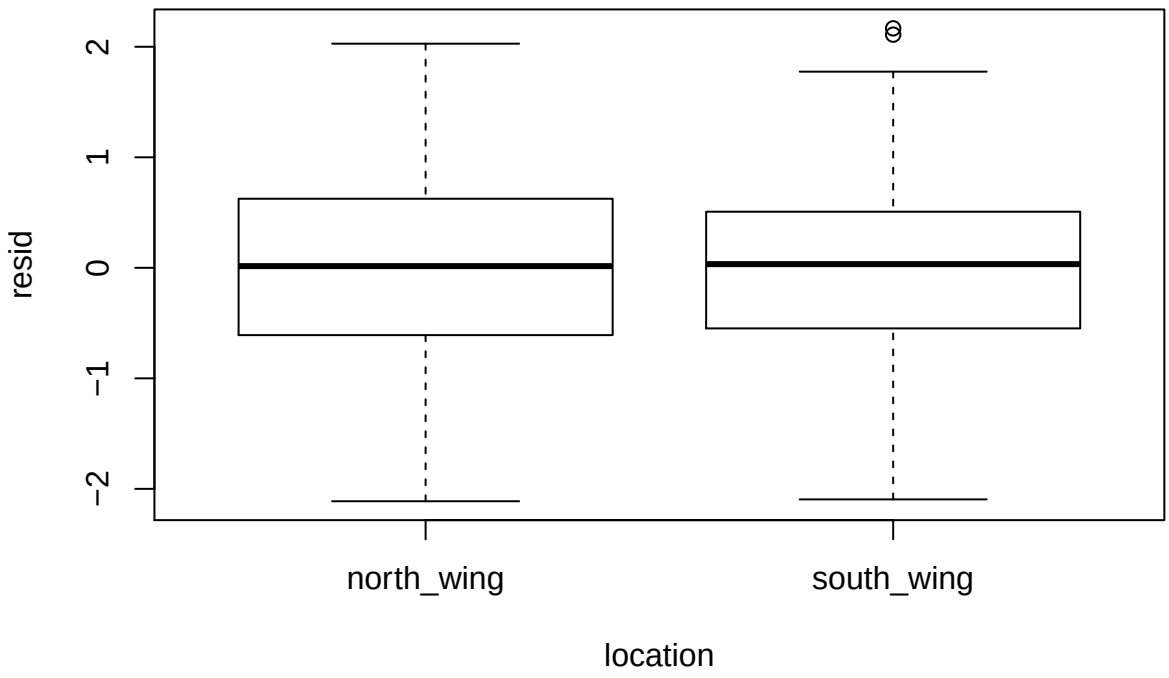

```
plot(resid ~ time_centered_2, data = data_wound_long_with_resid)
```




```
plot(resid ~ distance_window, data = data_wound_long_with_resid)
```



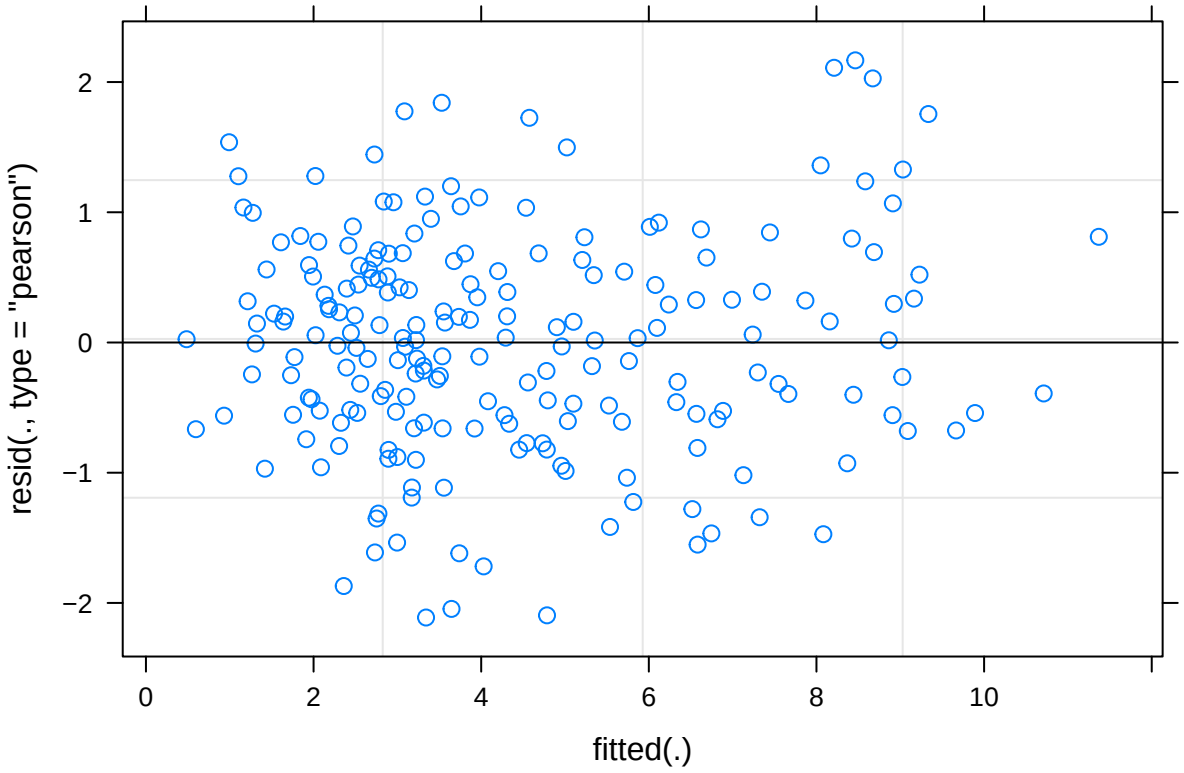

```
plot(resid ~ location, data = data_wound_long_with_resid)
```



3.5 Homoscedasticity

The homogeneity of variances on the observation level can be checked by viewing the same standardized residuals ~ predicted values plot as when checking linearity. Here, a funnel shape would indicate heteroscedasticity, but we don't see that in this plot.

```
plot(mod_rep_int_quad, arg = "pearson")
```



When working with mixed linear models we need to check for homoscedasticity across clusters as well.

We can run a significance test for that by fitting a linear model where we predict the squared residuals with the clustering variable (ID). Check the complete model F-test p-value. If it is < 0.05, heteroscedasticity on the cluster level might be problematic.

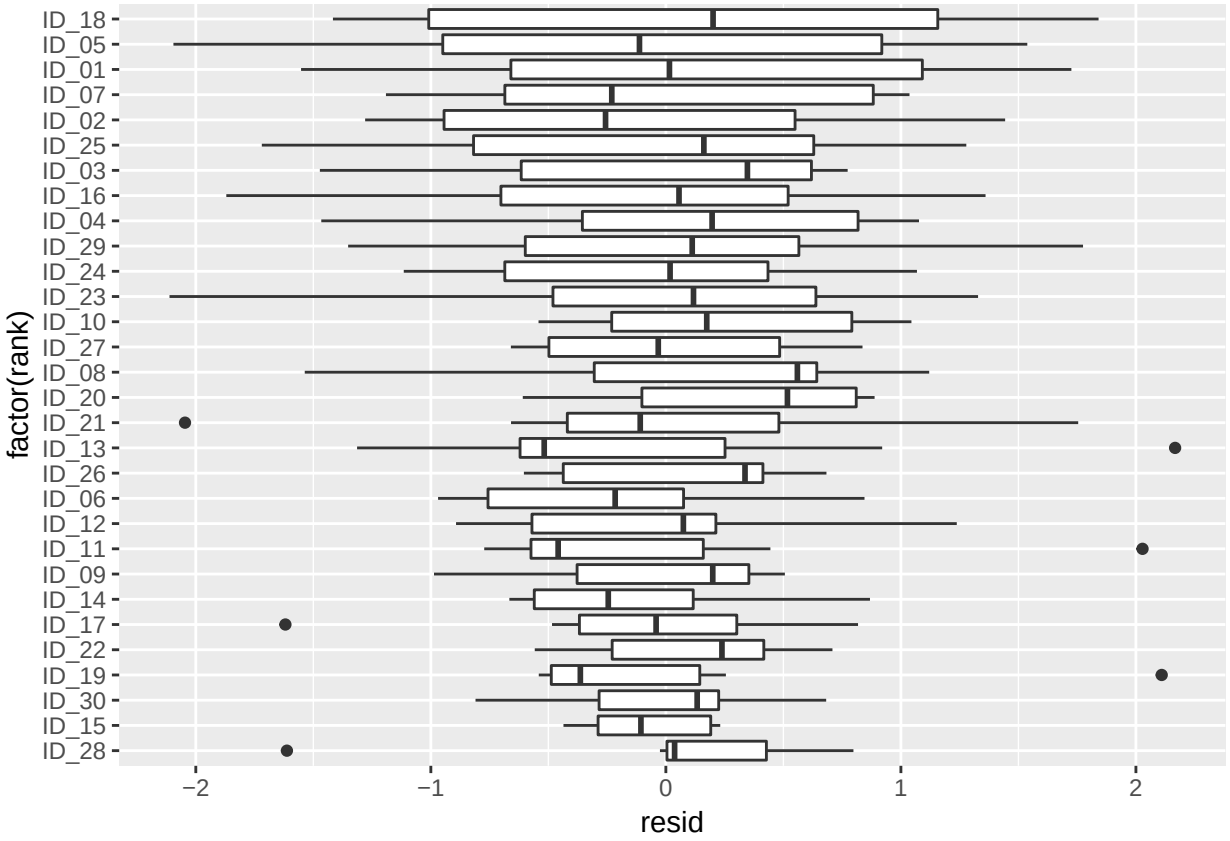
```
homosced_mod = lm(data_wound_long_with_resid$resid^2 ~ data_wound_long_with_resid$ID)
summary(homosced_mod)
```

You can also inspect the cyclone plot. Here we plot the boxplot of the residuals for each participant, and order these boxes according to the variance of the residal (actually the interquartile range). Here we would expect a gradual increase of the variance of the residual from top to buttom. If the increase is not consistent (some clusters have much larger variance than the previous one on the list), we can suspect heteroscedasticity across clusters/units.

```
# caluclate interquartile range within each cluster
IQR_of_residuals_by_participant = sapply(split(data_wound_long_with_resid,
    f = data_wound_long_with_resid$ID), function(x) IQR(x$resid))
# rank ordering them
rank = rank(IQR_of_residuals_by_participant)
# adding rank to the dataframe containing the residuals
data_wound_long_with_resid$rank = rep(rank, each = length(repeated_variables))
# creating a vector of participant IDs ordered based on the
# rank, this will be used as labels
IDforplot = unique(data_wound_long_with_resid$ID[order(data_wound_long_with_resid$rank)])

# create the plot
ggplot(data_wound_long_with_resid, aes(y = resid, x = factor(rank),
```

```
    labels = ID)) + geom_boxplot() + scale_x_discrete(labels = IDforplot) +
    coord_flip()
```



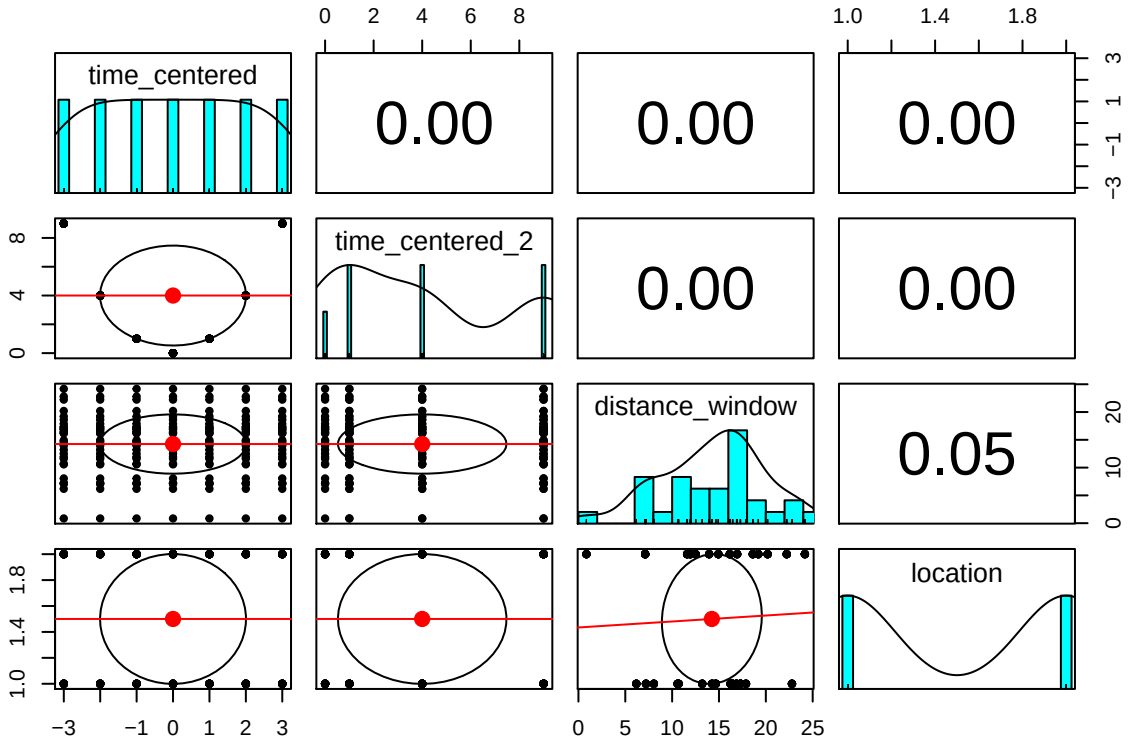
3.6 Multicollinearity

Finally, we should check for multicollinearity of the fixed effect predictors. Without a well established way to extract the vif from lmer models, we can look at the pariwise correlations of the predictors.

The correlations don't seem problematic.

Notice that we see no correlation between time and its quatratic term, because we have centered time.

```
pairs.panels(data_wound_long_centered_time[, c("time_centered",
    "time_centered_2", "distance_window", "location")], col = "red",
    lm = T)
```



4 Other resources

This document cannot be considered a complete guide for model diagnistics for mixed models.

Model diagnostics of mixed models is quite complex and methods and tools are still in development. The field has not cought up to these developments yet, and there is no real consensus on how to do model diagnostics on these models other than some basics mentioned above.

The sources I used to compile the above guide:

http://ademos.people.uic.edu/Chapter18.html

https://www.ssc.wisc.edu/sscc/pubs/MM/MM_DiagInfer.html

Loy, A., Hofmann, H., & Cook, D. (2017). Model Choice and Diagnostics for Linear Mixed-Effects Models Using Statistics on Street Corners. Journal of Computational and Graphical Statistics, 26(3), 478-492.

Some more redings that can help those who are interested in the cutting edge on this topic:

http://thestatsgeek.com/2014/08/17/robustness-of-linear-mixed-models/

https://stat.ethz.ch/pipermail/r-sig-mixed-models/2014q2/022160.html

Loy, A., Hofmann, H., & Cook, D. (2017). Model Choice and Diagnostics for Linear Mixed-Effects Models Using Statistics on Street Corners. Journal of Computational and Graphical Statistics, 26(3), 478-492.

Some additional functions that may be useful for diagnostics in the HLMdiag package: (this is mentioned here: http://ademos.people.uic.edu/Chapter18.html):

- case_delete() - iteratively delete groups corresponding to the levels of a hierarchical linear model, using lmer to fit the models for each deleted case

- covratio() - calculate measures of the change in the covariance matrices for the fixed effects based on the deletion of an observation, or group of observations,
- diagnostics() - is used to compute deletion diagnostics for a hierarchical linear model based on the building blocks returned by case_delete.
- HLMresid() - extracts residuals from a hierarchical linear model fit using lmer. Can provide a variety of different types of residuals based upon the specifications when you call the function
- leverage() - calculates the leverage of a hierarchical linear model fit
- mdffits() - calculate measures of the change in the fixed effects estimates based on the deletetion of an observation, or group of observations

For an alternative "consensus-based" approach on model diagnostics using plot lineups see Loy, Hofmann and Cook (2017).

Loy, A., Hofmann, H., & Cook, D. (2017). Model Choice and Diagnostics for Linear Mixed-Effects Models Using Statistics on Street Corners. Journal of Computational and Graphical Statistics, 26(3), 478-492.