Exercise 14 - Model diagnostics of mixed models

Zoltan Kekecs

19 november 2020

Contents

1 Abstract

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

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(tidyverse) # for tidy code and ggplot
library(influence.ME) # for influence (this will also load the lme4 package)
library(lattice) # for qqmath
library(lme4) # for mixed models
library(lmerTest)  # for significance test on lmer() mixed models
```

2.2 Load wound healing data

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

In this exercise we will work with simulated data about wound healing over time after a surgical procedure. We know that psychological factors, especailly 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).

There are the following variables in the dataset:

- ID: participant ID
- day_1, day_2, ..., day_7: Wound rating scores collected on different days of the study. The would was assessed by an expert first 1 day post surgery, and it was assessed once every day after that until 7 days after the operation. The higher the score the larger the wound size or the worse the wound condition (e.g. due to inflammation). The smaller the score the closer the wound is to being healed.
- distance_window: The distance of the patient's bed from the closest window in meters.
- location: The hospital wing in which the patient's bed is located can take value "north wing" or "sounth wing". This can be important if the amount of sunlight matters, since the south wing hospital rooms get more sunlight in this hospital.

```
data_wound = read_csv("https://raw.githubusercontent.com/kekecsz/PSYP14-Advanced-Scientific-Methods/mai
```

```
##
## -- Column specification ------------------------------------------------
## cols(
##   ID = col_character(),
```

```
##   day_1 = col_double(),
##   day_2 = col_double(),
##   day_3 = col_double(),
##   day_4 = col_double(),
##   day_5 = col_double(),
##   day_6 = col_double(),
##   day_7 = col_double(),
##   distance_window = col_double(),
##   location = col_character()
## )
```

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

We create a new data object where the data is restructured to a 'long format', and we also recode the variable 'days' to a numerical variable.

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

We center the variable 'days' to avoid problems with multicollinearity, and then create a new variable which contains the squared values of the 'days'.

```r
data_wound_long = data_wound_long %>%
  mutate(days_centered = days - mean(days)) %>%
  mutate(days_centered_sq = days_centered^2)
```

2.3 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.

```r
mod_rep_int_quad = lmer(wound_rating ~ days_centered + I(days_centered^2) + distance_window + location

data_wound_long = data_wound_long %>%
  mutate(resid = residuals(mod_rep_int_quad))

random_effects = as.data.frame(ranef(mod_rep_int_quad)[[1]])
names(random_effects) = c("intercept")
```

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 satisfied in the case of linear mixed models as well.

In addition, we also assume that

- we have modelled the **dependency structure of random effects** correctly,
- the **within-unit residual errors follow normal distributions**, and **have constant variance**.
- **random effects are normally distributed centered around 0** (the intercepts and slopes if any).

How to effectively and consistently check these assumptions is still being worked out by researchers and statisticians. But there are already some trends emerging about how we can do model diagnostics for mixed linear models. 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'.

```r
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 or violin plot. If there is a problem, you should see outliers that are considerably off the average coefficient value.

In the code below we create a long format dataset from the influence_group dataset and plot the distribution of the leave-one-out coefficients for each predictor in the model (including the intercept)

```r
data_plot_inflience = as_tibble(influence_group) %>%
  gather(colnames(influence_group), value = coefficient, key = predictor)

data_plot_inflience %>%
  ggplot() +
  aes(x = 1, y = coefficient, group = predictor) +
  geom_violin() +
  geom_jitter(width = 0.2) +
  facet_wrap( ~ predictor, scales = "free")
```
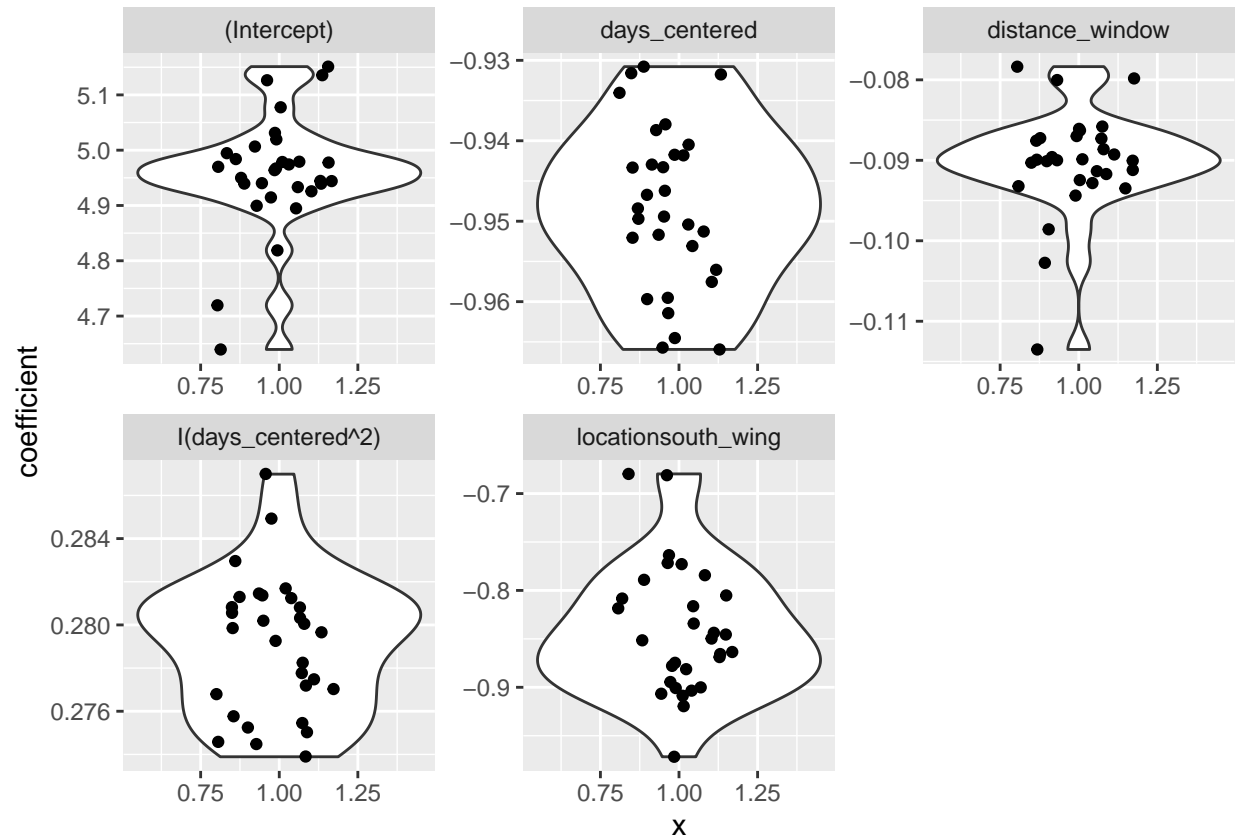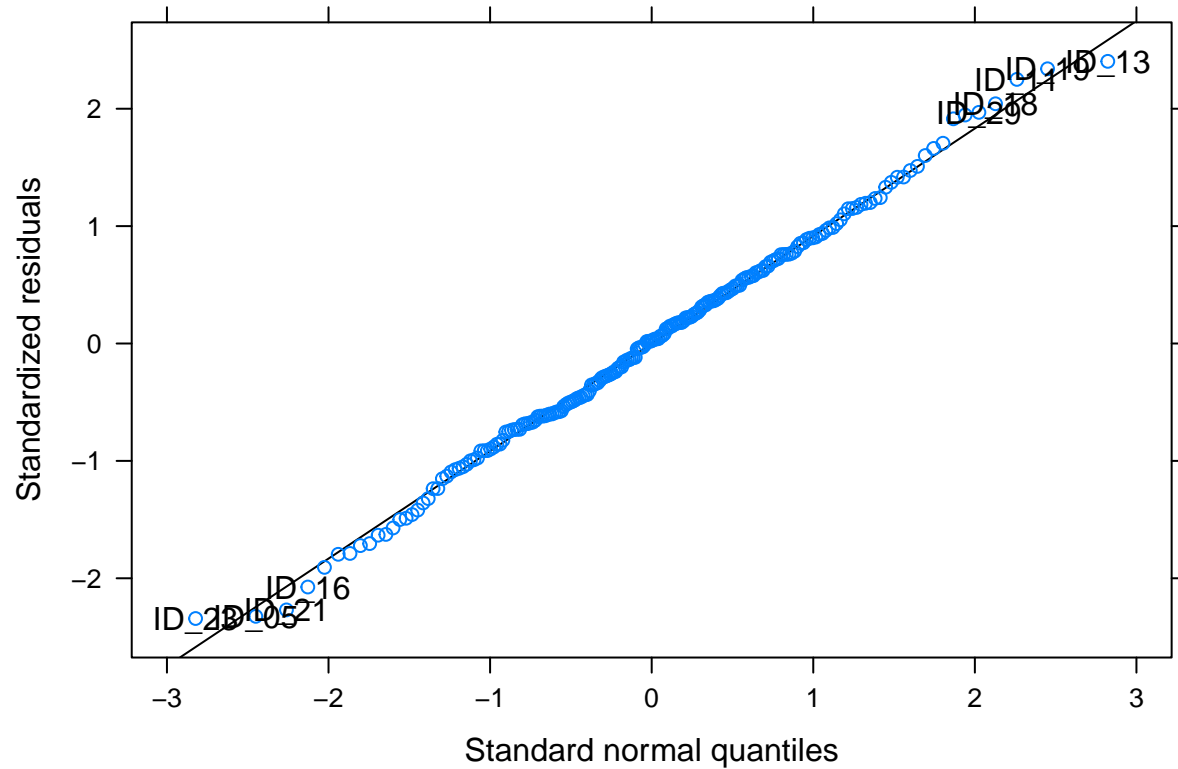
These plots do not indicate extreme influential cases.

3.3 Normality

3.3.1 normality of all residuals

You can check if the residuals on the observation level are normally distributed using the qqmath() or the qqnorm() function.
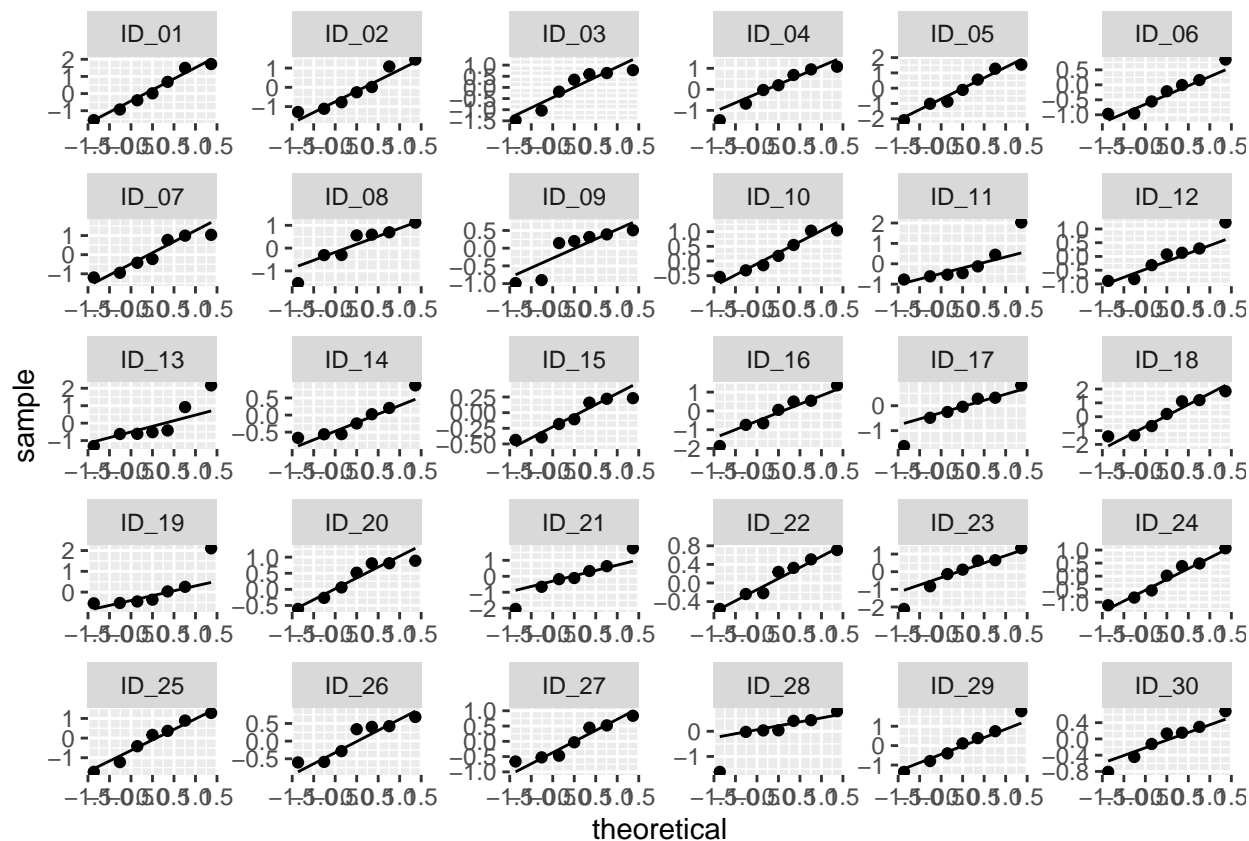
```
qqmath(mod_rep_int_quad, id=0.05)
```

3.3.2 normality of residuals within clusters

You can check the qq plot of the residuals by clusters using facet_wrap() in ggplot.

```
data_wound_long %>%
  ggplot() +
    aes(sample = resid) +
    stat_qq() +
    stat_qq_line() +
    facet_wrap( ~ ID, scales = "free")
```

3.3.3 normality of random effects

You can use the same qqmath 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. Furthermore, you can plot the classical qqplot using the stat_qq() and stat_qq_line() functions in ggplot, and can get the skew and kurtosis statistics for the random effects using describe().

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

```
## $ID
```

ID
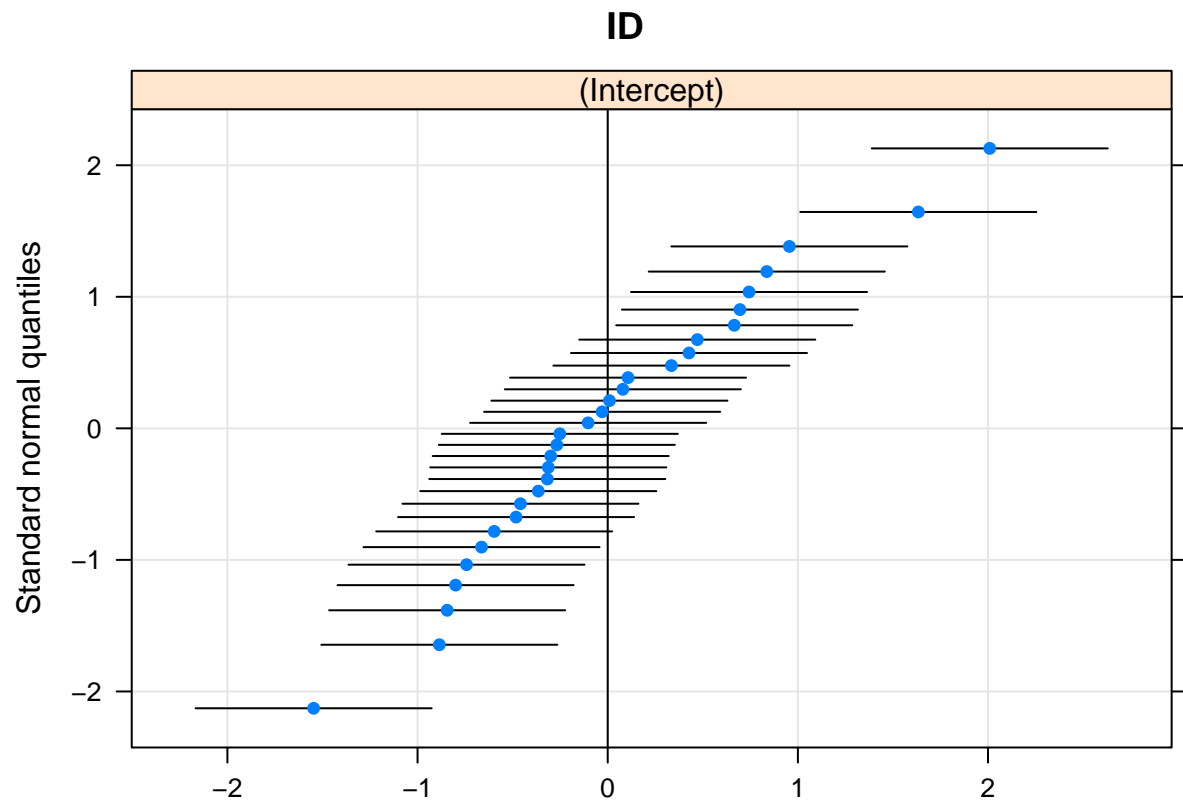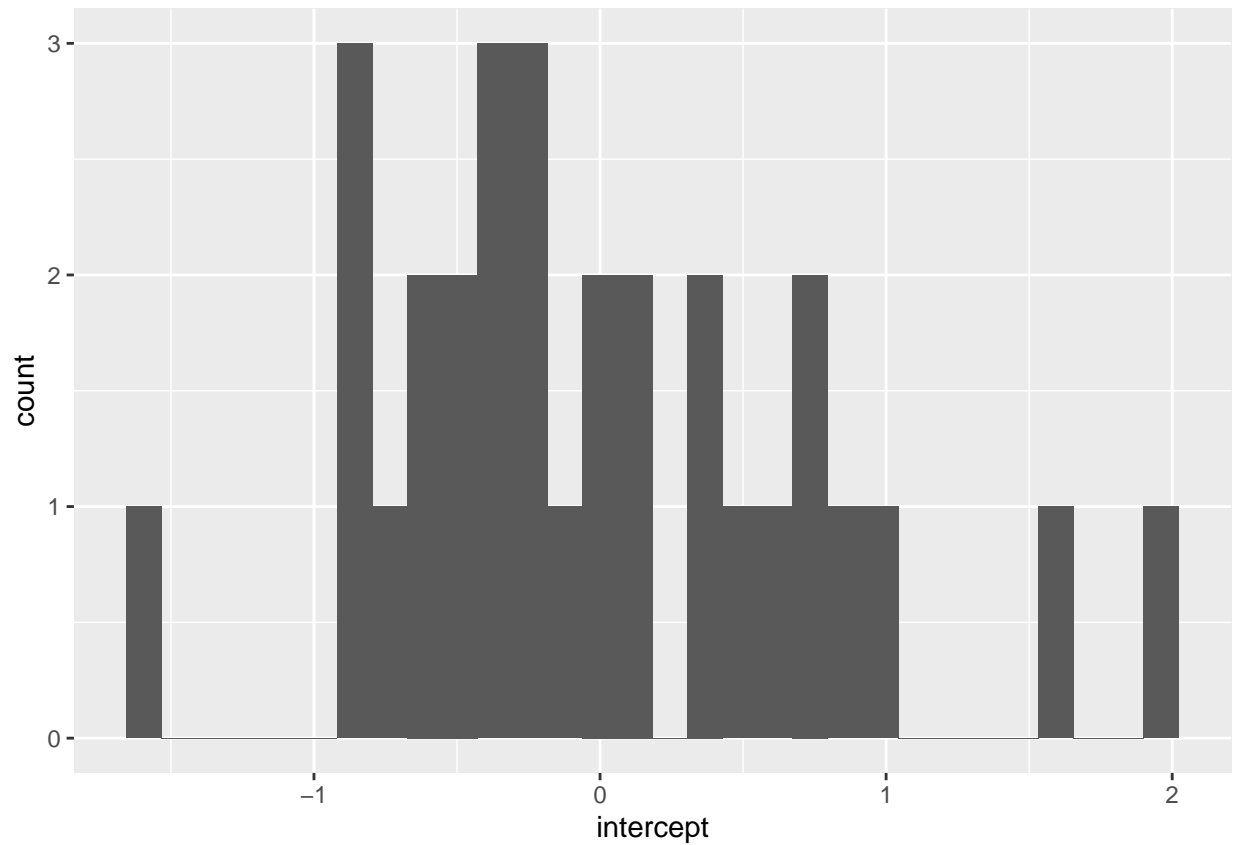


```
random_effects %>%
  ggplot() +
    aes(sample = intercept) +
    stat_qq() +
    stat_qq_line()
```
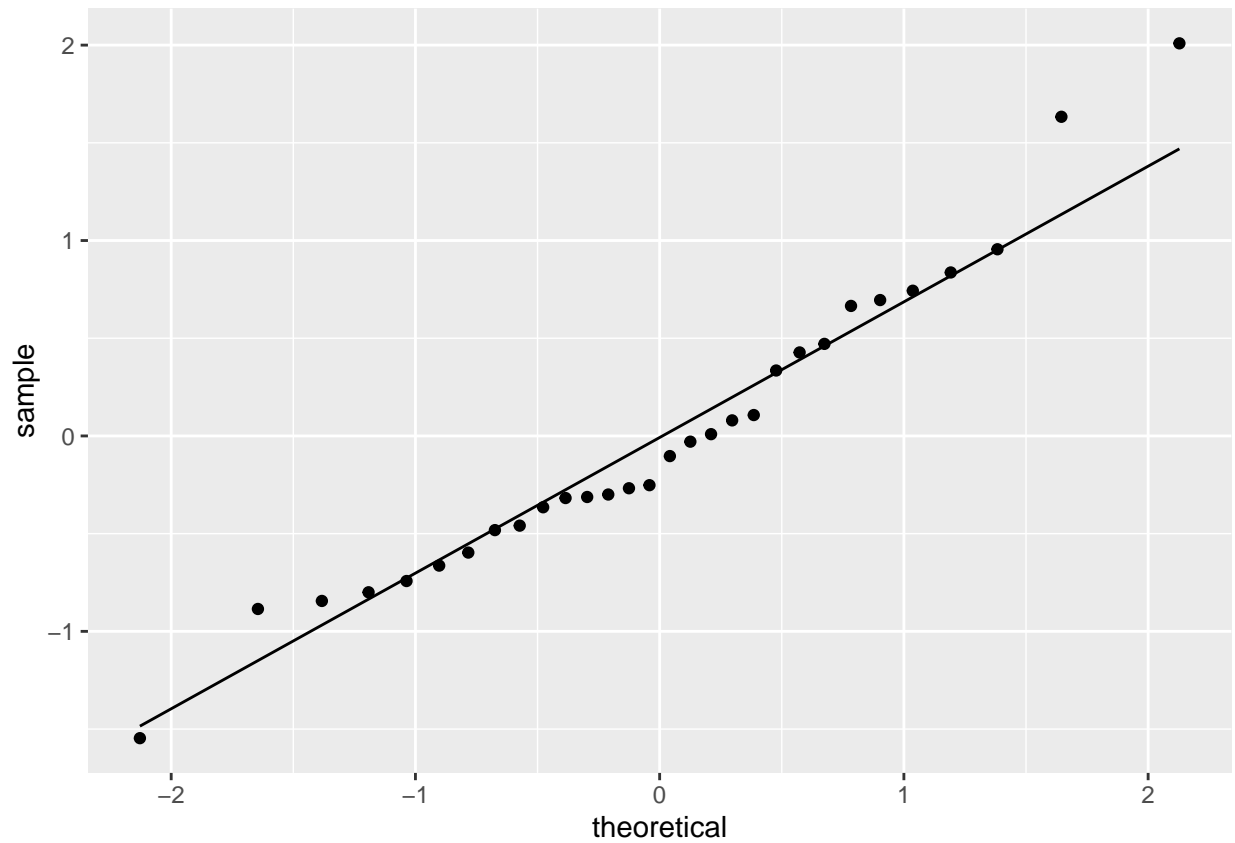
```
random_effects %>%
  ggplot() +
    aes(x = intercept) +
     geom_histogram()
```

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```
describe(random_effects$intercept)$skew
```
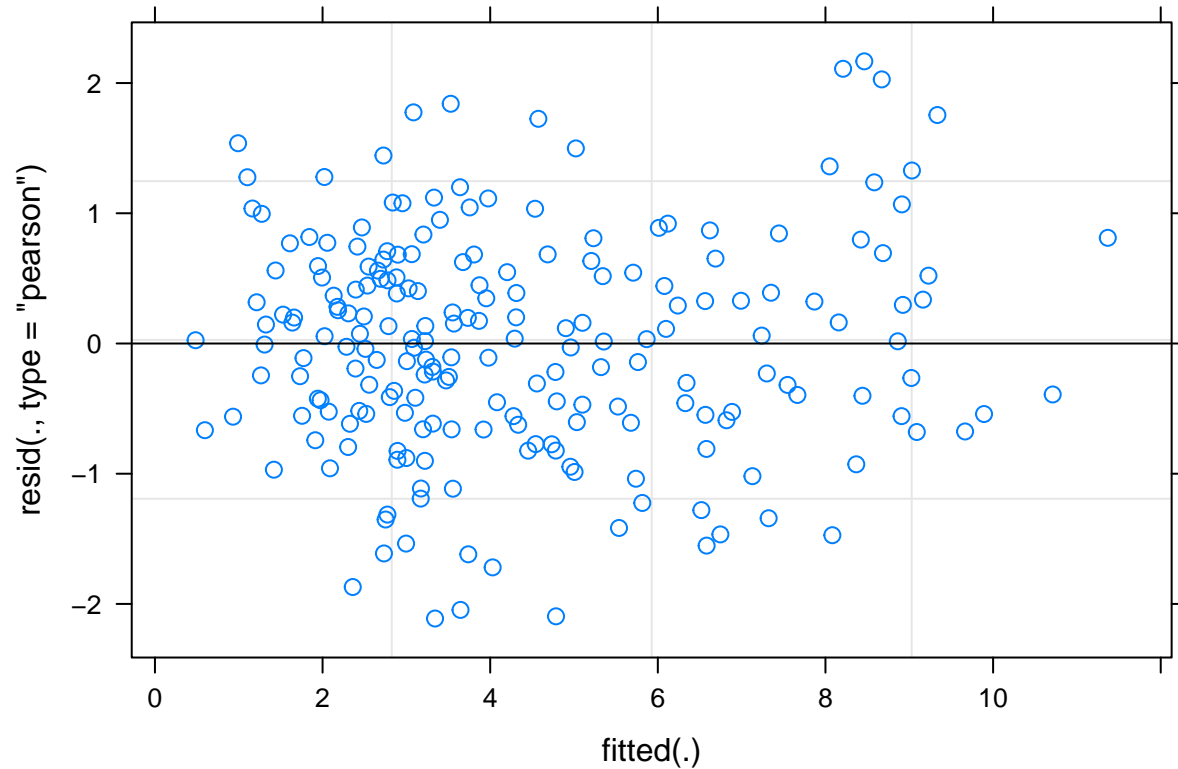
```
## [1] 0.6107886
```

```
describe(random_effects$intercept)$kurtosis
```

```
## [1] 0.176785
```

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 days_centered still has a nonlinear relationship with the residuals. Maybe we could decrease this by adding the cubic term of days to the model as well adding + I(days_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.
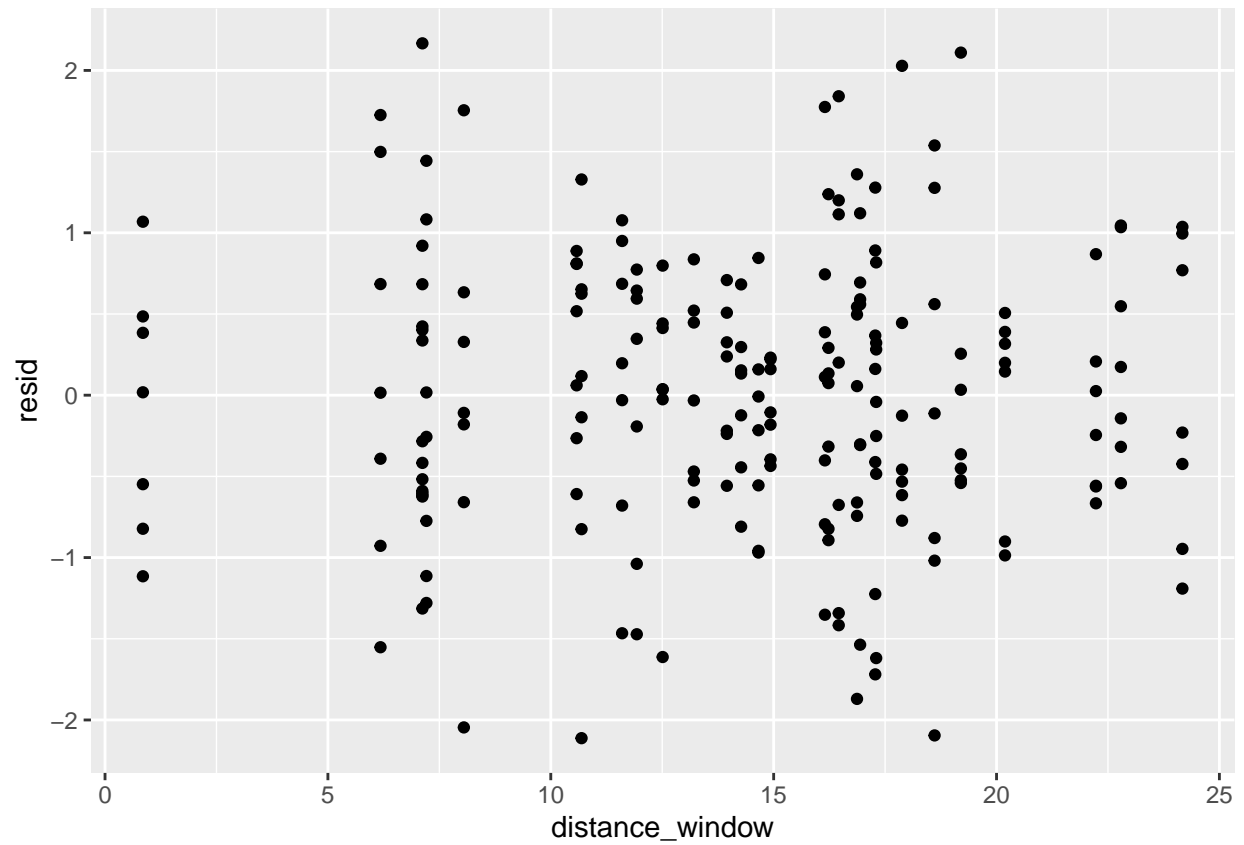
```
data_wound_long %>%
  ggplot() +
  aes(x = days_centered, y = resid) +
  geom_point()
```
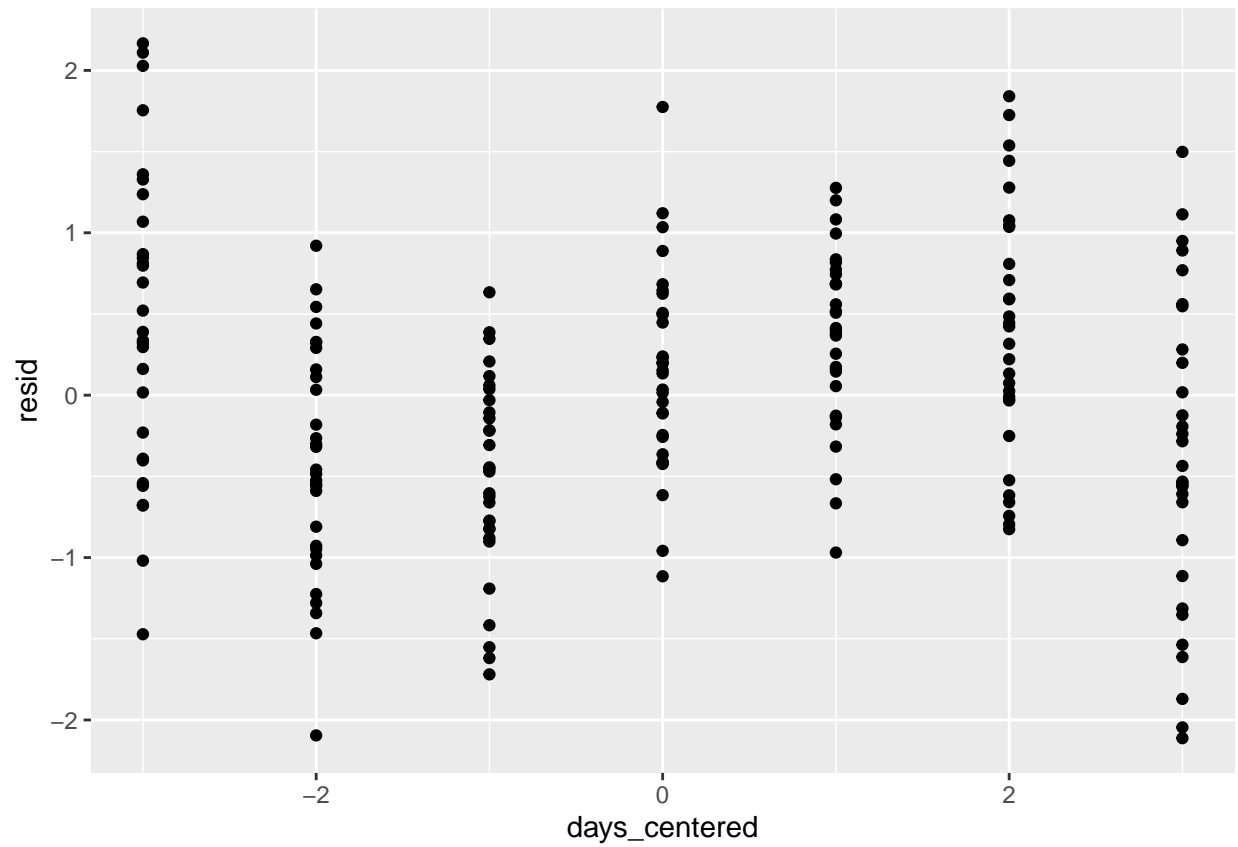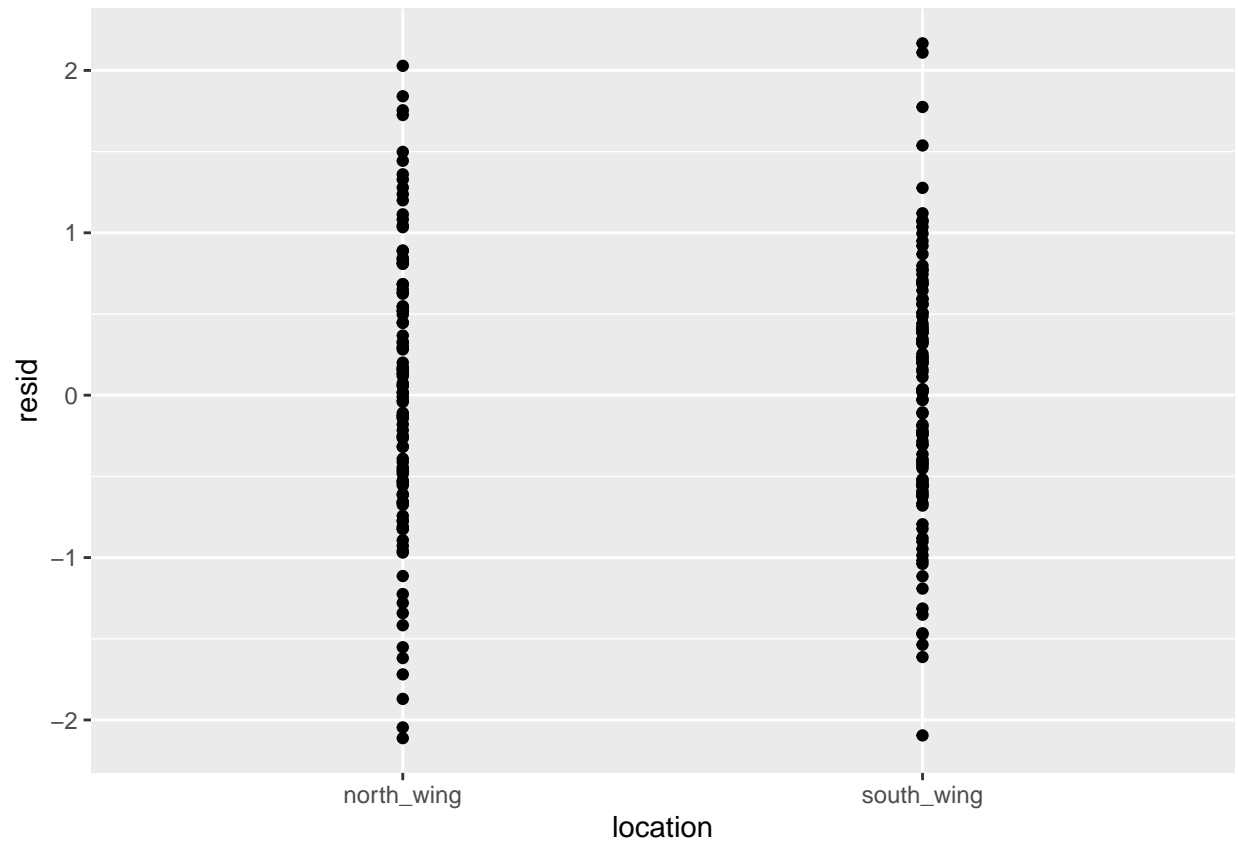
```
data_wound_long %>%
  ggplot() +
  aes(x = days_centered_sq, y = resid) +
  geom_point()
```
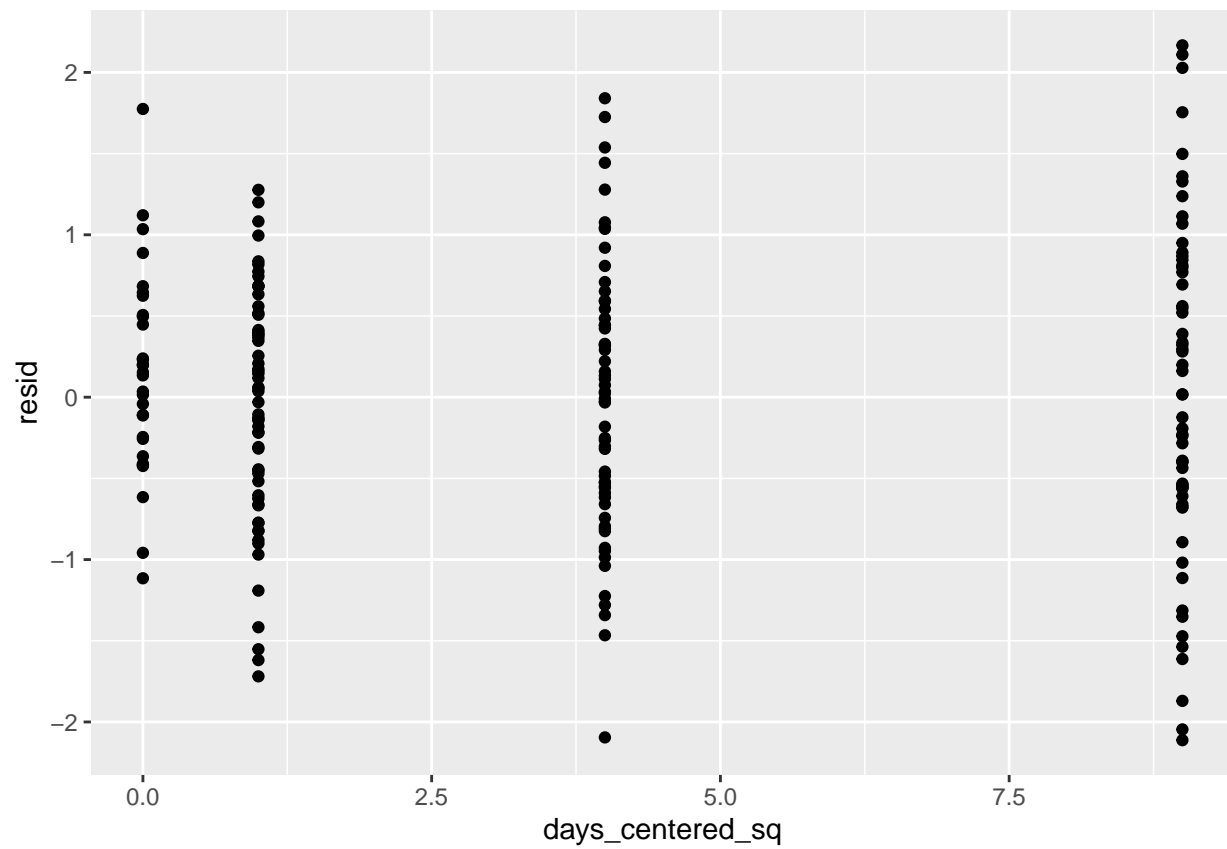
```
data_wound_long %>%
  ggplot() +
  aes(x = distance_window, y = resid) +
  geom_point()
```

```
data_wound_long %>%
  ggplot() +
  aes(x = location, y = resid) +
  geom_point()
```

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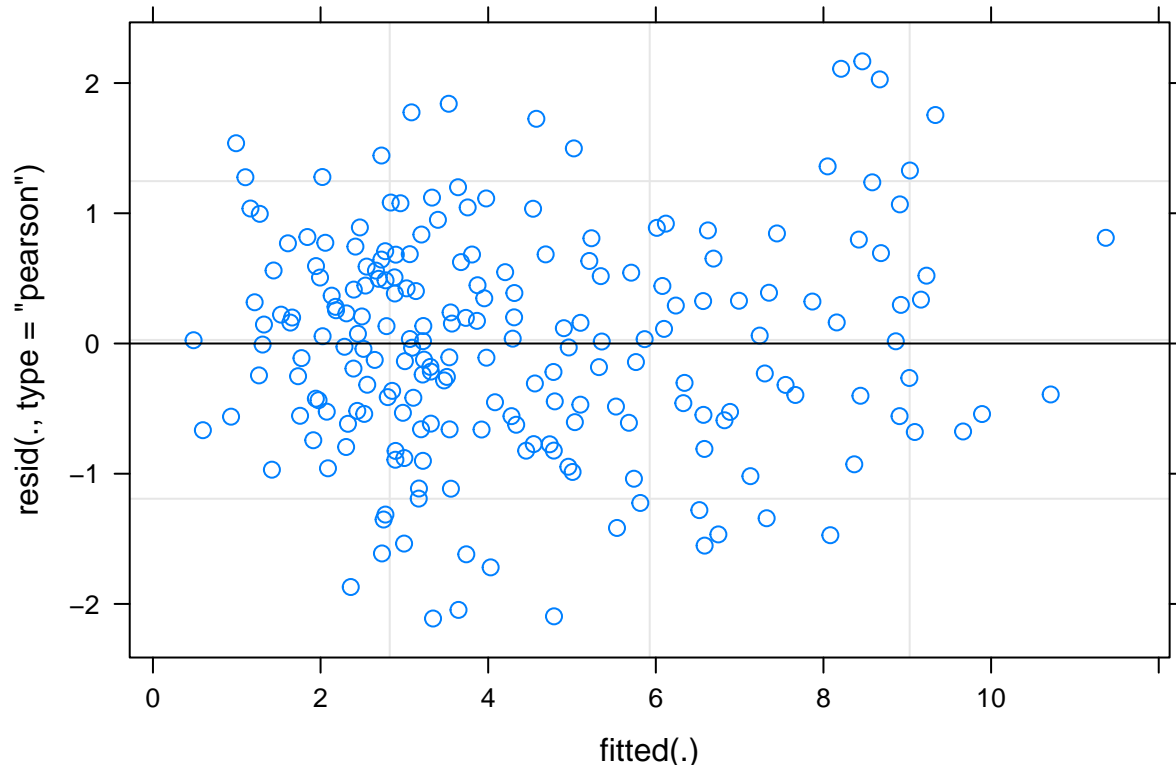
When working with mixed linear models we need to check for homoscedasticity across clusters as well.

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(resid^2 ~ ID, data = data_wound_long)
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, f = data_wound_long$ID), function(x) IQ
# rank ordering them
rank = rank(IQR_of_residuals_by_participant)
# adding rank to the dataframe containing the residuals
data_wound_long$rank = rep(rank, each = length(c("day_1", "day_2", "day_3", "day_4", "day_5",   "day_6"
# creating a vector of participant IDs ordered based on the rank, this will be used as labels
IDforplot = unique(data_wound_long$ID[order(data_wound_long$rank)])

# create the plot
ggplot(data_wound_long, aes(y = resid, x = factor(rank), labels = ID))+
  geom_boxplot()+
  scale_x_discrete(labels=IDforplot)+
```

```
coord_flip()
```
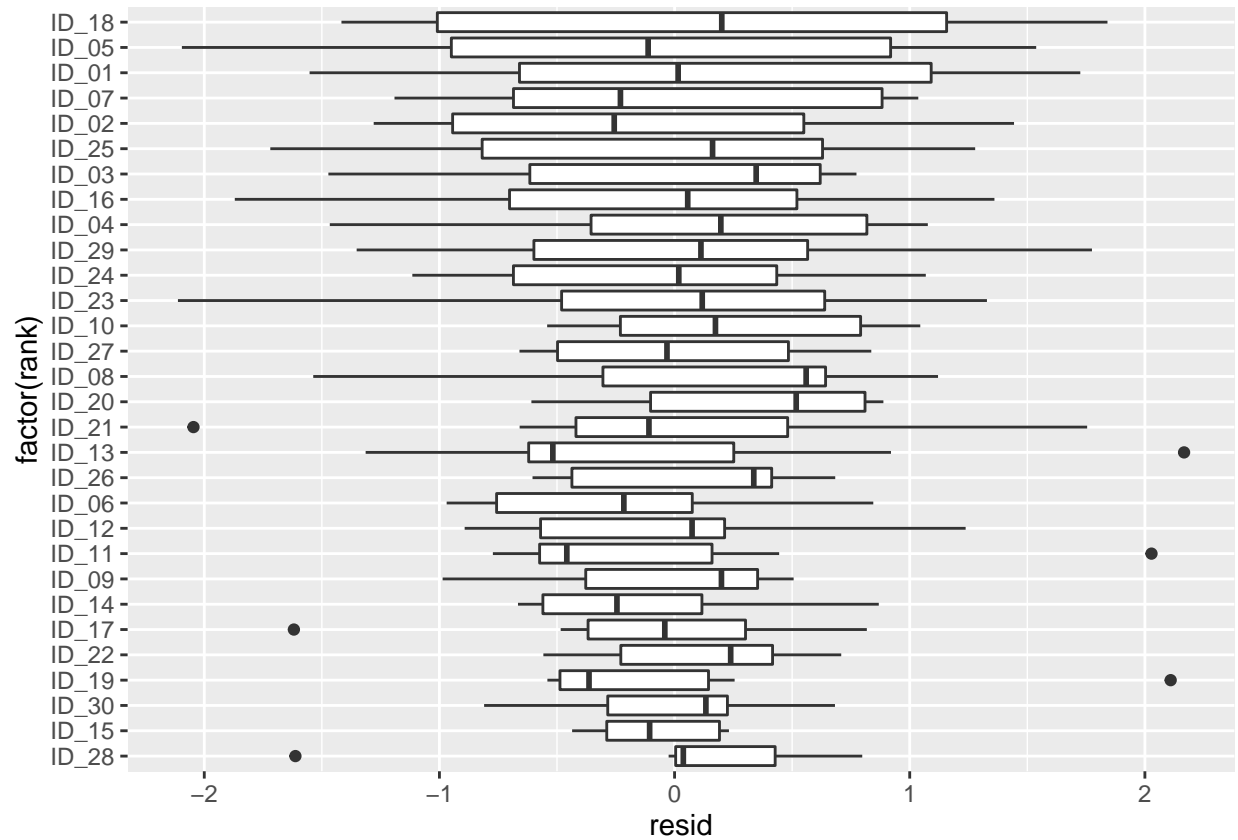

3.6 Multicollinearity

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 days and its quatratic term, because we have centered days.

```
pairs.panels(data_wound_long[,c("days_centered", "days_centered_sq", "distance_window", "location")], c
```
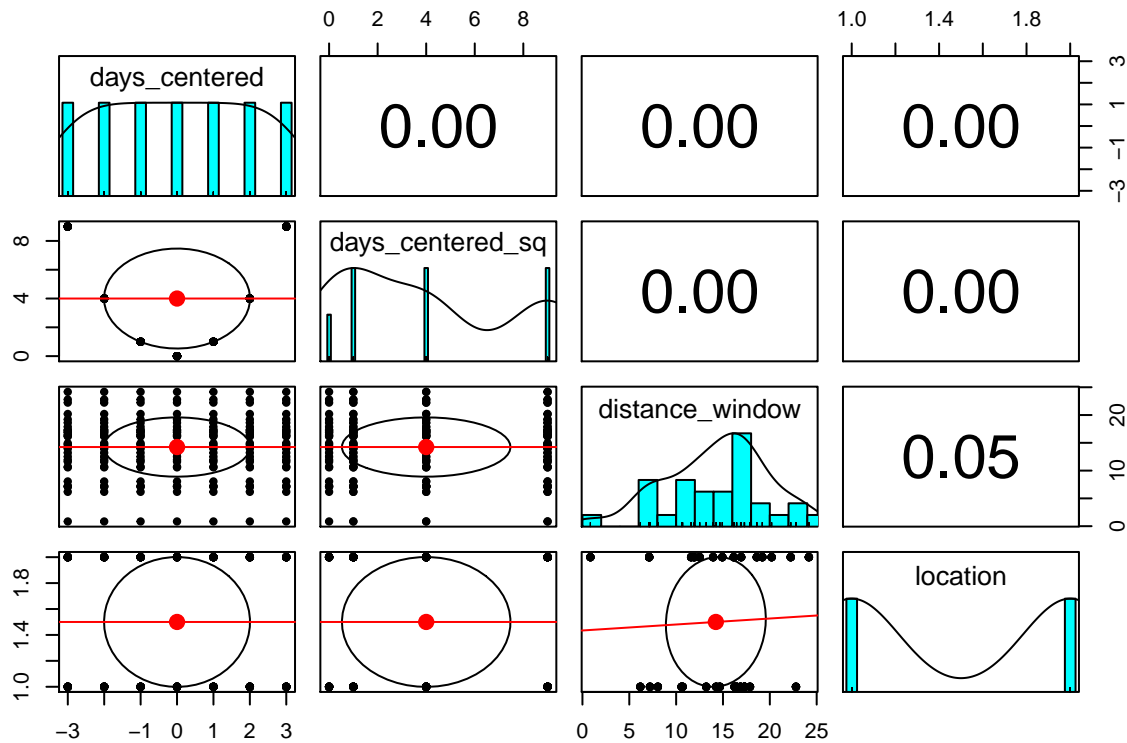
4 Other resources

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.