One-factor CRD with subsampling

the data

An experiment was conducted as a completely randomized design with sub-sampling: there were 4 treatments, and 4 plots for each treatment. Within each plot 3 measurements (subsamples) were taken. In the data file, the column “core” indicates the subsample number. The column “y” contains the response.

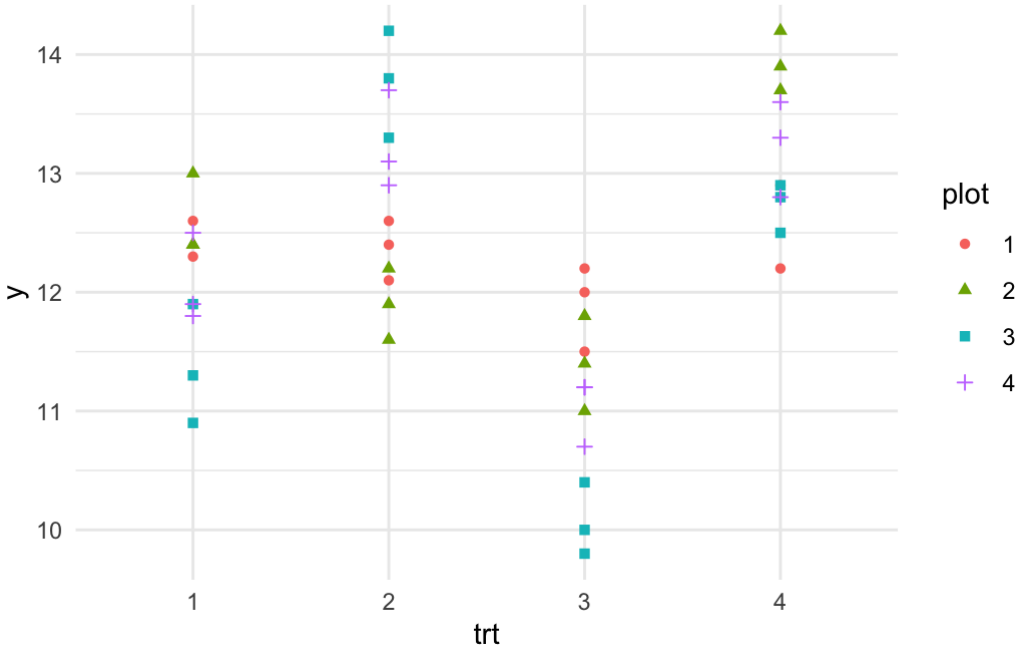
```
dat = read.csv("crdsub.csv", colClasses=c("factor","factor","factor","numeric"))
head(dat, n=4)
```

```
  trt plot core    y
1   1    1    1 12.6
2   1    1    2 11.9
3   1    1    3 12.3
4   1    2    1 13.0
```

```
with(dat, table(trt,plot))
```

```
   plot
trt 1 2 3 4
  1 3 3 3 3
  2 3 3 3 3
  3 3 3 3 3
  4 3 3 3 3
```

```
ggplot(dat, aes(y=y, x=trt, shape=plot, color=plot)) + geom_point() +theme_minimal()
```



manual analysis with fixed-effects

For the analysis, we can start with a fixed-effect model to get the ANOVA table. The random effect for "plot" should be be indicated with `plot`, because the plot values go from 1 to 4 only, and plot 1 means totally different plots across treatments (say). So we use `trt:plot` to get a different plot value for each plot (e.g. `1:1` for plot 1 in treatment 1, and `2:1` for plot 1 in treatment 2).

```
fit1 = lm(y ~ trt + trt:plot, dat)
anova(fit1) # warning: wrong F test for trt, because fixed-effect model
```

```
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value    Pr(>F)
trt        3 29.407  9.8024  76.507 1.097e-14
trt:plot  12 17.386  1.4488  11.308 2.342e-08
Residuals 32  4.100  0.1281
```

Warning about the output above: the F-test is wrong for the fixed effects (treatment), because the plot effects are considered fixed here instead of random. The test for plot variation is correct though.

Since this is a balanced design, so we can use the SS and MS from the `anova` function above (which returns type 1 SSs, but type 1 = type 3 SS with balanced designs). So here is a correct test for treatment differences:

```
f = 9.8024 / 1.4488; f              # f = 6.765875
pf(f, df1=3, df2=12, lower.tail=F) # p-value = 0.006365391
(1.4488 - 0.1281)/3                # sigma2_plot = 0.4402333
```

Warning again: the fixed effects model has wrong SEs for treatment means (e.g. intercept = mean of treatment 1 in plot 1 here) and wrong SEs for treatment differences.

```
head(summary(fit1)$coefficients, n=14)
```

```
              Estimate Std. Error    t value     Pr(>|t|)
(Intercept) 12.2666667  0.2066599 59.3567689 2.608221e-34
trt2         0.1000000  0.2922613  0.3421596 7.344680e-01
trt3        -0.3666667  0.2922613 -1.2545851 2.187158e-01
trt4         0.3666667  0.2922613  1.2545851 2.187158e-01
trt1:plot2   0.3333333  0.2922613  1.1405319 2.625293e-01
trt2:plot2  -0.4666667  0.2922613 -1.5967447 1.201523e-01
trt3:plot2  -0.5000000  0.2922613 -1.7107978 9.679780e-02
trt4:plot2   1.3000000  0.2922613  4.4480744 9.800566e-05
trt1:plot3  -0.9000000  0.2922613 -3.0794361 4.236568e-03
trt2:plot3   1.4000000  0.2922613  4.7902340 3.652662e-05
trt3:plot3  -1.8333333  0.2922613 -6.2729254 4.937112e-07
trt4:plot3   0.1000000  0.2922613  0.3421596 7.344680e-01
trt1:plot4  -0.2000000  0.2922613 -0.6843191 4.987001e-01
trt2:plot4   0.8666667  0.2922613  2.9653829 5.673462e-03
```

correct analysis: random-effects model

Use the `lme4` package for random effects. `nlme` is an alternative, but can only handle nested random effects, not crossed random effects.

```
library(lme4)
fit2 = lmer(y ~ trt + (1 | trt:plot), dat)
summary(fit2)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: y ~ trt + (1 | trt:plot)
   Data: dat

REML criterion at convergence: 73.5

Scaled residuals:
     Min       1Q   Median       3Q      Max
-1.47874 -0.77009  0.08077  0.61410  1.44532

Random effects:
 Groups   Name        Variance Std.Dev.
 trt:plot (Intercept) 0.4402   0.6635
 Residual             0.1281   0.3579
Number of obs: 48, groups:  trt:plot, 16

Fixed effects:
            Estimate Std. Error t value
(Intercept)  12.0750     0.3475  34.751
trt2          0.7417     0.4914   1.509
trt3         -0.9750     0.4914  -1.984
trt4          1.0583     0.4914   2.154

Correlation of Fixed Effects:
     (Intr) trt2   trt3
trt2 -0.707
trt3 -0.707  0.500
trt4 -0.707  0.500  0.500
```

note the same estimate for the 2 variance components as estimated above:

- 0.4402 for plot variation
- 0.1281 for subsample variation (between “cores”)

Now to test treatment effects:

```
anova(fit2) # tests fixed effects
```

```
Analysis of Variance Table
    npar Sum Sq Mean Sq F value
trt    3 2.6006 0.86687  6.7658
```

```
# then manual calculation of p-value based on 16-4 = 12 df denominator:
pf(6.7658, df1=3, df2=12, lower.tail=F) # p-value = 0.006365
```

```
[1] 0.006365647
```

and to test for variation between plots:

```
fit2.null = lm(y ~ trt, dat)
anova(fit2, fit2.null) # put complex model first!
```

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

```
Data: dat
Models:
fit2.null: y ~ trt
fit2: y ~ trt + (1 | trt:plot)
          npar     AIC     BIC  logLik deviance Chisq Df Pr(>Chisq)
fit2.null    5 107.635 116.991 -48.818   97.635
fit2         6  83.795  95.022 -35.898   71.795 25.84  1  3.709e-07
```

Warning: the test above is a LRT test. Downsides:

- it's only approximate, and
- it's conservative because $\sigma^2=0$ is at the boundary of the parameter space.

Alternative, use tools from the `lmerTest` package to get p-values:

```
library(lmerTest)
fit3 = lmer(y ~ trt + (1 | trt:plot), dat)
anova(fit3)
```

```
Type III Analysis of Variance Table with Satterthwaite's method
    Sum Sq Mean Sq NumDF DenDF F value   Pr(>F)
trt 2.6006 0.86687     3    12  6.7658 0.006366
```

```
drop1(fit3)
```

```
Single term deletions using Satterthwaite's method:

Model:
y ~ trt + (1 | trt:plot)
    Sum Sq Mean Sq NumDF DenDF F value   Pr(>F)
trt 2.6006 0.86687     3    12  6.7658 0.006366
```

```
ranova(fit3) # to test for random effects, but LRT. F-test better when appropriate
```

```
ANOVA-like table for random-effects: Single term deletions

Model:
y ~ trt + (1 | trt:plot)
               npar  logLik     AIC    LRT Df Pr(>Chisq)
<none>            6 -36.752  85.503
(1 | trt:plot)    5 -51.634 113.267 29.764  1   4.88e-08
```

pool subsamples?

Here we get something equivalent to the correct mixed-model analysis because the design is balanced.

Note that the result of the tests (f statistic & p-value) are the same as before, but the sums of squares for `trt` and residual SS are exactly 1/3 as large as before, because there are only 1/3 as many data points used in the analysis.

```
dat_byplot = dat %>% group_by(trt:plot) %>% summarize(trt=trt[1], plot=plot[1], y=mean
(y))
head(as.data.frame(dat_byplot), n=5)
```

```
  trt:plot trt plot        y
1      1:1   1    1 12.26667
2      1:2   1    2 12.60000
3      1:3   1    3 11.36667
4      1:4   1    4 12.06667
5      2:1   2    1 12.36667
```

```
fit4 = lm(y ~ trt, dat_byplot)
anova(fit4)
```

```
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value   Pr(>F)
trt        3 9.8024  3.2675  6.7658 0.006366
Residuals 12 5.7953  0.4829
```

```
summary(fit4)$coefficients
```

```
             Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 12.0750000  0.3474694 34.751266 2.055069e-13
trt2         0.7416667  0.4913959  1.509306 1.570928e-01
trt3        -0.9750000  0.4913959 -1.984144 7.058967e-02
trt4         1.0583333  0.4913959  2.153729 5.229395e-02
```

What we lack from this analysis is the ability to measure how subsampling helps. With this analysis, we can't use this experiment to guide future experiments in terms of how many subsamples are optimal.

treatment differences and contrasts

The output of the `lmer` fit shows the SEs to compare pairs of treatments (0.4914) and the SEs to get the confidence interval for a single treatment mean (0.3475 –here for treatment 1, but it's the same for all treatment means because the design is balanced).

```
# fit2@beta   # same values as below (fixed effects), no names
# fixef(fit2) # fixed effects
summary(fit2)$coefficients
```

```
             Estimate Std. Error   t value
(Intercept) 12.0750000  0.3474694 34.751264
trt2         0.7416667  0.4913959  1.509306
trt3        -0.9750000  0.4913959 -1.984144
trt4         1.0583333  0.4913959  2.153728
```

`lmerTest` also makes it very easy to make pairwise comparisons (but warning: *no* multiple comparison protection here)

```
ls_means(fit3)
```

```
Least Squares Means table:

     Estimate Std. Error df t value    lower    upper  Pr(>|t|)
trt1 12.07500    0.34747 12  34.751 11.31793 12.83207 2.055e-13
trt2 12.81667    0.34747 12  36.886 12.05960 13.57374 1.011e-13
trt3 11.10000    0.34747 12  31.945 10.34293 11.85707 5.588e-13
trt4 13.13333    0.34747 12  37.797 12.37626 13.89040 7.562e-14

  Confidence level: 95%
  Degrees of freedom method: Satterthwaite
```

```
ls_means(fit3, which="trt", pairwise=TRUE)
```

```
Least Squares Means table:

             Estimate Std. Error df t value     lower     upper Pr(>|t|)
trt1 - trt2 -0.741667   0.491396 12 -1.5093 -1.812326  0.328993 0.157093
trt1 - trt3  0.975000   0.491396 12  1.9841 -0.095660  2.045660 0.070590
trt1 - trt4 -1.058333   0.491396 12 -2.1537 -2.128993  0.012326 0.052294
trt2 - trt3  1.716667   0.491396 12  3.4934  0.646007  2.787326 0.004435
trt2 - trt4 -0.316667   0.491396 12 -0.6444 -1.387326  0.753993 0.531427
trt3 - trt4 -2.033333   0.491396 12 -4.1379 -3.103993 -0.962674 0.001376

  Confidence level: 95%
  Degrees of freedom method: Satterthwaite
```

or we can use the `emmeans` package

```
library(emmeans)
fit3_em = emmeans(fit3, "trt")
fit3_em
```

```
 trt emmean    SE df lower.CL upper.CL
 1     12.1 0.347 12     11.3     12.8
 2     12.8 0.347 12     12.1     13.6
 3     11.1 0.347 12     10.3     11.9
 4     13.1 0.347 12     12.4     13.9

Degrees-of-freedom method: kenward-roger
Confidence level used: 0.95
```

```
pairs(fit3_em) # Tukey correction by default, unlike above
```

```
 contrast estimate    SE df t.ratio p.value
 1 - 2      -0.742 0.491 12  -1.509  0.4621
 1 - 3       0.975 0.491 12   1.984  0.2468
 1 - 4      -1.058 0.491 12  -2.154  0.1916
 2 - 3       1.717 0.491 12   3.493  0.0200
 2 - 4      -0.317 0.491 12  -0.644  0.9155
 3 - 4      -2.033 0.491 12  -4.138  0.0065

Degrees-of-freedom method: kenward-roger
P value adjustment: tukey method for comparing a family of 4 estimates
```

```
pwpm(fit3_em, adjust="none", means=F) # no correction: LSD. bottom: t-values
```

```
       1      2      3      4
1         0.1571 0.0706 0.0523
2 -0.742         0.0044 0.5314
3  0.975  1.717         0.0014
4 -1.058 -0.317 -2.033

Row and column labels: trt
Upper triangle: P values
Lower triangle: Comparisons (estimate)   earlier vs. later
```

Next, let's test the contrast `mu1 + mu2 + m3 - 3*mu4`, say. The `contrast` function from `emmeans` is the easiest to use, in terms of setting up our coefficients:

```
res = contrast(fit3_em, list(trt123_vs_4 = c(1,1,1,-3)))
# summary(res)$estimate; summary(res)$t.ratio # to get more precision
res
```

```
 contrast    estimate  SE df t.ratio p.value
 trt123_vs_4    -3.41 1.2 12  -2.832  0.0151

Degrees-of-freedom method: kenward-roger
```

or we can use the `contest` function from `lmerTest`, which requires being careful when setting up our coefficients:

```
fixef(fit3) # to get the meaning and order of coefficients
```

```
(Intercept)        trt2        trt3        trt4
 12.0750000   0.7416667  -0.9750000   1.0583333
```

```
# sum(c(0,1,1,-3) * fixef(fit3)) # contrast value
contest(fit3, c(0,1,1,-3)) # "con"trast "test", from lmerTest library
```

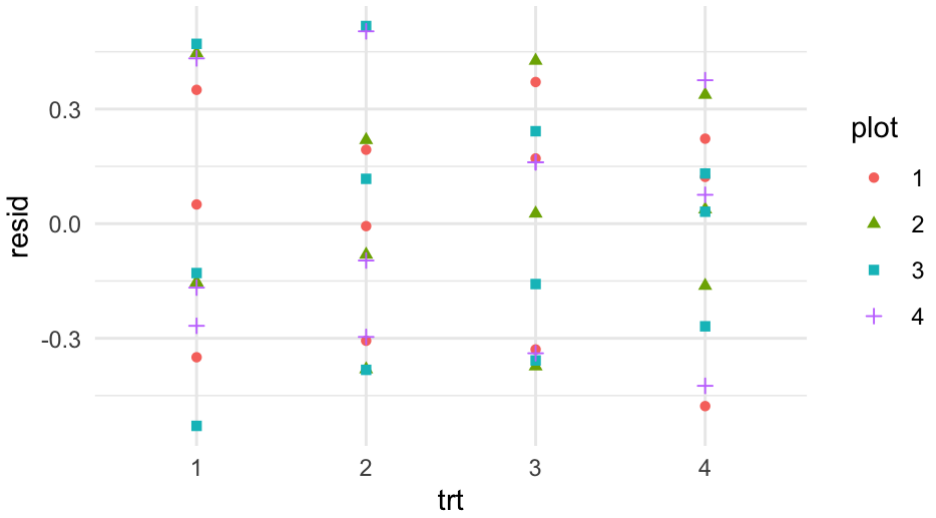
```
    Sum Sq  Mean Sq NumDF DenDF  F value    Pr(>F)
1 1.027315 1.027315     1    12 8.018069 0.0151302
```

Note the F-value above is `8.020224 = (-2.832)^2 = t^2`.

check assumptions

A plot of the total residuals that combine plot and subsample variation is *not* very helpful:

```
ggplot(data.frame(trt=dat$trt, resid=resid(fit2), plot=dat$plot),
  aes(x=trt, y=resid, color=plot, shape=plot)) +geom_point() +theme_minimal()
```



Let's first check the assumptions on the residuals at the subsample level, i.e. the term that is not accounted for in the model (the last residual level of variation).

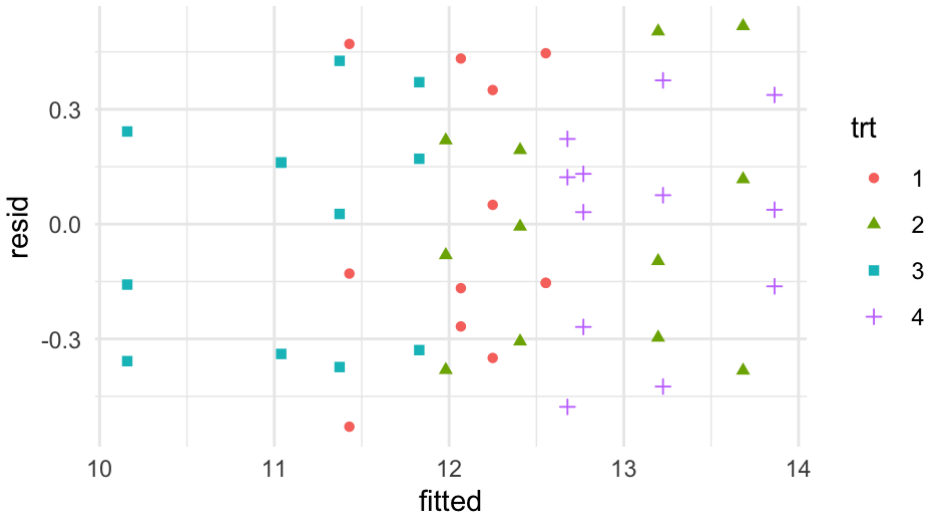
For this, it will be useful to get the model predictions that do or do not use the estimated random effects:

```
head(cbind(
  predict(fit2), # uses estimated random effects (re) by default
  predict(fit2, re.form=NA) )) # prediction for new plots: unknown random effects
```

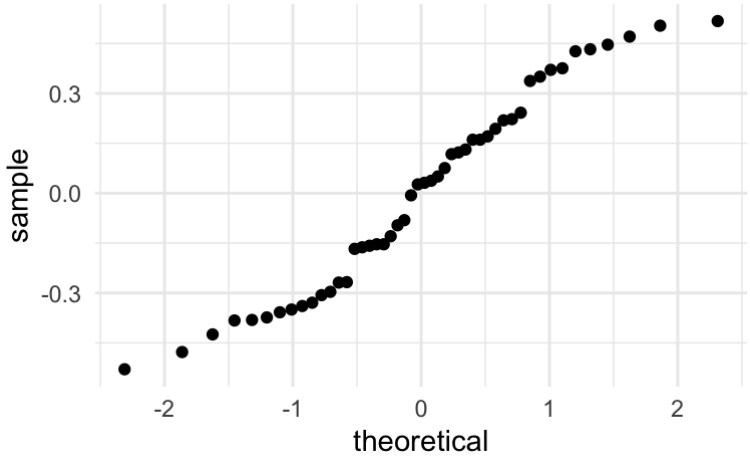
```
      [,1]   [,2]
1 12.24972 12.075
2 12.24972 12.075
3 12.24972 12.075
4 12.55357 12.075
5 12.55357 12.075
6 12.55357 12.075
```

To get the residuals at the subsample level, we calculate the difference between the observations and the predictions that know about the estimated plot (random) effects.

```
subsample_residuals = dat$y - predict(fit2)
tmp = data.frame(fitted=predict(fit2), resid=subsample_residuals, trt=dat$trt)
ggplot(tmp, aes(x=fitted, y=resid, color=trt,shape=trt)) +geom_point() +theme_minimal()
```




```
ggplot(tmp, aes(sample=resid)) + geom_qq() + theme_minimal()
```

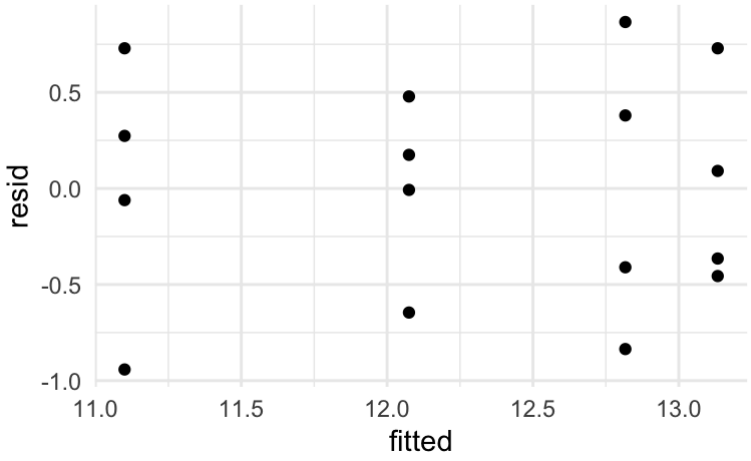


To look at residuals at the level of plots: we can extract the best linear unbiased estimates (BLUPs) of the random effects. In the residual plots below, we should have 16 points only: 1 per plot.

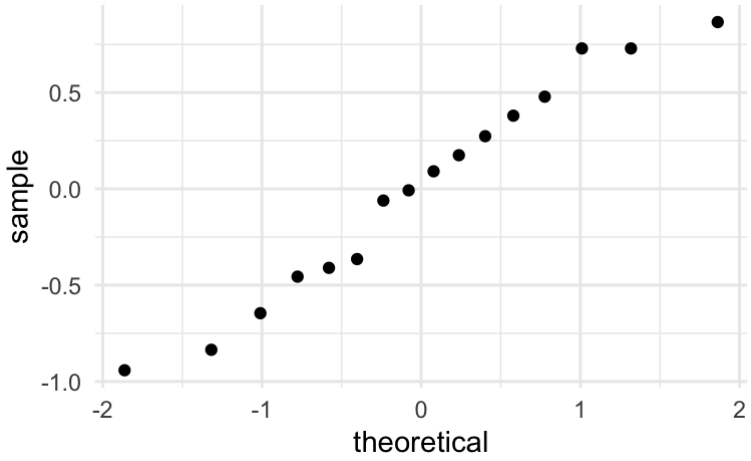
```
re = ranef(fit2); head(re$trt, n=4)
```

```
     (Intercept)
1:1  0.174716806
1:2  0.478572120
1:3 -0.645692543
1:4 -0.007596383
```

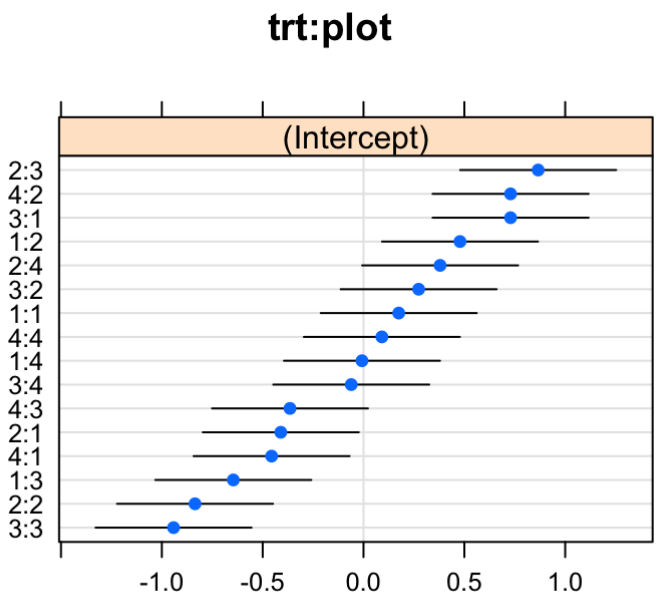
```
tmp = data.frame(resid = re[[1]][,1],
                 trt = substr(rownames(re[[1]]), 1,1))
tmp$fitted = predict(fit2, tmp, re.form=NA)
ggplot(tmp, aes(x=fitted, y=resid)) + geom_point() + theme_minimal()
```




```
ggplot(tmp, aes(sample=resid)) + geom_qq() + theme_minimal()
```




```
library(lattice) # for dotplot function
# BLUPs and 95% prediction intervals for random effects for plots
dotplot(ranef(fit2, condVar=TRUE))[[1]]
```



Conclusion: at both levels (plots and subsamples), the assumption of a normal distribution and equal variance seems adequate.