Random Effects Models

Neil J. Hatfield

Last Updated: April 25, 2023

In this tutorial, we are going to explore using the Random Effects Model in several ANOVA situations.

1 New Key Package

To fit Random Effects (and Mixed Effect) Models, we will need to use the `lme4` package. If you used the `checkSetup` function, then you should have this package installed. If you didn't, you'll want to install this package (`install.packages("lme4")`)

2 Looms and Fabric Strength

Our example comes from Test 1:

A textile company weaves fabric on a large number of looms which they then supply to other companies. As such, they would like the looms to be as homogeneous as possible so that the fabric has the same strength. One of the process engineers suspects that in addition to naturally occurring variation in the fabric strength within a bolt, there is variation in the strength due to what loom gets used. You will be allowed to use four of the company's looms and create 1 bolt of the same type of fabric (1 bolt = 100 yards). You will test four, 3-inch squares from each bolt for the fabric strength.

This is a CRD (One-way ANOVA) situation with an added wrinkle that we are going to need to develop a random sampling method to pick which four looms we will use in the study. Thus, we have a random effect. Our Hasse diagram (Figure 1) will need to reflect this.

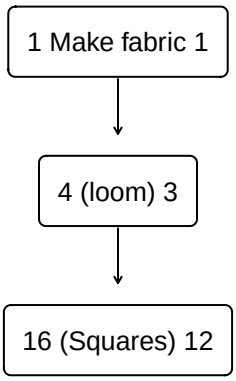

Figure 1: Hasse Diagram fro Loom/Fabric Strength Study

2.1 Fit the Model

We are going to fit both a Fixed and Random Effects model. We do so that you can see the differences/similarities between them and so that we can get useful output (i.e., the ANOVA table.)

```r
# Load Data
fabric <- read.table(
  file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/fabric.csv",
  header = TRUE,
  sep = ","
)

# Set Loom to be a Factor not integers
fabric$loom <- dplyr::recode_factor(
  .x = fabric$loom,
  `1` = "Loom 1",
  `2` = "Loom 2",
  `3` = "Loom 3",
  `4` = "Loom 4"
)

# Fixed effects model
loomFixed <- aov(
  formula = strength ~ loom,
  data = fabric
)

# Random effects model
loomRandom <- lme4::lmer(
  formula = strength ~ (1|loom),
  data = fabric
)
```

2.2 Check Assumptions

To check our assumptions we will need to use the random effects object, `loomRandom`. However, we will need to make use of the `residuals` function rather than `$residuals`.

2.2.1 Gaussian Residuals

Use a QQ plot like usual:

```r
# QQ plot for residuals
car::qqPlot(
  x = residuals(loomRandom), # Notice we're using the random effects model
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Residuals"
)
```

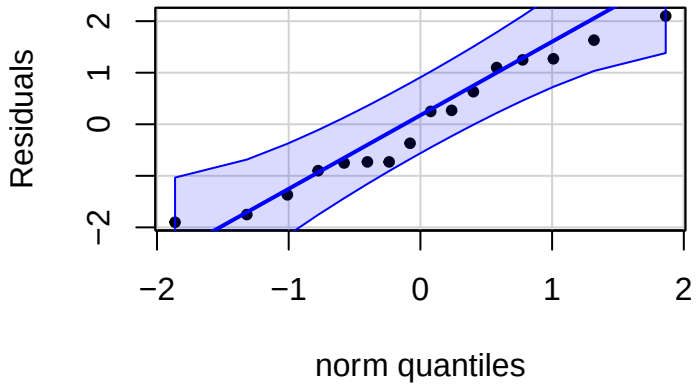

Figure 2: QQ Plot for Residuals

We still work with the QQ plot of Residuals exactly like we would for a fixed effects model.

2.2.2 Gaussian Treatment Effects

We will use another QQ plot for the treatment effects. To get the appropriate values, we will need to use `lme4::ranef` on our random effects model to call up the effects and then use the `whichel` to specify we want the loom factor. (You will need to wrap `ranef` in `unlist` to make sure that `qqPlot` can act on the values.)

```
# QQ plot for the Treatment effects
car::qqPlot(
  x = unlist( # This code will allow us to get the appropriate treatment effects
    lme4::ranef(
      object = loomRandom,
      whichel = c("loom")
    )
  ),
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Treatment Effects"
)
```

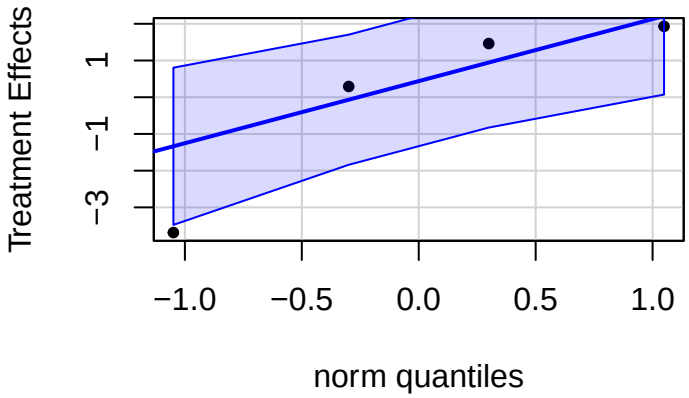

Figure 3: QQ Plot for Residuals

You interpret this plot just like a QQ plot for residuals. Unless there is an extreme violation, you can not

small/moderate violations, like the one here, and note that "we'll proceed with extreme caution".

2.2.3 Homoscedasticity

```
# Strip chart, notice the calls to the random effects model
ggplot(
  data = data.frame(
    residuals = residuals(loomRandom),
    fitted = fitted.values(loomRandom)
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 1) +
  theme_bw() +
  xlab("Fitted values") +
  ylab("Residuals")
```

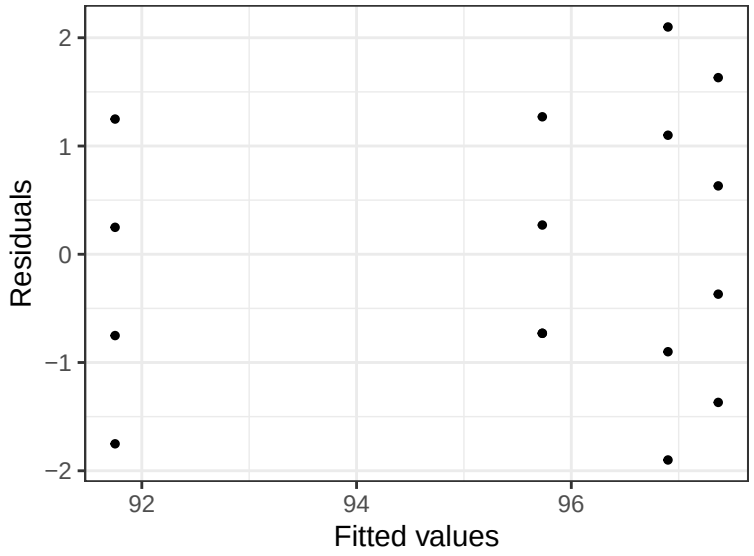

Figure 4: Strip Chart for Loom/Fabric Strength Study

Once again, we use the strip chart exactly as we would for a fixed effects model.

2.2.4 Independence of Observations

We don't know the measurement order so using an index plot and the Durbin-Watson statistic are not going to be meaningful. However, if we think through our study design, we will recognize that we used a valid random sampling process to select the looms. Further, since the looms don't feed into each other, they produced separate bolts of fabric. We used a second random sampling process to select squares of fabric to test. Thus, if there is any relationship between measurement units it will be dependent upon the loom used. Thus, we can reason that we satisfied the assumption of independence of observations.

If we were to know measurement order, then you would build an index plot and interpret the plot just as we would do in a fixed effects model. I would still use `residuals(randomEffectModel)` to call up the residuals.

2.3 Omnibus Results

```r
# The p-value is 0.0002 but R wanted to show 2e-4
# You can tell R to NOT use Scientific Notation with the following:
options(scipen = 999)

# Remember, we can use the fixed effects model to get the appropriate ANOVA table:
parameters::model_parameters(
    model = loomFixed,
    effectsize_type = c("eta", "omega", "epsilon")
) %>%
  knitr::kable(
    digits = 4,
  col.names = c("Source", "SS", "df", "MS", "F", "p-value",
                "Eta Sq.", "Omega Sq.", "Epsilon Sq."),
  caption = "ANOVA Table for Fabric Strength Study",
  align = c('l',rep('c',8)),
  booktab = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 10,
    latex_options = "HOLD_position"
  )
```

Table 1: ANOVA Table for Fabric Strength Study

Source	SS	df	MS	F	p-value	Eta Sq.	Omega Sq.	Epsilon Sq.
loom	89.1875	3	29.7292	15.6813	0.0002	0.7968	0.7335	0.746
Residuals	22.7500	12	1.8958					

2.4 Post Hoc Analysis

For random effects models, we're often interested in both point estimates and confidence intervals for several different parameters. While you can do these for fixed effects models, we often do pairwise comparisons instead.

2.4.1 Point Estimates

One of the parameters of interest for Random Effects is the Grand Mean. We can get this estimate through the following call:

```r
# Random Effects-Grand Mean Point Estimate
lme4::fixef(loomRandom)
```

```
## (Intercept)
##     95.4375
```

```r
# Fixed Effects-Grand Mean Point Estimate
coef(loomFixed)[1]
```

```
## (Intercept)
##     95.4375
```

Notice that the Grand Mean's point estimate is the same regardless of whether you use the Random Effects or the Fixed Effects model. This is due to the fact that we're estimating the same thing in both cases and, more importantly, *we're still thinking about the Grand Mean in the same way* (i.e., a fixed effect).

For Random Effects models, the primary parameters we want to estimate are the variances for treatments (σ^2_α) and residuals (σ^2). The best way to examine these is look at the output of the `summary` call on the random effects model:

```
# Point Estimates of Variance Components
summary(loomRandom)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: strength ~ (1 | loom)
##    Data: fabric
##
## REML criterion at convergence: 63.2
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.38018 -0.57260 -0.04342  0.82574  1.52491
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  loom     (Intercept) 6.958    2.638
##  Residual             1.896    1.377
## Number of obs: 16, groups:  loom, 4
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)   95.438      1.363   70.01
```

We want to focus on the "Random effects" portion of the raw output. The row labeled "loom" contains the point estimates for σ^2_α and σ_α: 6.958 and 2.638 respectively. Be sure that you double check which you're asked to report: Variance or Standard Deviation. The row labeled "Residual" has our point estimates for σ^2 and σ.

Keep in mind that `summary` returns raw output, which does not look professional. You can use this to look at your results, but you'll want to format them for reports. (Note: you can also get the point estimate for the Grand Mean from the "Fixed effects" listing.)

2.4.2 Optional: Factor Effects

If you wanted to examine point estimates of the treatment *effects*, you can also do so:

```
# Random Effects-Treatment Effects
lme4::ranef(loomRandom)
```

```
## $loom
##        (Intercept)
## Loom 1   1.9309741
## Loom 2  -3.6864051
## Loom 3   0.2925718
## Loom 4   1.4628591
##
## with conditional variances for "loom"
```

```
# Fixed Effects-Treatment Effects
## Don't forget to set the constraint
dummy.coef(loomFixed)$loom
```

```
##  Loom 1  Loom 2  Loom 3  Loom 4
##  2.0625 -3.9375  0.3125  1.5625
```

Notice that these *aren't* in agreement. This is a result of the differences in the ways that we are thinking about our factors between the two models.

2.4.3 Confidence Intervals

For confidence intervals, you'll need to remember to first apply your chosen method to account for the Multiple Comparison/Simultaneous Inference problem. For this example, we had chosen to control the SCI at $\mathcal{E}_I = 0.1$ with the Bonferonni method. Our testing family consists of 4 acts: the omnibus, and confidence intervals for the Grand Mean ($\mu_{..}$), variance of treatments (σ^2_α), and variance of residuals (σ^2). This makes our individualized Type I error rates 0.025 ($\mathcal{E}^\star_I = 0.1/4$), which we will use as our Unusualness Thresholds.

We can use the following code to make a professional looking table for our intervals:

```
# Confidence intervals for our random effects parameters
intervals <- confint(
  object = loomRandom,
  level = 0.975,
  oldNames = FALSE
)

row.names(intervals) <- c(
  "Treatment Standard Deviation",
  "Residual Standard Deviation",
  "Grand Mean"
)

knitr::kable(
  intervals,
  digits = 3,
  caption = "Upper and Lower Confidence Bounds-90\\% SCI, Bonferroni Adj.",
  align = c('l',rep('c',2)),
  booktab = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

Table 2: Upper and Lower Confidence Bounds-90% SCI, Bonferroni Adj.

	1.25 %	98.75 %
Treatment Standard Deviation	1.013	6.842
Residual Standard Deviation	0.926	2.361
Grand Mean	91.696	99.179

Notice that Table 2 provides estimates of the *Standard Deviations* not the Variances. If you need the Variances, you may just square the lower and upper bounds.

2.4.3.1 A Caution If you wanted to get confidence intervals for the *Fixed Effects* model's parameters of interest (i.e., Grand Mean and Treatment Effects), you can use the following code:

```r
# Confidence intervals for Fixed Effects Model
confint(
  object = loomFixed,
  level = 0.975,
  oldNames = FALSE
)
```

```
##                 1.25 %   98.75 %
## (Intercept) 94.5562774 96.318723
## loom1        0.5361776  3.588822
## loom2       -5.4638224 -2.411178
## loom3       -1.2138224  1.838822
```

You'll notice that interval for the Grand Mean is (94.56, 96.32) from this output, which isn't the same as the (91.70, 99.18) from Table 2. The confidence intervals from the Fixed Effects model are *too narrow* as they have not accounted for the fact that our treatments came from a randomized selection process. This is true for the Grand Mean as well as the treatment effects themselves. Keep in mind that for Random Effects models, our interest is on the *Variance of Treatment Effects* not the actual treatment effects.

3 Code Appendix

```
# Setting Document Options
knitr::opts_chunk$set(
  echo = FALSE,
  warning = FALSE,
  message = FALSE,
  fig.align = "center"
)

packages <- c("tidyverse", "knitr", "kableExtra",
              "parameters", "hasseDiagram", "DescTools",
              "lme4")
lapply(packages, library, character.only = TRUE)

options(knitr.kable.NA = "")
options(contrasts = c("contr.sum", "contr.poly"))

source("https://raw.github.com/neilhatfield/STAT461/master/rScripts/ANOVATools.R")
modelLabels <- c("1 Make fabric 1", "4 (loom) 3", "16 (Squares) 12")
modelMatrix <- matrix(
  data = c(FALSE, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE, TRUE, FALSE),
  nrow = 3,
  ncol = 3,
  byrow = FALSE
)
hasseDiagram::hasse(
 data = modelMatrix,
 labels = modelLabels
)

# Load Data
fabric <- read.table(
  file = "https://raw.github.com/neilhatfield/STAT461/master/dataFiles/fabric.csv",
  header = TRUE,
  sep = ","
)

# Set Loom to be a Factor not integers
fabric$loom <- dplyr::recode_factor(
  .x = fabric$loom,
  `1` = "Loom 1",
  `2` = "Loom 2",
  `3` = "Loom 3",
  `4` = "Loom 4"
)

# Fixed effects model
loomFixed <- aov(
  formula = strength ~ loom,
  data = fabric
)

# Random effects model
```

```r
loomRandom <- lme4::lmer(
  formula = strength ~ (1|loom),
  data = fabric
)

# QQ plot for residuals
car::qqPlot(
  x = residuals(loomRandom), # Notice we're using the random effects model
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Residuals"
)

# QQ plot for the Treatment effects
car::qqPlot(
  x = unlist( # This code will allow us to get the appropriate treatment effects
    lme4::ranef(
      object = loomRandom,
      whichel = c("loom")
    )
  ),
  distribution = "norm",
  envelope = 0.90,
  id = FALSE,
  pch = 20,
  ylab = "Treatment Effects"
)

# Strip chart, notice the calls to the random effects model
ggplot(
  data = data.frame(
    residuals = residuals(loomRandom),
    fitted = fitted.values(loomRandom)
  ),
  mapping = aes(x = fitted, y = residuals)
) +
  geom_point(size = 1) +
  theme_bw() +
  xlab("Fitted values") +
  ylab("Residuals")

# The p-value is 0.0002 but R wanted to show 2e-4
# You can tell R to NOT use Scientific Notation with the following:
options(scipen = 999)

# Remember, we can use the fixed effects model to get the appropriate ANOVA table:
parameters::model_parameters(
    model = loomFixed,
    effectsize_type = c("eta", "omega", "epsilon")
) %>%
  knitr::kable(
```

```
    digits = 4,
  col.names = c("Source", "SS", "df", "MS", "F", "p-value",
                "Eta Sq.", "Omega Sq.", "Epsilon Sq."),
  caption = "ANOVA Table for Fabric Strength Study",
  align = c('l',rep('c',8)),
  booktab = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 10,
    latex_options = "HOLD_position"
  )

# Random Effects-Grand Mean Point Estimate
lme4::fixef(loomRandom)

# Fixed Effects-Grand Mean Point Estimate
coef(loomFixed)[1]
# Point Estimates of Variance Components
summary(loomRandom)

# Random Effects-Treatment Effects
lme4::ranef(loomRandom)

# Fixed Effects-Treatment Effects
## Don't forget to set the constraint
dummy.coef(loomFixed)$loom

# Confidence intervals for our random effects parameters
intervals <- confint(
  object = loomRandom,
  level = 0.975,
  oldNames = FALSE
)

row.names(intervals) <- c(
  "Treatment Standard Deviation",
  "Residual Standard Deviation",
  "Grand Mean"
)

knitr::kable(
  intervals,
  digits = 3,
  caption = "Upper and Lower Confidence Bounds-90\\% SCI, Bonferroni Adj.",
  align = c('l',rep('c',2)),
  booktab = TRUE
) %>%
  kableExtra::kable_styling(
    bootstrap_options = c("striped", "condensed"),
    font_size = 12,
    latex_options = "HOLD_position"
  )
```

```r
# Confidence intervals for Fixed Effects Model
confint(
  object = loomFixed,
  level = 0.975,
  oldNames = FALSE
)
```