FES 524 Winter 2022 Lab 5

Contents

Additional random effects due to variation at multiple scales

In Lab 5 you will learn to fit models with multiple, nested random effects as well as multiple fixed effects. The random effects are based on elements of the study design, which involves blocking on watersheds and then measuring the two factors of interest on distinct physical units. The distinct physical units, which we will discuss later in the lab, vary in size. The dataset this week is an example of a *blocked split-plot* design, although it is possible to work with physical units of varying sizes and not have a split plot design.

Load R packages needed today

These packages are for data manipulation, plotting, fitting a linear mixed model, and doing comparisons of means among groups.

```
library(dplyr)
library(ggplot2)
library(nlme)
library(emmeans)
```

Read in the dataset

We will be working with the dataset `lab5.example.data.txt`, so make sure you have this file and have changed your working directory appropriately. As usual when we read in a dataset we'll take a look at the structure and make any necessary changes. Our two factors of interest today are the overstory species (`overstory`) and the type of tree the litter came from (`litterspp`). `biomass` is the response variable.

```
dbiomass = read.table("lab5.example.data.txt", header = TRUE)

head(dbiomass) # Look at the first 6 lines of the data set
```

```
  watershed overstory litterspp biomass
1         A        RA       ACMA       6
2         G        RA       ACMA       7
3         A        RA       ALRU      10
4         C        RA       ALRU      11
5         C        RA       ACMA      18
6         F        RA       ALRU      18
```

Check the structure of the dataset. I am not going to change the factor levels today so am leaving the factors of interest as `character` variables.

```
'data.frame':   64 obs. of  4 variables:
 $ watershed: chr  "A" "G" "A" "C" ...
 $ overstory: chr  "RA" "RA" "RA" "RA" ...
 $ litterspp: chr  "ACMA" "ACMA" "ALRU" "ALRU" ...
 $ biomass  : int  6 7 10 11 18 18 24 25 29 29 ...
```

A nested study design

There were three different sizes of physical units in this study design. The largest physical units are *watersheds*. The researchers picked two different *stands* in each watershed, one with a primarily Douglas-fir overstory and one with a primarily red alder overstory. The stands are the middle-sized physical units. Within each stand, the researchers put out four bags of litter in different locations in the stand, one for each litter type of interest. These litter bag locations are the smallest physical units in the study.

The measurement of the response was done at the level of the litter bag locations (one measurement of biomass for each litter bag in each stand in each watershed). One factor of interest was measured at the stand level (species of dominant overstory) and the other factor of interest was applied at the litter bag location level (type of litter).

Based on the description of the physical units, this study has a *nested* structure, with stands nested in watersheds and litter bag locations nested in stands. We recognize a nesting structure when we see that, e.g., the stand in one watershed is distinct from a stand in another watershed.

Implicit vs explicit names for physical units

Every unique watershed in the data is represented by a unique letter. The current dataset has *implicit* names for stands and litter bag locations. For example, the stands within watersheds do not have unique names. Instead, stands are represented by the overstory species factor. This can lead to confusion between the factor of interest that we will be using as a fixed effect and the physical units that cause random variation and should be treated as random effects. To avoid this confusion and any mistakes it might cause, we'll be making a new variable called `stand` where we'll give each stand present in the study a unique name. This is called *explicit* naming. See https://bbolker.github.io/mixedmodels-misc/glmmFAQ.html#nested-or-crossed for more discussion.

Because we will be working with linear mixed models, we don't have to uniquely name the smallest physical unit. The litter bag locations are our observation-level units and we know that we will get the residual error term (the observation-level random effect) by default in `lme()`. However, in your own work you might consider unique naming of all physical units just to help you keep factors of interest vs physical units straight. Such variables can also be needed when fitting generalized linear models (GLM's).

We will use the `interaction()` function to make unique names for stands. This function works well for perfectly crossed variables, where every level of one variable occurs with every level of the second variable. Another option is `paste()`, which we will see next week.

This work will be done in `mutate()` to avoid dollar sign notation. I'll make a variable to represent the unique litter bag locations while I'm at it for practice, even though I don't need this for analysis.

```
dbiomass = mutate(dbiomass,
                  stand = interaction(watershed, overstory),
                  location = interaction(stand, litterspp) )
head(dbiomass)
```

```
  watershed overstory litterspp biomass stand  location
1         A         RA      ACMA       6  A.RA A.RA.ACMA
2         G         RA      ACMA       7  G.RA G.RA.ACMA
3         A         RA      ALRU      10  A.RA A.RA.ALRU
4         C         RA      ALRU      11  C.RA C.RA.ALRU
5         C         RA      ACMA      18  C.RA C.RA.ACMA
6         F         RA      ALRU      18  F.RA F.RA.ALRU
```

Now each stand is uniquely identified and we have a variable that clearly represents the stand-to-stand variation to use as a random effect in the model.

Initial data exploration

Summary statistics

We'll be looking at our usual summary statistics. It would be appropriate to also look at summary statistics for each factor separately as well as for the factor combinations, which is not shown here to save space.

Several things to notice this week:

1. Biomass is strictly positive (doesn't include 0).
2. Standard deviations vary wildly among the combined factor groups.
3. The highest value for biomass is more than 80 times higher than the lowest value.

```
( sumdat = dbiomass %>%
    group_by(overstory, litterspp) %>%
    summarise(n = n(),
              mean = mean(biomass),
              sd = sd(biomass),
              min = min(biomass),
              max = max(biomass),
              .groups = "drop") )
```

```
# A tibble: 8 x 7
  overstory litterspp     n  mean    sd   min   max
  <chr>     <chr>     <int> <dbl> <dbl> <int> <int>
1 DF        ACMA          8  80.4  71.1    29   249
2 DF        ALRU          8 127.  107.     50   298
3 DF        PSME          8 189.  152.     53   500
4 DF        TSHE          8 163.   89.3    51   316
5 RA        ACMA          8  29.5  23.5     6    80
6 RA        ALRU          8  28.2  14.5    10    45
7 RA        PSME          8 154.   94.1    56   295
8 RA        TSHE          8 103.   65.2    30   204
```

```
# Examine the number of observations in the groups
    # We're looking for balanced vs unbalanced data
xtabs(~overstory + litterspp, data = dbiomass )
```

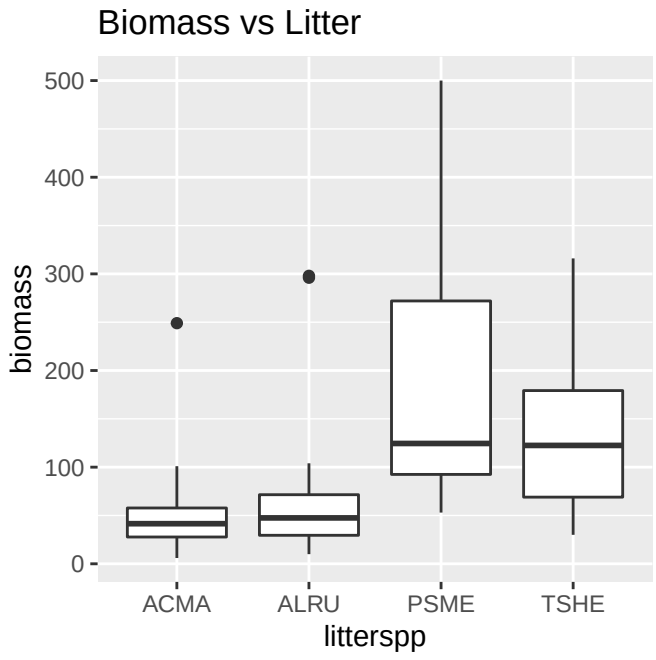
```
         litterspp
overstory ACMA ALRU PSME TSHE
       DF    8    8    8    8
       RA    8    8    8    8
```

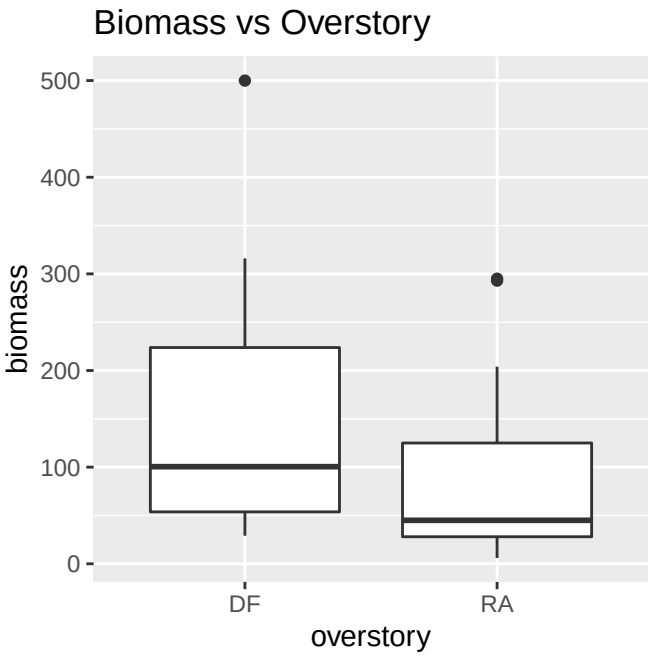
Graphical data exploration

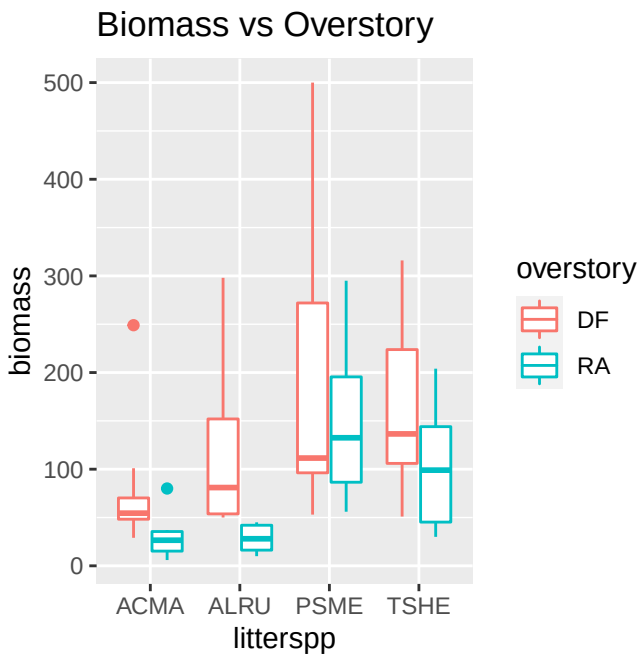
Boxplots Because we are working with only categorical variables and have a bit more data this week, we can use boxplots in our initial data exploration. We can see median, range, and the interquartile range in boxplots to get a sense of what the raw data look like.

In these data we see a lot of long tails and extremely different ranges among groups. I also note that the deciduous litters ACMA and ALRU look pretty different than the conifer litters, although that difference may depend on the overstory species (as expected).

```
# Graphical exploration

# Plot the raw data as boxplots
# First biomass vs each explanatory
qplot(x = litterspp, y = biomass,
      data = dbiomass,
     geom = "boxplot",
     main = "Biomass vs Litter")

qplot(x = overstory, y = biomass,
      data = dbiomass,
     geom = "boxplot",
     main = "Biomass vs Overstory")

# Factor combination: color by overstory, litterspp on x axis
qplot(x = litterspp, y = biomass,
      color = overstory,
     data = dbiomass,
      geom = "boxplot",
     main = "Biomass vs Overstory")
```





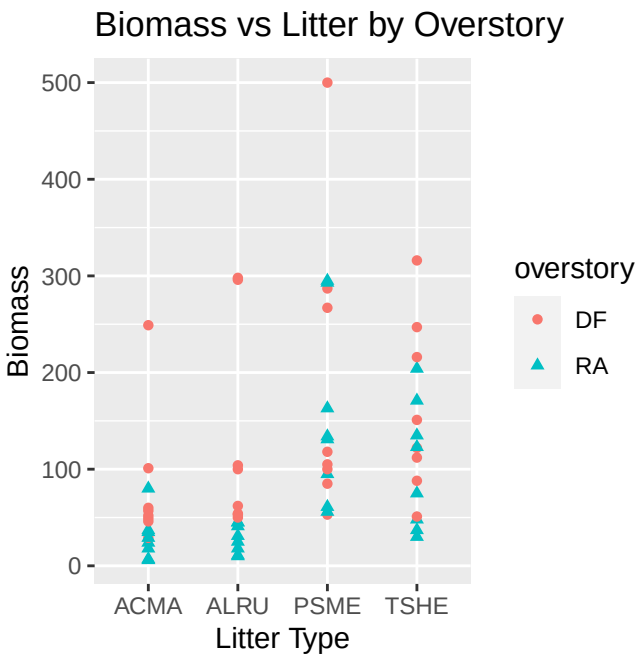


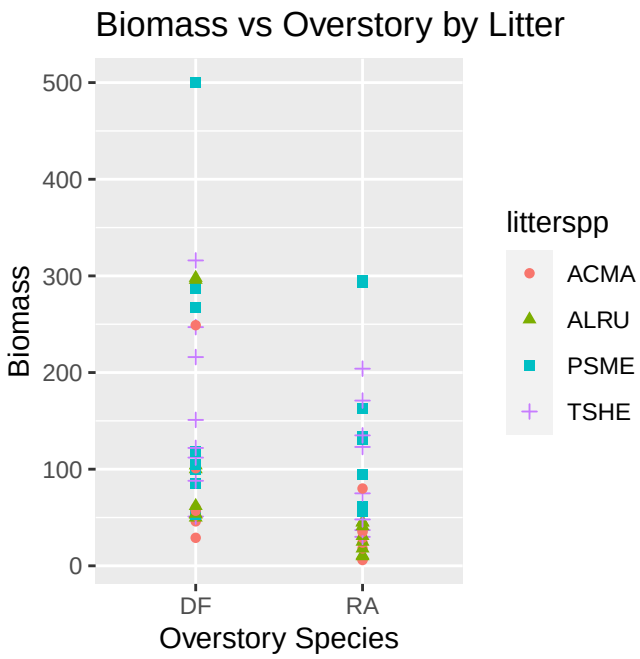
Scatterplots and jittering Here are the scatterplots we've been making each week. Like in Lab 4, we will use colors and shapes to add dimensions to the graphics.

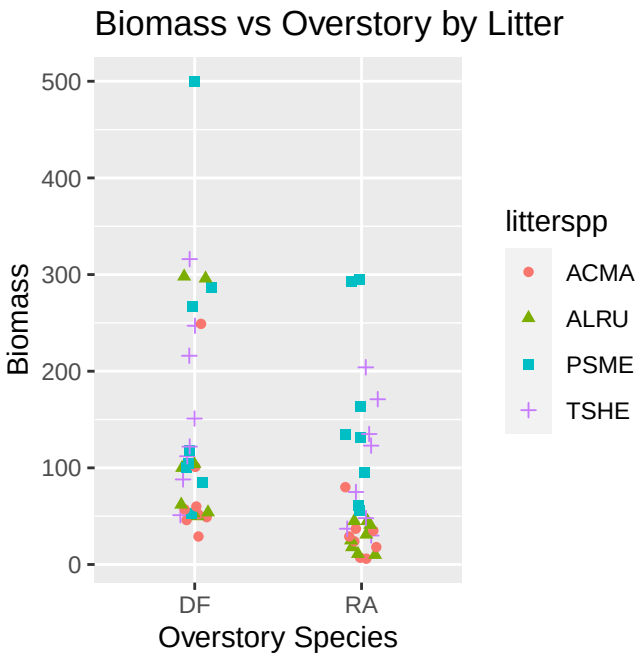
As our datasets get larger, it can be more difficult to see individual points in a scatterplot because points are all on top of each other. To solve that problem, we can *jitter* the points apart. Below you will see the use of `geom_jitter()`. Setting the `width` tells how much to jitter the points. I tend to jitter only a small amount in a scatterplot of groups like this. Notice the switch to using `ggplot()` directly when we making graphics like this as they are too complicated for `qplot()`.

I'm seeing similar patterns among groups here as I noted in the boxplots, but with an ability to see the individual points better.

```
qplot(x = litterspp, y = biomass,
      color = overstory,
      shape = overstory,
      data = dbiomass,
      xlab = "Litter Type",
      ylab = "Biomass",
      main = "Biomass vs Litter by Overstory")

# scatter plot of biomass vs overstory
qplot(x = overstory, y = biomass,
      color = litterspp,
      shape = litterspp,
     data = dbiomass,
     xlab = "Overstory Species",
     ylab = "Biomass",
     main = "Biomass vs Overstory by Litter")

# Add jitter
ggplot(dbiomass, aes(x = overstory,
                     y = biomass,
                 color = litterspp,
                 shape = litterspp) ) +
    geom_jitter(width = .1, height = 0) +
    labs(x = "Overstory Species",
         y = "Biomass",
         title = "Biomass vs Overstory by Litter")
```



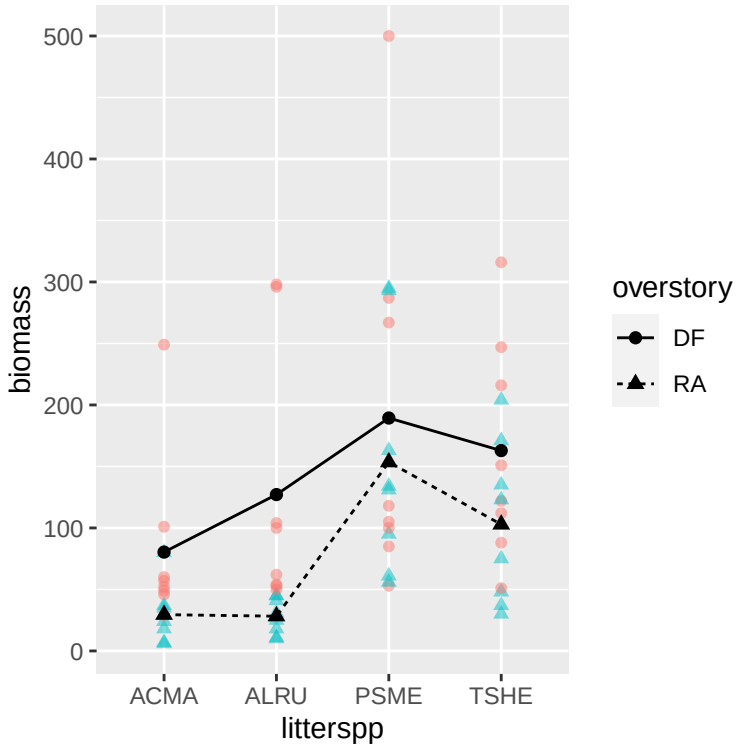




Interaction plot Because we have factors that are perfectly crossed, we need to think about the interaction. We'll explore the possibility that the effect of overstory species depends on the litter type, which researchers expected to have, with an interaction plot like we made in Lab 4. This is an exploratory plot, not a final plot to be included in a results section.

Notice that the difference in mean biomass across overstory is pretty small for the PSME litter but comparatively larger for ALRU. As expected, there appears to be an interaction.

```
ggplot(dbiomass, aes(x = litterspp,
                     y = biomass,
                 group = overstory,
                 linetype = overstory,
                 shape = overstory) ) +
    geom_point(alpha = .5, size = 1.75,
               aes(color = overstory) ) + # Add raw data
    stat_summary(fun = mean, geom = "point", size = 2) + # Add points for means
    stat_summary(fun = mean, geom = "line") # Connect points with lines
```



Fitting a linear mixed model with nested random effects in `lme()`

We will fit a linear mixed model using `lme()` from package **nlme**, where `watershed` and `stand` are random effects and the two factors of interest, `overstory` and `litterspp`, are fixed effects. We will include a term for the interaction between `overstory` and `litterspp`.

This week I use the short-cut coding for the fixed effects. Using the symbol `*` with two variables indicates I am putting each variable plus the interaction between the variables into the model. So `litterspp*overstory` coding expands to `litterspp + overstory + litterspp:overstory`. This is convenient for typing, but is less understandable when you come back to your code and are trying to figure out what model you fit.

Notice the use of the forward slash, `/`, in the random effects. The forward slash represents *nesting* in `lme()`. In the code for the `random` argument we are stating that `stand` is nested in `watershed`. The random effect of litter bag locations is the observation-level random effect, which is the residual error from the model.

```
model1 = lme(biomass ~ litterspp*overstory,
             random = ~1|watershed/stand,
               data = dbiomass)
```

Checking that the model structure is coded correctly

Before we check our assumptions, I wanted to take a moment and review how R and other software packages do what we tell them to even if what we are doing is wrong.

If we take a look at `model1`, we can check the structure of the random effects by examining the `Number of Groups` section. This tells us we have 8 watersheds in our data, which is true. It also tells us we have 16 stands. Since there are 2 stands for each of 8 watersheds, this is also true. The number in the line above, `Number of Observations`, matches the number of rows in our dataset. The structure of the model reflects the structure of our data, which makes us confident that we fit our random effects correctly.

```
model1
```

```
Linear mixed-effects model fit by REML
  Data: dbiomass
  Log-restricted-likelihood: -337.7844
  Fixed: biomass ~ litterspp * overstory
```

```
              (Intercept)             littersppALRU             littersppPSME
                  80.375                    46.750                   109.000
            littersppTSHE               overstoryRA littersppALRU:overstoryRA
                  82.500                   -50.875                   -48.000
littersppPSME:overstoryRA littersppTSHE:overstoryRA
                  15.000                    -9.125

Random effects:
 Formula: ~1 | watershed
        (Intercept)
StdDev:    24.87125

 Formula: ~1 | stand %in% watershed
        (Intercept) Residual
StdDev:  0.02601692 84.03225

Number of Observations: 64
Number of Groups:
           watershed stand %in% watershed
                   8                  16
```

Look at what happens if we were to put our variables "backwards" in `random`, essentially saying that watersheds are nested in stands. This happens a lot, especially for folks trained in SAS before they started learning R.

```
lme(biomass ~ litterspp*overstory,
    random = ~1|stand/watershed,
    data = dbiomass)
```

```
Linear mixed-effects model fit by REML
  Data: dbiomass
  Log-restricted-likelihood: -338.058
  Fixed: biomass ~ litterspp * overstory
              (Intercept)             littersppALRU             littersppPSME
                  80.375                    46.750                   109.000
            littersppTSHE               overstoryRA littersppALRU:overstoryRA
                  82.500                   -50.875                   -48.000
littersppPSME:overstoryRA littersppTSHE:overstoryRA
                  15.000                    -9.125

Random effects:
 Formula: ~1 | stand
        (Intercept)
StdDev:    17.01507

 Formula: ~1 | watershed %in% stand
        (Intercept) Residual
StdDev:    17.01547 84.26719

Number of Observations: 64
Number of Groups:
               stand watershed %in% stand
                  16                  16
```

The model fit without complaint, and if we weren't paying attention we might go on and use this model for inference. But if we check the `Number of Groups` we see that the model assumes the wrong number of watersheds (16 instead of 8). This would alert us that we defined the model incorrectly.

In addition, in `lme()` we can also incorrectly add the observation-level random effect as an extra random effect and the model will still fit.

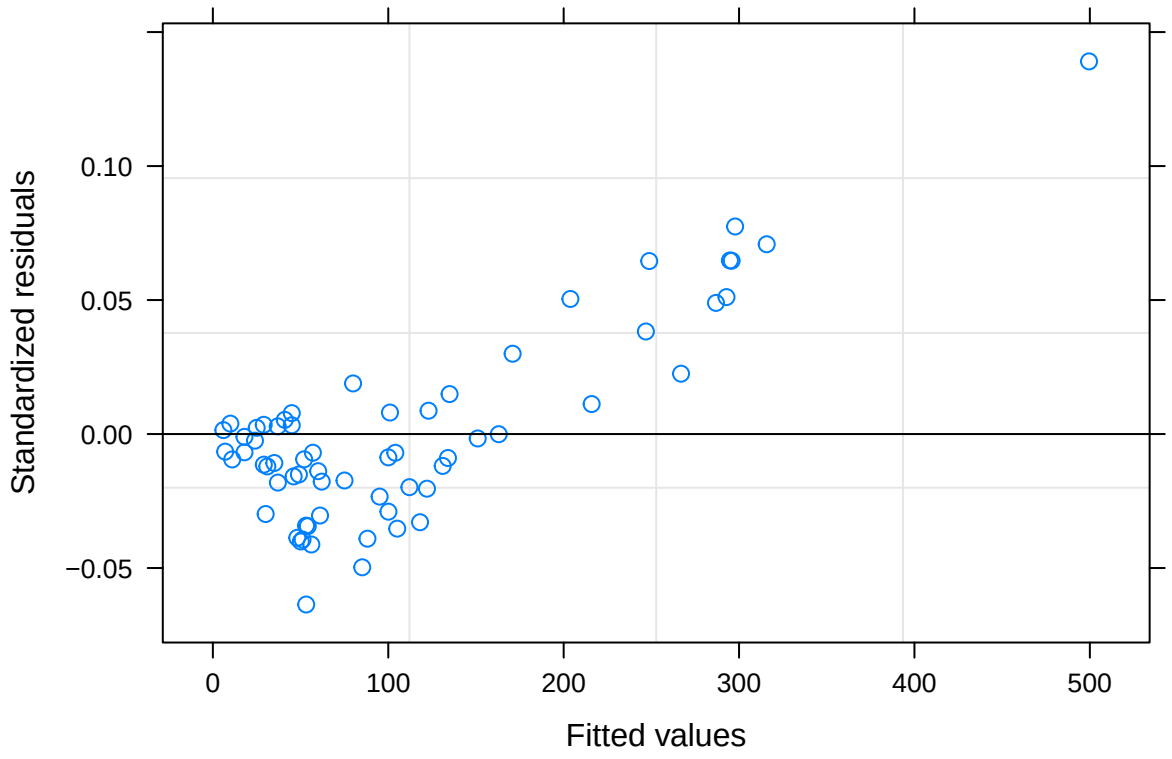
```
lme(biomass ~ litterspp*overstory,
    random = ~1|watershed/stand/location,
    data = dbiomass)
```

```
Linear mixed-effects model fit by REML
  Data: dbiomass
  Log-restricted-likelihood: -337.7844
  Fixed: biomass ~ litterspp * overstory
             (Intercept)             littersppALRU             littersppPSME
                  80.375                    46.750                   109.000
           littersppTSHE               overstoryRA littersppALRU:overstoryRA
                  82.500                   -50.875                   -48.000
littersppPSME:overstoryRA littersppTSHE:overstoryRA
                  15.000                    -9.125

Random effects:
 Formula: ~1 | watershed
        (Intercept)
StdDev:    24.87063

 Formula: ~1 | stand %in% watershed
        (Intercept)
StdDev:  0.02562114

 Formula: ~1 | location %in% stand %in% watershed
        (Intercept) Residual
StdDev:    83.96899 3.262773

Number of Observations: 64
Number of Groups:
                         watershed               stand %in% watershed
                                 8                                 16
location %in% stand %in% watershed
                                64
```

The model fits without warning, even though this extra random effect is entirely confounded with the residual error term. There isn't much in the output to tell use something is wrong, although we can see the number of groups for locations is the same as the number of observations.

However, this is a good example of a "pattern" in the residual vs fitted plot. The strong linear pattern indicates something is seriously wrong with the model. This particular pattern is one I've seen in models which incorrectly used an observation-level random effect in a linear mixed model.

```
plot( lme(biomass ~ litterspp*overstory,
          random = ~1|watershed/stand/location,
          data = dbiomass) )
```



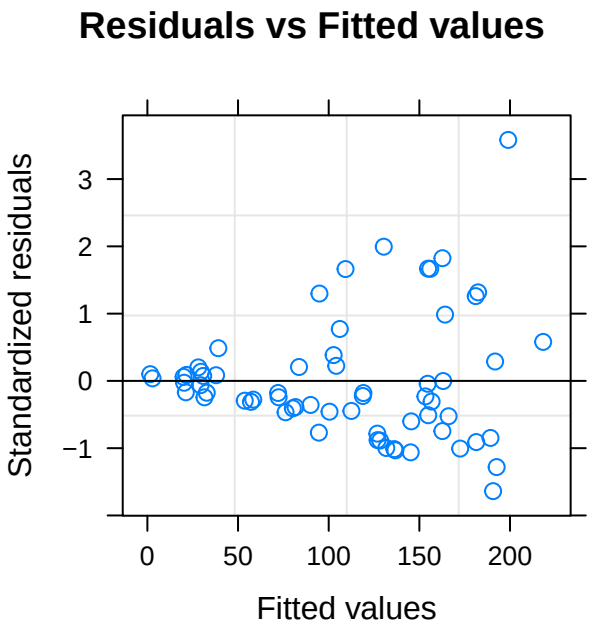
Checking model assumptions

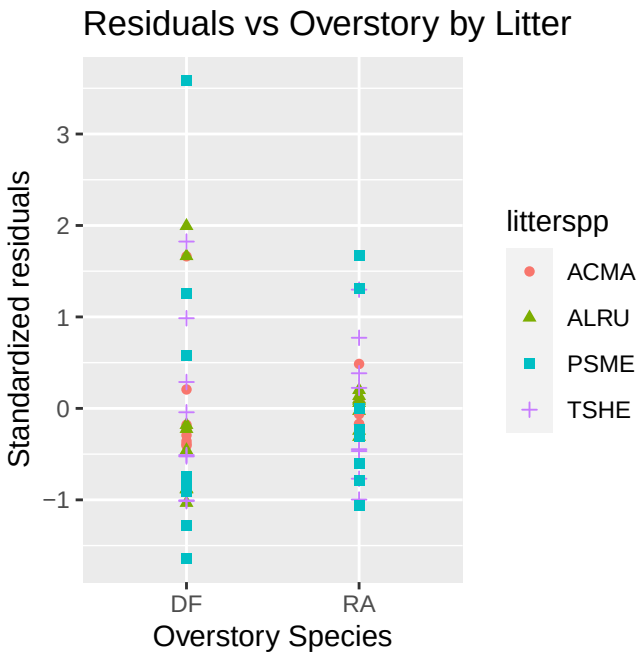
As always, we'll need to check the assumptions of the model using residual plots. We can add the residuals to the dataset `dbiomass`, and then plot the residuals vs the fitted values, the residuals vs the explanatory variables, and check the normality/symmetry of the residuals with a boxplot.

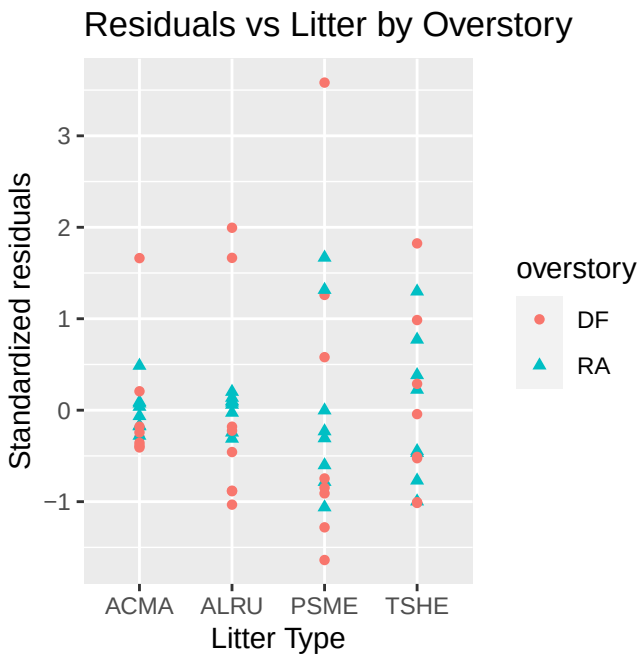
The residual plots from `model1` the model indicate a problem. In the first plot we see that the variance of the residuals increases with the fitted values, and further plots showed us that the groups don't appear to have homogeneous variances for either factor variable. The residuals also show a long right tail.

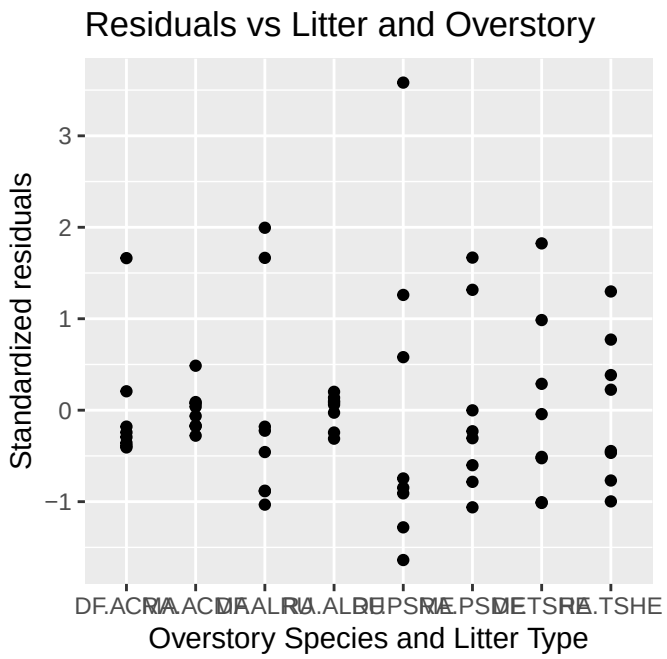
```
# Save the residual values for assumption checking.
dbiomass$res = resid(model1, type = "pearson")

# Plot residuals vs fitted values
plot(model1, main = "Residuals vs Fitted values")

# Make scatter plots of residuals vs explanatory variables
# overstory
qplot(x = overstory, y = res,
      color = litterspp,
      shape = litterspp,
      data = dbiomass,
      xlab = "Overstory Species",
      ylab = "Standardized residuals",
      main = "Residuals vs Overstory by Litter")

# litter type
qplot(x = litterspp, y = res,
      shape = overstory,
      color = overstory,
      data = dbiomass,
```

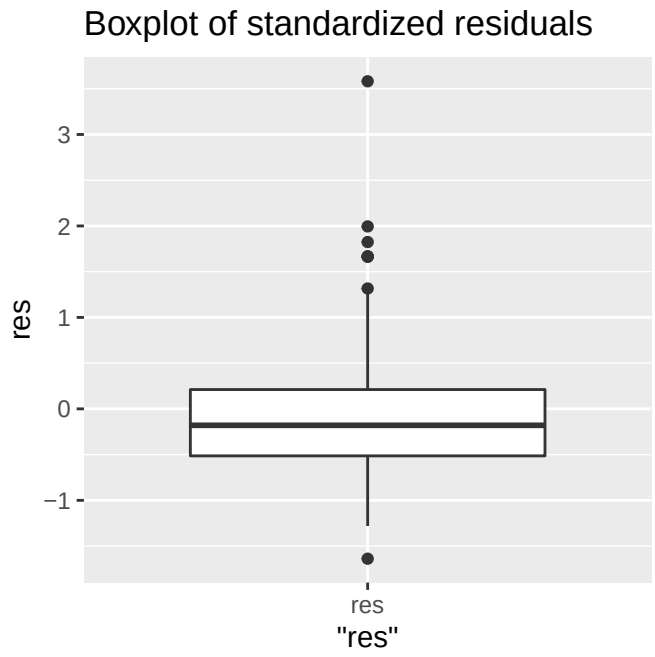
```
      xlab = "Litter Type",
      ylab = "Standardized residuals",
      main = "Residuals vs Litter by Overstory")

# combination of overstory and litter type
qplot(x = interaction(overstory, litterspp), y = res,
      data = dbiomass,
      xlab = "Overstory Species and Litter Type",
      ylab = "Standardized residuals",
      main = "Residuals vs Litter and Overstory")

# Check symmetry of residuals with boxplot
qplot(x = "res", y = res,
      data = dbiomass,
     geom = "boxplot",
     main = "Boxplot of standardized residuals")
```











Refitting the model when the assumptions are not met

The assumptions for `model1` are clearly not met. We could try to address the problem of nonconstant variance by allowing variances to differ among the levels of both of the factors, much like we saw last week. However, this would be a very complicated model. In addition, allowing for variances to differ among groups does not necessarily address that long right tail we see in the boxplot.

Let's think about our observed data and the residuals a little more. The values of our response, biomass, are strictly positive (so don't include zero). The residuals are right-skewed and the residual variance increases with the mean. A dataset like this is a good candidate for modeling the response with either a log-normal or a gamma distribution. Using the gamma distribution would mean we would have to switch to using a generalized linear *mixed* model, which we are not covering in this class. To fit a log-normal model, though, we can do a natural logarithm transformation on our response and stick with a linear mixed model.

Let's transform `biomass` and use `log(biomass)` as the response in a new model. We would need to go back and remake our exploratory plots with the transformed response variable, but we are not going to take the time today (but we would take

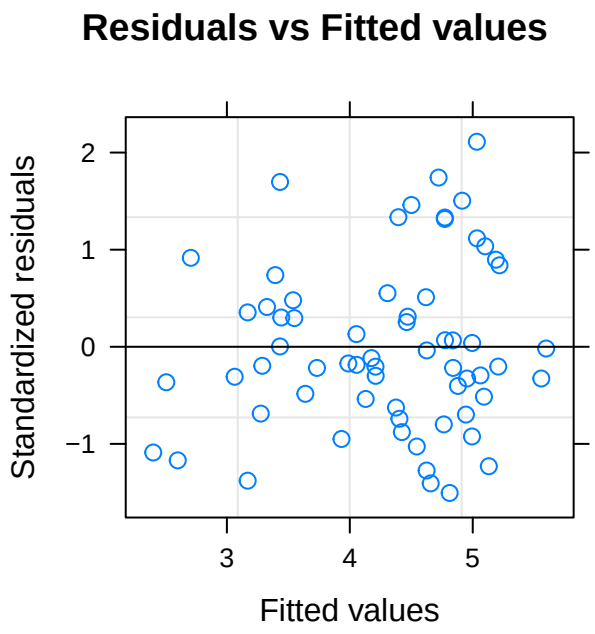
the time if this were a "real" analysis). Instead, we'll fit a second model, `model2`, using `log(biomass)`. Notice I write this out rather than using the newly transformed variable `lbio`. This is going to allow me to take advantage of some tools in **emmeans** later.

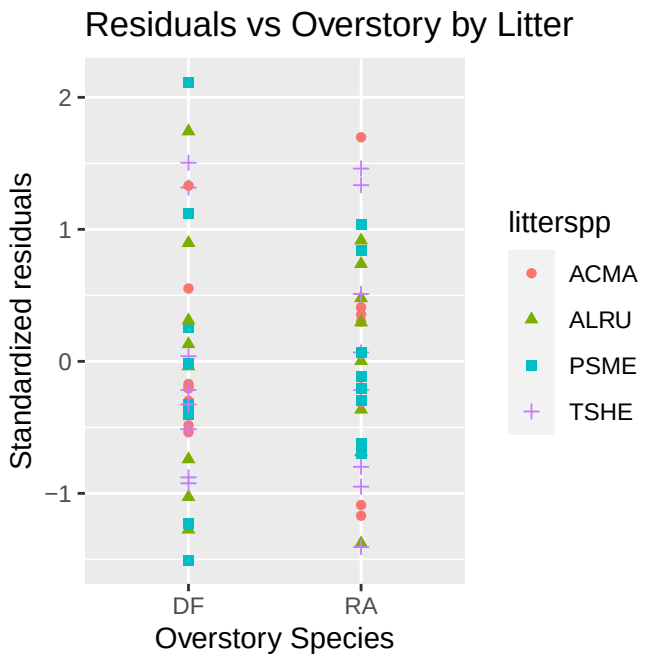
```
# We could make a new response variable, on the log scale,
    # for making exploratory plots
dbiomass$lbio = log(dbiomass$biomass)

# Fit a model with the transformed response
model2 = lme(log(biomass) ~ litterspp*overstory,
           random = ~1|watershed/stand,
             data = dbiomass)
```

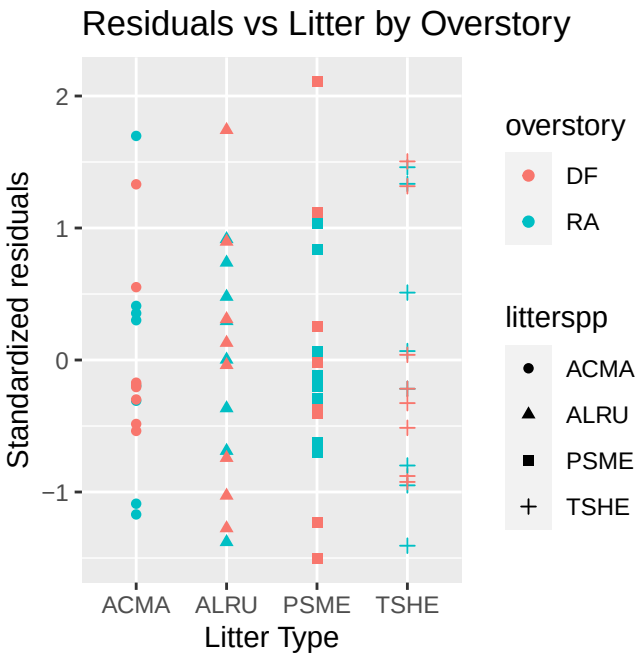
We'll still need to check our assumptions for this new model before we can use it to make inference. Things are looking much better, with reasonably constant variances among groups and more symmetric residuals.

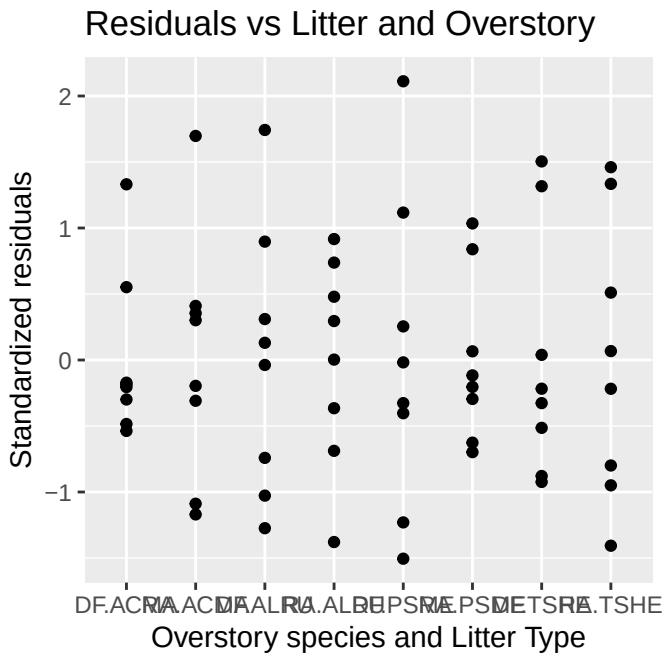
There is kind of a funny pattern in the residuals vs fitted values plot in the section in the middle where we have few data points. I'm guessing this is caused by having few data in that area and not anything more sinister.

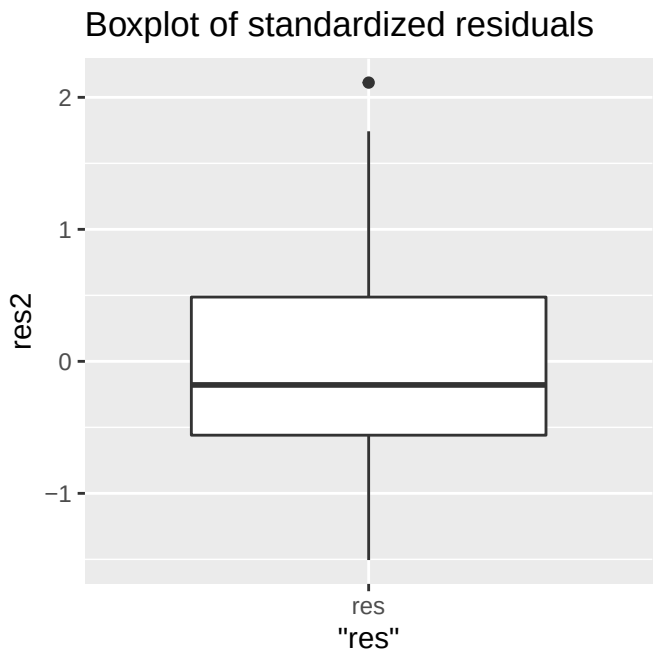
```
# Compute and save residual values
dbiomass$res2 = resid(model2, type = "pearson")

# Plot residuals vs fitted values
plot(model2, main = "Residuals vs Fitted values")

# Make scatter plots of residuals vs explanatory variables
# overstory
qplot(x = overstory, y = res2,
      color = litterspp,
      shape = litterspp,
      data = dbiomass,
      xlab = "Overstory Species",
      ylab = "Standardized residuals",
      main = "Residuals vs Overstory by Litter")

# litter type
qplot(x = litterspp, y = res2,
      color = overstory,
      shape = litterspp,
      data = dbiomass,
      xlab = "Litter Type",
      ylab = "Standardized residuals",
      main = "Residuals vs Litter by Overstory")

# interaction of overstory and litter type
qplot(x = interaction(overstory, litterspp), y = res2,
      data = dbiomass,
      xlab = "Overstory species and Litter Type",
      ylab = "Standardized residuals",
      main = "Residuals vs Litter and Overstory")

# Check symmetry of residuals with boxplot
qplot(x = "res", y = res2,
      data = dbiomass,
     geom = "boxplot",
     main = "Boxplot of standardized residuals")
```


Residuals vs Fitted values
Standardized residuals
2
1
0
−1
3
4
5
Fitted values


Residuals vs Overstory by Litter
Standardized residuals
2
1
0
−1
DF
RA
Overstory Species
litterspp
ACMA
ALRU
PSME
TSHE


Residuals vs Litter by Overstory
Standardized residuals
2
1
0
−1
ACMA
ALRU
PSME
TSHE
Litter Type
overstory
DF
RA
litterspp
ACMA
ALRU
PSME
TSHE


Residuals vs Litter and Overstory
Standardized residuals
2
1
0
−1
Overstory species and Litter Type



Model results

If the assumptions are now reasonably met, we can report any model results of interest from `anova()` and/or `summary()`. With an interaction in the model, the fixed effects section of `summary()` is unlikely to be useful.

```
anova(model2)
```

```
                    numDF denDF  F-value p-value
(Intercept)             1    42 905.6619  <.0001
litterspp               3    42  19.8215  <.0001
overstory               1     7  12.5848  0.0094
litterspp:overstory     3    42   4.0325  0.0131
```

```
summary(model2)
```

```
Linear mixed-effects model fit by REML
  Data: dbiomass
       AIC      BIC    logLik
  148.7095 170.9884 -63.35475

Random effects:
 Formula: ~1 | watershed
        (Intercept)
StdDev:   0.2572113

 Formula: ~1 | stand %in% watershed
        (Intercept) Residual
StdDev:   0.3366683 0.559389

Fixed effects:  log(biomass) ~ litterspp * overstory
                               Value Std.Error DF   t-value p-value
(Intercept)                 4.163621 0.2480976 42 16.782193  0.0000
littersppALRU               0.415810 0.2796945 42  1.486656  0.1446
littersppPSME               0.824391 0.2796945 42  2.947469  0.0052
littersppTSHE               0.785050 0.2796945 42  2.806813  0.0076
overstoryRA                -1.073411 0.3264435  7 -3.288197  0.0133
littersppALRU:overstoryRA -0.309694 0.3955477 42 -0.782949  0.4380
littersppPSME:overstoryRA  0.950903 0.3955477 42  2.404017  0.0207
```

```
littersppTSHE:overstoryRA  0.546995 0.3955477 42  1.382881  0.1740
 Correlation:
                          (Intr) ltALRU ltPSME ltTSHE ovrsRA lALRU: lPSME:
littersppALRU             -0.564
littersppPSME             -0.564  0.500
littersppTSHE             -0.564  0.500  0.500
overstoryRA               -0.658  0.428  0.428  0.428
littersppALRU:overstoryRA  0.399 -0.707 -0.354 -0.354 -0.606
littersppPSME:overstoryRA  0.399 -0.354 -0.707 -0.354 -0.606  0.500
littersppTSHE:overstoryRA  0.399 -0.354 -0.354 -0.707 -0.606  0.500  0.500

Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-1.5051788 -0.5593704 -0.1786940  0.4870866  2.1116289

Number of Observations: 64
Number of Groups:
           watershed stand %in% watershed
                   8                  16
```

Estimating group differences

This study was designed to answer two specific questions, listed below.

1. What is the difference in total microbial biomass in decomposing litter under a Douglas-fir overstory compared to under a red alder overstory for each litter type?
2. What is the difference in total microbial biomass in decomposing conifer litter under Douglas-fir overstory compared to the total microbial biomass in decomposing conifer litter under the red alder overstory?

Using emmeans for pairwise comparisons within groups

Question 1 can be answered by comparing the Douglas-fir and red alder overstories within each litter type. The **emmeans** package has options for pairwise comparisons of levels of one factor within another factor, which is done using the `|`.

```
emmeans(model2, pairwise ~ overstory|litterspp)
```

```
$emmeans
litterspp = ACMA:
 overstory emmean    SE df lower.CL upper.CL
 DF          4.16 0.248  7     3.58     4.75
 RA          3.09 0.248  7     2.50     3.68

litterspp = ALRU:
 overstory emmean    SE df lower.CL upper.CL
 DF          4.58 0.248  7     3.99     5.17
 RA          3.20 0.248  7     2.61     3.78

litterspp = PSME:
 overstory emmean    SE df lower.CL upper.CL
 DF          4.99 0.248  7     4.40     5.57
 RA          4.87 0.248  7     4.28     5.45

litterspp = TSHE:
 overstory emmean    SE df lower.CL upper.CL
 DF          4.95 0.248  7     4.36     5.54
 RA          4.42 0.248  7     3.84     5.01

Degrees-of-freedom method: containment
Results are given on the log (not the response) scale.
Confidence level used: 0.95
```

```
$contrasts
litterspp = ACMA:
 contrast estimate    SE df t.ratio p.value
 DF - RA     1.073 0.326  7   3.288  0.0133

litterspp = ALRU:
 contrast estimate    SE df t.ratio p.value
 DF - RA     1.383 0.326  7   4.237  0.0039

litterspp = PSME:
 contrast estimate    SE df t.ratio p.value
 DF - RA     0.123 0.326  7   0.375  0.7186

litterspp = TSHE:
 contrast estimate    SE df t.ratio p.value
 DF - RA     0.526 0.326  7   1.613  0.1509

Degrees-of-freedom method: containment
Results are given on the log (not the response) scale.
```

Back-transforming estimates and CI limits

We got a message after running the code above that

> Results are given on the log (not the response) scale.

We got this message because **emmeans** knew we did a log transformation. This is the reason I used `log(biomass)` as the response instead of using `lbio`.

We are going to want to make inference on the original scale. We can always do this manually using the inverse of the `log()` function, which is `exp()`, on all estimates and CI limits. However, the `emmeans()` function has built-in tools to help us with this via the `type` argument.

Using `type = "response"` gives us back-transformed estimates.

```
emmeans(model2, pairwise ~ overstory|litterspp,
        type = "response")
```

```
$emmeans
litterspp = ACMA:
 overstory response    SE df lower.CL upper.CL
 DF            64.3 15.95  7     35.8    115.6
 RA            22.0  5.45  7     12.2     39.5

litterspp = ALRU:
 overstory response    SE df lower.CL upper.CL
 DF            97.5 24.18  7     54.2    175.2
 RA            24.4  6.06  7     13.6     43.9

litterspp = PSME:
 overstory response    SE df lower.CL upper.CL
 DF           146.6 36.38  7     81.6    263.7
 RA           129.7 32.19  7     72.2    233.3

litterspp = TSHE:
 overstory response    SE df lower.CL upper.CL
 DF           141.0 34.98  7     78.4    253.5
 RA            83.3 20.66  7     46.3    149.7

Degrees-of-freedom method: containment
```

```
Confidence level used: 0.95
Intervals are back-transformed from the log scale

$contrasts
litterspp = ACMA:
 contrast ratio    SE df null t.ratio p.value
 DF / RA   2.93 0.955  7    1   3.288  0.0133

litterspp = ALRU:
 contrast ratio    SE df null t.ratio p.value
 DF / RA   3.99 1.302  7    1   4.237  0.0039

litterspp = PSME:
 contrast ratio    SE df null t.ratio p.value
 DF / RA   1.13 0.369  7    1   0.375  0.7186

litterspp = TSHE:
 contrast ratio    SE df null t.ratio p.value
 DF / RA   1.69 0.553  7    1   1.613  0.1509

Degrees-of-freedom method: containment
Tests are performed on the log scale
```

I'm happy with the direction of these comparisons, which is DF overstory divided by RA overstory. If I wanted the comparisons to be made in the other direction, though, I could use `revpairwise`.

Note that `emmeans()` returns standard errors on the original scale. These were calculated via the delta method, since SE cannot be back-transformed. However, the SE is not an appropriate measurement for asymmetric distributions and you should report CI limits, not SE.

We can pull out the `contrasts` from `emmeans()` to use as results. This week I'm going to save these as a separate object, named `comp_over`, for later use.

```
comp_over = emmeans(model2, pairwise ~ overstory|litterspp,
                type = "response")$contrast
comp_over
```

```
litterspp = ACMA:
 contrast ratio    SE df null t.ratio p.value
 DF / RA   2.93 0.955  7    1   3.288  0.0133

litterspp = ALRU:
 contrast ratio    SE df null t.ratio p.value
 DF / RA   3.99 1.302  7    1   4.237  0.0039

litterspp = PSME:
 contrast ratio    SE df null t.ratio p.value
 DF / RA   1.13 0.369  7    1   0.375  0.7186

litterspp = TSHE:
 contrast ratio    SE df null t.ratio p.value
 DF / RA   1.69 0.553  7    1   1.613  0.1509

Degrees-of-freedom method: containment
Tests are performed on the log scale
```

Then I'll go through the standard process of pulling out the information I want, including CI, and putting the results in a data.frame. This should look pretty familiar from past weeks.

```
( comp_over_df = summary(comp_over, infer = TRUE ) )
```

```
litterspp = ACMA:
```

```
 contrast ratio    SE df lower.CL upper.CL null t.ratio p.value
 DF / RA   2.93 0.955  7    1.352     6.33    1   3.288  0.0133

litterspp = ALRU:
 contrast ratio    SE df lower.CL upper.CL null t.ratio p.value
 DF / RA   3.99 1.302  7    1.843     8.63    1   4.237  0.0039

litterspp = PSME:
 contrast ratio    SE df lower.CL upper.CL null t.ratio p.value
 DF / RA   1.13 0.369  7    0.522     2.45    1   0.375  0.7186

litterspp = TSHE:
 contrast ratio    SE df lower.CL upper.CL null t.ratio p.value
 DF / RA   1.69 0.553  7    0.782     3.66    1   1.613  0.1509

Degrees-of-freedom method: containment
Confidence level used: 0.95
Intervals are back-transformed from the log scale
Tests are performed on the log scale
```

Degrees of freedom in emmeans for lme objects

We're not talking a lot about degrees of freedom in this class, but I wanted to at least mention it. The **emmeans** package uses the *containment* method by default for **lme** objects, which is an old standard for perfectly balanced data from experiments. In current versions of the package there are also Satterthwaite degrees of freedom available, which are likely an appropriate alternative in many situations where the containment method is going to underestimate degrees of freedom. You can get these via the `mode` argument. You can see more information in the **emmeans** "Models supported" vignette here.

Custom contrasts to average over some groups in emmeans

The second question is asking a question about conifer litter under the different overstory species We'll need to average over the two conifer litter types to calculate the "conifer" litter estimate for each overstory species We can do this using custom contrasts like we used last week.

The first step with custom contrasts is to get the means for each factor combinations using `emmeans()`. Make sure to do this on the model scale (i.e., don't use type = "response").

```
( emm_mod2 = emmeans(model2, ~litterspp*overstory) )
```

```
 litterspp overstory emmean    SE df lower.CL upper.CL
 ACMA      DF          4.16 0.248  7     3.58     4.75
 ALRU      DF          4.58 0.248  7     3.99     5.17
 PSME      DF          4.99 0.248  7     4.40     5.57
 TSHE      DF          4.95 0.248  7     4.36     5.54
 ACMA      RA          3.09 0.248  7     2.50     3.68
 ALRU      RA          3.20 0.248  7     2.61     3.78
 PSME      RA          4.87 0.248  7     4.28     5.45
 TSHE      RA          4.42 0.248  7     3.84     5.01

Degrees-of-freedom method: containment
Results are given on the log (not the response) scale.
Confidence level used: 0.95
```

Then we pull out the means of interest using vectors of 0 and 1. These vectors need to have a total length that matches the number of means in `emm_mod2`, which in this case is 8 (2 overstory species x 4 litter types).

For example, the DF overstory - PSME litter is the third value in `emm_mod2` so we use a 1 in third position of our 8 value vector.

```
df_psme = c(0, 0, 1, 0, 0, 0, 0, 0)
```

We pull the other three means of interest (for both conifer litters, PSME and TSHE, for the two overstory species) the same way.

```
df_tshe = c(0, 0, 0, 1, 0, 0, 0, 0)
ra_psme = c(0, 0, 0, 0, 0, 0, 1, 0)
ra_tshe = c(0, 0, 0, 0, 0, 0, 0, 1)
```

To get a "conifer" litter effect we'll need to average over the two conifer litter types within each overstory species. We do this by literally averaging our two vectors that represent the means together.

```
dfconif = (df_psme + df_tshe)/2
raconif = (ra_psme + ra_tshe)/2
```

The question is about a difference in overstory species for conifer litter, so we do this comparisons via subtraction in `contrast()`.

```
contrast(emm_mod2, method = list(dfconif_raconif = dfconif - raconif) )
```

```
 contrast        estimate   SE df t.ratio p.value
 dfconif_raconif    0.324 0.26  7   1.249  0.2517

Degrees-of-freedom method: containment
Results are given on the log (not the response) scale.
```

We again need to get this on the original scale via `type = "response"`. I'll save the contrast object with a name like I did above.

```
( comp_conif = contrast(emm_mod2,
                        method = list(dfconif_raconif = dfconif - raconif),
                        type = "response" ) )
```

```
 contrast        ratio    SE df null t.ratio p.value
 dfconif_raconif  1.38 0.359  7    1   1.249  0.2517

Degrees-of-freedom method: containment
Tests are performed on the log scale
```

And then this can be put into a data.frame, with CI, for reporting.

```
( comp_conif_df = summary(comp_conif, infer = c(TRUE, FALSE) ) )
```

```
 contrast        ratio    SE df lower.CL upper.CL
 dfconif_raconif  1.38 0.359  7    0.749     2.56

Degrees-of-freedom method: containment
Confidence level used: 0.95
Intervals are back-transformed from the log scale
```

Overall multiple comparisons adjustment

In this example we did two sets of comparisons. The way I approached things meant I didn't do any adjustments for multiple comparisons, which is likely appropriate for such a noisy, exploratory study.

However, there are times that we would want to adjust the entire *family* of comparisons. The **emmeans** package has a function, `rbind.emmGrid`, that allows us to do overall multiple comparisons adjustments for all the comparisons we've made. This is specific for **emmGrid** objects, so be sure you go to that help page and not the general `rbind()` help page.

By default `rbind()` does a Bonferroni adjustment, which is extremely conservative. It's probably too conservative for the vast majority of cases.

Here is an example, applying a Bonferroni adjustment to all 5 of comparisons we did.

```
rbind(comp_over, comp_conif)
```

```
 litterspp contrast        ratio    SE df null t.ratio p.value
 ACMA      DF / RA          2.93 0.955  7    1   3.288  0.0667
```

```
 ALRU      DF / RA          3.99 1.302  7    1   4.237  0.0193
 PSME      DF / RA          1.13 0.369  7    1   0.375  1.0000
 TSHE      DF / RA          1.69 0.553  7    1   1.613  0.7544
 .         dfconif_raconif  1.38 0.359  7    1   1.249  1.0000

Degrees-of-freedom method: containment
P value adjustment: bonferroni method for 5 tests
Tests are performed on the log scale
```

We can change the type of adjustment using the `adjust` argument.

We can use no adjustment, which is actually what we already have.

```
rbind(comp_over, comp_conif, adjust = "none")
```

```
 litterspp contrast        ratio    SE df null t.ratio p.value
 ACMA      DF / RA          2.93 0.955  7    1   3.288  0.0133
 ALRU      DF / RA          3.99 1.302  7    1   4.237  0.0039
 PSME      DF / RA          1.13 0.369  7    1   0.375  0.7186
 TSHE      DF / RA          1.69 0.553  7    1   1.613  0.1509
 .         dfconif_raconif  1.38 0.359  7    1   1.249  0.2517

Degrees-of-freedom method: containment
Tests are performed on the log scale
```

Or one of the other adjustment options, like a multivariate-t adjustment.

```
rbind(comp_over, comp_conif, adjust = "mvt")
```

```
 litterspp contrast        ratio    SE df null t.ratio p.value
 ACMA      DF / RA          2.93 0.955  7    1   3.288  0.0471
 ALRU      DF / RA          3.99 1.302  7    1   4.237  0.0141
 PSME      DF / RA          1.13 0.369  7    1   0.375  0.9900
 TSHE      DF / RA          1.69 0.553  7    1   1.613  0.4250
 .         dfconif_raconif  1.38 0.359  7    1   1.249  0.6254

Degrees-of-freedom method: containment
P value adjustment: mvt method for 5 tests
Tests are performed on the log scale
```

If we wanted to adjust just the first four comparisons we did on their own we could also do this via `rbind()`.

```
rbind(comp_over, adjust = "mvt")
```

```
 litterspp contrast ratio    SE df null t.ratio p.value
 ACMA      DF / RA   2.93 0.955  7    1   3.288  0.0446
 ALRU      DF / RA   3.99 1.302  7    1   4.237  0.0134
 PSME      DF / RA   1.13 0.369  7    1   0.375  0.9896
 TSHE      DF / RA   1.69 0.553  7    1   1.613  0.4159

Degrees-of-freedom method: containment
P value adjustment: mvt method for 4 tests
Tests are performed on the log scale
```

Estimating differences in main effects with emmeans

Researchers were not expressly interested in overall differences among the two factors but instead were expecting combined effects (i.e., an interaction). However, if there were questions specific to main effects we can compare across levels in one factor in **emmeans**. In this case we list only the factor of interest.

Here is the difference in mean log total biomass between overstory species. Note the message that **emmeans** gives us, making sure we recognize we are doing main effects by averaging over the effect of the other factor in the model.

```
emmeans(model2, pairwise ~ overstory)
```

```
NOTE: Results may be misleading due to involvement in interactions
```

```
$emmeans
 overstory emmean    SE df lower.CL upper.CL
 DF          4.67 0.179  7     4.25     5.09
 RA          3.89 0.179  7     3.47     4.32

Results are averaged over the levels of: litterspp
Degrees-of-freedom method: containment
Results are given on the log (not the response) scale.
Confidence level used: 0.95

$contrasts
 contrast estimate    SE df t.ratio p.value
 DF - RA     0.776 0.219  7   3.548  0.0094

Results are averaged over the levels of: litterspp
Degrees-of-freedom method: containment
Results are given on the log (not the response) scale.
```

Wrapping up the analysis

Graphic

Once we have our results, we can make tables and figures to include in our write-up. Here I will work on a a graphic and a table of results for the comparisons that answer question 1 and then make a table of summary statistics. I will leave any further graphics to the "Bonus graphics" portion of the lab.

First I'll add better names to describe each comparison, including both factors used in the comparison. I use `/` to indicate division, although I could have used the word "over" instead. I'm using the natural ordering here, since it makes sense to me to use the alphabetical order of the litter types.

```
# Make column of comparisons that include litterspp and add to dataset
comp_over_df
```

```
litterspp = ACMA:
 contrast ratio    SE df lower.CL upper.CL null t.ratio p.value
 DF / RA   2.93 0.955  7    1.352     6.33    1   3.288  0.0133

litterspp = ALRU:
 contrast ratio    SE df lower.CL upper.CL null t.ratio p.value
 DF / RA   3.99 1.302  7    1.843     8.63    1   4.237  0.0039

litterspp = PSME:
 contrast ratio    SE df lower.CL upper.CL null t.ratio p.value
 DF / RA   1.13 0.369  7    0.522     2.45    1   0.375  0.7186

litterspp = TSHE:
 contrast ratio    SE df lower.CL upper.CL null t.ratio p.value
 DF / RA   1.69 0.553  7    0.782     3.66    1   1.613  0.1509

Degrees-of-freedom method: containment
Confidence level used: 0.95
Intervals are back-transformed from the log scale
Tests are performed on the log scale
```

```
comp_over_df$contrast = c("DF / RA overstory, ACMA litter",
                          "DF / RA overstory, ALRU litter",
                          "DF / RA overstory, PSME litter",
```

```
                    "DF / RA overstory, TSHE litter")
```

```
( g1 = ggplot(comp_over_df, aes(x = ratio, y = contrast) ) + # Put response on x axis
        geom_errorbar(width = .2, lwd=.75,
                      aes(xmin = lower.CL, xmax = upper.CL) ) + # Add error bar
        geom_point(size = 2.5) + # Add points
        labs(x = "Ratio of Total Microbial Biomass (mg/g)",
            y = NULL) + # Label axes
        geom_vline(xintercept = 1, lty = 2) + # add a horiz line when ratio at 1
        geom_rect(alpha = .0625, ymin = 0, ymax = 5, xmin = .5, xmax = 2) +
    # add rect for practical difference
        scale_x_continuous(breaks = seq(0, 9, by = 1) ) + # Add more breaks on x
        theme_bw(base_size = 15) + # Make black and white,
                                   # increase base text size
        theme(axis.ticks.y = element_blank(),
             axis.text.y = element_blank(), # Remove axis labels and tick marks
             panel.grid.major.y = element_blank(), # Remove y gridlines
             panel.grid.minor = element_blank() ) +  # Remove minor gridlines
        geom_text(aes(x = 1.5, label = contrast), # Line up labels at x = 1.5
                nudge_y = 0.2, # Nudge labels above lines
                size = 4, hjust = 0) + # Place text label
        scale_y_discrete(limits = rev( comp_over_df$contrast) ) ) # Switch order of y axis
```

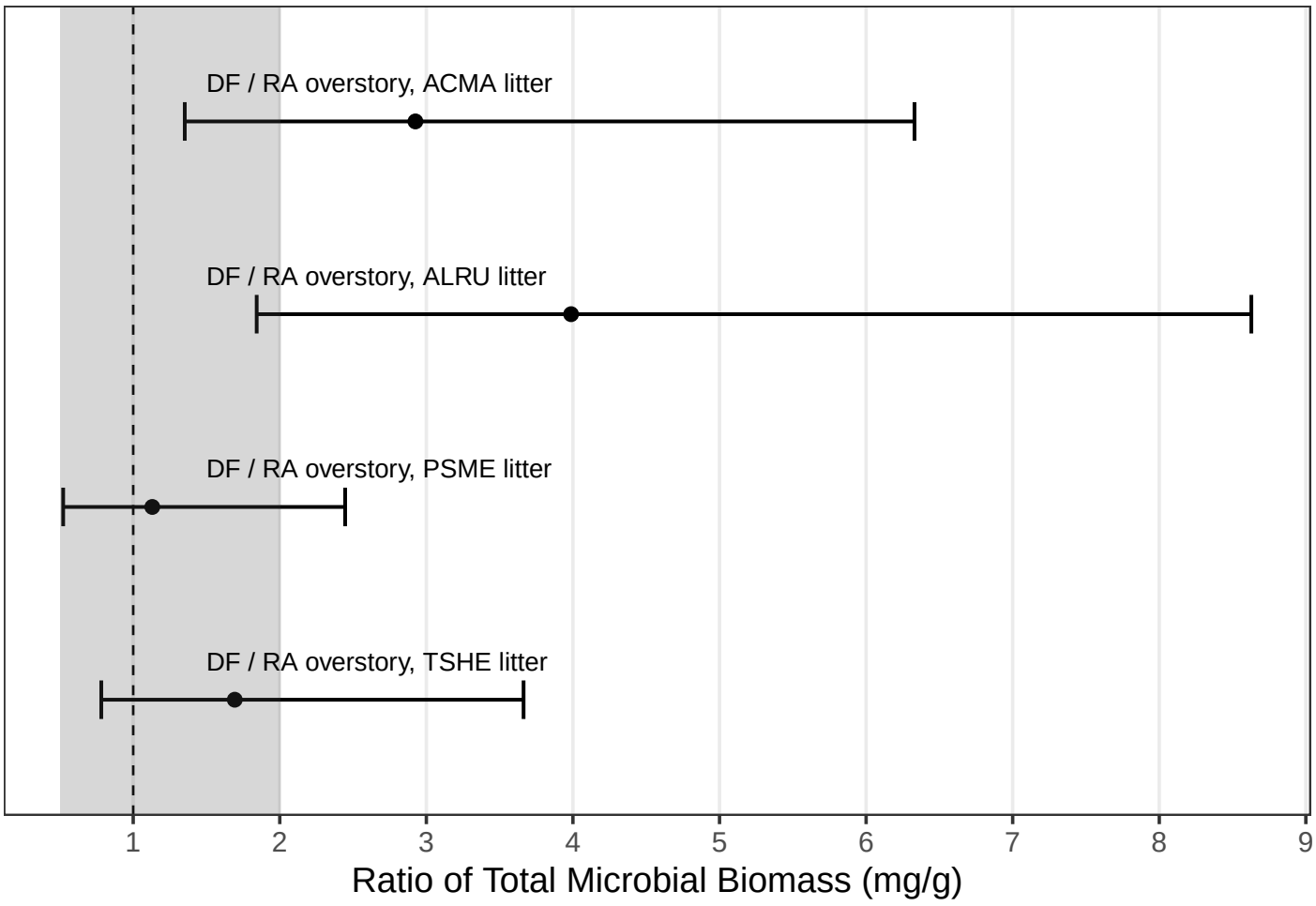


Table of results

The code for making the table of results is fairly complicated. I've annotated the code but will not go into things in detail. This gives you an idea of the sort of data manipulation code you would need if you want to do this work in R and not some

other program.

```
# Make a table for estimates of
    # ratios of medians and confidence intervals
    # to answer question 1
# Start by rounding everything to 1 digit
comp_over_tab = comp_over_df %>%
     select(contrast, ratio, lower.CL, upper.CL) %>%
     mutate(across(.cols = where(is.numeric), .fns= round, digits = 1) )

# I am going to use paste to make a single column for the confidence intervals
comp_over_tab = mutate(comp_over_tab,
                       ci = paste(lower.CL, upper.CL, sep = ", ") )

# I need to make the column names looks nicer
colnames(comp_over_tab) = c("Comparison", "Ratio of medians",
                            "Lower", "Upper", "95% CI")

# This is a table of results I could put in my write up
select(comp_over_tab, -Lower, -Upper)
```

Comparison	Ratio of medians	95% CI
DF / RA overstory, ACMA litter	2.9	1.4, 6.3
DF / RA overstory, ALRU litter	4.0	1.8, 8.6
DF / RA overstory, PSME litter	1.1	0.5, 2.4
DF / RA overstory, TSHE litter	1.7	0.8, 3.7

Summary table

Below I create a summary table of descriptive statistics. Notice that I summarize the data using the median as the measure of center and the interquartile range as a measure of spread. The median and interquartile range are generally a better way to describe skewed data like this compared to the mean and standard deviation. Also, our results are about ratios of medians, so showing the means of the observed data doesn't make a lot of sense. Do note that the observed medians are not identical to the estimated medians after back-transformation, as discussed in reading 4.2.

```
# A table of summary statistics by combined factor group, if desired
( sumtable = dbiomass %>%
        group_by("Overstory" = overstory,
                 "Litter" = litterspp) %>%
        summarise(n = n(),
                  Median = round( median(biomass) ),
                  "1st quartile" = round( quantile(biomass, .25) ),
                  "3rd quartile" = round( quantile(biomass, .75) ),
                  .groups = "drop") )
```

Overstory	Litter	n	Median	1st quartile	3rd quartile
DF	ACMA	8	54	48	70
DF	ALRU	8	81	54	152
DF	PSME	8	112	96	272
DF	TSHE	8	136	106	224
RA	ACMA	8	26	15	36
RA	ALRU	8	28	16	42
RA	PSME	8	132	86	196
RA	TSHE	8	99	45	144