Getting started with emmeans

March 25, 2019 · @aosmith16 · View source
analysis, teaching, emmeans

This post was last updated on 2021-11-04.

Package **emmeans** (formerly known as **lsmeans**) is enormously useful for folks wanting to do post hoc comparisons among groups after fitting a model. It has a very thorough set of vignettes (see the vignette topics here), is very flexible with a ton of options, and works out of the box with a lot of different model objects (and can be extended to others 👍).

I've been consistently recommending **emmeans** to students fitting models in R. However, often times students struggle a bit to get started using the package, possibly due to the sheer amount of flexibility and information in the vignettes.

I've put together some basic examples for using **emmeans**, meant to be a complement to the vignettes. Specifically this post will demonstrate a few of the built-in options for some standard post hoc comparisons; I will write a separate post about custom comparisons in **emmeans**.

Disclaimer: This post is about using a package in R and so unfortunately does not focus on appropriate statistical practice for model fitting and post hoc comparisons.

Table of Contents

R packages

I will load **magrittr** for the pipe in addition to **emmeans**.

```
library(emmeans) # v. 1.7.0
library(magrittr) # v. 2.0.1
```

The dataset and model

I've made a small dataset to use in this example.

The response variable is `resp`, which comes from the log-normal distribution, and the two crossed factors of interest are `f1` and `f2`. Each factor has two levels: a control called `c` as well as a second non-control level.

```
dat = data.frame(resp = c(1.6,0.3,3,0.1,3.2,0.2,0.4,0.4,2.8,
                          0.7,3.8,3,0.3,14.3,1.2,0.5,1.1,4.4,0.4,8.4),
                 f1 = factor(c("a","a","a","a","a",
                               "a","a","a","a","a","c","c","c","c","c",
                               "c","c","c","c","c")),
                 f2 = factor(c("1","c","1","c","1",
                               "c","1","c","1","c","1","c","1","c","1",
                               "c","1","c","1","c")))

str(dat)
```

```
# 'data.frame': 20 obs. of  3 variables:
#  $ resp: num  1.6 0.3 3 0.1 3.2 0.2 0.4 0.4 2.8 0.7 ...
#  $ f1  : Factor w/ 2 levels "a","c": 1 1 1 1 1 1 1 1 1 1 ...
#  $ f2  : Factor w/ 2 levels "1","c": 1 2 1 2 1 2 1 2 1 2 ...
```

The model I will use is a linear model with a log-transformed response variable and the two factors and their interaction as explanatory variables. This is the "true" model since I created these data so I'm skipping all model checks (which I would not do in a real analysis).

Note I use `log(resp)` in the model rather than creating a new log-transformed variable. This will allow me to demonstrate one of the convenient options available in `emmeans()` later.

Built in comparisons with emmeans()

The **emmeans** package has helper functions for commonly used post hoc comparisons (aka *contrasts*). For example, we can do pairwise comparisons via `pairwise` or `revpairwise`, treatment vs control comparisons via `trt.vs.ctrl` or `trt.vs.ctrlk`, and even consecutive comparisons via `consec`.

The available built-in functions for doing comparisons are listed in the documentation for `?"contrast-methods"`.

All pairwise comparisons

One way to use `emmeans()` is via formula coding for the comparisons. The formula is defined in the `specs` argument.

In my first example I do all pairwise comparisons for all combinations of `f1` and `f2`. The built-in function `pairwise` is put on the left-hand side of the formula of the `specs` argument. The factors with levels to compare among are on the right-hand side. Since I'm doing all pairwise comparisons, the combination of `f1` and `f2` are put in the formula.

The model object is passed to the first argument in `emmeans()`, `object`.

```
emm1 = emmeans(fit1, specs = pairwise ~ f1:f2)
```

Using the formula in this way returns an object with two parts. The first part, called `emmeans`, is the estimated marginal means along with the standard errors and confidence intervals. We can pull these out with dollar sign notation, which I demonstrate below.

These results are all on the *model* scale, so in this case these are estimated mean log response for each `f1` and `f2` combination. Note the message that `emmeans()` gives us about results being on the log scale in the output. It knows the model is on the log scale because I used `log(resp)` as the response variable.

```
emm1$emmeans
```

```
#  c  1  -0.102 0.445 16   -1.045    0.842
#  a  c  -1.278 0.445 16   -2.221   -0.334
#  c  c   1.335 0.445 16    0.392    2.279
#
# Results are given on the log (not the response) scale.
# Confidence level used: 0.95
```

The second part of the output, called `contrasts`, contains the comparisons of interest. It is this section that we are generally most interested in when answering a question about differences among groups. You can see which comparison is which via the `contrast` column.

These results are also on the model scale (and we get the same message in this section), and we'll want to put them on the original scale.

The comparisons are accompanied by statistical tests of the null hypothesis of "no difference", but lack confidence interval (CI) limits by default. We'll need to get these.

The `emmeans()` package automatically adjusts for multiple comparisons. Since we did all pairwise comparisons the package used a Tukey adjustment. The type of adjustment can be changed.

```
emm1$contrasts
```

```
#  contrast  estimate    SE df t.ratio p.value
#  a 1 - c 1    0.671 0.629 16   1.065  0.7146
#  a 1 - a c    1.847 0.629 16   2.934  0.0434
#  a 1 - c c   -0.766 0.629 16  -1.217  0.6253
#  c 1 - a c    1.176 0.629 16   1.869  0.2795
#  c 1 - c c   -1.437 0.629 16  -2.283  0.1438
#  a c - c c   -2.613 0.629 16  -4.152  0.0038
#
# Results are given on the log (not the response) scale.
# P value adjustment: tukey method for comparing a family of 4 estimates
```

Back-transforming results

Since I used a log transformation I can express the results as multiplicative differences in medians on the original (data) scale.

use the `type` argument for this. Using `type = "response"` will return results on the original scale. This works when the transformation is explicit in the model (e.g., `log(resp)`) and works similarly for link functions in generalized linear models.

You'll see the message changes in the output once I do this, indicating things were back-transformed from the model scale. We also are reminded that the tests were done on the model scale.

In the `contrast` column in the `contrasts` section we can see the expression of the comparisons has changed from additive comparisons (via subtraction) shown above to multiplicative comparisons (via division).

```
emmeans(fit1, specs = pairwise ~ f1:f2, type = "response")
```

```
# $emmeans
#  f1 f2 response    SE df lower.CL upper.CL
#  a  1     1.767 0.786 16    0.688    4.538
#  c  1     0.903 0.402 16    0.352    2.321
#  a  c     0.279 0.124 16    0.108    0.716
#  c  c     3.800 1.691 16    1.479    9.763
#
# Confidence level used: 0.95
# Intervals are back-transformed from the log scale
#
# $contrasts
#  contrast   ratio     SE df null t.ratio p.value
#  a 1 / c 1 1.9553 1.2306 16    1   1.065  0.7146
#  a 1 / a c 6.3396 3.9900 16    1   2.934  0.0434
#  a 1 / c c 0.4648 0.2926 16    1  -1.217  0.6253
#  c 1 / a c 3.2422 2.0406 16    1   1.869  0.2795
#  c 1 / c c 0.2377 0.1496 16    1  -2.283  0.1438
#  a c / c c 0.0733 0.0461 16    1  -4.152  0.0038
#
# P value adjustment: tukey method for comparing a family of 4 estimates
# Tests are performed on the log scale
```

Changing the multiple comparisons adjustment

The `adjust` argument can be used to change the type of multiple comparisons adjustment. All available options are listed and described in the documentation for `summary.emmGrid`

One option is to skip multiple comparisons adjustments all together, using `adjust = "none"`. If we use this the message about multiple comparisons disappears (since we didn't use one).

```
emm1.1 = emmeans(fit1, specs = pairwise ~ f1:f2, type = "response", adjust = "none")
emm1.1
```

```
# $emmeans
#  f1 f2 response    SE df lower.CL upper.CL
#  a  1     1.767 0.786 16    0.688    4.538
#  c  1     0.903 0.402 16    0.352    2.321
#  a  c     0.279 0.124 16    0.108    0.716
#  c  c     3.800 1.691 16    1.479    9.763
#
# Confidence level used: 0.95
# Intervals are back-transformed from the log scale
#
# $contrasts
#  contrast   ratio     SE df null t.ratio p.value
#  a 1 / c 1 1.9553 1.2306 16    1   1.065  0.3025
#  a 1 / a c 6.3396 3.9900 16    1   2.934  0.0097
#  a 1 / c c 0.4648 0.2926 16    1  -1.217  0.2412
#  c 1 / a c 3.2422 2.0406 16    1   1.869  0.0801
#  c 1 / c c 0.2377 0.1496 16    1  -2.283  0.0365
#  a c / c c 0.0733 0.0461 16    1  -4.152  0.0008
#
# Tests are performed on the log scale
```

Confidence intervals for comparisons

We will almost invariably want to report confidence intervals for any comparisons of interest. We need a separate function to get these. Here is an example using the `confint()` function with the default 95% CI (the confidence level can be changed, see `?confint.emmGrid`). I use the pipe to pass the `contrasts` into the `confint()` function.

```
emm1.1$contrasts %>%
     confint()
```

```
#  contrast   ratio     SE df lower.CL upper.CL
#  a 1 / c 1 1.9553 1.2306 16   0.5150    7.424
```

```
#  c 1 / a c 3.2422 2.0406 16   0.8539   12.311
#  c 1 / c c 0.2377 0.1496 16   0.0626    0.903
#  a c / c c 0.0733 0.0461 16   0.0193    0.278
#
# Confidence level used: 0.95
# Intervals are back-transformed from the log scale
```

The `confint()` function returns confidence intervals but gets rid of the statistical tests. Some people will want to also report the test statistics and p-values. In this case, we can use `summary()` instead of `confint()`, with `infer = TRUE`.

```
emm1.1$contrasts %>%
     summary(infer = TRUE)
```

```
#  contrast   ratio     SE df lower.CL upper.CL null t.ratio p.value
#  a 1 / c 1 1.9553 1.2306 16   0.5150    7.424    1   1.065  0.3025
#  a 1 / a c 6.3396 3.9900 16   1.6696   24.072    1   2.934  0.0097
#  a 1 / c c 0.4648 0.2926 16   0.1224    1.765    1  -1.217  0.2412
#  c 1 / a c 3.2422 2.0406 16   0.8539   12.311    1   1.869  0.0801
#  c 1 / c c 0.2377 0.1496 16   0.0626    0.903    1  -2.283  0.0365
#  a c / c c 0.0733 0.0461 16   0.0193    0.278    1  -4.152  0.0008
#
# Confidence level used: 0.95
# Intervals are back-transformed from the log scale
# Tests are performed on the log scale
```

Putting results in a data.frame

One of the really nice things about `emmeans()` is that it makes it easy to get the results into a nice format for making tables or graphics of results. This is because the results are converted to a data.frame with `confint()` or `summary()`.

If needed, the estimated marginal means can also be put into a data.frame. In this case we can use `as.data.frame()` to convert the `emmeans` to a data.frame for plotting or putting into a table of results. We can also use `as.data.frame()` directly on the contrasts above if we don't need `confint()` or `summary()` (not shown).

```
emm1.1$emmeans %>%
     as.data.frame()
```

```
# 1  a  1 1.7665354 0.7861763 16 0.6876870 4.557879
# 2  c  1 0.9034576 0.4020739 16 0.3517035 2.320806
# 3  a  c 0.2786518 0.1240109 16 0.1084753 0.715802
# 4  c  c 3.8004222 1.6913362 16 1.4794517 9.762542
```

Within group comparisons

While we *can* do all pairwise comparisons, there are certainly plenty of situations where the research question dictates that we only want a specific set of comparisons. A common example of this is when we want to compare the levels of one factor within the levels of another. Here I'll show comparisons among levels of `f1` for each level of `f2`.

The only thing that changes is the right-hand side of the `specs` formula. The code `f1|f2` translates to "compare levels of `f1` within each level of `f2`".

```
emm2 = emmeans(fit1, specs = pairwise ~ f1|f2, type = "response")
emm2
```

```
# $emmeans
# f2 = 1:
#  f1 response    SE df lower.CL upper.CL
#  a     1.767 0.786 16    0.688    4.538
#  c     0.903 0.402 16    0.352    2.321
#
# f2 = c:
#  f1 response    SE df lower.CL upper.CL
#  a     0.279 0.124 16    0.108    0.716
#  c     3.800 1.691 16    1.479    9.763
#
# Confidence level used: 0.95
# Intervals are back-transformed from the log scale
#
# $contrasts
# f2 = 1:
#  contrast  ratio     SE df null t.ratio p.value
#  a / c    1.9553 1.2306 16    1   1.065  0.3025
#
# f2 = c:
#  contrast  ratio     SE df null t.ratio p.value
#  a / c    0.0733 0.0461 16    1  -4.152  0.0008
#
# Tests are performed on the log scale
```

comparisons. This is because the package default is to correct for the number of comparisons *within* each group instead of across groups. In this case there is only a single comparison in each group.

If we consider the family of comparisons to be all comparisons regardless of group and want to correct for multiple comparisons, we can do so via `rbind.emmGrid`.

Here is an example of passing `contrasts` to `rbind()` to correct for multiple comparisons. The default adjustment is Bonferroni, which can be much too conservative when the number of comparisons is large. You can control the multiple comparisons procedure via `adjust`.

The results of `rbind()` can also conveniently be used with `summary()`, `confint()`, and/or `as.data.frame()`.

```
emm2$contrasts %>%
     rbind()
```

```
#  f2 contrast  ratio     SE df null t.ratio p.value
#  1  a / c    1.9553 1.2306 16    1   1.065  0.6050
#  c  a / c    0.0733 0.0461 16    1  -4.152  0.0015
#
# P value adjustment: bonferroni method for 2 tests
# Tests are performed on the log scale
```

Main effects comparisons

Even if we have multiple factors in the model, complete with an interaction term, we can still do "overall" comparisons among groups if our research question indicated that main effects were important to estimate.

Doing main effects in the presence of an interaction means we *average over* the levels of the other factor(s). The `emmeans()` function gives both a warning about the interaction and a message indicating which factor was averaged over to remind us of this.

Here is the estimated main effect of `f1`. Since we are only interested in overall comparisons of that factor it is the only factor given on the right-hand side of the `specs` formula.

```
emmeans(fit1, specs = pairwise ~ f1)
```

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


# $emmeans
#  f1 emmean    SE df lower.CL upper.CL
#  a  -0.354 0.315 16  -1.0215    0.313
#  c   0.617 0.315 16  -0.0503    1.284
#
# Results are averaged over the levels of: f2
# Results are given on the log (not the response) scale.
# Confidence level used: 0.95
#
# $contrasts
#  contrast estimate    SE df t.ratio p.value
#  a - c      -0.971 0.445 16  -2.182  0.0443
#
# Results are averaged over the levels of: f2
# Results are given on the log (not the response) scale.
```

Treatment vs control example

The **emmeans** package has built-in helper functions for comparing each group mean to the control mean. If the control group is the in the first row of the `emmeans` section of the output, this set of comparisons can be requested via `trt.vs.ctrl` .

Note the default multiple comparisons adjustment is a Dunnett adjustment.

```
emmeans(fit1, specs = trt.vs.ctrl ~ f1:f2)


# $emmeans
#  f1 f2 emmean    SE df lower.CL upper.CL
#  a  1   0.569 0.445 16   -0.374    1.512
#  c  1  -0.102 0.445 16   -1.045    0.842
#  a  c  -1.278 0.445 16   -2.221   -0.334
#  c  c   1.335 0.445 16    0.392    2.279
#
# Results are given on the log (not the response) scale.
# Confidence level used: 0.95
#
# $contrasts
#  contrast  estimate    SE df t.ratio p.value
#  c 1 - a 1   -0.671 0.629 16  -1.065  0.5857
#  a c - a 1   -1.847 0.629 16  -2.934  0.0262
```

```
# Results are given on the log (not the response) scale.
# P value adjustment: dunnettx method for 3 tests
```

Using `trt.vs.ctrl` means we ended up comparing each group mean to the "a 1" group since it is in the first row. In the example I'm using the control group, "c c", is actually the *last* group listed in the `emmeans` section. When the control group is the last group in `emmeans` we can use `trt.vs.ctrlk` to get the correct set of comparisons.

```
emmeans(fit1, specs = trt.vs.ctrlk ~ f1:f2)
```

```
# $emmeans
#  f1 f2 emmean    SE df lower.CL upper.CL
#  a  1   0.569 0.445 16   -0.374    1.512
#  c  1  -0.102 0.445 16   -1.045    0.842
#  a  c  -1.278 0.445 16   -2.221   -0.334
#  c  c   1.335 0.445 16    0.392    2.279
#
# Results are given on the log (not the response) scale.
# Confidence level used: 0.95
#
# $contrasts
#  contrast  estimate    SE df t.ratio p.value
#  a 1 - c c   -0.766 0.629 16  -1.217  0.4947
#  c 1 - c c   -1.437 0.629 16  -2.283  0.0935
#  a c - c c   -2.613 0.629 16  -4.152  0.0021
#
# Results are given on the log (not the response) scale.
# P value adjustment: dunnettx method for 3 tests
```

That gives us what we want in this case. However, if the control group was some other group, like "c 1", we could use `trt.vs.ctrlk` with the `ref` argument to define which row in the `emmeans` section represents the control group.

The "c 1" group is the second row in the `emmeans` so we can use `ref = 2` to define this group as the control group.

```
emmeans(fit1, specs = trt.vs.ctrlk ~ f1:f2, ref = 2)
```

```
# $emmeans
#  f1 f2 emmean    SE df lower.CL upper.CL
#  a  1   0.569 0.445 16   -0.374    1.512
```

```
#  c  c   1.335 0.445 16    0.392    2.279
#
# Results are given on the log (not the response) scale.
# Confidence level used: 0.95
#
# $contrasts
#  contrast  estimate    SE df t.ratio p.value
#  a 1 - c 1    0.671 0.629 16   1.065  0.5857
#  a c - c 1   -1.176 0.629 16  -1.869  0.1937
#  c c - c 1    1.437 0.629 16   2.283  0.0935
#
# Results are given on the log (not the response) scale.
# P value adjustment: dunnettx method for 3 tests
```

Finally, if we want to reverse the order of subtraction in the treatment vs control comparisons we can use the `reverse` argument.

```
emmeans(fit1, specs = trt.vs.ctrlk ~ f1:f2, ref = 2, reverse = TRUE)
```

```
# $emmeans
#  f1 f2 emmean    SE df lower.CL upper.CL
#  a  1   0.569 0.445 16   -0.374    1.512
#  c  1  -0.102 0.445 16   -1.045    0.842
#  a  c  -1.278 0.445 16   -2.221   -0.334
#  c  c   1.335 0.445 16    0.392    2.279
#
# Results are given on the log (not the response) scale.
# Confidence level used: 0.95
#
# $contrasts
#  contrast  estimate    SE df t.ratio p.value
#  c 1 - a 1   -0.671 0.629 16  -1.065  0.5857
#  c 1 - a c    1.176 0.629 16   1.869  0.1937
#  c 1 - c c   -1.437 0.629 16  -2.283  0.0935
#
# Results are given on the log (not the response) scale.
# P value adjustment: dunnettx method for 3 tests
```

Alternative code for comparisons

The `emmeans()` package offers the option to do comparisons in two steps instead of in one step the way I have been using it so far. I personally find this alternative most useful when

objects, which can be attractive in some situations.

The first step is to use `emmeans()` to calculate the marginal means of interest. We still use the formula in `specs` with the factor(s) of interest on the right-hand side but no longer put anything on the left-hand side of the tilde.

We can still use `type` in `emmeans()` but cannot use `adjust` (since we don't adjust for multiple comparisons until we've actually done comparisons 😉).

```
emm3 = emmeans(fit1, specs = ~ f1:f2, type = "response")
emm3
```

```
#  f1 f2 response    SE df lower.CL upper.CL
#  a  1     1.767 0.786 16    0.688    4.538
#  c  1     0.903 0.402 16    0.352    2.321
#  a  c     0.279 0.124 16    0.108    0.716
#  c  c     3.800 1.691 16    1.479    9.763
#
# Confidence level used: 0.95
# Intervals are back-transformed from the log scale
```

We then get the comparisons we want in a second step using the `contrast()` function. We request the comparisons we want via `method`. When using built-in comparisons like I am here, we give the comparison function name as a string (meaning in quotes). Also see the `pairs()` function, which is for the special case of all pairwise comparisons.

We can use `adjust` in `contrast()` to change the multiple comparisons adjustment.

```
contrast(emm3, method = "pairwise", adjust = "none")
```

```
#  contrast   ratio     SE df null t.ratio p.value
#  a 1 / c 1 1.9553 1.2306 16    1   1.065  0.3025
#  a 1 / a c 6.3396 3.9900 16    1   2.934  0.0097
#  a 1 / c c 0.4648 0.2926 16    1  -1.217  0.2412
#  c 1 / a c 3.2422 2.0406 16    1   1.869  0.0801
#  c 1 / c c 0.2377 0.1496 16    1  -2.283  0.0365
#  a c / c c 0.0733 0.0461 16    1  -4.152  0.0008
#
# Tests are performed on the log scale
```

output we want and put them into a data.frame for plotting/saving. Again, I think the real strength of `contrast()` comes when we want custom comparisons, and I'll demonstrate these in my next post on custom contrasts.

Just the code, please

Here's the code without all the discussion. Copy and paste the code below or you can download an R script of uncommented code from here.

```
library(emmeans) # v. 1.7.0
library(magrittr) # v. 2.0.1

dat = data.frame(resp = c(1.6,0.3,3,0.1,3.2,0.2,0.4,0.4,2.8,
                          0.7,3.8,3,0.3,14.3,1.2,0.5,1.1,4.4,0.4,8.4),
                 f1 = factor(c("a","a","a","a","a",
                               "a","a","a","a","a","c","c","c","c","c",
                               "c","c","c","c","c")),
                 f2 = factor(c("1","c","1","c","1",
                               "c","1","c","1","c","1","c","1","c","1",
                               "c","1","c","1","c")))

str(dat)

fit1 = lm(log(resp) ~ f1 + f2 + f1:f2, data = dat)

emm1 = emmeans(fit1, specs = pairwise ~ f1:f2)

emm1$emmeans
emm1$contrasts

emmeans(fit1, specs = pairwise ~ f1:f2, type = "response")

emm1.1 = emmeans(fit1, specs = pairwise ~ f1:f2, type = "response", adjust = "none")
emm1.1

emm1.1$contrasts %>%
     confint()

emm1.1$contrasts %>%
     summary(infer = TRUE)

emm1.1$emmeans %>%
     as.data.frame()

emm2 = emmeans(fit1, specs = pairwise ~ f1|f2, type = "response")
```

```
emm2$contrasts %>%
  rbind()

emmeans(fit1, specs = pairwise ~ f1)

emmeans(fit1, specs = trt.vs.ctrl ~ f1:f2)

emmeans(fit1, specs = trt.vs.ctrlk ~ f1:f2)

emmeans(fit1, specs = trt.vs.ctrlk ~ f1:f2, ref = 2)

emmeans(fit1, specs = trt.vs.ctrlk ~ f1:f2, ref = 2, reverse = TRUE)

emm3 = emmeans(fit1, specs = ~ f1:f2, type = "response")
emm3

contrast(emm3, method = "pairwise", adjust = "none")
```

analysis teaching emmeans

9 Comments aosmith.rbind.io Disqus' Privacy Policy Login

Favorite 9 Tweet Share Sort by Best

Join the discussion…

LOG IN WITH OR SIGN UP WITH DISQUS ?

Name

CupOfTea • 2 years ago
Wow, this was such a great post! Thanks so much for the time and effort that has gone into it. Been using emmeans for a while now but this has opened up many new possibilities for me. Great job 👍👍👍
2 • Reply • Share ›

lemiha • 3 years ago
Thanks for providing this tutorial. Much appreciated.
2 • Reply • Share ›

Meinhard Ploner • 3 years ago
Very good post about multiple comparisons in R. I used the emmeans package to perform post-hoc tests after a Mixed Effects Model (using package lmer) and it works like a charm. Thx.
2 • Reply • Share ›

KingGiinko • 2 years ago
Great thread and intro. I really enjoyed it
1 • Reply • Share ›

that R spits out for emmeans. my response was on its original scale, I didn't need to transform it in anyway. Why is that I still get the same message?

Edit: On second thought, I answered my own question. I think it might because i tried this for gamma GLM and my link was "log"

1 • Reply • Share ›

Yuhao Zhao • 25 days ago

Thanks for this nice tutorial! I have a quick question on how to interpret the results from results of emmeans() on model only contain categorical variables. Here the categorical variables are land use type, namely farmland, forest, and mix.

My model looks is: model = lmer(Y ~ type)

And I used this: emmeans(model, pairwise ~ type, adjust="tukey")

The results are:
$emmeans
type emmean SE df lower.CL upper.CL
farmland -2.24 0.217 60.6 -2.68 -1.81
forest -2.27 0.149 42.9 -2.57 -1.97
mix -1.76 0.255 61.0 -2.27 -1.25

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

$contrasts

see more

• Reply • Share ›

Yair Barnatan • 2 months ago

Great tutorial. What about variables that interact, how could I specify I want to perform contrast on such interaction? thanks!

• Reply • Share ›

Ariel Muldoon Mod → Yair Barnatan • a month ago

I don't think I understand your question. This example specifically shows emmeans with a model with an interaction, starting by showing "all pairwise" among combinations of the factors but also hits on main effects of one variable and comparisons across one factor within levels of another ("within group comparisons").

1 • Reply • Share ›

Yuhao Zhao → Yair Barnatan • 25 days ago

Maybe you could try this?

https://stackoverflow.com/q...

• Reply • Share ›