Gina Nichols - GAMs

Miranda Tilton

October 30, 2020

Data processing

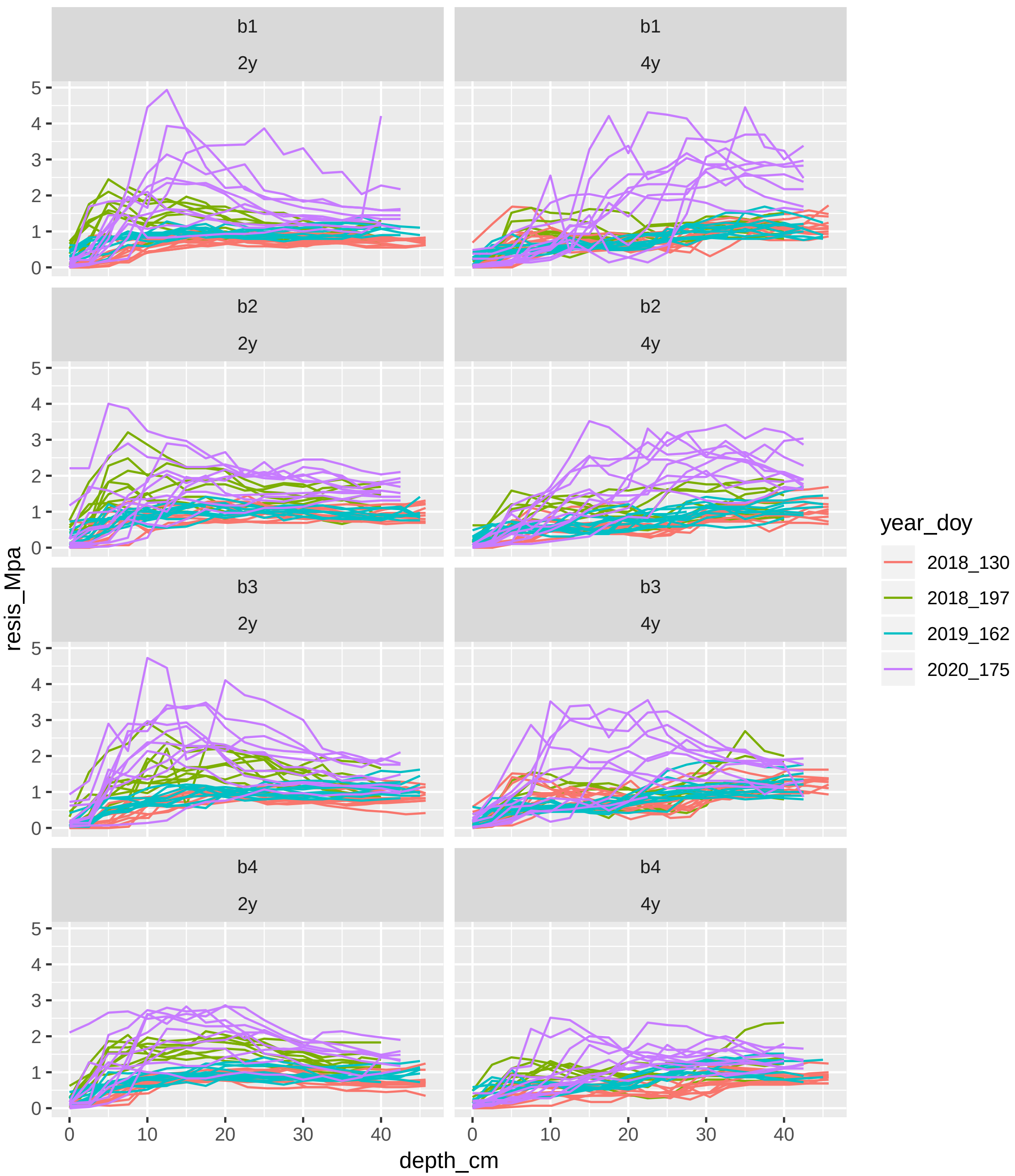
```
# remotes::install_github("vanichols/maRsden")
library(magrittr)
library(maRsden)
library(dplyr)
library(ggplot2)
library(lemon) # chunk option `render = lemon_print` makes tables prettier
```

```
myd <-
  mrs_penetrom %>%
  left_join(mrs_plotkey) %>%
  #filter(year != "2020") %>%
  mutate(resis_Mpa = resis_kpa/1000) %>%
  select(year, doy, block, rot_trt, plot_id, rep_id, depth_cm, resis_Mpa) %>%
  arrange(block, plot_id, rep_id, depth_cm)

# make new factor variables and convert old trt/block variables into factors
# sorry, I couldn't figure out how to do this in mutate() without gnarly warnings
myd$year_doy <- as.factor(paste(myd$year, myd$doy, sep = "_"))
myd$trt_yr <- as.factor(paste(myd$rot_trt, myd$year, myd$doy, sep = "_"))
# myd$trt_block_yr <- as.factor(paste(myd$rot_trt, myd$block, myd$year, myd$doy, sep = "_"))
myd$block <- as.factor(myd$block)
myd$rot_trt <- as.factor(myd$rot_trt)

head(myd, 12)
```

year	doy	block	rot_trt	plot_id	rep_id	depth_cm	resis_Mpa	year_doy	trt_yr
2018	130	b1	2y	2018_13	2018_13-1	0.00	0.0000000	2018_130	2y_2018_130
2018	130	b1	2y	2018_13	2018_13-1	2.54	0.8618450	2018_130	2y_2018_130
2018	130	b1	2y	2018_13	2018_13-1	5.08	0.9307926	2018_130	2y_2018_130
2018	130	b1	2y	2018_13	2018_13-1	7.62	0.4481594	2018_130	2y_2018_130
2018	130	b1	2y	2018_13	2018_13-1	10.16	0.6205284	2018_130	2y_2018_130
2018	130	b1	2y	2018_13	2018_13-1	12.70	0.6894760	2018_130	2y_2018_130
2018	130	b1	2y	2018_13	2018_13-1	15.24	0.7584236	2018_130	2y_2018_130
2018	130	b1	2y	2018_13	2018_13-1	17.78	0.7928974	2018_130	2y_2018_130
2018	130	b1	2y	2018_13	2018_13-1	20.32	0.7928974	2018_130	2y_2018_130
2018	130	b1	2y	2018_13	2018_13-1	22.86	0.7584236	2018_130	2y_2018_130
2018	130	b1	2y	2018_13	2018_13-1	25.40	0.6205284	2018_130	2y_2018_130
2018	130	b1	2y	2018_13	2018_13-1	27.94	0.6205284	2018_130	2y_2018_130

Data visualization

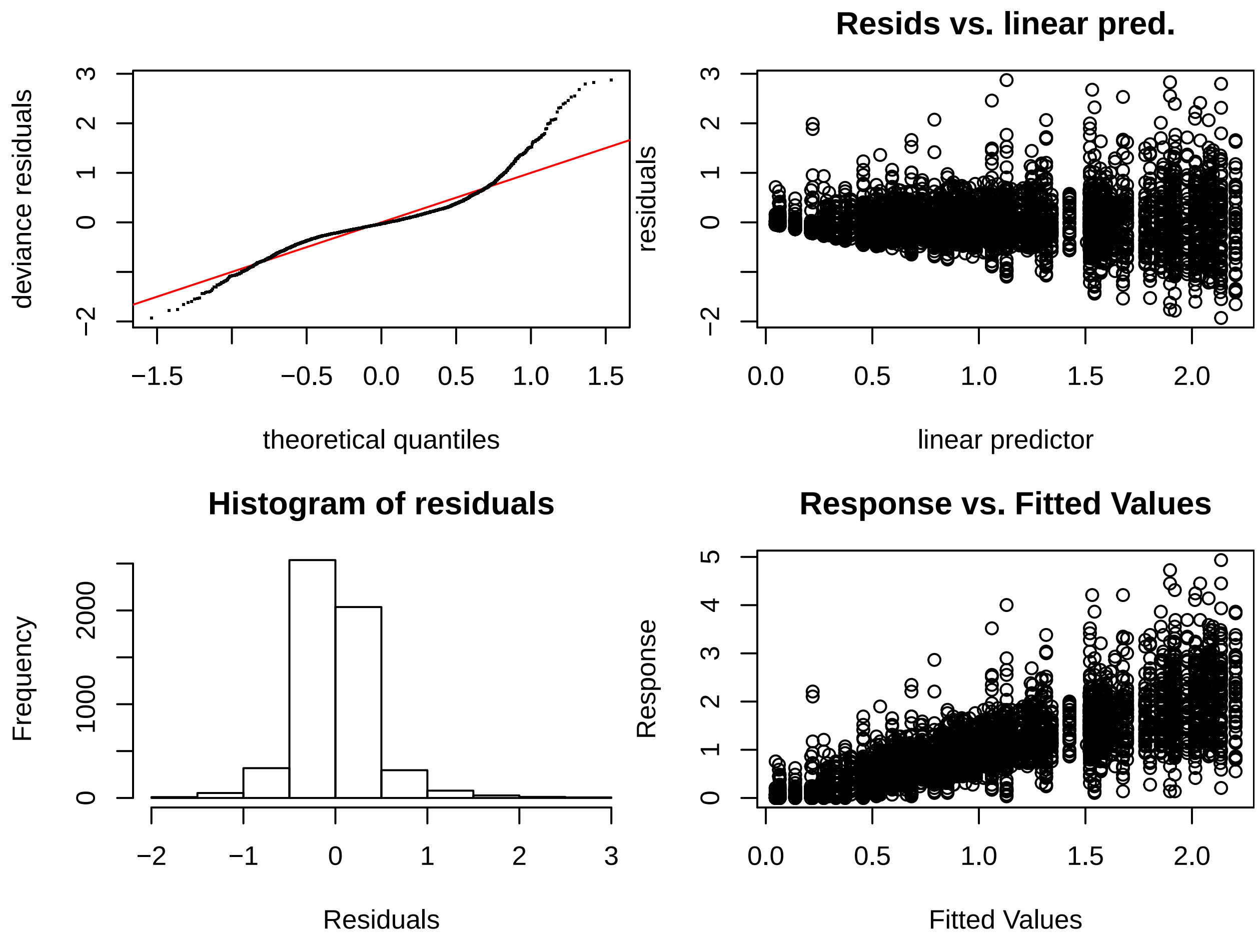
```
# may want to account for interaction w/ year? looks like pattern changes over time
ggplot(data = myd) +
  geom_line(aes(x = depth_cm, y = resis_Mpa, group = rep_id, color = year_doy)) +
  facet_wrap( ~ block + rot_trt, ncol = 2)
```



Fit and check GAM

```
library(mgcv)
mod1 <- gam(resis_Mpa ~ s(depth_cm, by = trt_yr, bs = "cr", k = 8) + trt_yr,
            data = myd, method = "REML")
```

```
# plot(mod, residuals = TRUE, shade = TRUE)
par(mar = c(4, 4, 3, 0))
mgcv::gam.check(mod1)
```




```
##
## Method: REML   Optimizer: outer newton
## full convergence after 6 iterations.
## Gradient range [-6.082002e-05,5.147386e-05]
## (score 2938.772 & scale 0.1693312).
## Hessian positive definite, eigenvalue range [1.079045,2677.514].
## Model rank =  64 / 64
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
```

```
##
##                                    k'  edf k-index p-value
## s(depth_cm):trt_yr2y_2018_130 7.00 4.45    1.03    0.97
## s(depth_cm):trt_yr2y_2018_197 7.00 5.87    1.03    0.97
## s(depth_cm):trt_yr2y_2019_162 7.00 4.40    1.03    0.99
## s(depth_cm):trt_yr2y_2020_175 7.00 6.17    1.03    1.00
## s(depth_cm):trt_yr4y_2018_130 7.00 5.72    1.03    0.98
## s(depth_cm):trt_yr4y_2018_197 7.00 5.38    1.03    0.98
## s(depth_cm):trt_yr4y_2019_162 7.00 4.80    1.03    0.96
## s(depth_cm):trt_yr4y_2020_175 7.00 5.05    1.03    0.99
```

To check a GAM with `gam.check()`, we look for a few things:

- If `edf` is too close to `k'`, we may need more knots
- `k-index` is the estimate divided by the residual variance. The further below 1 this is, the more likely it is that there is missed pattern left in the residuals.
- The p-value for the `k-index` is computed by simulation: the residuals are randomly re-shuffled `k.rep` times to obtain the null distribution of the differencing variance estimator, if there is no pattern in the residuals.
- If the p-value is too close to zero, there is a significant pattern in the residuals that should be addressed.

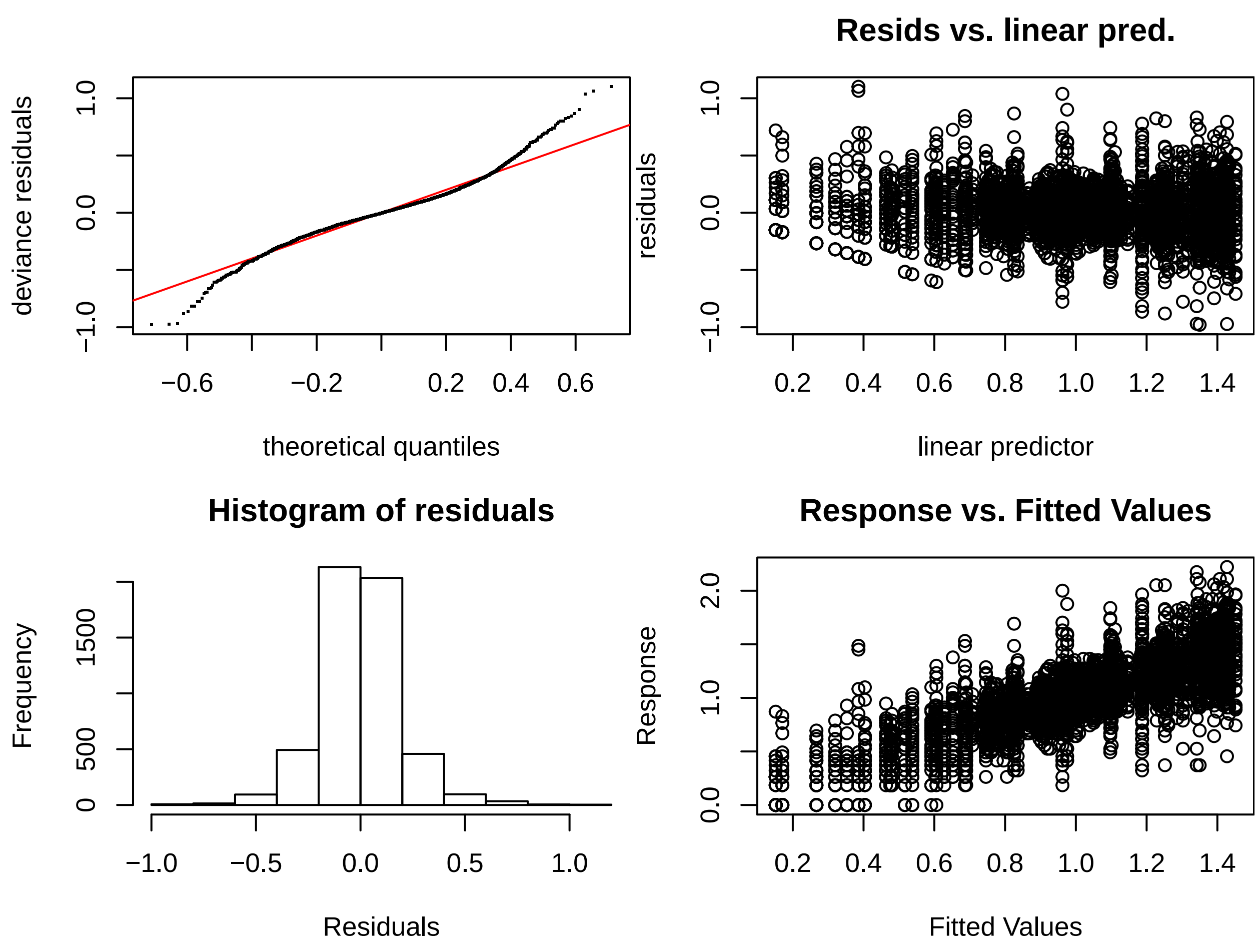
In this case, the `edf` values are not too close to 7, and the `k-index` is close to 1, but the residuals are heteroskedastic (i.e., have a megaphone shape), likely because the curves are all close to zero at zero depth but spread out a lot for higher depths.

Next, I transform `resis_Mpa` with the square root (since all resistance values are non-negative) and fit another model, to unify/control variances at high depths.

Adjust for non-constant variance of `depth_cm` and refit

```
mod2 <- gam(sqrt(resis_Mpa) ~ s(depth_cm, by = trt_yr, bs = "cr", k = 8) + trt_yr,
            data = myd, method = "REML")
```

```
# plot(mod, residuals = TRUE, shade = TRUE)
par(mar = c(4, 4, 3, 0))
mgcv::gam.check(mod2)
```



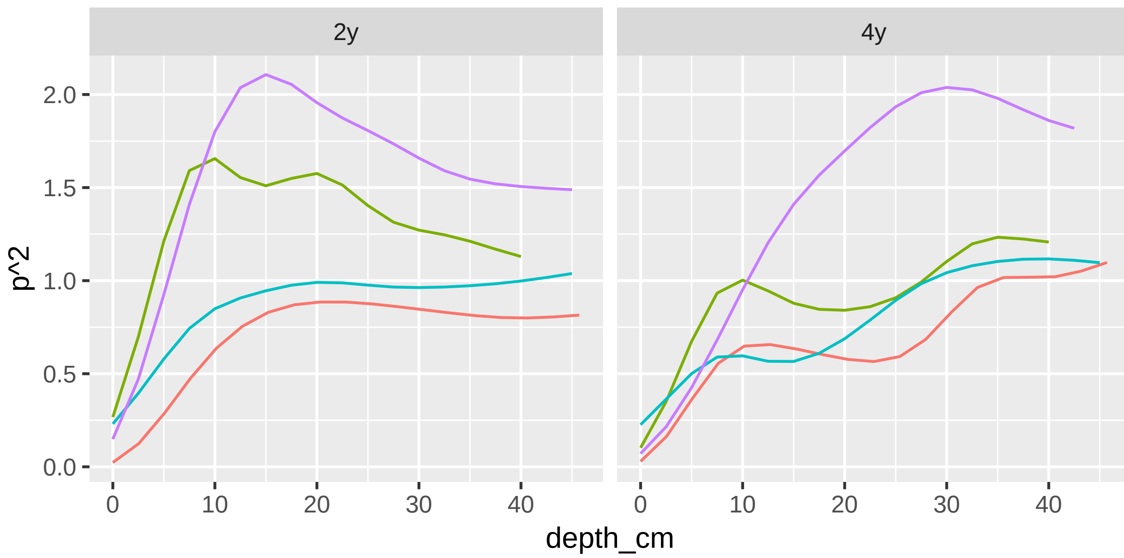

```
##
## Method: REML   Optimizer: outer newton
## full convergence after 6 iterations.
## Gradient range [-8.449579e-05,7.707071e-05]
## (score -1178.483 & scale 0.03614058).
## Hessian positive definite, eigenvalue range [1.952712,2677.519].
## Model rank =  64 / 64
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
```

```
##                                  k'  edf k-index p-value
## s(depth_cm):trt_yr2y_2018_130 7.00 5.84    1.01    0.68
## s(depth_cm):trt_yr2y_2018_197 7.00 6.16    1.01    0.72
## s(depth_cm):trt_yr2y_2019_162 7.00 5.55    1.01    0.72
## s(depth_cm):trt_yr2y_2020_175 7.00 6.25    1.01    0.71
## s(depth_cm):trt_yr4y_2018_130 7.00 6.49    1.01    0.63
## s(depth_cm):trt_yr4y_2018_197 7.00 5.98    1.01    0.70
## s(depth_cm):trt_yr4y_2019_162 7.00 5.98    1.01    0.73
## s(depth_cm):trt_yr4y_2020_175 7.00 5.37    1.01    0.70
```

Looking at the residuals plots for each group (two pages further down this document), it seems that there is still an issue with non-constant variance, but this time by year/doy instead of depth. To investigate, I remove the problem year and fit the model one more time.

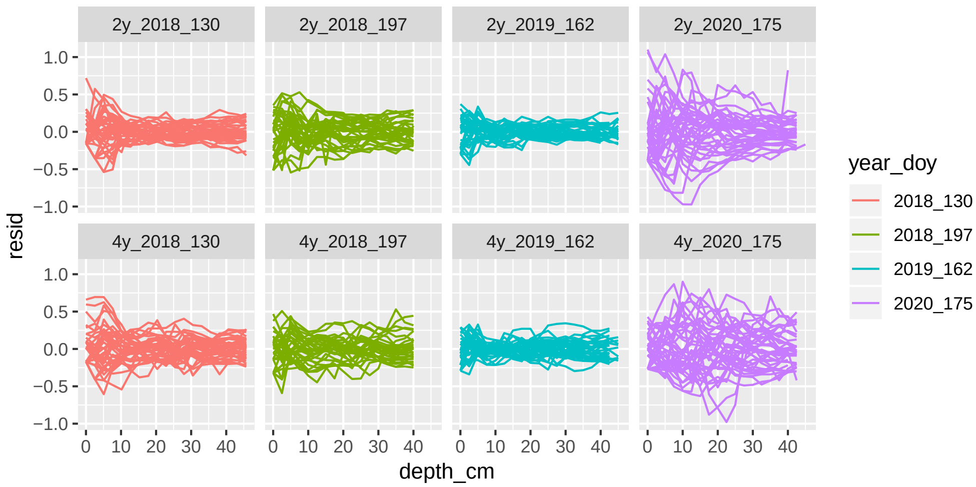
View fitted model by group

```r
# view the 24 fitted curves to visually inspect differences
myd$p <- predict(mod2)

ggplot(data = myd) +
  geom_line(aes(x = depth_cm, y = p^2, color = year_doy, group = year_doy)) + # note p^2
  facet_wrap( ~ rot_trt, ncol = 2) +
  guides(color = FALSE)
```



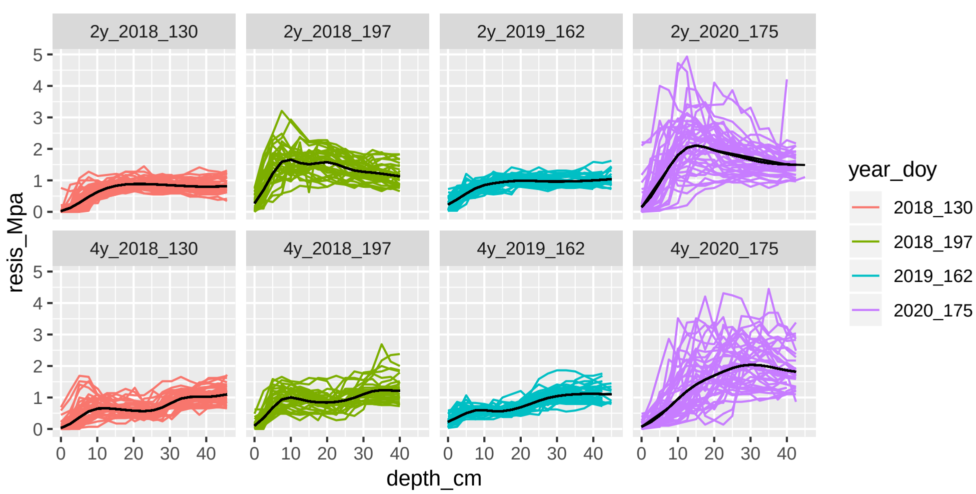
View residuals by group

```r
myd$resid <- mod2$residuals

ggplot(data = myd) +
  geom_line(aes(x = depth_cm, y = resid, group = rep_id, color = year_doy)) +
  facet_wrap( ~ trt_yr, ncol = length(unique(myd$year_doy)))
```



View data with fitted curves by group

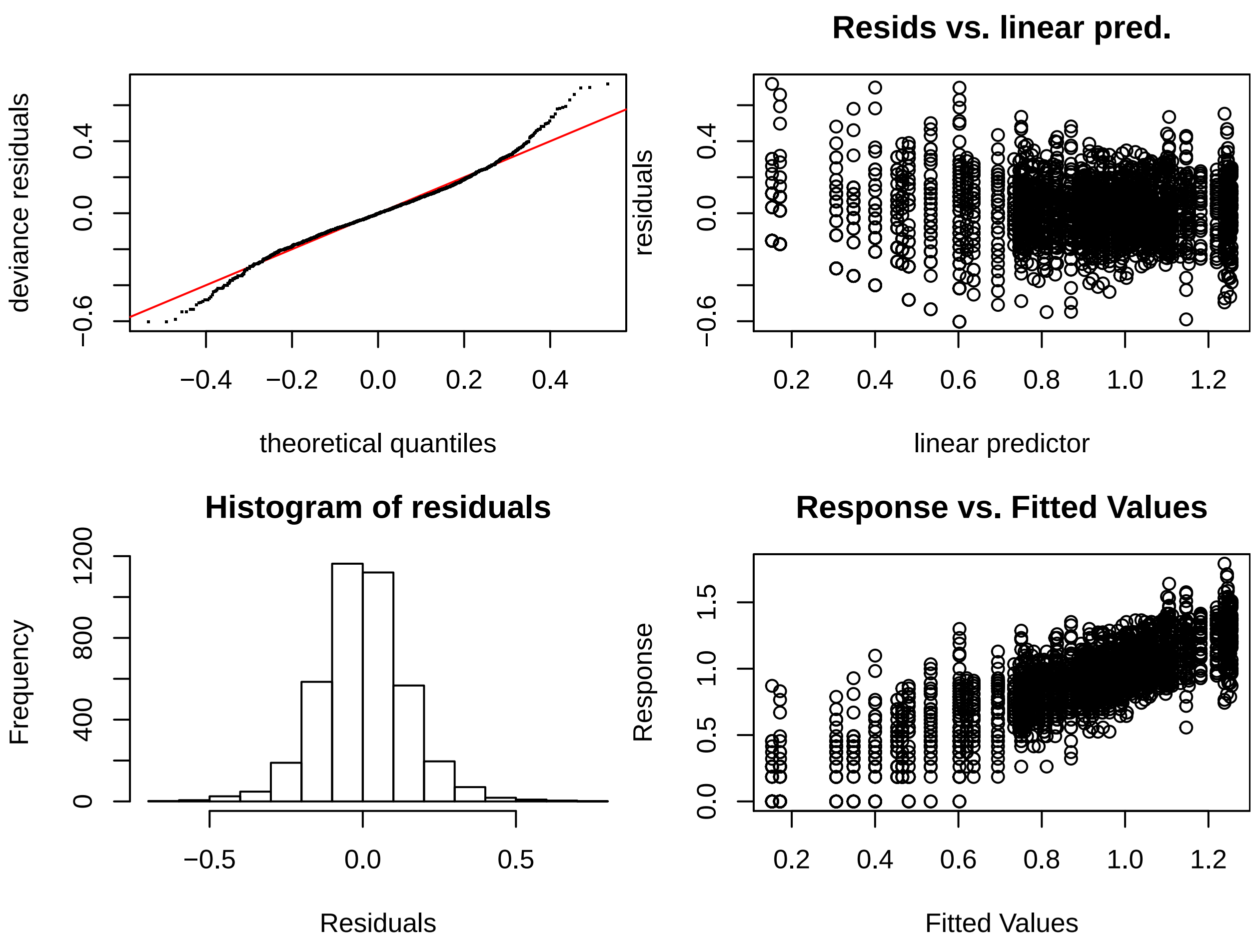
```r
ggplot(data = myd) +
  geom_line(aes(x = depth_cm, y = resis_Mpa, group = rep_id, color = year_doy)) +
  geom_line(aes(x = depth_cm, y = p^2, group = rep_id), color = "black") + # note p^2
  facet_wrap( ~ trt_yr, ncol = length(unique(myd$year_doy)))
```



Remove year that is most unlike the others and (re)refit

```
myd2 <- filter(myd, year != 2020)
myd2$trt_yr <- as.factor(as.character(myd2$trt_yr))

mod3 <- gam(sqrt(resis_Mpa) ~ s(depth_cm, by = trt_yr, bs = "cr", k = 12) + trt_yr,
            data = myd2, method = "REML")
```

```
# plot(mod, residuals = TRUE, shade = TRUE)
par(mar = c(4, 4, 3, 0))
mgcv::gam.check(mod3)
```



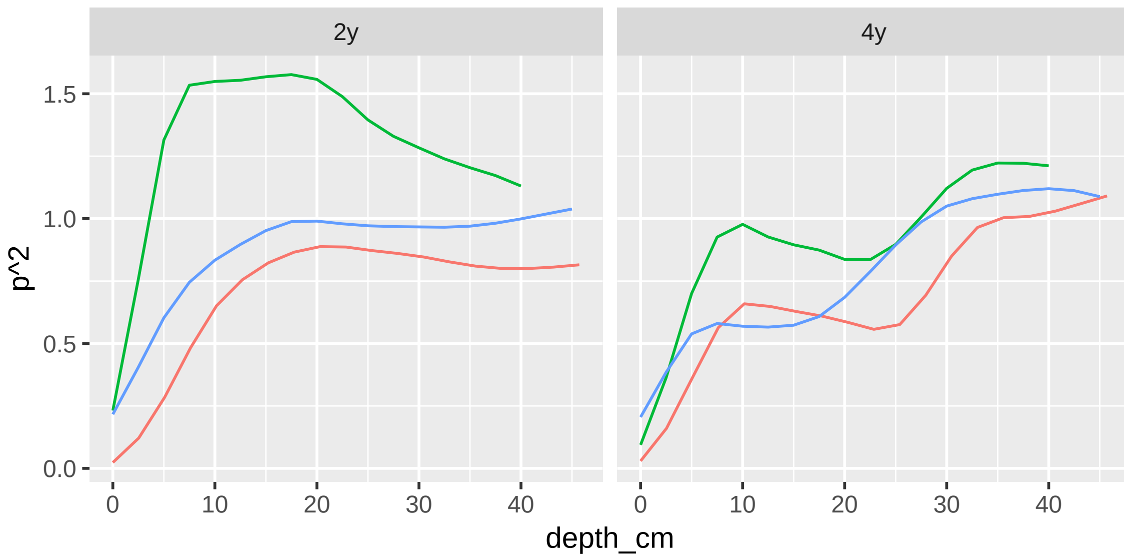

```
##
## Method: REML   Optimizer: outer newton
## full convergence after 5 iterations.
## Gradient range [-0.0001170755,7.73191e-05]
## (score -1920.789 & scale 0.02126053).
## Hessian positive definite, eigenvalue range [2.525453,1995.537].
## Model rank =  72 / 72
##
```

```
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##                                    k'   edf k-index p-value
## s(depth_cm):trt_yr2y_2018_130 11.00  7.28    0.94  <2e-16 ***
## s(depth_cm):trt_yr2y_2018_197 11.00  8.54    0.94  <2e-16 ***
## s(depth_cm):trt_yr2y_2019_162 11.00  7.09    0.94  <2e-16 ***
## s(depth_cm):trt_yr4y_2018_130 11.00  8.77    0.94  <2e-16 ***
## s(depth_cm):trt_yr4y_2018_197 11.00  8.01    0.94  <2e-16 ***
## s(depth_cm):trt_yr4y_2019_162 11.00  7.99    0.94  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Whoops, that made the p-values worse… Let’s stick with the second model?

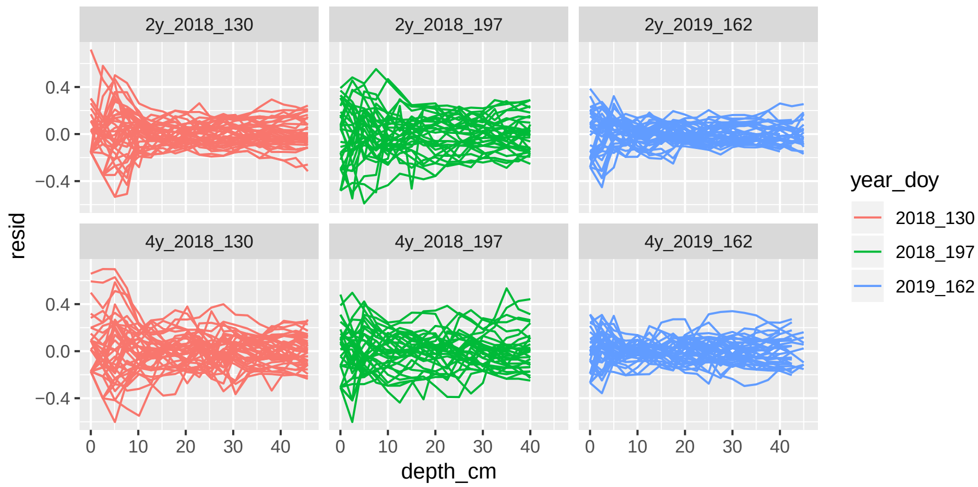
View fitted model by group

```r
# view the 24 fitted curves to visually inspect differences
myd2$p <- predict(mod3)

ggplot(data = myd2) +
  geom_line(aes(x = depth_cm, y = p^2, color = year_doy, group = year_doy)) + # note p^2
  facet_wrap( ~ rot_trt, ncol = 2) +
  guides(color = FALSE)
```



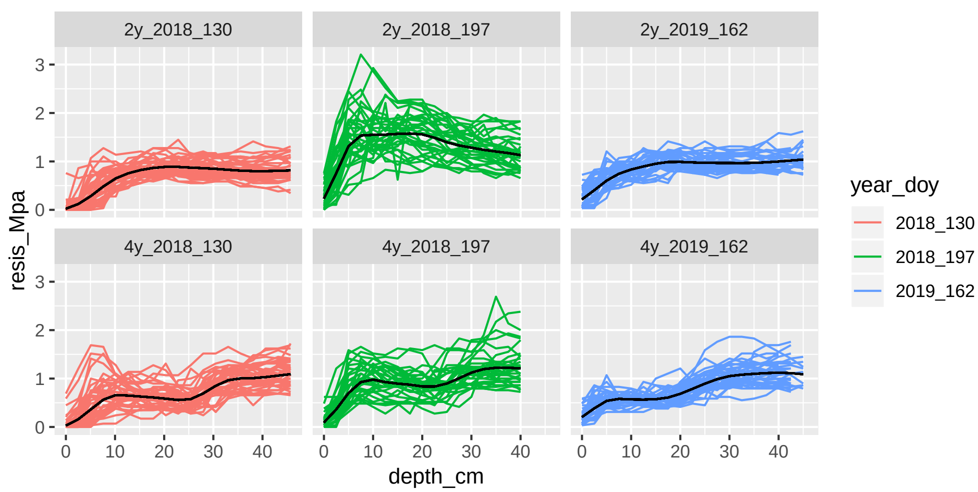
View residuals by group

```
myd2$resid <- mod3$residuals

ggplot(data = myd2) +
  geom_line(aes(x = depth_cm, y = resid, group = rep_id, color = year_doy)) +
  facet_wrap( ~ trt_yr, nrow = 2)
```



View data with fitted curves by group

```
ggplot(data = myd2) +
  geom_line(aes(x = depth_cm, y = resis_Mpa, group = rep_id, color = year_doy)) +
  geom_line(aes(x = depth_cm, y = p^2, group = rep_id), color = "black") + # note p^2
  facet_wrap( ~ trt_yr, nrow = 2)
```



Differences between treatments - 1st way

To test for differences between treatments, we'll use the following function from https://fromthebottomoftheheap.net/2017/10/10/difference-splines-i/:

```
smooth_diff <- function(model, newdata, f1, f2, var, alpha = 0.05,
                        unconditional = FALSE) {
    xp <- predict(model, newdata = newdata, type = 'lpmatrix')
    c1 <- grepl(f1, colnames(xp))
    c2 <- grepl(f2, colnames(xp))
    r1 <- newdata[[var]] == f1
    r2 <- newdata[[var]] == f2
    ## difference rows of xp for data from comparison
    X <- xp[r1, ] - xp[r2, ]
    ## zero out cols of X related to splines for other lochs
    X[, ! (c1 | c2)] <- 0
    ## zero out the parametric cols
    X[, !grepl('^s\\(', colnames(xp))] <- 0
    dif <- X %*% coef(model)
    se <- sqrt(rowSums((X %*% vcov(model, unconditional = unconditional)) * X))
    crit <- qt(alpha/2, df.residual(model), lower.tail = FALSE)
    upr <- dif + (crit * se)
    lwr <- dif - (crit * se)
    data.frame(pair = paste(f1, f2, sep = '-'),
               diff = dif,
               se = se,
               upper = upr,
               lower = lwr)
}
```

We want to compare the differences between treatments, within years, and averaging over blocks. First, we set up a data frame that pairs the variable levels between treatments, within year/doy values (and leaving out mention of blocks).

```
# get all combinations year/doy (excluding block this time)
base_str <- unique(myd$year_doy)

# add trt to the front of each base str, yielding pairwise df
var_df <- data.frame(var1 = paste("2y", base_str, sep = "_"),
                     var2 = paste("4y", base_str, sep = "_"),
                     stringsAsFactors = FALSE)

head(var_df)
```

var1	var2
2y_2018_130	4y_2018_130
2y_2018_197	4y_2018_197
2y_2019_162	4y_2019_162
2y_2020_175	4y_2020_175

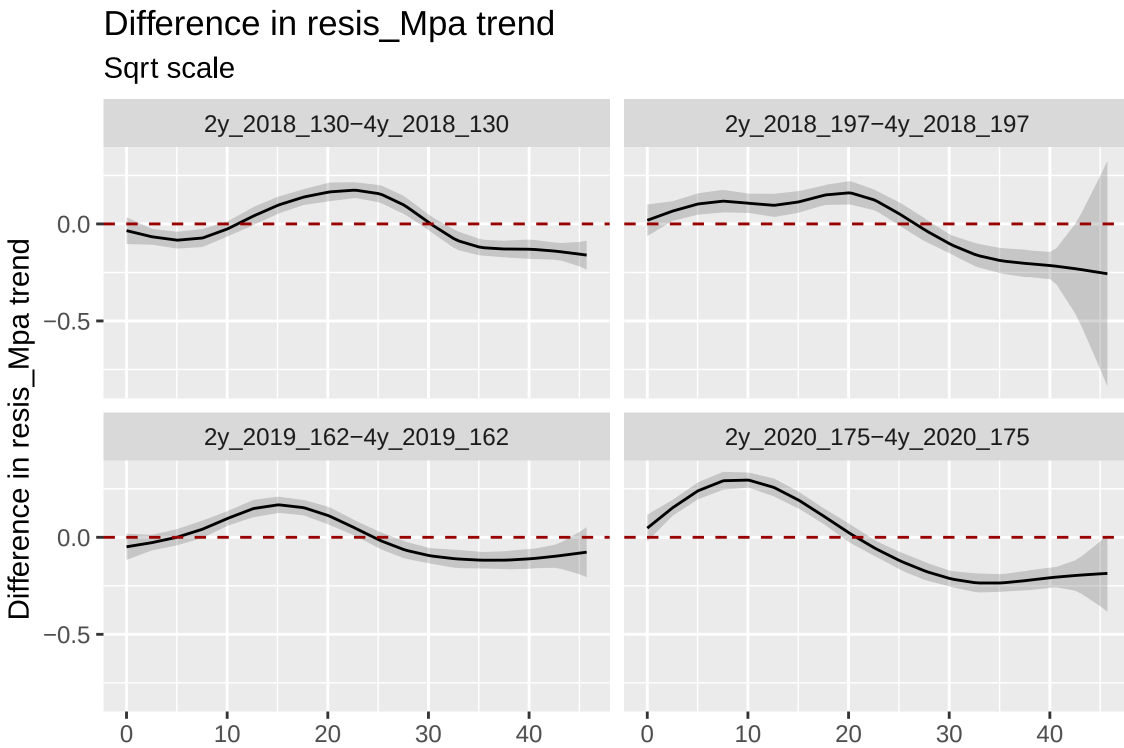
Next, we create a data frame for prediction that contains the depth values and levels of `trt_yr`. (NOTE: using `mod2`)

```
newdata <- expand.grid(depth_cm = unique(myd$depth_cm),
                       trt_yr = levels(myd$trt_yr))

out <- purrr::map_dfr(1:nrow(var_df), function(i) {
  d <- smooth_diff(mod2, newdata,
              f1 = var_df[i,1], f2 = var_df[i,2],
              var = "trt_yr")
  d$pair <- as.character(d$pair) # prevent map_dfr from combining factors
  return(d)
})

comp <- cbind(depth_cm = unique(myd$depth_cm), out) # add depth values
comp$pair <- as.factor(comp$pair) # make this a factor again for ggplot2
head(comp)
```

depth_cm	pair	diff	se	upper	lower
0.00	2y_2018_130-4y_2018_130	-0.0344902	0.0353856	0.0348801	-0.1038604
2.54	2y_2018_130-4y_2018_130	-0.0664567	0.0208016	-0.0256770	-0.1072365
5.08	2y_2018_130-4y_2018_130	-0.0836554	0.0222847	-0.0399683	-0.1273425
7.62	2y_2018_130-4y_2018_130	-0.0713186	0.0235548	-0.0251414	-0.1174958
10.16	2y_2018_130-4y_2018_130	-0.0224009	0.0200018	0.0168108	-0.0616125
12.70	2y_2018_130-4y_2018_130	0.0427829	0.0238498	0.0895384	-0.0039725

Now, we can plot the difference between treatments, with associated confidence bands, for each block and year/doy combination. Any depth values where the shading does not cross the dashed red line (at diff = 0) are points where the treatment resistances differ significantly. Note: this is all still on the sqrt scale. In other words, this is the difference of the expected square root of the response. To conduct inference on the original scale, check out the next section.

```
ggplot(comp, aes(x = depth_cm, y = diff, group = pair)) +
    geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +
    geom_line() +
    geom_hline(aes(yintercept = 0), colour="#990000", linetype="dashed") +
    facet_wrap(~ pair, ncol = 2) +
    labs(x = NULL, y = 'Difference in resis_Mpa trend') +
  ggtitle("Difference in resis_Mpa trend", subtitle = "Sqrt scale")
```


Difference in resis_Mpa trend
Sqrt scale
2y_2018_130−4y_2018_130
2y_2018_197−4y_2018_197
2y_2019_162−4y_2019_162
2y_2020_175−4y_2020_175
Difference in resis_Mpa trend
0.0
−0.5
0
10
20
30
40

Differences between treatments - 2nd way

Here's a quick intro to `emmeans` by example:

```
library(emmeans)
em <- emmeans(mod2, specs = "trt_yr") # get estimated marginal means
em
```

```
##  trt_yr      emmean     SE   df lower.CL upper.CL
##  2y_2018_130  0.942 0.0175 5315    0.907    0.976
##  2y_2018_197  1.248 0.0210 5315    1.207    1.289
##  2y_2019_162  0.996 0.0168 5315    0.963    1.029
##  2y_2020_175  1.383 0.0185 5315    1.347    1.420
##  4y_2018_130  0.755 0.0185 5315    0.719    0.791
##  4y_2018_197  0.920 0.0207 5315    0.880    0.961
##  4y_2019_162  0.857 0.0177 5315    0.822    0.892
##  4y_2020_175  1.326 0.0172 5315    1.292    1.359
##
## Results are given on the sqrt (not the response) scale.
## Confidence level used: 0.95
```

```
pairs(em) # get pairwise contrasts
```

```
##  contrast                  estimate     SE   df t.ratio p.value
##  2y_2018_130 - 2y_2018_197  -0.3061 0.0273 5315 -11.198 <.0001
##  2y_2018_130 - 2y_2019_162  -0.0542 0.0243 5315  -2.228 0.3351
##  2y_2018_130 - 2y_2020_175  -0.4419 0.0255 5315 -17.318 <.0001
##  2y_2018_130 - 4y_2018_130   0.1864 0.0255 5315   7.304 <.0001
##  2y_2018_130 - 4y_2018_197   0.0211 0.0272 5315   0.778 0.9942
##  2y_2018_130 - 4y_2019_162   0.0846 0.0249 5315   3.390 0.0161
##  2y_2018_130 - 4y_2020_175  -0.3843 0.0245 5315 -15.659 <.0001
##  2y_2018_197 - 2y_2019_162   0.2519 0.0269 5315   9.369 <.0001
##  2y_2018_197 - 2y_2020_175  -0.1358 0.0280 5315  -4.853 <.0001
##  2y_2018_197 - 4y_2018_130   0.4925 0.0280 5315  17.604 <.0001
##  2y_2018_197 - 4y_2018_197   0.3273 0.0295 5315  11.101 <.0001
##  2y_2018_197 - 4y_2019_162   0.3907 0.0275 5315  14.231 <.0001
##  2y_2018_197 - 4y_2020_175  -0.0781 0.0271 5315  -2.884 0.0760
##  2y_2019_162 - 2y_2020_175  -0.3877 0.0250 5315 -15.486 <.0001
##  2y_2019_162 - 4y_2018_130   0.2405 0.0250 5315   9.608 <.0001
##  2y_2019_162 - 4y_2018_197   0.0753 0.0267 5315   2.820 0.0902
##  2y_2019_162 - 4y_2019_162   0.1387 0.0245 5315   5.675 <.0001
##  2y_2019_162 - 4y_2020_175  -0.3301 0.0240 5315 -13.730 <.0001
##  2y_2020_175 - 4y_2018_130   0.6282 0.0262 5315  23.981 <.0001
##  2y_2020_175 - 4y_2018_197   0.4630 0.0278 5315  16.655 <.0001
##  2y_2020_175 - 4y_2019_162   0.5264 0.0256 5315  20.534 <.0001
##  2y_2020_175 - 4y_2020_175   0.0576 0.0252 5315   2.283 0.3032
##  4y_2018_130 - 4y_2018_197  -0.1652 0.0278 5315  -5.943 <.0001
##  4y_2018_130 - 4y_2019_162  -0.1018 0.0256 5315  -3.971 0.0019
##  4y_2018_130 - 4y_2020_175  -0.5706 0.0252 5315 -22.601 <.0001
##  4y_2018_197 - 4y_2019_162   0.0634 0.0273 5315   2.325 0.2799
##  4y_2018_197 - 4y_2020_175  -0.4054 0.0269 5315 -15.067 <.0001
##  4y_2019_162 - 4y_2020_175  -0.4688 0.0247 5315 -19.006 <.0001
##
## P value adjustment: tukey method for comparing a family of 8 estimates
```

You can specify `specs = pairwise ~ var` to get the contrasts at the same time as the estimates.

```
em <- emmeans(mod2, specs = pairwise ~ "trt_yr")
em
```

```
## $emmeans
##  trt_yr      emmean     SE   df lower.CL upper.CL
##  2y_2018_130  0.942 0.0175 5315    0.907    0.976
##  2y_2018_197  1.248 0.0210 5315    1.207    1.289
##  2y_2019_162  0.996 0.0168 5315    0.963    1.029
##  2y_2020_175  1.383 0.0185 5315    1.347    1.420
##  4y_2018_130  0.755 0.0185 5315    0.719    0.791
##  4y_2018_197  0.920 0.0207 5315    0.880    0.961
##  4y_2019_162  0.857 0.0177 5315    0.822    0.892
##  4y_2020_175  1.326 0.0172 5315    1.292    1.359
##
## Results are given on the sqrt (not the response) scale.
## Confidence level used: 0.95
##
## $contrasts
##  contrast                  estimate     SE   df t.ratio p.value
##  2y_2018_130 - 2y_2018_197  -0.3061 0.0273 5315 -11.198 <.0001
##  2y_2018_130 - 2y_2019_162  -0.0542 0.0243 5315  -2.228 0.3351
##  2y_2018_130 - 2y_2020_175  -0.4419 0.0255 5315 -17.318 <.0001
##  2y_2018_130 - 4y_2018_130   0.1864 0.0255 5315   7.304 <.0001
##  2y_2018_130 - 4y_2018_197   0.0211 0.0272 5315   0.778 0.9942
##  2y_2018_130 - 4y_2019_162   0.0846 0.0249 5315   3.390 0.0161
##  2y_2018_130 - 4y_2020_175  -0.3843 0.0245 5315 -15.659 <.0001
##  2y_2018_197 - 2y_2019_162   0.2519 0.0269 5315   9.369 <.0001
##  2y_2018_197 - 2y_2020_175  -0.1358 0.0280 5315  -4.853 <.0001
##  2y_2018_197 - 4y_2018_130   0.4925 0.0280 5315  17.604 <.0001
##  2y_2018_197 - 4y_2018_197   0.3273 0.0295 5315  11.101 <.0001
##  2y_2018_197 - 4y_2019_162   0.3907 0.0275 5315  14.231 <.0001
##  2y_2018_197 - 4y_2020_175  -0.0781 0.0271 5315  -2.884 0.0760
##  2y_2019_162 - 2y_2020_175  -0.3877 0.0250 5315 -15.486 <.0001
##  2y_2019_162 - 4y_2018_130   0.2405 0.0250 5315   9.608 <.0001
##  2y_2019_162 - 4y_2018_197   0.0753 0.0267 5315   2.820 0.0902
##  2y_2019_162 - 4y_2019_162   0.1387 0.0245 5315   5.675 <.0001
##  2y_2019_162 - 4y_2020_175  -0.3301 0.0240 5315 -13.730 <.0001
##  2y_2020_175 - 4y_2018_130   0.6282 0.0262 5315  23.981 <.0001
##  2y_2020_175 - 4y_2018_197   0.4630 0.0278 5315  16.655 <.0001
##  2y_2020_175 - 4y_2019_162   0.5264 0.0256 5315  20.534 <.0001
##  2y_2020_175 - 4y_2020_175   0.0576 0.0252 5315   2.283 0.3032
##  4y_2018_130 - 4y_2018_197  -0.1652 0.0278 5315  -5.943 <.0001
##  4y_2018_130 - 4y_2019_162  -0.1018 0.0256 5315  -3.971 0.0019
##  4y_2018_130 - 4y_2020_175  -0.5706 0.0252 5315 -22.601 <.0001
##  4y_2018_197 - 4y_2019_162   0.0634 0.0273 5315   2.325 0.2799
##  4y_2018_197 - 4y_2020_175  -0.4054 0.0269 5315 -15.067 <.0001
##  4y_2019_162 - 4y_2020_175  -0.4688 0.0247 5315 -19.006 <.0001
##
## P value adjustment: tukey method for comparing a family of 8 estimates
```

Even better, `type = "response"` will transform estimates back from the square-root scale to original scale. (Here, contrasts are still on sqrt scale, more info here)

```
em <- emmeans(mod2, specs = pairwise ~ "trt_yr", type = "response")
em
```

```
## $emmeans
##  trt_yr      response     SE   df lower.CL upper.CL
##  2y_2018_130    0.886 0.0330 5315    0.823    0.952
##  2y_2018_197    1.557 0.0523 5315    1.456    1.661
##  2y_2019_162    0.991 0.0335 5315    0.927    1.058
##  2y_2020_175    1.914 0.0513 5315    1.815    2.016
##  4y_2018_130    0.570 0.0280 5315    0.517    0.626
##  4y_2018_197    0.847 0.0382 5315    0.774    0.924
##  4y_2019_162    0.734 0.0304 5315    0.676    0.795
##  4y_2020_175    1.758 0.0455 5315    1.670    1.848
##
## Confidence level used: 0.95
## Intervals are back-transformed from the sqrt scale
##
## $contrasts
##  contrast                  estimate     SE   df t.ratio p.value
##  2y_2018_130 - 2y_2018_197  -0.3061 0.0273 5315 -11.198 <.0001
##  2y_2018_130 - 2y_2019_162  -0.0542 0.0243 5315  -2.228 0.3351
##  2y_2018_130 - 2y_2020_175  -0.4419 0.0255 5315 -17.318 <.0001
##  2y_2018_130 - 4y_2018_130   0.1864 0.0255 5315   7.304 <.0001
##  2y_2018_130 - 4y_2018_197   0.0211 0.0272 5315   0.778 0.9942
##  2y_2018_130 - 4y_2019_162   0.0846 0.0249 5315   3.390 0.0161
##  2y_2018_130 - 4y_2020_175  -0.3843 0.0245 5315 -15.659 <.0001
##  2y_2018_197 - 2y_2019_162   0.2519 0.0269 5315   9.369 <.0001
##  2y_2018_197 - 2y_2020_175  -0.1358 0.0280 5315  -4.853 <.0001
##  2y_2018_197 - 4y_2018_130   0.4925 0.0280 5315  17.604 <.0001
##  2y_2018_197 - 4y_2018_197   0.3273 0.0295 5315  11.101 <.0001
##  2y_2018_197 - 4y_2019_162   0.3907 0.0275 5315  14.231 <.0001
##  2y_2018_197 - 4y_2020_175  -0.0781 0.0271 5315  -2.884 0.0760
##  2y_2019_162 - 2y_2020_175  -0.3877 0.0250 5315 -15.486 <.0001
##  2y_2019_162 - 4y_2018_130   0.2405 0.0250 5315   9.608 <.0001
##  2y_2019_162 - 4y_2018_197   0.0753 0.0267 5315   2.820 0.0902
##  2y_2019_162 - 4y_2019_162   0.1387 0.0245 5315   5.675 <.0001
##  2y_2019_162 - 4y_2020_175  -0.3301 0.0240 5315 -13.730 <.0001
##  2y_2020_175 - 4y_2018_130   0.6282 0.0262 5315  23.981 <.0001
##  2y_2020_175 - 4y_2018_197   0.4630 0.0278 5315  16.655 <.0001
##  2y_2020_175 - 4y_2019_162   0.5264 0.0256 5315  20.534 <.0001
##  2y_2020_175 - 4y_2020_175   0.0576 0.0252 5315   2.283 0.3032
##  4y_2018_130 - 4y_2018_197  -0.1652 0.0278 5315  -5.943 <.0001
##  4y_2018_130 - 4y_2019_162  -0.1018 0.0256 5315  -3.971 0.0019
##  4y_2018_130 - 4y_2020_175  -0.5706 0.0252 5315 -22.601 <.0001
##  4y_2018_197 - 4y_2019_162   0.0634 0.0273 5315   2.325 0.2799
##  4y_2018_197 - 4y_2020_175  -0.4054 0.0269 5315 -15.067 <.0001
##  4y_2019_162 - 4y_2020_175  -0.4688 0.0247 5315 -19.006 <.0001
##
## P value adjustment: tukey method for comparing a family of 8 estimates
```

Technically, these are estimates for depth = `mean(depth_cm)`. Say we want to specify estimating for `depth_cm = 5`:

```
em <- emmeans(mod2, specs = pairwise ~ "trt_yr", type = "response", at = list(depth_cm = 5))
em
```

```
## $emmeans
##  trt_yr      response     SE   df lower.CL upper.CL
##  2y_2018_130    0.284 0.0180 5315    0.249    0.320
##  2y_2018_197    1.213 0.0438 5315    1.129    1.301
##  2y_2019_162    0.579 0.0246 5315    0.532    0.629
##  2y_2020_175    0.926 0.0339 5315    0.860    0.993
##  4y_2018_130    0.360 0.0212 5315    0.320    0.403
##  4y_2018_197    0.674 0.0324 5315    0.612    0.739
##  4y_2019_162    0.501 0.0241 5315    0.455    0.549
##  4y_2020_175    0.425 0.0217 5315    0.384    0.469
##
## Confidence level used: 0.95
## Intervals are back-transformed from the sqrt scale
##
## $contrasts
##  contrast                  estimate     SE   df t.ratio p.value
##  2y_2018_130 - 2y_2018_197  -0.5690 0.0261 5315 -21.784 <.0001
##  2y_2018_130 - 2y_2019_162  -0.2286 0.0234 5315  -9.768 <.0001
##  2y_2018_130 - 2y_2020_175  -0.4297 0.0244 5315 -17.574 <.0001
##  2y_2018_130 - 4y_2018_130  -0.0678 0.0245 5315  -2.774 0.1018
##  2y_2018_130 - 4y_2018_197  -0.2885 0.0260 5315 -11.102 <.0001
##  2y_2018_130 - 4y_2019_162  -0.1751 0.0240 5315  -7.298 <.0001
##  2y_2018_130 - 4y_2020_175  -0.1198 0.0238 5315  -5.046 <.0001
##  2y_2018_197 - 2y_2019_162   0.3403 0.0256 5315  13.280 <.0001
##  2y_2018_197 - 2y_2020_175   0.1393 0.0266 5315   5.241 <.0001
##  2y_2018_197 - 4y_2018_130   0.5011 0.0266 5315  18.847 <.0001
##  2y_2018_197 - 4y_2018_197   0.2805 0.0280 5315  10.016 <.0001
##  2y_2018_197 - 4y_2019_162   0.3938 0.0262 5315  15.049 <.0001
##  2y_2018_197 - 4y_2020_175   0.4491 0.0259 5315  17.314 <.0001
##  2y_2019_162 - 2y_2020_175  -0.2010 0.0239 5315  -8.403 <.0001
##  2y_2019_162 - 4y_2018_130   0.1608 0.0239 5315   6.719 <.0001
##  2y_2019_162 - 4y_2018_197  -0.0599 0.0255 5315  -2.348 0.2677
##  2y_2019_162 - 4y_2019_162   0.0535 0.0235 5315   2.280 0.3047
##  2y_2019_162 - 4y_2020_175   0.1088 0.0232 5315   4.688 0.0001
##  2y_2020_175 - 4y_2018_130   0.3618 0.0250 5315  14.502 <.0001
##  2y_2020_175 - 4y_2018_197   0.1412 0.0265 5315   5.337 <.0001
##  2y_2020_175 - 4y_2019_162   0.2545 0.0245 5315  10.388 <.0001
##  2y_2020_175 - 4y_2020_175   0.3098 0.0243 5315  12.772 <.0001
##  4y_2018_130 - 4y_2018_197  -0.2207 0.0265 5315  -8.339 <.0001
##  4y_2018_130 - 4y_2019_162  -0.1073 0.0245 5315  -4.377 0.0003
##  4y_2018_130 - 4y_2020_175  -0.0520 0.0243 5315  -2.143 0.3875
##  4y_2018_197 - 4y_2019_162   0.1134 0.0260 5315   4.354 0.0004
##  4y_2018_197 - 4y_2020_175   0.1687 0.0258 5315   6.535 <.0001
##  4y_2019_162 - 4y_2020_175   0.0553 0.0238 5315   2.323 0.2810
##
## P value adjustment: tukey method for comparing a family of 8 estimates
```

Let's work with the estimates and contrasts as two separate objects.

```
em <- emmeans(mod2, specs = "trt_yr", type = "response") # estimates
pem <- pairs(em) # contrasts
```

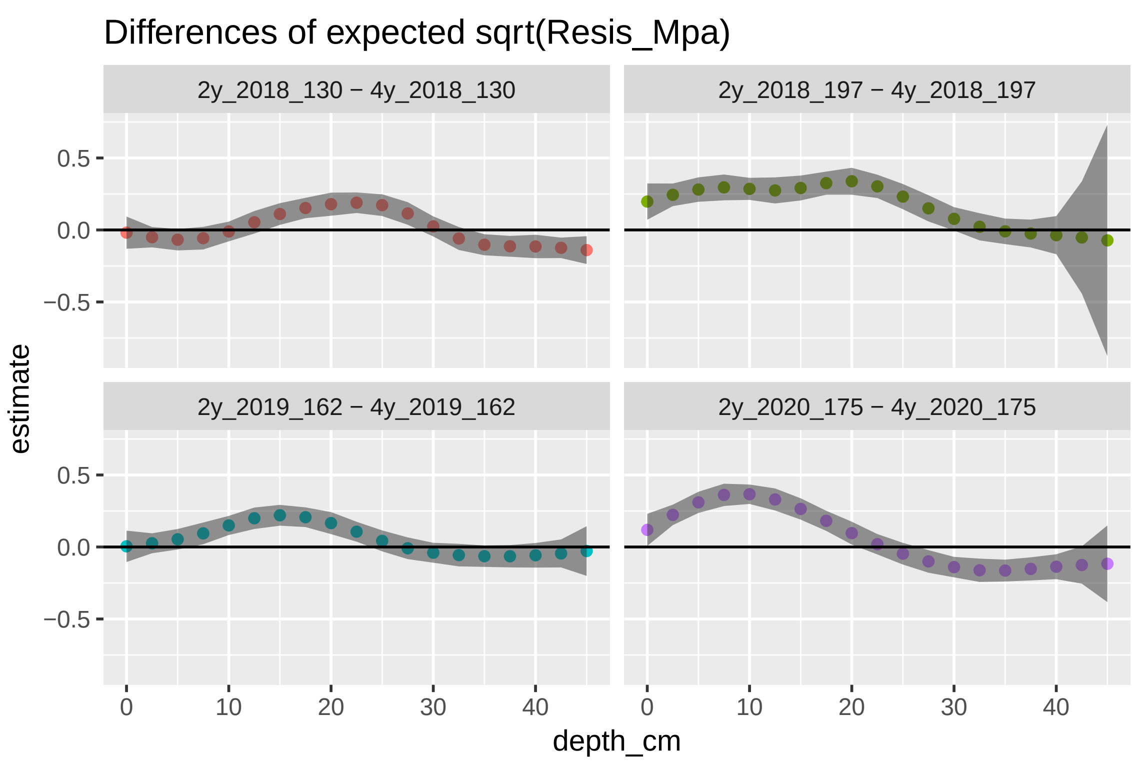
The `summary` of a pairs object can provide CIs of the estimated differences when `infer = c(TRUE, TRUE)`: (using `kable` to print only 3 digits - otherwise, output is too wide and p-vals end up on separate page)

```
knitr::kable(summary(pem, infer = c(TRUE, TRUE)), digits = 3) # using kable to fit on 1pg
```

contrast	estimate	SE	df	lower.CL	upper.CL	t.ratio	p.value
2y_2018_130 - 2y_2018_197	-0.306	0.027	5315.376	-0.389	-0.223	-11.198	0.000
2y_2018_130 - 2y_2019_162	-0.054	0.024	5315.376	-0.128	0.020	-2.228	0.335
2y_2018_130 - 2y_2020_175	-0.442	0.026	5315.376	-0.519	-0.365	-17.318	0.000
2y_2018_130 - 4y_2018_130	0.186	0.026	5315.376	0.109	0.264	7.304	0.000
2y_2018_130 - 4y_2018_197	0.021	0.027	5315.376	-0.061	0.103	0.778	0.994
2y_2018_130 - 4y_2019_162	0.085	0.025	5315.376	0.009	0.160	3.390	0.016
2y_2018_130 - 4y_2020_175	-0.384	0.025	5315.376	-0.459	-0.310	-15.659	0.000
2y_2018_197 - 2y_2019_162	0.252	0.027	5315.376	0.170	0.333	9.369	0.000
2y_2018_197 - 2y_2020_175	-0.136	0.028	5315.376	-0.221	-0.051	-4.853	0.000
2y_2018_197 - 4y_2018_130	0.492	0.028	5315.376	0.408	0.577	17.604	0.000
2y_2018_197 - 4y_2018_197	0.327	0.029	5315.376	0.238	0.417	11.100	0.000
2y_2018_197 - 4y_2019_162	0.391	0.027	5315.376	0.307	0.474	14.231	0.000
2y_2018_197 - 4y_2020_175	-0.078	0.027	5315.376	-0.160	0.004	-2.884	0.076
2y_2019_162 - 2y_2020_175	-0.388	0.025	5315.376	-0.464	-0.312	-15.486	0.000
2y_2019_162 - 4y_2018_130	0.241	0.025	5315.376	0.165	0.316	9.608	0.000
2y_2019_162 - 4y_2018_197	0.075	0.027	5315.376	-0.006	0.156	2.820	0.090
2y_2019_162 - 4y_2019_162	0.139	0.024	5315.376	0.065	0.213	5.675	0.000
2y_2019_162 - 4y_2020_175	-0.330	0.024	5315.376	-0.403	-0.257	-13.730	0.000
2y_2020_175 - 4y_2018_130	0.628	0.026	5315.376	0.549	0.708	23.981	0.000
2y_2020_175 - 4y_2018_197	0.463	0.028	5315.376	0.379	0.547	16.655	0.000
2y_2020_175 - 4y_2019_162	0.526	0.026	5315.376	0.449	0.604	20.534	0.000
2y_2020_175 - 4y_2020_175	0.058	0.025	5315.376	-0.019	0.134	2.283	0.303
4y_2018_130 - 4y_2018_197	-0.165	0.028	5315.376	-0.250	-0.081	-5.943	0.000
4y_2018_130 - 4y_2019_162	-0.102	0.026	5315.376	-0.180	-0.024	-3.971	0.002
4y_2018_130 - 4y_2020_175	-0.571	0.025	5315.376	-0.647	-0.494	-22.601	0.000
4y_2018_197 - 4y_2019_162	0.063	0.027	5315.376	-0.019	0.146	2.325	0.280
4y_2018_197 - 4y_2020_175	-0.405	0.027	5315.376	-0.487	-0.324	-15.067	0.000
4y_2019_162 - 4y_2020_175	-0.469	0.025	5315.376	-0.544	-0.394	-19.006	0.000

Now, it would be awesome to let emmeans do the work of estimating the difference between two curves at all of the values we care about. In fact, applying `emmeans` with `at = list(depth_cm = v)` for many values of `v` gives us the same difference curves from `smooth_diff`, the first method of inference we discussed.

```r
vec <- seq(from = 0, to = 45, by = 2.5) # can use finer seq if desired

diffs_df <- purrr::map_dfr(vec, function(v) {
  em <- emmeans(mod2, specs = "trt_yr", type = "response", at = list(depth_cm = v))
  pem <- pairs(em)
  df <- as.data.frame(summary(pem, infer = c(TRUE, TRUE)))
  df$depth_cm <- v
  df
})

# grab only the rows that compare the treatments within the same year
str <- paste(var_df$var1, "-", var_df$var2)
diffs_df <- dplyr::filter(diffs_df, contrast %in% str)

ggplot(diffs_df) +
  geom_point(aes(x = depth_cm, y = estimate, color = contrast)) +
  geom_ribbon(aes(x = depth_cm, ymin = lower.CL, ymax = upper.CL), alpha = .5) +
  facet_wrap(~contrast) + geom_hline(yintercept = 0) +
  guides(color = FALSE) +
  ggtitle("Differences of expected sqrt(Resis_Mpa)")
```



We can also use a similar strategy to extract the predictions and their CIs, and plot them on the same curve to visualize where differences are significant on the original scale.

```
vec <- seq(from = 0, to = 45, by = 2.5)

ests_df <- purrr::map_dfr(vec, function(v) {
  em <- emmeans(mod2, specs = "trt_yr", type = "response", at = list(depth_cm = v))
  df <- as.data.frame(em)
  df$depth_cm <- v

  # want trt information separately for faceeting later
  df <- tidyr::separate(df, col = trt_yr, sep = "y_", remove = FALSE,
                        into = c("trt", "yr_doy"))
  df
})

ests_df <- arrange(ests_df, depth_cm)

ggplot(ests_df, aes(group = trt, color = trt, fill = trt)) +
  geom_line(aes(x = depth_cm, y = response)) +
  geom_ribbon(aes(x = depth_cm, ymin = lower.CL, ymax = upper.CL),
              alpha = .5) +
  facet_wrap(~yr_doy)
```