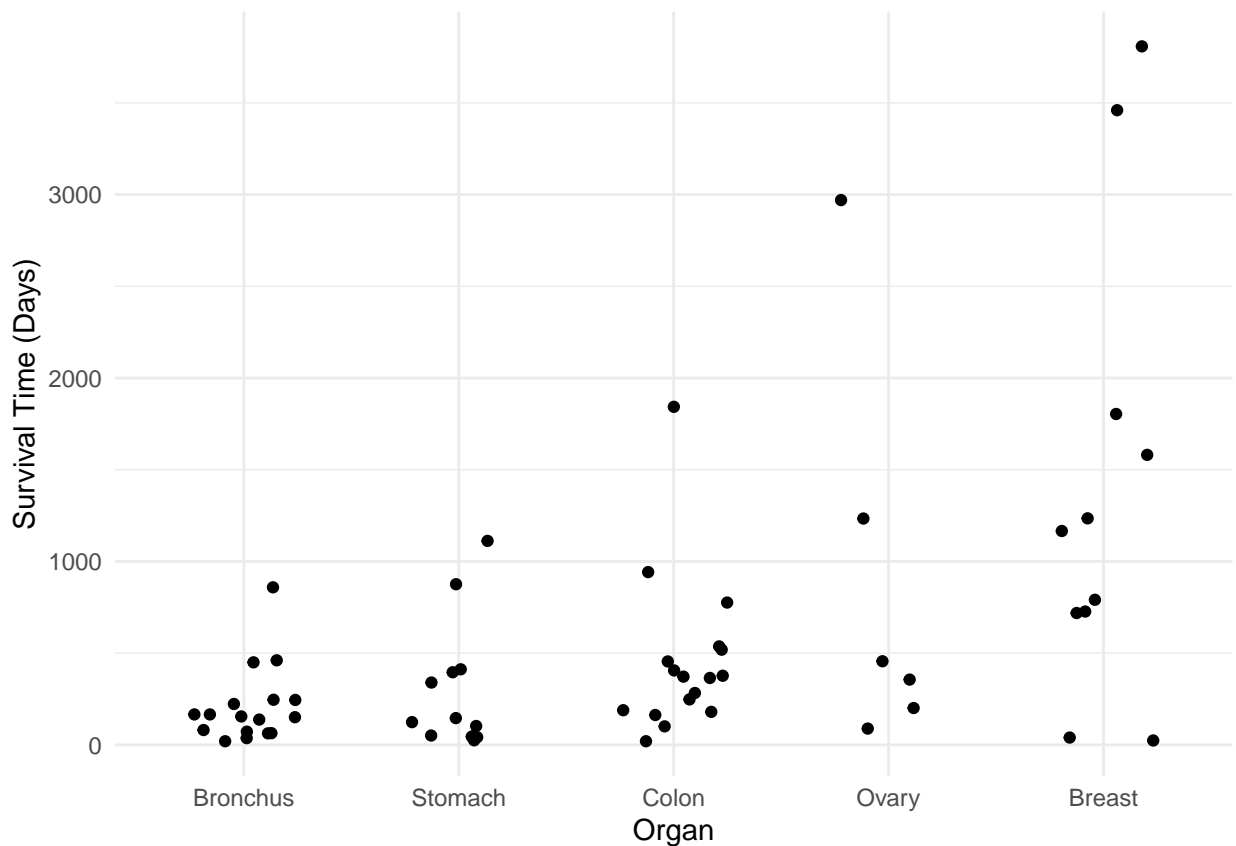# Friday, February 14

Solutions for Heteroscedasticity

We will discuss four solutions to heteroscedasticity in linear and nonlinear regression: variance-stabilizing transformations, weighted least squares, robust standard errors, and models that do not assume homoscedasticity.

Variance-Stabilizing Transformations

The idea is to use $Y_i^* = g(Y_i)$ instead of Y_i as the response variable, where g is a *variance-stabilizing transformation*.

Example: Consider again the cancer survival time data.

```
library(Stat2Data)
data(CancerSurvival)
CancerSurvival$Organ <- with(CancerSurvival, reorder(Organ, Survival, mean))
p <- ggplot(CancerSurvival, aes(x = Organ, y = Survival)) +
   geom_jitter(height = 0, width = 0.25) +
   labs(y = "Survival Time (Days)") + theme_minimal()
plot(p)
```
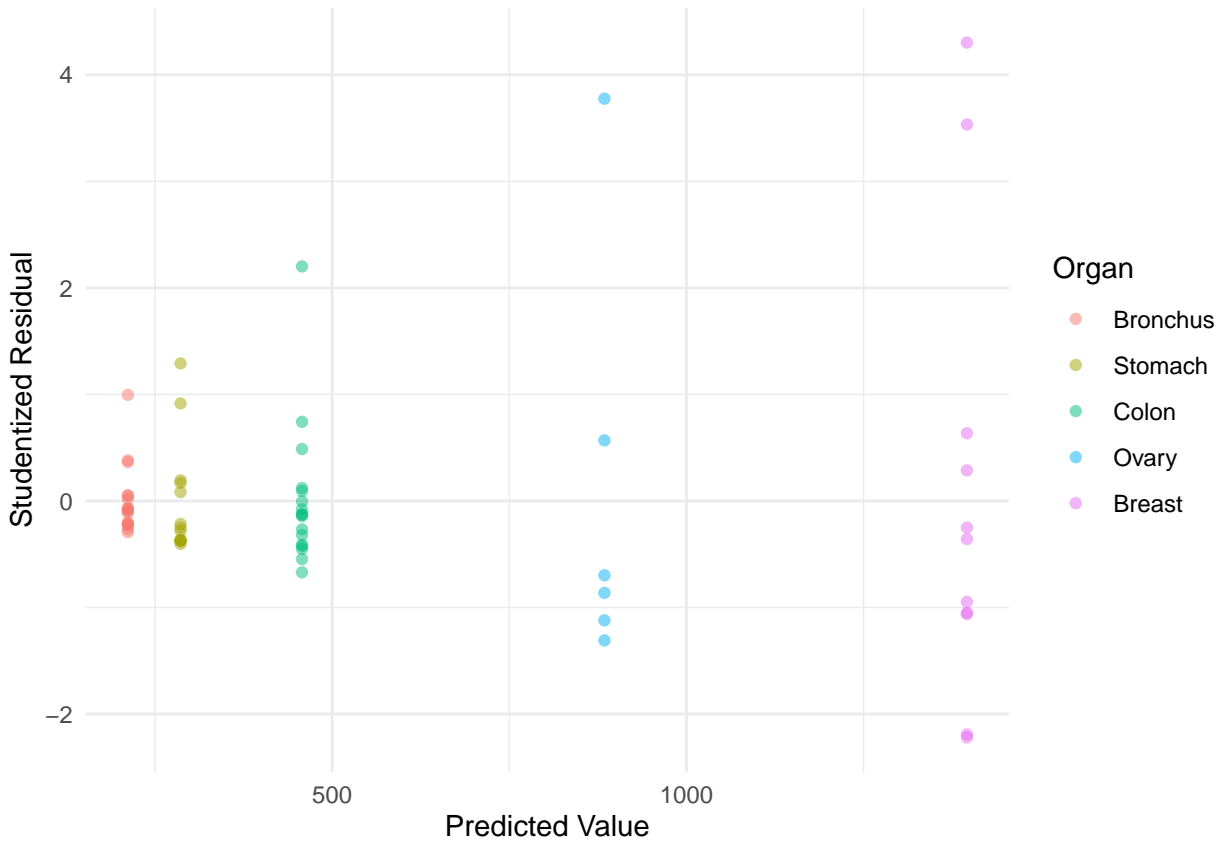
```
m <- lm(Survival ~ Organ, data = CancerSurvival)

CancerSurvival$yhat <- predict(m)
CancerSurvival$rest <- rstudent(m)

p <- ggplot(CancerSurvival, aes(x = yhat, y = rest, color = Organ)) +
  geom_point(alpha = 0.5) + theme_minimal() +
  labs(x = "Predicted Value", y = "Studentized Residual")
plot(p)
```
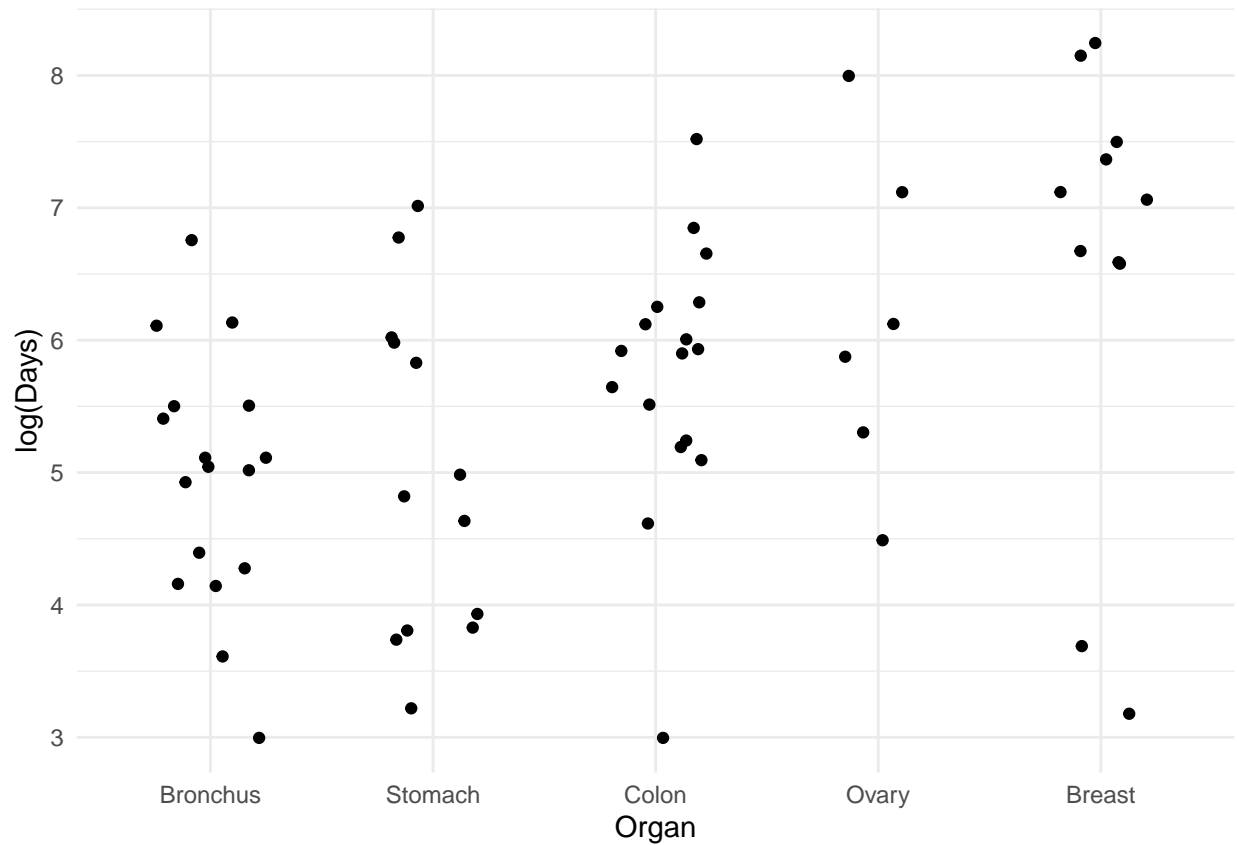

A model for *log* time might exhibit something closer to homoscedasticity.

```
p <- ggplot(CancerSurvival, aes(x = Organ, y = log(Survival))) +
  geom_jitter(height = 0, width = 0.25) +
  labs(y = "log(Days)") + theme_minimal()
plot(p)
```
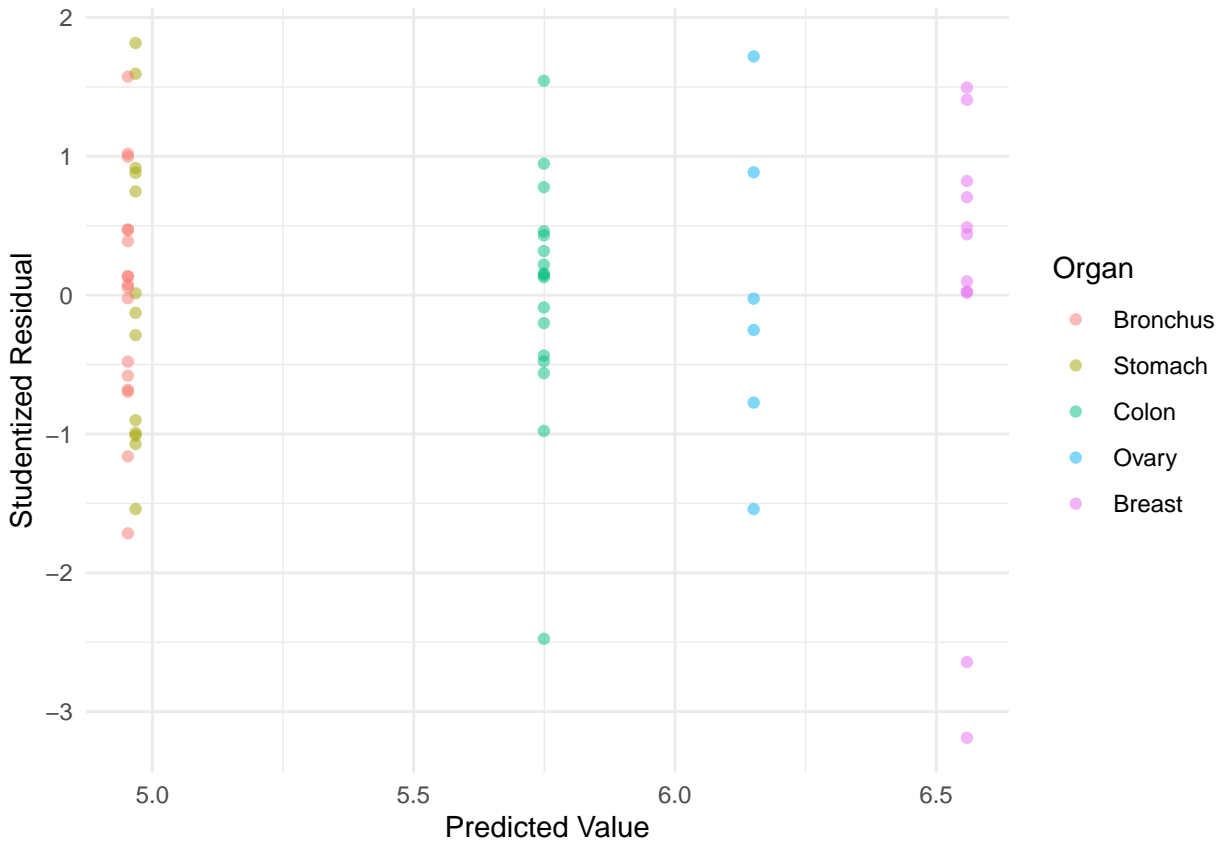
```
m <- lm(log(Survival) ~ Organ, data = CancerSurvival)

CancerSurvival$yhat <- predict(m)
CancerSurvival$rest <- rstudent(m)

p <- ggplot(CancerSurvival, aes(x = yhat, y = rest, color = Organ)) +
    geom_point(alpha = 0.5) + theme_minimal() +
    labs(x = "Predicted Value", y = "Studentized Residual")
plot(p)
```

Comments on variance-stabilizing transformations.

1. Depending on the situation, other transformations may exhibit variance-stabilizing properties. Some common transformations are $\sqrt{Y_i}$, $\log Y_i$, $1/\sqrt{Y_i}$ and $1/Y_i$ for right-skewed response variables, and $n_i \arcsin(\sqrt{Y_i})$ when Y_i is a proportion with a denominator of n_i.

2. A limitation of variance stabilizing transformations is that it is often difficult (and undesirable) to to *interpret* the model in terms of the transformed response variable (although there are exceptions as we will later see with the log transformation in the context of accelerated failure time models for survival data).

3. It is important to note that for any *nonlinear* transformation that $E[g(Y)] \neq g[E(Y)]$ (i.e., the expected transformed response does not necessarily equal the transformed expected response). For example, $E[\log Y] \neq \log E(Y)$. So we cannot obtain inferences for the expected response by applying the inverse of the transformation function. For example, $\exp[E(\log Y)] \neq E(Y)$.

Weighted Least Squares

A *weighted* least squares (WLS) estimator of the regression model parameters minimizes

$$\sum_{i=1}^{n} w_i (y_i - \hat{y}_i)^2,$$

were $w_i > 0$ is the *weight* for the i-th observation. So-called *ordinary least squares* (OLS) or *unweighted least squares* is a special case where all $w_i = 1$.

To account for heteroscedasticity, the weights should be *inversely proportional to the variance of the response* so that

$$w_i \propto \frac{1}{\text{Var}(Y_i)}.$$

Estimation is *efficient* meaning that the *true* standard errors (which are not necessarily the *reported* standard errors shown by software since these are estimates and may be biased without using weights as defined above) are as small as they can be when using weighted least squares.

Example: Consider the following data.

```r
turkeys <- data.frame(
  weight = c(674, 764, 795, 796, 826, 782, 834, 836, 830),
  pens = c(10, 5, 2, 2, 5, 5, 2, 2, 5),
  dosea = c(c(0, 0.12, 0.22, 0.32, 0.44), rep(0, 4)),
  doseb = c(rep(0, 5), c(0.12, 0.22, 0.32, 0.44))
)
turkeys
```

```
  weight pens dosea doseb
1    674   10  0.00  0.00
2    764    5  0.12  0.00
3    795    2  0.22  0.00
4    796    2  0.32  0.00
5    826    5  0.44  0.00
6    782    5  0.00  0.12
7    834    2  0.00  0.22
8    836    2  0.00  0.32
9    830    5  0.00  0.44
```
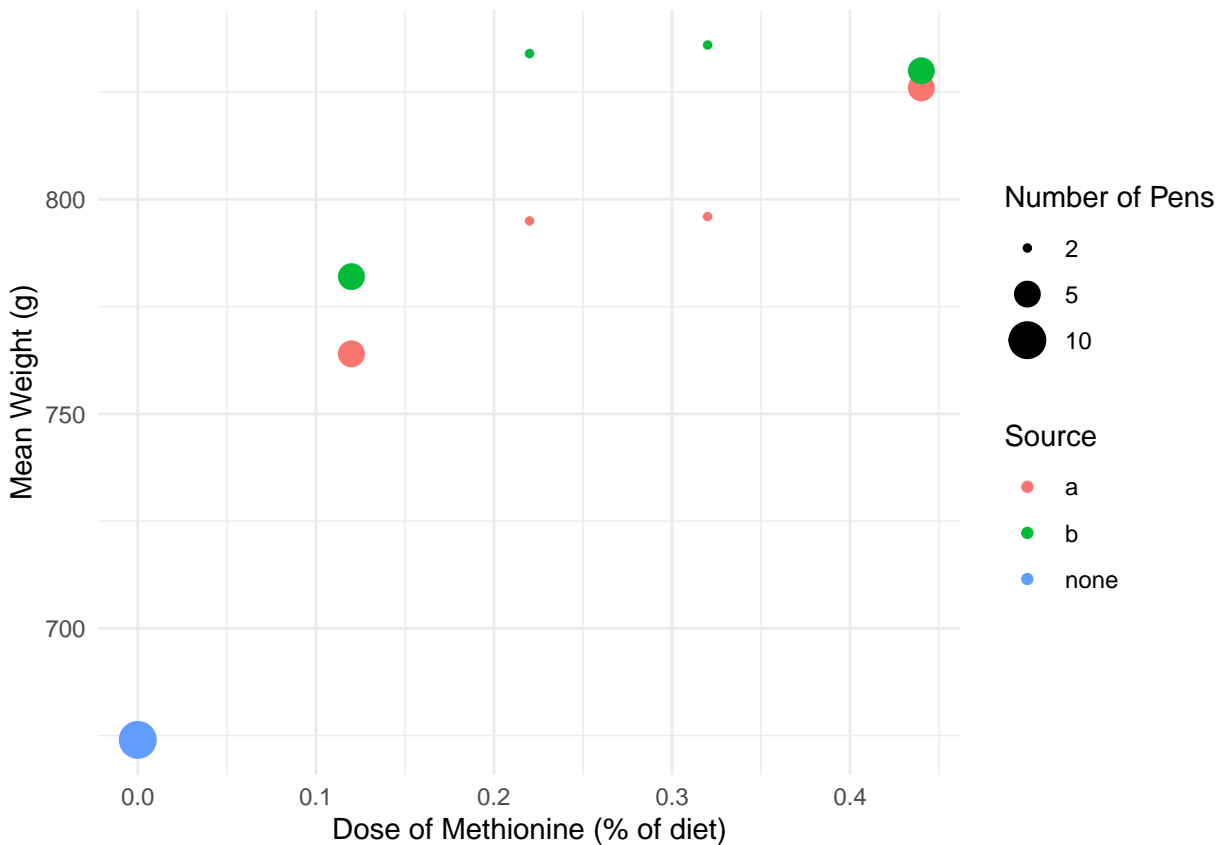
For plotting and modeling convenience we will rearrange the data a bit.

```r
library(dplyr)
turkeys <- turkeys |>
  mutate(dose = dosea + doseb) |>
  mutate(source = case_when(
    dose == 0 ~ "none",
    dosea > 0 ~ "a",
    doseb > 0 ~ "b")
  ) |>
  select(-dosea, -doseb)
turkeys
```

```
  weight pens dose source
1    674   10 0.00   none
2    764    5 0.12      a
3    795    2 0.22      a
4    796    2 0.32      a
5    826    5 0.44      a
6    782    5 0.12      b
7    834    2 0.22      b
8    836    2 0.32      b
9    830    5 0.44      b
```

```r
library(ggplot2)
p <- ggplot(turkeys, aes(x = dose, y = weight, color = source)) +
  theme_minimal() + geom_point(aes(size = pens)) +
  scale_size(breaks = c(2, 5, 10)) +
  labs(x = "Dose of Methionine (% of diet)", y = "Mean Weight (g)",
    color = "Source", size = "Number of Pens")
plot(p)
```

Suppose we want to estimate the following model.

$$E(W_i) = \begin{cases} \gamma, & \text{if no methionine was given,} \\ \alpha + (\gamma - \alpha)2^{-d_i/\lambda_a}, & \text{if methionine was given from source } a, \\ \alpha + (\gamma - \alpha)2^{-d_i/\lambda_b}, & \text{if methionine was given from source } b. \end{cases}$$

Note that this is like the Von Bertalanffy growth model. Here α is an asymptote for expected weight that we approach as dose increases, γ is the expected response for a zero dose , and λ_a and λ_b are "half-life" parameters. We can estimate this model as follows using *ordinary* (i.e., unweighted) least squares as follows.

```
m.ols <- nls(weight ~ case_when(
  source == "none" ~ gamma,
  source == "a" ~ alpha + (gamma - alpha) * 2^(-dose/lambdaa),
  source == "b" ~ alpha + (gamma - alpha) * 2^(-dose/lambdab)),
  start = list(alpha = 825, gamma = 675, lambdaa = 0.1, lambdab = 0.1),
  data = turkeys)
summary(m.ols)$coefficients
```
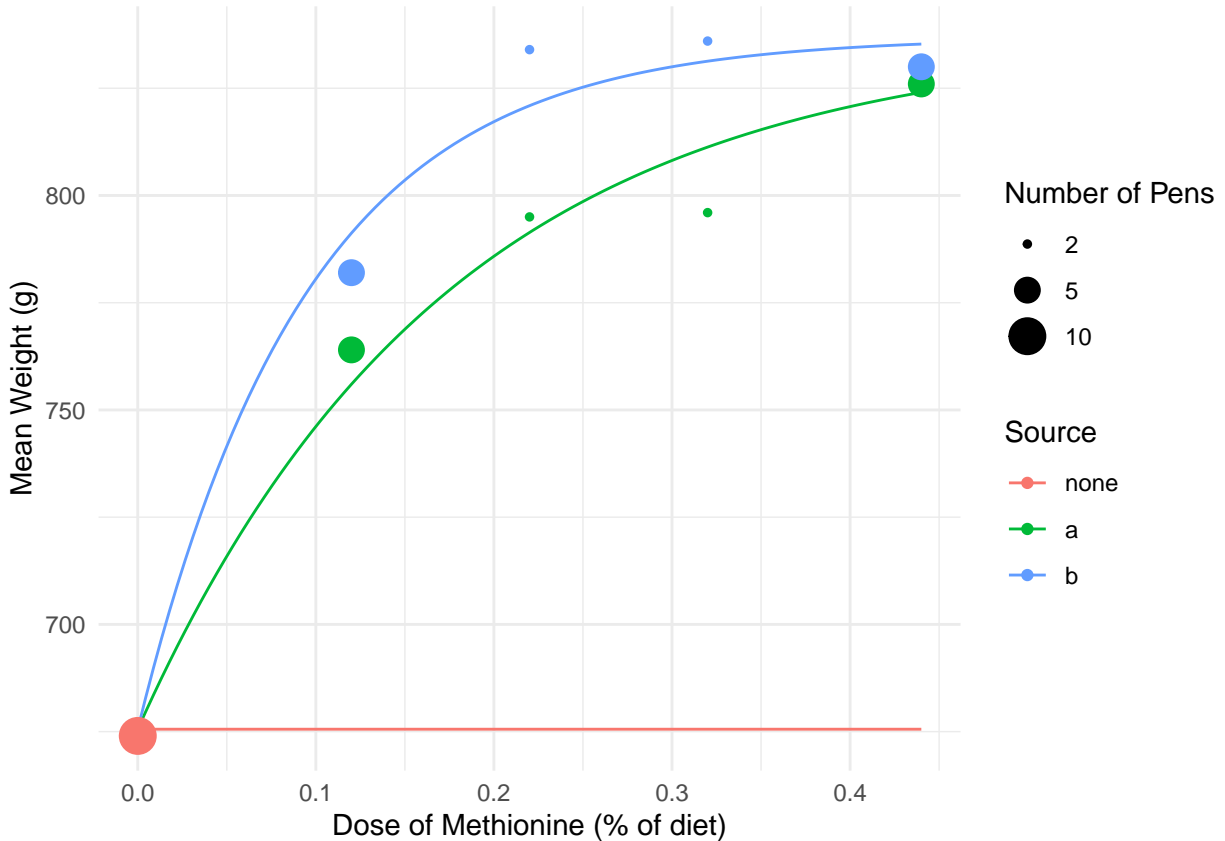
```
          Estimate Std. Error t value  Pr(>|t|)
alpha    836.87362    8.87663  94.278 2.545e-09
gamma    675.56291   10.92399  61.842 2.092e-08
lambdaa    0.12049    0.02250   5.355 3.051e-03
lambdab    0.06589    0.01601   4.115 9.222e-03
```

```
d <- expand.grid(source = c("none","a","b"), dose = seq(0, 0.44, length = 100))
d$yhat <- predict(m.ols, newdata = d)
p <- ggplot(turkeys, aes(x = dose, y = weight, color = source)) +
  geom_line(aes(y = yhat), data = d) +
```

```
    theme_minimal() + geom_point(aes(size = pens)) +
    scale_size(breaks = c(2, 5, 10)) +
    labs(x = "Dose of Methionine (% of diet)", y = "Mean Weight (g)",
      color = "Source", size = "Number of Pens")
plot(p)
```


The response variable is an *mean* of several observations so that

$$Y_i = \frac{Z_{i1} + Z_{i2} + \cdots + Z_{in_i}}{n_i}$$

where Z_{ij} is the length of the j-th pen that goes into the i-th average, and a total of n_i pens go into the i-th average. If $\text{Var}(Z_{ij}) = \sigma^2$ then $\text{Var}(Y_i) = \sigma^2/n_i$. Thus the weights should be

$$w_i \propto \frac{1}{\sigma^2/n_i} = \frac{n_i}{\sigma^2}.$$

Assuming $1/\sigma^2$ is a constant for all observations, we can define the weights of the means as $w_i = n_i$. The weights can be specified in `lm` and `nls` (and other functions for regression) using the `weights` argument.

```
m.wls <- nls(weight ~ case_when(
  source == "none" ~ gamma,
  source == "a" ~ alpha + (gamma - alpha) * 2^(-dose/lambdaa),
  source == "b" ~ alpha + (gamma - alpha) * 2^(-dose/lambdab)),
  start = list(alpha = 825, gamma = 675, lambdaa = 0.1, lambdab = 0.1),
  data = turkeys, weights = pens)
summary(m.wls)$coefficients
```

```
        Estimate Std. Error t value  Pr(>|t|)
```
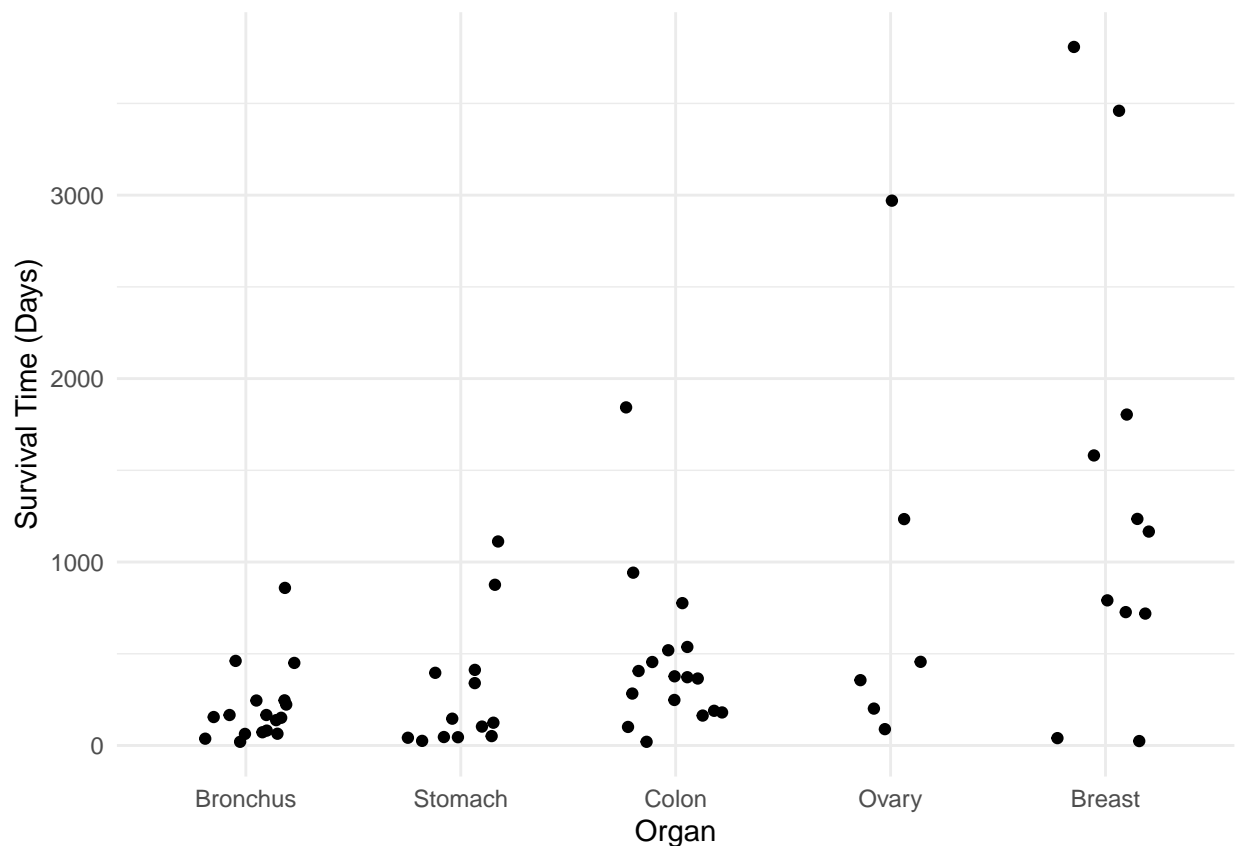
```
alpha    834.78331      6.7761 123.195 6.684e-10
gamma    674.30393      5.5420 121.671 7.113e-10
lambdaa    0.10997      0.0163   6.746 1.086e-03
lambdab    0.06857      0.0117   5.860 2.051e-03
```

```
summary(m.ols)$coefficients
```

```
          Estimate Std. Error t value  Pr(>|t|)
alpha    836.87362    8.87663  94.278 2.545e-09
gamma    675.56291   10.92399  61.842 2.092e-08
lambdaa    0.12049    0.02250   5.355 3.051e-03
lambdab    0.06589    0.01601   4.115 9.222e-03
```

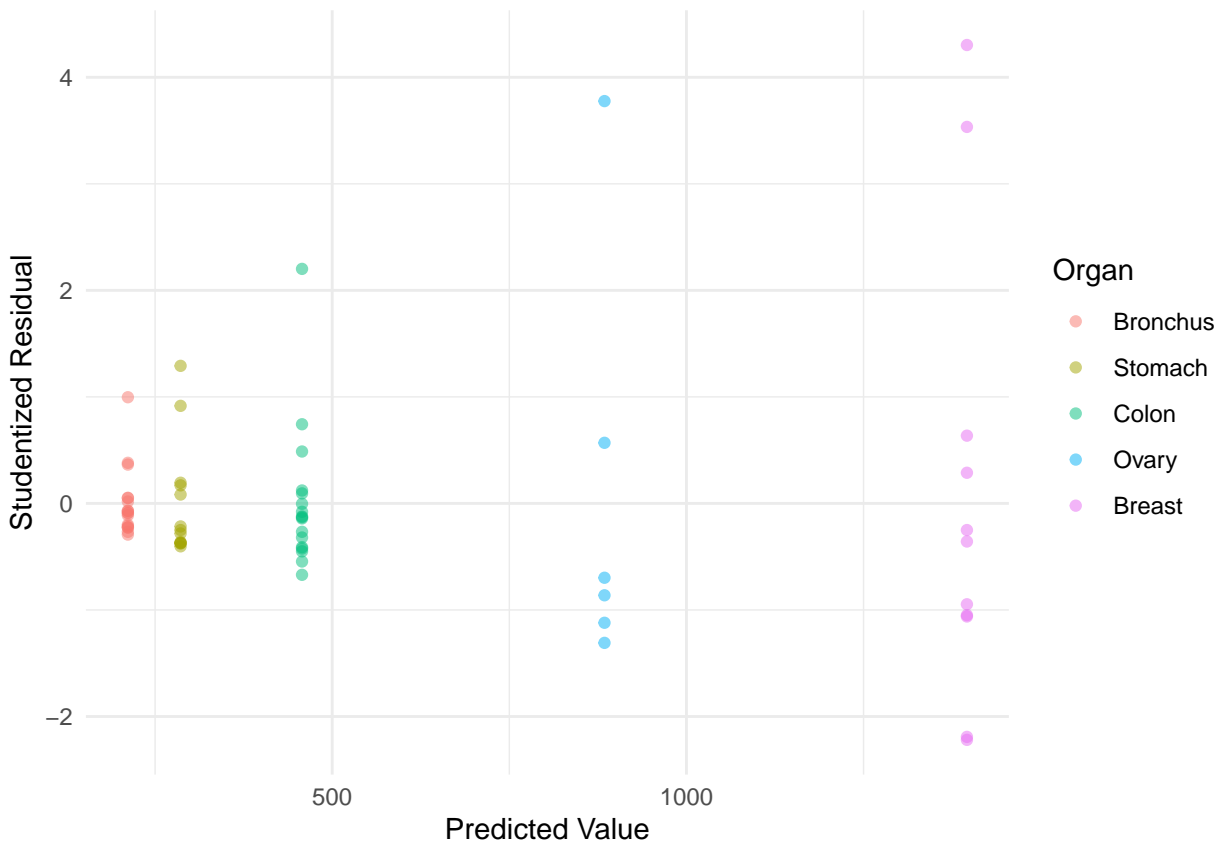**Example**: Consider again the cancer survival time data.

```
p <- ggplot(CancerSurvival, aes(x = Organ, y = Survival)) +
   geom_jitter(height = 0, width = 0.25) +
   labs(y = "Survival Time (Days)") + theme_minimal()
plot(p)
```



```
m.ols <- lm(Survival ~ Organ, data = CancerSurvival)

CancerSurvival$yhat <- predict(m.ols)
CancerSurvival$rest <- rstudent(m.ols)

p <- ggplot(CancerSurvival, aes(x = yhat, y = rest, color = Organ)) +
   geom_point(alpha = 0.5) + theme_minimal() +
   labs(x = "Predicted Value", y = "Studentized Residual")
plot(p)
```

8

There are a couple of ways we could go with these data. One is that since we have a categorical explanatory variable with multiple observations per category, we could *estimate* the variance of Y_i of each organ, and then set the weights to the reciprocals of these estimated variances.

```
library(dplyr)
CancerSurvival |> group_by(Organ) |>
  summarize(variance = var(Survival), weight = 1/var(Survival))
```

```
# A tibble: 5 x 3
  Organ    variance       weight
  <fct>       <dbl>        <dbl>
1 Bronchus   44041. 0.0000227
2 Stomach   119930. 0.00000834
3 Colon     182473. 0.00000548
4 Ovary    1206875. 0.000000829
5 Breast   1535038. 0.000000651
```

We can use the following to compute weights and add them to the data frame.

```
CancerSurvival <- CancerSurvival |>
   group_by(Organ) |> mutate(w = 1/var(Survival))
head(CancerSurvival)
```

```
# A tibble: 6 x 3
# Groups:   Organ [1]
  Survival Organ            w
     <int> <fct>         <dbl>
1      124 Stomach 0.00000834
2       42 Stomach 0.00000834
```

```
3           25 Stomach 0.00000834
4           45 Stomach 0.00000834
5          412 Stomach 0.00000834
6           51 Stomach 0.00000834
```

Now let's estimate the model using weighted least squares with these weights and inspect the residuals.
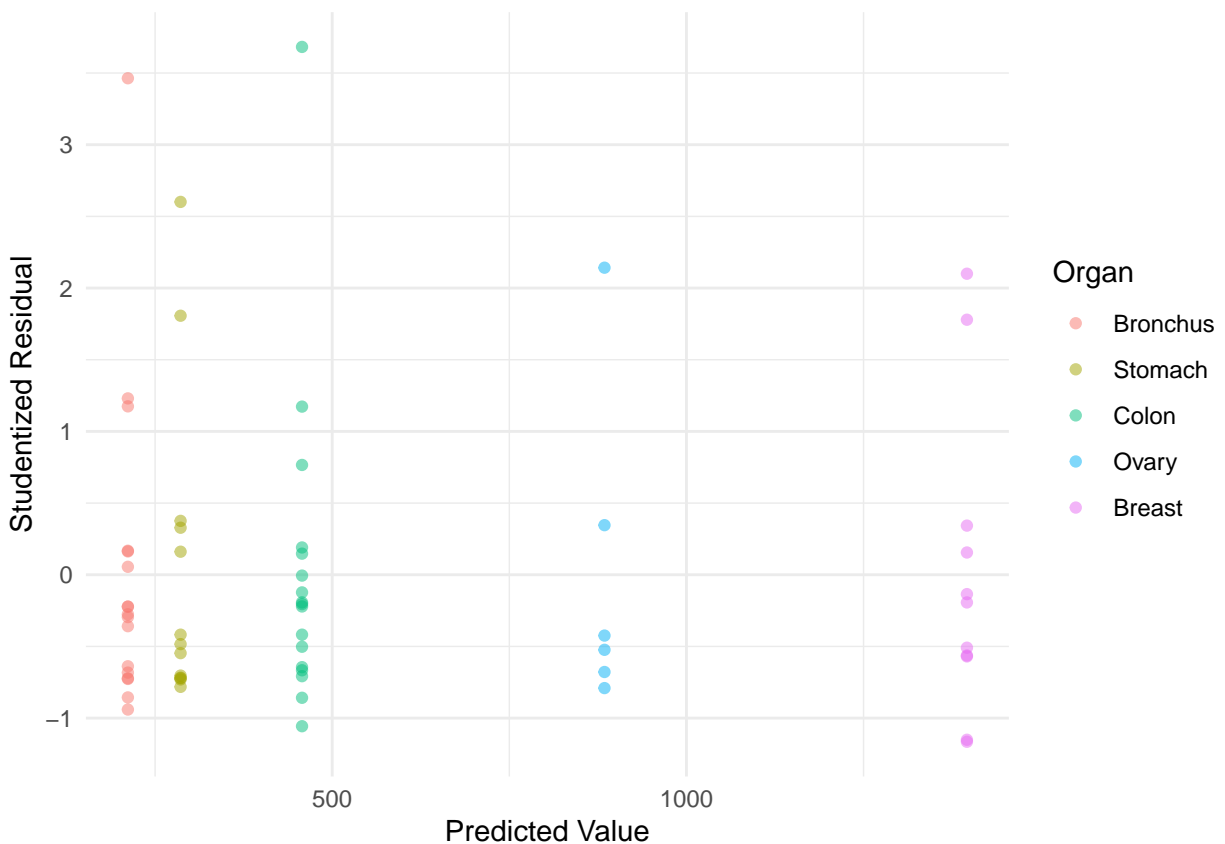
```
m.wls <- lm(Survival ~ Organ, weights = w, data = CancerSurvival)

CancerSurvival$yhat <- predict(m.wls)
CancerSurvival$rest <- rstudent(m.wls)

p <- ggplot(CancerSurvival, aes(x = yhat, y = rest, color = Organ)) +
    geom_point(alpha = 0.5) + theme_minimal() +
    labs(x = "Predicted Value", y = "Studentized Residual")
plot(p)
```


Note how this affects our inferences.

```
cbind(summary(m.ols)$coefficients, confint(m.ols))
```

```
             Estimate Std. Error t value   Pr(>|t|)     2.5 % 97.5 %
(Intercept)    211.59      162.4  1.3030 0.1976373 -113.34   536.5
OrganStomach    74.41      246.7  0.3017 0.7639784 -419.20   568.0
OrganColon     245.82      229.6  1.0704 0.2887820 -213.70   705.3
OrganOvary     672.75      317.9  2.1160 0.0385749   36.56 1308.9
OrganBreast   1184.32      259.1  4.5713 0.0000253  665.91 1702.7
```

```r
cbind(summary(m.wls)$coefficients, confint(m.wls))
```

```
              Estimate Std. Error t value  Pr(>|t|)   2.5 %  97.5 %
(Intercept)     211.59       50.9  4.1571 0.0001057  109.74   313.4
OrganStomach     74.41      108.7  0.6846 0.4963078 -143.10   291.9
OrganColon      245.82      115.4  2.1296 0.0373858   14.85   476.8
OrganOvary      672.75      451.4  1.4904 0.1414343 -230.45  1575.9
OrganBreast    1184.32      377.0  3.1413 0.0026291  429.92  1938.7
```

```r
organs <- unique(CancerSurvival$Organ)
trtools::contrast(m.ols, a = list(Organ = organs), cnames = organs)
```

```
         estimate     se   lower  upper tvalue df    pvalue
Stomach     286.0  185.7  -85.57  657.6  1.540 59 1.289e-01
Bronchus    211.6  162.4 -113.34  536.5  1.303 59 1.976e-01
Colon       457.4  162.4  132.48  782.3  2.817 59 6.587e-03
Ovary       884.3  273.3  337.39 1431.3  3.235 59 1.993e-03
Breast     1395.9  201.9  991.96 1799.9  6.915 59 3.770e-09
```

```r
trtools::contrast(m.wls, a = list(Organ = organs), cnames = organs)
```

```
         estimate     se  lower  upper tvalue df    pvalue
Stomach     286.0  96.05  93.81  478.2  2.978 59 0.0042091
Bronchus    211.6  50.90 109.74  313.4  4.157 59 0.0001057
Colon       457.4 103.60 250.10  664.7  4.415 59 0.0000437
Ovary       884.3 448.49 -13.10 1781.8  1.972 59 0.0533281
Breast     1395.9 373.56 648.41 2143.4  3.737 59 0.0004228
```

```r
trtools::contrast(m.ols,
  a = list(Organ = "Breast"),
  b = list(Organ = c("Bronchus","Stomach","Colon","Ovary")),
  cnames = c("Breast vs Bronchus", "Breast vs Stomach",
    "Breast vs Colon","Breast vs Ovary"))
```

```
                   estimate    se  lower upper tvalue df    pvalue
Breast vs Bronchus   1184.3 259.1  665.9  1703  4.571 59 0.0000253
Breast vs Stomach    1109.9 274.3  561.1  1659  4.046 59 0.0001533
Breast vs Colon       938.5 259.1  420.1  1457  3.622 59 0.0006083
Breast vs Ovary       511.6 339.8 -168.4  1192  1.506 59 0.1375263
```

```r
trtools::contrast(m.wls,
  a = list(Organ = "Breast"),
  b = list(Organ = c("Bronchus","Stomach","Colon","Ovary")),
  cnames = c("Breast vs Bronchus", "Breast vs Stomach",
    "Breast vs Colon","Breast vs Ovary"))
```

```
                   estimate    se  lower upper tvalue df   pvalue
Breast vs Bronchus   1184.3 377.0  429.9  1939 3.1413 59 0.002629
Breast vs Stomach    1109.9 385.7  338.1  1882 2.8776 59 0.005572
Breast vs Colon       938.5 387.7  162.8  1714 2.4209 59 0.018577
Breast vs Ovary       511.6 583.7 -656.4  1680 0.8765 59 0.384340
```

Here's how you can do the comparison of one level with all others using the `contrast` function from the **emmeans** package.

```r
library(emmeans)
emmeans::contrast(emmeans(m.wls, ~ Organ), method = "trt.vs.ctrl", ref = "Breast",
```

```
   reverse = TRUE, adjust = "none", infer = TRUE)
```

```
contrast             estimate  SE df lower.CL upper.CL t.ratio p.value
Breast - Bronchus        1184 377 59      430     1939   3.141  0.0026
Breast - Stomach         1110 386 59      338     1882   2.878  0.0056
Breast - Colon            938 388 59      163     1714   2.421  0.0186
Breast - Ovary            512 584 59     -656     1680   0.876  0.3843
```

```
Confidence level used: 0.95
```

Another approach is to assume that the variance of the response variable is some function of its expected response, and thus the weights are a function of the expected response. With right-skewed response variables one common functional relationship is that

$$\text{Var}(Y_i) \propto E(Y_i),$$

or, more generally,

$$\text{Var}(Y_i) \propto E(Y_i)^p,$$

where p is some power (usually $p \geq 1$). So the weights would then be

$$w_i \propto \frac{1}{E(Y_i)^p}.$$

We do not know $E(Y_i)$, but $\hat{y}_i$ is an estimate of $E(Y_i)$. But we need the weights to compute $\hat{y}_i$!

Two situations:

1. Estimates of the model parameters and thus $\hat{y}_i$ *do not* depend on the weights.[1] Here we can compute the weights with an initial regression model without weights.

2. Estimates of the model parameters and thus $\hat{y}_i$ *do* depend on the weights. An approach we can use here is *iteratively weighted least squares*.

It can be shown that $\hat{y}_i$ does not depend on the weights for the model for the `CancerSurvival` model. We can use the estimates from ordinary least squares to obtain weights of $w_i = 1/\hat{y}_i^p$.
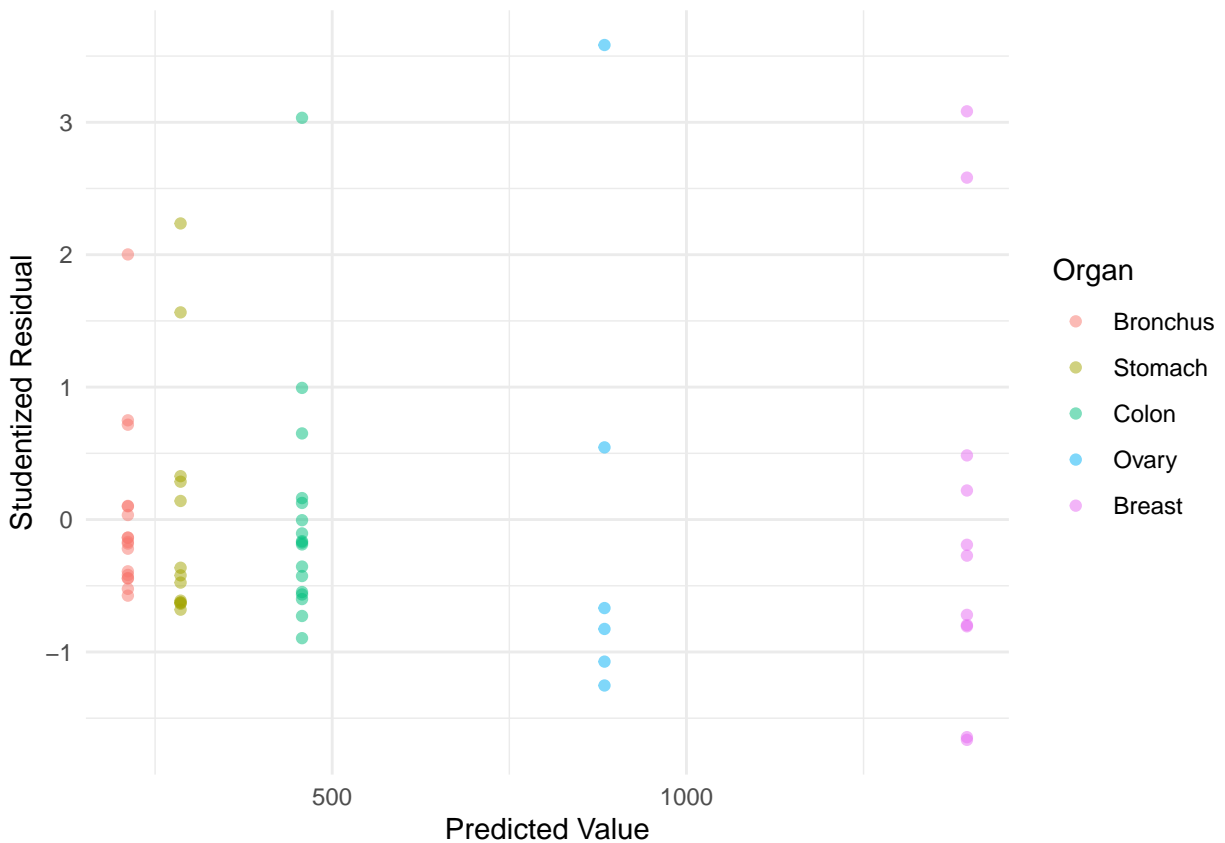
```
m.ols <- lm(Survival ~ Organ, data = CancerSurvival)

CancerSurvival$w <- 1/predict(m.ols)
m.wls <- lm(Survival ~ Organ, data = CancerSurvival, weights = w)

CancerSurvival$yhat <- predict(m.wls)
CancerSurvival$rest <- rstudent(m.wls)

p <- ggplot(CancerSurvival, aes(x = yhat, y = rest, color = Organ)) +
   geom_point(alpha = 0.5) + theme_minimal() +
   labs(x = "Predicted Value", y = "Studentized Residual")
plot(p)
```

---

[1]In practice the most common situation where this will happen is with models with only a single categorical explanatory variable (or models with multiple categorical explanatory variables that include interactions).
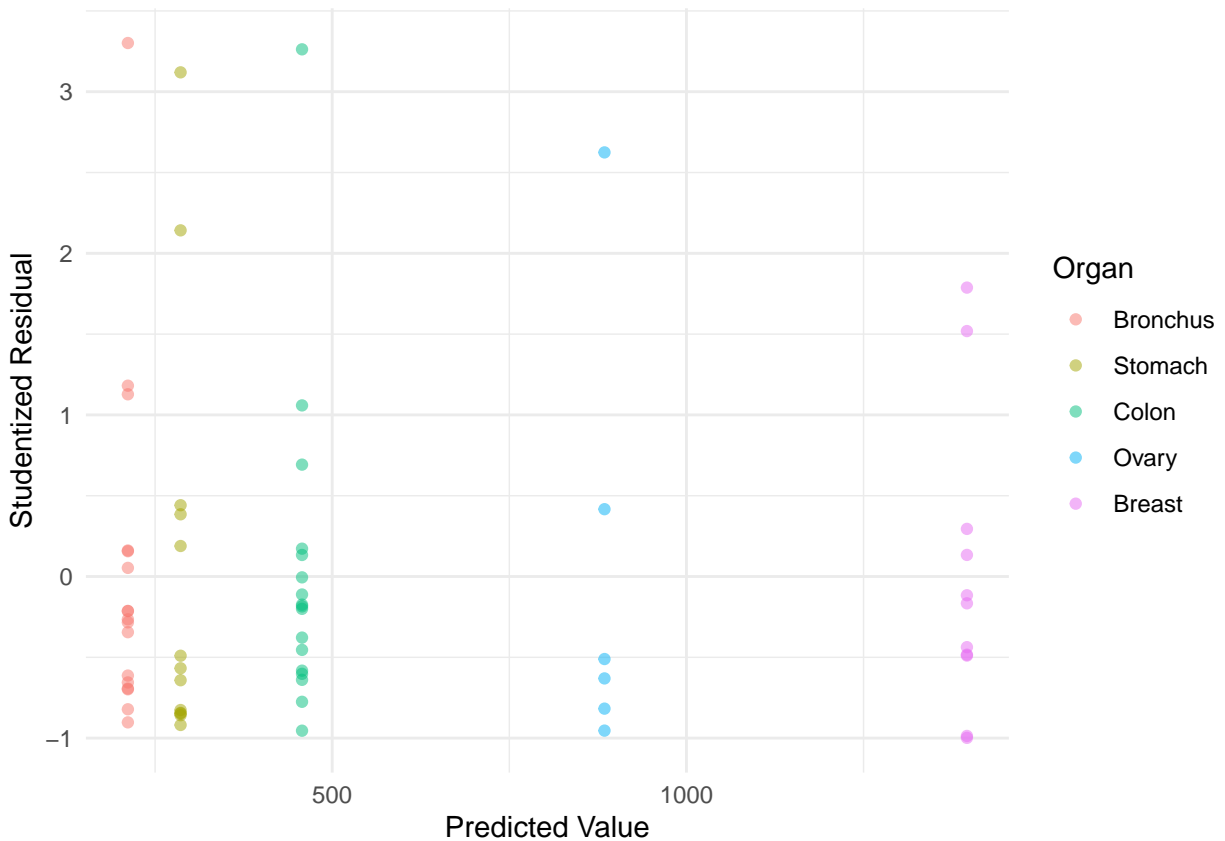
Maybe we could do better. Let's try $p = 2$ — i.e., $\mathrm{Var}(Y_i) \propto E(Y_i)^2$.

```r
m.ols <- lm(Survival ~ Organ, data = CancerSurvival)

CancerSurvival$w <- 1/predict(m.ols)^2
m.wls <- lm(Survival ~ Organ, data = CancerSurvival, weights = w)

CancerSurvival$yhat <- predict(m.wls)
CancerSurvival$rest <- rstudent(m.wls)

p <- ggplot(CancerSurvival, aes(x = yhat, y = rest, color = Organ)) +
    geom_point(alpha = 0.5) + theme_minimal() +
    labs(x = "Predicted Value", y = "Studentized Residual")
plot(p)
```

Example: Consider again following data from a study on the effects of fuel reduction on biomass.
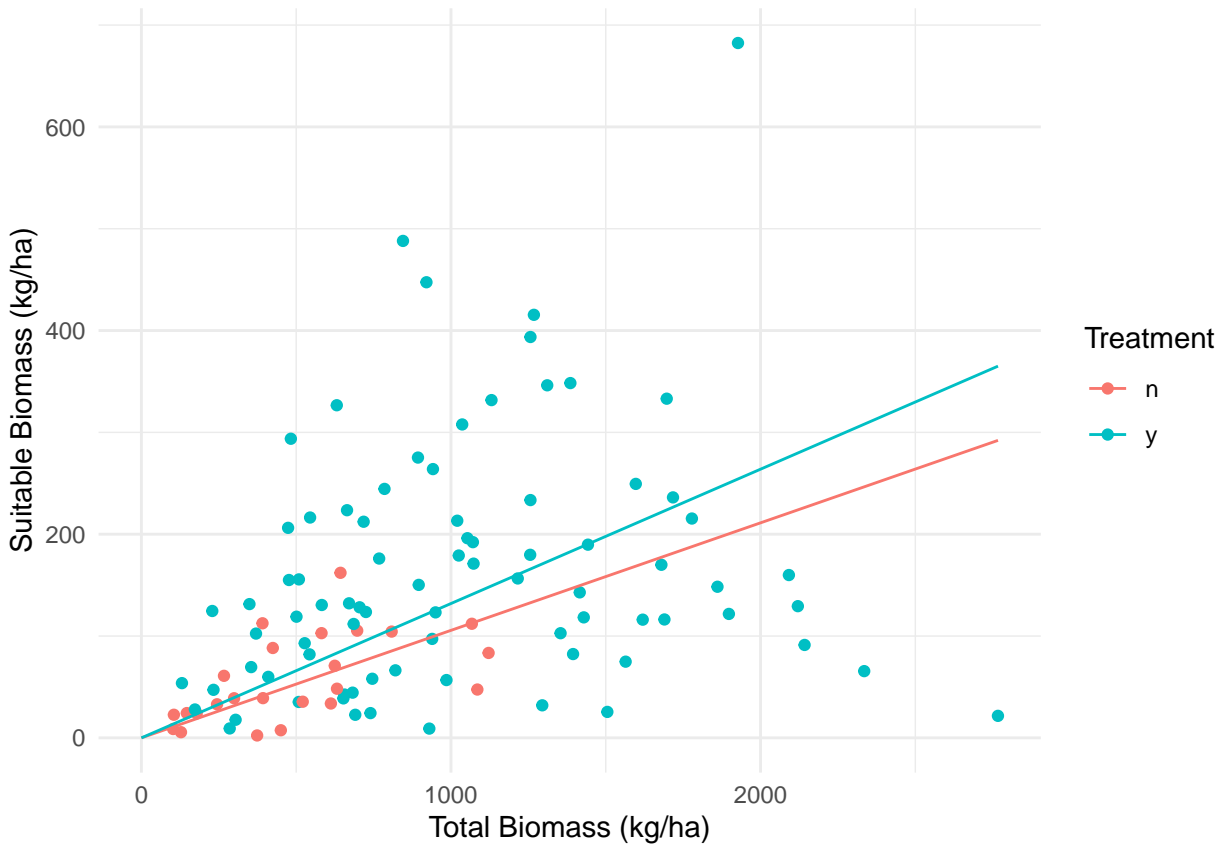
```r
library(trtools) # for biomass data

m <- lm(suitable ~ -1 + treatment:total, data = biomass)
summary(m)$coefficients
```

```
                 Estimate Std. Error t value Pr(>|t|)
treatmentn:total   0.1056    0.04183   2.524 1.31e-02
treatmenty:total   0.1319    0.01121  11.773 7.61e-21
```

```r
d <- expand.grid(treatment = c("n","y"), total = seq(0, 2767, length = 10))
d$yhat <- predict(m, newdata = d)

p <- ggplot(biomass, aes(x = total, y = suitable, color = treatment)) +
  geom_point() + geom_line(aes(y = yhat), data = d) + theme_minimal() +
  labs(x = "Total Biomass (kg/ha)",
    y = "Suitable Biomass (kg/ha)",
    color = "Treatment")
plot(p)
```

```
biomass$yhat <- predict(m)
biomass$rest <- rstudent(m)

p <- ggplot(biomass, aes(x = yhat, y = rest, color = treatment)) +
   geom_point() + theme_minimal() +
   labs(x = "Predicted Value",
      y = "Studentized Residual",
      color = "Treatment")
plot(p)
```

Here we might also assume that $\text{Var}(Y_i) \propto E(Y_i)^p$, with weights of $w_i = 1/\hat{y}_i$. But here things are a bit more complicated for this model: the w_i depend on the $\hat{y}_i$, the $\hat{y}_i$ depend on the w_i. In the model for the `CancerSurvival` data this was not an issue because there the estimates of the model parameters, and thus $\hat{y}_i$, did not depend on the weights so we could use ordinary least squares where all $w_i = 1$ to get the $\hat{y}_i$. But that is not true for this model. But we can solve this problem using *iteratively weighted least squares*.

```
biomass$w <- 1 # initial weights are all equal to one
for (i in 1:5) {
  m.wls <- lm(suitable ~ -1 + treatment:total, weights = w, data = biomass)
  print(coef(m.wls)) # optional
  biomass$w <- 1 / predict(m.wls)
}
```

```
treatmentn:total treatmenty:total
          0.1056           0.1319
treatmentn:total treatmenty:total
          0.1155           0.1578
treatmentn:total treatmenty:total
          0.1155           0.1578
treatmentn:total treatmenty:total
          0.1155           0.1578
treatmentn:total treatmenty:total
          0.1155           0.1578
```

```
summary(m)$coefficients
```

```
                 Estimate Std. Error t value Pr(>|t|)
treatmentn:total   0.1056    0.04183   2.524 1.31e-02
treatmenty:total   0.1319    0.01121  11.773 7.61e-21
```

```
summary(m.wls)$coefficients
```

```
                 Estimate Std. Error t value  Pr(>|t|)
treatmentn:total   0.1155    0.02887    4.00 1.189e-04
treatmenty:total   0.1578    0.01268   12.45 2.501e-22
```