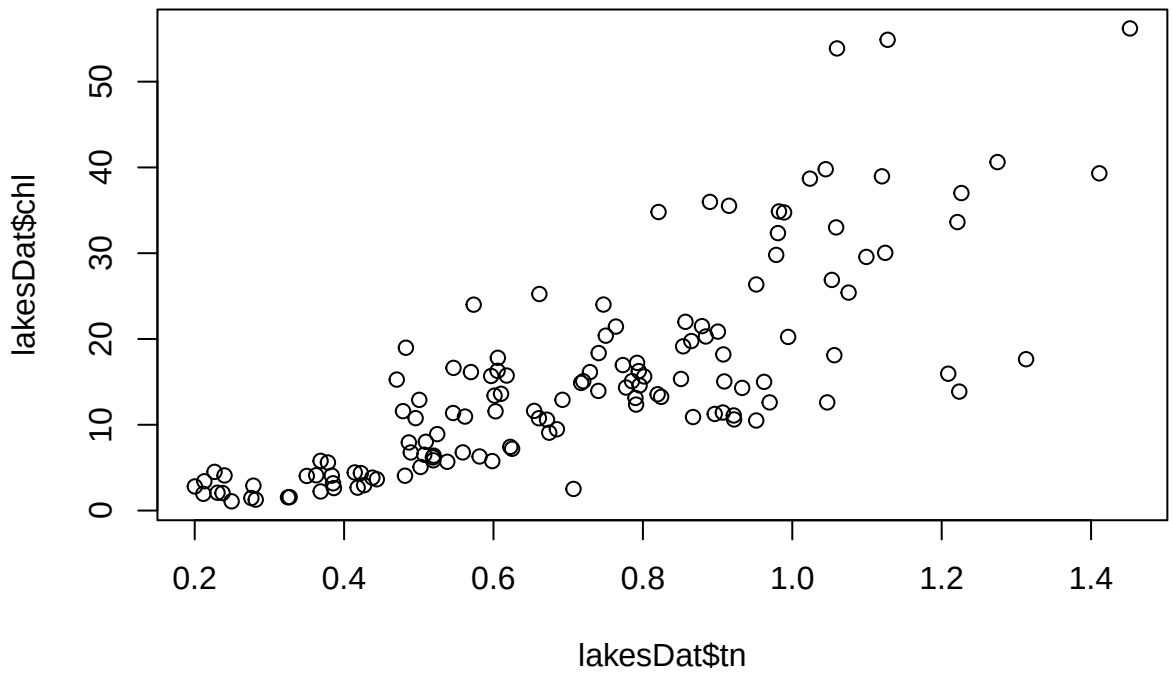
5200 Take-Home

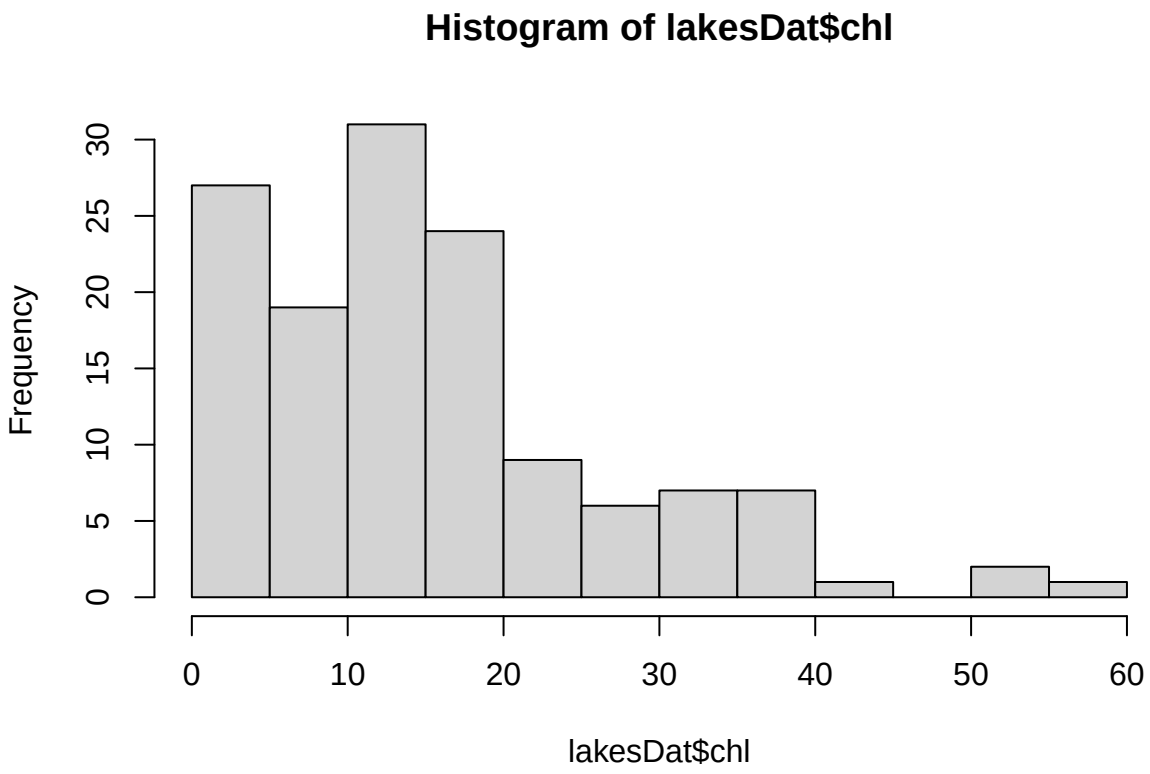
Sam Olson

Q1: CHL & TN (Plains vs. Ozarks)

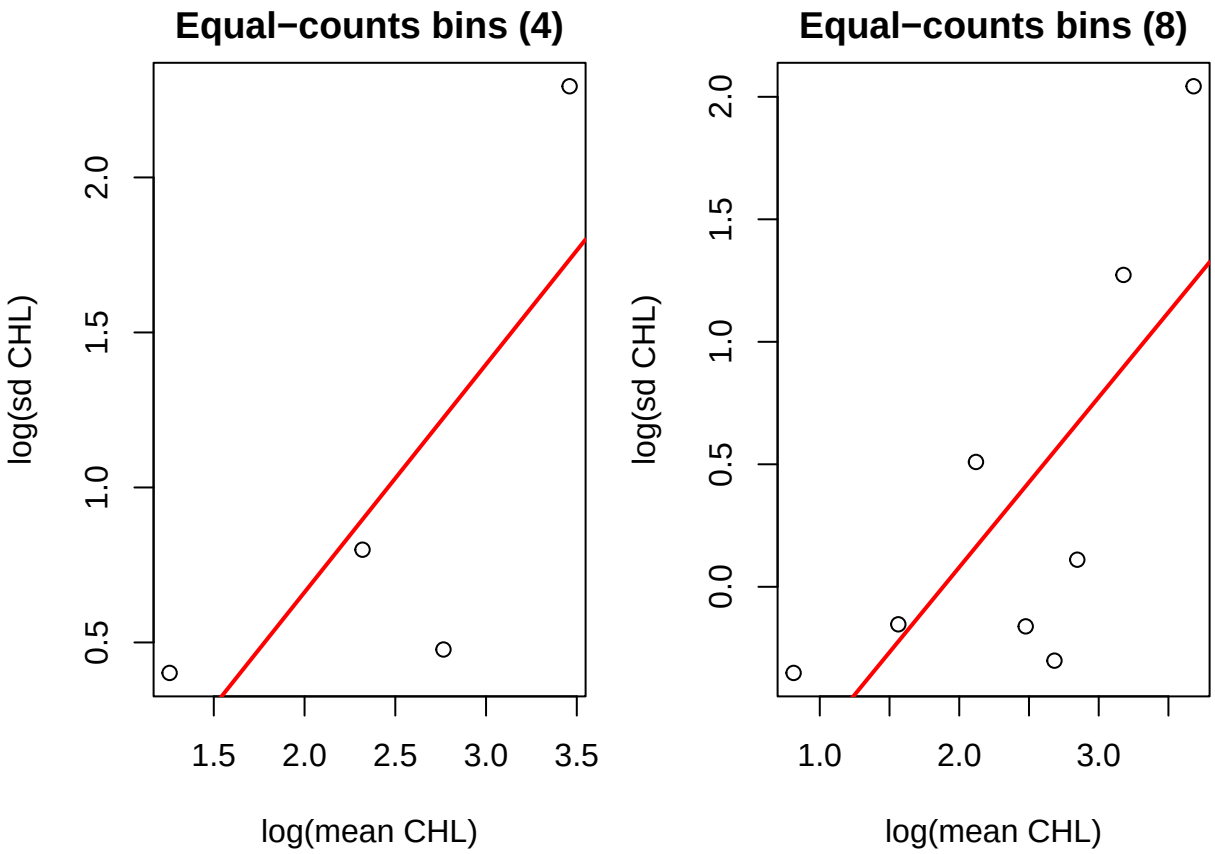
```
plot(y = lakesDat$chl, x = lakesDat$tn)
```



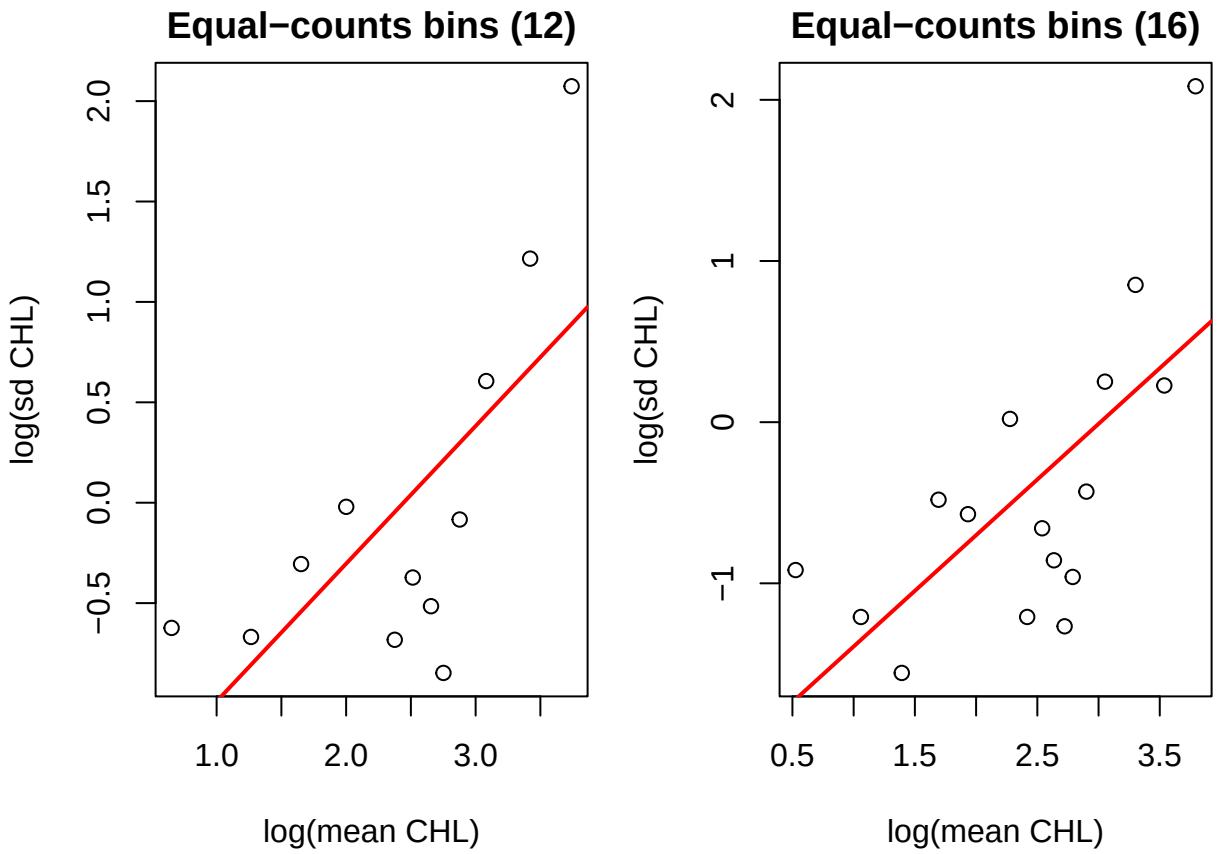

```
hist(lakesDat$chl)
```



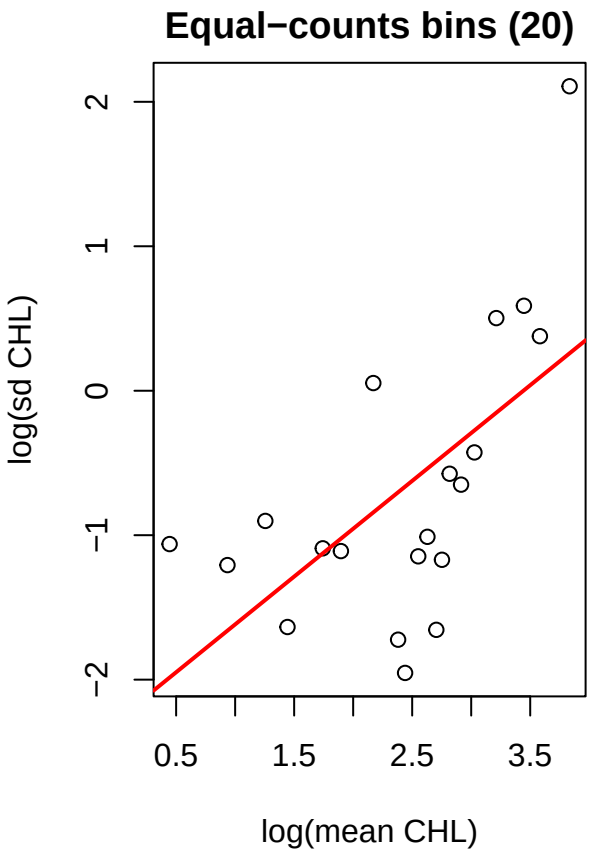

```
# Box Cox
# log(lakesDat$chl)

bc_plot <- function(y, nbins, main) {
  q <- quantile(y, probs = seq(0, 1, length.out = nbins + 1), na.rm = TRUE)
  cuts <- cut(y, breaks = q, include.lowest = TRUE)
  m <- tapply(y, cuts, mean, na.rm = TRUE)
  s <- tapply(y, cuts,  sd,   na.rm = TRUE)

  plot(x = log(m), y = log(s),
       xlab = "log(mean CHL)", ylab = "log(sd CHL)", main = main)
  fit <- lm(log(s) ~ log(m))
  abline(fit, col = "red", lwd = 2)
  invisible(fit)
}

op <- par(mfrow = c(1, 2), mar = c(4, 4, 2, 1))
fit4 <- bc_plot(lakesDat$chl, nbins = 4,  main = "Equal-counts bins (4)")
fit8 <- bc_plot(lakesDat$chl, nbins = 8, main = "Equal-counts bins (8)")
```

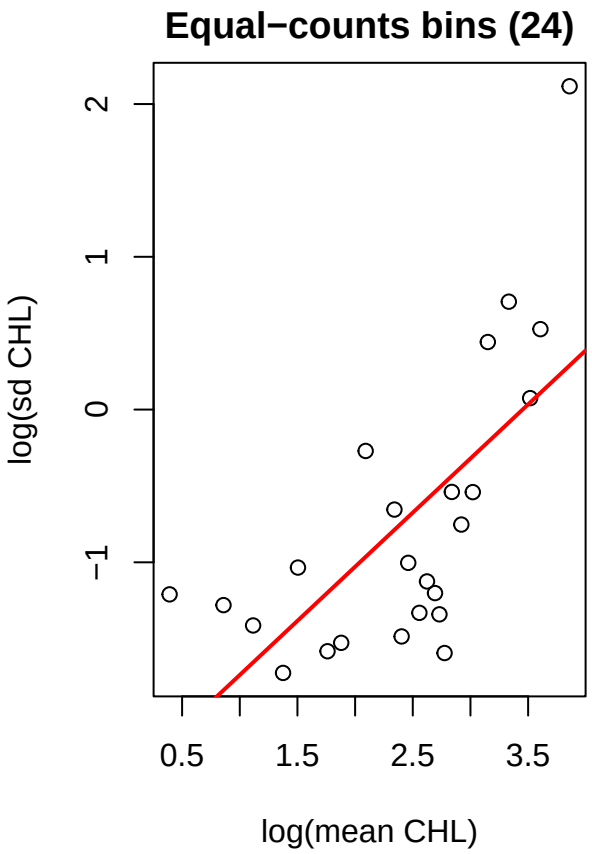



```
fit12 <- bc_plot(lakesDat$chl, nbins = 12,  main = "Equal-counts bins (12)")
fit16 <- bc_plot(lakesDat$chl, nbins = 16, main = "Equal-counts bins (16)")
```




```
fit20 <- bc_plot(lakesDat$chl, nbins = 20,  main = "Equal-counts bins (20)")
fit24 <- bc_plot(lakesDat$chl, nbins = 24, main = "Equal-counts bins (24)")
```






```r
par(op)

summ_tbl <- function(fit, nbins) {
  s <- summary(fit)
  ct <- coef(s)
  data.frame(
    Bins      = nbins,
    Term      = rownames(ct),
    Estimate  = round(ct[, 1], 4),
    SE        = round(ct[, 2], 4),
    t_value   = round(ct[, 3], 2),
    p_value   = formatC(ct[, 4], format = "e", digits = 2),
    R2        = round(s$r.squared, 3),
    Adj_R2    = round(s$adj.r.squared, 3),
    N         = s$df[1] + s$df[2] + 1,
    check.names = FALSE
  )
}

tab <- rbind(summ_tbl(fit4, 4), summ_tbl(fit8, 8), summ_tbl(fit12, 12), summ_tbl(fit16, 16), summ_tbl(fi

kbl(tab, booktabs = TRUE,
    caption = "Linear regressions for Box-Cox mean-sd plots (log(sd) ~ log(mean))") |>
  kable_styling(full_width = FALSE, position = "center")
```

```
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
```

```
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

```
## Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

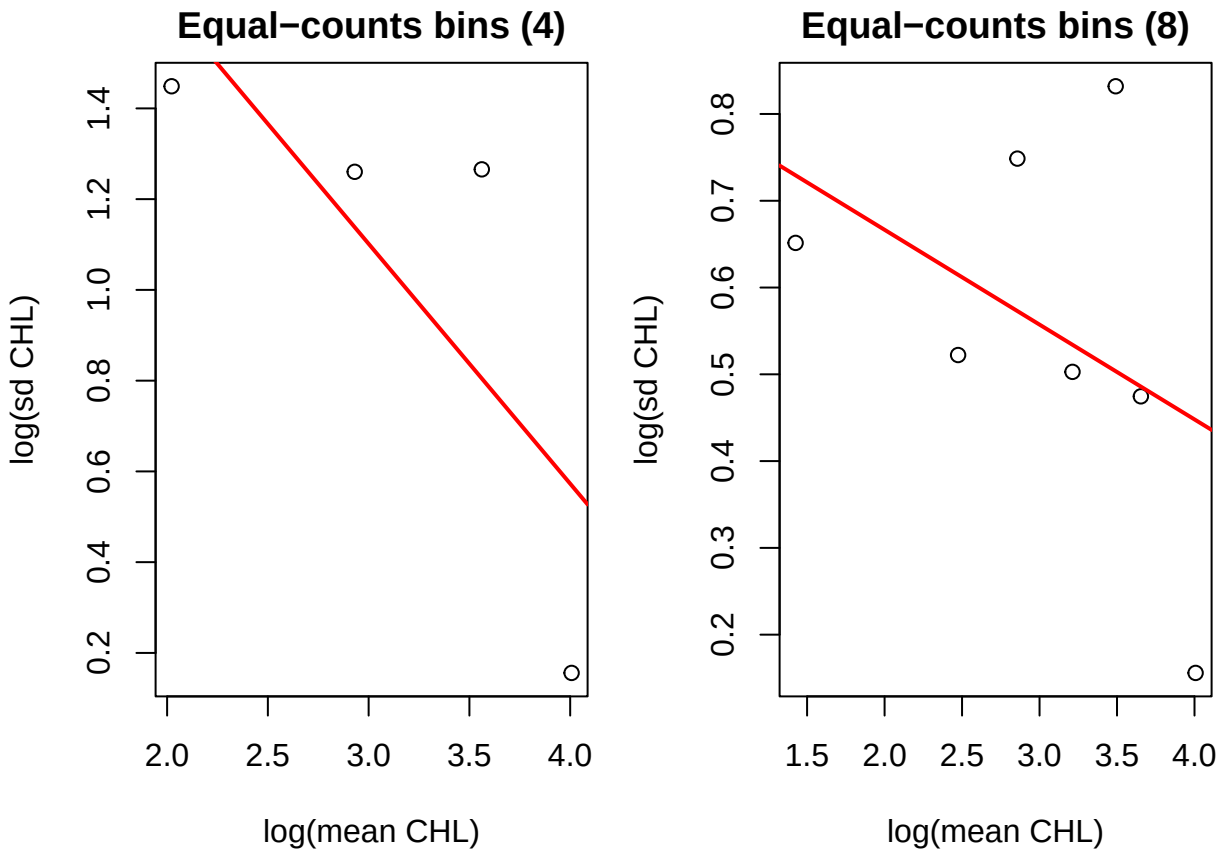
```
## Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

```
bc_plot_equal <- function(y, nbins, main) {
  # equal-width break points
  q <- seq(from = min(y, na.rm = TRUE),
```

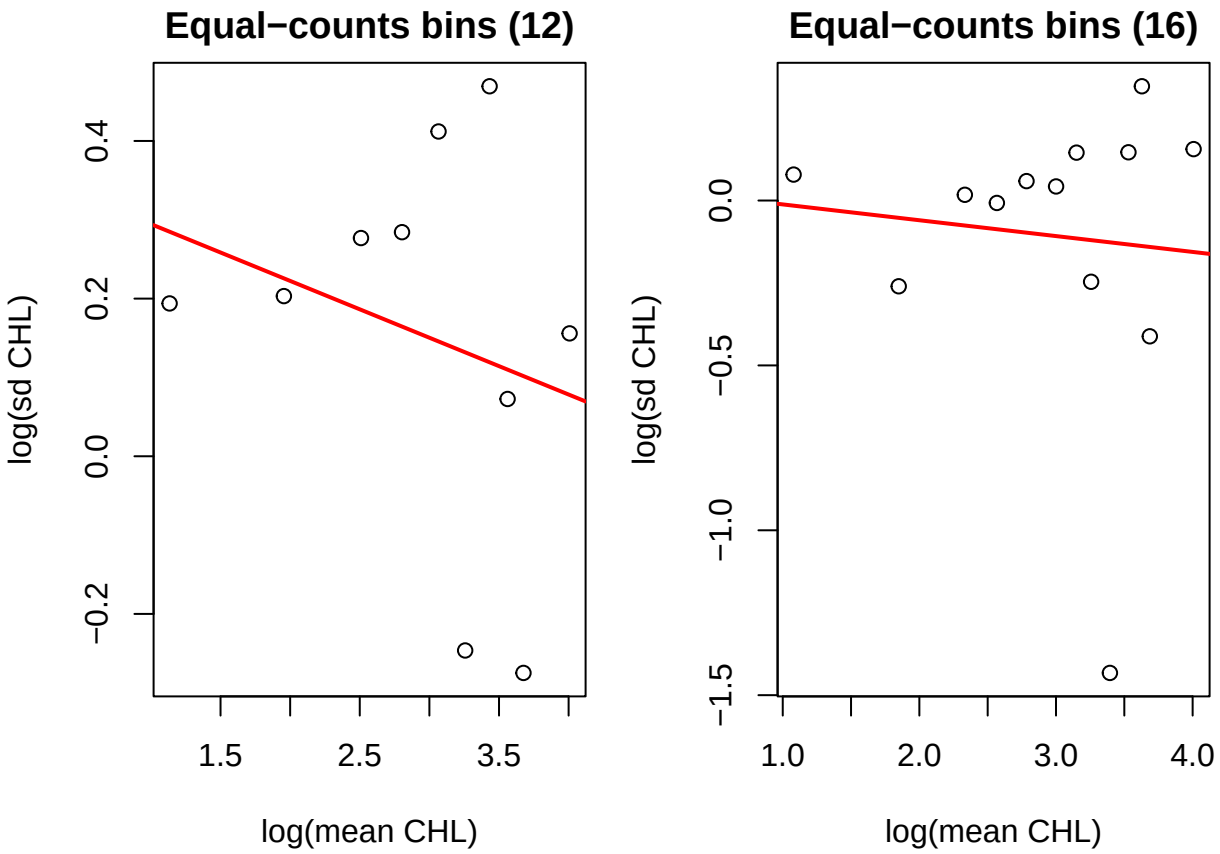
Table 1: Linear regressions for Box–Cox mean–sd plots (log(sd) log(mean))

	Bins	Term	Estimate	SE	t_value	p_value	R2	Adj_R2	N
(Intercept)	4	(Intercept)	-0.8080	1.1171	-0.72	5.45e-01	0.590	0.385	5
log(m)	4	log(m)	0.7350	0.4334	1.70	2.32e-01	0.590	0.385	5
(Intercept)1	8	(Intercept)	-1.3080	0.6758	-1.94	1.01e-01	0.536	0.459	9
log(m)1	8	log(m)	0.6939	0.2634	2.63	3.88e-02	0.536	0.459	9
(Intercept)2	12	(Intercept)	-1.6749	0.5820	-2.88	1.64e-02	0.477	0.425	13
log(m)2	12	log(m)	0.6856	0.2271	3.02	1.29e-02	0.477	0.425	13
(Intercept)3	16	(Intercept)	-2.0827	0.5297	-3.93	1.50e-03	0.444	0.404	17
log(m)3	16	log(m)	0.6907	0.2067	3.34	4.84e-03	0.444	0.404	17
(Intercept)4	20	(Intercept)	-2.2799	0.5306	-4.30	4.34e-04	0.362	0.327	21
log(m)4	20	log(m)	0.6623	0.2070	3.20	4.98e-03	0.362	0.327	21
(Intercept)5	24	(Intercept)	-2.4461	0.4215	-5.80	7.72e-06	0.457	0.433	25
log(m)5	24	log(m)	0.7086	0.1646	4.31	2.86e-04	0.457	0.433	25

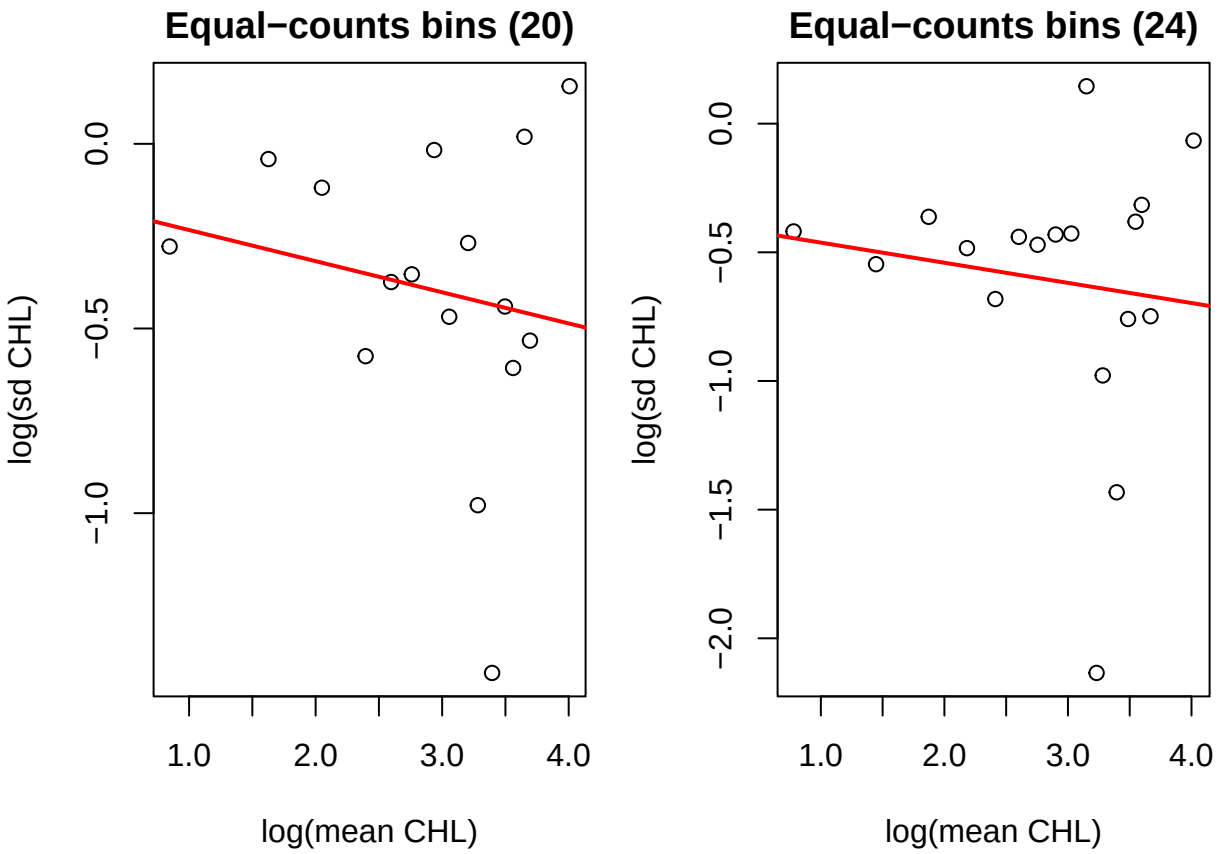
```
          to   = max(y, na.rm = TRUE),
          length.out = nbins + 1)

  cuts <- cut(y, breaks = q, include.lowest = TRUE)
  m <- tapply(y, cuts, mean, na.rm = TRUE)
  s <- tapply(y, cuts,  sd,   na.rm = TRUE)

  plot(x = log(m), y = log(s),
       xlab = "log(mean CHL)", ylab = "log(sd CHL)", main = main)
  fit <- lm(log(s) ~ log(m))
  abline(fit, col = "red", lwd = 2)
  invisible(fit)
}

op <- par(mfrow = c(1, 2), mar = c(4, 4, 2, 1))
fit4 <- bc_plot_equal(lakesDat$chl, nbins = 4,  main = "Equal-counts bins (4)")
fit8 <- bc_plot_equal(lakesDat$chl, nbins = 8, main = "Equal-counts bins (8)")
```




```
fit12 <- bc_plot_equal(lakesDat$chl, nbins = 12,  main = "Equal-counts bins (12)")
fit16 <- bc_plot_equal(lakesDat$chl, nbins = 16, main = "Equal-counts bins (16)")
```




```
fit20 <- bc_plot_equal(lakesDat$chl, nbins = 20,  main = "Equal-counts bins (20)")
fit24 <- bc_plot_equal(lakesDat$chl, nbins = 24, main = "Equal-counts bins (24)")
```




```
par(op)

tab <- rbind(summ_tbl(fit4, 4), summ_tbl(fit8, 8), summ_tbl(fit12, 12), summ_tbl(fit16, 16), summ_tbl(f
kbl(tab, booktabs = TRUE,
    caption = "Linear regressions for Box-Cox mean-sd plots (log(sd) ~ log(mean))") |>
  kable_styling(full_width = FALSE, position = "center")
```

```
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")

## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
```

Table 2: Linear regressions for Box–Cox mean–sd plots (log(sd)   log(mean))

	Bins	Term	Estimate	SE	t_value	p_value	R2	Adj_R2	N
(Intercept)	4	(Intercept)	2.6874	0.9974	2.69	1.15e-01	0.593	0.389	5
log(m)	4	log(m)	-0.5286	0.3100	-1.71	2.30e-01	0.593	0.389	5
(Intercept)1	8	(Intercept)	0.8850	0.3219	2.75	4.04e-02	0.183	0.020	8
log(m)1	8	log(m)	-0.1093	0.1031	-1.06	3.38e-01	0.183	0.020	8
(Intercept)2	12	(Intercept)	0.3666	0.2966	1.24	2.51e-01	0.064	-0.052	11
log(m)2	12	log(m)	-0.0721	0.0971	-0.74	4.79e-01	0.064	-0.052	11
(Intercept)3	16	(Intercept)	0.0362	0.5000	0.07	9.44e-01	0.008	-0.082	14
log(m)3	16	log(m)	-0.0480	0.1641	-0.29	7.75e-01	0.008	-0.082	14
(Intercept)4	20	(Intercept)	-0.1491	0.3751	-0.40	6.97e-01	0.032	-0.037	17
log(m)4	20	log(m)	-0.0843	0.1241	-0.68	5.08e-01	0.032	-0.037	17
(Intercept)5	24	(Intercept)	-0.3841	0.4439	-0.87	4.00e-01	0.017	-0.045	19
log(m)5	24	log(m)	-0.0782	0.1495	-0.52	6.08e-01	0.017	-0.045	19

```
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

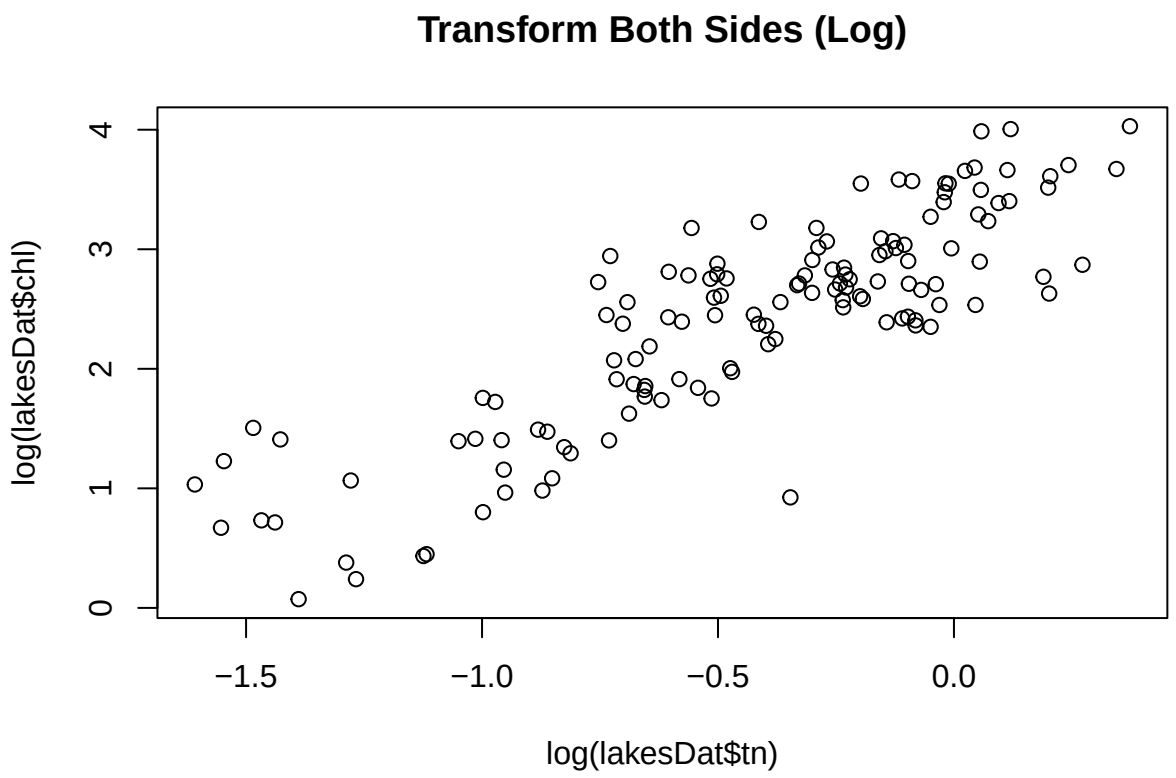
```
## Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

```
## Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

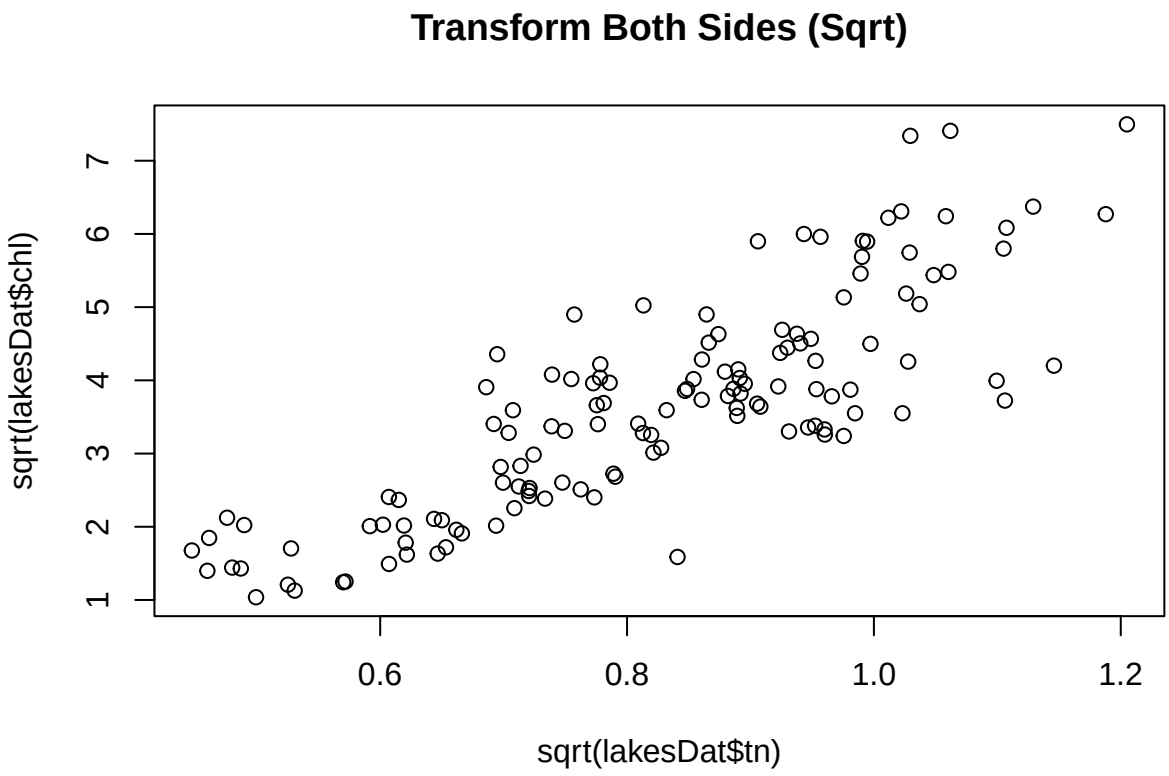
Box Cox aren't great, perhaps indicating some other random component if we decide to use a generalized

linear model.

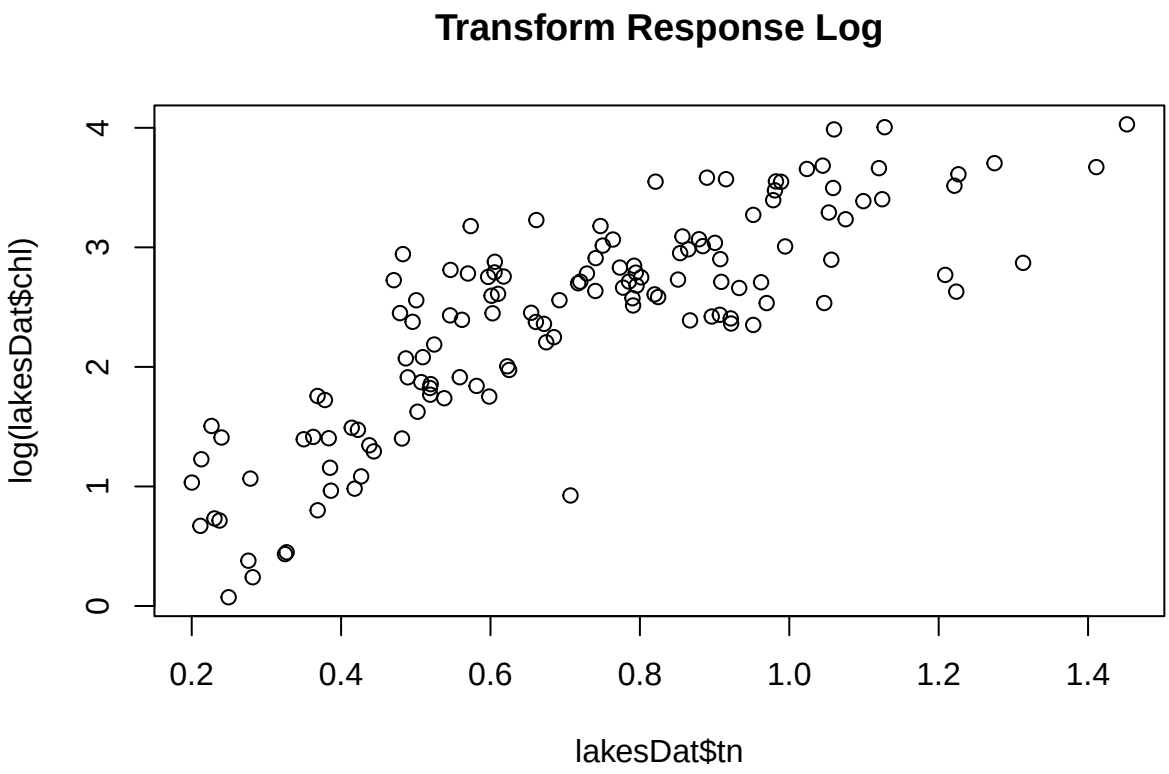
```
plot(y = log(lakesDat$chl), x = log(lakesDat$tn), main = "Transform Both Sides (Log)")
```



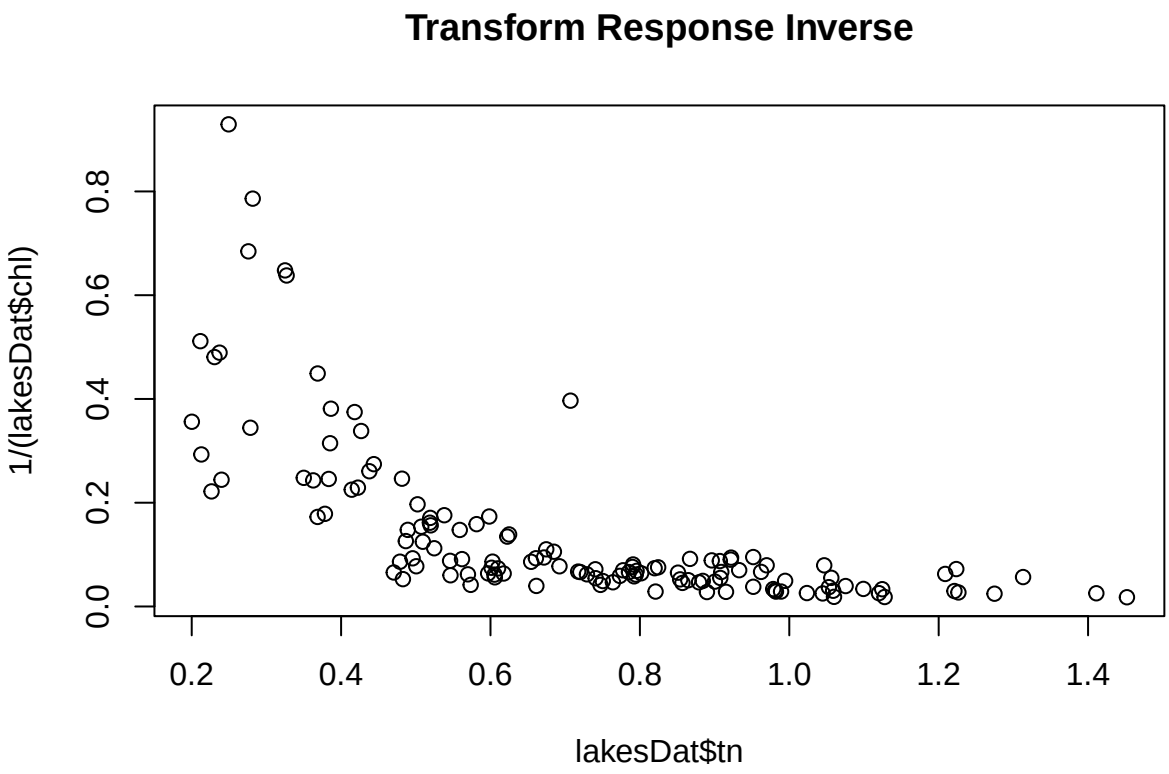

```
plot(y = sqrt(lakesDat$chl), x = sqrt(lakesDat$tn), main = "Transform Both Sides (Sqrt)")
```



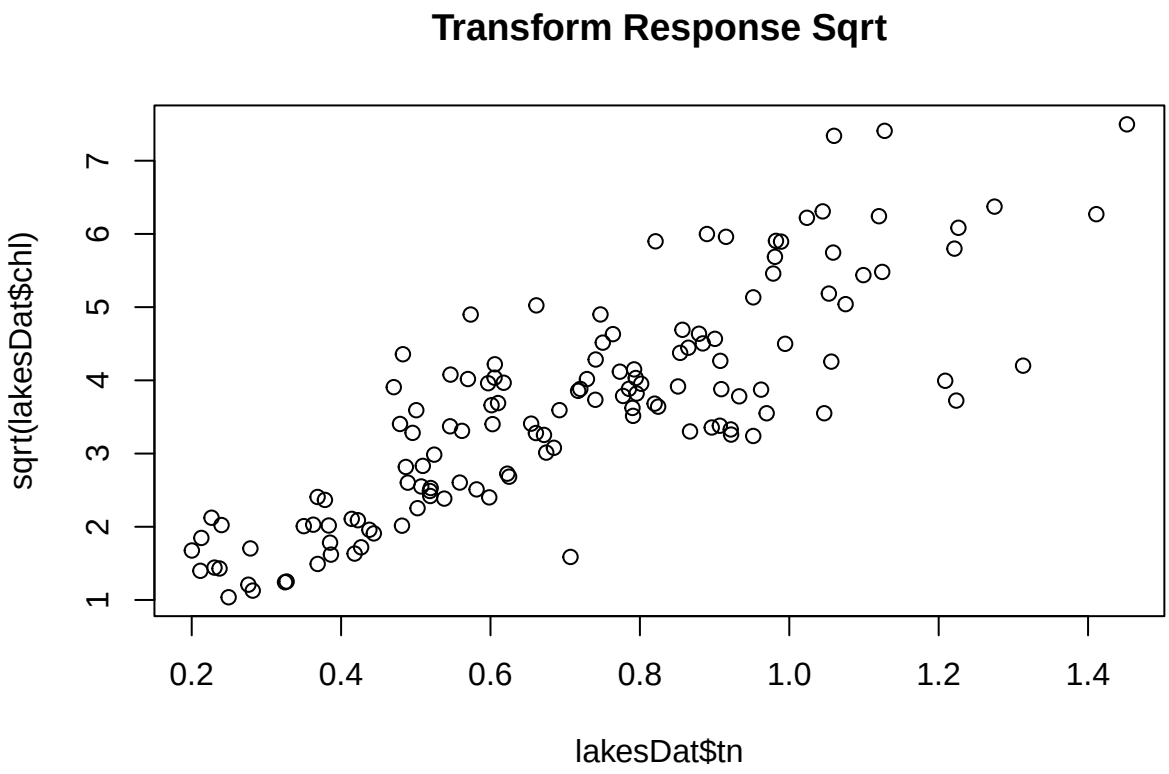

```
plot(y = log(lakesDat$chl), x = lakesDat$tn, main = "Transform Response Log")
```



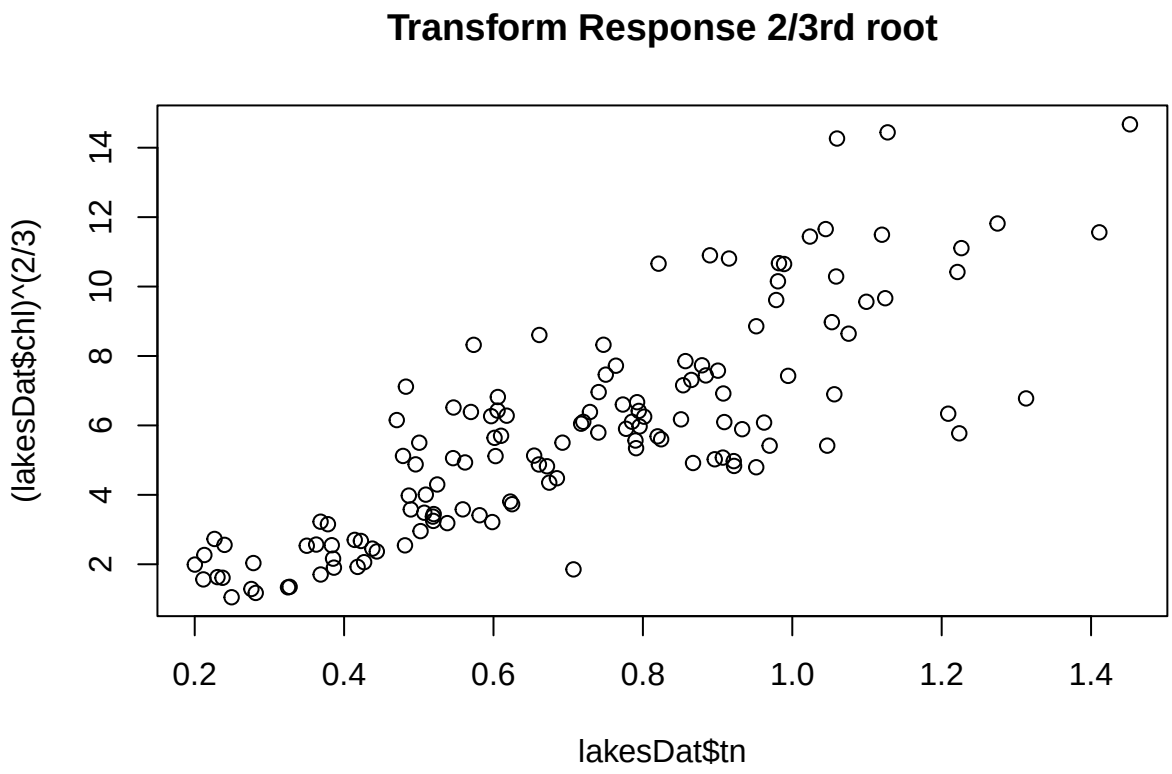

```
plot(y = 1/(lakesDat$chl), x = lakesDat$tn, main = "Transform Response Inverse")
```



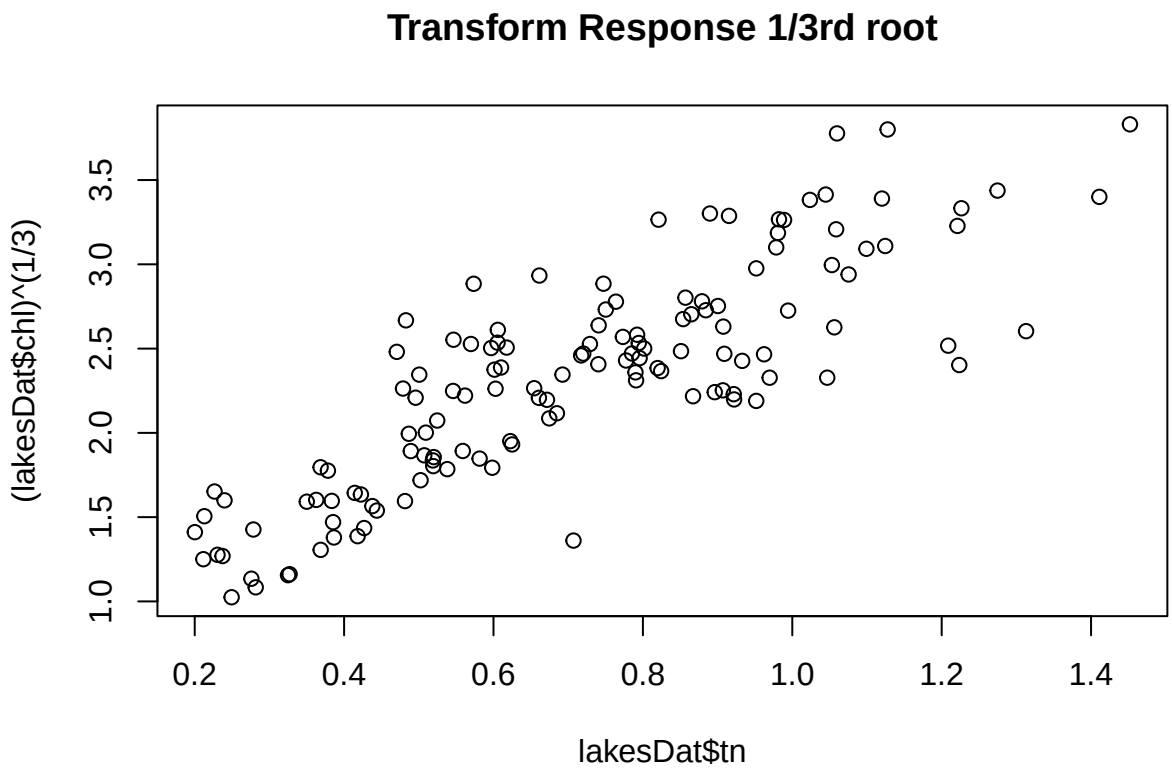

```
plot(y = sqrt(lakesDat$chl), x = lakesDat$tn, main = "Transform Response Sqrt")
```



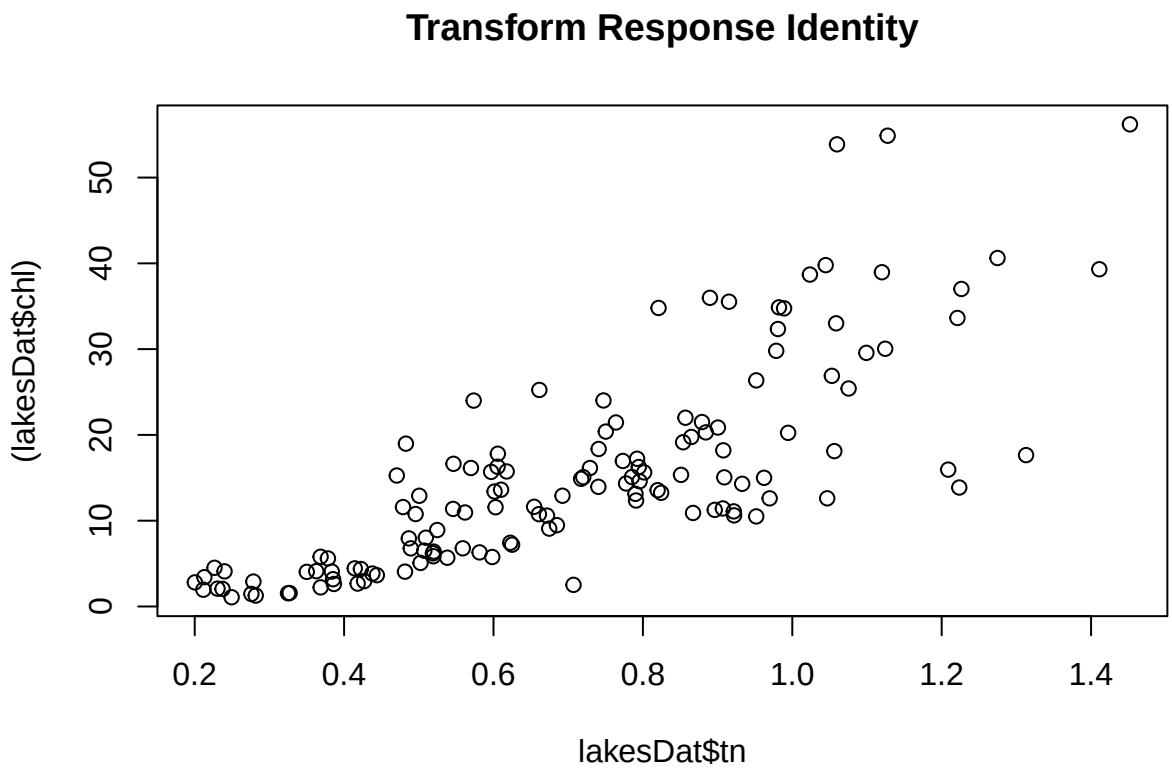

```
plot(y = (lakesDat$chl)^(2/3), x = lakesDat$tn, main = "Transform Response 2/3rd root")
```




```
plot(y = (lakesDat$chl)^(1/3), x = lakesDat$tn, main = "Transform Response 1/3rd root")
```




```
plot(y = (lakesDat$chl), x = lakesDat$tn, main = "Transform Response Identity")
```



GLM: try gamma and normal random components ($\theta = 1, 2$)

With sqrt-link (appears to be ok at linearizing)

8 power link

```
X <- cbind(1, lakesDat$tn)
Y <- lakesDat$chl

mod_normal_sqrt <- basic.glm(
  xmat   = X,
  y      = Y,
  link   = 8,
  random = 4,
  pwr = 1/2
)
```

```
## MODEL DEFINITION:
## Random Component: Normal
##
## Link Function: Power
##
## Number of Covariates (including intercept):  2
## Number of Observations:  134
##
##

## Warning in log(1 - y): NaNs produced
```

```
## Warning in log(1 - muhat): NaNs produced

## ESTIMATION RESULTS:
##
## Coefficient Estimates:  0.8655268 4.081976
## Inverse Information:
##             [,1]        [,2]
## [1,]  0.07219398 -0.07536236
## [2,] -0.07536236  0.08627472
## Estimated Phi:  0.0189735
## Unscaled Deviance:  6957.07
## Scaled Deviance:  132
## Saturated Model Log Likelihood:  388.7735
## Fitted Model Log Likelihood:  322.7735
```

```
mod_normal_13 <- basic.glm(
  xmat   = X,
  y      = Y,
  link   = 8,
  random = 4,
  pwr = 1/3
)
```

```
## MODEL DEFINITION:
## Random Component: Normal
##
## Link Function: Power
##
## Number of Covariates (including intercept):  2
## Number of Observations:  134
##
##

## Warning in log(1 - y): NaNs produced
## Warning in log(1 - y): NaNs produced

## ESTIMATION RESULTS:
##
## Coefficient Estimates:  1.232779 1.654832
## Inverse Information:
##             [,1]        [,2]
## [1,]  0.01321295 -0.01325010
## [2,] -0.01325010  0.01446031
## Estimated Phi:  0.01867206
## Unscaled Deviance:  7069.387
## Scaled Deviance:  132
## Saturated Model Log Likelihood:  389.8465
## Fitted Model Log Likelihood:  323.8465
```

```
mod_gamma_sqrt <- basic.glm(
  xmat   = X,
  y      = Y,
```

```
  link   = 8,
  random = 5,
  pwr = 1/2
)
```

```
## MODEL DEFINITION:
## Random Component: Gamma
##
## Link Function: Power
##
## Number of Covariates (including intercept):  2
## Number of Observations:  134
##
##
```

```
## Warning in log(1 - y): NaNs produced
## Warning in log(1 - y): NaNs produced
```

```
## ESTIMATION RESULTS:
##
## Coefficient Estimates:  0.5125294 4.546641
## Inverse Information:
##             [,1]        [,2]
## [1,]  0.01676142 -0.02681144
## [2,] -0.02681144  0.05389379
## Estimated Phi:  4.910914
## Unscaled Deviance:  26.5909
## Scaled Deviance:  130.5856
## Saturated Model Log Likelihood:  -18.78068
## Fitted Model Log Likelihood:  -84.0735
```

```
mod_gamma_13 <- basic.glm(
  xmat   = X,
  y      = Y,
  link   = 8,
  random = 5,
  pwr = 1/3
)
```

```
## MODEL DEFINITION:
## Random Component: Gamma
##
## Link Function: Power
##
## Number of Covariates (including intercept):  2
## Number of Observations:  134
##
##
```

```
## Warning in log(1 - y): NaNs produced
## Warning in log(1 - y): NaNs produced
```

```
## ESTIMATION RESULTS:
##
## Coefficient Estimates:  0.9246554 2.052324
## Inverse Information:
##              [,1]         [,2]
## [1,]  0.004602832 -0.006649603
## [2,] -0.006649603  0.011697532
## Estimated Phi:  4.70624
## Unscaled Deviance:  27.41648
## Scaled Deviance:  129.0285
## Saturated Model Log Likelihood:  -21.7314
## Fitted Model Log Likelihood:  -86.24565
```

```
combine_glm_results <- function(mod, model_name) {
  beta_hat <- mod$estb[, 1]
  se_hat <- sqrt(diag(mod$invinf))
  z <- 1.96
  LCL <- beta_hat - z * se_hat
  UCL <- beta_hat + z * se_hat

  phi <- mod$ests$phi
  # unscaled deviance
  udev <- mod$ests$udev
  sdev <- mod$ests$sdev
  # scaled deviance
  # sdev <- udev / phi

  data.frame(
    Term        = c("Intercept", "Body Mass"),
    Estimate    = beta_hat,
    SE          = se_hat,
    Phi         = phi,
    UnscaledDev = udev,
    ScaledDev   = sdev,
    LogLik      = mod$ests$loglik.fitted,
    LCL         = LCL,
    UCL         = UCL,
    Model       = model_name,
    check.names = FALSE
  )
}

tab_combined <- rbind(
  combine_glm_results(mod_normal_sqrt, "Normal (sqrt link)"),
  combine_glm_results(mod_normal_13, "Normal 1/3rd link"),
  combine_glm_results(mod_gamma_sqrt,    "Gamma (sqrt link)"),
  combine_glm_results(mod_gamma_13, "Gamma 1/3rd link")
)

kableExtra::kbl(
  tab_combined,
  digits = 4,
  align  = "lrrrrrrrrrl",
  row.names = FALSE,
```

```r
  caption = "Model comparison with Wald 95\\% CIs and both unscaled and scaled deviances"
) |>
  kableExtra::kable_styling(
    full_width = FALSE,
    position   = "center",
    latex_options = c("hold_position", "scale_down")
  )
```

```
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

```
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning in attr(.knitEnv$meta, "knit_meta_id"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

```
## Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
```

```
## Use 'xfun::attr2()' instead.
## See help("Deprecated")

## Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 3: Model comparison with Wald 95% CIs and both unscaled and scaled deviances

Term	Estimate	SE	Phi	UnscaledDev	ScaledDev	LogLik	LCL	UCL	Model
Intercept	0.8655	0.2687	0.0190	6957.0700	132.0000	322.7735	0.3389	1.3922	Normal (sqrt link)
Body Mass	4.0820	0.2937	0.0190	6957.0700	132.0000	322.7735	3.5063	4.6577	Normal (sqrt link)
Intercept	1.2328	0.1149	0.0187	7069.3867	132.0000	323.8465	1.0075	1.4581	Normal 1/3rd link
Body Mass	1.6548	0.1203	0.0187	7069.3867	132.0000	323.8465	1.4191	1.8905	Normal 1/3rd link
Intercept	0.5125	0.1295	4.9109	26.5909	130.5856	-84.0735	0.2588	0.7663	Gamma (sqrt link)
Body Mass	4.5466	0.2322	4.9109	26.5909	130.5856	-84.0735	4.0916	5.0017	Gamma (sqrt link)
Intercept	0.9247	0.0678	4.7062	27.4165	129.0285	-86.2456	0.7917	1.0576	Gamma 1/3rd link
Body Mass	2.0523	0.1082	4.7062	27.4165	129.0285	-86.2456	1.8403	2.2643	Gamma 1/3rd link

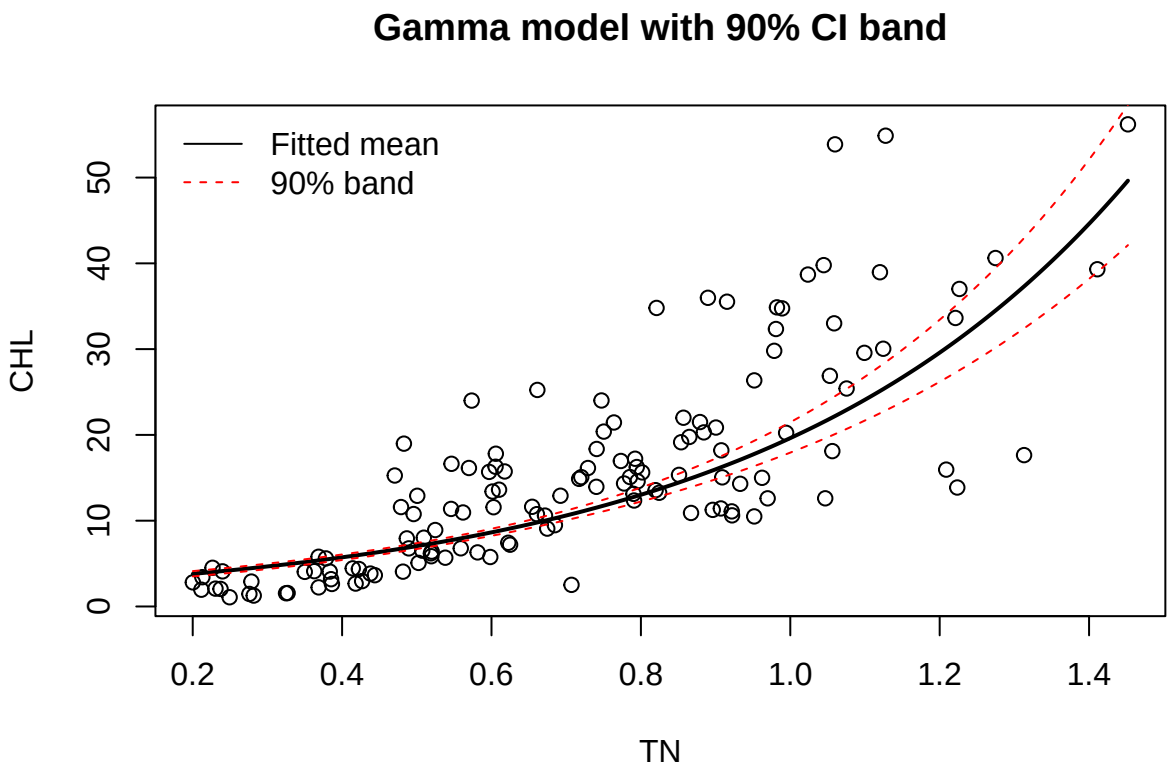
```
# beta_hat <- mod_gamma_sqrt$estb[,1]
# vcov_mat <- mod_gamma_sqrt$invinf
# se_hat <- sqrt(diag(vcov_mat))

beta_hat <- mod_gamma_13$estb[,1]
vcov_mat <- mod_gamma_13$invinf
se_hat <- sqrt(diag(vcov_mat))


# 90% CI
z90 <- qnorm(0.95)

wald_lo <- beta_hat - z90 * se_hat
wald_hi <- beta_hat + z90 * se_hat

param_labels <- c("$\\beta_0$ (Intercept)", "$\\beta_1$ (Body Mass)")

wald_table <- data.frame(
  Parameter = param_labels,
  Estimate  = formatC(beta_hat,  format = "f", digits = 3),
  SE        = formatC(se_hat,    format = "f", digits = 4),
  LCL       = formatC(wald_lo,   format = "f", digits = 3),
  UCL       = formatC(wald_hi,   format = "f", digits = 3)
)

kable(wald_table, caption = "Wald 90% Confidence Intervals (Gamma model)")
```

```
## Warning in attr(x, "align"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")

## Warning in attr(x, "format"): 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 4: Wald 90% Confidence Intervals (Gamma model)

Parameter	Estimate	SE	LCL	UCL
β_0 (Intercept)	0.925	0.0678	0.813	1.036
β_1 (Body Mass)	2.052	0.1082	1.874	2.230

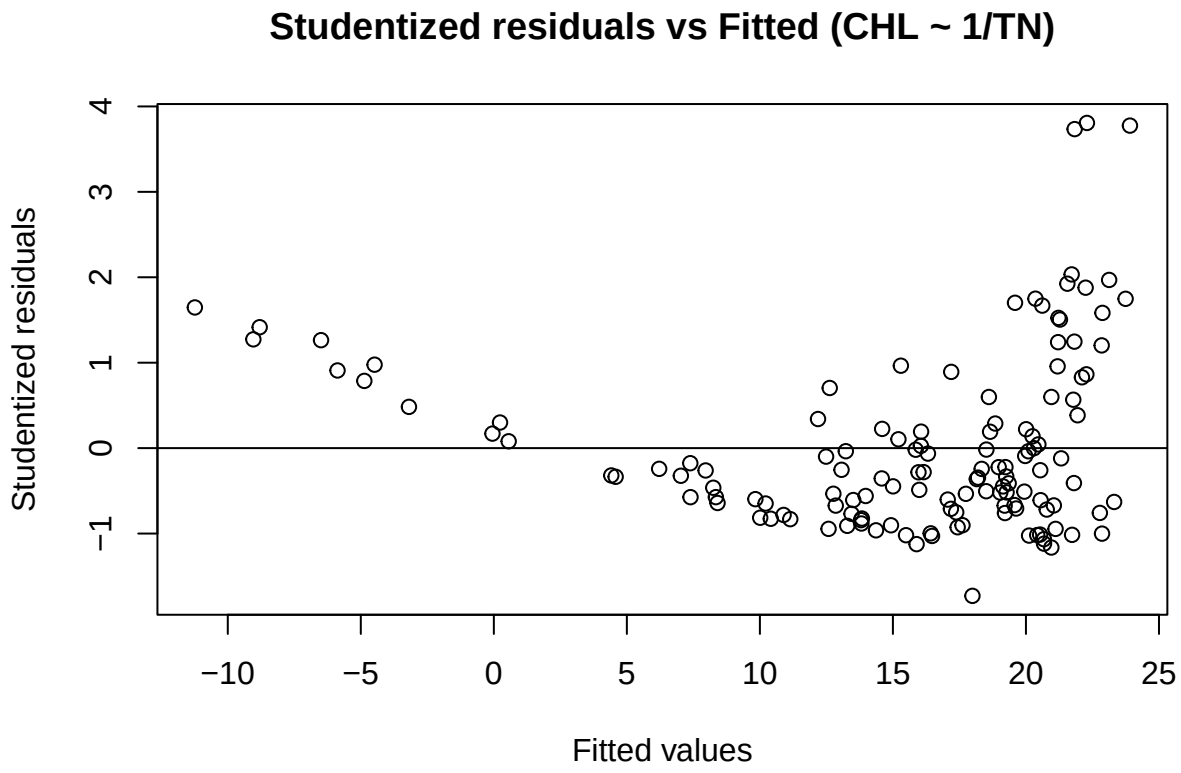
```
x_grid <- seq(min(lakesDat$tn), max(lakesDat$tn), length.out = 200)
X_grid <- cbind(1, x_grid)

eta_hat <- X_grid %*% beta_hat
se_eta <- sqrt(rowSums((X_grid %*% vcov_mat) * X_grid))

eta_lo <- eta_hat - z90 * se_eta
eta_hi <- eta_hat + z90 * se_eta

mu_hat <- exp(eta_hat)
mu_lo <- exp(eta_lo)
mu_hi <- exp(eta_hi)

gamma_band <- data.frame(
  bm = x_grid,
  fit = mu_hat,
  lower = mu_lo,
  upper = mu_hi
)

plot(lakesDat$tn, lakesDat$chl, col = "black",
     xlab = "TN", ylab = "CHL", main = "Gamma model with 90% CI band")

lines(x_grid, mu_hat, lwd = 2, col = "black")
lines(x_grid, mu_lo,  lwd = 1, col = "red", lty = 2)
lines(x_grid, mu_hi,  lwd = 1, col = "red", lty = 2)

legend("topleft",
       legend = c("Fitted mean", "90% band"),
       col = c("black", "red"), lty = c(1, 2), bty = "n")
```



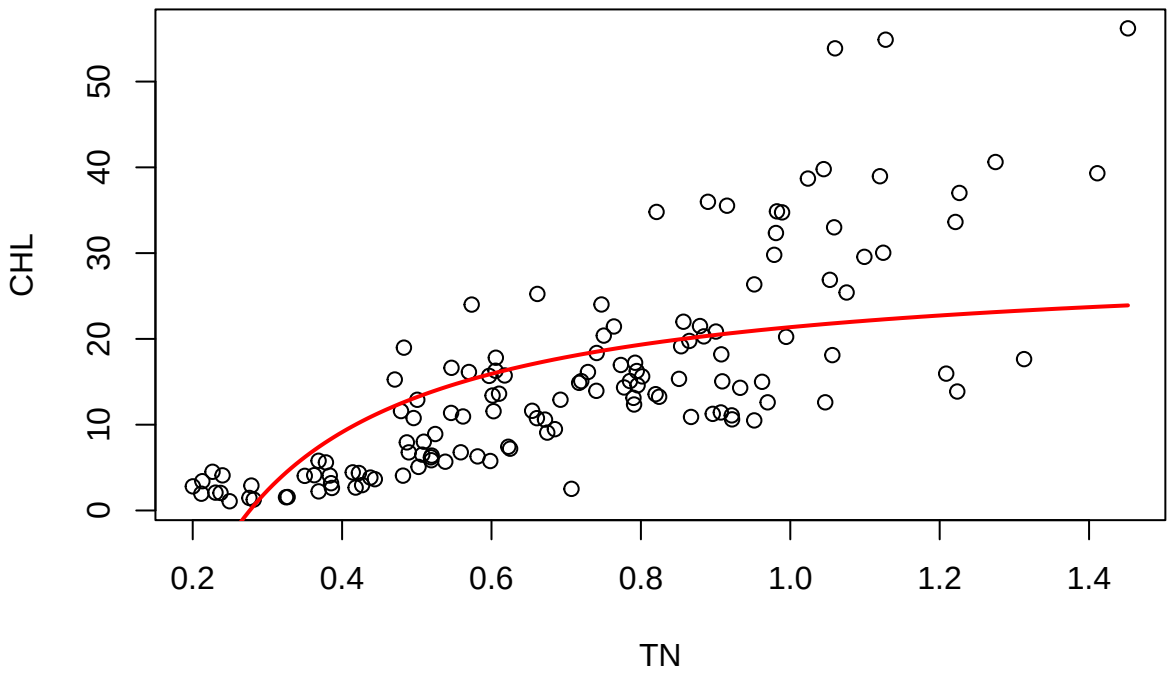

```r
lakesDat$phi <- 1 / lakesDat$tn
fit_phi <- lm(chl ~ phi, data = lakesDat)

summary(fit_phi)
```

```
##
## Call:
## lm(formula = chl ~ phi, data = lakesDat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.468  -6.303  -2.715   4.067  32.587
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  29.5291     1.6605  17.784   <2e-16 ***
## phi          -8.1603     0.8451  -9.657   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.05 on 132 degrees of freedom
## Multiple R-squared:  0.414,  Adjusted R-squared:  0.4095
## F-statistic: 93.25 on 1 and 132 DF,  p-value: < 2.2e-16
```

```r
stud_resid <- rstudent(fit_phi)
plot(fitted(fit_phi), stud_resid,
     xlab = "Fitted values", ylab = "Studentized residuals",
     main = "Studentized residuals vs Fitted (CHL ~ 1/TN)")
abline(h = 0)
```




```r
plot(lakesDat$tn, lakesDat$chl,
     xlab = "TN", ylab = "CHL")

tn.grid <- seq(min(lakesDat$tn), max(lakesDat$tn), length.out = 200)
pred <- predict(fit_phi,
                newdata = data.frame(phi = 1 / tn.grid))

lines(tn.grid, pred, col = "red", lwd = 2)
```




```
lakesDat$phi <- 1 / lakesDat$tn
fit_phi <- lm(chl ~ phi, data = lakesDat)

mu_hat <- fitted(fit_phi)   # estimated means
e_hat  <- resid(fit_phi)    # ordinary residuals

aux <- lm(log(e_hat^2) ~ log(mu_hat))
```

```
## Warning in log(mu_hat): NaNs produced
```

```
coef(aux)
```

```
## (Intercept) log(mu_hat)
##   0.9944561   0.7322746
```

```
delta_hat <- 0.5 * coef(aux)[2]   # slope approx 2delta  ->  delta = slope / 2
delta_hat
```

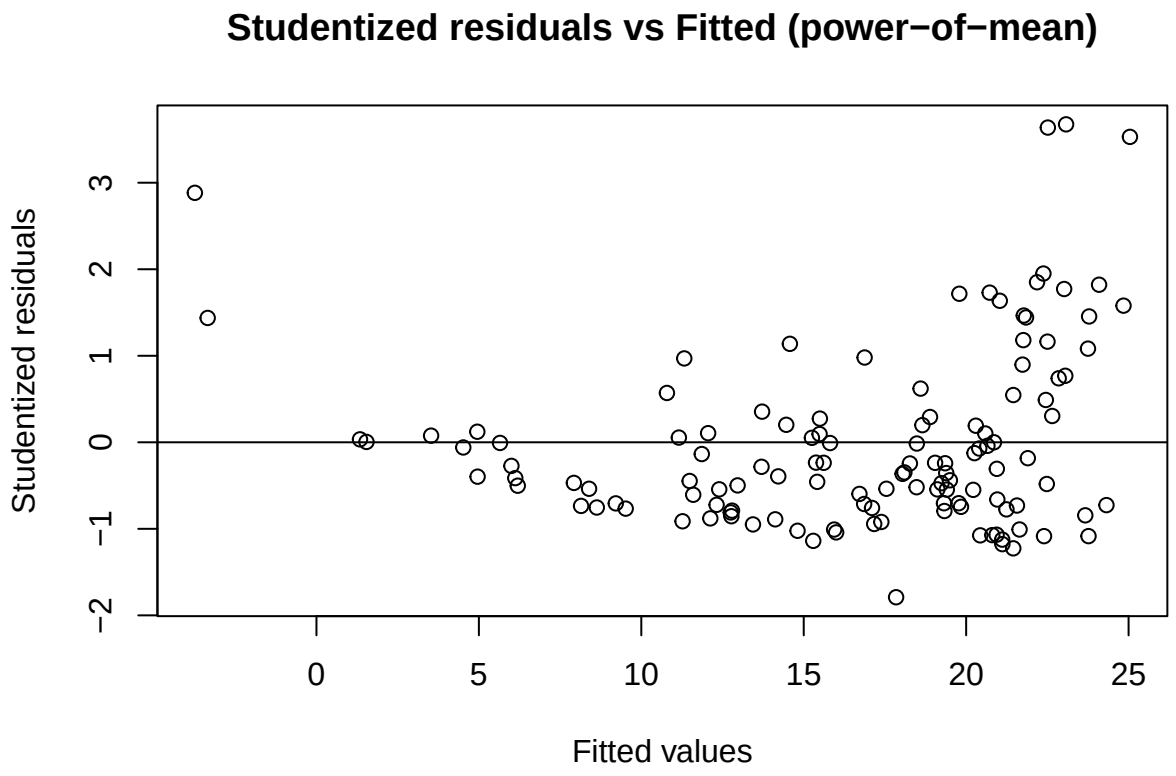
```
## log(mu_hat)
##   0.3661373
```

```
delta_hat1 <- 0.25
delta_hat2 <- 0.5
```

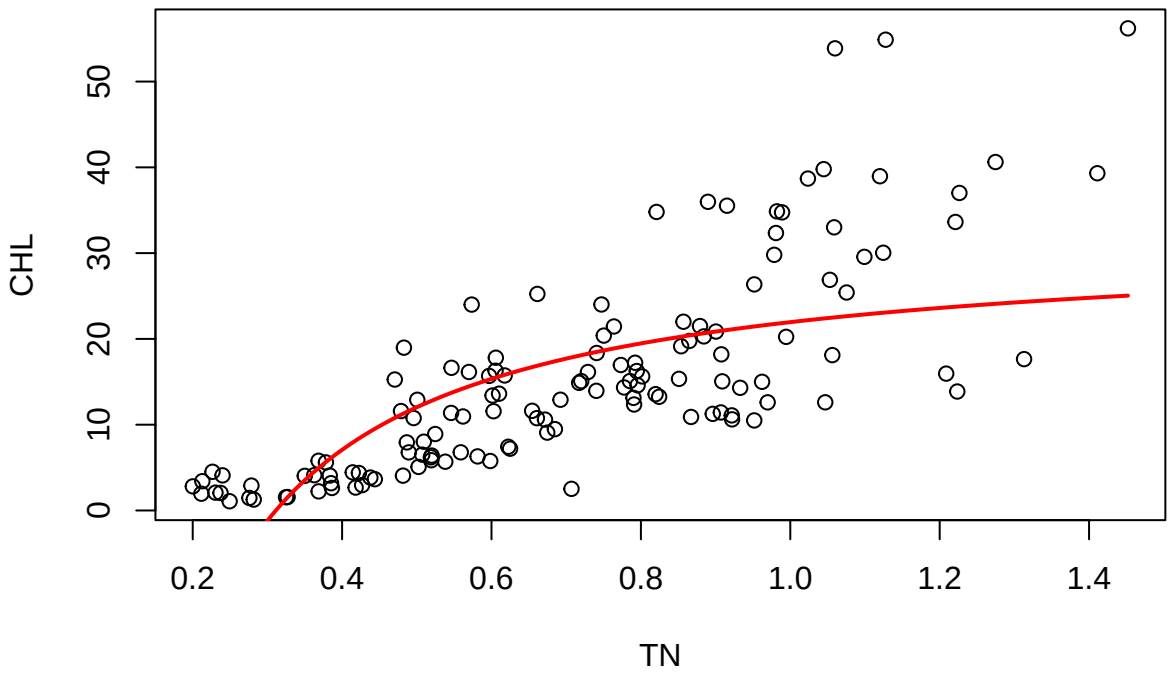
```r
w <- 1 / (mu_hat^(2 * delta_hat1))

fit_phi_pom <- lm(chl ~ phi, data = lakesDat, weights = w)

summary(fit_phi_pom)
```

```
##
## Call:
## lm(formula = chl ~ phi, data = lakesDat, weights = w)
##
## Weighted Residuals:
##    Min     1Q Median     3Q    Max
## -7.440 -3.065 -1.288  1.230 14.636
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  31.8748     1.7339   18.38   <2e-16 ***
## phi          -9.9211     0.8713  -11.39   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.21 on 123 degrees of freedom
##   (9 observations deleted due to missingness)
## Multiple R-squared:  0.5132, Adjusted R-squared:  0.5092
## F-statistic: 129.7 on 1 and 123 DF,  p-value: < 2.2e-16
```

```r
stud_resid2 <- rstudent(fit_phi_pom)
plot(fitted(fit_phi_pom), stud_resid2,
     xlab = "Fitted values",
     ylab = "Studentized residuals",
     main = "Studentized residuals vs Fitted (power-of-mean)")
abline(h = 0)
```




```r
plot(lakesDat$tn, lakesDat$chl,
     xlab = "TN", ylab = "CHL")

tn.grid <- seq(min(lakesDat$tn), max(lakesDat$tn), length.out = 200)
pred <- predict(fit_phi_pom,
                newdata = data.frame(phi = 1 / tn.grid))

lines(tn.grid, pred, col = "red", lwd = 2)
```

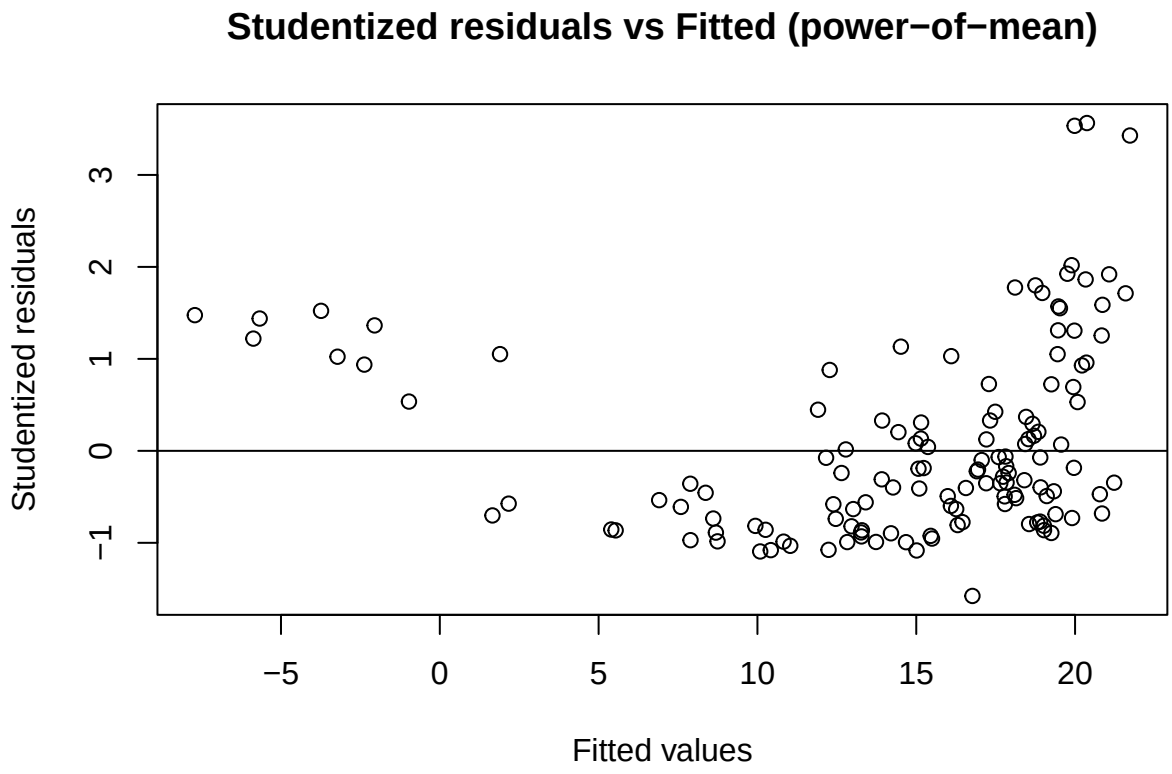



```r
# delta_hat1 <- 0.25
delta_hat2 <- 0.50
# w <- 1 / (mu_hat^(2 * delta_hat2))
eps <- 1e-6
w <- 1 / ((abs(mu_hat) + eps)^(2 * delta_hat2))

# ISSUE NEGATIVE WEIGHTS
fit_phi_pom <- lm(chl ~ phi, data = lakesDat, weights = w)

summary(fit_phi_pom)
```

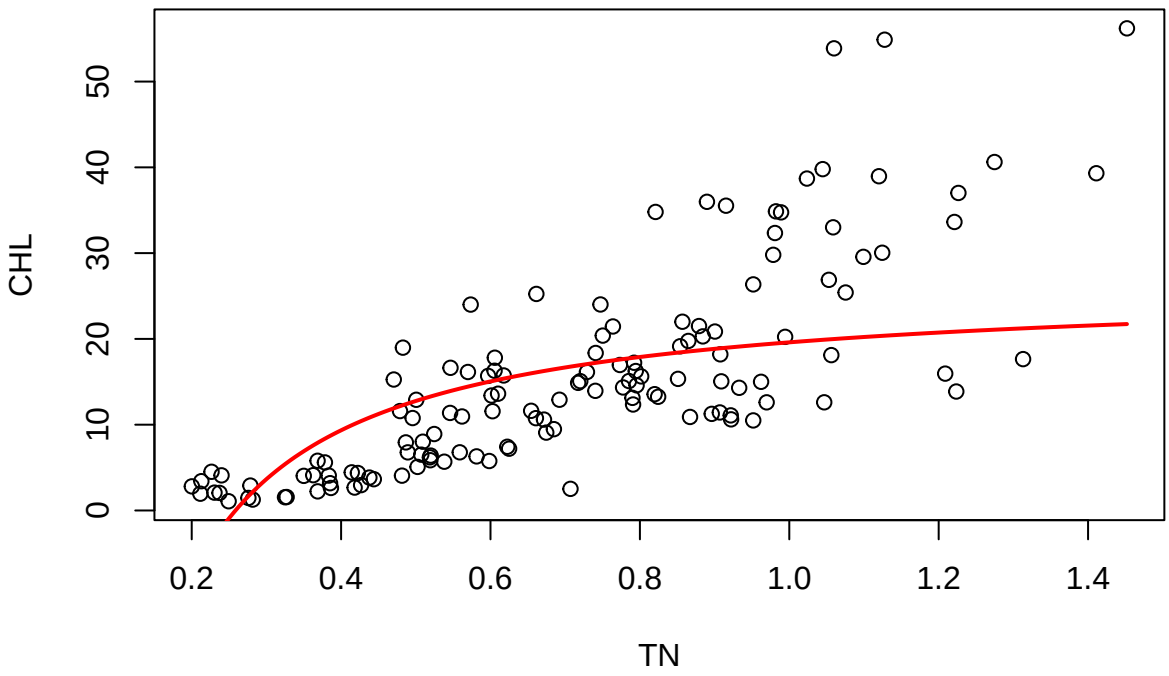
```
##
## Call:
## lm(formula = chl ~ phi, data = lakesDat, weights = w)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3593 -1.5773 -0.5614  1.4023  7.3083
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  26.4351     1.3833   19.11   <2e-16 ***
## phi          -6.8352     0.4158  -16.44   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 2.151 on 132 degrees of freedom
## Multiple R-squared:  0.6718, Adjusted R-squared:  0.6694
## F-statistic: 270.2 on 1 and 132 DF,  p-value: < 2.2e-16
```

```r
stud_resid2 <- rstudent(fit_phi_pom)
plot(fitted(fit_phi_pom), stud_resid2,
     xlab = "Fitted values",
     ylab = "Studentized residuals",
     main = "Studentized residuals vs Fitted (power-of-mean)")
abline(h = 0)
```

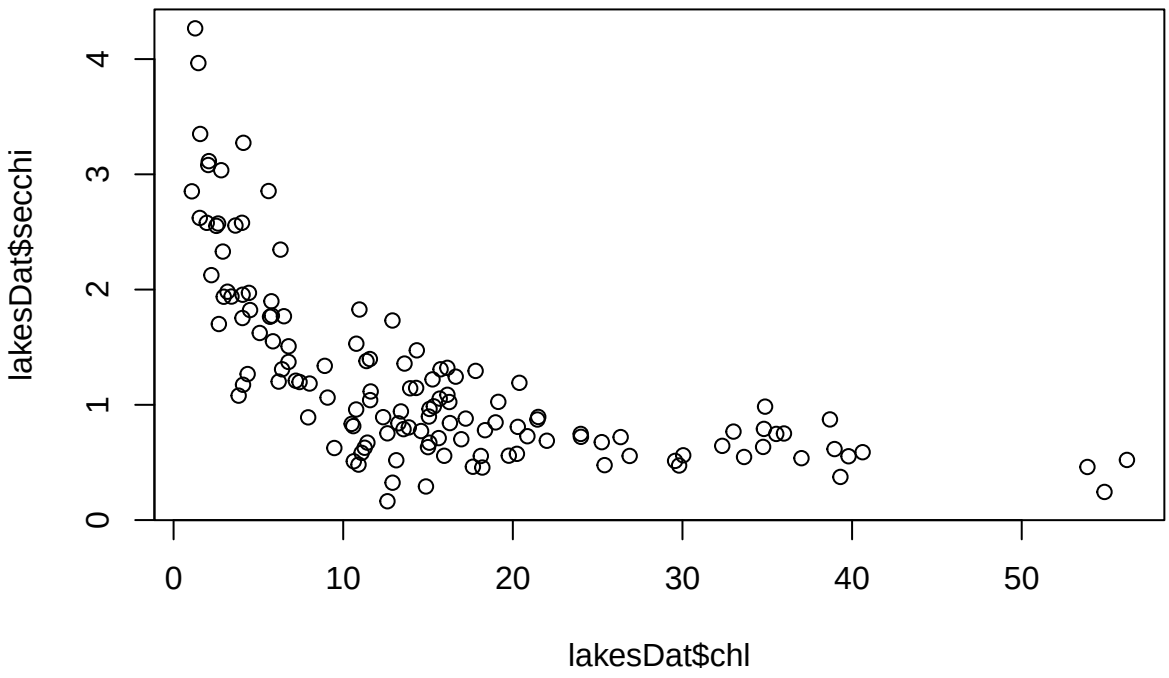



```r
plot(lakesDat$tn, lakesDat$chl,
     xlab = "TN", ylab = "CHL")

tn.grid <- seq(min(lakesDat$tn), max(lakesDat$tn), length.out = 200)
pred <- predict(fit_phi_pom,
                newdata = data.frame(phi = 1 / tn.grid))

lines(tn.grid, pred, col = "red", lwd = 2)
```

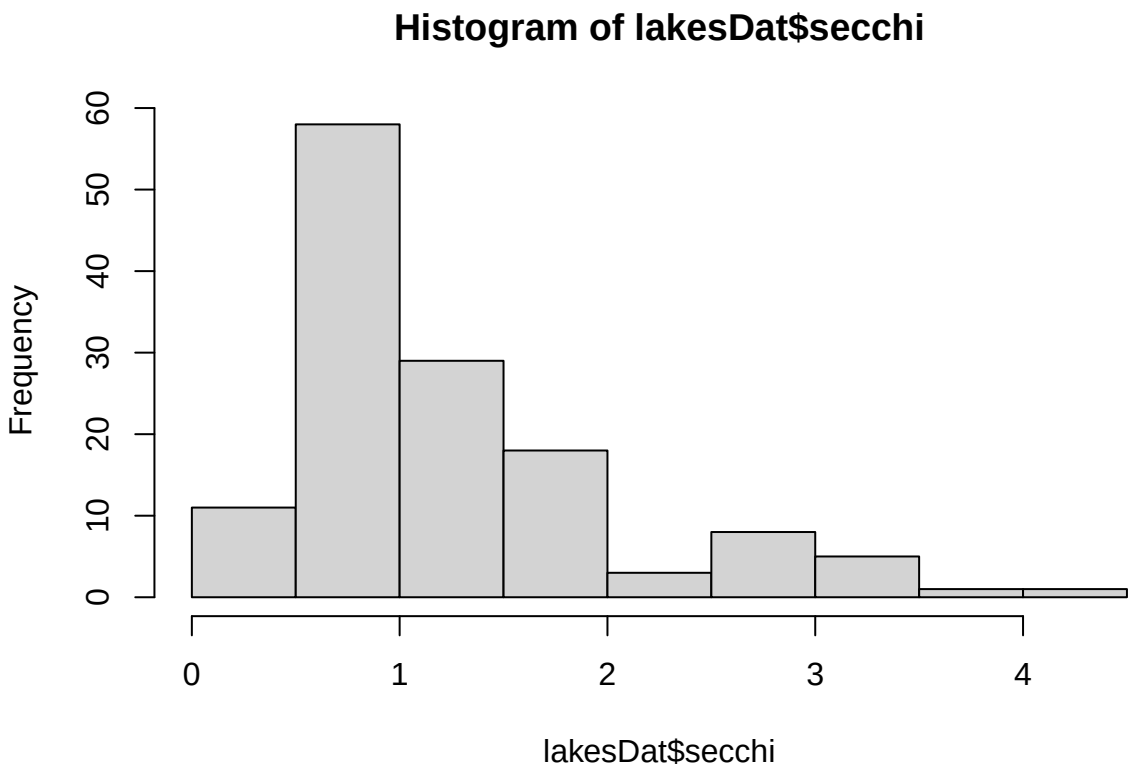


Q2: SECCHI & CHL (Plains vs. Ozarks)

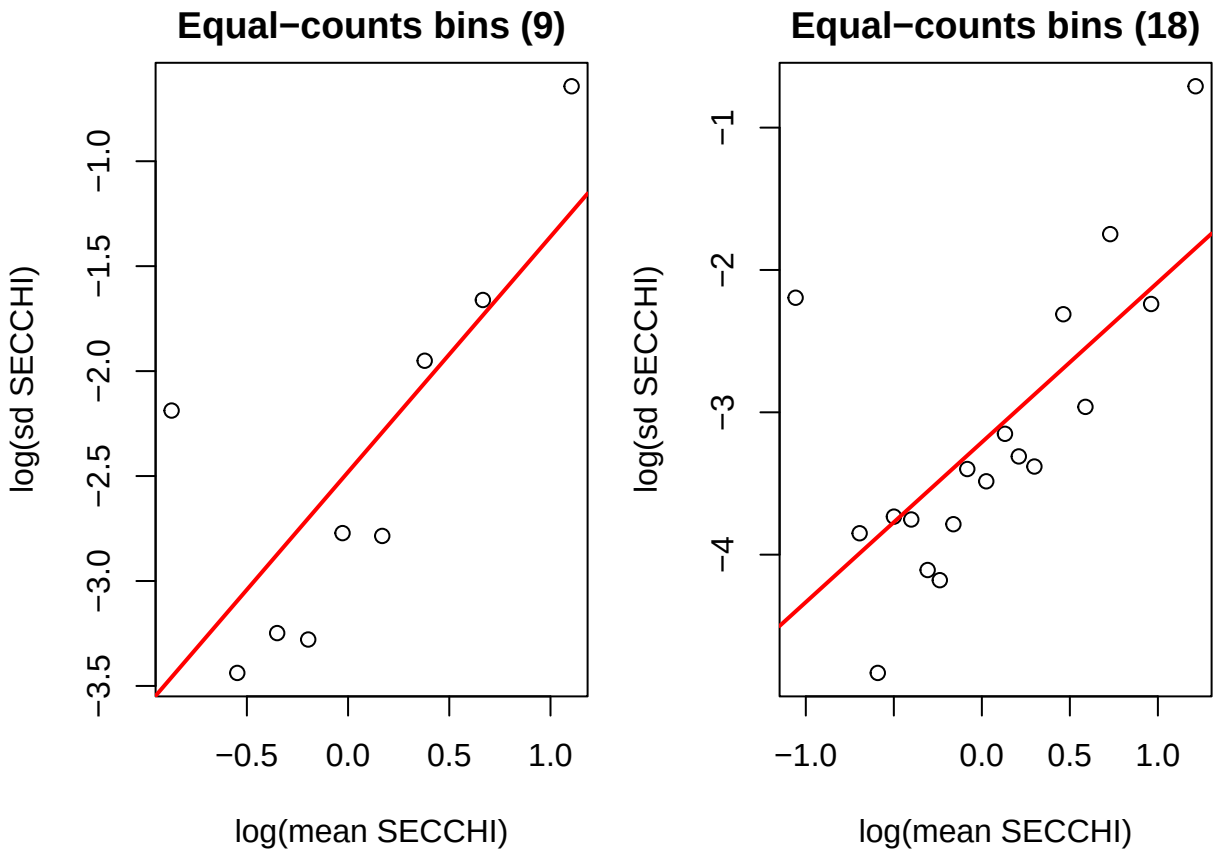
```
plot(x = lakesDat$chl, y = lakesDat$secchi)
```



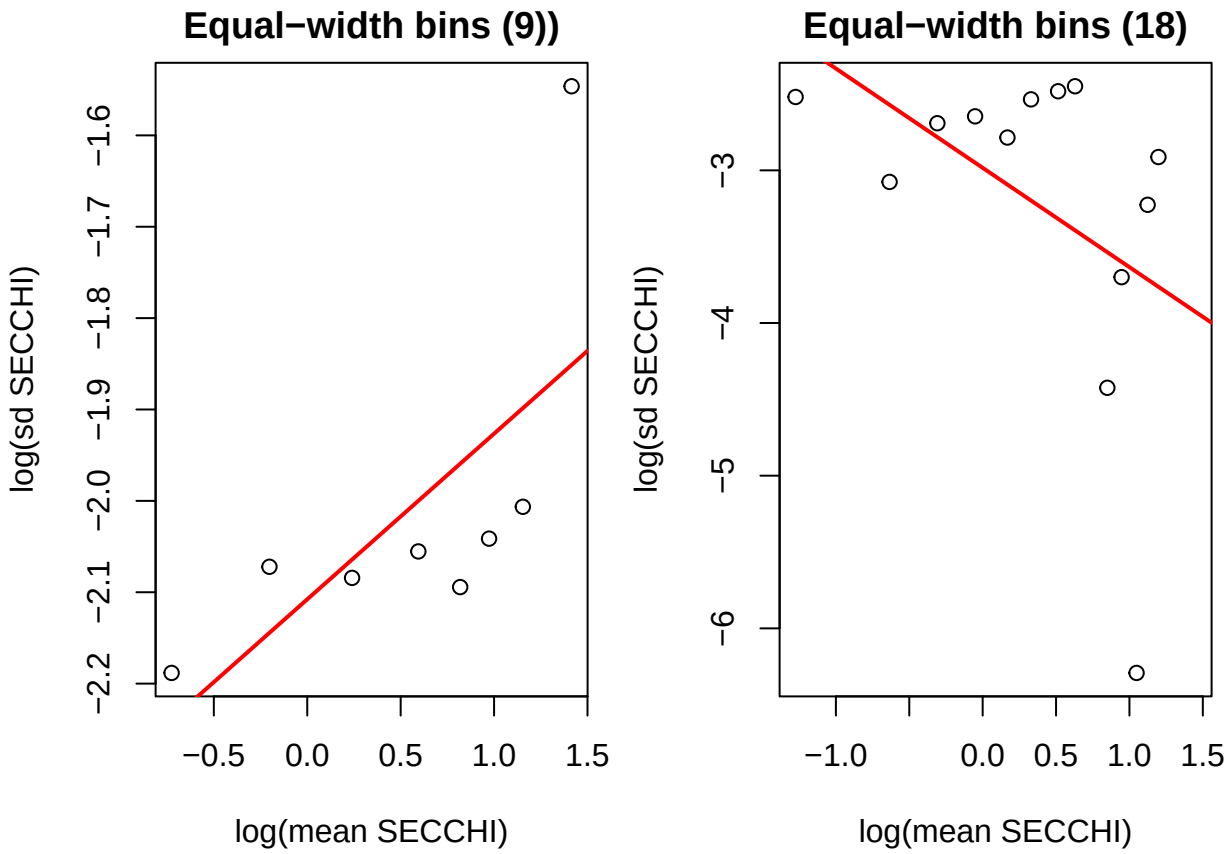

```
hist(lakesDat$secchi)
```




```r
# Box Cox
# log(lakesDat$chl)

bc_plot <- function(y, nbins, main) {
  q <- quantile(y, probs = seq(0, 1, length.out = nbins + 1), na.rm = TRUE)
  cuts <- cut(y, breaks = q, include.lowest = TRUE)
  m <- tapply(y, cuts, mean, na.rm = TRUE)
  s <- tapply(y, cuts,  sd,   na.rm = TRUE)

  plot(x = log(m), y = log(s),
       xlab = "log(mean SECCHI)", ylab = "log(sd SECCHI)", main = main)
  fit <- lm(log(s) ~ log(m))
  abline(fit, col = "red", lwd = 2)
  invisible(fit)
}

op <- par(mfrow = c(1, 2), mar = c(4, 4, 2, 1))
fit5 <- bc_plot(lakesDat$secchi, nbins = 9,  main = "Equal-counts bins (9)")
fit17 <- bc_plot(lakesDat$secchi, nbins = 18, main = "Equal-counts bins (18)")
```

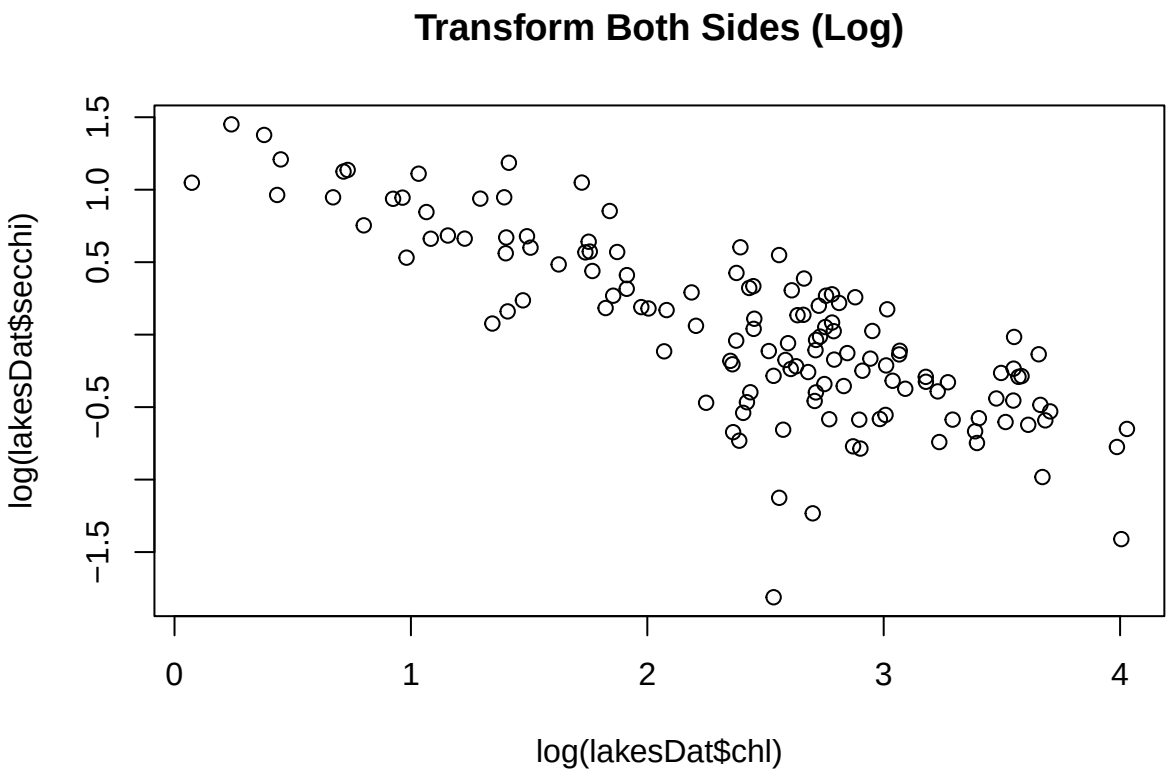



```r
par(op)

bc_plot_equal <- function(y, nbins, main) {
  # equal-width break points
  q <- seq(from = min(y, na.rm = TRUE),
           to   = max(y, na.rm = TRUE),
           length.out = nbins + 1)

  cuts <- cut(y, breaks = q, include.lowest = TRUE)
  m <- tapply(y, cuts, mean, na.rm = TRUE)
  s <- tapply(y, cuts,  sd,   na.rm = TRUE)

  plot(x = log(m), y = log(s),
       xlab = "log(mean SECCHI)", ylab = "log(sd SECCHI)", main = main)
  fit <- lm(log(s) ~ log(m))
  abline(fit, col = "red", lwd = 2)
  invisible(fit)
}

op <- par(mfrow = c(1, 2), mar = c(4, 4, 2, 1))
fit5  <- bc_plot_equal(lakesDat$secchi, nbins = 9,  main = "Equal-width bins (9))")
fit17 <- bc_plot_equal(lakesDat$secchi, nbins = 18, main = "Equal-width bins (18)")
```

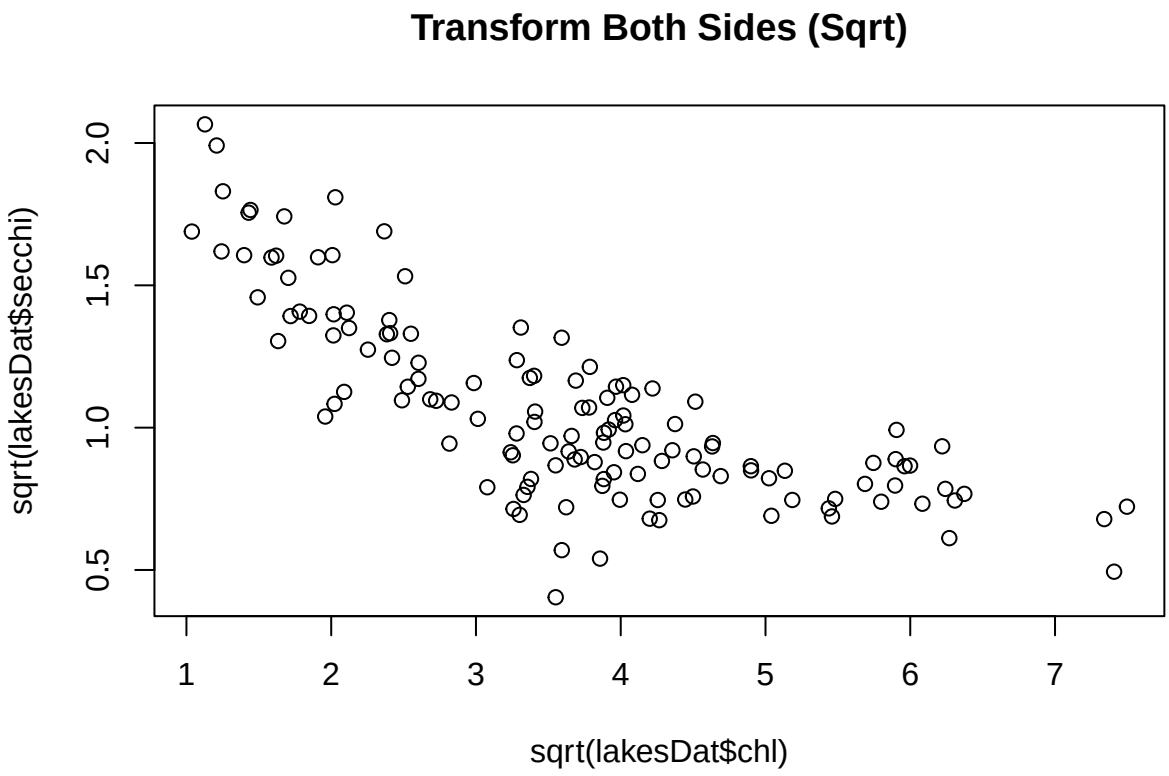



```
par(op)
```

```
plot(y = log(lakesDat$secchi), x = log(lakesDat$chl), main = "Transform Both Sides (Log)")
```




```
plot(y = sqrt(lakesDat$secchi), x = sqrt(lakesDat$chl), main = "Transform Both Sides (Sqrt)")
```



Volume of a cone:

$$V = \pi r^2 \frac{h}{3}$$

h would be the Secchi "depth", so we could have instead:

$$h = \frac{3V}{r^2}$$

and CHL is technically a volumetric measure $\mu g/L$

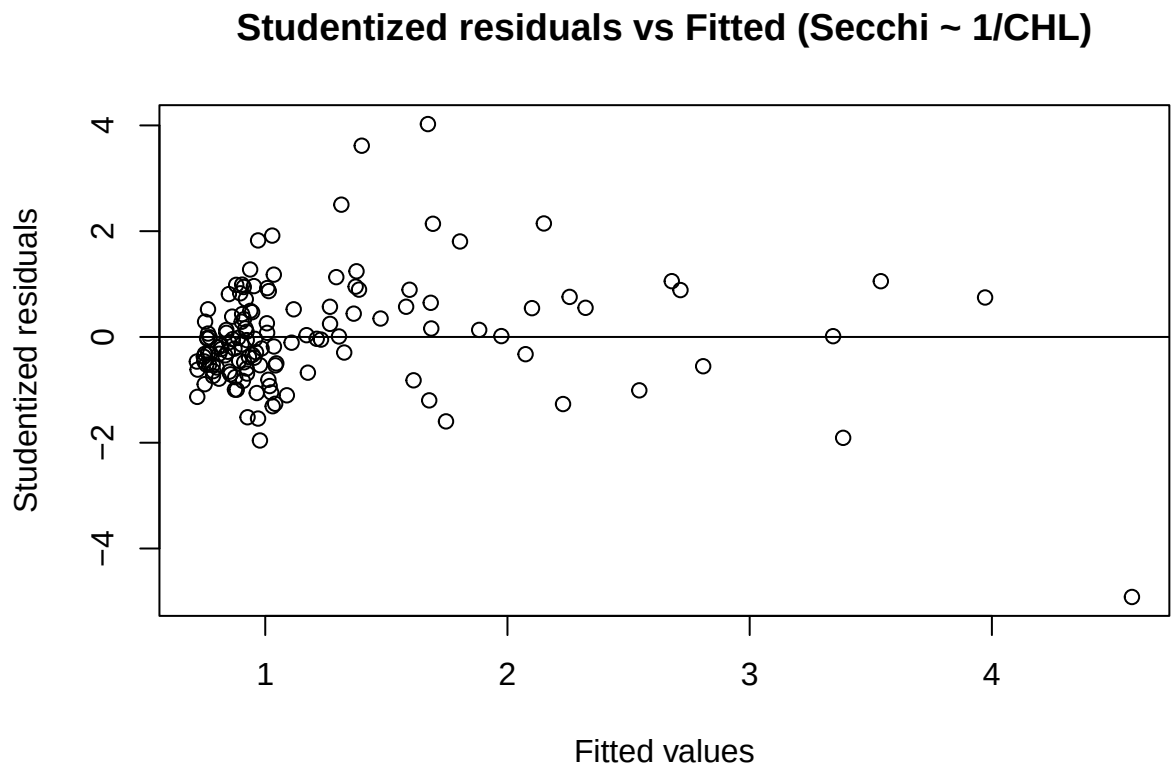
depth is in meters, with a "standard" disc with radius of 30 cm (12 inches)

```
# library(mgcv)
lakesDat$phi <- 1 / lakesDat$chl
fit_phi <- lm(secchi ~ phi, data = lakesDat)

summary(fit_phi)
```

```
##
## Call:
## lm(formula = secchi ~ phi, data = lakesDat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

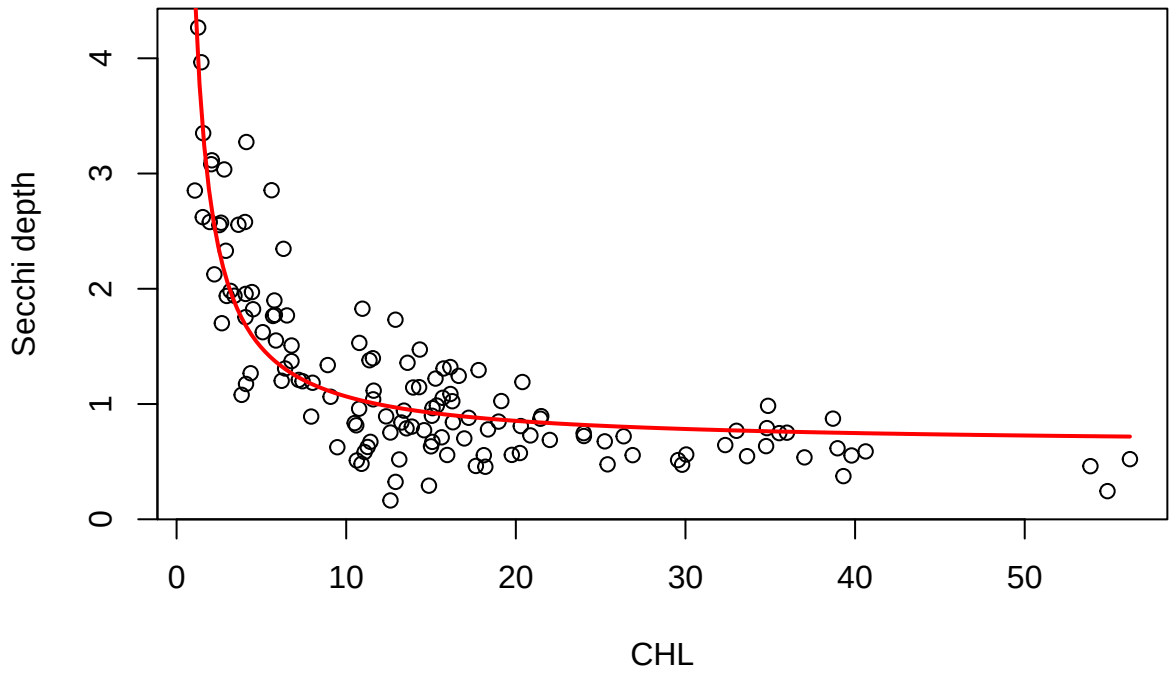
```
## -1.72580 -0.22843 -0.02316  0.22644  1.60189
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.64217    0.04849   13.24   <2e-16 ***
## phi          4.23555    0.22787   18.59   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4225 on 132 degrees of freedom
## Multiple R-squared:  0.7236, Adjusted R-squared:  0.7215
## F-statistic: 345.5 on 1 and 132 DF,  p-value: < 2.2e-16
```

```
stud_resid <- rstudent(fit_phi)
plot(fitted(fit_phi), stud_resid,
     xlab = "Fitted values", ylab = "Studentized residuals",
     main = "Studentized residuals vs Fitted (Secchi ~ 1/CHL)")
abline(h = 0)
```




```
plot(lakesDat$chl, lakesDat$secchi,
     xlab = "CHL", ylab = "Secchi depth")

chl.grid <- seq(min(lakesDat$chl), max(lakesDat$chl), length.out = 200)
pred <- predict(fit_phi,
                newdata = data.frame(phi = 1 / chl.grid))
```

```r
lines(chl.grid, pred, col = "red", lwd = 2)
```

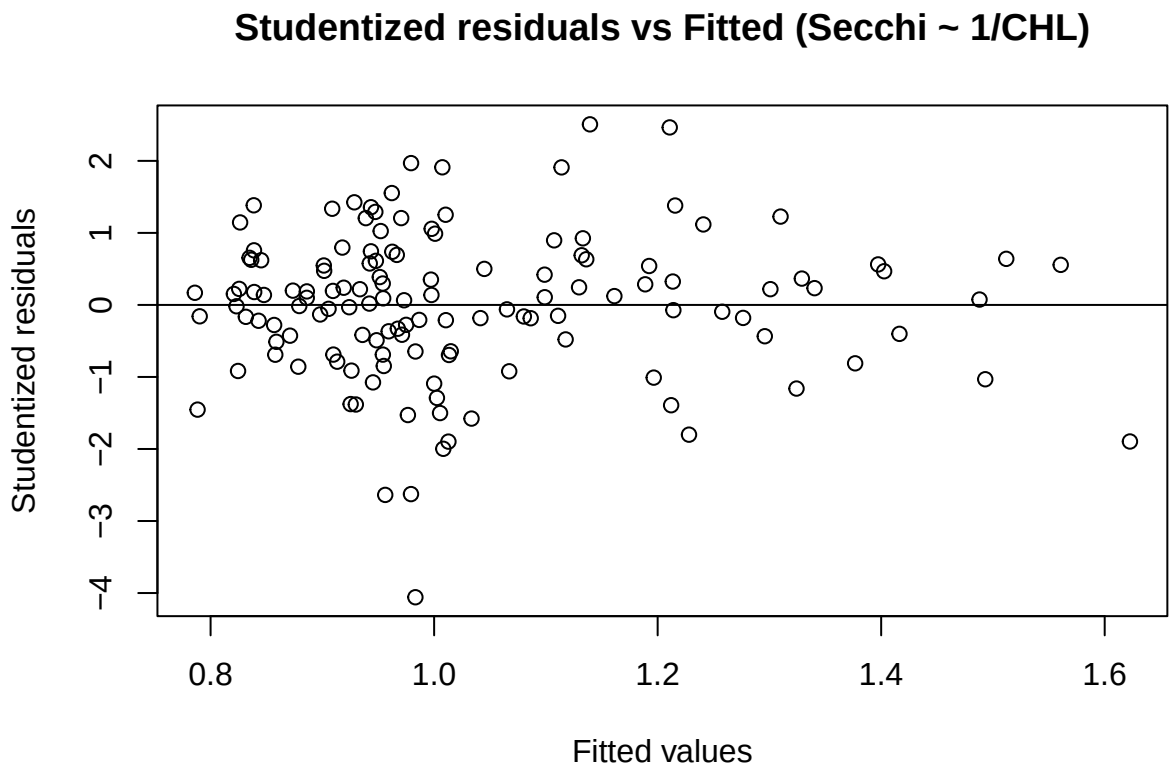



```r
# fit1 <- lm(formula = secchi ~ chl, data = lakesDat)
# fit2 <- lm(formula = secchi ~ chl + chl2, data = lakesDat)

# stud_resid <- rstudent(fit1)
# plot(fit1$fitted.values, stud_resid,
#      xlab = "Fitted Values", ylab = "Studentized Residuals",
#      main = "Studentized Residuals vs Fitted")
# abline(h = 0, col = "black")
#
# plot(x = lakesDat$chl, y = lakesDat$secchi)
# abline(fit1, col = "red")

# stud_resid <- rstudent(fit2)
# plot(fit2$fitted.values, stud_resid,
#      xlab = "Fitted Values", ylab = "Studentized Residuals",
#      main = "Studentized Residuals vs Fitted")
# abline(h = 0, col = "black")
#
# plot(x = lakesDat$chl, y = lakesDat$secchi)
# abline(fit2, col = "red")
```

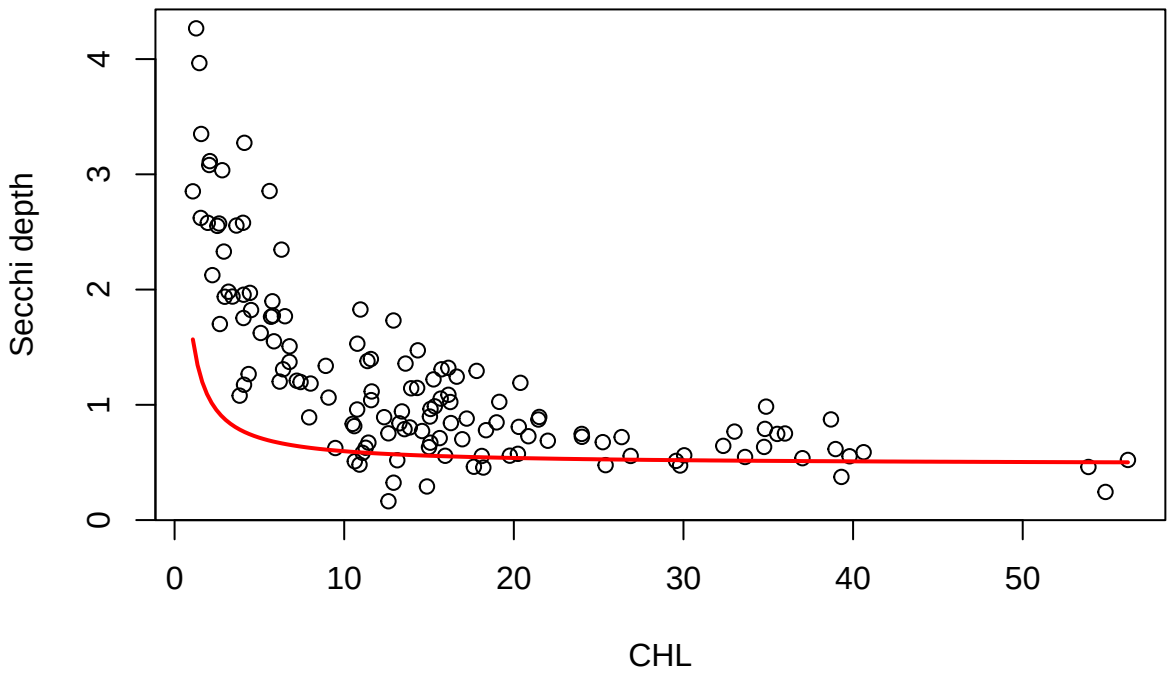
```r
lakesDat$phi <- 1 / lakesDat$chl
lakesDat$secchi_cr <- lakesDat$secchi^(1/3)
lakesDat$phi_cr    <- lakesDat$phi^(1/3)

fit_phi <- lm(secchi_cr ~ phi_cr, data = lakesDat)
summary(fit_phi)
```

```
##
## Call:
## lm(formula = secchi_cr ~ phi_cr, data = lakesDat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.43656 -0.05760  0.01085  0.07025  0.27920
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.48033    0.03213   14.95   <2e-16 ***
## phi_cr       1.17052    0.06511   17.98   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1142 on 132 degrees of freedom
## Multiple R-squared:   0.71,  Adjusted R-squared:  0.7078
## F-statistic: 323.2 on 1 and 132 DF,  p-value: < 2.2e-16
```

```r
stud_resid <- rstudent(fit_phi)
plot(fitted(fit_phi), stud_resid,
     xlab = "Fitted values", ylab = "Studentized residuals",
     main = "Studentized residuals vs Fitted (Secchi ~ 1/CHL)")
abline(h = 0)
```




```r
plot(lakesDat$chl, lakesDat$secchi,
     xlab = "CHL", ylab = "Secchi depth")

chl.grid <- seq(min(lakesDat$chl), max(lakesDat$chl), length.out = 200)
pred <- predict(fit_phi,
                newdata = data.frame(phi_cr = 1 / chl.grid))

lines(chl.grid, pred, col = "red", lwd = 2)
```




```
lakesDat$phi <- 1 / lakesDat$chl
fit_phi <- lm(secchi ~ phi, data = lakesDat)

mu_hat <- fitted(fit_phi)    # estimated means
e_hat  <- resid(fit_phi)     # ordinary residuals

aux <- lm(log(e_hat^2) ~ log(mu_hat))
coef(aux)
```

```
## (Intercept) log(mu_hat)
##   -3.577655    1.227564
```

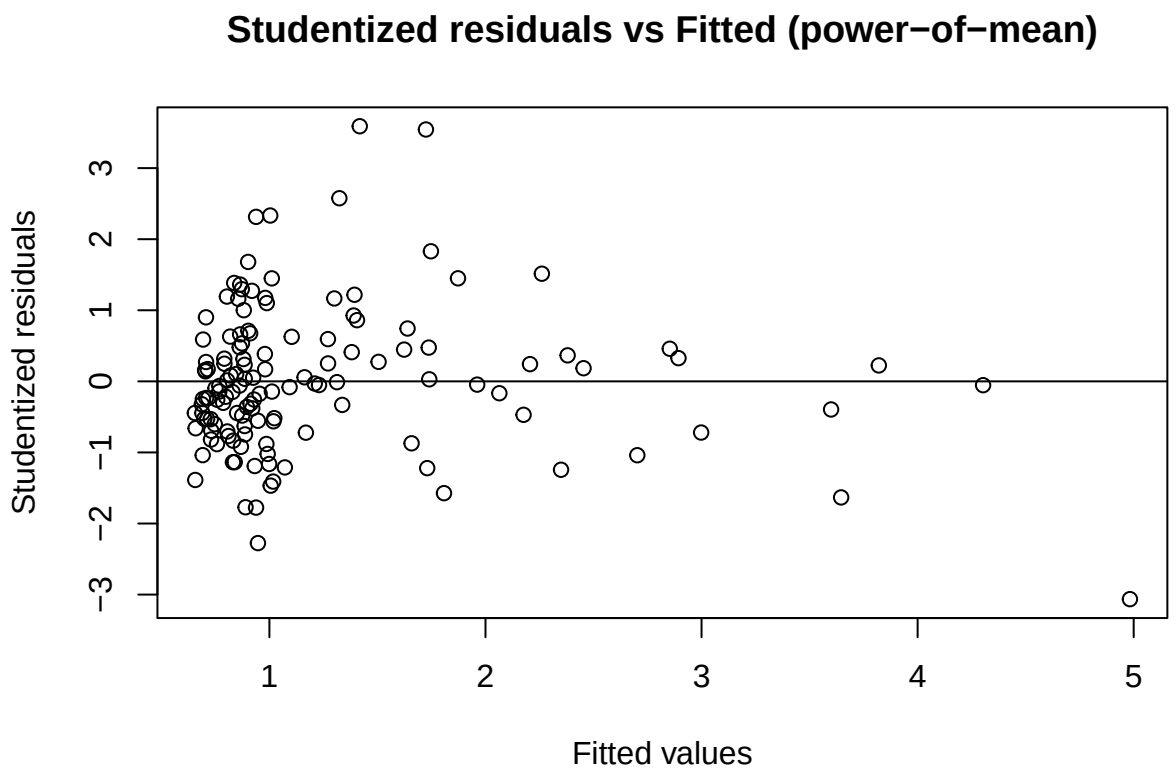
```
delta_hat <- 0.5 * coef(aux)[2]    # slope approx 2delta  ->  delta = slope / 2
delta_hat
```

```
## log(mu_hat)
##   0.6137819
```

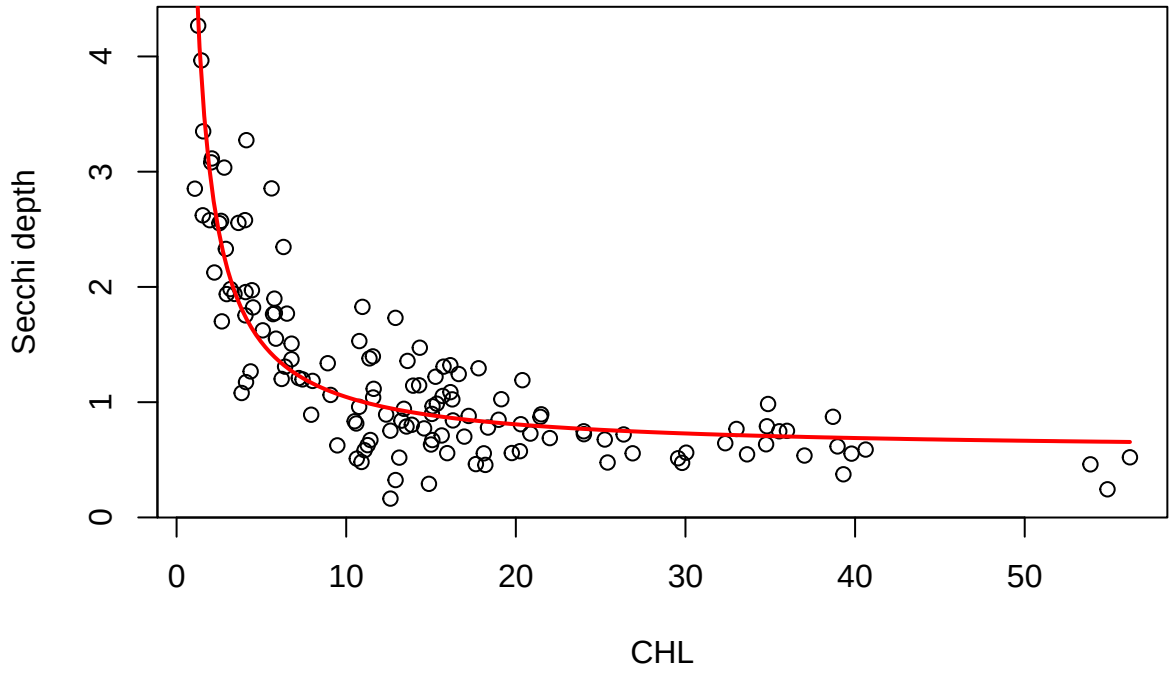
```
delta_hat1 <- 0.5
delta_hat2 <- 0.75
w <- 1 / (mu_hat^(2 * delta_hat1))

fit_phi_pom <- lm(secchi ~ phi, data = lakesDat, weights = w)

summary(fit_phi_pom)
```

```
##
## Call:
## lm(formula = secchi ~ phi, data = lakesDat, weights = w)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -0.99560 -0.20964 -0.01972  0.16966  1.21518
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.57036    0.04246   13.43   <2e-16 ***
## phi          4.74807    0.29918   15.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3551 on 132 degrees of freedom
## Multiple R-squared:  0.6561, Adjusted R-squared:  0.6535
## F-statistic: 251.9 on 1 and 132 DF,  p-value: < 2.2e-16
```

```r
stud_resid2 <- rstudent(fit_phi_pom)
plot(fitted(fit_phi_pom), stud_resid2,
     xlab = "Fitted values",
     ylab = "Studentized residuals",
     main = "Studentized residuals vs Fitted (power-of-mean)")
abline(h = 0)
```



```r
plot(lakesDat$chl, lakesDat$secchi,
     xlab = "CHL", ylab = "Secchi depth")

chl.grid <- seq(min(lakesDat$chl), max(lakesDat$chl), length.out = 200)
pred <- predict(fit_phi_pom,
                newdata = data.frame(phi = 1 / chl.grid))

lines(chl.grid, pred, col = "red", lwd = 2)
```

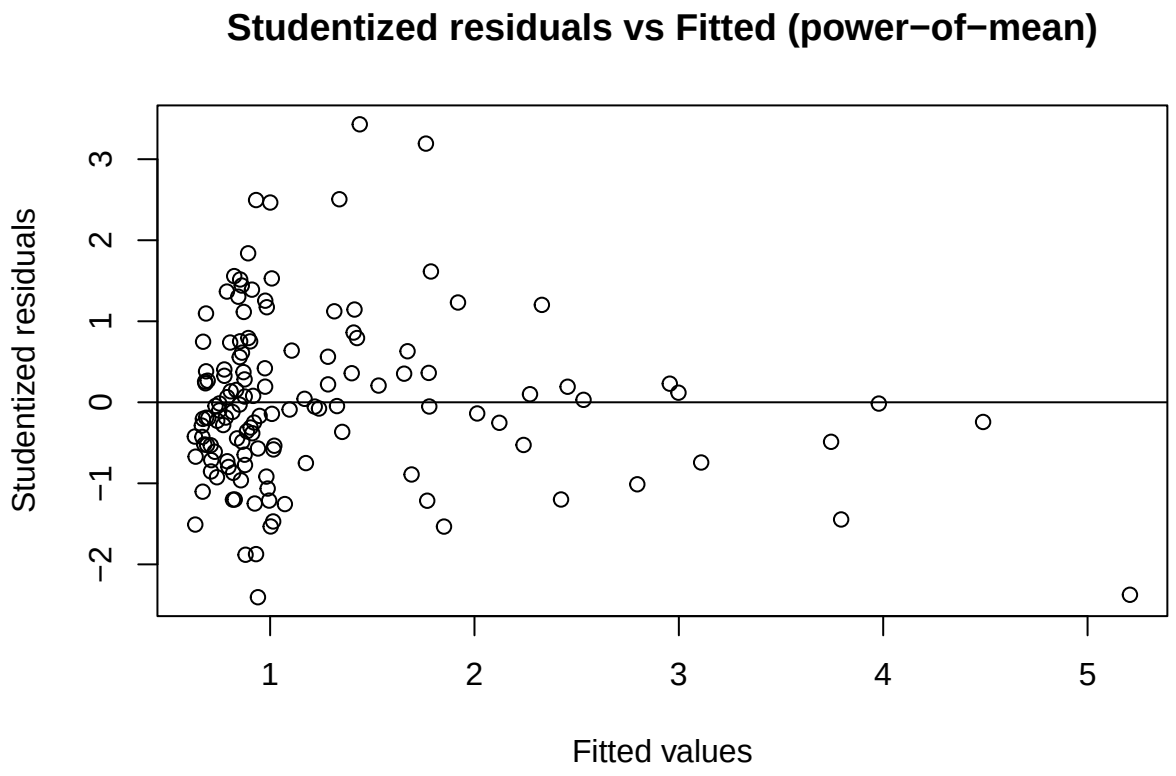



```r
# delta 2
w <- 1 / (mu_hat^(2 * delta_hat2))

fit_phi_pom <- lm(secchi ~ phi, data = lakesDat, weights = w)

summary(fit_phi_pom)
```

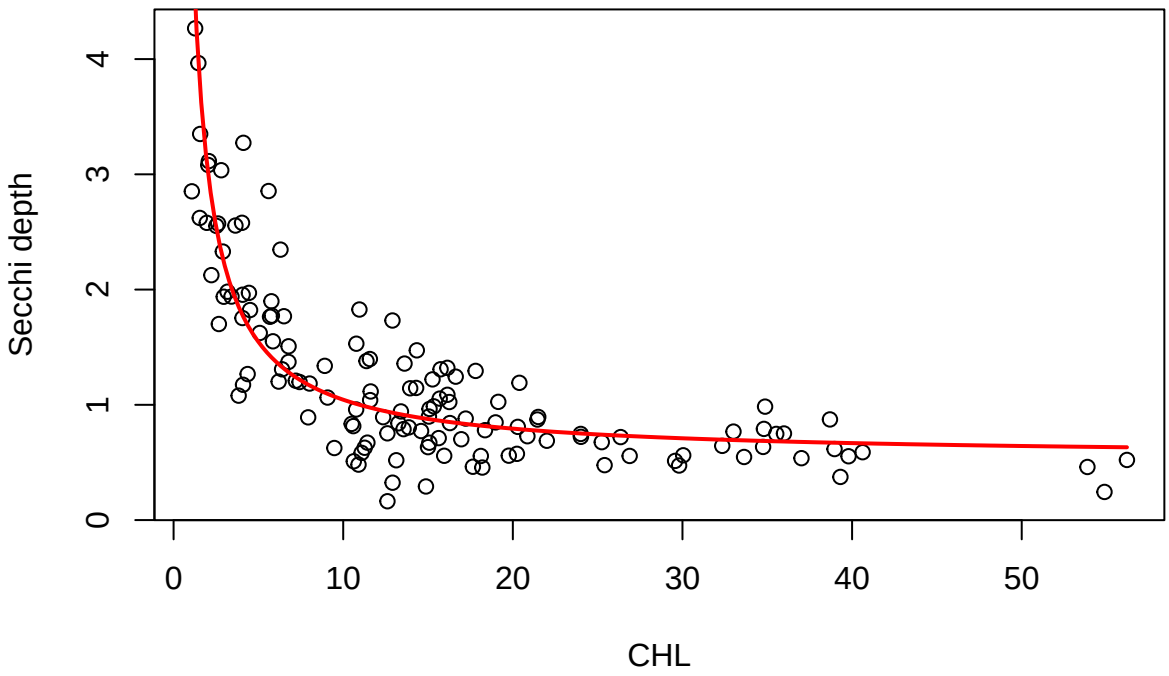
```
##
## Call:
## lm(formula = secchi ~ phi, data = lakesDat, weights = w)
##
## Weighted Residuals:
##      Min       1Q   Median       3Q      Max
## -0.79009 -0.20183 -0.02175  0.17537  1.10152
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.54232    0.04107   13.21   <2e-16 ***
## phi          5.02003    0.34553   14.53   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3357 on 132 degrees of freedom
## Multiple R-squared:  0.6152, Adjusted R-squared:  0.6123
## F-statistic: 211.1 on 1 and 132 DF,  p-value: < 2.2e-16
```

```r
stud_resid2 <- rstudent(fit_phi_pom)
plot(fitted(fit_phi_pom), stud_resid2,
     xlab = "Fitted values",
     ylab = "Studentized residuals",
     main = "Studentized residuals vs Fitted (power-of-mean)")
abline(h = 0)
```




```r
plot(lakesDat$chl, lakesDat$secchi,
     xlab = "CHL", ylab = "Secchi depth")

chl.grid <- seq(min(lakesDat$chl), max(lakesDat$chl), length.out = 200)
pred <- predict(fit_phi_pom,
                newdata = data.frame(phi = 1 / chl.grid))

lines(chl.grid, pred, col = "red", lwd = 2)
```



Or with a cylinder:

$$V = \pi r^2 h \Rightarrow h = \frac{V}{\pi r^2}$$

Q3: LRT SECCHI & CHL (Plains vs. Ozarks)