Understanding Tweedie's Formula

In this notebook I try to implement Tweedie's Formula as presented by Efron.

More specifically, I will try to replicate the simulation in section 3 on page 7.

"One hundred datasets **z**, each of length N=1000, were generated according to"

$$\mu_i \sim e^{-\mu}, (\mu > 0) \quad \text{and} \quad z_i|\mu_i \sim N(\mu_i, 1), \quad \text{fori=1,2,...,1000.}$$

```
# simulation parameters
N <- 1e3  # number of observations per dataset
nsims <- 100  # number of datasets
set.seed(123)
```

I will first go over the case of 1 dataset.

```
# mu_i is distributed as a standard exponential random variable
mu_i <- rexp(N)

# z_i is distributed as a normal normal variable with mean mu_i, variance 1
z_i <- rnorm(N, mean = mu_i, sd = 1)
```

"For each **z**, the curve z+l_hat_prime(z) was computed"

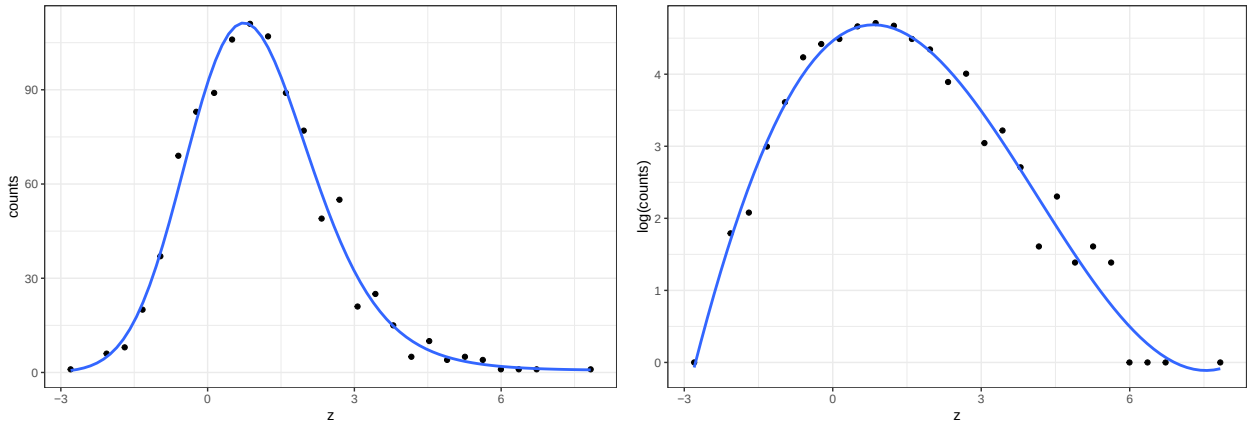
Efron uses a natural spline model with J=5 degrees of freedom to model the counts of the histogram of z-values. I use a 5th degree polynomial GLM with log link function following the earlier description on page 6 (Lindsey's method).

```
# create bins by cut()
# count nr of z_i in bin
# last mutate derives the middle of the bin as mean(left_bound, right_bound)
counts <- cut(z_i, 30) %>%
  as_tibble_col("bins") %>%
  count(bins) %>%
  mutate(logn = log(n)) %>%
  mutate(z = map_dbl(str_extract_all(bins, "-?\\d+\\.?\\d+"), function(x) mean(as.numeric(x)) ) )
```

Visualization of the counts + model and logcounts + model:

```
#visualization of counts + polynomial glm (J = 5) with log link fit
counts %>%
  ggplot(aes(z, n)) +
  geom_point() +
  geom_smooth(method = "glm", formula = y ~ poly(x, 5), se = F,
              method.args = list(family=poisson(link="log"))) +
  labs(y = "counts")

logcounts_plot <- counts %>%
  ggplot(aes(z, logn)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ poly(x, 5), se = F) +
```

```
  labs(y = "log(counts)")
logcounts_plot
```



Now we must find the derivative of this curve.

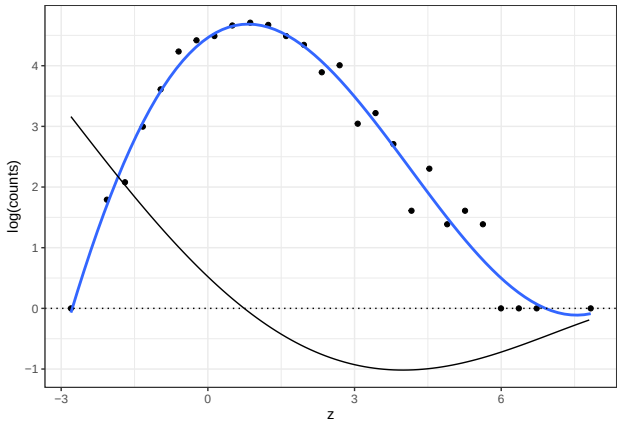
```
# glm fit
fit <- glm(n ~ poly(z, 5, raw = T), family = poisson(link = "log"), data = counts)

# retrieve coefficients
betas <- coef(fit)

# derivative of l_hat(z) or l_hat'(z)
l_prime <- function(x){
  x <- c(1, poly(x, degree = 4, raw=T))

  as.vector(x %*% (betas[2:length(betas)] * 1:5 ))
}
```

Question: I am a bit lost on why fitting a smooth curve to the logcounts is 'enough' (according to the description on pages 6-7) as an estimate of $l(z)$, $\hat{l}(z)$? I thought the curve fitting the logcounts is *proportional* to $\hat{l}(z)$ because $y_k \sim Poi(\nu_k) \quad k = 1, 2, ..., K \quad \nu_k = Ndf_\beta(x_k)$ and not $y_k \sim Poi(f_\beta(x_k))$.

Question: Or perhaps did I fail to execute Lindsey's method properly? On page 6 it says β_0 is determined from $\boldsymbol{\beta}$ by the requirement that $f(z)$ should integrate to 1. However, it is only stated that $MLE\hat{\boldsymbol{\beta}}$ can be calculated using familiar GLM software. I looked up other references (Section 3 of Efron (2008) and Efron and Tibshirani (1996)) only to find +- the same explanation. I'm afraid I don't see (yet) how that should be done then.

Visualization of logcounts model (blue) + derivative (black):

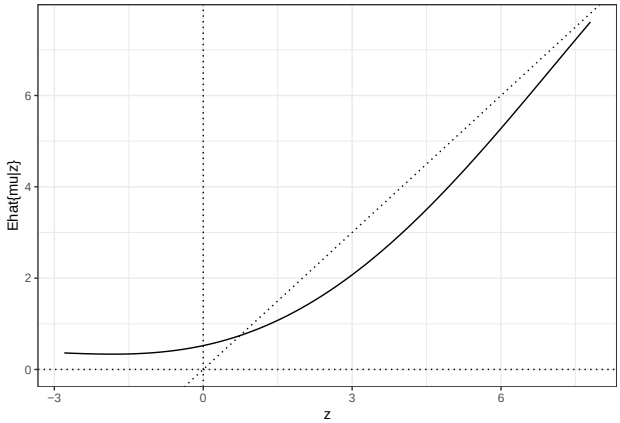
```
derivative <- tibble(z = seq(min(counts$z), max(counts$z), by = 0.1)) %>%
  rowwise() %>%
  mutate(lprime = l_prime(z))

logcounts_plot +
  geom_line(aes(z, lprime), data = derivative, inherit.aes = F) +
  geom_hline(yintercept = 0, linetype = "dotted") # as reference
```



This looks OK.

Finally a visualization of the curve $z + \hat{l}'(z)$ vs z:

```
derivative %>%
  mutate(estimate = z + lprime) %>%
  ggplot(aes(z, estimate)) +
  geom_line() +
  geom_hline(yintercept = 0, linetype = "dotted") +
  geom_vline(xintercept = 0, linetype = "dotted") +
  geom_abline(slope =1, intercept = 0, linetype = "dotted") +
  labs(y = "Ehat{mu|z}")
```



Now we do this 100 times and

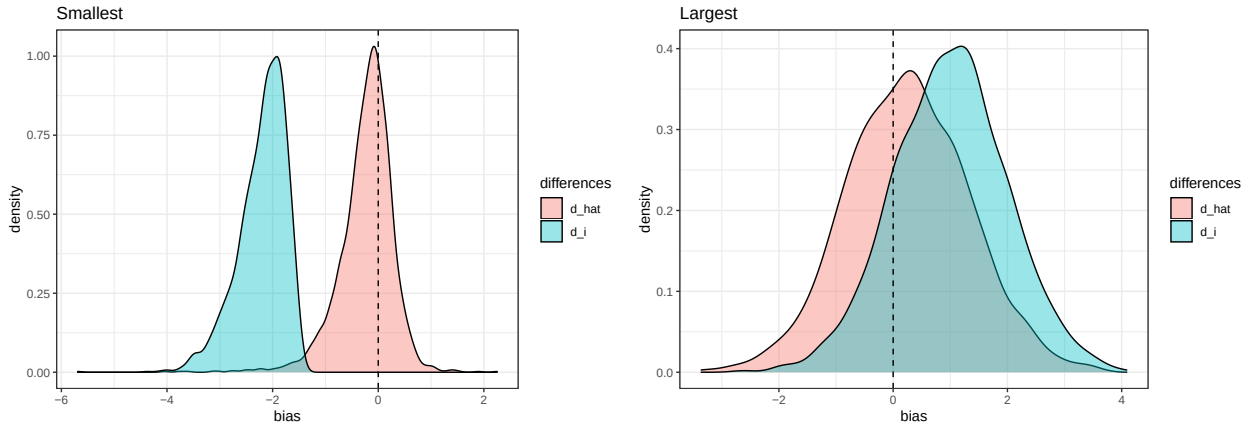
> "the corrected estimates mu_hat_i = z_i + l_hat_prime(z_i) were calculated for the 20 largest and 20 smallest z_i's"

```
# create_smallest_and_largest() does the above steps and outputs 20 smallest and largest
smallest_results <- tibble()
largest_results <- tibble()
for (i in 1:nsims){
  results <- create_smallest_and_largest(i)

  smallest_results <- smallest_results %>%
    bind_rows(results$smallest)

  largest_results <- largest_results %>%
```

```
   bind_rows(results$largest)
}
```

And we plot the results.

```
smallest_results %>%
  mutate(d_i = z_i-mu_i,
         d_hat = corrected - mu_i) %>%
  pivot_longer(c(d_i, d_hat)) %>%
  ggplot(aes(value, fill = name)) +
  geom_density(alpha = 0.4) +
  labs(x = "bias",
       fill = "differences",
       title = "Smallest") +
  geom_vline(xintercept = 0, linetype = "dashed")

largest_results %>%
  mutate(d_i = z_i-mu_i,
         d_hat = corrected - mu_i) %>%
  pivot_longer(c(d_i, d_hat)) %>%
  ggplot(aes(value, fill = name)) +
  geom_density(alpha = 0.4) +
  labs(x = "bias",
       fill = "differences",
       title = "Largest") +
  geom_vline(xintercept = 0, linetype = "dashed")
```



Indeed, the corrected differences (red) are closer to 0 than the uncorrected (blue).

Remark: Given the fact that I think I missed some things (see questions above), I'm quite surprised with the results, as they do seem in line with Efron.

Question: σ was assumed to be known. What happens when we have to estimate σ?

References

Efron, Bradley. 2008. "Microarrays, Empirical Bayes and the Two-Groups Model." *Statistical Science* 23 (1): 1–22. https://doi.org/10.1214/07-sts236.

Efron, Bradley, and Robert Tibshirani. 1996. "Using Specially Designed Exponential Families for Density Estimation." *The Annals of Statistics* 24 (6): 2431–61. https://doi.org/10.1214/aos/1032181161.