MKTG776 HW3

Jordan Farrer

2017-02-07

Contents

1 Question 1

We will take by definition that the mean (first moment) of the NBD model is

$$E[X] = \bar{x} = \frac{r}{\alpha} \tag{1}$$

and the variance (second moment) of the NDB model is

$$Var[X] = s^2 = \frac{r}{\alpha} + \frac{r}{\alpha^2} \tag{2}$$

We can substitute (1) into (2) to get

$$s^2 = \bar{x} + \frac{\bar{x}}{\alpha} \tag{3}$$

Now we can simply rearrange (3) to estimate the model parameter *alpha* as a function of the mean and variance:

$$\hat{\alpha} = \frac{\bar{x}}{s^2 - \bar{x}} \tag{4}$$

The model parameter r is a bit easier, we can simply rearrange (1) to find

$$\hat{r} = \hat{\alpha}\bar{x} \tag{5}$$

2 Question 2

We first load the provided prescription data. Below are all 16 records:

```
pacman::p_load(tidyverse, forcats, pander, ggrepel)
panderOptions('round', 2)
panderOptions('keep.trailing.zeros', TRUE)
options(scipen = 10)

prescription_data <- readxl::read_excel("HW prescription data.xls")

prescription_data %>%
  pander(caption = "Raw Prescription Data")
```

Table 1: Raw Prescription Data

x	n_x
0	1650
1	138
2	55
3	30
4	15
5	10
6	8
7	4
8	6
9	3
10	1
11	0
12	1
13	1
14	0
15	1

2.1 NBD

2.1.1 MLE

```
# For Zero-inflated Negative Binomial Distribution, calculates P(X=x)
fn_zinbd <- function(x, r, alpha, pi) {
  p_x <- (gamma(r + x) / (gamma(r) * factorial(x))) * (alpha / (alpha + 1))^r * (1 / (alpha + 1))^x
  if(x == 0) {
    return(pi + (1 - pi) * p_x)
  } else {
    return((1 - pi) * p_x)
  }
}

# Calculates the log-likelihood of the NBD (including
# zero-inflated)
fn_max_ll <- function(par, zero_inflated = FALSE, counts) {
  r <- par[1]
  alpha <- par[2]
  if (zero_inflated) {
    pi <- par[3]
  } else {
    pi <- 0
  }
  ll <- sum(log(sapply(counts, fn_zinbd, r, alpha, pi)))

  return(-ll)
}

counts <-
  prescription_data %>%
    rename(times = n_x) %>%
    invoke_rows(.f = rep, .collate = 'rows') %>%
    select(count = .out) %>%
    unlist() %>%
    unname()

params_nbd <- nlminb(c(1, 1), fn_max_ll, lower = c(0, 0), upper = c(Inf, Inf),
                 zero_inflated = FALSE, counts = counts)
```

Table 2: MLE

parameter	value
r	0.1079
alpha	0.3197

2.1.2 Method of Moments

```
alpha_mom <- mean(counts) / (sd(counts)^2 - mean(counts))
r_mom <- alpha_mom * mean(counts)
```

Table 3: Method of Moments

parameter	value
r	0.1095
alpha	0.3244

2.1.3 Means and Zeros

```
fn_means_and_zeros <- function(par, counts) {
  alpha <- par[1]
  f = ((alpha / (alpha + 1))^(alpha * mean(counts)) - sum(counts == 0) / length(counts))^2

}

alpha_maz <- nlminb(c(1), fn_means_and_zeros, lower = c(0), upper = c(Inf), counts = counts)$par[1]
r_maz <- alpha_maz * mean(counts)
```

Table 4: Means and Zeros

parameter	value
r	0.1081
alpha	0.3203

2.1.4 Comparison

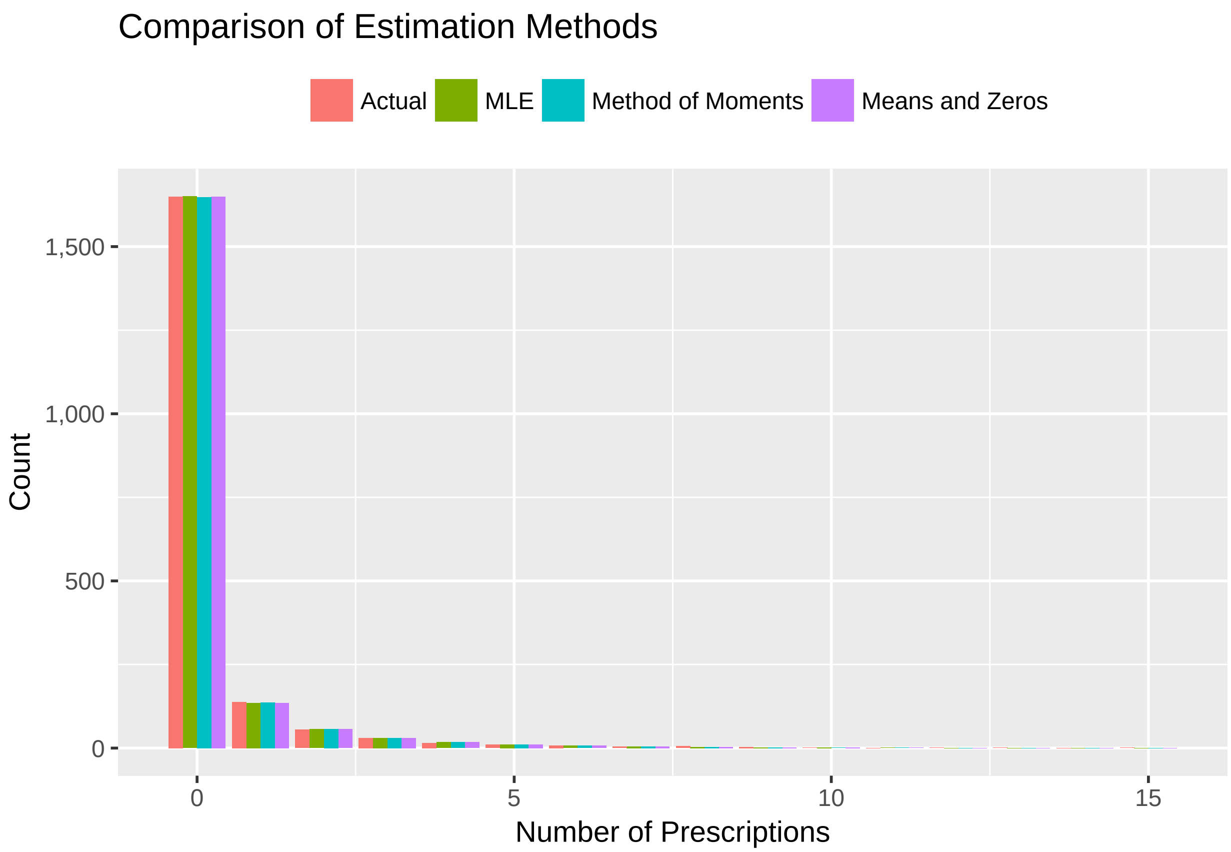
The table below shows the estimates for *alpha* and r using the three estimation methods. We see that the results are remarkably similar. The smallest difference appears to be between the MLE and Means and Zeros methods.

```
data_frame(
  Method = c("MLE","Method of Moments", "Means and Zeros")
  , alpha = c(params_nbd$par[2], alpha_mom, alpha_maz)
  , r = c(params_nbd$par[1], r_mom, r_maz)
) %>%
  pander(caption = c("NBD Model with Different Estimation Methods"),
         round = 4)
```

Table 5: NBD Model with Different Estimation Methods

Method	alpha	r
MLE	0.3197	0.1079
Method of Moments	0.3244	0.1095
Means and Zeros	0.3203	0.1081

```
prescription_data %>%
  rename(Actual = n_x) %>%
  mutate(
    "MLE" = sapply(x, fn_zinbd, params_nbd$par[1], params_nbd$par[2], 0) * sum(Actual)
    , "Method of Moments" = sapply(x, fn_zinbd, r_mom, alpha_mom, 0) * sum(Actual)
    , "Means and Zeros" = sapply(x, fn_zinbd, r_maz, alpha_maz, 0) * sum(Actual)
  ) %>%
  gather(method, value, -x) %>%
  mutate(method = factor(method, levels = c('Actual','MLE','Method of Moments',
                                            'Means and Zeros'))) %>%
  ggplot() +
  geom_bar(aes(x = x, y = value, fill = method), stat = 'identity', position = 'dodge') +
  labs(y = "Count", x = "Number of Prescriptions", fill = NULL,
       title = "Comparison of Estimation Methods") +
  scale_x_continuous(labels = scales::pretty_breaks()) +
  scale_y_continuous(labels = scales::comma) +
  theme(legend.position = 'top')
```



2.2 Zero-Inflated NBD

Without even performing the estimation of the zero-inflated NBD, we imagine that the value of π will be close to zero because the NBD model alone closely fit the data.

```r
params_zinbd <- nlminb(c(1, 1, .5), fn_max_ll, lower = c(0, 0, 0), upper = c(Inf, Inf, 1),
                 zero_inflated = TRUE, counts = counts)
```

We perform the estimation and see that $\pi = 0$ in the zero-inflated NBD and thus the two other model parameters, *alpha* and r, are the same.

```r
data_frame(
  Method = c("NBD", "Zero-Inflated NBD")
  , r = c(params_nbd$par[1], params_zinbd$par[1])
  , alpha = c(params_nbd$par[2], params_zinbd$par[2])
  , pi = c(NA, params_zinbd$par[3])
) %>%
  pander(caption = c("MLE for NBD and Zero-Inflated NBD"),
         round = 4, missing = "")
```

Table 6: MLE for NBD and Zero-Inflated NBD

Method	r	alpha	pi
NBD	0.1079	0.3197	
Zero-Inflated NBD	0.1079	0.3197	0

Next, we attempt to perform the χ^2 goodness-of-fit test. However, we immediately see that most (9 of 16) cells have fewer than 5 expected counts. This violates the traditional rule-of-thumb of 80% of cells must have 5 or more expected counts.

```r
prescription_data %>%
  rename(Actual = n_x) %>%
  mutate(
    Expected = sapply(x, fn_zinbd, params_nbd$par[1], params_nbd$par[2], 0) * sum(Actual)
    , chi.squared = (Actual - Expected)^2 / Expected
  ) %>%
  pander(caption = "Actual vs Expected for Zero-Inflated NBD")
```

Table 7: Actual vs Expected for Zero-Inflated NBD

x	Actual	Expected	chi.squared
0	1650	1650.24	0.00
1	138	134.91	0.07
2	55	56.63	0.05
3	30	30.15	0.00
4	15	17.75	0.43
5	10	11.05	0.10
6	8	7.13	0.11
7	4	4.71	0.11
8	6	3.17	2.52
9	3	2.17	0.32
10	1	1.50	0.16
11	0	1.04	1.04
12	1	0.73	0.10
13	1	0.52	0.46
14	0	0.37	0.37
15	1	0.26	2.10

To remedy this, we roll-up to 10+. Unfortunately, this still violates our rule-of-thumb as 3/11 cells have less than 5 expected counts. With reserved expectations, we carry out the test anyway.

```r
summary_nbd <-
  prescription_data %>%
    rename(Actual = n_x) %>%
    mutate(
        p_x = sapply(x, fn_zinbd, params_nbd$par[1], params_nbd$par[2], 0)
    ) %>%
    mutate(
      x_factor = if_else(x < 10, as.character(x), "10+")
      , x_factor = factor(x_factor, levels = c(as.character(0:9), "10+"))
      , p_x2 = if_else(x < 10, p_x, 0)
      , p_x3 = if_else(x < 10, p_x, 1 - sum(p_x2))
      , Expected = p_x3 * sum(Actual)
    ) %>%
    group_by(x_factor, Expected) %>%
    summarise(Actual = sum(Actual)) %>%
    mutate(chi.squared = (Actual - Expected)^2 / Expected) %>%
    select(x = x_factor, Actual, Expected, chi.squared)

summary_nbd %>%
  pander(caption = "Actual vs Expected for Zero-Inflated NBD, Truncated Right Tail")
```

Table 8: Actual vs Expected for Zero-Inflated NBD, Truncated Right Tail

x	Actual	Expected	chi.squared
0	1650	1650.24	0.00
1	138	134.91	0.07
2	55	56.63	0.05
3	30	30.15	0.00
4	15	17.75	0.43
5	10	11.05	0.10
6	8	7.13	0.11
7	4	4.71	0.11
8	6	3.17	2.52
9	3	2.17	0.32
10+	4	5.08	0.23

The p-value for the χ^2 goodness-of-fit test indicates that we have no evidence to believe that data and the model's expected values come from separate population. In others words, the model fit is good.

```r
p_value_gof <- pchisq(sum(summary_nbd$chi.squared), df = 11-3-1, lower.tail = FALSE)
```

p-value $= 0.7881391$

2.3 Likelihood Ratio Test

The likelihood ratio test with the null hypothesis that spike (π) is equal to 0, can be performed using

```r
ll <- fn_max_ll(params_nbd$par, zero_inflated = FALSE, counts)
ll_zi <- fn_max_ll(params_zinbd$par, zero_inflated = TRUE, counts)

lrt_stat <- 2 * (ll_zi - ll)
p_value_lrt <- pchisq(lrt_stat, df = 1, lower.tail = FALSE)
```

The p-value $= 0.9999827$ indicates that we have no evidence to reject the null hypothesis that spike (π) is equal to 0, as expected.

2.4 Model Selection

Based on the distribution in 2.1, which model we select may not be that relevant, so we select the NBD from MLE.

Using a recursive implementation of the NBD for an arbitrary t, we create the following distribution for the number of prescriptions over a 12-month period:

```r
fn_nbd_recursive <- function(x, r, alpha, t) {
  if (x == 0) {
    p_x <- (alpha / (alpha + t))^r
  } else {
    p_x <- (t * (r+x-1)) / (x * (alpha + t)) * fn_nbd_recursive(x - 1, r, alpha, t)
  }
  return(p_x)
}

prescription_data %>%
  rename(Actual = n_x) %>%
  mutate(
    Expected = sapply(x, fn_nbd_recursive, params_nbd$par[1], params_nbd$par[2], 12) * sum(Actual)
  ) %>%
  select(x, Expected) %>%
  pander(caption = "Expected distribution for the number of prescriptions over a 12-month period")
```

Table 9: Expected distribution for the number of prescriptions over a 12-month period

x	Expected
0	1296.82
1	136.28
2	73.53
3	50.33
4	38.09
5	30.48
6	25.27
7	21.48
8	18.59
9	16.31
10	14.47
11	12.95
12	11.68
13	10.60
14	9.66
15	8.85

3 Question 3

We create a dataset of the toliet paper data:

```
tp <-
  data_frame(
    brand = factor(c("Charmin", "Angel Soft", "Private Label", "Category"),
                   levels = c("Charmin", "Angel Soft", "Private Label", "Category"))
    , penetration = c(0.4262, 0.2960, 0.2572, 0.9)
    , purchase_per_buyer = c(4.25, 3.55, 3.97, 9.55)
  )
```

brand	penetration	purchase_per_buyer
Charmin	0.43	4.25
Angel Soft	0.30	3.55
Private Label	0.26	3.97
Category	0.90	9.55

We then perform the means and zeros to find r and α for each brand and the category.

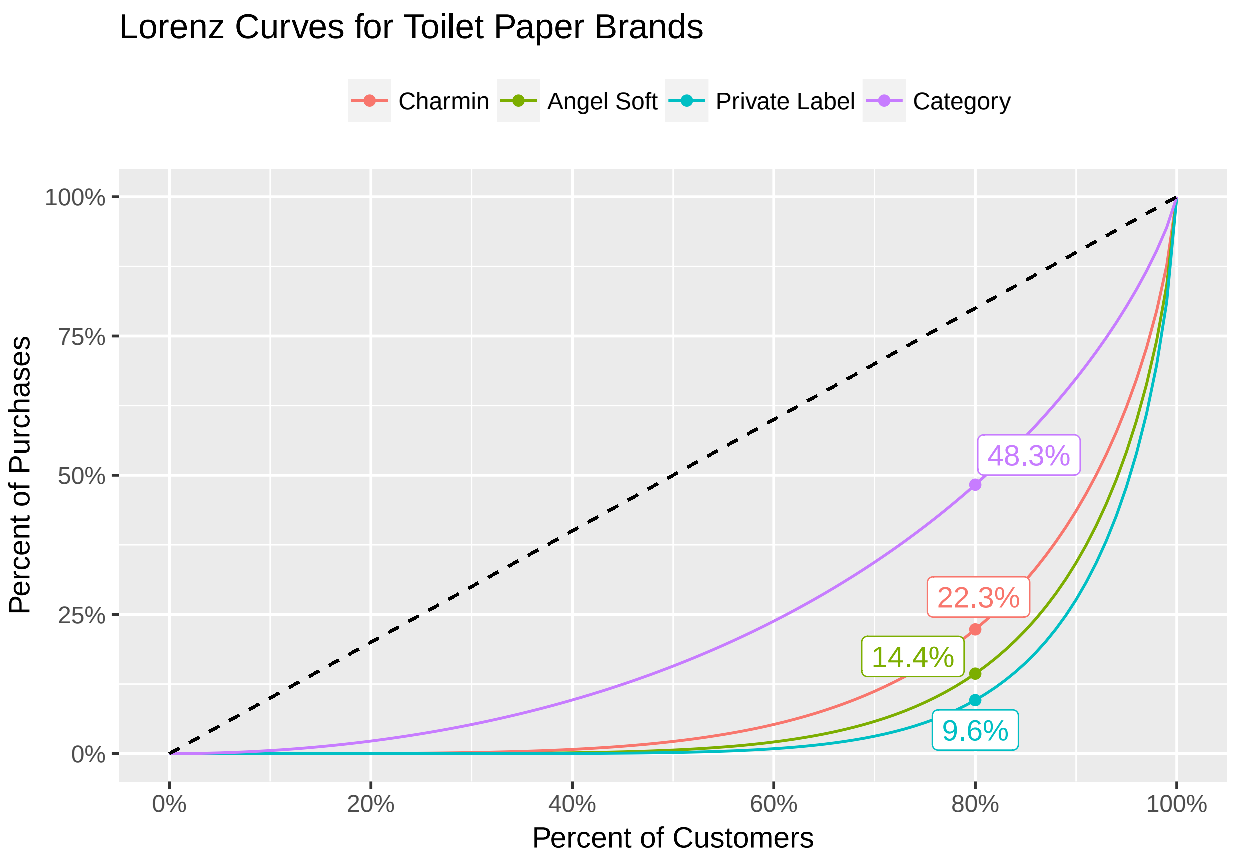
```
fn_means_and_zeros_aggregate <- function(par, mean, zeros) {
  alpha <- par[1]
  return(((par[1] / (par[1] + 1))^(par[1] * mean) - zeros)^2)
}

fn_means_and_zeros_alpha <- function(mean, zeros) {
  nlminb(start = c(1), objective = fn_means_and_zeros_aggregate,
         lower = c(0), upper = c(Inf), mean = mean, zeros = zeros)$par[1]
}

tp_mean_and_zeros <-
  tp %>%
    mutate(
      alpha = map2_dbl(purchase_per_buyer * penetration, 1 - penetration, fn_means_and_zeros_alpha)
      , r = alpha * (purchase_per_buyer * penetration)
    )
```

Table 11: Parameter Estimation Using Means and Zeros for Toilet Paper Brands

brand	penetration	purchases per buyer	r	alpha
Charmin	0.43	4.25	0.27	0.15
Angel Soft	0.30	3.55	0.18	0.18
Private Label	0.26	3.97	0.14	0.14
Category	0.90	9.55	1.03	0.12

Then we plot the Lorenz curves:

```
Lp <-
  tp_mean_and_zeros %>%
  crossing(p = seq(from = 0, to = 1, by  = 0.01)) %>%
  mutate(L_p = pgamma(qgamma(p, r, 1), r + 1))

rule_8020 <-
  Lp %>%
  filter(p == 0.8)

Lp %>%
  ggplot(aes(x = p, y = L_p, colour = brand)) +
  geom_line() +
  geom_segment(aes(x = 0, xend = 1, y = 0, yend = 1), colour = "black", linetype = "dashed", size = .5) +
  labs(x = "Percent of Customers", y = "Percent of Purchases", title = "Lorenz Curves for Toilet Paper Brands",
       colour = NULL) +
  scale_x_continuous(breaks = scales::pretty_breaks(), labels = scales::percent) +
  scale_y_continuous(labels = scales::percent) +
  theme(legend.position = "top") +
  geom_point(data = rule_8020, aes(wx = p, y = L_p, colour = brand)) +
  geom_label_repel(data = rule_8020, aes(wx = p, y = L_p, colour = brand, label = scales::percent(L_p)), show.legen
```

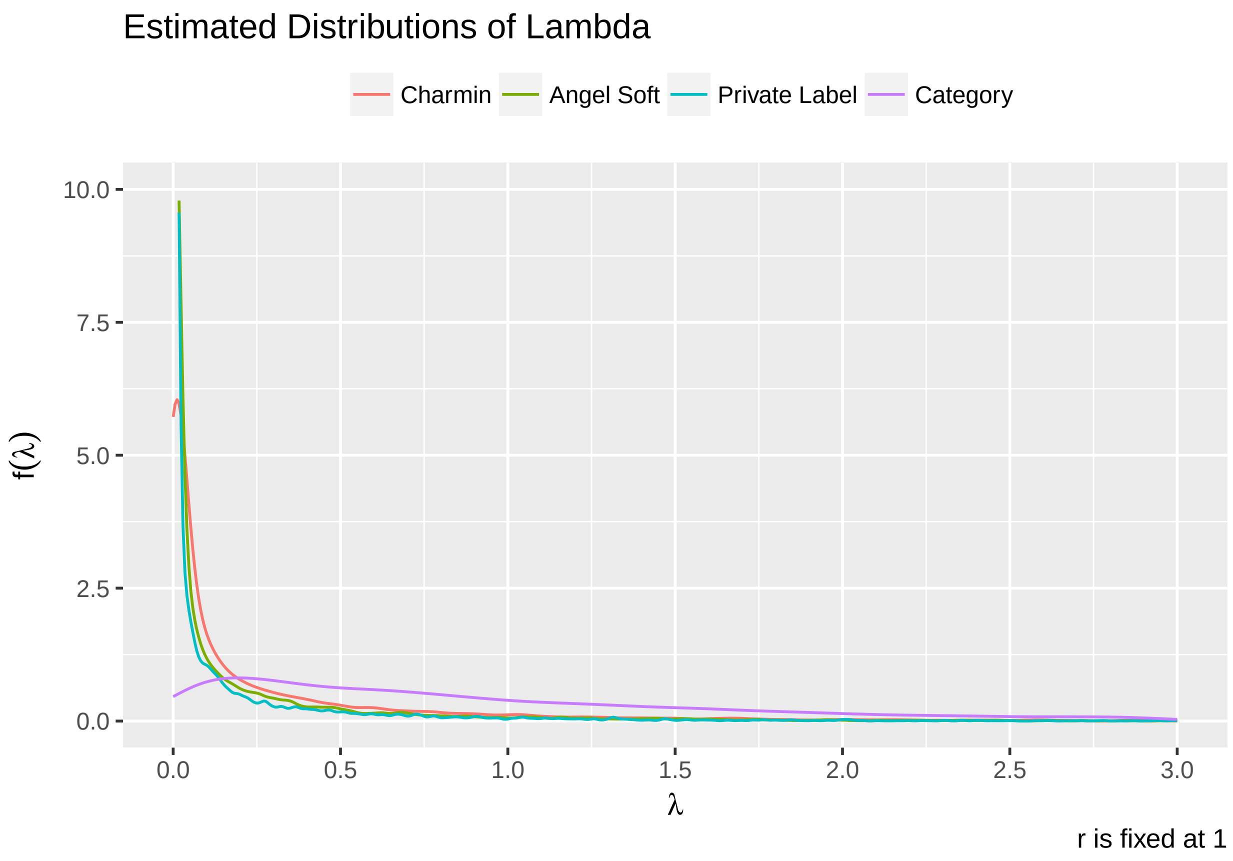


We find that in 1996, the most concentrated brand is Private Label products and the least concentrated brand is Charmin. Specifically, 80% of Private Label customers account for only 9.6% of the purchases (i.e. 20% accounts for 90.4%) while for Charmin 80% of the customers account for 22.3% of the purchases (and thus 20% account for 77.3%). Angel Soft is in the middle: 80% of the customers account for 14.4% of purchases. The more "bowed" the Lorenz curve is the more concentrated the purchasing within the brand.

When comparing the concentrations of the brands to the concentration of the toilet paper category, the category appears much less concentrated. At the category-level, 80% of the customer account for 48.3% of the purchases. This makes intuitive sense because brand loyalty exists (in the case of Private Label it's loyalty to cheapest price). There are some people who only buy Charmin or Angel Soft, but there are not only some people that buy toilet paper. Everyone buys toilet paper (or 90% from drug stores and groceries), but not everyone buys the same amount. This is evidenced by the bowed nature of the purple curve above. There are households that buy more than others.

This comparison indicates that there is less customer heterogeneity at the brand-level than there is at the category-level. In other words, at the brand-level there are a select few that have large λ's and instead most people have small λ's. In contrast, at the category-level, this is slightly more dispersion. This contrast can be seen directly by looking at the value of the shape parameter r in the table above or at the estimated (gamma) distribution of λ holding the scale parameter α constant.

```
tp_mean_and_zeros %>%
  mutate(
    gamma = map(r, function(.x, .y) {rgamma(10000, .x, 1)})
  ) %>%
  unnest() %>%
  ggplot(aes(x = gamma, colour = brand)) +
  geom_line(stat = "density") +
  theme(legend.position = "top") +
  scale_x_continuous(breaks = scales::pretty_breaks(), limits = c(NA, 3)) +
  scale_y_continuous(limits = c(NA, 10)) +
  labs(x = expression(lambda), y = expression(f(lambda)), colour = NULL,
       title = "Estimated Distributions of Lambda", caption = "r is fixed at 1")
```



4 Question 4

4.1 MAU for 80:20 Rules

```
fn_find_mau <- function(par) {
  dau = 178/305
  mau <- par[1]

  p0_day = 1 - dau
  p0_mon = 1 - mau

  alpha <- par[2]
  r <- log(p0_day) / log(alpha / (alpha + 1))
  p0_mo_est <- (alpha / (alpha + 30.5))^r

  f <- (pgamma(qgamma(.8, r, 1), r + 1) - .2)^2 + (p0_mon - p0_mo_est)^2
  return(f)
}

params_fb <- nlminb(start = c(.25, 1), objective = fn_find_mau, lower = c(0, 0), upper = c(1, Inf), control = list(
mau <- params_fb$par[1]
```

Using DAU data from Monday's class (178/305m), the MAU have to be **250** or (250/ 305) in order for Facebook to conform perfectly to the 80:20 rule. This would imply that 80% of Facebook users account for 20% of the visits. From class the number of MAUs was 229.

4.2 Weekly

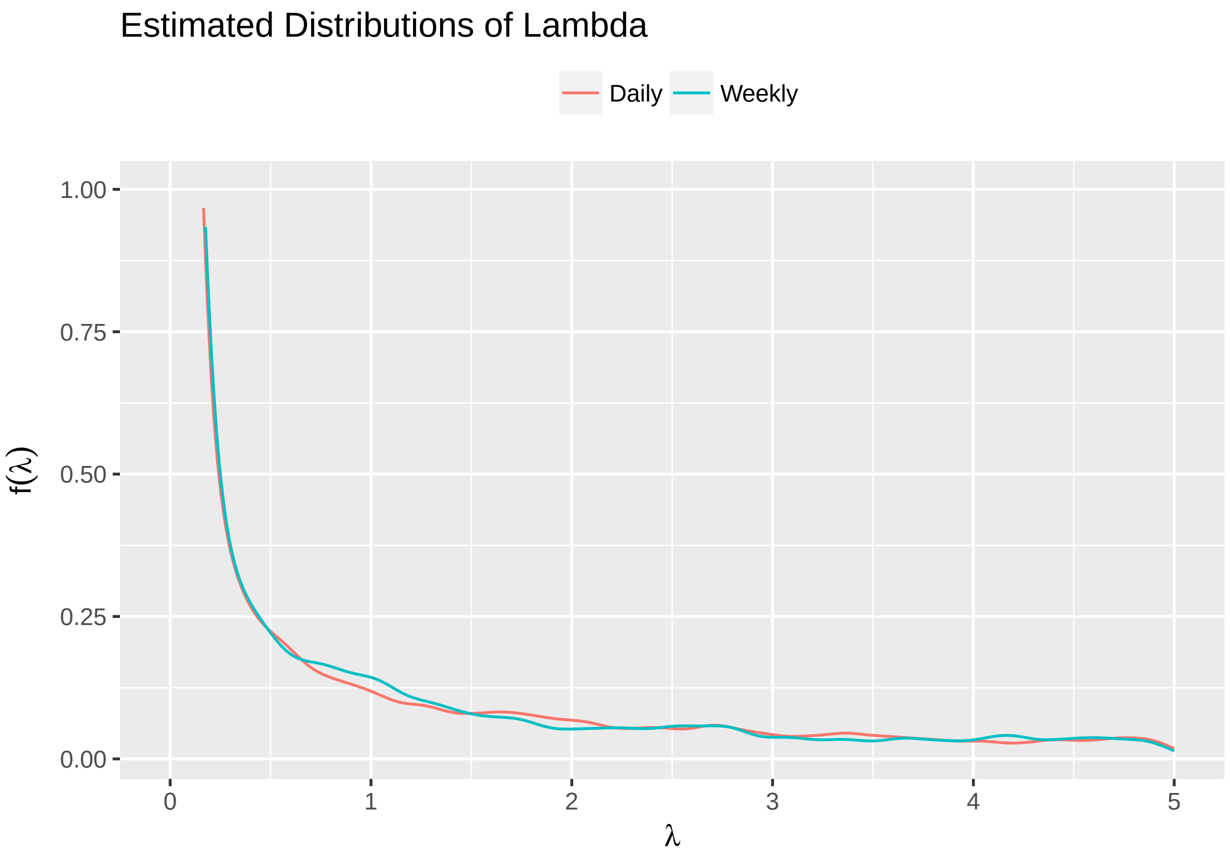
Now we need to use the following formula to convert from day to week, were $t = \frac{13}{3}$:

$$P(X(t) = x) = \begin{cases} \left(\frac{\alpha}{\alpha+t}\right)^r & x = 0 \\ \frac{t(r+x-1)}{x(\alpha+t)} P(X(t) = x-1) & x = 1, 2, 3, \ldots \end{cases} \tag{6}$$

```
fn_alpha_for_t_optim <- function(par, period) {
  dau = 178/305
  mau <- 229/305

  p0_day = 1 - dau
  p0_mon = 1 - mau

  t_to_month <- ifelse(period == "Weekly", 13/3, 30.5)
  t <- ifelse(period == "Weekly", 1/7, 1)
  #t = 1

  alpha <- par[1]
  r <- log(p0_day) / log(alpha / (alpha + t))
  p0_mo_est <- (alpha / (alpha + t_to_month))^r

  f <-  (p0_mon - p0_mo_est)^2
  return(f)
}

fn_alpha_for_t <- function(period) {
 alpha <- nlminb(start = c(1), objective = fn_alpha_for_t_optim, period = period, lower = 0, upper = Inf)$par[1]
 return(alpha)
}

dau_model_param <-
  data_frame(
    period = c("Daily", "Weekly")
  ) %>%
  mutate(
    alpha = map_dbl(period, fn_alpha_for_t)
    , r = log(1 - 178/305) / log(alpha / (alpha + ifelse(period == "Weekly", 1/7, 1)))
  ) %>%
  select(period, r, alpha)

dau_model_param %>%
  pander(round = 4)
```

period	r	alpha
Daily	0.1504	0.0030
Weekly	0.1506	0.0004

```r
dau_model_param %>%
  mutate(
    gamma = map2(r, alpha, function(.x, .y) {rgamma(10000, .x, .y)})
  ) %>%
  unnest() %>%
  ggplot(aes(x = gamma, colour = period)) +
  geom_line(stat = "density") +
  theme(legend.position = "top") +
  scale_x_continuous(breaks = scales::pretty_breaks(), limits = c(NA, 5)) +
  scale_y_continuous(limits = c(NA, 1)) +
  labs(x = expression(lambda), y = expression(f(lambda)), colour = NULL,
       title = "Estimated Distributions of Lambda")
```


Estimated Distributions of Lambda
Daily
Weekly
f(λ)
λ