MKTG776 HW7

Jordan Farrer

2017-03-29

1 Results

Below is the business travel dataset used for fitting latent-class count models (NBD and Poission):

Table 1: Business Travel Data for 464 People

N	observed
0	83
1	44
2	47
3	53
4	55
5	54
6	41
7	30
8	21
9	14
10+	22

Below are the parameter estimates for the NBD and zero-inflated NBD models:

Table 2: Parameter Estimates for NBD and Zero-Inflated NBD Models

model	r	alpha	pi
NBD	1.7907	0.4580	
ZI-NBD	5.2410	1.1594	0.1461

We fit 4 Poission models - one with 1 segments, one with 2 segments, etc. Below are the parameters estimates for lambda (mean of the segments) and theta (% of travelers in each segment).

Table 3: Parameter Estimates for Latent-Class Poisson Models

model	parameter	Seg 1	Seg 2	Seg 3	Seg 4
1-Seg	lambda	3.8092			
2-Seg	lambda	0.5640	5.1650		
2-Seg	theta	0.2920	0.7080		
3-Seg	lambda	7.0418	0.3134	3.8593	
3-Seg	theta	0.2492	0.2288	0.5221	
4-Seg	lambda	8.0242	0.0100	1.0473	4.4800
4-Seg	theta	0.1322	0.1123	0.1740	0.5815

Below is a table that summarizes each of the six models fit in this exercise:

Table 4: Latent-Class Count Model Comparison

model	LL	# params	BIC	$\chi^2\ p-value$
NBD	-1089.2522	2	2190.7841	0.0000
ZI-NBD	-1067.2224	3	2152.8645	0.7868
1-Seg	-1229.2838	1	2464.7075	0.0000
2-Seg	-1072.5437	3	2163.5071	0.0266
3-Seg	-1065.5628	5	2161.8250	0.9886
4-Seg	-1065.3431	7	2173.6653	0.9858

We would select the Zero-Inflated NBD model as our final model. This model has the lowest BIC, only 3 parameters, and a significant p-value for the χ^2 goodness-of-fit test. While the 3-segment Poission has a lower log-likelihood, it requires 5 parameters.

2 Code

Load business travel dataset

```
biz_travel_data <- readxl::read_excel("Biz travel data.xlsx", sheet = 1,
                                      col_names = c('N', 'observed'), skip = 2)
biz_travel_observed <- (readxl::read_excel("Biz travel data.xlsx") %>% names())[2] %>% as.integer()
```

Create functions to find parameters to (zero-inflated) NBD model

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

# Deals with X+ situation
fn_zinbd_px <- function(x, r, alpha, pi) {
  x1 <- as.integer(str_replace(x, "\\+" ,""))
  if (str_detect(x, "\\+")) {
    return(1-sum(purrr::map_dbl(0:(x1-1), fn_zinbd_formula, r, alpha, pi)))
  } else {
    return(fn_zinbd_formula(x1, r, alpha, pi))
  }
}

# Calculates the log-likelihood of the NBD (including zero-inflated)
fn_zinbd_ll <- function(par, data, zero_inflated) {

  pi <- if_else(zero_inflated, par[3], 0)

  data2 <-
    data %>%
    rowwise() %>%
    mutate(p_x = fn_zinbd_px(x = N, r = par[1], alpha = par[2], pi = pi)) %>%
    mutate(ll = observed * log(p_x))

  return(-sum(data2$ll))
```

```
}


fn_zinbd_model <- function(model, data, zero_inflated) {

  init_par <- list(nbd = list(start = c(1,1), lower = c(0,0), upper = c(Inf,Inf)),
                   zinbd = list(start = c(1,1,.5), lower = c(0,0,0), upper = c(Inf,Inf,1)))

  init_par2 <- init_par[[zero_inflated + 1]]

  pars <- nlminb(start = init_par2$start, fn_zinbd_ll, lower = init_par2$lower,
                 upper = init_par2$upper, data = data, zero_inflated = zero_inflated)$par
  return(
    data_frame(model = model, r = pars[1], alpha = pars[2], pi = if_else(zero_inflated, pars[3], NA_real_))
  )

}
```

Find the NBD parameters

```
nbd_params <-
fn_zinbd_model("NBD", biz_travel_data, FALSE) %>%
  bind_rows(
    fn_zinbd_model("ZI-NBD", biz_travel_data, TRUE)
  )
```

Create function to find parameters for latent-class poission models

```
# Poission with arbitary number of lambdas and thetas
fn_lcp_formula <- function(x, lambdas, thetas) {
  p_x <- sum(dpois(x, lambdas) * exp(thetas) / sum(exp(thetas)))
  return(p_x)
}

# Deals with X+ situation
fn_lcp_px <- function(x, lambdas, thetas) {
  x1 <- as.integer(str_replace(x, "\\+" ,""))
  if (str_detect(x, "\\+")) {
    return(1-sum(purrr::map_dbl(0:(x1-1), fn_lcp_formula, lambdas, thetas)))
  } else {
    return(fn_lcp_formula(x1, lambdas, thetas))
  }
}

# Calculates the log-likelihood of the NBD (including zero-inflated)
fn_lcp_ll <- function(start, data, seg) {

  lambdas <- start[1:seg]
  if (seg > 1) {
    thetas <- c(start[(seg + 1):length(start)], 0)
  } else {
    thetas <- 0
  }

  data2 <-
    data %>%
    rowwise() %>%
    mutate(p_x = fn_lcp_px(x = N, lambdas, thetas)) %>%
    mutate(ll = observed * log(p_x))
```

```
  return(-sum(data2$ll))
}


fn_lcp_model <- function(model, data, seg) {

  start <- c(runif(seg, 1, 1), runif(seg-1, -1, 1))
  lower <- c(rep(0.01, seg), rep(-Inf, seg - 1))
  upper <- rep(Inf, seg)

  pars <- nlminb(start = start, fn_lcp_ll, lower = lower, upper = upper,
                 data = data, seg = seg, control = list(x.tol = 1e-10))$par


  if (seg > 1) {
    parameters <- c(rep("lambda", seg), rep("theta", seg))
    thetas <- exp(c(pars[(seg + 1):length(start)], 0))
    estimates <- c(pars[1:seg], thetas / sum(thetas))
  } else {
    parameters <- "lambda"
    estimates <- pars
  }

  return(
    data_frame(model = rep(model, length(estimates)), parameter = parameters, estimate = estimates)
  )
}
```

Find the latent-class poission parameters

```
lcp_params <-
  fn_lcp_model("1-Seg", biz_travel_data, 1) %>%
    bind_rows(
      fn_lcp_model("2-Seg", biz_travel_data, 2),
      fn_lcp_model("3-Seg", biz_travel_data, 3),
      fn_lcp_model("4-Seg", biz_travel_data, 4)
    )
```

Calculate the NBD results

```
fn_zinbd_ll_results <- function(data, r, alpha, pi, total_obs) {

  params = sum(!is.na(c(r, alpha, pi)))
  pi <- ifelse(!is.na(pi), pi, 0.0)

  data2 <-
    data %>%
    rowwise() %>%
    mutate(p_x = fn_zinbd_px(x = N, r = r, alpha = alpha, pi = pi)) %>%
    ungroup() %>%
    mutate(
      ll = observed * log(p_x)
      , expected =  total_obs * p_x
      , chisq = (observed - expected)^2 / expected
    ) %>%
    summarise(
          ll = sum(ll)
        , chisq = sum(chisq)
        , percent_expected = sum(expected > 5) / n()
        , cells = n()
```

```r
    ) %>%
    mutate(
      BIC = -2 * ll + params * log(total_obs)
      , p.value = pchisq(chisq, df = cells - params -1, lower.tail = FALSE)
      , params = params
    )

  if (data2$percent_expected < 0.8) {
    stop("Less than 80% of the cells have more than 5 counts")
  }

  return(
    data2 %>%
      select(
        LL = ll
        , `# params` = params
        , BIC = BIC
        , `$\\chi^2\\,p-value$` = p.value
      )
  )

}

nbd_results <-
  nbd_params %>%
    crossing(biz_travel_data) %>%
    group_by(model, r, alpha, pi) %>%
    nest(.key = travel_data) %>%
    mutate(pred = pmap(list(travel_data, r, alpha, pi), fn_zinbd_ll_results, biz_travel_observed)) %>%
    select(model, pred) %>%
    unnest()
```

Calculate the latent-class poission results

```r
# Poission with arbitary number of lambdas and thetas
fn_lcp_formula2 <- function(x, lambdas, thetas) {
  if (length(lambdas) == 1) {
    p_x <- sum(dpois(x, lambdas))
  } else {
    p_x <- sum(dpois(x, lambdas) * thetas)
  }
  return(p_x)
}

# Deals with X+ situation
fn_lcp_px2 <- function(x, lambdas, thetas) {
  x1 <- as.integer(str_replace(x, "\\+" ,""))
  if (str_detect(x, "\\+")) {
    return(1-sum(purrr::map_dbl(0:(x1-1), fn_lcp_formula2, lambdas, thetas)))
  } else {
    return(fn_lcp_formula2(x1, lambdas, thetas))
  }
}

fn_lcp_ll_results <- function(data, lambdas, thetas, total_obs) {

  if (length(lambdas) == 1) {
    params <- 1
  } else {
```

```r
    params <- sum(!is.na(c(lambdas, thetas))) - 1
  }

  data2 <-
    data %>%
    rowwise() %>%
    mutate(p_x = fn_lcp_px2(x = N, lambdas, thetas)) %>%
    ungroup() %>%
    mutate(
      ll = observed * log(p_x)
      , expected =  total_obs * p_x
      , chisq = (observed - expected)^2 / expected
    ) %>%
    summarise(
          ll = sum(ll)
        , chisq = sum(chisq)
        , percent_expected = sum(expected > 5) / n()
        , cells = n()
      ) %>%
    mutate(
      BIC = -2 * ll + params * log(total_obs)
      , p.value = pchisq(chisq, df = cells - params -1, lower.tail = FALSE)
      , params = params
    )

  if (data2$percent_expected < 0.8) {
    stop("Less than 80% of the cells have more than 5 counts")
  }

  return(
    data2 %>%
      select(
        LL = ll
        , `# params` = params
        , BIC = BIC
        , `$\\chi^2\\,p-value$` = p.value
      )
  )

}

lcp_results <-
  lcp_params %>%
  mutate(num = row_number()) %>%
  spread(parameter, estimate) %>%
  select(-num) %>%
  group_by(model) %>%
  summarise(
    lambdas = list(lambda)
    , thetas = list(theta)
  ) %>%
  mutate(
    lambdas = map(lambdas, na.omit)
    , thetas = map(thetas, na.omit)
  ) %>%
  left_join(
    lcp_params %>%
      distinct(model) %>%
```

```
      crossing(biz_travel_data) %>%
      nest(N, observed)
    , by = 'model'
  ) %>%
  mutate(pred = pmap(list(data, lambdas, thetas), fn_lcp_ll_results, biz_travel_observed)) %>%
  select(model, pred) %>%
  unnest()
```

Output summary of each model

```
nbd_results %>%
  bind_rows(
    lcp_results
  ) %>%
  pander(caption = "Latent-Class Count Model Comparison")
```