Stat 135 Q58

Jiying Zou

March 17, 2017

58

Parameter Estimation

Berkson is sampling waiting times between events, so we expect his data to be draws from an exponential distribution with unknown parameter λ.

We can estimate this parameter using maximum likelihood:

$$lik(\lambda) = \prod_{i=1}^{n} \lambda e^{-\lambda x_i} = \lambda^n e^{-\lambda \sum_{i=1}^{n} x_i}$$

$$l(\lambda) = nlog(\lambda) - \lambda \sum_{i=1}^{n} x_i$$

$$l'(\lambda) = \frac{n}{\lambda} - \sum_{i=1}^{n} x_i = 0$$

$$n = \lambda \sum_{i=1}^{n} x_i$$

$$\lambda = \frac{1}{\frac{1}{n} \sum_{i=1}^{n} x_i}$$

$$\hat{\lambda}_{MLE} = \frac{1}{\bar{x}}$$

We can estimate $\bar{x}$ using our dataset by taking the midpoint of each time interval, multiplying them by their frequencies, and dividing by the total number of observations.

First, let’s create the dataframe:

(Technical notes: For the last category with unlimited upper bound, I am using the pseudo-value of 99,999, after which the CDF of the exponential is about the same. However, for convenience we will still use 23,439 as the “midpoint” of the last interval so that our calculations make sense.)

```
#Dataframe Creation
lower <- c(0,60,120,181,243,306,369,432,497,562,628,689,1130,1714,2125,2567,3044,3562
,4130,4758,5460,6255,7174,8260,9590,11304,13719,14347,15049,15845,16763,17849,19179,2
0893,23309,27439)

upper <- c(60,120,181,243,306,369,432,497,562,628,689,1130,1714,2125,2567,3044,3562,4
130,4758,5460,6255,7174,8260,9590,11304,13719,14347,15049,15845,16763,17849,19179,208
93,23309,27439,99999)

midpt <- lower + (upper - lower)/2
midpt[midpt > 27439] = 27439 #correct the last midpoint value

freq <- c(115,104,99,106,113,104,101,106,104,96,512,524,468,531,461,526,506,509,520,5
40,542,499,494,500,550,465,104,97,101,104,92,102,103,110,112,100)

df <- data.frame(lower, upper, midpt, freq)
```

Then, let’s find $\bar{x}$, the average time between events, and $\hat{\lambda}_{MLE}$:

```
x_bar <- sum(df$midpt*df$freq)/sum(df$freq)
x_bar
```

```
## [1] 5884.375
```

```
lambda_mle <- 1/x_bar
lambda_mle
```

```
## [1] 0.0001699416
```

Calculate Expected Frequencies

Let X = time between events, then X ~ Exp($\hat{\lambda}_{MLE}$) with CDF $F(x_i) = 1 - e^{-\lambda x_i}$

The expected frequencies can be calculated by:

$$Expected\,frequency = \#observations\ *\ P(wait\ time\,falls\ in\ time\ interval)$$

$$= \#observations\ *\ [P(X \leq x_{upper}) - P(X \leq x_{lower})]$$

$$= \#observations\ *\ [(1 - e^{-\lambda x_{upper}}) - (1 - e^{-\lambda x_{lower}})]$$

```
#Function to return vector of expected frequencies
#upper and lower are input vectors
exp_int_cdf <- function(lambda, upper, lower, ttl){
  upper_cdf <- (1-exp(-lambda*upper))
  lower_cdf <- (1-exp(-lambda*lower))
  return(ttl * (upper_cdf - lower_cdf))
}

#Append expected frequencies
df <- df %>%
  mutate(expected = exp_int_cdf(lambda_mle, upper, lower, sum(df$freq)))
```

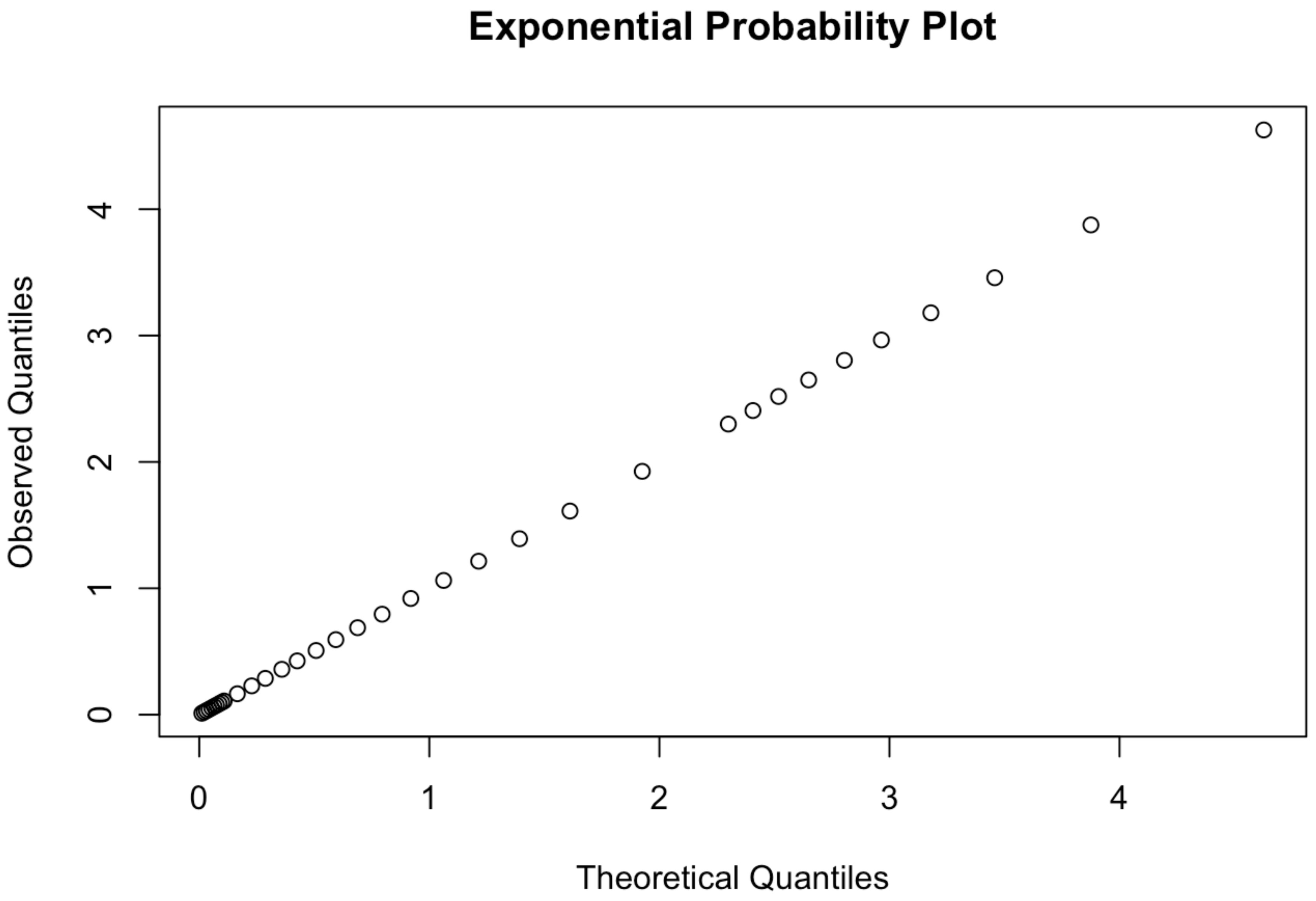
Probability Plot

Since it is difficult to plot the observed values against expected values (due to time intervals being given, with frequencies…), we create a QQ plot by plotting the quantiles of observed values against their quantiles of expected (theoretical) values, and then evaluate linearity.

```
#Quantiles of observed interval midpoints
obs_per <- c() #vector of percentiles
obs_prob <- df$freq/sum(df$freq)
for(i in 1:length(obs_prob))
  {obs_per[i] <- sum(obs_prob[1:i])}
obs_quant <- qexp(obs_per) #observed quantiles

#Quantiles of expected interval midpoints
exp_per <- c() #vector of percentiles
exp_prob <- df$freq/sum(df$freq)
for(i in 1:length(exp_prob))
  {exp_per[i] <- sum(exp_prob[1:i])}
exp_quant <- qexp(exp_per) #expected quantiles

qqplot(exp_quant, obs_quant, main = "Exponential Probability Plot", xlab = "Theoretic
al Quantiles", ylab = "Observed Quantiles")
```



The graph shows clear linearity, meaning that the observed quantiles of our data match well with expected quantiles from the theoretical distribution. Thus we can say that it is reasonable to assume an exponential distribution with rate $\hat{\lambda}_{MLE}$ underlies our data generation!