Case Study: duration of hospitalization

Data preparation

```
Variable   Description
---------- -----------------------------------
DUR        Duration of hospitalization (days)
AGE        Age (years)
SEX        1 = male 2 = female
TEMP       Body temperature (degrees Fahrenheit)
WBC        White blood cells per 100 ml blood
ANTIB      Antibiotic use: 1 = yes 2 = no
CULT       Blood culture taken 1 = yes 2 = no
SERV       Service: 1 = medical 2 = surgical
---------- -----------------------------------
```

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------- tidyverse 1.2.1 --

## v ggplot2 3.2.0     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## Warning: package 'dplyr' was built under R version 3.6.1

## -- Conflicts ------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

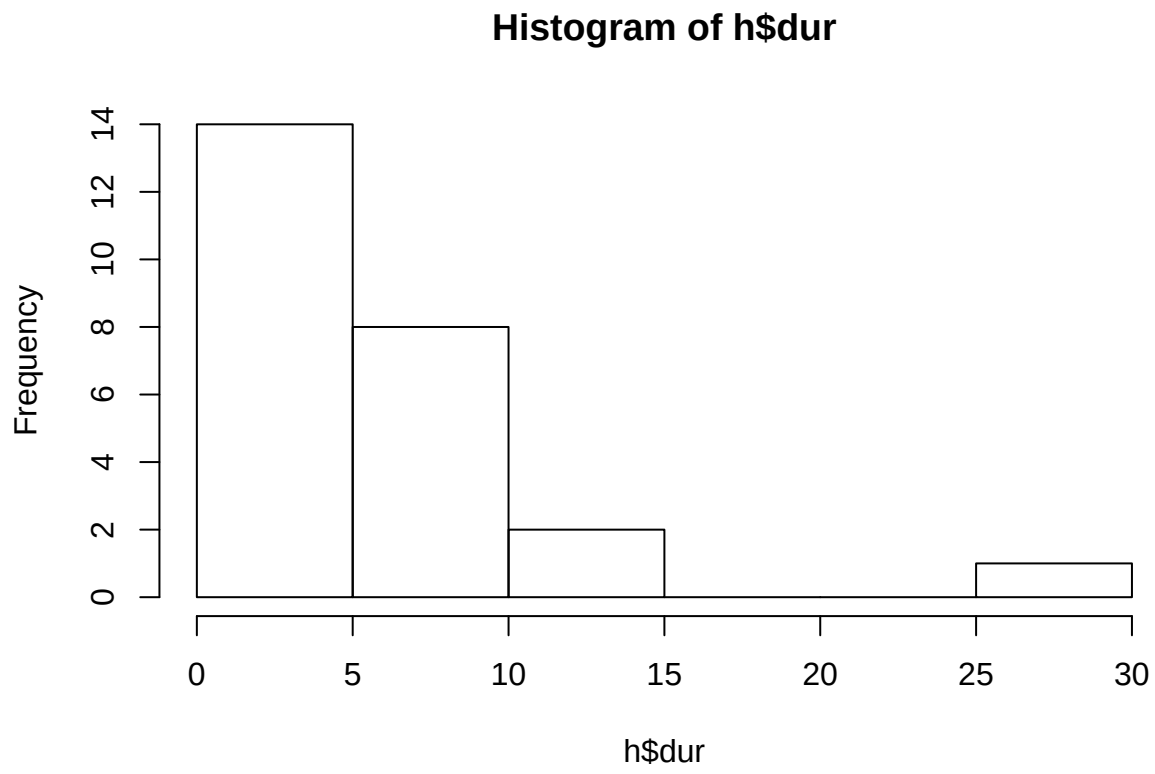
```
raw <-
  read_tsv("./hospitalization.csv",
           col_types = cols(id = 'c',
                            dur = 'd',
                            age = 'd',
                            sex = 'c',
                            temp = 'd',
                            wbc = 'd',
                            antib = 'c',
                            cult = 'c',
                            serv = 'c'))

h <- mutate(raw,
  sex = factor(sex, levels = c('1', '2'), labels = c('m', 'f')),
  antib = factor(antib, levels = c('2', '1'), labels = c('no', 'yes')),
  cult = factor(cult, levels = c('2', '1'), labels = c('no', 'yes')),
  serv = factor(serv, levels = c('1', '2'), labels = c('medical', 'surgical')),
  temp = (temp - 32.0) * 5/9  ## convert to Celsius
  ) %>%
  select(-id)
```

Exploratory analysis

```
summary(h)
```

```
##       dur              age        sex         temp            wbc
##  Min.   : 0.0   Min.   : 4.00   m:11   Min.   :36.00   Min.   : 3.00
##  1st Qu.: 2.0   1st Qu.:25.00   f:14   1st Qu.:36.67   1st Qu.: 5.00
##  Median : 5.0   Median :41.00          Median :36.78   Median : 7.00
##  Mean   : 5.6   Mean   :41.24          Mean   :36.84   Mean   : 7.84
##  3rd Qu.: 8.0   3rd Qu.:56.00          3rd Qu.:37.00   3rd Qu.:11.00
##  Max.   :27.0   Max.   :82.00          Max.   :37.50   Max.   :14.00
##  antib     cult          serv
##  no :18   no :19   medical : 9
##  yes: 7   yes: 6   surgical:16
##
##
##
##
```

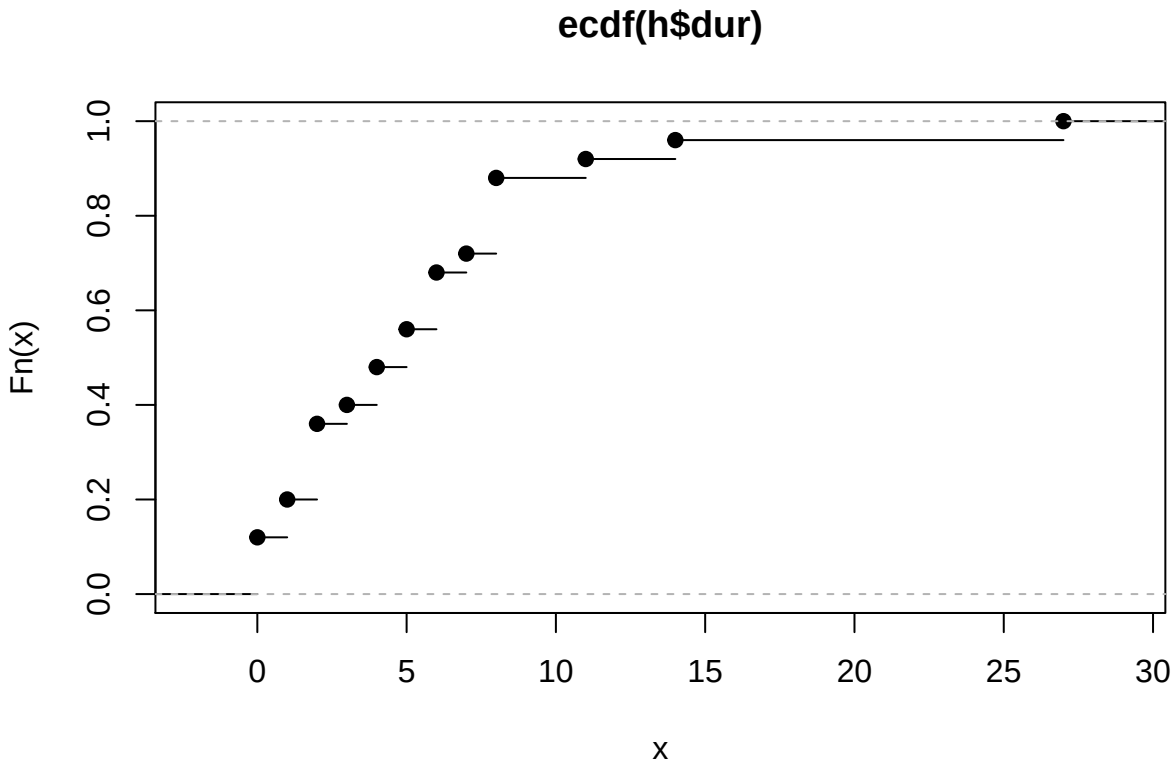
```
hist(h$dur)
```



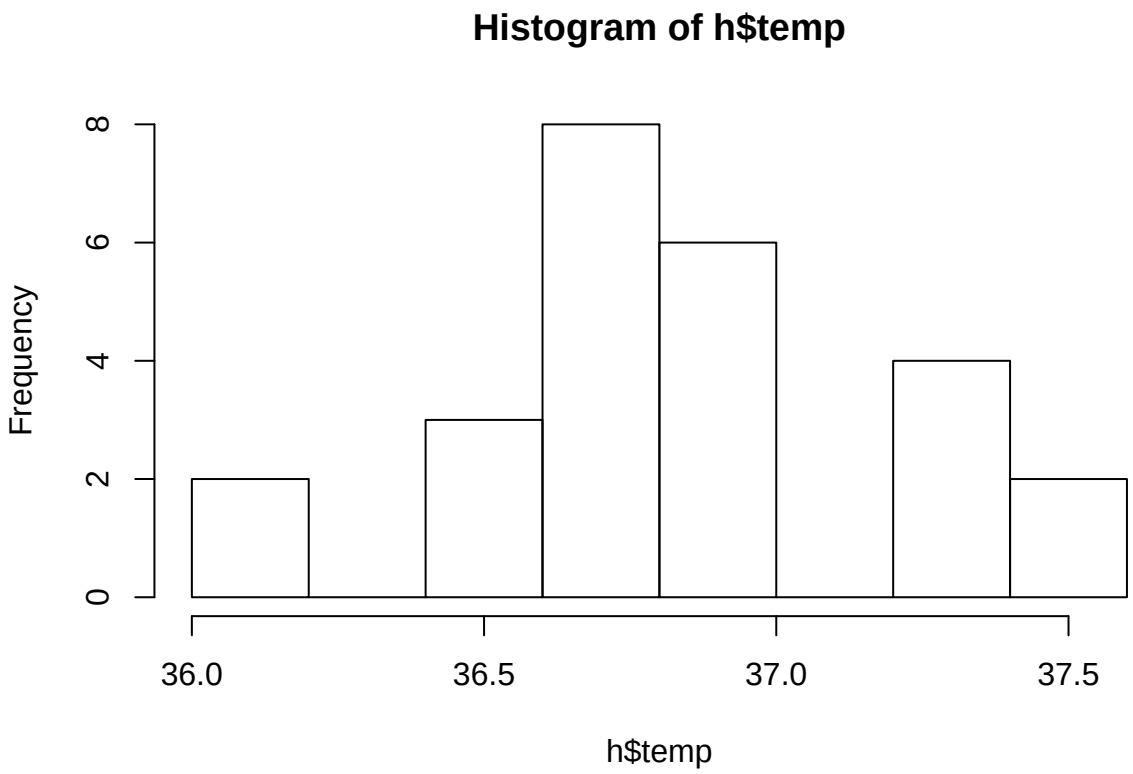

```
Fn <- ecdf(h$dur)
summary(Fn)
```

```
## Empirical CDF:     12 unique values with summary
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   2.750   5.500   7.333   8.750  27.000
```

```
plot(Fn)
```




```
hist(h$temp)
```



Q1. How many patients go through overnight hospitalization?

```
table(h$dur == 0)
```

```
##
## FALSE  TRUE
##    22     3
```

```
prop.table(table(h$dur == 0))
```

```
##
## FALSE  TRUE
##  0.88  0.12
```

Overall, 12% of the patients go through overnight hospitalization.

We can get a confidence interval using the Binomial test:

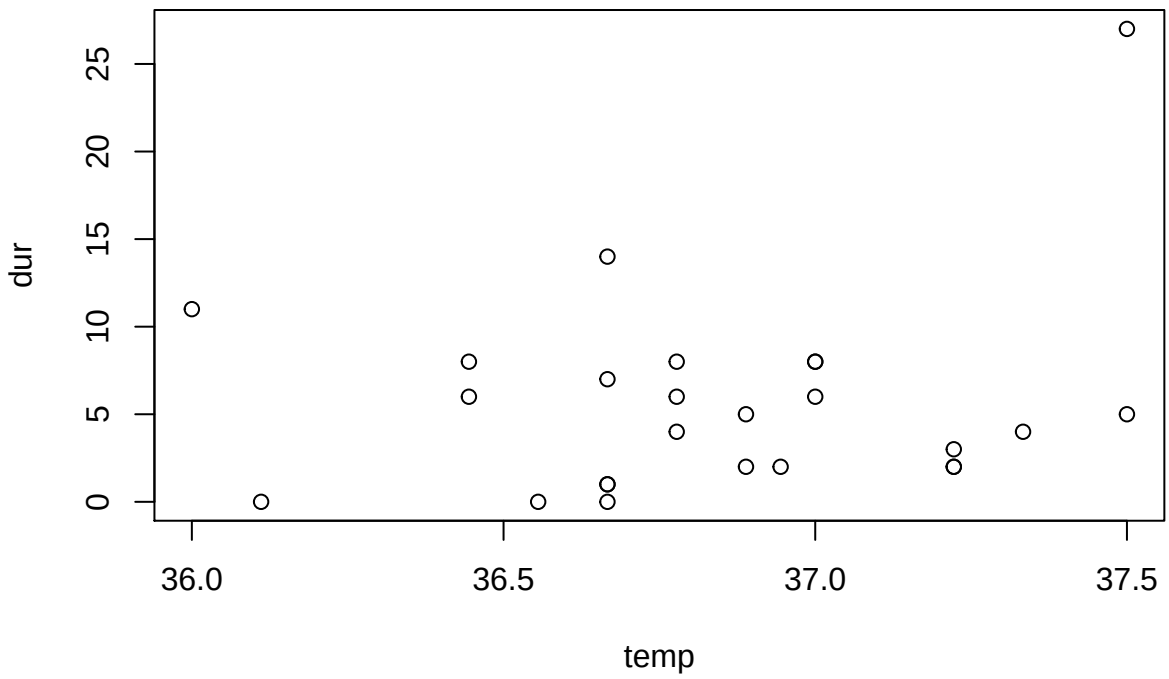
```
with(h, binom.test(table(dur == 0)))
```

```
##
```

```
##  Exact binomial test
##
## data:  table(dur == 0)
## number of successes = 22, number of trials = 25, p-value =
## 0.0001565
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.6878097 0.9745346
## sample estimates:
## probability of success
##                   0.88
```

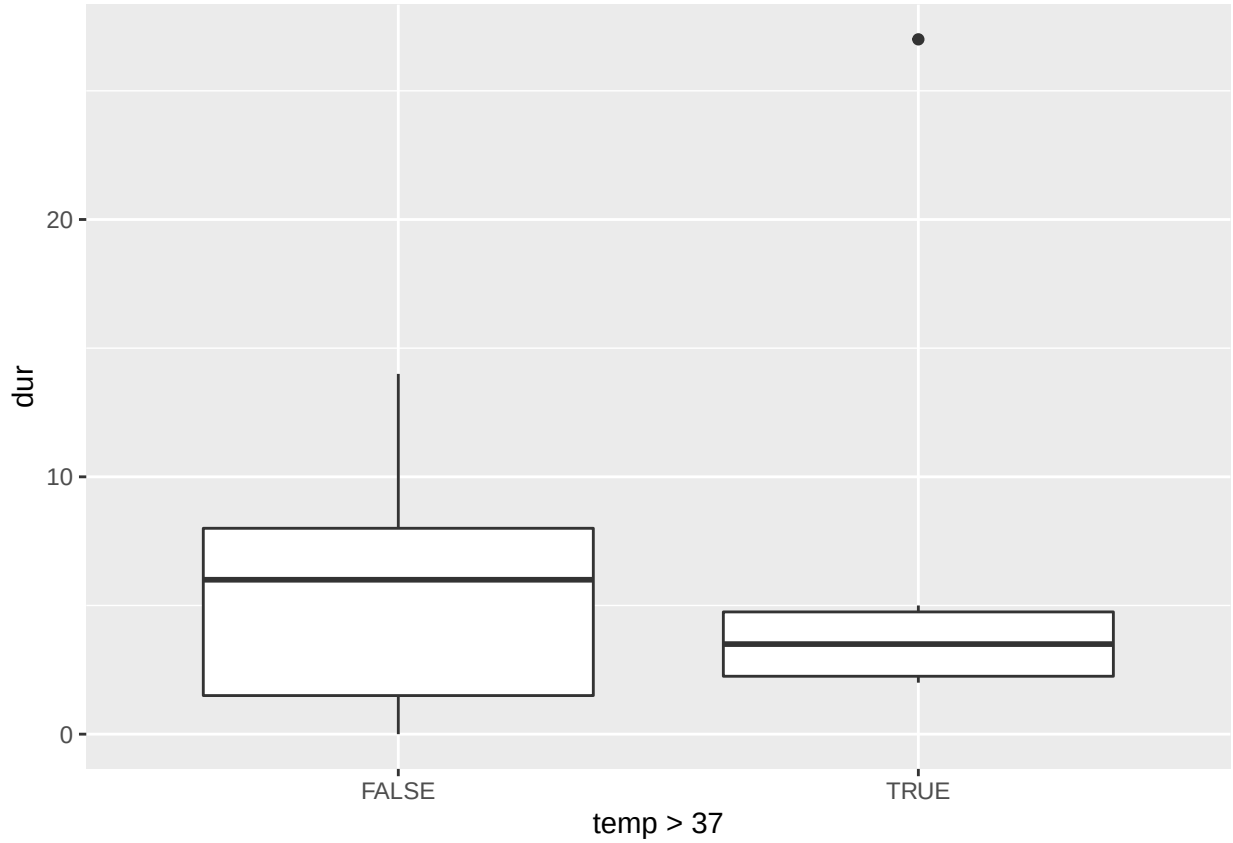
Q2. Is the body temperature at admission predictive of the duration of the hospitalization?

```
plot(dur ~ temp, data = h)
```



Meh!

```
ggplot(h, aes(x = temp > 37, y = dur)) +
  geom_boxplot()
```




```
with(h, table(temp > 37, dur > 0,deparse.level = 2))
```

```
##          dur > 0
## temp > 37 FALSE TRUE
##     FALSE     3   16
##     TRUE      0    6
```

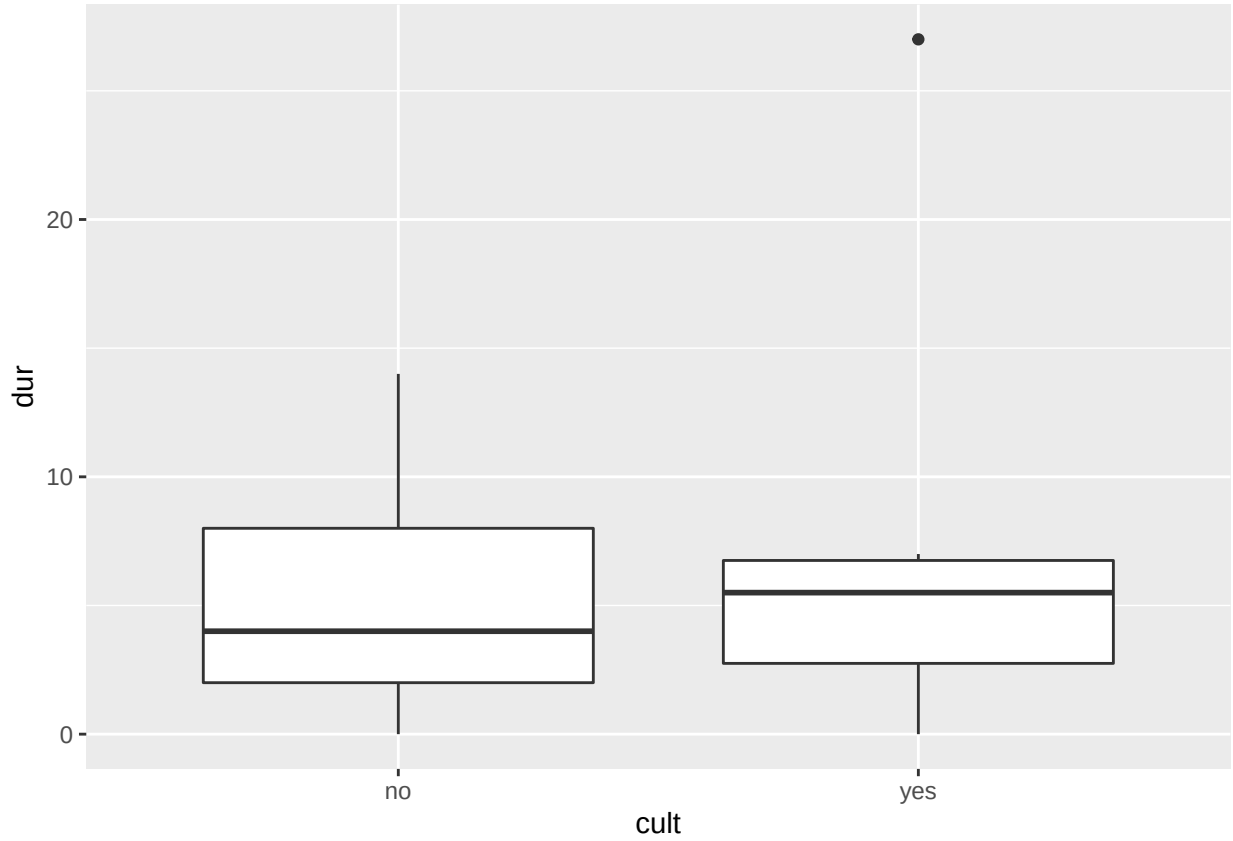
```
#deparse.level: show the topic of true and false
```

```
with(h, prop.table(table(temp > 37, dur > 0,deparse.level = 2), 1))
```

```
##          dur > 0
## temp > 37     FALSE      TRUE
##     FALSE 0.1578947 0.8421053
##     TRUE  0.0000000 1.0000000
```

Q3. What about blood works?

```
ggplot(h, aes(x = cult, y = dur)) +
  geom_boxplot()
```




```
with(h, table(cult, dur > 0))
```

```
##
## cult  FALSE TRUE
##   no      2   17
##   yes     1    5
```

```
with(h, prop.table(table(cult, dur > 0), 1))
```

```
##
## cult      FALSE      TRUE
##   no  0.1052632 0.8947368
##   yes 0.1666667 0.8333333
```

```
#,1)) means the margin, so the raw equal to one
```

Q4. Statistical Modeling

Data preparation

```r
h$y <- h$dur > 0
mean_temperature <- mean(h$temp)
h$x <- h$temp - mean_temperature
```

The Likelihood function

```r
logLik <- function(theta) {
  alpha <- theta[1]
  beta <- theta[2]
  linear_predictor <- alpha + beta * h$x
  probabilities <- plogis(linear_predictor)
  log_terms <- dbinom(h$y,
                      size = 1,
                      prob = probabilities,
                      log = TRUE)
  sum(log_terms)
  #in the formula it's the product but in log scale we compute the sum

}
help(plogis)
```

```
## starting httpd help server ... done
```

```r
debugonce(logLik)#able to do debugging in R
logLik(c(3, 3))
```

```
## debugging in: logLik(c(3, 3))
## debug at <text>#1: {
##     alpha <- theta[1]
##     beta <- theta[2]
##     linear_predictor <- alpha + beta * h$x
##     probabilities <- plogis(linear_predictor)
##     log_terms <- dbinom(h$y, size = 1, prob = probabilities,
##         log = TRUE)
##     sum(log_terms)
## }
## debug at <text>#2: alpha <- theta[1]
## debug at <text>#3: beta <- theta[2]
## debug at <text>#4: linear_predictor <- alpha + beta * h$x
## debug at <text>#5: probabilities <- plogis(linear_predictor)
## debug at <text>#6: log_terms <- dbinom(h$y, size = 1, prob = probabilities, log = TRUE)
## debug at <text>#10: sum(log_terms)
## exiting from: logLik(c(3, 3))

## [1] -7.568636
```

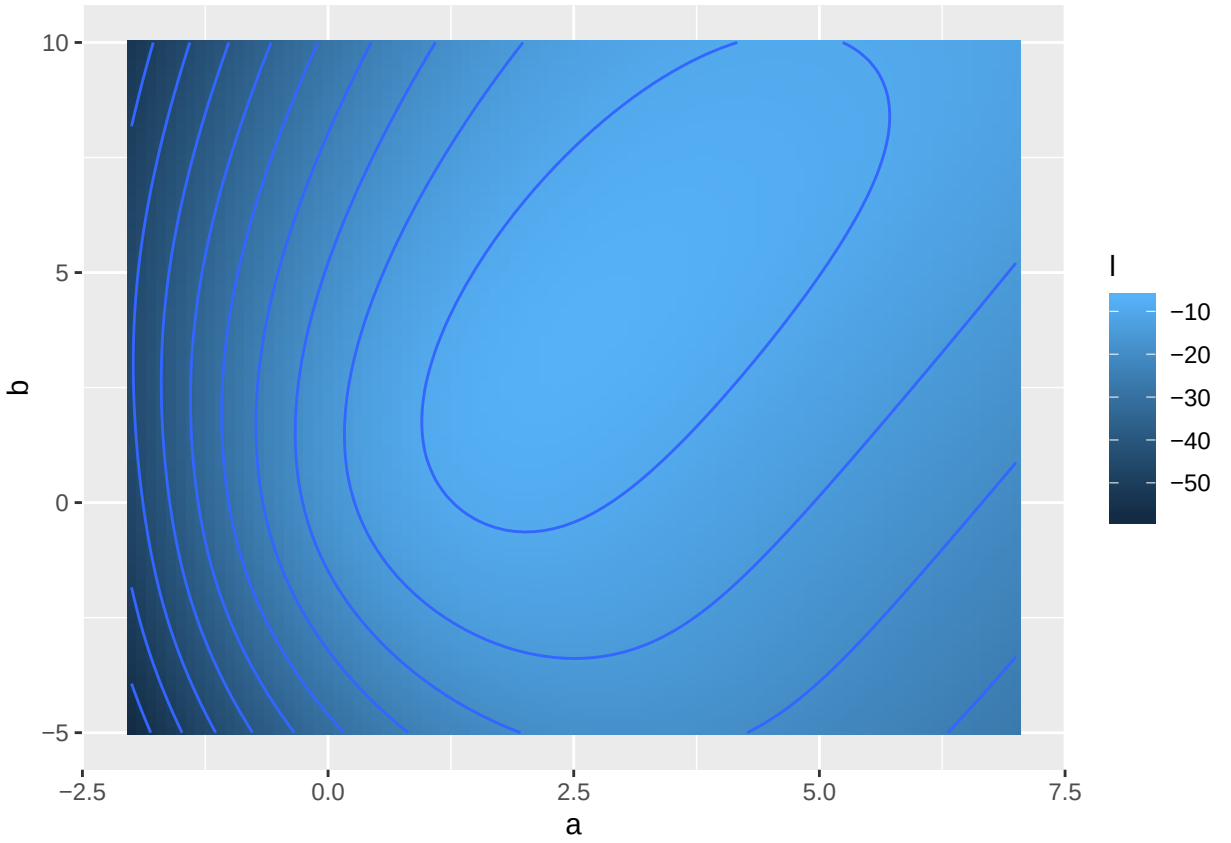
visualize the logLikelihood function

prepare the data:

```
d<-
  expand.grid(a=seq(-2,7,by=0.1),
              b=seq(-5,10,by=0.1)) %>%
  mutate(ab=map2(a,b,c),
         l=map_dbl(ab,logLik))
```

visualize using ggplot:

```
ggplot(d,aes(x=a,y=b))+
  geom_tile(aes(fill=l))+
  stat_contour(aes(z=l))
```



we want to find a Theda (alpha and beta) to get the maximum likelihood

Numerical maximization

```
library(maxLik)
```

```
## Warning: package 'maxLik' was built under R version 3.6.1
```

```
## Loading required package: miscTools
```

```
## Warning: package 'miscTools' was built under R version 3.6.1
```

```
##
## Please cite the 'maxLik' package as:
## Henningsen, Arne and Toomet, Ott (2011). maxLik: A package for maximum likelihood estimation in R. C
##
## If you have questions, suggestions, or comments regarding the 'maxLik' package, please use a forum o
## https://r-forge.r-project.org/projects/maxlik/
```

```
fit <- maxLik(logLik, start = c(alpha = 0, beta = 0))
summary(fit)
```

```
## --------------------------------------------
## Maximum Likelihood estimation
## Newton-Raphson maximisation, 6 iterations
## Return code 1: gradient close to zero
## Log-Likelihood: -7.176829
## 2  free parameters
## Estimates:
##       Estimate Std. error t value Pr(> t)
## alpha   2.6038     0.9102   2.861 0.00423 **
## beta    3.6667     2.0751   1.767 0.07723 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## --------------------------------------------
```

conduct a confidence interval: estimate+- 2*std

```
confint(fit)
```

```
##            2.5 %   97.5 %
## alpha  0.8198029 4.387821
## beta  -0.4004012 7.733872
```

Logistic regression

$H_0 : B = 0, H_1 : B \neq 0$ for x we do not reject H0, the body temperature has not significantly effect on the duration

```
fit2 <- glm(y ~ x, data = h, family = 'binomial')
#family default is guassian, even y is not pro can still work 1,2,3,4...n for example
summary(fit2)
```

```
##
## Call:
## glm(formula = y ~ x, family = "binomial", data = h)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0524   0.1893   0.3125   0.5094   1.3816
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)   2.6038     0.9158   2.843  0.00447 **
## x             3.6667     2.0942   1.751  0.07996 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 18.346  on 24  degrees of freedom
## Residual deviance: 14.354  on 23  degrees of freedom
## AIC: 18.354
##
## Number of Fisher Scoring iterations: 6
```

```
confint(fit2)
```

```
## Waiting for profiling to be done...
```

```
##                  2.5 %   97.5 %
## (Intercept) 1.18689527 5.010155
## x           0.06464185 8.776097
```

Model-based predictions

In the probability scale:

```
predict(fit2,
        newdata = data.frame(x = 38 - mean_temperature),
        type = "response") #for probability
```

```
##         1
## 0.9989578
```

This is the so-called 'linear predictor':

```
predict(fit2,
        newdata = data.frame(x = 38 - mean_temperature),
        type = "link") #for linear predictor
```

```
##        1
## 6.865371
```

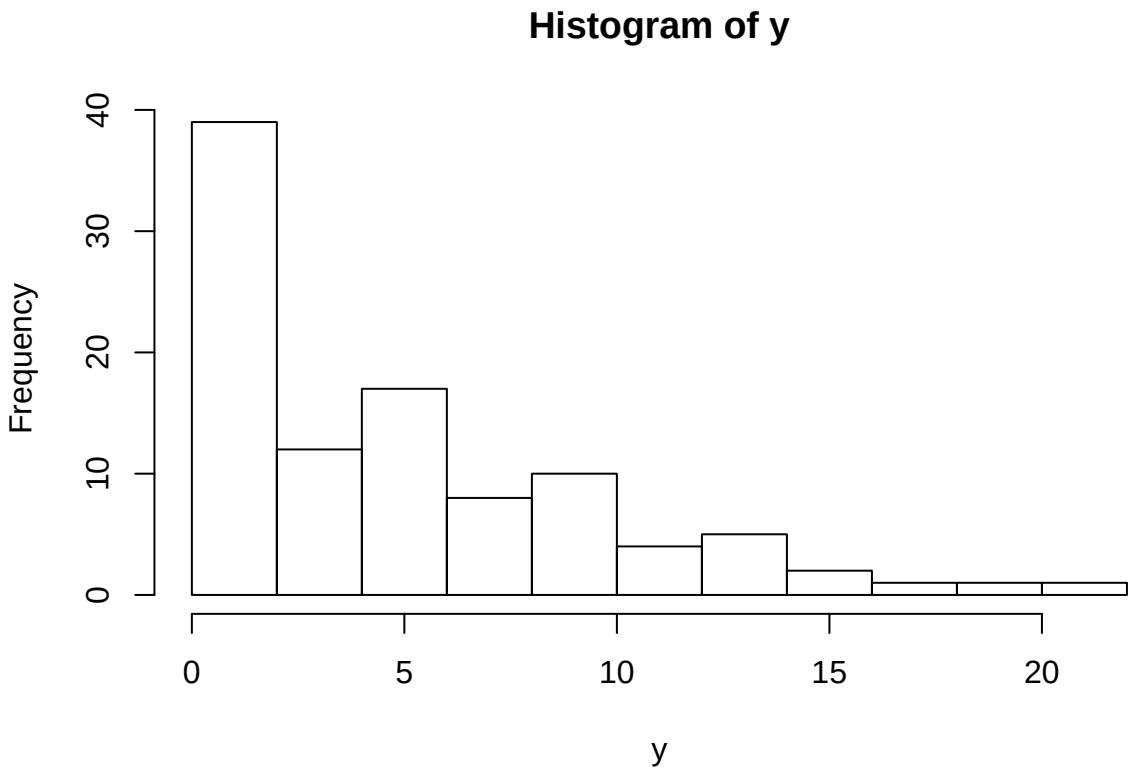
```
# can be useful when you just want to compare with all the observation and you just want to sort it...
```

Exercises with simulated data

Data generation

```r
set.seed(1234)
rate <- 0.2
y <- rexp(100, rate = rate)
```

```r
hist(y)
```



Mean and median: close to one and another

```r
mean(y) #sample
```

```
## [1] 4.882299
```

```r
1/rate #theoretical
```

```
## [1] 5
```

```r
median(y) #sample
```

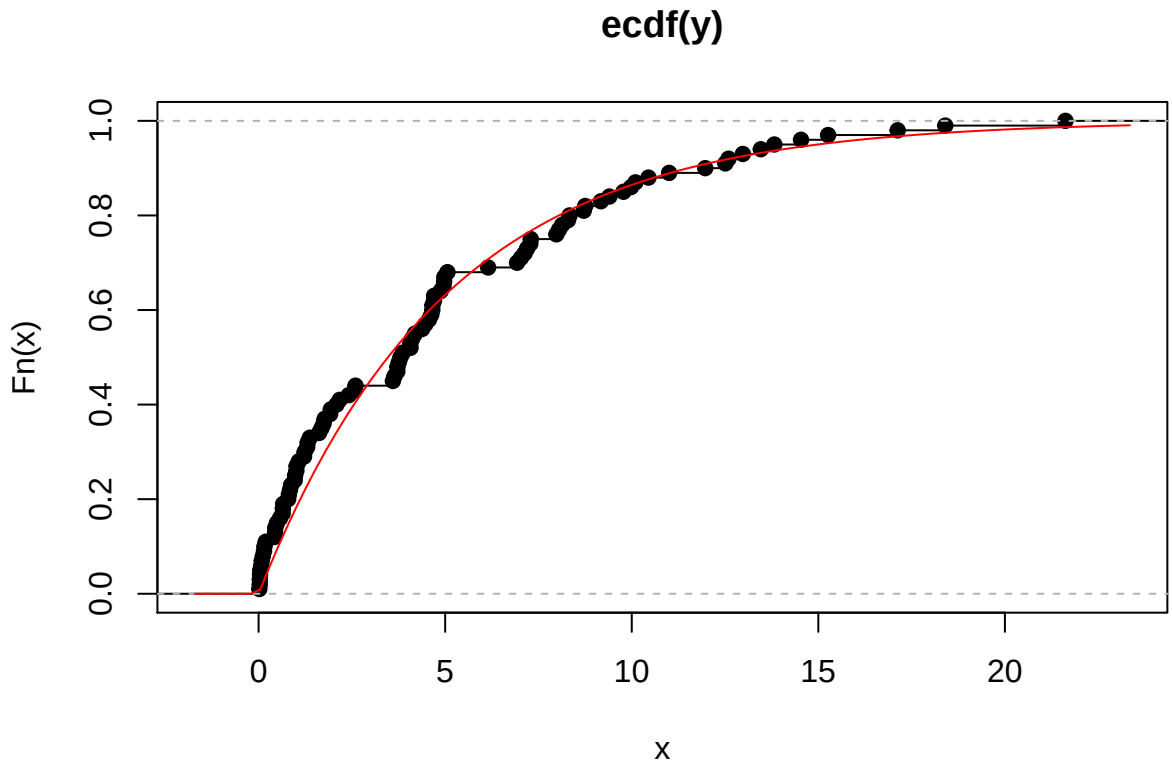
```
## [1] 3.835192
```

```r
log(2)/rate
```

```
## [1] 3.465736
```

ECDF

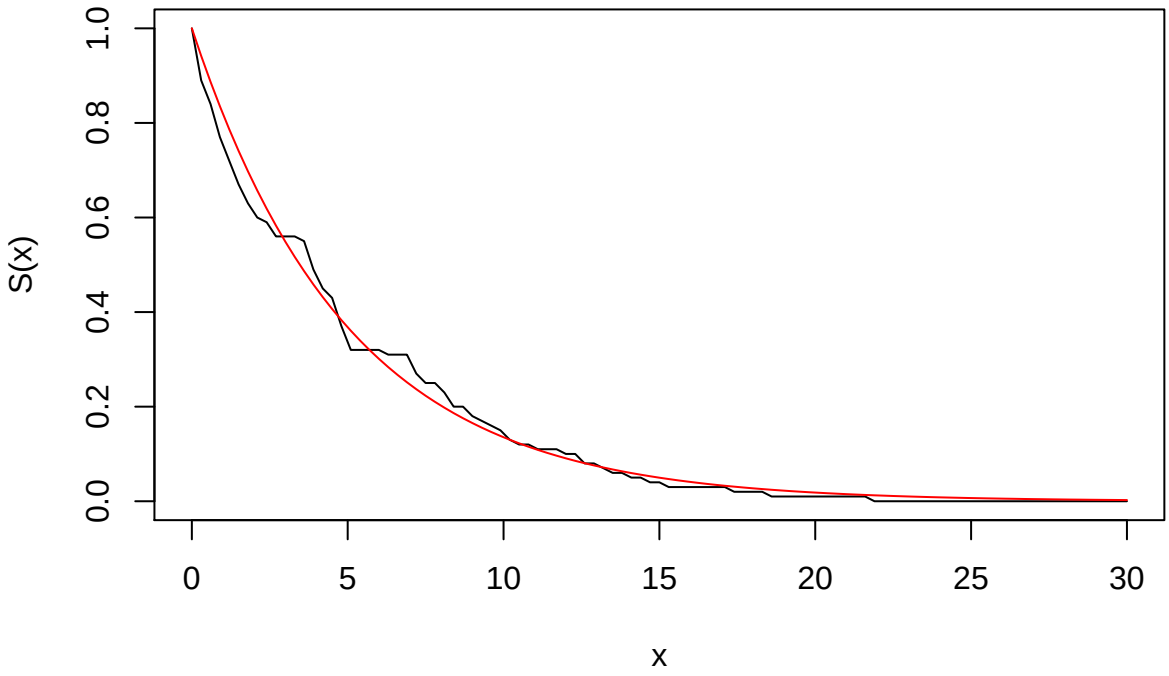
```r
F <- ecdf(y) #sample approach to theoretical
plot(F)
curve(pexp(x, rate = rate), col = "red", add = TRUE)
```




```r
#theoretical one
```

Survival function

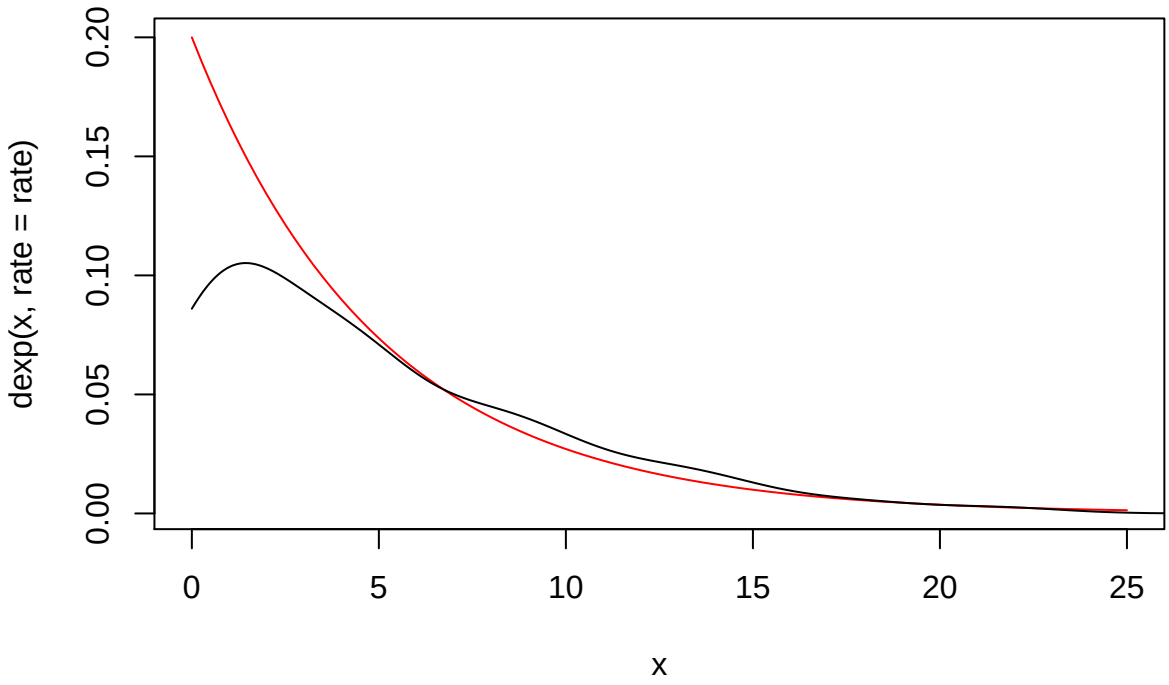
```r
S <- function(t) 1 - F(t)
curve(S(x), from = 0, to = 30)
curve(pexp(x, rate = rate, lower.tail = FALSE), col = "red", add = TRUE)
```



Density and hazard functions

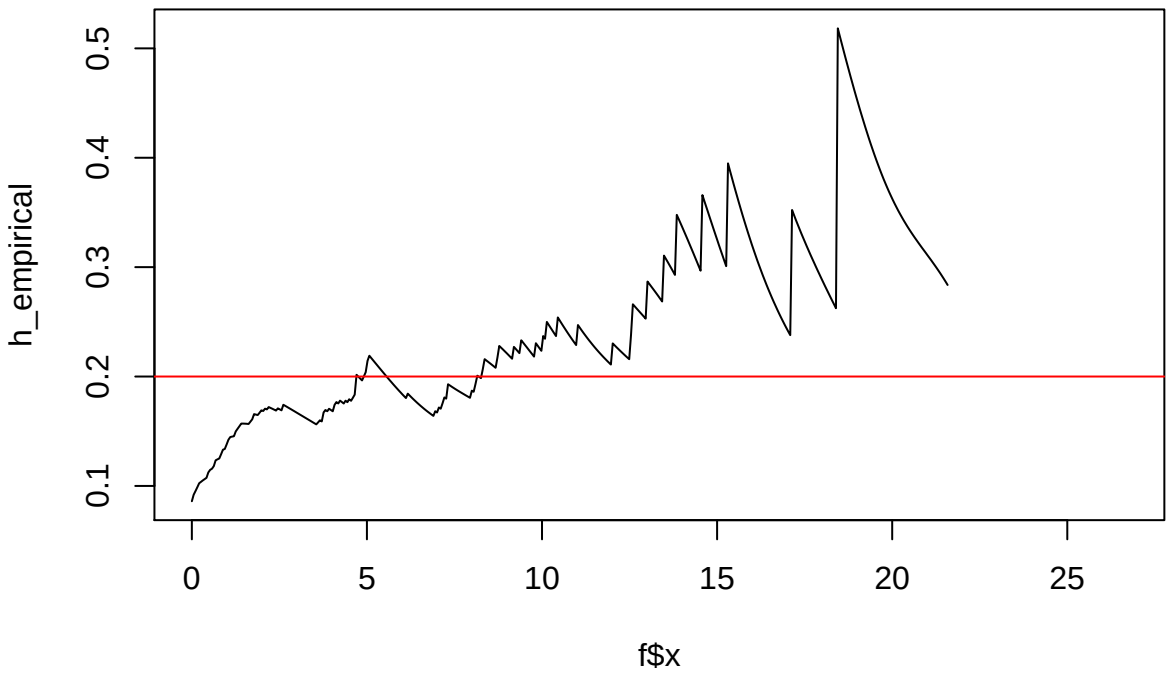
doesn't fit so well

```
f <- density(y, from = 0)
curve(dexp(x, rate = rate), col = "red", from = 0, to = 25)
lines(f)
```



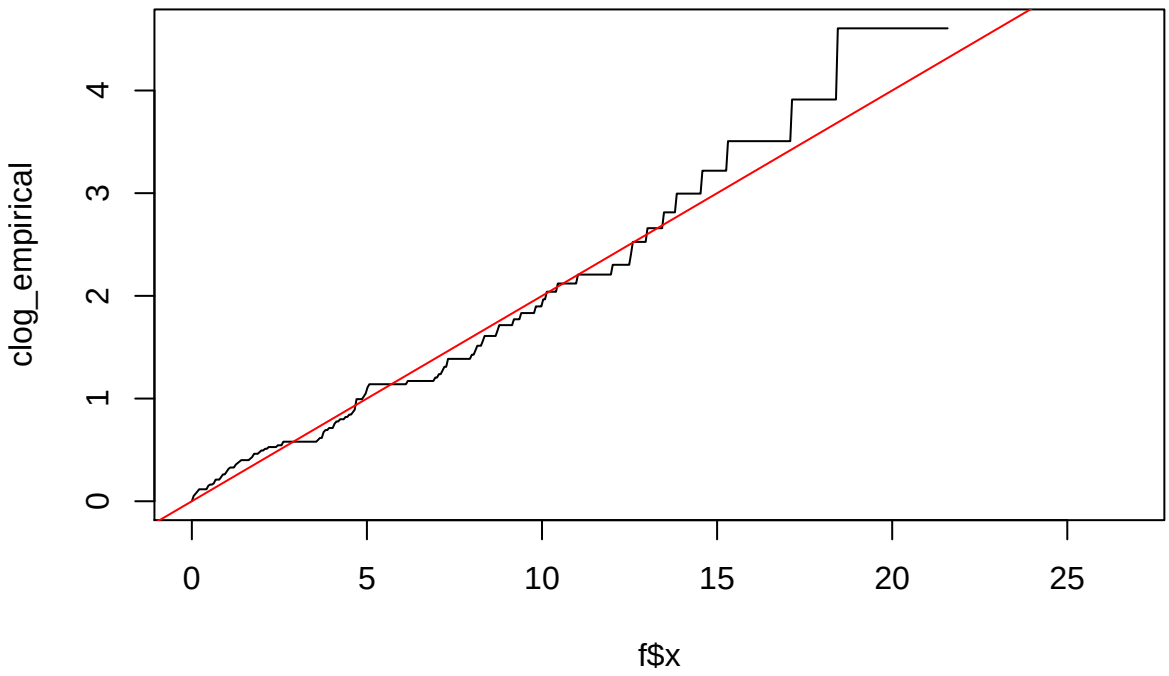
doesn't fit so well

```
h_empirical <- f$y / S(f$x)
plot(f$x, h_empirical, type = "l")
abline(h = rate, col = "red")
```



now let's do it parametically ## Minus-log survival function

```
clog_empirical <- -log(S(f$x))
plot(f$x, clog_empirical, type = "l")
abline(a = 0, b = rate, col = "red")
```

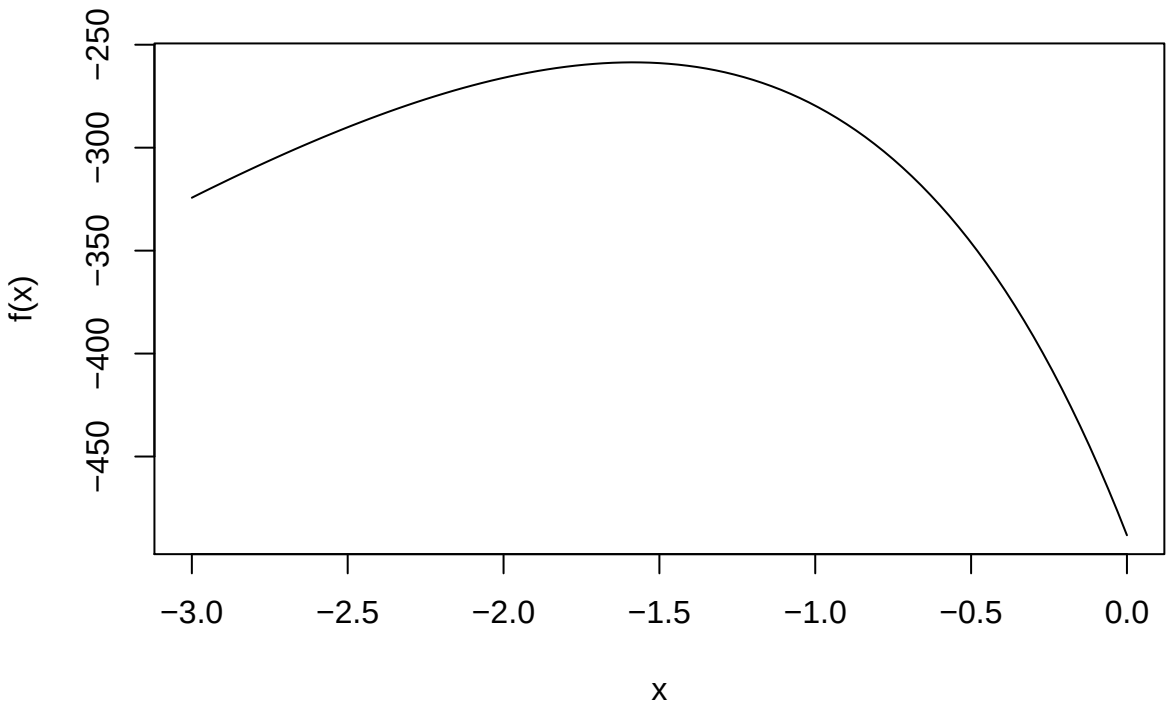


Maximum Likelihood Estimation

The log-likelihood function

```
logLik <- function(logRate) {
  sum(dexp(y, rate = exp(logRate), log = TRUE))
}
```

```
f <- Vectorize(logLik)
curve(f(x), from = -3, to = 0)
```



Numerical Maximization

```
library(maxLik)
fit <- maxLik(f, start = c(logRate = 0))
summary(fit)
```

```
## --------------------------------------------
## Maximum Likelihood estimation
## Newton-Raphson maximisation, 6 iterations
## Return code 1: gradient close to zero
## Log-Likelihood: -258.5616
## 1  free parameters
## Estimates:
##         Estimate Std. error t value Pr(> t)
## logRate   -1.586      0.100  -15.86  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## --------------------------------------------
```

```
rate_MLE_numerical <- unname(exp(fit$estimate))
rate_MLE_numerical
```

```
## [1] 0.2048215
```

Comparing different estimators

Mean and median

```
c(theoretical = 1/rate,
  empirical = mean(y),
  MLE_numerical = 1/rate_MLE_numerical)
```

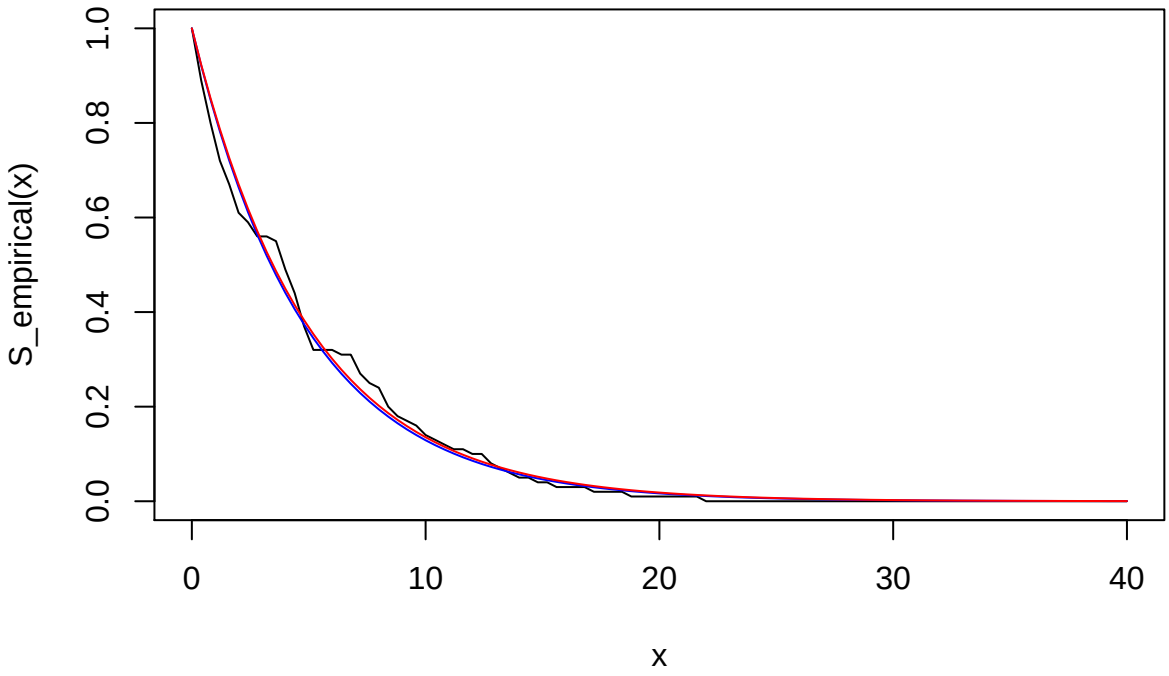
```
##   theoretical     empirical MLE_numerical
##      5.000000      4.882299      4.882299
```

```
c(theoretical = log(2) / rate,
  empirical = median(y),
  MLE_numerical = log(2) / rate_MLE_numerical)
```

```
##   theoretical     empirical MLE_numerical
##      3.465736      3.835192      3.384152
```

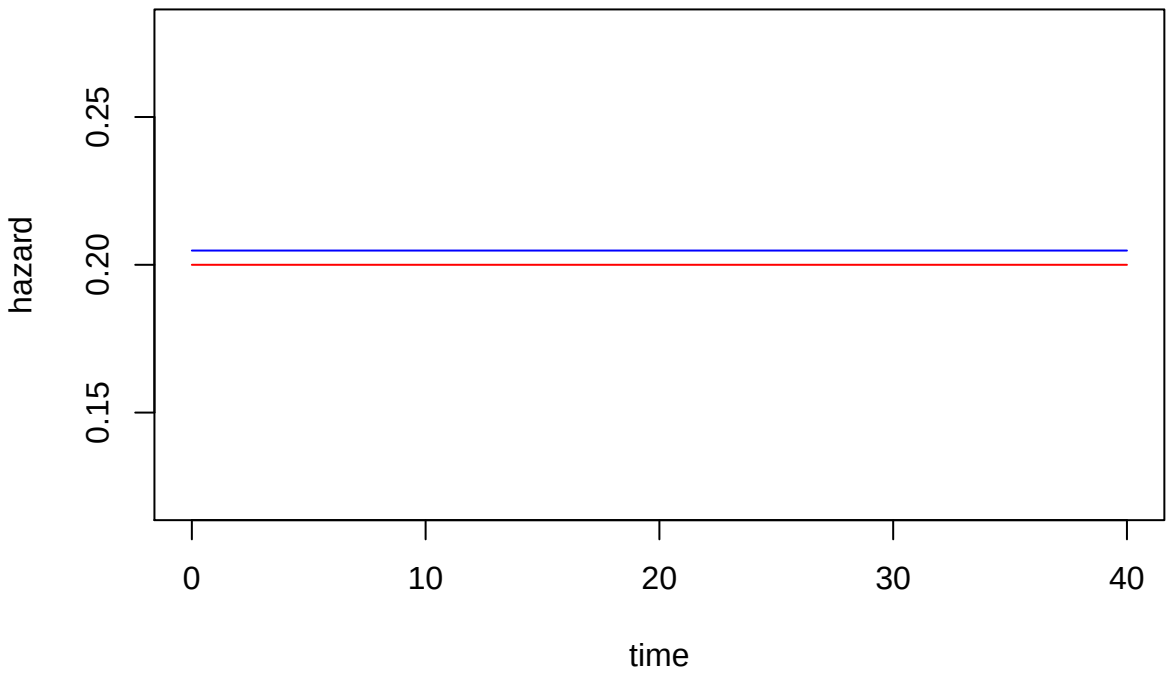
Survival function

```
Fn <- ecdf(y)
S_empirical <- function(t) 1.0 - Fn(t)
S_MLE_numerical <- function(t) pexp(t, rate = rate_MLE_numerical, lower.tail = FALSE)
S_theoretical <- function(t) pexp(t, rate = rate, lower.tail = FALSE)

curve(S_empirical, from = 0, to = 40)
curve(S_MLE_numerical, col = "blue", add = TRUE)
curve(S_theoretical, col = "red", add = TRUE)
```



Hazard function

```
h_theoretical <- Vectorize(function(t) rate)
h_MLE_numerical <- Vectorize(function(t) rate_MLE_numerical)

curve(h_theoretical, from = 0, to = 40, col = "red",
      xlab = "time", ylab = "hazard")
curve(h_MLE_numerical, col = "blue", add = TRUE)
```



Fit the exponential distribution to the hospital duration data

MLE

```
logLik <- function(logRate) {
  sum(dexp(h$dur, rate = exp(logRate), log = TRUE))
}
fit <- maxLik(logLik, start = 0)
summary(fit)
```

```
## --------------------------------------------
## Maximum Likelihood estimation
## Newton-Raphson maximisation, 6 iterations
## Return code 1: gradient close to zero
## Log-Likelihood: -68.06916
## 1  free parameters
## Estimates:
##      Estimate Std. error t value Pr(> t)
## [1,]   -1.723      0.200  -8.613  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## --------------------------------------------
```

A model linking body temperature to risk

Loglikelihood function

```
logLik <- function(theta) {
  alpha <- theta[1]
  beta <- theta[2]
  terms <- dexp(h$dur,
                rate = exp(alpha + beta * (h$temp - 36)),
                log = TRUE)
  sum(terms)
}
```

MLE

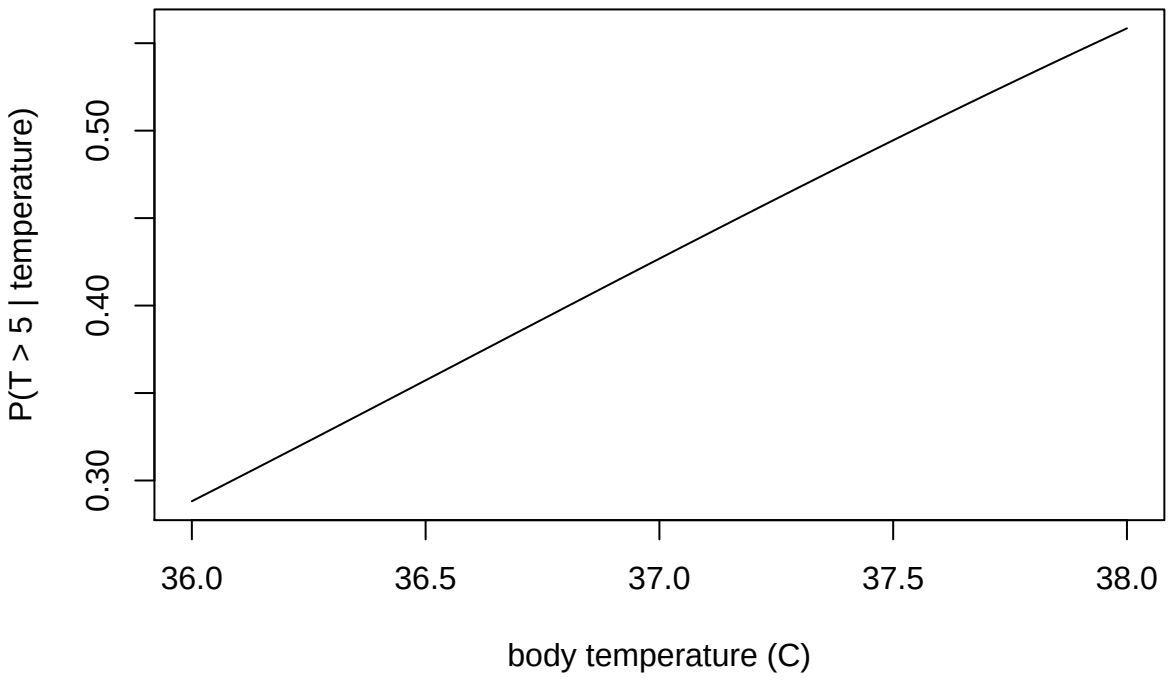
```
fit <- maxLik(logLik, start = c(alpha = 0, beta = 0))
summary(fit)
```

```
## --------------------------------------------
## Maximum Likelihood estimation
## Newton-Raphson maximisation, 6 iterations
## Return code 1: gradient close to zero
## Log-Likelihood: -67.71926
## 2  free parameters
## Estimates:
##       Estimate Std. error t value Pr(> t)
## alpha  -1.3909     0.4281  -3.249 0.00116 **
## beta   -0.3795     0.4517  -0.840 0.40090
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## --------------------------------------------
```

Q. What's the probability of staying 5 days or more?

ow.tail=False we get 1-cdf, so instead of p that duration less equal to 5 we get the inverse

```
curve(pexp(5.0,
           rate = exp(fit$estimate[1] + fit$estimate[2] * (x - 36)),
           lower.tail = FALSE),
      from = 36, to = 38,
      xlab = "body temperature (C)",
      ylab = "P(T > 5 | temperature)")
```



How good is the exponential distribution for our data?

We can only proceed with some approximate, qualitative assessments.

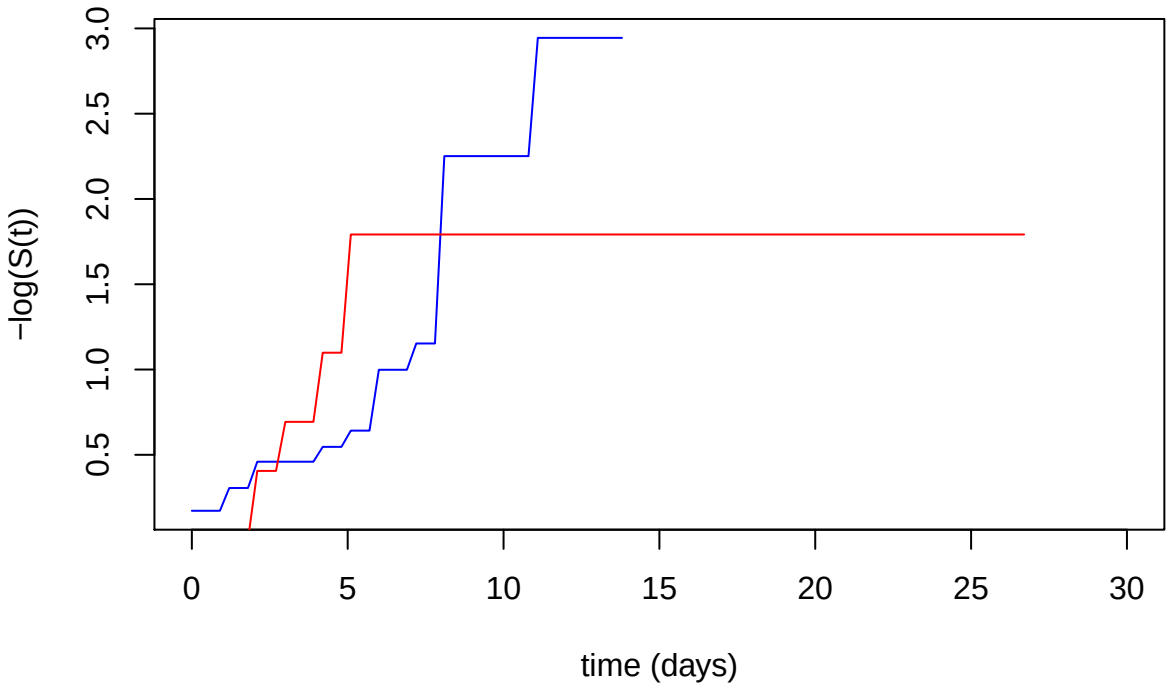
Lets split the body temperature variable into just 2 levels: 'low' (temp<=37) and 'high' (temp>37), and compare empirical and theoretical complementary-log-log survival functions.

```
h_low <- subset(h, 36 <= temp & temp <= 37)
h_high <- subset(h, 37 < temp & temp <= 38)
```

```
clog <- function(y) {
  function(t) {
    S <- 1.0 - ecdf(y)(t)
    -log(S)
  }
}
```

```
curve(clog(h_low$dur)(x),
      from = 0, to = 30, col = "blue",
      xlab = "time (days)",
      ylab = "-log(S(t))")
abline(a = 0, b = exp(fit$estimate[1] + fit$estimate[2] * 36.5),
       col = 'blue', lty = 2)

curve(clog(h_high$dur)(x),
```

```
      from = 0, to = 30, col = "red", add = TRUE)
abline(a = 0, b = exp(fit$estimate[1] + fit$estimate[2] * 37.5),
       col = 'red', lty = 2)
```



The model fits the data rather poorly.

Trying more flexible models (e.g., Weibull) is left as an exercise for the reader.