DATA 604 Assignment 8: Input Analysis

Dan Smilowitz

March 25, 2017

A function is created to analyze given distributions for data using the `fitdistrplus` library:

```
# extract ggplot legends: github.com/hadley/ggplot2/wiki/Share-a-legend-between-two-ggplot2-graphs
g_leg <- function(a.gplot) {
  tmp <- ggplot_gtable(ggplot_build(a.gplot))
  leg <- which(sapply(tmp$grobs, function(x) x$name) == "guide-box")
  legend <- tmp$grobs[[leg]]
  return(legend)
}

find_fit <- function(x, dists) {
  # set up container for fits
  fits <- vector(mode = 'list', length = length(dists))
  names(fits) <- dists
  # fit each distribution
  for (i in 1:length(dists)) {
    d <- dists[i]
    fits[[d]] <- fitdist(x, d)
  }
  # create plots comparing fits
  p1 <- denscomp(fits, plotstyle = 'ggplot', legendtext = dists,
                 demp = TRUE, dempcol = 'black', fitlty = 1) +
    theme_minimal() + theme(legend.position='bottom', legend.direction = 'horizontal')
  p2 <- qqcomp(fits, plotstyle = 'ggplot', legendtext = dists, ylab = 'Emp. quantiles') +
    theme_minimal() + theme(legend.position='bottom', legend.direction = 'horizontal')
  p3 <- cdfcomp(fits, plotstyle = 'ggplot', legendtext = dists, fitlty = 1) +
    theme_minimal() + theme(legend.position='none')
  p4 <- ppcomp(fits, plotstyle = 'ggplot', legendtext = dists, ylab = 'Emp. probability') +
    theme_minimal() + theme(legend.position='none')
  # create legends
  leg_line <- g_leg(p1)
  leg_point <- g_leg(p2)
  # arrange plots & legends; draw box
  grid.arrange(p1 + theme(legend.position = 'none'), p2 + theme(legend.position = 'none'),
               p3, p4, leg_line, leg_point, nrow = 3, heights = c(0.43, 0.43, 0.14))
  grid.rect(width = 0.99, height = 0.99, gp = gpar(fill = NA))
  # get log likelihoods & find max
  l <- unlist(lapply(fits, `[[`, 'loglik'))
  f <- which.max(l)
  # create table of log likelihoods
  emphasize.strong.cols(f)
  pander(l)
  # get & return estimated parameters
  f <- names(f)
  e <- fits[[f]]$estimate
  return(e)
}
```

The function takes data and a vector of distributions as inputs and produces apprpriate plots and a table of log likelihood of the fit distributions, and returns parameter estimates for the fit with the best log likelihood. The following libraries are required to execute this function:

```
library(fitdistrplus)
library(ggplot2)
library(gridExtra)
library(grid)
library(pander)
```

Data for problems 1-3 is combined into a single Excel file with multiple sheets and read into R:

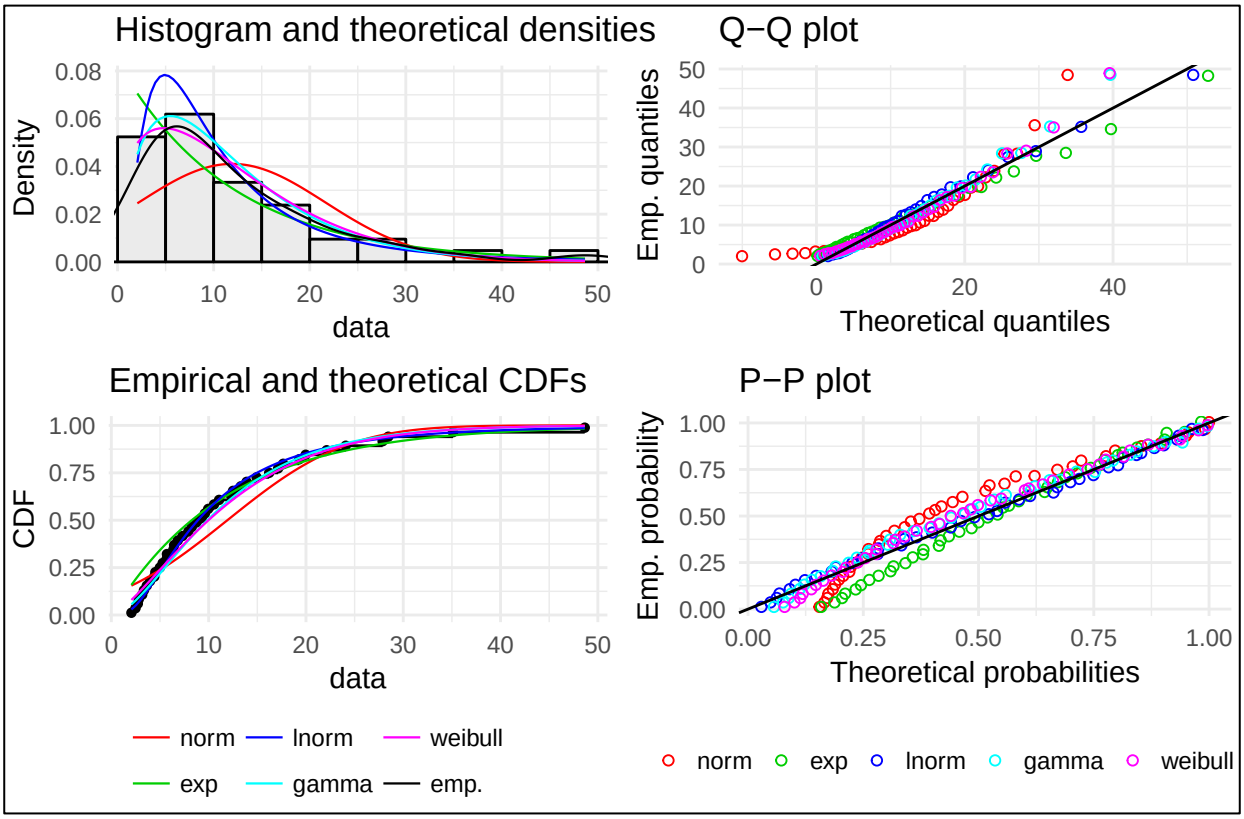
```
library(readxl)
q1 <- read_excel('wk8.xlsx', sheet = '06_01', col_names = FALSE)$X0
q2 <- read_excel('wk8.xlsx', sheet = '06_02', col_names = FALSE)$X0
q3 <- read_excel('wk8.xlsx', sheet = '06_03', col_names = FALSE)$X0
```

The distributions that will be fit to this data are stored:

```
d_cont <- c('norm', 'exp', 'lnorm', 'gamma', 'weibull')
d_disc <- c('geom', 'nbinom')
```

Problem 1

The relevant plots and log likelihoods of the fit distributions are presented below:

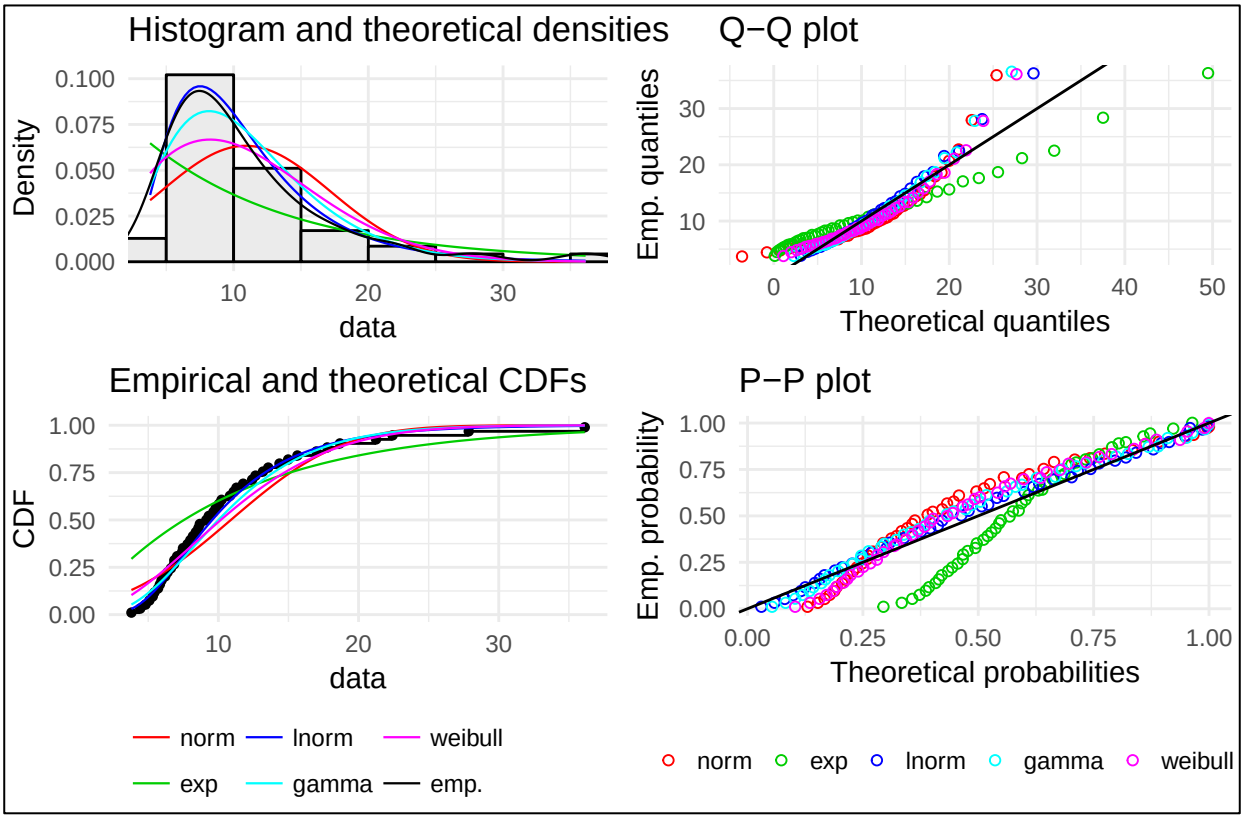


norm	exp	lnorm	gamma	weibull
-155.1	-146.1	**-140.5**	-142.1	-143.2

Based on the information above and the estimated fit paramters, the distribution that should be used to generate interarrival times is a log-normal distribution with parameters $\mu = 2.1851$ and $\sigma = 0.7712$. The associated Simio expression is `Random.Lognormal(2.1851, 0.7712)`.

Problem 2

As above, the continuous distributions are fit to the data, returning the following results:

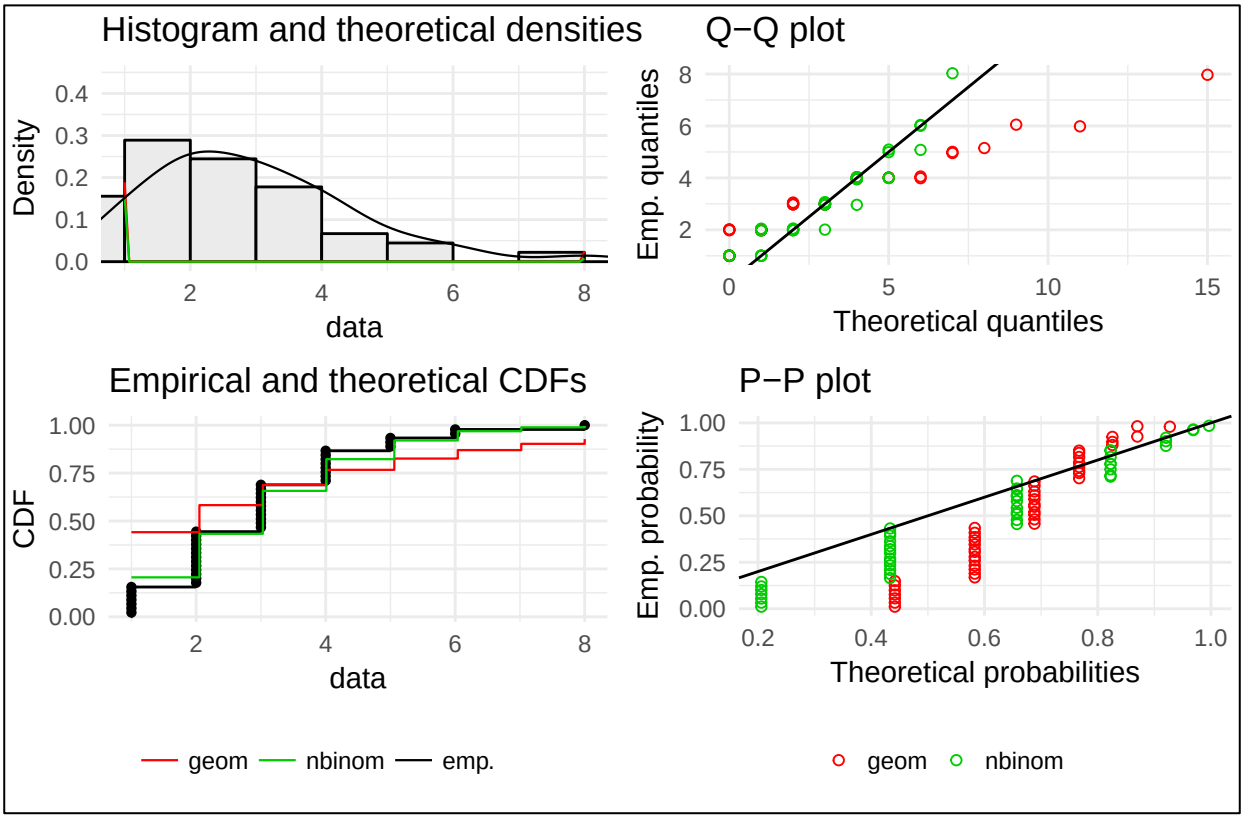


norm	exp	lnorm	gamma	weibull
-153.2	-159.2	**-139.3**	-142.2	-146.1

Based on the information above and the estimated fit paramters, the distribution that should be used to generate call durations is a log-normal distribution with parameters $\mu = 2.2579$ and $\sigma = 0.4906$. The associated Simio expression is `Random.Lognormal(2.2579, 0.4906)`.

Problem 3

The same procedure as Problems 1 & 2 is followed, this time fitting discrete distributions:



geom	nbinom
-100.6	**-81.14**

Based on the table above and the estimated fit paramters, the distribution that should be used to generate the number of extra tech-support people needed is a negative binomial distribution with parameters $r = 4.1369309 \times 10^6$ and $\mu = 2.9554$.

Simio accepts a probability parameter p rather than a mean parameter μ, where $p = r/(r + \mu)$; therefore the associated Simio expression is `Random.NegativeBinomial(4136931, 0.9999993)`.

Problem 4

The CDF for the uniform distribution is given by

$$F_X(x) = \begin{cases} 0 & x < a \\ \frac{x-a}{b-a} & a \le x \le b \\ 1 & x > b \end{cases}$$

Setting the middle section equal to u and solving for u yields

$$F_X^{-1}(u) = u(b-a) + a$$

Problem 5

The CDF for the Weibull distribution, for $x \geq 0$, is

$$F_X(x) = 1 - e^{(-x/\lambda)^k}$$

Setting the $F_X(x) = u$ and solving for u yields

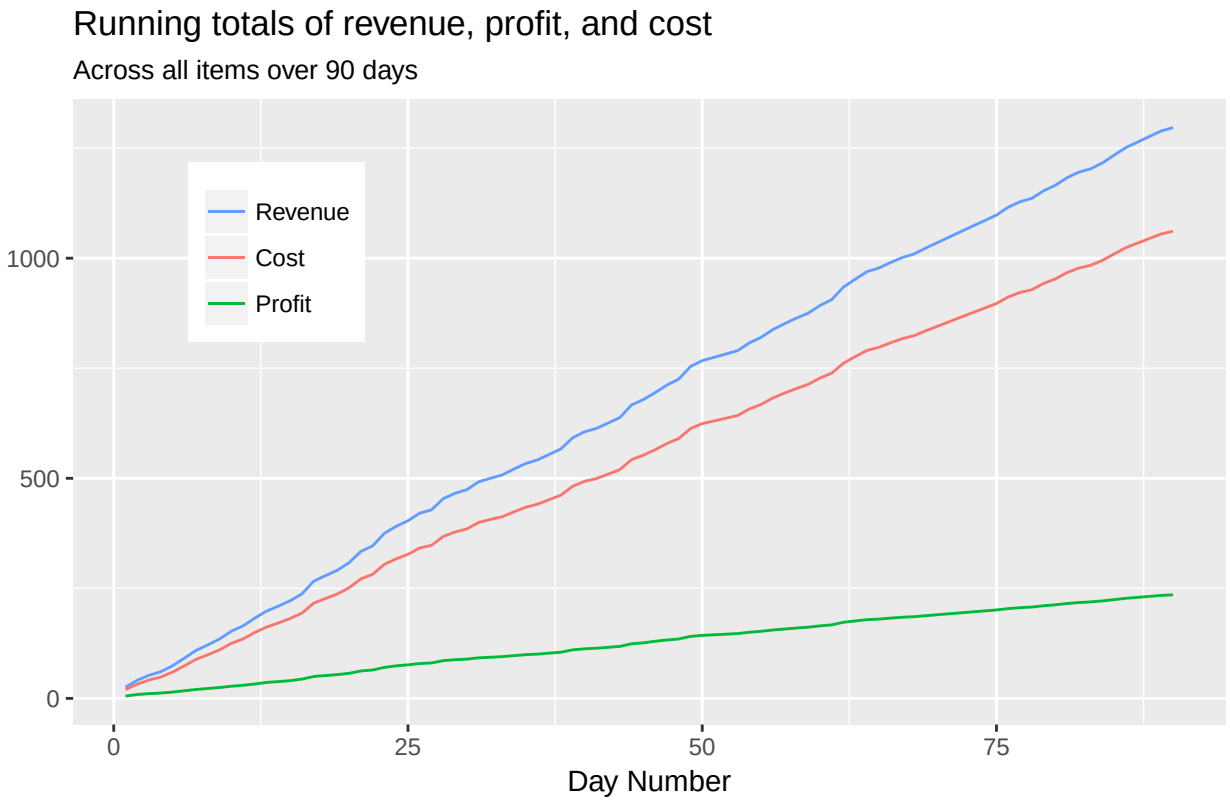
$$F_X^{-1}(u) = \lambda \left[-\ln(1-u)\right]^{1/k}$$

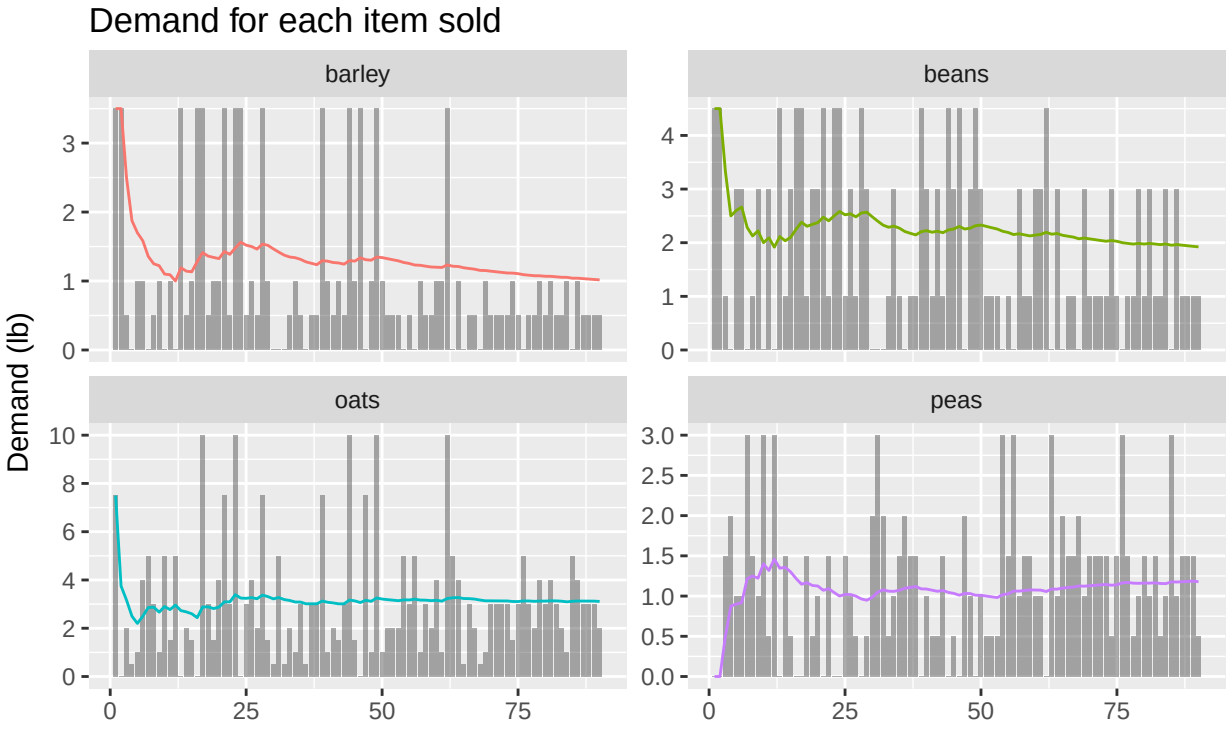
Problem 8

```
# set up provided info
produce <- c('oats', 'peas', 'beans', 'barley')
wholesale <- c(1.05, 3.17, 1.99, 0.95)
retail <- c(1.29, 3.76, 2.23, 1.65)
num_days <- 90

# set up demands
demand <- list()
demand[['oats']] <- data.frame(
  weight = c(0, 0.5, 1, 1.5, 2.0, 3.0, 4.0, 5.0, 7.5, 10.0),
  p = c(0.05, 0.07, 0.09, 0.11, 0.15, 0.25, 0.10, 0.09, 0.06, 0.03)
)
demand[['peas']] <- data.frame(
  weight = c(0, 0.5, 1.0, 1.5, 2.0, 3.0), p = c(0.1, 0.2, 0.2, 0.3, 0.1, 0.1)
)
demand[['beans']] <- data.frame(
  weight = c(0, 1.0, 3.0, 4.5), p = c(0.2, 0.4, 0.3, 0.1)
)
demand[['barley']] <- data.frame(
  weight = c(0, 0.5, 1.0, 3.5), p = c(0.2, 0.4, 0.3, 0.1)
)

# create container list for simulation results
walther <- list()

library(dplyr)
# create simulation for each list
for (p in 1:length(produce)) {
  # get 90 simulated sales for produce item
  set.seed(42) # set random number generator seed for replicability
  sales <- sample(demand[[produce[p]]]$weight, prob = demand[[produce[p]]]$p,
                  size = num_days, replace = TRUE)
  # store day number and calculate revenue, cost, and profit
  df <- data.frame(day_no = 1:num_days, sales) %>%
    mutate(revenue = sales * retail[p],
           cost = sales * wholesale[p],
           profit = revenue - cost)
  # add item name
  df$item <- rep(produce[p], num_days)
  # store in list
  walther[[p]] <- df
}
```

```r
# collapse list into single data frame (tbl_df)
walther <- tbl_df(bind_rows(walther))

# calculate daily and running sums
daily <- walther %>%
  group_by(day_no) %>%
  summarise(revenue = sum(revenue),
            cost = sum(cost),
            profit = sum(profit)) %>%
  mutate(
    running_revenue = cumsum(revenue),
    running_cost = cumsum(cost),
    running_profit = cumsum(profit)
  )

# calculate total revenue, cost, and profit
tot <- daily[nrow(daily), 5:7]

# get daily and average sales by item
item <- walther %>%
  group_by(item) %>%
  mutate(avg_sales = cummean(sales))
```

The simulation generated $1296.52 in revenue with an associated cost of $1061.22, yielding a profit of $235.31. The running totals of these numbers across the 90 days simulated are shown below, followed by plots of the daily demand for each item.





Bars = Daily Value; Lines = Average Value