Part I: Maximum Likelihood Estimation of Willingness-To-Pay Distribution

Introduction

Willingness-To-Pay (WTP) is the maximum amount that a customer is willing to pay for a product. A customer will purchase the product if and only if his/her WTP is larger than or equal to the price. For a population of customers, their WTP collectively forms a distribution. Understanding the WTP distribution plays a critical role in pricing and other related activities.

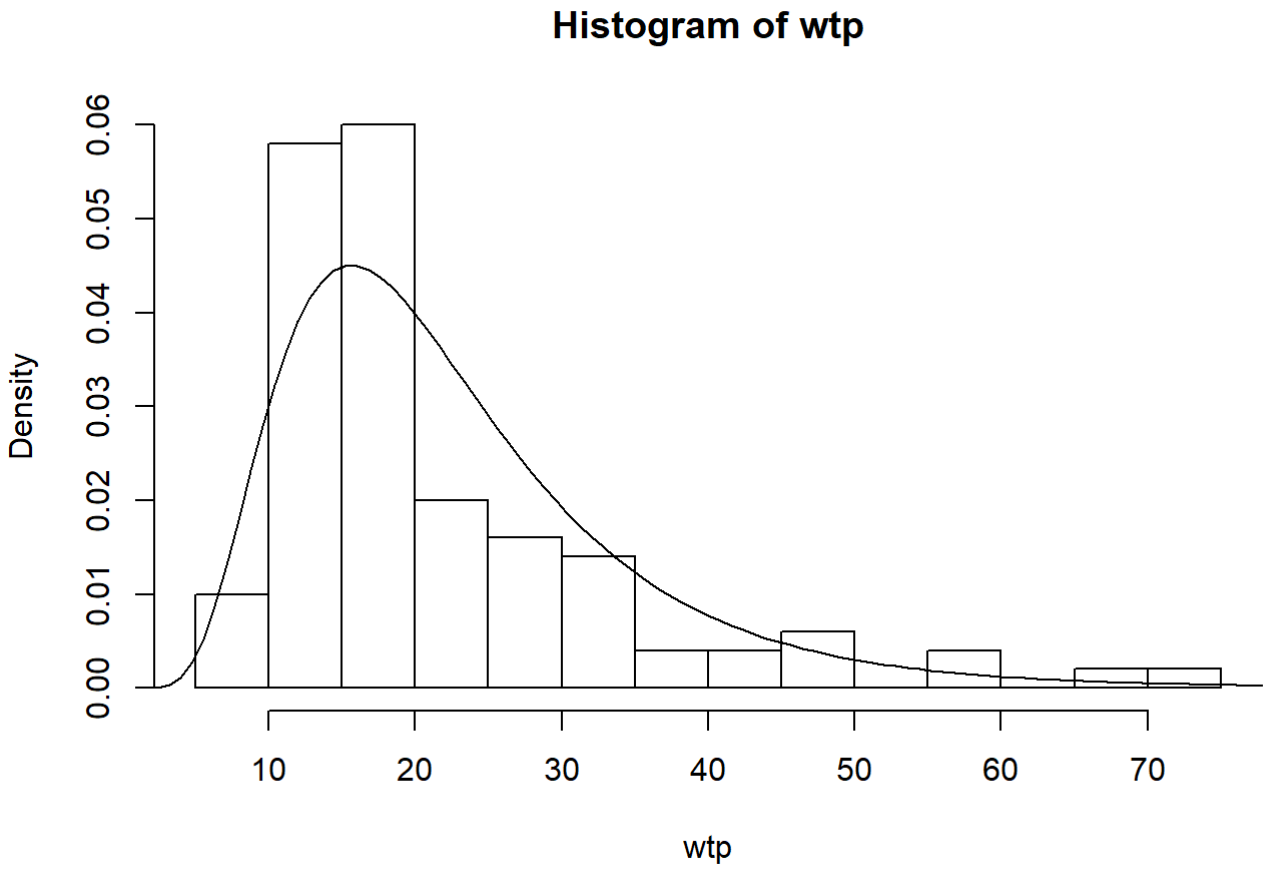
There are not many good ways to learn the WTP distribution. Often, we need to estimate it from the customers' actual purchase decisions. In the following dataset, we simulate 100 historical transactions with the customers' true **WTP**, the **price** offered to the customer, and **purchase** that indicates whether the customer took the deal or not. Below is how the data file is generated

```
set.seed(1234)  # set seed for random number generator
N <- 100  # number of data points
wtp <- rlnorm(N, meanlog=3, sdlog=0.5)  # WTP follows a log-normal distribution with (3, 0.5)
data <- data.frame(price=round(runif(N, min=10, max=40)))  # price is uniform[10, 40], rounded
 to integers
data$purchase <- (wtp >= data$price)  # purchase is TRUE if WTP >= price
summary(data)
```

```
##      price        purchase
##  Min.   :10.00   Mode :logical
##  1st Qu.:18.00   FALSE:72
##  Median :25.50   TRUE :28
##  Mean   :25.04   NA's :0
##  3rd Qu.:32.00
##  Max.   :40.00
```

The true WTP has mean 21.2910975 and standard deviation 12.7693991, and its distribution is as below.

```
hist(wtp, breaks=15, probability=TRUE)
curve(dlnorm(x, 3, 0.5), from=0, to=80, add=TRUE)
```



In the following, we shall try to estimate the WTP distribution based on the data with **price** and **purchase**information only.

Questions and Answers

1. Suppose that we believe that the customers are from a population with their WTP following a *normal* distribution $\mathrm{N}(\mu, \sigma^2)$ N(μ,σ2). Estimate μμ and σ^2 σ2 using MLE. (1 Mark)

Answer:

```
# Formulate the log-likelihood function
LL.normal <- function(theta, data) {
    mu <- theta[1]
    sigma <- theta[2]
    ll1 = pnorm(data$price, mean=mu, sd=sigma, lower.tail=FALSE, log.p=TRUE)
    ll2 = pnorm(data$price, mean=mu, sd=sigma, lower.tail=TRUE, log.p=TRUE)
    ll = ifelse(data$purchase,ll1, ll2)
    return(sum(ll))
}

# Invoke optim() to find the optimal parameters
output.normal <- optim(c(0, 1), LL.normal, method="L-BFGS-B", lower=c(-Inf, 1e-6), upper=c(Inf,
Inf), control=list(fnscale=-1), data=data)
output.normal # optimal parameters are in output.normal$par
```

```
## $par
## [1] 16.95320 11.62228
##
## $value
## [1] -48.13454
##
## $counts
## function gradient
##       23       23
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Mean is estimated to be 16.9531969 and variance is estimated to be 135.0773142.

2. Now suppose the customersâ□□ WTP follows a *Gamma* distribution with shape parameter αα and rate parameter ββ, estimate αα and ββ using MLE. (hint: check out the pgamma() function in R) (1 Mark)

Answer:

```
# Formulate the log-likelihood function
LL.gamma <- function(theta, data) {
    shape <- theta[1]
    rate <- theta[2]
    ll1 = pgamma(data$price, shape=shape, rate=rate, lower.tail=FALSE, log.p=TRUE)
    ll2 = pgamma(data$price, shape=shape, rate=rate, lower.tail=TRUE, log.p=TRUE)
    ll = ifelse(data$purchase,ll1, ll2)
    return(sum(ll))
}

# Invoke optim() to find the optimal parameters
output.gamma <- optim(c(0, 1), LL.gamma, method="L-BFGS-B", lower=c(1e-6, 1e-6), upper=c(Inf, I
nf), control=list(fnscale=-1), data=data)
output.gamma # optimal parameters are in output.gamma$par
```

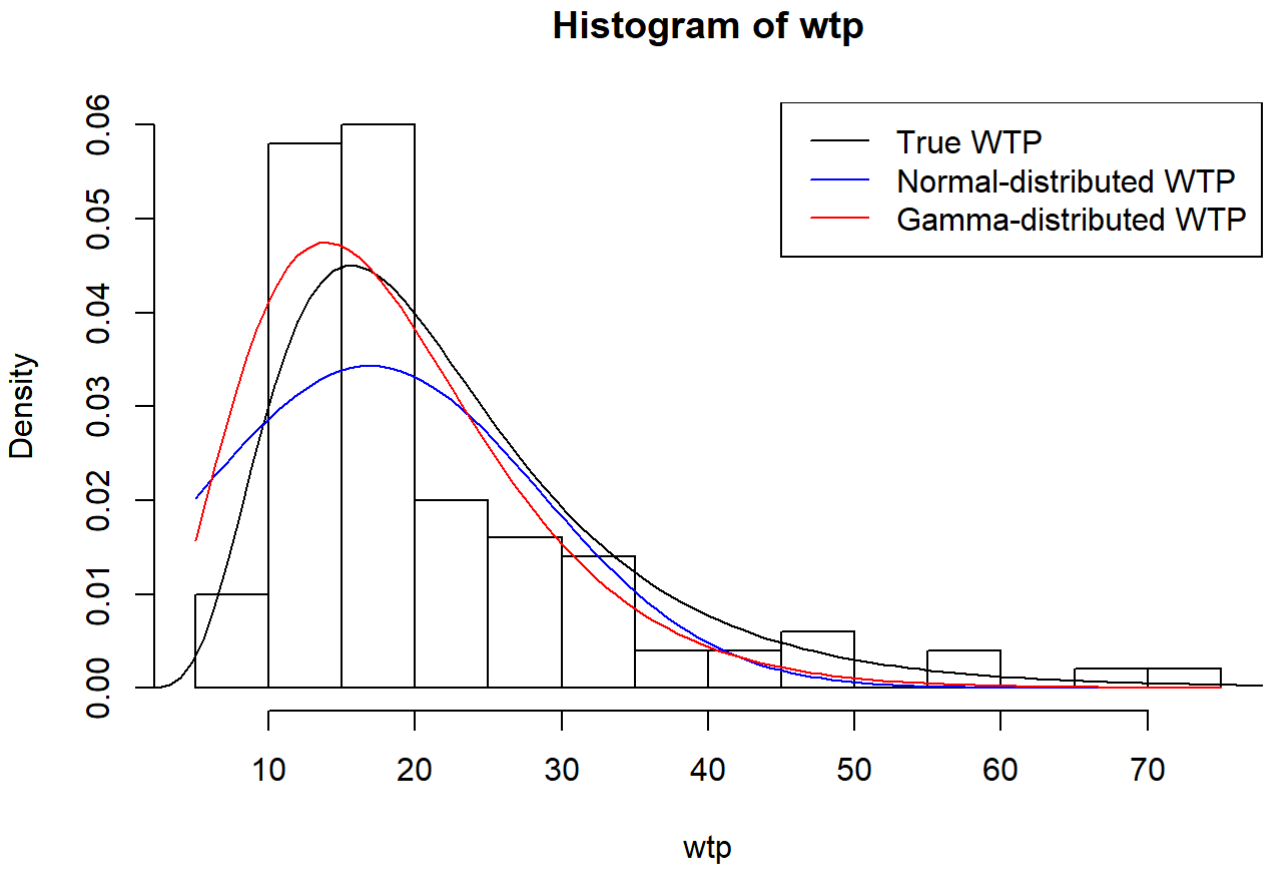
```
## $par
## [1] 3.8975483 0.2085433
##
## $value
## [1] -46.50905
##
## $counts
## function gradient
##       29       29
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Alpha is estimated to be 3.8975483 and beta is estimated to be 0.0434903.

3. Plot the distributions estimated in Q1 and Q2 together with the true WTP distribution from the data generating model. Briefly comment on the estimated distributions. (1 Mark)

Answer:

```
hist(wtp, breaks=15, probability=TRUE)
curve(dlnorm(x, 3, 0.5), from=0, to=80, add=TRUE)
curve(dnorm(x, output.normal$par[1], output.normal$par[2]), add=TRUE, col="blue") # normal-dist
ributed WTP in blue
curve(dgamma(x, output.gamma$par[1], output.gamma$par[2]), add=TRUE, col="red") # gamma-distrib
uted WTP in red
legend("topright", c("True WTP", "Normal-distributed WTP", "Gamma-distributed WTP"), col=c("bla
ck", "blue", "red"), lwd=1)
```



The gamma distribution is closer to the true wtp distribution with a similar peak and skewness to the left. The normal distribution is bell-shaped with a lower peak compared to the true distribution.

Even with limited observations on the price and purchase observations, we can estimate the WTP distribution. Even if we assumed the wrong underlying distribution, MLE will find a distribution within the assumed family that is “closest” to the true distribution.

Part II: MLE of the Probit model

Introduction

In the previous questions, we tried to estimate the WTP distribution from their binary purchase decisions and price information. There, we assumed WTP is simply random and is drawn from a distribution every time a customer shows up. Now think about a more general version of the problem in the sense that the WTP actually depends on other factors such as product quality, weather, or whatever. For the sake of exercise, suppose there are two such (continuous) factors **X1** and **X2** that we are able to keep track of. Every customer, at the time of

walking into your shop, first observes X_1 X1 and X_2 X2, then thinks about what his/her WTP is, and finally compare the WTP with the price and make a purchase decision. The whole dataset we will have should look like this:

Transaction id	Price	X1	X2	Purchase
1	30	1.8	24	FALSE
2	26	2.5	12	FALSE
…	…	…	…	…

Generating Data

Let's construct a simple linear model that incorporates such dependency on factors X_1 X1 and X_2 X2:

$$WTP = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon.$$
WTP=β0+β1X1+β2X2+ϵ.

Here $\epsilon \sim N(0, \sigma^2)$ ϵ~N(0,σ2). We next simulate a dataset like the table above.

```
# true parameters
beta0 <- -2
beta1 <- 3
beta2 <- -1
sigma <- 2

set.seed(1234)  # set seed for random number generator
N <- 200  # number of data points
X1 <- runif(N, min=1, max=3)  # X1 ~ Uniform(1, 3)
X2 <- rexp(N, rate=1)  # X2 ~ Exponential(1)
price <- rgamma(N, shape=2, rate=1)  # price ~ Gamma(2, 1)
data <- data.frame(id=1:N, x1=X1, x2=X2, price=price)

# simulate the observations according to the model we built
epsilon <- rnorm(N, mean=0, sd=sigma)  # epsilon ~ N(0, sigma)
wtp <- beta0 + beta1 * data$x1 + beta2 * data$x2 + epsilon
data$purchase <- (wtp >= data$price)
head(data)
```

```
##   id       x1         x2     price purchase
## 1  1 1.227407 0.32150927 4.1534871    FALSE
## 2  2 2.244599 0.05671876 1.8000617     TRUE
## 3  3 2.218549 0.96312247 2.6158244     TRUE
## 4  4 2.246759 0.53571094 0.3014186     TRUE
## 5  5 2.721831 0.05261699 2.9331581     TRUE
## 6  6 2.280621 0.46460377 1.1031438     TRUE
```

Finally, we can split the simulated data into two parts (60%–40%). The first part will be the training data, and the second part will be used for testing. Since the data points are independent, we can just take the first 60% as a random training sample.

```
data.train <- data[1:round(N * 0.6), ]
data.test <- data[(round(N * 0.6) + 1):N, ]
```

Questions and Answers

1. Estimate the model parameters using MLE on the training data. Hint: how to write the likelihood? It should be related to Prob(Purchase == 1) and Prob(Purchase == 0). (1 Mark)

Answer:

```
# Formulate the log-likelihood function
LL.training <- function(theta, data) {
    beta0 <- theta[1]
    beta1 <- theta[2]
    beta2 <- theta[3]
    sigma <- theta[4]
    ll1 = pnorm(data$price- beta0 - beta1 * data$x1 - beta2 * data$x2, mean=0, sd=sigma, lower.
tail=FALSE, log.p=TRUE)
    ll2 = pnorm(data$price- beta0 - beta1 * data$x1 - beta2 * data$x2, mean=0, sd=sigma, lower.
tail=TRUE, log.p=TRUE)
    ll = ifelse(data$purchase, ll1, ll2)
    return(sum(ll))
}

# Invoke optim() to find the optimal parameters
output.training <- optim(c(0, 0, 0, 1), LL.training, method="L-BFGS-B", lower=c(-Inf, -Inf, -In
f, 1e-6), upper=c(Inf, Inf, Inf, Inf), control=list(fnscale=-1), data=data.train)
output.training # optimal parameters are in output.training$par
```

```
## $par
## [1] -2.412665  3.236513 -1.128806  1.581025
##
## $value
## [1] -43.82024
##
## $counts
## function gradient
##       20       20
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

The estimates for Beta0, Beta1, Beta2 and sigma are -2.4126654, 3.2365133, -1.1288058 and 1.5810247 respectively.

2. Having obtained the estimate, can you predict â□□Purchaseâ□□ decision of future customers when Price, X1, and X2 are given? Use the second half of the data (Test) for prediction, and compare the true â□□Purchaseâ□□ and predicted â□□Purchaseâ□□ in the Test dataset. (1 Mark)

Answer:

```
data.test$predicted.wtp <- output.training$par[1] + output.training$par[2] * data.test$x1 + out
put.training$par[3] * data.test$x2
data.test$purchase.prediction <- ifelse(data.test$predicted.wtp > data.test$price, TRUE, FALSE)
(sum(data.test$purchase.prediction != data.test$purchase)/nrow(data.test))*100
```

```
## [1] 17.5
```

The number of times our purchase prediction differed from the actual purchase is 14. This accounts for 17.5% of the Test dataset.

Session Info

```
print(sessionInfo(), locale=FALSE)
```

```
## R version 3.3.3 (2017-03-06)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18362)
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] backports_1.0.5 magrittr_1.5    rprojroot_1.2   formatR_1.4
##  [5] tools_3.3.3     htmltools_0.3.5 Rcpp_0.12.6     stringi_1.1.1
##  [9] rmarkdown_1.3   knitr_1.14      stringr_1.0.0   digest_0.6.9
## [13] evaluate_0.9
```