Unbabel Challenge

Federico Di Martino

April 3rd 2019

Setup

```
## Load packages
library(tidyverse)

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

## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  2.0.1     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0

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

## set seed
set.seed(12345)
```

Loading in the data

```
## Load in each csv file
clients <- read.csv(".\\dataset\\clients.csv")
editors <- read.csv(".\\dataset\\editors.csv")
tasks   <- read.csv(".\\dataset\\tasks.csv")
tickets <- read.csv(".\\dataset\\tickets.csv")

## Initialise placeholder constants
## Place holder value = 1 because this would have the
## least effect on the equations
constant_alpha <- 1 ## Placeholder
constant_beta <- 1  ## Placeholder
constant_gamma <- 1 ## Placeholder



## Rename ids, Xs, to avoid duplicates
names(clients)[2] <- "client_id"
names(editors)[2] <- "editor_id"
names(tasks)[2]   <- "task_id"
names(tickets)[2] <- "ticket_id"
```

```
names(clients)[1] <- "X_clients"
names(editors)[1] <- "X_editors"
names(tasks)[1]   <- "X_tasks"
names(tickets)[1] <- "X_tickets"
```

What questions are we asking?

The Customers' Problem

Customers are reporting that the quality is not stable. So I shall examine the quality distribution and try to decrease it's variance, without causing the median to decrease.

The Editors' Problem

A large group of editors are reporting that they're not getting any tasks. I will look at the assignment probability and then the price distribution, with an eye at increasing the latter.

Preliminary Data Wrangling

Some of the variables in the challenge instructions aren't entirely defined so I will be making some assumptions about their complete meaning.

Language Pairs

In the data, only the language pair of each ticket is given, not those of the editors. So there is no way to know if there is correspondence.

Data Relationships

Combine data tables such that calculations can be done from each row.

First, figure what the columns are:

There are columns in the data set marked only as X. They are not mentioned in instruction booklet. They are: In clients, unknown, all unique values. In editors, unknown, all unique values. In tasks it's clearlt just the row number offset by 1 so we should drop it. In tickets it looks like the row number with occasional "jumps", unknown what it represents, all unique values. Apart from tasks X, I have decided to not drop any unless they later anbsolutely proven to be irrelevant.

In tickets there are separate client_id and client_id.1 columns. There are the same number (50) of unique values in .1 as there are unique values of the "id" in the clients table, with the same format of values. Will check to see if any of these correspond. That leaves client_id to investigate. Will see how it compares to "X" in the clients table.

Investigating the Variables

```
## X from clients, client_id from tickets
X_client_id_intersection <- intersect(clients$X_clients,
tickets$client_id)
length(X_client_id_intersection)

## [1] 11

## id from clients, client_id.1 from tickets
id_client_id.1_intersection <- intersect(clients$client_id,
tickets$client_id.1)
length(id_client_id.1_intersection )

## [1] 50
```

X, client_id intersection contains only 11 values, since they are both numbers of uncertain origin I don't think they are really related.

id, client_id.1 intersection is a perfect union. I am concluding that client_id.1 is the real variable for the client ids. The original client_id shall be renamed to something else for now, but not dropped, since it might still be something significant.

```
names(tickets)[3] <- "unknown_client_id"
names(tickets)[9] <- "client_id"
```

Combining Tables

```
## Drop unneccesary variables
tasks <- select(tasks, -X_tasks)



## Combine tables

tasks_tickets <- left_join(tasks, tickets, by = "ticket_id")

## Warning: Column `ticket_id` joining factors with different levels,
coercing
## to character vector

tasks_tickets_clients <- left_join(tasks_tickets, clients, by =
"client_id")
```

Assigning tasks to editors.

The tasks are randomly assigned, I am assuming a uniform distribution, for now. This means that each editor get approximately the same number of tasks. This shifts the problem to solve for the editors to ensuring the prices are high enough.

```
## Assign random editor id to each row
assigned_editors <- tasks_tickets_clients
assigned_editors$editor_id <- sample(editors$editor_id ,
```

```
                                        size = nrow(assigned_editors) ,
replace = TRUE)

## Join editor table onto assigned table
assigned_all <- left_join(assigned_editors, editors, by = "editor_id")
```

Calculating the price of each task and ticket

The price of each task is calculated differently depending on whether the language pair of the editor is contained in the set of he language pairs of the task. However, in the data, there is no indication as to what the language pairs of the editors are. So I will calculate the prices of the tasks assuming that the language pairs always match. If they didn't match, the prices would be different but the quality would always be 0 and I'm not sure how much valuable analysis can be done assuming no quality to the work. The price of each ticket is defined as the average of the price of the related tasks.

Price Calculation

```
##renaming number_words variables to clarify one for tasks and other
for tickets
names(assigned_all)[2] <- "number_words_task"
names(assigned_all)[7] <- "number_words_ticket"


## Calculating Price per task, assuming all language pairs match
calculated_price_task <- mutate(assigned_all,
                                price = case_when
                                 (domain == "travel"
                                  ~ constant_alpha * number_words_task
* travel,
                                  domain == "fintech"
                                  ~ constant_alpha * number_words_task
* fintech,
                                  domain == "health_care"
                                  ~ constant_alpha * number_words_task
* health_care,
                                  domain == "ecommerce"
                                  ~ constant_alpha * number_words_task
* ecommerce,
                                  domain == "sports"
                                  ~ constant_alpha * number_words_task
* sports,
                                  domain == "gamming"
                                  ~ constant_alpha * number_words_task
* gamming
                                 )
                                )
```

```
calculated_price_ticket <-calculated_price_task  %>%
group_by(ticket_id) %>% mutate(price_ticket = mean(price))

calculated_price_ticket <- ungroup(calculated_price_ticket)
```

Calculating the Quality

Divide the space Q (0-100) into S-1 buckets. S = 5 therefore Q will be divided into 4 buckets. First we need too look at how each editor is assigned to each interval.

Quality Interval Assignment

D is the domain. So to calculate quality I will cycle through each skill level per relevant task domain. There are 5 skill levels and 4 intervals, so I have decided to assign the bottom two skill levels to the bottom interval.

Total probability must be equal to 1. To ensure that we can calculate a value for the constant beta.

Therefore with equally sized intervals,

```
## P(A) = 0.25, A = 25, 4 intervals.

constant_beta = (1-(3 * ( 0.25/25 ) ) )/ (0.25 * 25)
```

beta = constant_beta

So for each task, there is a 97% probability that the interval assigned is the one where S(E) = D and a 1% one to be assigned to each of the other ones.

Domain identification

```
## For these steps use intermediate variables qc_1, _2 etc. (quality
calculation)

## Generate variable with four randomly assigned levels
## One 97% likely to be generated, the others 1%
qc_1 <- mutate(calculated_price_ticket, prob_levels = sample(c("a",
"b", "c", "d"),
                                                             size =
nrow(calculated_price_ticket),
                                                             prob =
c(0.97, 0.01, 0.01, 0.01),
                                                             replace =
TRUE) )

## identify domain skill used for task, create skill used column to
simplify
## following steps by avoiding checking domain each time
qc_2 <- mutate(qc_1, skill_used = case_when(
```

```
                    domain == "travel"      ~ travel,
                    domain == "fintech"     ~ fintech,
                    domain == "health_care" ~  health_care,
                    domain == "ecommerce"   ~ ecommerce,
                    domain == "sports"      ~ sports,
                    domain == "gamming"     ~ gamming
                  )
                )
```

Interval Level Assignment for Each Skill Level

```
## Use conditionals (||) and random level assignment (a,b etc) in here
qc_3 <- mutate(qc_2 , interval = case_when(
  skill_used == 1 & prob_levels == "a" ~ "int1",
  skill_used == 1 & prob_levels == "b" ~ "int2",
  skill_used == 1 & prob_levels == "c" ~ "int3",
  skill_used == 1 & prob_levels == "d" ~ "int4",

  skill_used == 2 & prob_levels == "a" ~ "int1",
  skill_used == 2 & prob_levels == "b" ~ "int2",
  skill_used == 2 & prob_levels == "c" ~ "int3",
  skill_used == 2 & prob_levels == "d" ~ "int4",

  skill_used == 3 & prob_levels == "b" ~ "int1",
  skill_used == 3 & prob_levels == "a" ~ "int2",
  skill_used == 3 & prob_levels == "c" ~ "int3",
  skill_used == 3 & prob_levels == "d" ~ "int4",

  skill_used == 4 & prob_levels == "b" ~ "int1",
  skill_used == 4 & prob_levels == "c" ~ "int2",
  skill_used == 4 & prob_levels == "a" ~ "int3",
  skill_used == 4 & prob_levels == "d" ~ "int4",

  skill_used == 5 & prob_levels == "b" ~ "int1",
  skill_used == 5 & prob_levels == "c" ~ "int2",
  skill_used == 5 & prob_levels == "d" ~ "int3",
  skill_used == 5 & prob_levels == "a" ~ "int4"




  )
)
```

Calculating the quality

Decided to assign the quality for each interval as the upper limit of that interval, 25, 50, 75 and 100 respectively. The quality for a ticket is defined as the average of the task qualities.

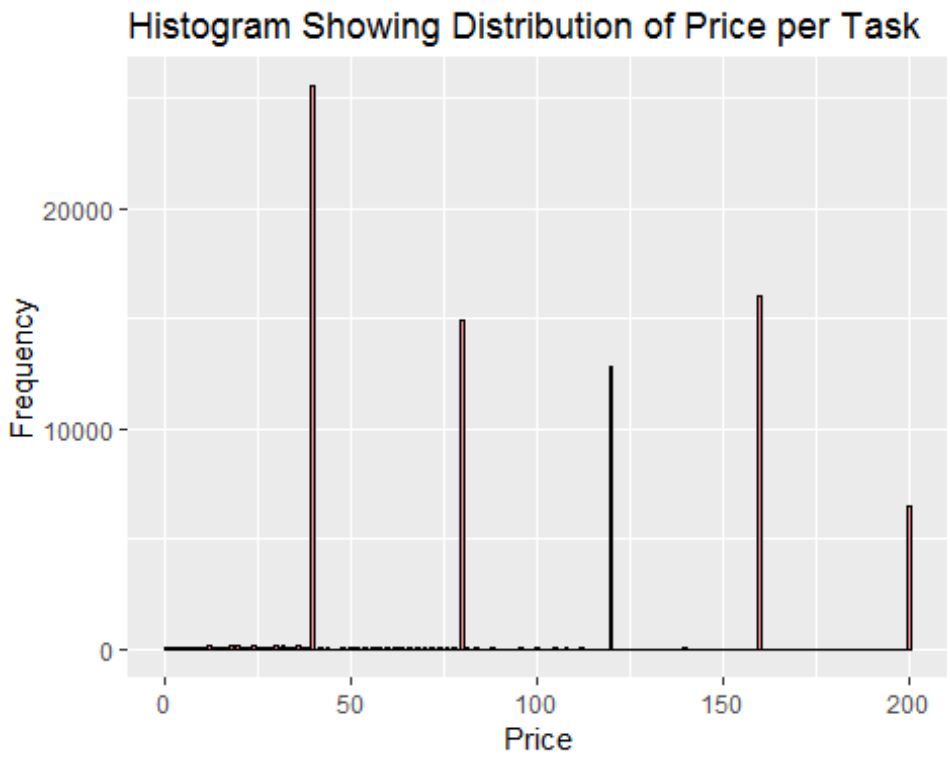
```
## Calculate task quality
```

```
qc_4 <- mutate(qc_3, quality_score = case_when(

        interval == "int1" ~ 25,
        interval == "int2" ~ 50,
        interval == "int3" ~ 75,
        interval == "int4" ~ 100

)
 )


##Calculate ticket quality (average of task Q)
calculated_quality <-qc_4  %>% group_by(ticket_id) %>%
mutate(quality_ticket = mean(quality_score))

calculated_quality <- ungroup(calculated_quality)
```

Analyse price distribution

Price per Task

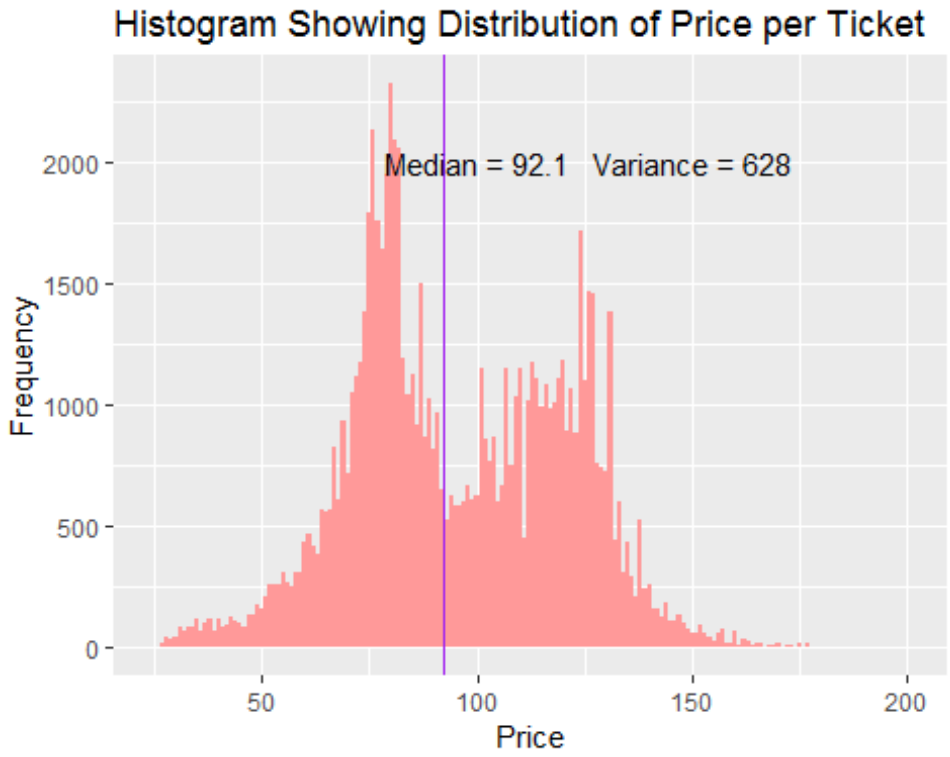
```
g1 <- ggplot(calculated_quality, aes(price))
g1 <- g1 + geom_histogram(binwidth = 1,fill="#FF9999", colour="black")
g1 <- g1 + labs(title = "Histogram Showing Distribution of Price per
Task",
                x = "Price", y = "Frequency")

g1
```



As expected from the equation, the vast majority of tasks are multiples of 40 (most tasks are of that length) corresponding to the various skill levels, with a small number of other prices from the few non-40 word tasks.

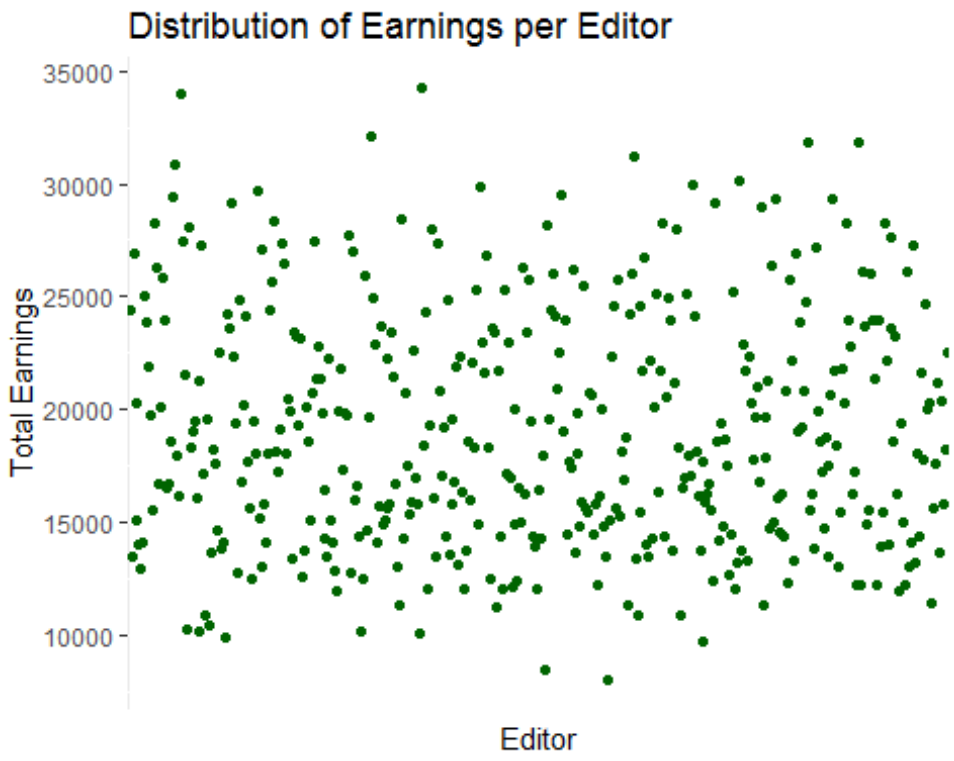
Price per Ticket

```
g2 <- ggplot(group_by(calculated_quality, ticket_id),
aes(price_ticket))
g2 <- g2 + geom_histogram(binwidth = 1,fill="#FF9999")
g2 <- g2 + labs(title = "Histogram Showing Distribution of Price per
Ticket",
                x = "Price", y = "Frequency")
g2median <- median(calculated_quality$price_ticket) ## 92.1
g2variance <- var(calculated_quality$price_ticket)  ## 628
g2 <- g2 + geom_vline(aes(xintercept = g2median),col='purple',size=0.5)
+
                      ## geom_text(aes(label=round(g2median,1),y=0,x=
g2median),
                      ## vjust=-1,col='black',size=5)
                      annotate("text", x = 100, y = 2000, label =
"Median = 92.1") +
                      annotate("text", x = 150, y = 2000, label =
"Variance = 628")
g2
```



Distribution of Earnings per Editor

```
##Calculate editor earnings
calculated_earnings <-calculated_quality  %>% group_by(editor_id) %>%
mutate(earnings_total = sum(price))

g3 <- ggplot(calculated_earnings, aes(editor_id, earnings_total))
g3 <- g3 + geom_point(fill="#FF9999", color =  "dark green")
g3 <- g3 + labs(title = "Distribution of Earnings per Editor",
                x = "Editor", y = "Total Earnings") +
      theme(axis.text.x=element_blank(), axis.ticks.x=element_blank() )

g3
```

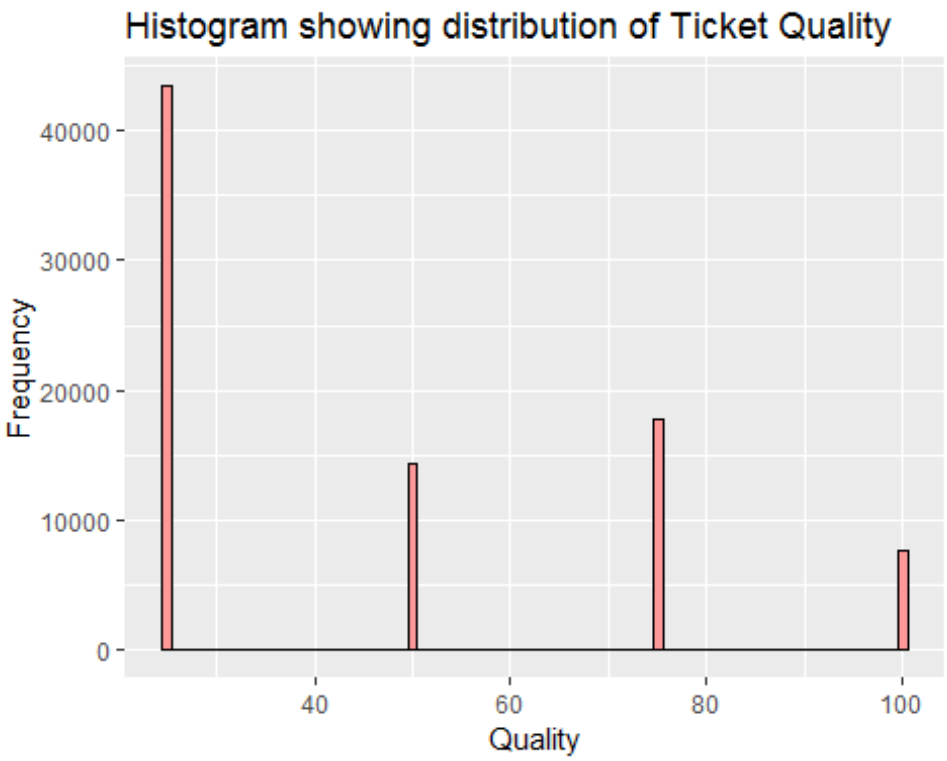


There is no clear pattern here. Given the uniform random assignment of tasks to editors, this makes sense.

Analyzing the quality

Quality per Task

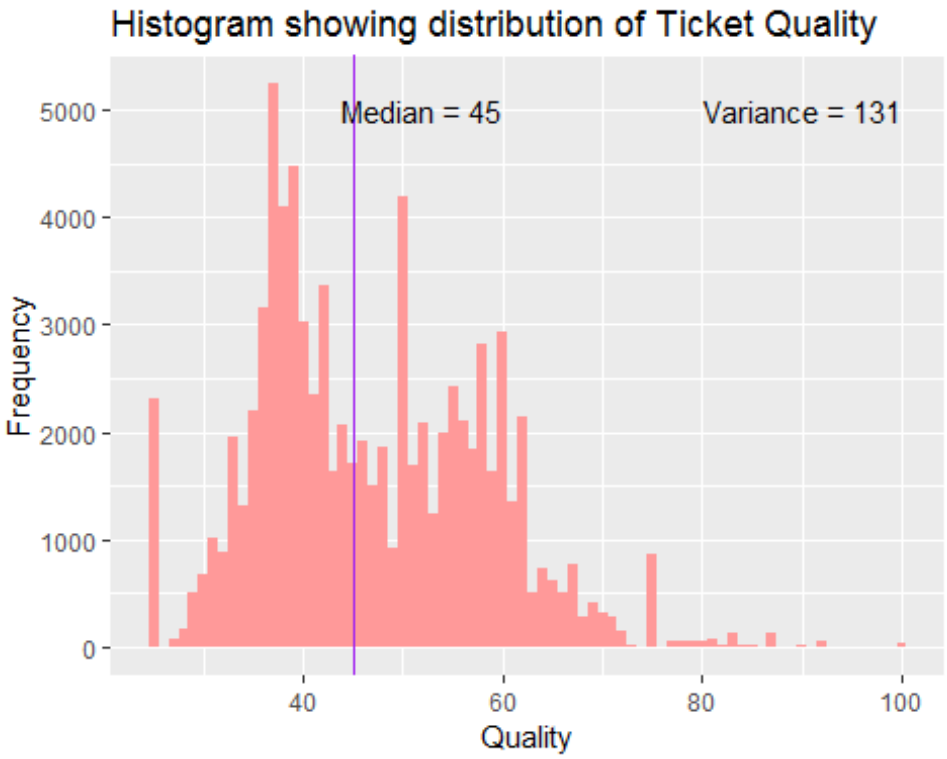
```
g4 <- ggplot(calculated_quality, aes(quality_score)) +
             geom_histogram(binwidth = 1,fill="#FF9999",
colour="black") +
             labs(title = "Histogram showing distribution of Ticket
Quality ",

                  x = "Quality", y = "Frequency")

g4
```



Distribution of task quality makes sense given the equations, with many more at Quality = 25 because of the two skill levels in that interval.

Quality per Ticket

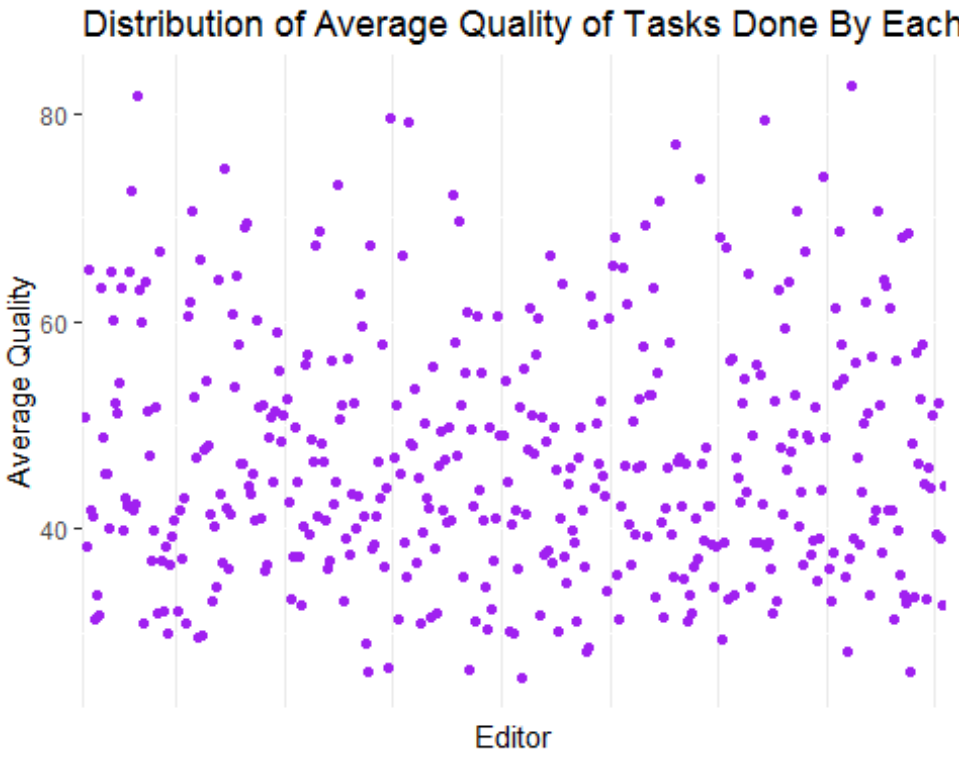
```
g5median <- median(calculated_quality$quality_ticket) ## 45
g5variance <- var(calculated_quality$quality_ticket)  ## 131.806
g5 <- ggplot(calculated_quality, aes(quality_ticket)) +
             geom_histogram(binwidth = 1,fill="#FF9999") +
             labs(title = "Histogram showing distribution of Ticket
Quality",
                  x = "Quality", y = "Frequency") +
             geom_vline(aes(xintercept =
g5median),col='purple',size=0.5) +
             ## geom_text(aes(label=round(g5median,1),y=0,x= g5median),
             ##           vjust=-1,col='black',size=5) +
             annotate("text", x = 52, y = 5000, label = "Median = 45")
+
             annotate("text", x = 90, y = 5000, label = "Variance =
131")

g5
```



Quality distribution is skewed towards the low end. Distribtion has a similar shape to that of the ticket price distribution. This makes sense considering that the editor skill level is used to calcuate both Quality and Price.

Average Quality per Editor

```
##Calculate editor earnings
calculated_avg_quality <-calculated_quality  %>% group_by(editor_id) %>
% mutate(avg_editor_q = mean(quality_score))


g6 <- ggplot(calculated_avg_quality, aes(editor_id, avg_editor_q))
g6 <- g6 + geom_point(fill="#FF9999", color =  "purple")
g6 <- g6 + labs(title = "Distribution of Average Quality of Tasks Done
By Each Editor",
                x = "Editor", y = "Average Quality") +
      theme(axis.text.x=element_blank(), axis.ticks.x=element_blank() )

g6
```



Once again there is no clear pattern here. Given the uniform random assignment of tasks to editors, this makes sense.

Solving the problems

It is clear that both problems (inconsistent quality for the clients and insufficient tasks to create enough income for the editors) have a common root: how the tasks are assigned. Uniform random assignment does not take the editors' skills aren't taken into account. This means that can editors are assigned to tasks with domains where they have a relatively low skill when they could be a better fit for other domains. The editor skill level factors directly into how price and quality are calculated.

I will attempt to improve the quality and price distribution by changing how the tasks are assigned.

Rethinking task assignment.

I will rewrite the editor to task assignment such that it takes editor skill level in the relevant domain into account, without unbalancing the number of tasks per editor too much. To do this, I first identify each editor's weakest domain, (the one in which they have the lowest skill, ties broken randomly), then I will divide tasks by domain and assign to them only the editors who do not have that domain as their weakest. I have chosen this because which skill is the weakest is much more evenly distributed than the way the strongest skills are.

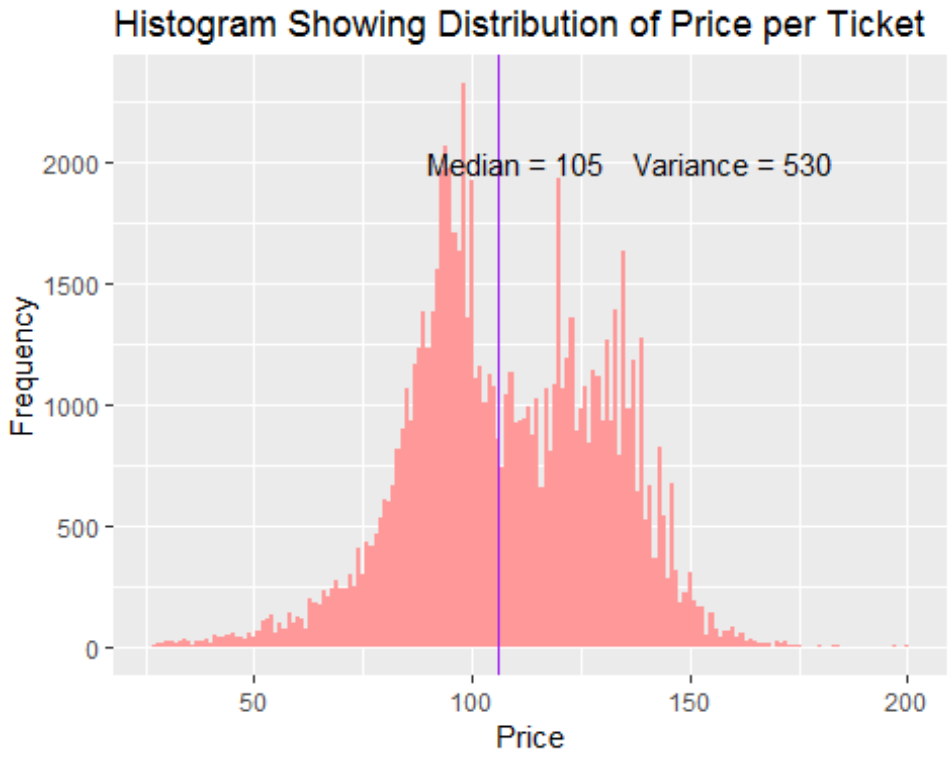
```r
## detect worst skill
editors_min <- editors
skills_levels <- select(editors, travel:health_care)


## Invert skill levels so that max.col finds lowest
r_invert <- function(x) 1/x

skills_levels <- sapply(skills_levels, r_invert)


editors_min$min_domain <- colnames(skills_levels)
[max.col(skills_levels, ties.method = "random")]




## Sample each min type separetely, then attach
## so sample for travel tasks, then fintech etc.
## this avoids nesting nightmare

## get travel rows in tasks
travel_rows <- filter(tasks_tickets_clients, domain == "travel")
## get non-mininum travel rows in editors
non_min_travel <- filter(editors_min,  min_domain != "travel")
## assign
travel_rows_assigned <- travel_rows
travel_rows_assigned$editor_id <- sample(non_min_travel$editor_id,
                                size = nrow(travel_rows),
                                replace = TRUE)

## repeat with other domains

## health_care
health_care_rows <- filter(tasks_tickets_clients, domain ==
"health_care")
non_min_health_care <- filter(editors_min,  min_domain !=
"health_care")
health_care_rows_assigned <- health_care_rows
health_care_rows_assigned$editor_id <-
sample(non_min_health_care$editor_id,
                                size = nrow(health_care_rows),
                                replace = TRUE)

## fintech
fintech_rows <- filter(tasks_tickets_clients, domain == "fintech")
non_min_fintech <- filter(editors_min,  min_domain != "fintech")
```

```r
fintech_rows_assigned <- fintech_rows
fintech_rows_assigned$editor_id <- sample(non_min_fintech$editor_id,
                                size = nrow(fintech_rows),
                                replace = TRUE)

## ecommerce
ecommerce_rows <- filter(tasks_tickets_clients, domain == "ecommerce")
non_min_ecommerce <- filter(editors_min,  min_domain != "ecommerce")
ecommerce_rows_assigned <- ecommerce_rows
ecommerce_rows_assigned$editor_id <-
sample(non_min_ecommerce$editor_id,
                                size = nrow(ecommerce_rows),
                                replace = TRUE)

## sports
sports_rows <- filter(tasks_tickets_clients, domain == "sports")
non_min_sports <- filter(editors_min,  min_domain != "sports")
sports_rows_assigned <- sports_rows
sports_rows_assigned$editor_id <- sample(non_min_sports$editor_id,
                                size = nrow(sports_rows),
                                replace = TRUE)

## gamming
gamming_rows <- filter(tasks_tickets_clients, domain == "gamming")
non_min_gamming <- filter(editors_min,  min_domain != "gamming")
gamming_rows_assigned <- gamming_rows
gamming_rows_assigned$editor_id <- sample(non_min_gamming$editor_id,
                                size = nrow(gamming_rows),
                                replace = TRUE)



## attach them
assigned_editors_2 <- bind_rows(travel_rows_assigned,
                                health_care_rows_assigned,
                                fintech_rows_assigned,
                                ecommerce_rows_assigned,
                                sports_rows_assigned,
                                gamming_rows_assigned)









## Join editor table onto assigned table
assigned_all_2 <- left_join(assigned_editors_2, editors, by =
"editor_id")
```

```r
##
## Repeating all the previous steps with the new assignments
##


##renaming number_words variables to clarify one for tasks and other
for tickets
names(assigned_all_2)[2] <- "number_words_task"
names(assigned_all_2)[7] <- "number_words_ticket"


## Calculating Price per task, assuming all language pairs match
calculated_price_task_2 <- mutate(assigned_all_2,
                                  price = case_when
                                   (domain == "travel"
                                    ~ constant_alpha * number_words_task
* travel,
                                    domain == "fintech"
                                    ~ constant_alpha * number_words_task
* fintech,
                                    domain == "health_care"
                                    ~ constant_alpha * number_words_task
* health_care,
                                    domain == "ecommerce"
                                    ~ constant_alpha * number_words_task
* ecommerce,
                                    domain == "sports"
                                    ~ constant_alpha * number_words_task
* sports,
                                    domain == "gamming"
                                    ~ constant_alpha * number_words_task
* gamming
                                   )
                                  )


calculated_price_ticket_2 <-calculated_price_task_2  %>%
group_by(ticket_id) %>% mutate(price_ticket = mean(price))

calculated_price_ticket_2 <- ungroup(calculated_price_ticket_2)


## For these steps use intermediate variables qc_1_2, _2_2 etc.
(quality calculation)

## Generate variable with four randomly assigned levels
## One 97% likely to be generated, the others 1%
qc_1_2 <- mutate(calculated_price_ticket_2,
                 prob_levels = sample(c("a", "b", "c", "d"),
```

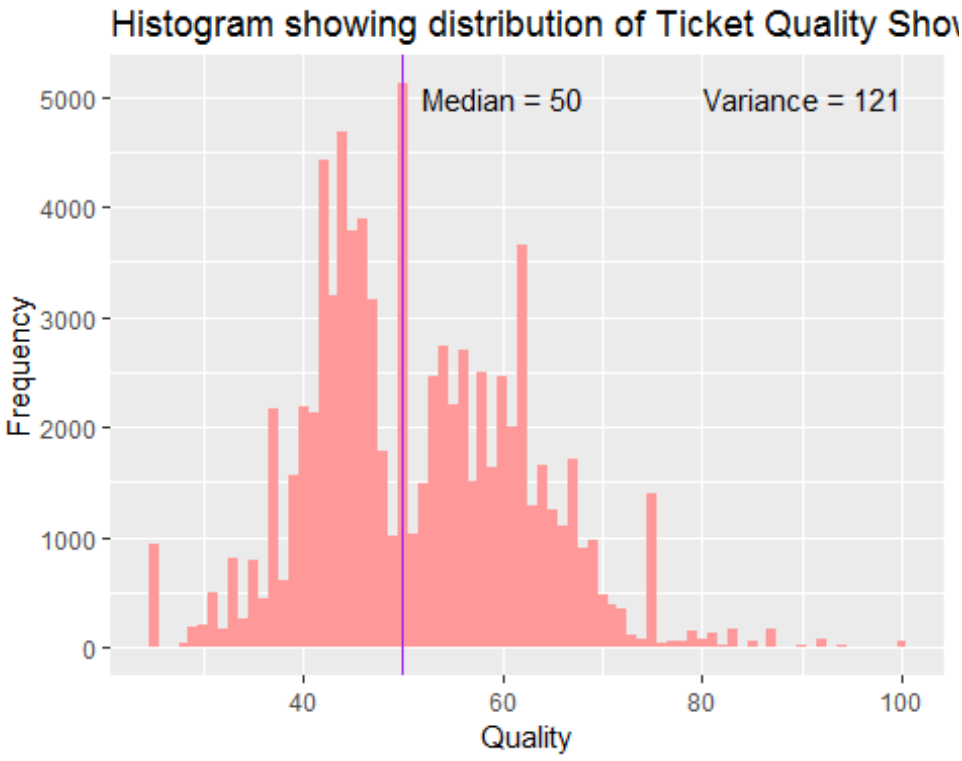
```
                                    size =
nrow(calculated_price_ticket_2),
                                    prob = c(0.97, 0.01, 0.01, 0.01),
                                     replace = TRUE) )

## identify domain skill used for task, create skill used column to
simplify
## following steps by avoiding checking domain each time
qc_2_2 <- mutate(qc_1_2, skill_used = case_when(
                    domain == "travel"      ~ travel,
                    domain == "fintech"     ~ fintech,
                    domain == "health_care" ~  health_care,
                    domain == "ecommerce"   ~ ecommerce,
                    domain == "sports"      ~ sports,
                    domain == "gamming"     ~ gamming
                  )
                )


## Use conditionals (||) and random level assignment (a,b etc) in here
qc_3_2 <- mutate(qc_2_2 , interval = case_when(
  skill_used == 1 & prob_levels == "a" ~ "int1",
  skill_used == 1 & prob_levels == "b" ~ "int2",
  skill_used == 1 & prob_levels == "c" ~ "int3",
  skill_used == 1 & prob_levels == "d" ~ "int4",

  skill_used == 2 & prob_levels == "a" ~ "int1",
  skill_used == 2 & prob_levels == "b" ~ "int2",
  skill_used == 2 & prob_levels == "c" ~ "int3",
  skill_used == 2 & prob_levels == "d" ~ "int4",

  skill_used == 3 & prob_levels == "b" ~ "int1",
  skill_used == 3 & prob_levels == "a" ~ "int2",
  skill_used == 3 & prob_levels == "c" ~ "int3",
  skill_used == 3 & prob_levels == "d" ~ "int4",

  skill_used == 4 & prob_levels == "b" ~ "int1",
  skill_used == 4 & prob_levels == "c" ~ "int2",
  skill_used == 4 & prob_levels == "a" ~ "int3",
  skill_used == 4 & prob_levels == "d" ~ "int4",

  skill_used == 5 & prob_levels == "b" ~ "int1",
  skill_used == 5 & prob_levels == "c" ~ "int2",
  skill_used == 5 & prob_levels == "d" ~ "int3",
  skill_used == 5 & prob_levels == "a" ~ "int4"



  )
```

```
)

## Calculate task quality

qc_4_2 <- mutate(qc_3_2, quality_score = case_when(

        interval == "int1" ~ 25,
        interval == "int2" ~ 50,
        interval == "int3" ~ 75,
        interval == "int4" ~ 100

)
 )


##Calculate ticket quality (average of task Q)
calculated_quality_2 <- qc_4_2  %>%
                        group_by(ticket_id) %>%
                        mutate(quality_ticket = mean(quality_score))

calculated_quality_2 <- ungroup(calculated_quality_2)

g2_2median <- median(calculated_quality_2$price_ticket) ## 105
g2_2variance <- var(calculated_quality_2$price_ticket)  ## 530
g2_2 <- ggplot(group_by(calculated_quality_2, ticket_id),
aes(price_ticket))
g2_2 <- g2_2 + geom_histogram(binwidth = 1,fill="#FF9999")
g2_2 <- g2_2 + labs(title = "Histogram Showing Distribution of Price
per Ticket",
                x = "Price", y = "Frequency") +
        geom_vline(aes(xintercept = g2_2median),col='purple',size=0.5)
+
        ## geom_text(aes(label=round(g2_2median,1),y=0,x= g2_2median),
        ##              vjust=-1,col='black',size=5)
        annotate("text", x = 110, y = 2000, label = "Median = 105") +
        annotate("text", x = 160, y = 2000, label = "Variance = 530")

g2_2
```



This new price per ticket distribution has a 14% higher median and a 16% lower variance. A significant improvement!

```
g5_2median <- median(calculated_quality_2$quality_ticket) ## 50
g5_2variance <- var(calculated_quality_2$quality_ticket)  ## 121
g5_2 <- ggplot(calculated_quality_2, aes(quality_ticket)) +
             geom_histogram(binwidth = 1,fill="#FF9999") +
             labs(title = "Histogram showing distribution of Ticket
Quality Showing Median",
                  x = "Quality", y = "Frequency") +
             geom_vline(aes(xintercept =
g5_2median),col='purple',size=0.5) +
             ## geom_text(aes(label=round(g5_2median,1),y=0,x=
g5_2median),
             ##           vjust=-1,col='black',size=5)
            annotate("text", x = 60, y = 5000, label = "Median = 50") +
            annotate("text", x = 90, y = 5000, label = "Variance =
121")

g5_2
```



This new quality per ticket distribution has a 11% higher median and a 8% lower variance. Also a signficant improvement!

Ideas For Possible Next Steps

If increasing the medians and decreasing the variances further is desired, reassigning the editors to task but not choosing editors whose skill in the relevant domain is one of their two worst is an idea. Another idea would be to obtain the language pair data for the editors see how the overall distributions change considering the different equations for non-matching language pairs.