Valuation Survey Analysis

Data Cleaning

First we'll load up the raw survey data from Google Forms.

Raw Data

Timestamp	How.much.would.we.need.to.pay.you.to.arrive.20.minutes.late… In.USD.	X40.minutes.late.	X1.hour.late.	X1.hour.and.20.minutes.late.
7/7/2016 14:52:12	$50.00	$200.00	$200.00	$200.00
7/7/2016 16:51:01	$5.00	$20.00	$20.00	$20.00
7/8/2016 11:17:50	$20.00	$40.00	$60.00	I wouldn't accept.
7/11/2016 13:26:12	$30.00	$50.00	$50.00	$50.00
7/11/2016 15:06:24	$20.00	$50.00	$80.00	$120.00

Next, lets's use more intuitive column names, add an id number so we can keep track of which responses come from the same individuals, and split the data into separate dataframes for each scenario.

Brunch with Friends

brunch.20	brunch.40	brunch.60	brunch.80	brunch.100	brunch.120	id
$50.00	$200.00	$200.00	$200.00	$200.00	$100.00	1
$5.00	$20.00	$20.00	$20.00	$30.00	$30.00	2
$20.00	$40.00	$60.00	I wouldn't accept.	I wouldn't accept.	$200.00	3

Waiting on a Sunny Day

sunny.20	sunny.40	sunny.60	sunny.80	sunny.100	sunny.120	id
$10.00	$40.00	$100.00	$200.00	$250.00	$300.00	1
$10.00	$20.00	$30.00	$50.00	$75.00	$100.00	2
$15.00	$25.00	$40.00	$50.00	$75.00	$100.00	3

Waiting in the Rain

rainy.20	rainy.40	rainy.60	rainy.80	rainy.100	rainy.120	id
$30.00	$100.00	$300.00	$500.00	$1,000.00	$2,000.00	1
$15.00	$30.00	$50.00	$75.00	$100.00	$150.00	2
$5.00	$20.00	$30.00	$50.00	$75.00	$100.00	3

Now let's convert the the string responses to the correct datatypes, and restructure the dataframes so that each row contains three columns: the responder's id, the number of minutes waited, and the responder's valuation.

Brunch with Friends

id	minutes	cost

id	minutes	minutes 20	maxcost 50

id	minutes	minutes 40	cost 200

id	minutes	cost
1	60	200

Waiting on a Sunny Day

id	minutes	cost
1	20	10
1	40	40
1	60	100

Waiting in the Rain

id	minutes	cost
1	20	30
1	40	100
1	60	300

Beverage Valuations

id	cost
1	5
2	20
3	10

Finally, let's normalize the data by respondant.

```
# Find the minimum and maximum responses across all columns
# for each respondant.
id_max <- ungroup(bind_rows(brunch, sunny, rainy, beverage) %>%
          group_by(id) %>%
            do(data.frame(max_val = max(.$cost))))

normalize <- function(df){
  ungroup(df %>%
          left_join(id_max, by = c('id' = 'id')) %>%
            group_by(id) %>%
              mutate(cost = cost / max_val)
          )
}

brunch <- normalize(brunch)
sunny <- normalize(sunny)
rainy <- normalize(rainy)
beverage <- normalize(beverage)
```

```
## Warning in left_join_impl(x, y, by$x, by$y, suffix$x, suffix$y): joining
## factors with different levels, coercing to character vector
```

Brunch with Friends

id	minutes	cost	max_val
1	20	0.1	500

id	minutes	cost	cost	max_val
1	60	0.4		500

Waiting on a Sunny Day

id	minutes	cost	max_val
1	20	0.02	500
1	40	0.08	500
1	60	0.20	500

Waiting in the Rain

id	minutes	cost	max_val
1	20	0.06	500
1	40	0.20	500
1	60	0.60	500

Beverage Valuations

id	cost	max_val
1	0.0100000	500
2	0.1333333	150
3	0.0500000	200

Models

Now let's create a linear model for each scenario.
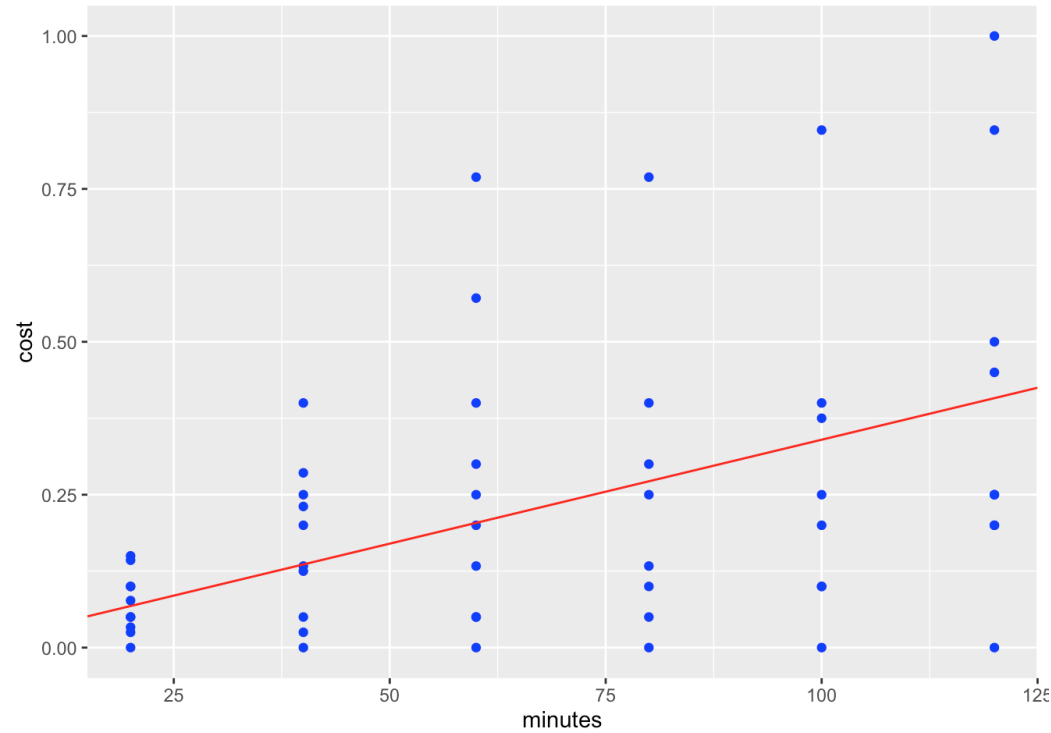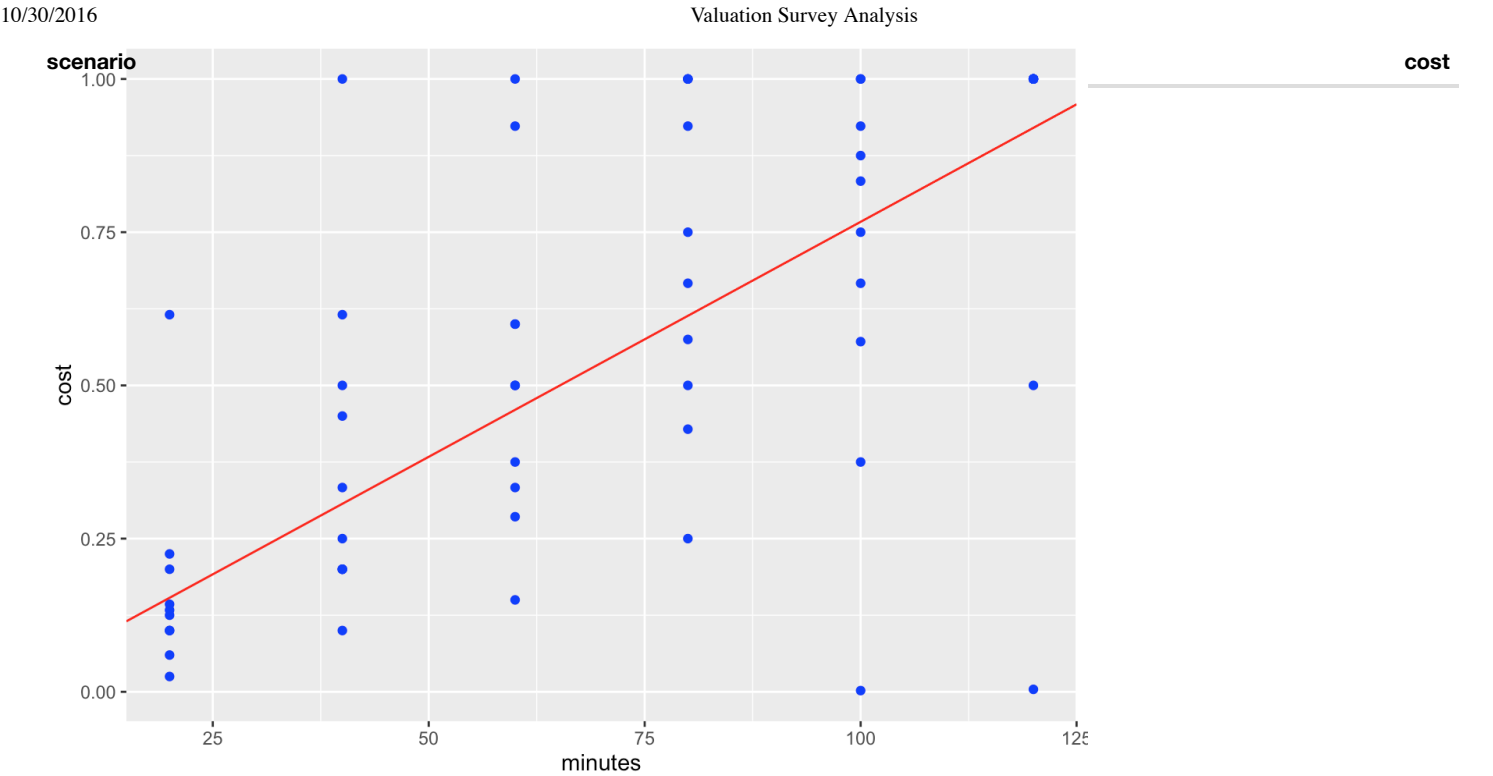
Brunch

```
##
## Call:
## lm(formula = cost ~ 0 + minutes, data = brunch)
##
## Coefficients:
## minutes
##   0.7601
```

Sunny
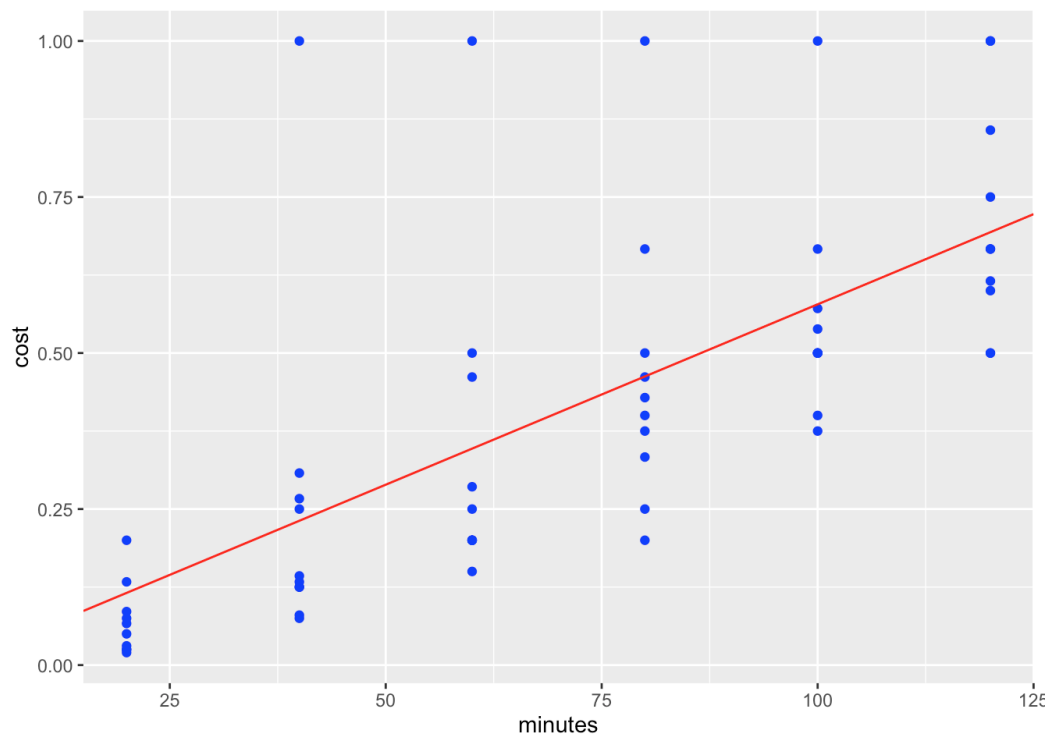

```
##
## Call:
## lm(formula = cost ~ 0 + minutes, data = sunny)
##
## Coefficients:
## minutes
##    1.362
```

Rainy

```
##
## Call:
## lm(formula = cost ~ 0 + minutes, data = rainy)
##
## Coefficients:
## minutes
##   1.685
```

Beverage

For the beverage question we just have a single data-point for each respondant ("How much would we need to pay you to forgo your morning beverage"). So we will take the mean as our coefficient.

```
kable(beverage[1:3,])
```

```
avg_beverage_value <- mean(beverage$cost)
avg_beverage_value
```

```
## [1] 9.7
```

Results

Let's see how much (on average) people value thier time in each scenario.

(For Brunch, Sunny, and Rainy: the cost is the value of one minute spent at the bus stop. For Beverage the cost is the valuation of one beverage.)

scenario	cost
brunch	0.760101
sunny	1.361648
rainy	1.684725
beverage	9.700000

We want to use integers in the experiment (we aren't dealing in fractions of coins), so let's convert these coefficients to whole numbers.

```
results$cost <- as.integer(round(results$cost * 1000))
kable(results)
```

scenario	wait_pentaly	delay_reward	destination_reward

scenario	cost
brunch	760
sunny	1362
rainy	1685
beverage	9700

It looks like we could have a problem here… The valuation of time at brunch seems lower than the valuation of time waited at the bus stop (which doesn't seem correct). Also, we don't have values for different "destinations".

Mapping onto Experiment Scenarios

Now let's take the values we've obained and figure out what the values for each scenario should be.

- **wait_penalty** = penalty per minute at the bus stop
- **delay_reward** = reward per minute spent BEFORE going to the bus stop
- **destination_reward** = reward per minute spent at the destination

```
experiment_brunch <- data.frame(
  wait_pentaly = results$cost[2], # We'll use the sunny value rather than rainy
  delay_reward = results$cost[2], # since we don't specify the weather in the
                                  # scenario.
  destination_reward = results$cost[1]
)

experiment_sunny <- data.frame(
  wait_pentaly = results$cost[2], # Sunny Value
  delay_reward = results$cost[2], # Sunny Value
  destination_reward = results$cost[1]
)

experiment_rainy <- data.frame(
  wait_pentaly = results$cost[3], # Rainy value
  delay_reward = results$cost[3], # Rainy value
  destination_reward = results$cost[1]
)

experiment_beverage <- data.frame(
  wait_pentaly = results$cost[3], # Rainy value
  delay_reward = results$cost[3], # Rainy value
  destination_reward = results$cost[1],
  beverage_bonus = results$cost[4] # Beverage value.
)
```

Brunch

wait_pentaly	delay_reward	destination_reward
136	136	76

Sunny

wait_pentaly	delay_reward	destination_reward
136	136	76

Rainy

wait_pentaly	delay_reward	destination_reward

Beverage

wait_pentaly	delay_reward	destination_reward	beverage_bonus
168	168	76	970