Pilot 2

Alex Kale

5/14/2019

Recap of Pilot 1

In pilot 1 we asked participants to view distributions of past scores for two teams competing in a game and judge the probability that Team A would score higher than Team B in a future game (a.k.a., probability of superiority or common language effect size). We then asked them to bet between 1 and 1000 coins that team A would win. We structured the payoff so that the utility-optimal bet was a linear function of the probability that Team A would win. Each participant comlpeted 20 trials at different combinations of effect size and standard deviation of scores using one of three visualizations: means and intervals, means only, or HOPs.

We found that user judgments of probability of superiority were substantially more biased toward 50% in the conditions that emphasized the mean. However, the results of the betting task were less conclusive. Bets seemed to cluster around 1, 500, and 1000 coins suggesting that participants did not use the full granularity of the betting scale. We tried categorizing bets as low, medium, and high but still were unable to make much of the results. One possible reason for this is that the betting data were very noisy, perhaps becasue the payoff scheme of the task required participants to trade off a tiered tax on winnings from their bet vs a constant tax on the amount they did not bet. Even though there are real world financial decisions that resemble these incentives, it is highly unlikely that users were able to form an intuition about what their decision rule should be.

Pilot 2

In pilot two, we keep the probability of superiority judgment that worked well in pilot 1, but we change the incentivized decision task.

Inspiration for New Betting Task

We draw inspiration from Joslyn's road salting paradigm where participants must decide whether to spend money to salt the roads based on weather forecasts about the probability of freezing temperatures. The key aspect of this study that we find compelling is that participants are deciding whether to make an intervention by trading off the cost of intervention with the cost and probability of potential damages. However, this task is slightly simplified relative to real world situations where people make decisions based on effect size information. First, in Josyln's task applying salt to the road guarantees the prevention of damage entirely, whereas in most situations intervention will only reduce either the probability or the severity of potential damages. Second, in Joslyn's task participants should always intervene if probabilities of freezing temperatures exceed 17%, whereas in real world situations stakes are dynamic and decisions may be subject to non-linear distortions in the perception of probability. Third, Joslyn's paradigm only examines incentivized decisions which are framed as potential losses, whereas real world siturations are sometimes framed as potential gains. Also, the use of the familiar context of weather introduces biases which may not generalize such as the tendency to mistake confidence intervals on forecasts for daily low and high temperatures. In our new task, we aim for a similarly intuitive incentivized decision about interventions, but we also modify the scenario and payoff scheme to make it more realistic and representative of the types of situations in which people make decisions based on effect size information.

New Betting Task: Scenario and Payoff Scheme

In our new task, users play the role of an owner of a mid-sized manufacturing business that makes widgets. They are presented with scenarios in which they must make a decision about whether to continue to user their existing machinery or to pay for a new machine based on the probability of gaining or losing a contract.

In the gain framing trails, users are told they will *pick up a new contract* worth $X if they produce more than a certain *number of widgets* next year, but they can improve their chances of getting that contract if they pay $C for a new machine. The optimal decision rule in this case maximizes expected gains.

$$X * p(contract| \sim machine) < X * p(contract|machine) - C$$

If we assume a constant ratio between the value of the contract and the cost of intervention $K = \frac{X}{C}$, this can be expressed as in terms of the difference in probability of gaining the contract with and without a new machine.

$$p(contract| \sim machine) + \frac{1}{K} < p(contract|machine)$$

The decision rule can also be expressed in terms of the risk ratio of getting the contract with vs without intervention.

$$1 + \frac{1}{K * p(contract| \sim machine)} < \frac{p(contract|machine)}{p(contract| \sim machine)}$$

The loss framing trails are similarly set up. Users are told they will *lose an existing contract* worth $X if they produce more than a certain *number of defective widgets* next year, but they can improve their chances of keeping that contract if they pay $C for a new machine. The optimal decision rule in this case minimizes expected losses.

$$X * p(\sim contract| \sim machine) > X * p(\sim contract|machine) + C$$

Again, if we assume a constant ratio between the value of the contract and the cost of intervention $K = \frac{X}{C}$, this can be expressed as in terms of the difference in probability of losing the contract with and without a new machine.

$$p(\sim contract| \sim machine) - \frac{1}{K} > p(\sim contract|machine)$$

The decision rule can also be expressed in terms of the risk ratio of losing the contract with vs without intervention.

$$1 - \frac{1}{K * p(\sim contract| \sim machine)} > \frac{p(\sim contract|machine)}{p(\sim contract| \sim machine)}$$

Each trial, users will receive feedback based on their decision and a simulated outcome regarding the contract in question. We will tally the dollar value of contracts gained/kept minus the cost of interventions across trials, and users will receive a proportional amount as a bonus through MTurk.

Visualization Conditions

Users will be shown distributions of the number of widgets or defective widgets produced in past years by both their current equipent and the new machine they might want to buy. Number of widgets will be visualized as *means only, means with intervals, HOPs with means, densities, quantile dotplots, intervals only, and HOPs*. These conditions span a continuum of how much the make the mean available to users, from emphasizing the mean to only encoding it implicitly.

We might also manipulate how data is processed before it is visualized to test different ways of presenting effect size. In addition to showing distributions of numbers of widgets and defective widgets for each machine, we could also show single distributions of the difference between the number of widgets produced by the two machines or even a the ratio of the number of widgets produced by the two machines. In the single distribution conditions we would use supplementary text to describe the average widget production with the current machine as a baseline. These manipulations would test whether showing a single derived measure is actually more helpful to users than just showing two distributions. We will need to think about situations in which this comparison should be considered representative. If we do this it would be the focus of the second experiment in the study.

Data Conditions

We manipulate the probability of the new machine producing more widgets than the old machine (p_superiority), sampling at linear intervals in logodds units. When p_superiority is greater than 0.5, the decision task is framed as a gain scenario where the user needs to manufacture at least 500 million widgets next year to get a new contract. When p_superiority is less than 0.5, the decision task is framed as a loss scenario where the user needs to manufacture no more than 75 defective widgets per million next year to keep an existing contract.

```r
# linear sampling of log odds for ground truth probability of superiority for the new machine
n_trials <- 24
logodds <- seq(log(0.025/(1-0.025)), log(0.975/(1-0.975)), length.out = n_trials) # 8
p_superiority <- 1 / (1 + exp(-logodds))

print(p_superiority)
```

```
##  [1] 0.02500000 0.03405957 0.04624645 0.06251174 0.08399386 0.11197654
##  [7] 0.14777771 0.19254342 0.24694046 0.31079046 0.38275856 0.46026266
## [13] 0.53973734 0.61724144 0.68920954 0.75305954 0.80745658 0.85222229
## [19] 0.88802346 0.91600614 0.93748826 0.95375355 0.96594043 0.97500000
```

We also manipulate the baseline probability of gaining/keeping the contract with the old machine. We sample three levels of this baseline probability: 0.5 where the old machine is as likely as a coin flip to result in the contract, 0.15 where the old machine is fairly unlikely to result in the contract, and 0.85 where the old machine has a good chance of resulting in the contract.

```r
# baseline probability of gaining/keeping a contract with the old machine
baseline <- c(.15, .5, .85) # c(.15, .5)

# initialize data conditions dataframe
conds_df <- data.frame(
  "p_superiority" = rep(p_superiority, length(baseline)),
  "baseline" = sort(rep(baseline, length(p_superiority))))

head(conds_df)
```

```
##   p_superiority baseline
## 1    0.02500000     0.15
## 2    0.03405957     0.15
## 3    0.04624645     0.15
## 4    0.06251174     0.15
## 5    0.08399386     0.15
## 6    0.11197654     0.15
```

As stated above, we set the threshold for gaining the new contract 500 million widgets and the threshold for keeping the old contract at 75 defective widgets per million.

```
# label gain vs loss framing trials based on p_superiority and add contract thresholds
conds_df <- conds_df %>%
  mutate(frame = if_else(p_superiority > .5, "gain", "loss"),
         threshold = if_else(frame=="gain",
                             500, # million widgets required to gain contract
                             75)) # defective widgets per million required to keep contract

head(conds_df)
```

```
##   p_superiority baseline frame threshold
## 1    0.02500000     0.15  loss        75
## 2    0.03405957     0.15  loss        75
## 3    0.04624645     0.15  loss        75
## 4    0.06251174     0.15  loss        75
## 5    0.08399386     0.15  loss        75
## 6    0.11197654     0.15  loss        75
```

We control the standard deviation of the distribution of the difference in widgets between the two machines (sd_diff) by setting it to 15. In the gain framing this is 15 million widgets. In the loss framing, this is 15 defective widgets per million. Since the value of sd_diff is relative to the threshold for gaining/keeping the contract, we can think of this variable as constant across trials. We then derive the mean difference in the number of widgets produced by the new minus the old machine (mean_diff) from sd_diff and p_superiority.

```
# add columns for the mean and standard deviation of the difference in the number of widgets produced by the new
 vs old machine
# depending on the gain vs loss frame, these values represent millions of widgets vs defective widgets per millio
n
conds_df <- conds_df %>%
  mutate(sd_diff = 15, # std(new - old)
         mean_diff = sd_diff * qnorm(p_superiority)) # mean(new - old)

head(conds_df)
```

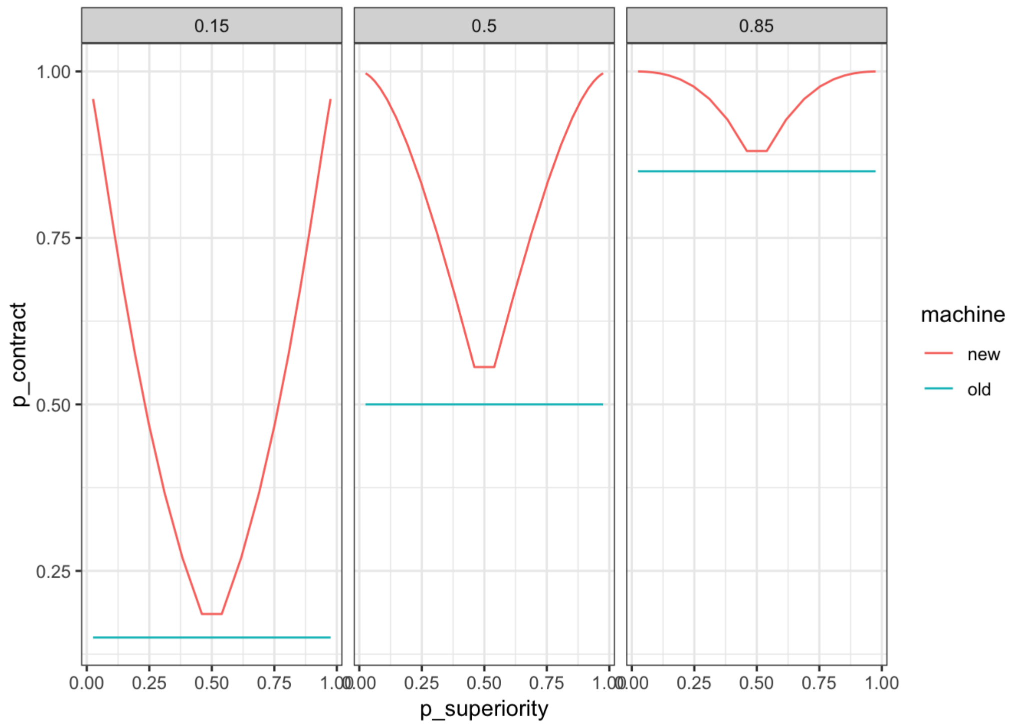
```
##   p_superiority baseline frame threshold sd_diff mean_diff
## 1    0.02500000     0.15  loss        75      15 -29.39946
## 2    0.03405957     0.15  loss        75      15 -27.36327
## 3    0.04624645     0.15  loss        75      15 -25.23588
## 4    0.06251174     0.15  loss        75      15 -23.01038
## 5    0.08399386     0.15  loss        75      15 -20.68048
## 6    0.11197654     0.15  loss        75      15 -18.24125
```

We derive the standard deviation of the number of widgets produced by the machines from year to year (sd) from sd_diff, variance sum law, and the assumption that the machines have equal and independent variances. We derive the mean number of widgets produced by each machine (mean) from the threshold for gaining/keeping the contract, the sd of widgets for each machine, and the mean_diff between the number of widgets for the new minus the old machine. We derive the probability of gaining/keeping the contract from the threshold, mean, and sd.

```
# double the length of the dataframe to add information per machine, creating a stimulus dataframe with a row per
distribution to visualize
stim_df <- map_df(seq_len(2), ~conds_df)
stim_df$machine <- sort(rep(c("new", "old"), length(stim_df$p_superiority)/2))

# add columns for the mean and standard deviation of widgets for each machine and the probability of gaining/keep
ing the contract
stim_df <- stim_df %>%
  mutate(sd = sqrt(stim_df$sd_diff ^ 2 / 2), # assume equal and independent variances in the number of widgets pr
oduced by each machine
        mean = if_else(machine=="old",
                       if_else(frame=="gain", # old machine is at baseline
                               threshold - sd * qnorm(1 - baseline),
                               threshold - sd * qnorm(baseline)),
                       if_else(frame=="gain", # new machine is at difference from baseline
                               threshold - sd * qnorm(1 - baseline) + mean_diff,
                               threshold - sd * qnorm(baseline) + mean_diff)),
        p_contract = if_else(frame=="gain", # probability of exceeding threshold to gain/keep contract
                              1 - pnorm((threshold - mean)/sd),
                              pnorm((threshold - mean)/sd)))

# spread values per machine across columns to get back to a conditions dataframe one row per trial
conds_df <- stim_df %>% # explanation: https://kieranhealy.org/blog/archives/2018/11/06/spreading-multiple-value
s/
  gather(variable, value, -(p_superiority:machine)) %>%
  unite(temp, machine, variable) %>%
  spread(temp, value)

head(conds_df)
```

```
##   p_superiority baseline frame threshold sd_diff mean_diff new_mean
## 1    0.02500000     0.15  loss        75      15 -29.39946 56.59358
## 2    0.02500000     0.50  loss        75      15 -29.39946 45.60054
## 3    0.02500000     0.85  loss        75      15 -29.39946 34.60750
## 4    0.03405957     0.15  loss        75      15 -27.36327 58.62977
## 5    0.03405957     0.50  loss        75      15 -27.36327 47.63673
## 6    0.03405957     0.85  loss        75      15 -27.36327 36.64370
##   new_p_contract  new_sd old_mean old_p_contract  old_sd
## 1      0.9586627 10.6066 85.99304           0.15 10.6066
## 2      0.9972127 10.6066 75.00000           0.50 10.6066
## 3      0.9999300 10.6066 64.00696           0.85 10.6066
## 4      0.9386332 10.6066 85.99304           0.15 10.6066
## 5      0.9950576 10.6066 75.00000           0.50 10.6066
## 6      0.9998506 10.6066 64.00696           0.85 10.6066
```

This results in an experimental design where the probability of gaining/keeping the contract for the new machine increases monotonically with p_superiority. This means that users should intervene only at extreme values of p_superiority. Even though the decision rule is not defined in terms of p_superiority, users can user effect size as a proxy for the decision task.



Keeping The Experiment Balanced Across Baseline Conditions

There are three variables which we set contingent on the baseline in order to maintain a consistent task structure and payoff scheme.

Balancing Task Structure

The first of these variables is the ratio $K = \frac{X}{C}$, where $\$X$ is the contract value and $\$C$ is the cost of the new machine. Since we hold the cost of the new machine constant at $1M, we can think of K as the *value of the contract in millions of dollars*. We need to set K according to the baseline probability of gaining/keeping the contract in order to *maintain an an equal number of trials where users should vs shouldn't intervene* at each level of baseline x gain/loss framing.

A feature of this kind of intervention problem is that the decision to intervene is only ambiguous under certain circumstances. For example, if the baseline probability of gaining/keeping the contract with the old machine is low, you should always intervene and buy the new machine unless the cost of intervention is high relative to the potential payoff of the contract. Similarly, if the baseline is high, you should never intervene unless the potential payoff is high relative to the cost of intervention.

```
# ratio (K) of value of contract (X) over cost of intervention (C)
conds_df <- conds_df %>%
  mutate(K = if_else(baseline == .15, # set K (cost of new machine in millions) contingent on baseline,
                     1.8,             # so there are an equal number of trials where participants should vs shoul
dn't intervene
                     if_else(baseline == .5,
                             2.25,
                             6.85))) # where baseline == .85

conds_df %>% group_by(baseline, K) %>% summarise()
```

```
## # A tibble: 3 x 2
## # Groups:   baseline [?]
##   baseline     K
##      <dbl> <dbl>
## 1     0.15  1.8
## 2     0.5   2.25
## 3     0.85  6.85
```

We want to check that we have an equal number of trials where intervening is and is not the optimal choice. We also want to make sure that this balance is maintained across all levels of baseline x framing.

```
# determine whether or not intervention is utility optimal on each trial
conds_df <- conds_df %>%
  mutate(should_intervene = if_else(frame=="gain",
                                    (old_p_contract + 1 / K) < new_p_contract, # gain framing decision rule
                                    ((1 - old_p_contract) - 1 / K) > (1 - new_p_contract))) # loss framing decisi
on rule

conds_df %>%
  group_by(baseline, K, frame) %>%
  summarise(intervene = sum(should_intervene), n_trials = n())
```

```
## # A tibble: 6 x 5
## # Groups:   baseline, K [?]
##   baseline     K frame intervene n_trials
##      <dbl> <dbl> <chr>     <int>    <int>
## 1     0.15  1.8  gain          6       12
## 2     0.15  1.8  loss          6       12
## 3     0.5   2.25 gain          6       12
## 4     0.5   2.25 loss          6       12
## 5     0.85  6.85 gain          6       12
## 6     0.85  6.85 loss          6       12
```

Balancing Expected Payoffs

The other two variables we set depending on the baseline probability of gaining/keeping the contract are the *starting value of the company* and the *exchange rate* of actual dollars in MTurk bonus per million of dollars in company value. We do this because with the value of the contract K varying across baseline conditions, workers in different conditions have the chance to win different amounts of company value (i.e., workers who start with a better machine also have higher contract values). *We need to make sure that workers receive a fair bonus regardless of what condition they are assigned to*.

The first consequence of higher contract values in higher baseline conditions is that these workers have more to lose in the loss framing trials. We offset this by *setting the starting value of the company equal to the maximum possible amount participants could lose*, if they buy the new machine every trial and always fail to gain/keep the contract.

```
# starting value depends on maximum possible loss (if they pay for the new machine every time and never gain\keep
the contract)
conds_df <- conds_df %>% mutate(starting_company_value = n_trials + n_trials / 2 * K)

conds_df %>% group_by(baseline, K, starting_company_value) %>% summarise()
```

```
## # A tibble: 3 x 3
## # Groups:   baseline, K [?]
##   baseline     K starting_company_value
##      <dbl> <dbl>                  <dbl>
## 1     0.15  1.8                    45.6
## 2     0.5   2.25                   51
## 3     0.85  6.85                  106.
```

However, this seems to introduce further inequity across conditions because workers who start with better machines also start with higher valued companies. We offset the relative disadvantage of users in the lower baseline conditions by giving them a higher exchange rate of MTurk bonus per million dollars of company value. Specifically, *we set the exchange rate to mainain a consistent expected bonus of $2.5 if participants make the optimal decision on every trial*.

```
# add expected change in company value on each trial to conditions dataframe
conds_df <- conds_df %>%
  mutate(expected_payoff = if_else(frame=="gain",
                                   # gain frame
                                   if_else(should_intervene,
                                           new_p_contract * K - 1,
                                           old_p_contract * K),
                                   # loss frame
                                   if_else(should_intervene,
                                           (1 - new_p_contract) * (-1 * K) - 1,
                                           (1 - old_p_contract) * (-1 * K))))

# set exchange rates in each baseline condition based on the expected company value at the end of the experiment
payoff_df <- conds_df %>% group_by(baseline, K, starting_company_value) %>%
  summarise(
    # expected and max company value
    expected_company_value = mean(starting_company_value) + sum(expected_payoff),
    max_company_value = mean(starting_company_value) + n_trials / 2 * mean(K), # if they always gain/keep the con
tract and never buy the new machine
    # bonus stats with exhange rate based on constant expected bonus of $2.5
    exchange_rate = round(2.5 / expected_company_value, 5), # five decimal places are necessary to avoid substant
ial rounding error
    expected_bonus = expected_company_value * exchange_rate,
    max_bonus = max_company_value * exchange_rate)

# merge these payoff variables into the conditions dataframe
conds_df <- conds_df %>% full_join(payoff_df, by = c("baseline","K","starting_company_value"))

knitr::kable(payoff_df)
```

baseline	K	starting_company_value	expected_company_value	max_company_value	exchange_rate	expected_bonus	max_bonus
0.15	1.80	45.6	34.14471	67.2	0.07322	2.500075	4.920384
0.50	2.25	51.0	52.05106	78.0	0.04803	2.500012	3.746340
0.85	6.85	106.2	163.99401	188.4	0.01524	2.499269	2.871216