Notebook: Races vs Tournament

Last updated: 23 June, 2017

Abstract

This document contains data analysis for the paper "Races or Tournaments?".

Contents

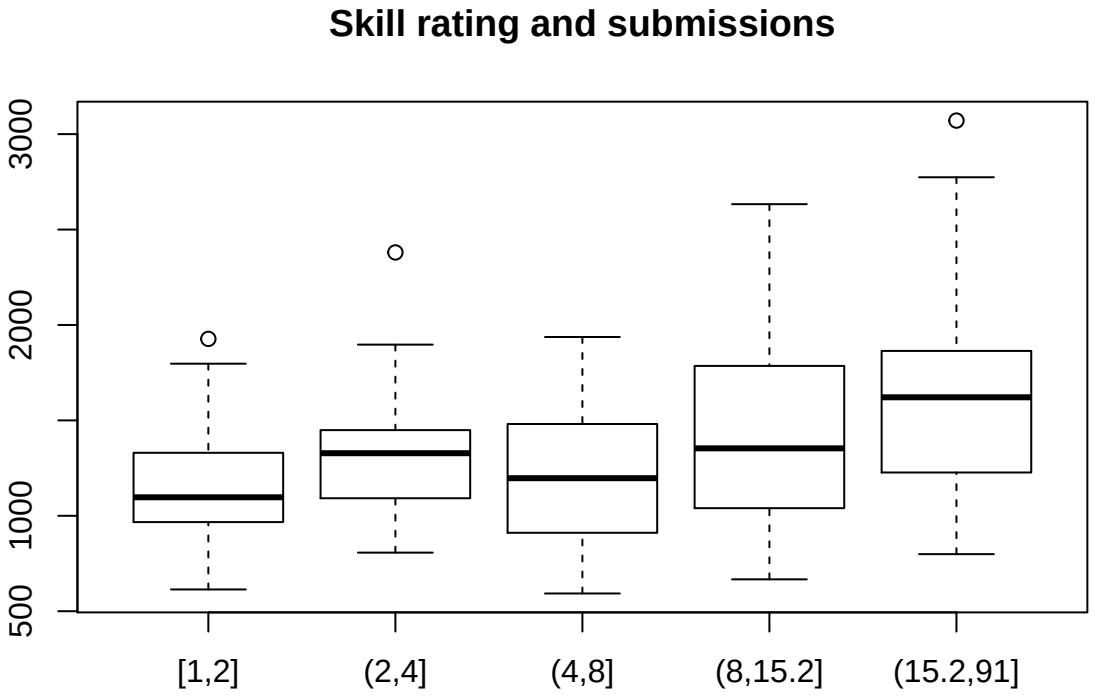
1 Descriptive statistics

Table 1 shows descriptive statistics for these variables. It also provides p-values from a series of Kruskal-Wallis non-parametric tests, showing no evidence of systematic differences across treatment groups (the lowest p-value was xxxx). Hence, our randomization appears successful.

```
dat <- within(races, {
    year <- as.numeric(format(member_date, "%Y"))
    hours <- week1 + week2 + week3 + week4
    rating <- mm_rating
    submissions <- mm_events
    registrations <- mm_reg
    lpaid <- log(paid)
    male <- gender == "Male"
    grad <- educ == "Doctorate/PhD" | educ == "Postgraduate/Master of arts"
    below30 <- age == "<20 years old" | age == "20-25 years old" |
        age == "26-30 years"
})

# FUNC to compute descriptive statistics
descriptives <- function(dat, treat = races$treatment) {
    balance.test <- function(x) {
        l <- split(x, treat)
        kruskal.test(l)$p.val
    }
    mu <- sapply(dat, mean, na.rm = TRUE)
    q50 <- sapply(dat, median, na.rm = TRUE)
    lo <- sapply(dat, min, na.rm = TRUE)
    hi <- sapply(dat, max, na.rm = TRUE)
    std <- sapply(dat, sd, na.rm = TRUE)
    pval <- sapply(dat, balance.test)
    n <- sapply(dat, function(x) sum(!is.na(x)))
    tab <- cbind(mu, q50, std, lo, hi, n, pval)
    colnames(tab) <- c("Mean", "Median", "St.Dev.", "Min", "Max",
        "Obs.", "P-value")
    return(tab)
}
covars <- with(dat, data.frame(year, rating, registrations, submissions,
    lpaid, nwins, ntop10, risk, hours, male, timezone, grad,
    below30))
tab <- descriptives(covars)
xtab <- xtable(tab)
digits(xtab) <- c(1, 1, rep(0, ncol(tab) - 2), 3)
align(xtab) <- c("@{}l", rep("r", ncol(tab)))
caption(xtab) <- "Descriptive statistics"
label(xtab) <- "summary"
render.xtable(xtab)



attach(races)
pcl <- cut(mm_events, breaks = quantile(mm_events[mm_events >
    0], p = 0:5/5), include = TRUE)
boxplot(mm_rating ~ pcl)
title("Skill rating and submissions")
```

Table 1: Descriptive statistics

	Mean	Median	St.Dev.	Min	Max	Obs.	P-value
year	2009.9	2010	4	2001	2015	299	0.596
rating	1322.4	1278	425	593	3071	205	0.989
registrations	17.6	9	23	1	161	299	0.626
submissions	7.2	2	12	0	91	299	0.867
lpaid	8.4	8	3	3	14	139	0.791
nwins	0.3	0	2	0	27	299	0.370
ntop10	1.6	0	5	0	64	299	0.273
risk	6.4	7	2	1	10	279	0.958
hours	31.3	24	25	0	192	277	0.995
male	1.0	1	0	0	1	276	0.375
timezone	2.1	2	5	-8	10	277	0.389
grad	0.5	0	1	0	1	278	0.208
below30	0.7	1	0	0	1	278	0.506

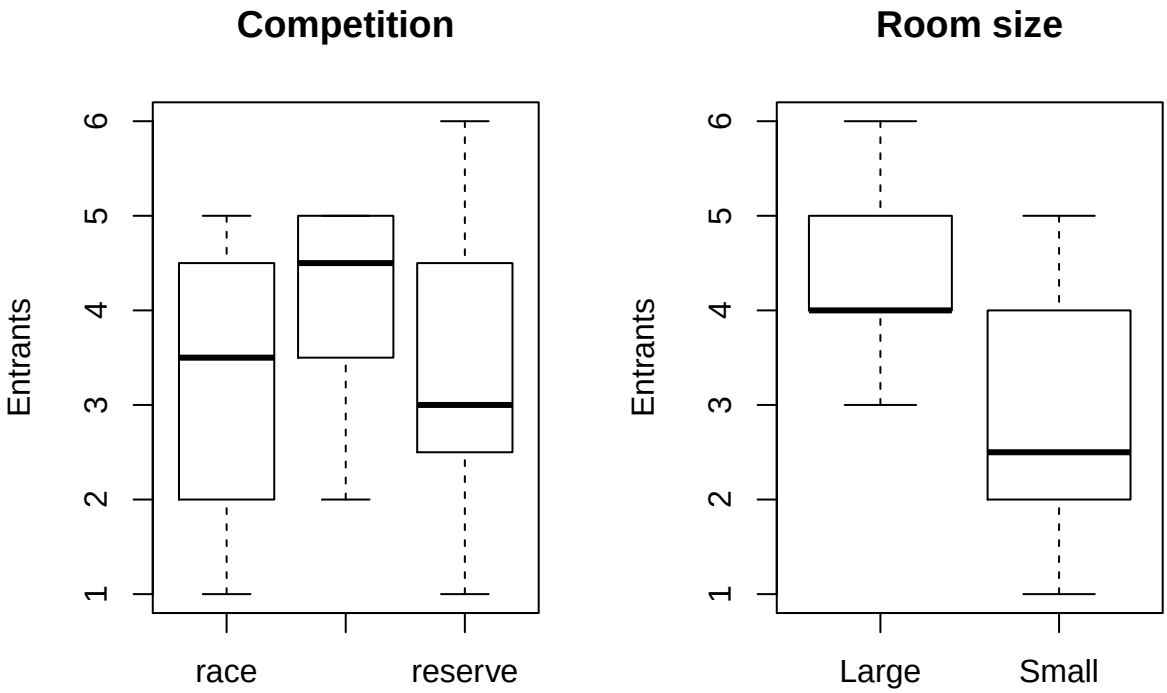



```
detach(races)

# Note: 1. rating is positively associated with time (perhaps
# selection of the top coders over time) 2. Also the variance
# seems positively associated with time.
```

2 Entry

We examine differences in the count of room entrants between different competition styles and large and small room sizes.

```
# Compute room entrants
counts <- aggregate(submit ~ room + room_size + treatment, data = races,
    sum)

par(mfrow = c(1, 2))
boxplot(submit ~ treatment, data = counts, ylab = "Entrants")
title("Competition")
boxplot(submit ~ room_size, data = counts, ylab = "Entrants")
title("Room size")
```




```
# ********************************************************************************#
# COMPETITION Test differences between threshold and
# tournament using Mann-Whitney's non-parametric test
# ********************************************************************************#
wilcox.test(submit ~ treatment == "tournament", data = counts,
    alternative = "less", exact = FALSE, correct = FALSE)
```

```
##
##  Wilcoxon rank sum test
##
## data:  submit by treatment == "tournament"
## W = 42.5, p-value = 0.08877
## alternative hypothesis: true location shift is less than 0
```

```
# Small sample demands for Bootstrap resampling Bootstrap
# permutation test, however, does not work well because with
# many ties p-values are unstable
mean.diff <- function(d, i) {
```

```
    diff(tapply(d$submit[i], d$treatment == "tournament", mean))
}
t.boot <- boot(counts, mean.diff, R = 9999, sim = "permutation")
hist(t.boot$t, breaks = "Scott")
abline(v = t.boot$t0)
mean(t.boot$t0 < t.boot$t)
```

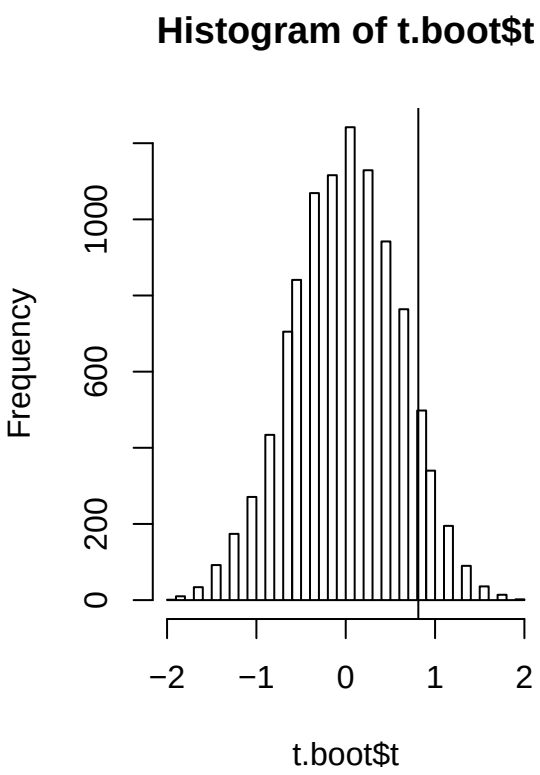
```
## [1] 0.06770677
```

```
mean(t.boot$t0 <= t.boot$t)
```

```
## [1] 0.1175118
```

```
# Bootstrap Wilcox works better ...

# ***************************************# ROOM SIZE Test
# differences between small and large
# ***************************************#
wilcox.test(submit ~ room_size, data = counts, exact = FALSE,
    correct = FALSE)
```

```
##
##  Wilcoxon rank sum test
##
## data:  submit by room_size
## W = 114, p-value = 0.01302
## alternative hypothesis: true location shift is not equal to 0
```



We then turn to the individual propensity to submit.

- We strongly reject differences in the propensity to submit between large and small rooms

- We do not have enough data for differences between competition styles

```
# Room size
tab.size <- with(races, table(submit, room_size))
fisher.test(tab.size)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  tab.size
## p-value = 1
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  0.5689643 1.6912635
## sample estimates:
## odds ratio
##  0.9846648
```

```
# Test differences in individual propensity to submit ==>
# Largely not significant!!!
```

```
# Competition styles ==> no overall significance
tab <- with(races, table(submit, treatment))
fisher.test(tab)
```

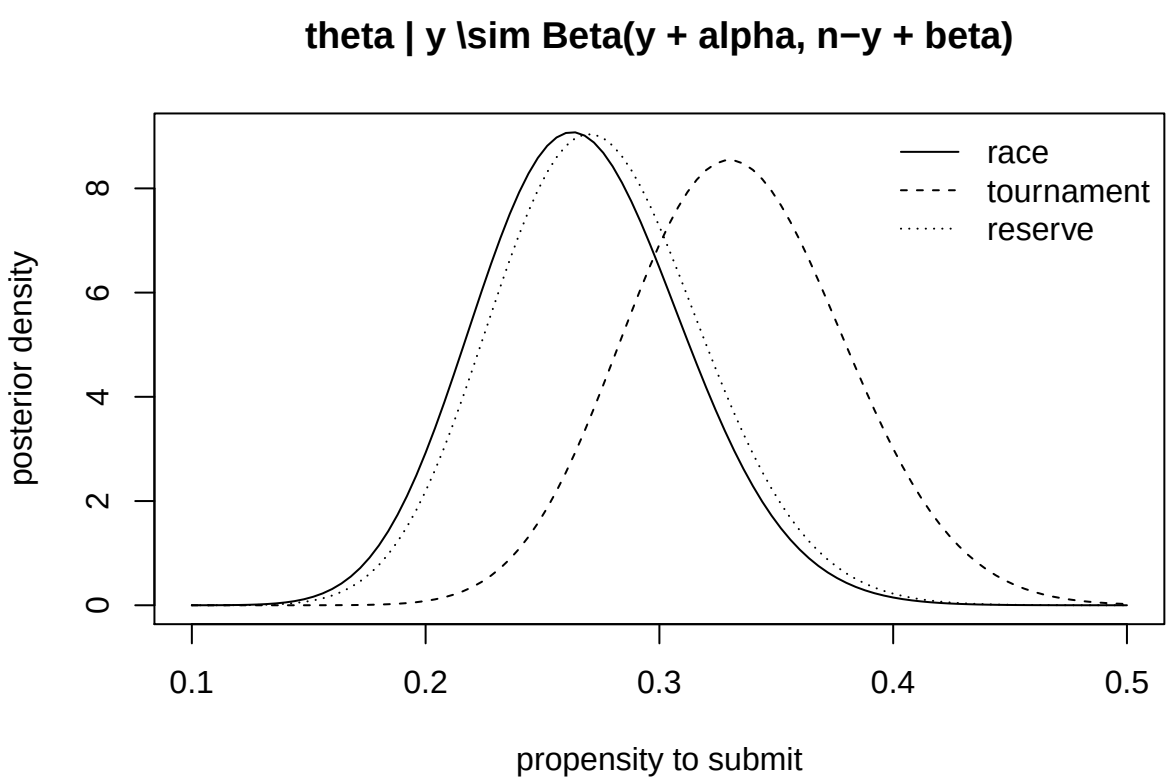
```
##
##  Fisher's Exact Test for Count Data
##
## data:  tab
## p-value = 0.5255
## alternative hypothesis: two.sided
```

```
# Competition styles with target one-sided p-value is small
# but not significant
tab <- with(races, table(submit, treatment == "tournament"))
fisher.test(tab, alternative = "greater")
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  tab
## p-value = 0.1558
## alternative hypothesis: true odds ratio is greater than 1
## 95 percent confidence interval:
##  0.8435475       Inf
## sample estimates:
## odds ratio
##   1.355339
```

One idea is to use Bayesian analysis. If we assume uniform priors, we can easily compute the posteriors in each treatment. Not sure what we can do with this result.

```r
# Bayesian analysis Compute posteriors Beta(1 + success, 1 +
# n - success)
plot.posterior <- function(x, ...) {
    tab <- with(races, table(submit, treatment))
    param <- data.frame(tab + 1)
    index <- param$treatment == x
    shape <- param[index, ]
    curve(dbeta(x, shape$Freq[2], shape$Freq[1]), ...)
}

# PLOTs
plot.posterior("race", from = 0.1, to = 0.5, xlab = "propensity to submit",
    ylab = "posterior density")
plot.posterior("tournament", add = TRUE, lty = 2)
plot.posterior("reserve", add = TRUE, lty = 3)
title("theta | y \\sim Beta(y + alpha, n-y + beta)")
legend("topright", legend = levels(races$treatment), lty = 1:3,
    bty = "n")
```



Our model indicates a latent variable y^* appraoch.

- We first look at logistic regression for entry
- Then we look at regression for the scores
- Finally, we use a Tobit-type of approach to back up the primitives of our model

2.1 Binary regression model for entry

For each individual we observe a variable Z_i, an indicator for the "competition style" of the room person i was assigned to; a binary outcome variable Y_i, equal to 1 if person i made a valid code submission during the competition [non example and with score greater than zero], and 0 otherwise; and a vector of covariates X_i,

these include measures of individual ability, such as the skill rating, platform experience, time availability during the competition, and risk aversion.
Since the outcome variable was a binary indicator, we estimate the logistic regression model:

$$\Pr(Y_i = 1 \mid Z_i, X_i) = \text{logit}^{-1}(\beta_0 + \beta_1 Z_i + \sum \beta_k X_{i,k})$$

where $logit(\cdot)$ is the logistic distribution function. [Results are below]

```
# Create dataset for regression imputing missing values at
# random
dat.imp <- within(dat, {
    set.seed(25978)
    rating.100 <- rating/100
    hours.imp <- impute(hours, "random")
    risk.imp <- impute(risk, "random")
    grad.imp <- impute(grad, "random")
    male.imp <- impute(male, "random")
    below30.imp <- impute(below30, "random")
    timezone.imp <- impute(timezone, "random")
    expert <- cut(submissions, quantile(submissions[!is.na(rating)]),
        include = TRUE)
})


# First model has no covariates (i.e., $\gamma=0$)
summary(m0 <- glm(submit ~ treatment, binomial(logit), data = dat.imp))
```

```
##
## Call:
## glm(formula = submit ~ treatment, family = binomial(logit), data = dat.imp)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8950  -0.7934  -0.7806   1.4891   1.6353
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.03236    0.22839  -4.520 6.18e-06 ***
## treatmenttournament  0.32418    0.31207   1.039    0.299
## treatmentreserve     0.03774    0.32077   0.118    0.906
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 358.81  on 298  degrees of freedom
## Residual deviance: 357.49  on 296  degrees of freedom
## AIC: 363.49
##
## Number of Fisher Scoring iterations: 4
```

```
# The second model adds the main skill rating measure which
# is available for 2/3 of our population. It seemed better to
# rescale rating in 100-point units and center it on the
# median value. Thus, the estimate of the intercept can be
# easily transformed in the probability of participation of
# the median rated individual assigned to a room with a race
```

```
# competition style.

summary(m1 <- update(m0, " ~ . + rating.100"))
```

```
##
## Call:
## glm(formula = submit ~ treatment + rating.100, family = binomial(logit),
##     data = dat.imp)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5846  -0.9441  -0.7840   1.2855   1.8284
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.15239    0.55935  -3.848 0.000119 ***
## treatmenttournament  0.31570    0.36635   0.862 0.388824
## treatmentreserve     0.20445    0.36931   0.554 0.579855
## rating.100           0.10470    0.03597   2.911 0.003607 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 268.13  on 204  degrees of freedom
## Residual deviance: 258.38  on 201  degrees of freedom
##   (94 observations deleted due to missingness)
## AIC: 266.38
##
## Number of Fisher Scoring iterations: 4
```

```
# The third column adds time availability (hours)
summary(m2 <- update(m1, " ~ . + hours.imp"))
```

```
##
## Call:
## glm(formula = submit ~ treatment + rating.100 + hours.imp, family = binomial(logit),
##     data = dat.imp)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6605  -0.9468  -0.7345   1.2738   1.8450
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.664748   0.617858  -4.313 1.61e-05 ***
## treatmenttournament  0.272190   0.373408   0.729  0.46604
## treatmentreserve     0.217824   0.372268   0.585  0.55846
## rating.100           0.109861   0.036498   3.010  0.00261 **
## hours.imp            0.014708   0.006568   2.239  0.02514 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 268.13  on 204  degrees of freedom
## Residual deviance: 252.54  on 200  degrees of freedom
##   (94 observations deleted due to missingness)
## AIC: 262.54
##
## Number of Fisher Scoring iterations: 4
```

```
# ... and experience (measured in quantiles)
summary(m3 <- update(m1, " ~ . + expert"))
```

```
##
## Call:
## glm(formula = submit ~ treatment + rating.100 + expert, family = binomial(logit),
##     data = dat.imp)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5636  -0.9789  -0.6616   1.1967   1.9411
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.51173    0.59965  -4.189 2.81e-05 ***
## treatmenttournament  0.27011    0.37274   0.725   0.4687
## treatmentreserve     0.17441    0.37682   0.463   0.6435
## rating.100           0.08134    0.03814   2.132   0.0330 *
## expert(2,6]          0.98853    0.44373   2.228   0.0259 *
## expert(6,13]         0.97346    0.44200   2.202   0.0276 *
## expert(13,91]        0.87059    0.46567   1.870   0.0615 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 268.13  on 204  degrees of freedom
## Residual deviance: 251.39  on 198  degrees of freedom
##   (94 observations deleted due to missingness)
## AIC: 265.39
##
## Number of Fisher Scoring iterations: 4
```

```
summary(m4 <- update(m1, "~ . + hours.imp + expert"))
```

```
##
## Call:
## glm(formula = submit ~ treatment + rating.100 + hours.imp + expert,
##     family = binomial(logit), data = dat.imp)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
```

```
## -1.6348  -0.9575  -0.6351   1.2033   1.9777
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -3.021885   0.660379  -4.576 4.74e-06 ***
## treatmenttournament  0.223710   0.380045   0.589   0.5561
## treatmentreserve     0.191657   0.378718   0.506   0.6128
## rating.100           0.085645   0.038749   2.210   0.0271 *
## hours.imp            0.014999   0.006861   2.186   0.0288 *
## expert(2,6]          0.968727   0.451177   2.147   0.0318 *
## expert(6,13]         0.974604   0.446754   2.182   0.0291 *
## expert(13,91]        0.893421   0.475069   1.881   0.0600 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 268.13  on 204  degrees of freedom
## Residual deviance: 245.80  on 197  degrees of freedom
##   (94 observations deleted due to missingness)
## AIC: 261.8
##
## Number of Fisher Scoring iterations: 4

# ... add demographics to m2
summary(m5 <- update(m4, " ~ . + timezone.imp + grad.imp + below30.imp + male.imp"))


##
## Call:
## glm(formula = submit ~ treatment + rating.100 + hours.imp + expert +
##     timezone.imp + grad.imp + below30.imp + male.imp, family = binomial(logit),
##     data = dat.imp)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7920  -0.9406  -0.6226   1.1733   1.9981
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.849557   1.380306  -2.064   0.0390 *
## treatmenttournament  0.234995   0.385664   0.609   0.5423
## treatmentreserve     0.263459   0.391451   0.673   0.5009
## rating.100           0.098799   0.041313   2.391   0.0168 *
## hours.imp            0.016878   0.007024   2.403   0.0163 *
## expert(2,6]          1.045782   0.457871   2.284   0.0224 *
## expert(6,13]         0.983540   0.457268   2.151   0.0315 *
## expert(13,91]        0.791015   0.493969   1.601   0.1093
## timezone.imp        -0.015756   0.031375  -0.502   0.6156
## grad.impTRUE        -0.430909   0.327791  -1.315   0.1887
## below30.impTRUE     -0.594834   0.338120  -1.759   0.0785 .
## male.impTRUE         0.182640   1.221643   0.150   0.8812
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 268.13  on 204  degrees of freedom
## Residual deviance: 240.98  on 193  degrees of freedom
##   (94 observations deleted due to missingness)
## AIC: 264.98
##
## Number of Fisher Scoring iterations: 4
```

```
# ... add risk aversion
summary(m6 <- update(m4, " ~ . + risk.imp"))
```

```
##
## Call:
## glm(formula = submit ~ treatment + rating.100 + hours.imp + expert +
##     risk.imp, family = binomial(logit), data = dat.imp)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6386  -0.9510  -0.6328   1.2033   1.9768
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.988244   0.811324  -3.683  0.00023 ***
## treatmenttournament  0.224630   0.380283   0.591  0.55473
## treatmentreserve     0.193324   0.379460   0.509  0.61042
## rating.100           0.085258   0.039116   2.180  0.02929 *
## hours.imp            0.015101   0.007022   2.151  0.03151 *
## expert(2,6]          0.967509   0.451494   2.143  0.03212 *
## expert(6,13]         0.974888   0.446772   2.182  0.02910 *
## expert(13,91]        0.889506   0.478170   1.860  0.06285 .
## risk.imp            -0.005082   0.071287  -0.071  0.94317
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 268.13  on 204  degrees of freedom
## Residual deviance: 245.79  on 196  degrees of freedom
##   (94 observations deleted due to missingness)
## AIC: 263.79
##
## Number of Fisher Scoring iterations: 4
```

```
# ... add everything
summary(m7 <- update(m5, " ~ . + risk.imp"))
```

```
##
## Call:
## glm(formula = submit ~ treatment + rating.100 + hours.imp + expert +
```

```
##     timezone.imp + grad.imp + below30.imp + male.imp + risk.imp,
##     family = binomial(logit), data = dat.imp)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7993  -0.9406  -0.6328   1.1886   1.9946
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.759978   1.487335  -1.856   0.0635 .
## treatmenttournament  0.237122   0.385882   0.614   0.5389
## treatmentreserve     0.265808   0.391796   0.678   0.4975
## rating.100           0.097963   0.041625   2.353   0.0186 *
## hours.imp            0.017094   0.007179   2.381   0.0173 *
## expert(2,6]          1.043824   0.458001   2.279   0.0227 *
## expert(6,13]         0.984107   0.457184   2.153   0.0314 *
## expert(13,91]        0.782535   0.496644   1.576   0.1151
## timezone.imp        -0.015107   0.031645  -0.477   0.6331
## grad.impTRUE        -0.438479   0.331211  -1.324   0.1855
## below30.impTRUE     -0.595226   0.338146  -1.760   0.0784 .
## male.impTRUE         0.174609   1.222941   0.143   0.8865
## risk.imp            -0.011856   0.073272  -0.162   0.8715
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 268.13  on 204  degrees of freedom
## Residual deviance: 240.95  on 192  degrees of freedom
##   (94 observations deleted due to missingness)
## AIC: 266.95
##
## Number of Fisher Scoring iterations: 4
```

```
# Compare models
models <- list(m0, m1, m2, m3, m4, m5, m6, m7)
stargazer(models, type = "text")
```

```
##
## ===============================================================================================
##                                          Dependent variable:
##                     -------------------------------------------------------------------------
##                                                  submit
##                         (1)       (2)       (3)       (4)       (5)       (6)       (7)       (8)
## -----------------------------------------------------------------------------------------------
## treatmenttournament   0.324     0.316     0.272     0.270     0.224     0.235     0.225     0.237
##                      (0.312)   (0.366)   (0.373)   (0.373)   (0.380)   (0.386)   (0.380)   (0.386)
##
## treatmentreserve      0.038     0.204     0.218     0.174     0.192     0.263     0.193     0.266
##                      (0.321)   (0.369)   (0.372)   (0.377)   (0.379)   (0.391)   (0.379)   (0.392)
##
## rating.100                    0.105***  0.110***   0.081**   0.086**   0.099**   0.085**   0.098**
```

```
##                                (0.036)   (0.036)   (0.038)   (0.039)  (0.041)   (0.039)  (0.042)
##
## hours.imp                               0.015**             0.015**  0.017**   0.015**  0.017**
##                                         (0.007)             (0.007)  (0.007)   (0.007)  (0.007)
##
## expert(2,6]                                       0.989**   0.969**  1.046**   0.968**  1.044**
##                                                   (0.444)   (0.451)  (0.458)   (0.451)  (0.458)
##
## expert(6,13]                                      0.973**   0.975**  0.984**   0.975**  0.984**
##                                                   (0.442)   (0.447)  (0.457)   (0.447)  (0.457)
##
## expert(13,91]                                     0.871*    0.893*    0.791    0.890*    0.783
##                                                   (0.466)   (0.475)  (0.494)   (0.478)  (0.497)
##
## timezone.imp                                                          -0.016              -0.015
##                                                                       (0.031)             (0.032)
##
## grad.imp                                                              -0.431              -0.438
##                                                                       (0.328)             (0.331)
##
## below30.imp                                                           -0.595*             -0.595*
##                                                                       (0.338)             (0.338)
##
## male.imp                                                               0.183               0.175
##                                                                       (1.222)             (1.223)
##
## risk.imp                                                                         -0.005    -0.012
##                                                                                  (0.071)   (0.073)
##
## Constant            -1.032*** -2.152*** -2.665*** -2.512*** -3.022*** -2.850** -2.988*** -2.760*
##                      (0.228)   (0.559)   (0.618)   (0.600)   (0.660)  (1.380)   (0.811)  (1.487)
##
## ------------------------------------------------------------------------------------------------
## Observations           299       205       205       205       205      205       205      205
## Log Likelihood      -178.747  -129.192  -126.272  -125.695  -122.899 -120.488  -122.896 -120.475
## Akaike Inf. Crit.    363.493   266.383   262.543   265.389   261.798  264.975   263.793  266.949
## ================================================================================================
## Note:                                                                *p<0.1; **p<0.05; ***p<0.01
```

```r
# Compare models in terms of prediction accuracy
accuracy <- function(fit) {
    yhat <- predict(fit, type = "response")
    tab <- table(predicted = ifelse(yhat > 0.5, 1, 0), actual = fit$y)
    tp <- tab[2, 2]
    fp <- tab[2, 1]
    fn <- tab[1, 2]
    precision <- tp/(tp + fp)
    recall <- tp/(tp + fn)
    fmeasure <- 2 * precision * recall/(precision + recall)
    list(conf.table = tab, precision = precision, recall = recall,
        f.measure = fmeasure)
}
accuracy(m1)
```

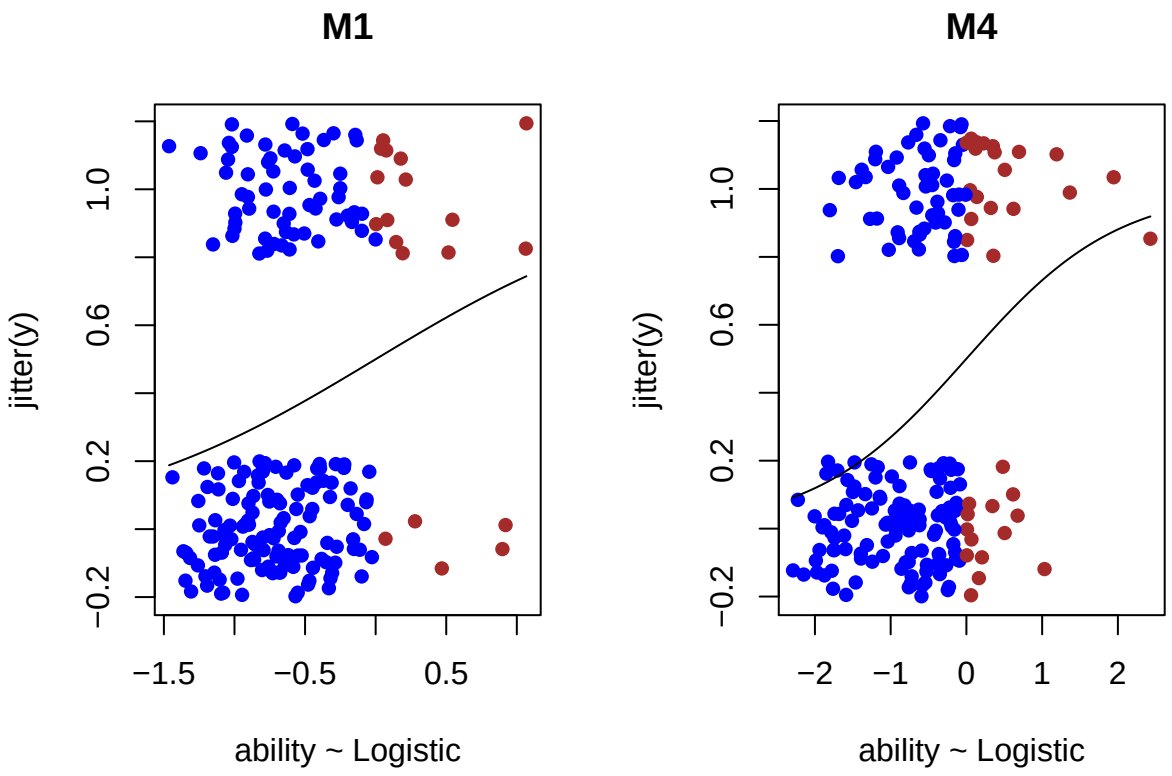
```
## $conf.table
```

```
##           actual
## predicted   0   1
##         0 126  60
##         1   5  14
##
## $precision
## [1] 0.7368421
##
## $recall
## [1] 0.1891892
##
## $f.measure
## [1] 0.3010753
```

```
accuracy(m4)
```

```
## $conf.table
##           actual
## predicted   0   1
##         0 117  54
##         1  14  20
##
## $precision
## [1] 0.5882353
##
## $recall
## [1] 0.2702703
##
## $f.measure
## [1] 0.3703704
```

```
summarize.fit <- function(x) {
    yhat <- predict(x)
    with(x, plot(jitter(y) ~ yhat, col = ifelse(yhat > 0, "brown",
        "blue"), pch = 16, xlab = "ability ~ Logistic"))
    curve(ilogit, add = T)
}
par(mfrow = c(1, 2))
summarize.fit(m1)
title("M1")
summarize.fit(m4)
title("M4")
```




```
# Using Akaike criterion to select the best model, we now
# compute the model for each treatment alone
summary(m4.race <- update(m4, "~ . -treatment", subset = treatment ==
    "race"))
```

```
##
## Call:
## glm(formula = submit ~ rating.100 + hours.imp + expert, family = binomial(logit),
##     data = dat.imp, subset = treatment == "race")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3576  -0.8870  -0.6202   1.0724   1.9123
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -3.19401    1.14108  -2.799  0.00512 **
## rating.100     0.11434    0.07253   1.576  0.11493
## hours.imp      0.01693    0.01301   1.301  0.19333
## expert(2,6]    1.15160    0.74129   1.554  0.12030
## expert(6,13]   0.28067    0.81908   0.343  0.73185
## expert(13,91]  0.12448    0.88162   0.141  0.88771
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 85.612  on 67  degrees of freedom
```

```
## Residual deviance: 77.949  on 62  degrees of freedom
##   (31 observations deleted due to missingness)
## AIC: 89.949
##
## Number of Fisher Scoring iterations: 4
```

```
summary(m4.tour <- update(m4, "~ . -treatment", subset = treatment ==
    "tournament"))
```

```
##
## Call:
## glm(formula = submit ~ rating.100 + hours.imp + expert, family = binomial(logit),
##     data = dat.imp, subset = treatment == "tournament")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3072  -0.8695  -0.5077   0.9183   2.1477
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -4.30329    1.23282  -3.491 0.000482 ***
## rating.100     0.17457    0.07894   2.211 0.027014 *
## hours.imp      0.02179    0.01197   1.820 0.068719 .
## expert(2,6]    0.76336    0.85111   0.897 0.369778
## expert(6,13]   1.42465    0.83266   1.711 0.087091 .
## expert(13,91]  1.00773    0.90277   1.116 0.264307
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 92.367  on 68  degrees of freedom
## Residual deviance: 73.652  on 63  degrees of freedom
##   (31 observations deleted due to missingness)
## AIC: 85.652
##
## Number of Fisher Scoring iterations: 4
```

```
summary(m4.rese <- update(m4, "~ . -treatment", subset = treatment ==
    "reserve"))
```

```
##
## Call:
## glm(formula = submit ~ rating.100 + hours.imp + expert, family = binomial(logit),
##     data = dat.imp, subset = treatment == "reserve")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1725  -1.0304  -0.6758   1.2378   1.8433
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)   -1.103709   1.078003  -1.024    0.306
## rating.100    -0.011692   0.065915  -0.177    0.859
## hours.imp     -0.004682   0.015603  -0.300    0.764
## expert(2,6]    0.908819   0.820728   1.107    0.268
## expert(6,13]   1.186993   0.747693   1.588    0.112
## expert(13,91]  1.272824   0.791994   1.607    0.108
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 89.446  on 67  degrees of freedom
## Residual deviance: 85.463  on 62  degrees of freedom
##   (32 observations deleted due to missingness)
## AIC: 97.463
##
## Number of Fisher Scoring iterations: 4
```

```
models <- list(m4, m4.race, m4.tour, m4.rese)
stargazer(models, type = "text")
```

```
##
## ===========================================================
##                              Dependent variable:
##                     ------------------------------------
##                                    submit
##                        (1)       (2)       (3)      (4)
## -----------------------------------------------------------
## treatmenttournament   0.224
##                      (0.380)
##
## treatmentreserve      0.192
##                      (0.379)
##
## rating.100           0.086**    0.114    0.175**  -0.012
##                      (0.039)   (0.073)   (0.079)  (0.066)
##
## hours.imp            0.015**    0.017    0.022*   -0.005
##                      (0.007)   (0.013)   (0.012)  (0.016)
##
## expert(2,6]          0.969**    1.152     0.763    0.909
##                      (0.451)   (0.741)   (0.851)  (0.821)
##
## expert(6,13]         0.975**    0.281    1.425*    1.187
##                      (0.447)   (0.819)   (0.833)  (0.748)
##
## expert(13,91]        0.893*     0.124     1.008    1.273
##                      (0.475)   (0.882)   (0.903)  (0.792)
##
## Constant            -3.022*** -3.194*** -4.303*** -1.104
##                      (0.660)   (1.141)   (1.233)  (1.078)
##
## -----------------------------------------------------------
## Observations           205       68        69       68
```

```
## Log Likelihood      -122.899   -38.974   -36.826  -42.732
## Akaike Inf. Crit.    261.798   89.949    85.652   97.463
## ============================================================
## Note:                          *p<0.1; **p<0.05; ***p<0.01
```

```
# Using simpler model
summary(m2.race <- update(m2, "~ . -treatment", subset = treatment ==
    "race"))
```

```
##
## Call:
## glm(formula = submit ~ rating.100 + hours.imp, family = binomial(logit),
##     data = dat.imp, subset = treatment == "race")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.4946  -0.8544  -0.7012   1.3305   1.8498
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.75184    1.04124  -2.643  0.00822 **
## rating.100   0.11124    0.06398   1.739  0.08208 .
## hours.imp    0.01694    0.01295   1.308  0.19092
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 85.612  on 67  degrees of freedom
## Residual deviance: 80.975  on 65  degrees of freedom
##   (31 observations deleted due to missingness)
## AIC: 86.975
##
## Number of Fisher Scoring iterations: 4
```

```
summary(m2.tour <- update(m2, "~ . -treatment", subset = treatment ==
    "tournament"))
```

```
##
## Call:
## glm(formula = submit ~ rating.100 + hours.imp, family = binomial(logit),
##     data = dat.imp, subset = treatment == "tournament")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1822  -0.8724  -0.5789   1.0166   2.0202
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.83514    1.13085  -3.391 0.000695 ***
## rating.100   0.19594    0.07377   2.656 0.007906 **
## hours.imp    0.02400    0.01222   1.964 0.049533 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 92.367  on 68  degrees of freedom
## Residual deviance: 76.886  on 66  degrees of freedom
##   (31 observations deleted due to missingness)
## AIC: 82.886
##
## Number of Fisher Scoring iterations: 5
```

```
summary(m2.rese <- update(m2, "~ . -treatment", subset = treatment ==
    "reserve"))
```

```
##
## Call:
## glm(formula = submit ~ rating.100 + hours.imp, family = binomial(logit),
##     data = dat.imp, subset = treatment == "reserve")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1220  -0.9696  -0.9198   1.3749   1.5916
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.678731   1.003985  -0.676    0.499
## rating.100   0.024927   0.061200   0.407    0.684
## hours.imp   -0.006866   0.014804  -0.464    0.643
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 89.446  on 67  degrees of freedom
## Residual deviance: 88.988  on 65  degrees of freedom
##   (32 observations deleted due to missingness)
## AIC: 94.988
##
## Number of Fisher Scoring iterations: 4
```

```
# Model w/reserve is very hard to predict
models <- list(m2, m2.race, m2.tour, m2.rese)
stargazer(models, type = "text")
```

```
##
## ===============================================================
##                                 Dependent variable:
##                     ------------------------------------------
##                                        submit
##                         (1)        (2)        (3)       (4)
## ---------------------------------------------------------------
## treatmenttournament    0.272
##                       (0.373)
##
## treatmentreserve       0.218
```

```
##                       (0.372)
##
## rating.100           0.110***   0.111*   0.196***   0.025
##                       (0.036)   (0.064)   (0.074)  (0.061)
##
## hours.imp             0.015**    0.017    0.024**  -0.007
##                       (0.007)   (0.013)   (0.012)  (0.015)
##
## Constant            -2.665*** -2.752*** -3.835*** -0.679
##                       (0.618)   (1.041)   (1.131)  (1.004)
##
## ------------------------------------------------------------
## Observations            205       68        69       68
## Log Likelihood       -126.272   -40.487   -38.443  -44.494
## Akaike Inf. Crit.     262.543   86.975    82.886   94.988
## ============================================================
## Note:                           *p<0.1; **p<0.05; ***p<0.01
```

```
# This identification result holds under the assumption that
# the skill distribution is logistic. Next, we relax this
# distributional assumption. First, we check different
# distributions [e.g., probit, cauchit, cloglog]. We find
# Probit model seems slightly better in terms of deviance
probit <- update(m4, family = binomial(probit))
cauchit <- update(m4, family = binomial(cauchit))
cloglog <- update(m4, family = binomial(cloglog))
fit <- list(logit = m4, probit = probit, cauchit = cauchit, cloglog = cloglog)
cbind(deviance = sapply(fit, deviance), n = sapply(fit, function(x) length(x$y)))
```

```
##         deviance   n
## logit   245.7979 205
## probit  245.5361 205
## cauchit 246.7807 205
## cloglog 245.8205 205
```

```
# ... but not necessarily in terms of prediction accuracy
accuracy(probit)
```

```
## $conf.table
##          actual
## predicted   0   1
##         0 118  56
##         1  13  18
##
## $precision
## [1] 0.5806452
##
## $recall
## [1] 0.2432432
##
## $f.measure
## [1] 0.3428571
```

```
accuracy(m4)
```

```
## $conf.table
##           actual
## predicted   0   1
##         0 117  54
##         1  14  20
##
## $precision
## [1] 0.5882353
##
## $recall
## [1] 0.2702703
##
## $f.measure
## [1] 0.3703704
```

Under the assumption of independence, the parameter β_1 is the average treatment effect of the competition style on participation. Correlation in decisions can be introduced with a quasi-binomial specification.
The estimtes have a structural interpetation. In terms of our contest theoretical model, due to the monotonicity of equilibrium, entry occurs only if the person i's ability is greater than a given threshold a_0 (the marginal type). This threshold is determined by the competition style among other things. Given in our experimental data one can assume that all else is equal except the competition style, this coefficient identifies the marginal type conditional on the competition style.
Here we try some more stuff ...

```
# Klein-spady ... Not sure about this implementation!
m.np <- np::npindexbw(as.numeric(submit) ~ rating + lhours.imp +
    treatment, data = dat, method = "kleinspady")
```

```
## Error in np::npindexbw(as.numeric(submit) ~ rating + lhours.imp + treatment, : could not find functi
```

```
summary(m.np)
```

```
## Error in summary(m.np): object 'm.np' not found
```

```
# Bayesian models
model <- "submit ~ treatment + tothours.imp + mm_rating.100 + educ+gender+timezone"
# bayes <- arm::bayesglm(model, family=binomial)
# summary(bayes)

# GAMs Generalized additive models
b <- mgcv::gam(submit ~ treatment + s(lhours.imp, rating), data = dat,
    family = binomial)
```

```
## Error in eval(expr, envir, enclos): object 'lhours.imp' not found
```

3 Timing of entry decision using panel data

Here we consider the panel of submissions. Next we look at the timing of submissions

```
data(scores)

# Create panel
create.panel <- function(z, id) {
    diffmin <- function(x) x - min(x)
    z$date <- z$timestamp %>% as.Date %>% as.numeric %>% diffmin
    z$y <- 1
    g <- expand.grid(date = seq(min(z$date), max(z$date)), coder_id = id,
        y = 0)
    g2 <- rbind(g, z[, c("date", "coder_id", "y")])
    g2$date <- ifelse(g2$date < 3, 1, ifelse(g2$date < 5, 2,
        ifelse(g2$date < 7, 3, 4)))
    aggregate(y ~ date + coder_id, data = g2, FUN = sum)
}

scores.panel0 <- create.panel(scores, races$coder_id)
scores.panel <- merge(scores.panel0, races, by = "coder_id")


# Unconditional semi-parametric [Odds(first) = exp(alpha +
# beta * t )]
scores.panel$submit <- scores.panel$y > 0
clog <- clogit(submit ~ treatment + strata(date2), data = scores.panel)
```

```
## Error in strata(date2): object 'date2' not found
```

```
summary(clog)
```

```
## Error in summary(clog): object 'clog' not found
```

```
# Logistic
m.log <- glm(submit ~ treatment + factor(date), binomial(logit),
    data = scores.panel)
summary(m.log)
```

```
##
## Call:
## glm(formula = submit ~ treatment + factor(date), family = binomial(logit),
##     data = scores.panel)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.6938  -0.6138  -0.5635  -0.5287   2.0297
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.78616    0.19861  -8.994   <2e-16 ***
## treatmenttournament  0.13586    0.19251   0.706    0.480
## treatmentreserve     0.02642    0.19592   0.135    0.893
## factor(date)2       -0.13734    0.23461  -0.585    0.558
## factor(date)3        0.07646    0.22583   0.339    0.735
## factor(date)4        0.34873    0.21687   1.608    0.108
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1057.1  on 1195  degrees of freedom
## Residual deviance: 1051.3  on 1190  degrees of freedom
## AIC: 1063.3
##
## Number of Fisher Scoring iterations: 4
```

```
## Add rating
clog2 <- clogit(submit ~ treatment + I(mm_rating/100) + strata(date),
    data = scores.panel)
summary(clog2)
```

```
## Call:
## coxph(formula = Surv(rep(1, 1196L), submit) ~ treatment + I(mm_rating/100) +
##     strata(date), data = scores.panel, method = "exact")
##
##   n= 820, number of events= 164
##    (376 observations deleted due to missingness)
##
##                          coef exp(coef) se(coef)     z Pr(>|z|)
## treatmenttournament 0.01022   1.01027  0.22049 0.046    0.963
## treatmentreserve    0.16576   1.18029  0.21639 0.766    0.444
## I(mm_rating/100)    0.09301   1.09747  0.01969 4.724 2.31e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                     exp(coef) exp(-coef) lower .95 upper .95
## treatmenttournament     1.010     0.9898    0.6558     1.556
## treatmentreserve        1.180     0.8472    0.7723     1.804
## I(mm_rating/100)        1.097     0.9112    1.0559     1.141
##
## Rsquare= 0.028   (max possible= 0.621 )
## Likelihood ratio test= 22.95  on 3 df,   p=4.14e-05
## Wald test            = 22.97  on 3 df,   p=4.098e-05
## Score (logrank) test = 24.24  on 3 df,   p=2.23e-05
```

```
# Add week hours
hours <- 0
for (i in 1:4) {
    w <- with(scores.panel, switch(i, `1` = week1, `2` = week2,
        `3` = week3, `4` = week4))
    index <- scores.panel$date == i
    hours[index] <- w[index]
}
clog3 <- clogit(submit ~ treatment + hours + strata(date), data = scores.panel)
summary(clog3)
```

```
## Call:
## coxph(formula = Surv(rep(1, 1196L), submit) ~ treatment + hours +
##     strata(date), data = scores.panel, method = "exact")
##
```

```
##   n= 1109, number of events= 179
##    (87 observations deleted due to missingness)
##
##                         coef exp(coef) se(coef)     z Pr(>|z|)
## treatmenttournament 0.06715   1.06946  0.20204 0.332  0.73961
## treatmentreserve    0.10970   1.11595  0.20108 0.546  0.58536
## hours               0.03124   1.03173  0.01024 3.051  0.00228 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                     exp(coef) exp(-coef) lower .95 upper .95
## treatmenttournament     1.069     0.9351    0.7198     1.589
## treatmentreserve        1.116     0.8961    0.7525     1.655
## hours                   1.032     0.9692    1.0112     1.053
##
## Rsquare= 0.008   (max possible= 0.577 )
## Likelihood ratio test= 8.89  on 3 df,   p=0.03085
## Wald test            = 9.53  on 3 df,   p=0.02299
## Score (logrank) test = 9.96  on 3 df,   p=0.01893
```

```
clog4 <- clogit(submit ~ treatment + I(mm_rating/100) + hours +
    strata(date), data = scores.panel)
summary(clog4)
```

```
## Call:
## coxph(formula = Surv(rep(1, 1196L), submit) ~ treatment + I(mm_rating/100) +
##     hours + strata(date), data = scores.panel, method = "exact")
##
##   n= 780, number of events= 153
##    (416 observations deleted due to missingness)
##
##                          coef exp(coef) se(coef)      z Pr(>|z|)
## treatmenttournament -0.17592   0.83869  0.23367 -0.753    0.452
## treatmentreserve     0.15441   1.16697  0.21940  0.704    0.482
## I(mm_rating/100)     0.09380   1.09834  0.02041  4.595 4.33e-06 ***
## hours                0.05136   1.05270  0.01316  3.902 9.53e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                     exp(coef) exp(-coef) lower .95 upper .95
## treatmenttournament    0.8387     1.1923    0.5305     1.326
## treatmentreserve       1.1670     0.8569    0.7591     1.794
## I(mm_rating/100)       1.0983     0.9105    1.0553     1.143
## hours                  1.0527     0.9499    1.0259     1.080
##
## Rsquare= 0.043   (max possible= 0.617 )
## Likelihood ratio test= 34.45  on 4 df,   p=6.038e-07
## Wald test            = 33.52  on 4 df,   p=9.348e-07
## Score (logrank) test = 36.09  on 4 df,   p=2.769e-07
```

```
# Compare models
models <- list(clog, clog2, clog3, clog4, m.log)
```

```
## Error in eval(expr, envir, enclos): object 'clog' not found
```

```
stargazer(models, type = "text")
```

```
##
## ===========================================================
##                              Dependent variable:
##                     ------------------------------------
##                                      submit
##                         (1)       (2)       (3)      (4)
## -----------------------------------------------------------
## treatmenttournament   0.272
##                      (0.373)
##
## treatmentreserve      0.218
##                      (0.372)
##
## rating.100          0.110***   0.111*   0.196***   0.025
##                      (0.036)   (0.064)   (0.074)  (0.061)
##
## hours.imp            0.015**    0.017    0.024**  -0.007
##                      (0.007)   (0.013)   (0.012)  (0.015)
##
## Constant            -2.665*** -2.752*** -3.835*** -0.679
##                      (0.618)   (1.041)   (1.131)  (1.004)
##
## -----------------------------------------------------------
## Observations           205       68        69       68
## Log Likelihood      -126.272   -40.487   -38.443  -44.494
## Akaike Inf. Crit.    262.543   86.975    82.886   94.988
## ===========================================================
## Note:                           *p<0.1; **p<0.05; ***p<0.01
```

We transform data in a panel with the date of the first submission. The probability of entry in a given date is modeled with a conditional logit. We find that the probability of entry in a given date decreases over time. (if it was at random 1/8 chances of having a submission in a given date). The estimated drop is about 13 percent from one date to the next (a reduction of above 100% from the first to the last date). Using interactions with treatment dummies, we find that the decrease in the probability of entry is negative in all treatments but it is about 25 percent in the race (with pvalue<0.05) it is only 8 (pval) and 6 (pval) percent for tournaments and reserve, respectively.

```
# Create variable
dmin <- function(x) x - min(x)
scores$date <- scores$timestamp %>% as.Date %>% as.numeric %>%
    dmin

create.panel <- function(data) {
    data$submit_first <- 1
    g <- with(data, expand.grid(date = seq(min(date), max(date)),
        coder_id = unique(coder_id), submit_first = 0))
    g2 <- rbind(g, data[, c("date", "coder_id", "submit_first")])
    aggregate(submit_first ~ date + coder_id, g2, FUN = sum)
}

scores.panel <- create.panel(subset(scores, submission == 1))
```

```
z <- merge(scores.panel, races, by = "coder_id")


# Unconditional semi-parametric [Odds(first) = exp(alpha +
# beta * t )]
fit <- rep()
fit$clog <- clogit(submit_first ~ date, data = z)
summary(fit$clog)
```

```
## Call:
## coxph(formula = Surv(rep(1, 774L), submit_first) ~ date, data = z,
##     method = "exact")
##
##   n= 774, number of events= 86
##
##         coef exp(coef) se(coef)      z Pr(>|z|)
## date -0.1510    0.8598   0.0461 -3.275  0.00106 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      exp(coef) exp(-coef) lower .95 upper .95
## date    0.8598      1.163    0.7855    0.9412
##
## Rsquare= 0.014   (max possible= 0.498 )
## Likelihood ratio test= 11.17  on 1 df,   p=0.0008303
## Wald test            = 10.73  on 1 df,   p=0.001055
## Score (logrank) test = 11.02  on 1 df,   p=0.0008998
```

```
# Unconditional parametric [Odds(first) = exp(alpha + beta *
# t )]
fit$logi <- glm(submit_first ~ date, data = z, binomial(logit))
summary(fit$logi)
```

```
##
## Call:
## glm(formula = submit_first ~ date, family = binomial(logit),
##     data = z)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.6252  -0.5441  -0.4394  -0.3801   2.3687
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.53318    0.18853  -8.132 4.21e-16 ***
## date        -0.15121    0.04614  -3.277  0.00105 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 539.99  on 773  degrees of freedom
## Residual deviance: 528.81  on 772  degrees of freedom
```

```
## AIC: 532.81
##
## Number of Fisher Scoring iterations: 5
```

```
# Unconditional with interaction
fit$logi2 <- glm(submit_first ~ date + date:treatment, data = z,
    binomial(logit))
summary(fit$logi2)
```

```
##
## Call:
## glm(formula = submit_first ~ date + date:treatment, family = binomial(logit),
##     data = z)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.6267  -0.5273  -0.4601  -0.3801   2.5737
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -1.52808    0.18858  -8.103 5.36e-16 ***
## date                      -0.21835    0.07201  -3.032  0.00243 **
## date:treatmenttournament   0.08546    0.07472   1.144  0.25273
## date:treatmentreserve      0.09344    0.07689   1.215  0.22423
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 539.99  on 773  degrees of freedom
## Residual deviance: 526.96  on 770  degrees of freedom
## AIC: 534.96
##
## Number of Fisher Scoring iterations: 5
```

```
# Note the difference when the intercept is not there

# Conditional [Odds(first) = exp(alpha_i + beta * t )]
fit$coclog <- clogit(submit_first ~ date + strata(coder_id),
    data = z)
summary(fit$coclog)
```

```
## Call:
## coxph(formula = Surv(rep(1, 774L), submit_first) ~ date + strata(coder_id),
##     data = z, method = "exact")
##
##   n= 774, number of events= 86
##
##         coef exp(coef) se(coef)      z Pr(>|z|)
## date -0.1340    0.8746   0.0433 -3.095  0.00197 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      exp(coef) exp(-coef) lower .95 upper .95
```

```
## date    0.8746      1.143    0.8034    0.9521
##
## Rsquare= 0.013   (max possible= 0.386 )
## Likelihood ratio test= 9.93  on 1 df,   p=0.001627
## Wald test            = 9.58  on 1 df,   p=0.00197
## Score (logrank) test = 9.81  on 1 df,   p=0.001735
```

```
# Interactions with treatment dummies [Odds(first) =
# exp(alpha_i + beta_i * t )]
fit$coclog2 <- clogit(submit_first ~ date:treatment + strata(coder_id),
    data = z)
summary(fit$coclog2)
```

```
## Call:
## coxph(formula = Surv(rep(1, 774L), submit_first) ~ date:treatment +
##     strata(coder_id), data = z, method = "exact")
##
##   n= 774, number of events= 86
##
##                              coef exp(coef) se(coef)      z Pr(>|z|)
## date:treatmentrace       -0.27991   0.75585  0.08801 -3.180  0.00147 **
## date:treatmenttournament -0.08726   0.91644  0.06847 -1.274  0.20253
## date:treatmentreserve    -0.06708   0.93512  0.07522 -0.892  0.37256
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                          exp(coef) exp(-coef) lower .95 upper .95
## date:treatmentrace          0.7559      1.323    0.6361    0.8982
## date:treatmenttournament    0.9164      1.091    0.8013    1.0481
## date:treatmentreserve       0.9351      1.069    0.8069    1.0837
##
## Rsquare= 0.018   (max possible= 0.386 )
## Likelihood ratio test= 14.16  on 3 df,   p=0.002689
## Wald test            = 12.53  on 3 df,   p=0.005761
## Score (logrank) test = 13.61  on 3 df,   p=0.003487
```

```
# Note: the above model is the same as clogit(submit_first ~
# date + date:treatment + strata(coder_id), data=z)
```

```
# Coefficients
sapply(fit, coef)
```

```
## $clog
##       date
## -0.1510078
##
## $logi
## (Intercept)        date
##  -1.5331826  -0.1512076
##
## $logi2
##              (Intercept)                      date date:treatmenttournament
##              -1.52808481              -0.21835013                0.08545552
```

```
##    date:treatmentreserve
##               0.09344282
##
## $coclog
##       date
## -0.1339918
##
## $coclog2
##       date:treatmentrace date:treatmenttournament    date:treatmentreserve
##              -0.27990753              -0.08725850              -0.06707555
```

```
# Days as factors


# # Compute first submission time time.max <- '2015-03-16
# 11:59:59 EDT' time.min <- '2015-03-08 12:00:00 EDT'
# time.censored <- 192 fsutime <-
# as.numeric(difftime(races.sub$timestamp, time.min ,
# unit='hours')) dat <- with(races.sub, aggregate(fsutime ~
# handle, FUN=min)) dat$fsu <- 1 dat2 <- merge(dat, races,
# by='handle', all=TRUE) dat2$fsu[is.na(dat2$fsu)] <- 0
# dat2$fsutime[is.na(dat2$fsutime)] <- time.censored surv <-
# rep() surv$m1 <- survreg(Surv(fsutime, fsu) ~ 1, dat2,
# dist='weibull', scale=1) surv$m2 <- survreg(Surv(fsutime,
# fsu) ~ treatment + mmevents, dat2 , dist='weibull',
# scale=1) surv$m3 <- survreg(Surv(fsutime, fsu) ~ treatment
# + mmevents, dat2, dist='exponential') surv$tobin <-
# survreg(Surv(fsutime, fsu, type='left') ~ treatment +
# mmevents, data=dat2, dist='gaussian') stargazer(surv,
# type='text') # A model with different baseline survival
# shapes for two groups, i.e., # two different scale
# parameters survreg(Surv(time, status) ~ ph.ecog + age +
# strata(sex), lung) # There are multiple ways to
# parameterize a Weibull distribution. The survreg # function
# embeds it in a general location-scale family, which is a #
# different parameterization than the rweibull function, and
# often leads # to confusion.  # survreg's scale =
# 1/(rweibull shape) # survreg's intercept = log(rweibull
# scale) # For the log-likelihood all parameterizations lead
# to the same value.  y <- rweibull(1000, shape=2, scale=5)
# survreg(Surv(y)~1, dist='weibull') # Economists fit a model
# called 'tobit regression', which is a standard # linear
# regression with Gaussian errors, and left censored data.
# tobinfit <- survreg(Surv(durable, durable>0, type='left') ~
# age + quant, data=tobin, dist='gaussian')
```

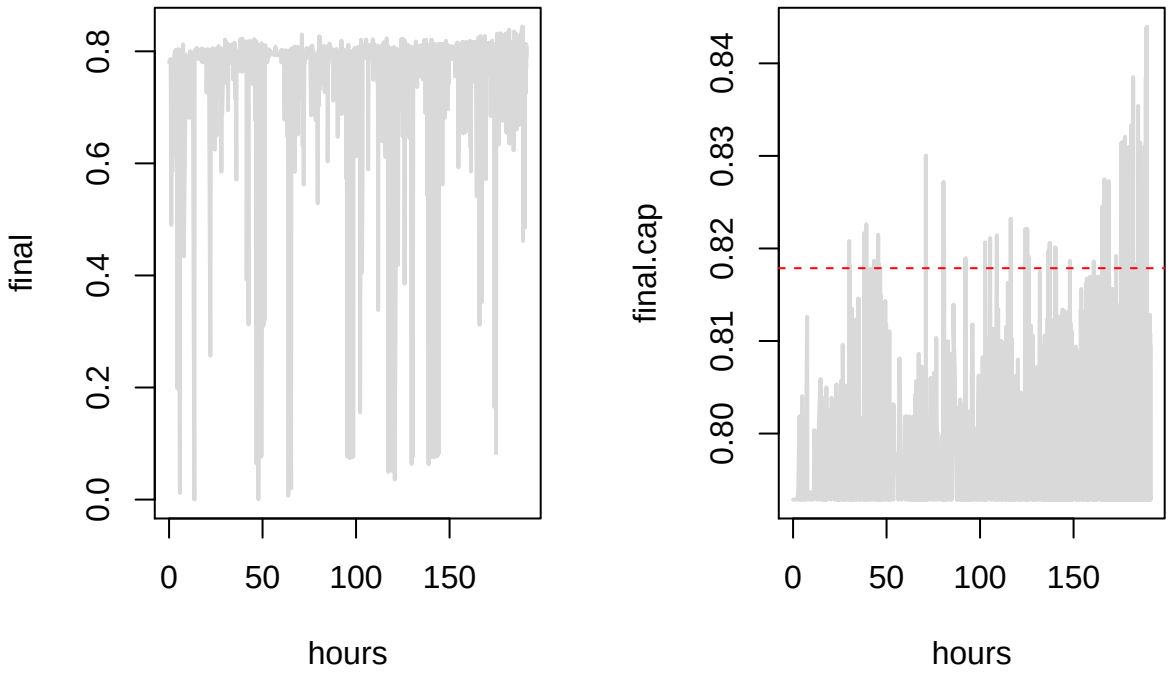
4 Analysis of scores

As a measure of the ability to make correct predictions of domain expert annotations we used the "F-score" defined as the harmonic mean of precision and recall: $F = 2 * (precision * recall)/(precision + recall)$). This score was computed on 300 abstracts (100 of which were not disculosed to avoid overfitting) with about xxxx entities to correctly identify from a dictionary with xxxx labels.

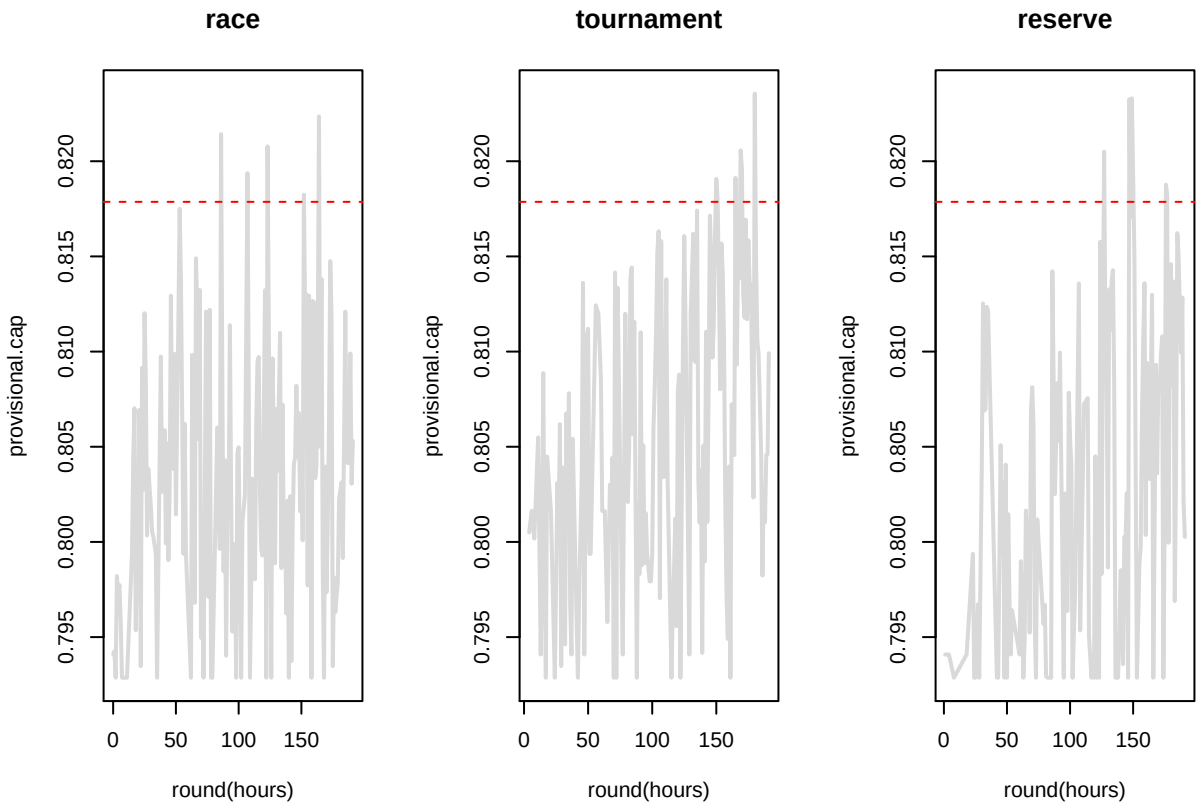
```
baseline <- 0.792867
target <- 0.817866
winner <- 0.843962
```

The baseline F-score achieved by NIH researchers was 0.792867. We set up a hard-to-reach F-score target for the race competition which was 0.817866 (about a 3 percent increase of the baseline). The winner achieved a score of 0.843962. This represents a 5.1095 percentage points increase compared to the baseline which can be regarded as a very remarkable improvement (more than 6 percent).
Figure below shows the provisional scores over the submission time. The left panel shows the raw scores that are very volitile because some preliminary submissions failed to compile or encountered other issues that either terminated the program resulting in a zero or lead to very low scores. This high volatility, however, does not reflect genuine variation in the quality of submissions. The right panel shows the scores capped from [below to reduce volatility due to experimentation with imperfect solutions].

```
# Load libraries
require(races)
require(stargazer)
# require(mgcv) require(arm)

# Load Data
data(races)
data(scores)

# Merge
dat <- merge(scores, races[, c("coder_id", "treatment", "room")],
    by = "coder_id")
dat <- dat[order(dat$timestamp), ]

attach(dat)
provisional <- provisional/1e+06
final <- final/1e+06
hours <- as.numeric(difftime(timestamp, min(timestamp), unit = "hours"))

# New variable
last_submission <- ave(submission, coder_id, FUN = max)
provisional.cap <- ifelse(provisional < baseline, baseline, provisional)
final.cap <- ifelse(final < baseline, baseline, final)


# General plot
par(mfrow = c(1, 2))
plot(aggregate(final ~ hours, FUN = mean), type = "l", lwd = 2,
    col = gray(0.85))
plot(aggregate(final.cap ~ hours, FUN = mean), type = "l", lwd = 2,
    col = gray(0.85))
abline(h = target, lty = 2, col = 2)
```

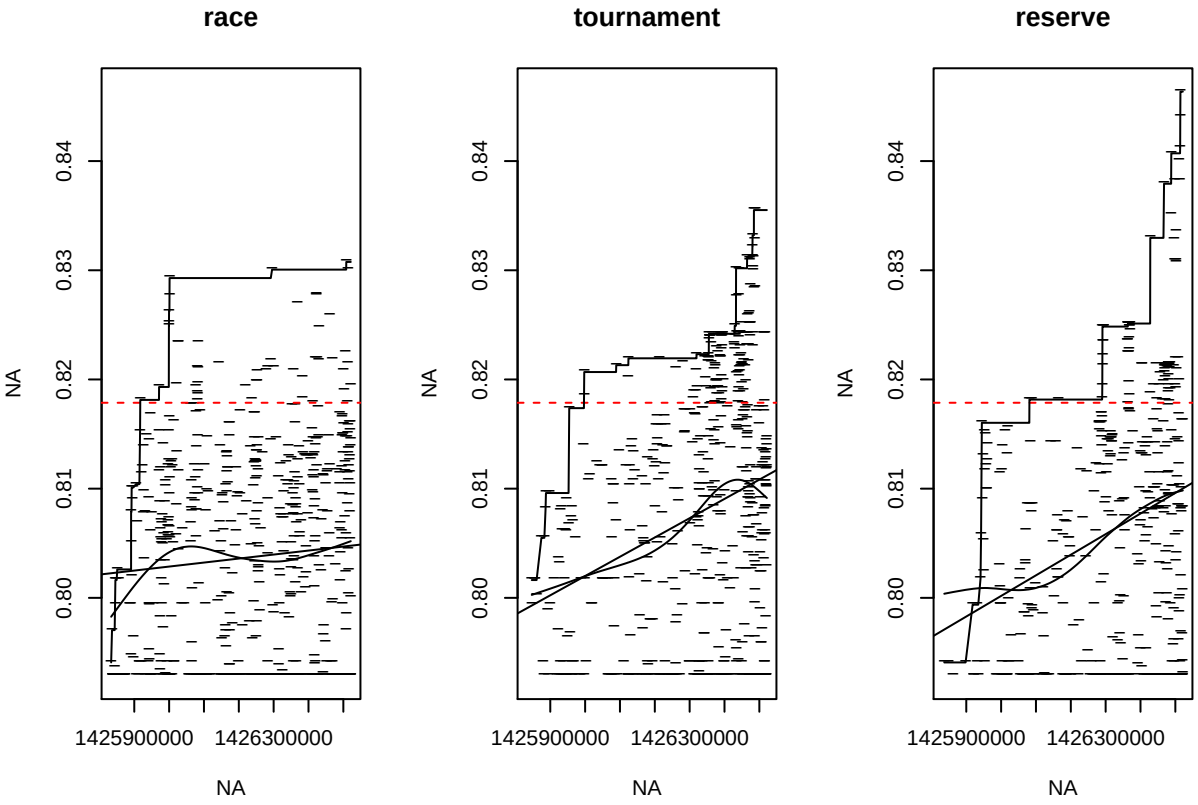



```
# Add winner dots

winners <- function() {
    out <- matrix(ncol = 2, nrow = 3)
    rownames(out) <- levels(treatment)
    colnames(out) <- c("hour", "final")
    w <- head(which(final > target & treatment == "race"), 1)
    out[1, ] <- c(hours[w], final[w])
    final.max <- tapply(final, treatment, FUN = max, na.rm = TRUE)
    w <- which(final == final.max["tournament"])
    out[2, ] <- c(hours[w], final[w])
    w <- which(final == final.max["reserve"])
    out[3, ] <- c(hours[w], final[w])
    out
}
# points(winners())

# By treatment
x <- aggregate(provisional.cap ~ round(hours) + treatment, FUN = mean)
index <- split(1:nrow(x), x[, 2])
par(mfrow = c(1, 3))
for (i in 1:3) {
    plot(x[index[[i]], -2], ylim = range(x[, 3]), type = "l",
        lwd = 2, col = gray(0.85))
    title(names(index)[i])
    abline(h = target, lty = 2, col = 2)
}
```




```
# plot(aggregate(provisional.cap ~ round(hours), FUN=mean),
# type='l', lwd=2, col=gray(.85))
# points(aggregate(provisional ~ round(hours), FUN=mean))


# By treatment
i <- split(1:nrow(dat), treatment)
xlim <- range(timestamp)
ylim <- range(provisional.cap)

max.score <- function(x) {
    n <- length(x)
    y <- numeric(n)
    for (i in 1:n) y[i] <- max(x[1:i])
    return(y)
}


plot.scores <- function(index, add = FALSE, ...) {
    x <- timestamp[index]
    y <- provisional.cap[index]
    if (add == FALSE)
        plot(NA, NA, xlim = xlim, ylim = ylim)
    points(x, y, pch = "-", ...)
    abline(h = target, lty = 2, col = 2)
    lines(smooth.spline(x, y, df = 4))
    abline(lm(y ~ x))
    lines(x, max.score(y))
}

par(mfrow = c(1, 3))
for (k in levels(treatment)) {
```

```
    plot.scores(i[[k]])
    title(k)
}
```

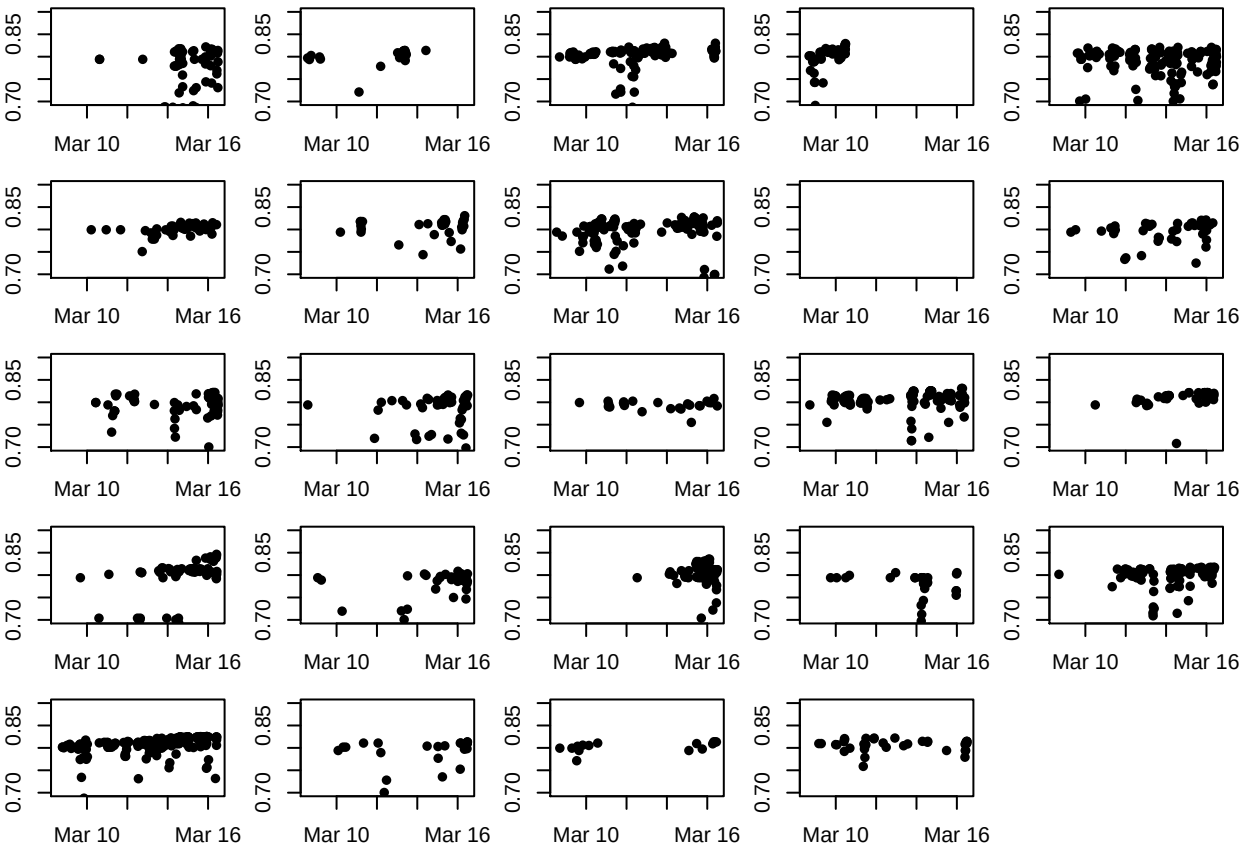



```
# submission.cap <- ifelse(submission>15, 15, submission)
# boxplot(provisional/1e6 ~ submission.cap, pch='-')
# abline(h=0.8, col=2)

# Time series
l <- split(dat, paste(treatment, room))
xlim <- range(timestamp)
ylim <- c(0.7, 0.9)
par(mfrow = c(5, 5), mar = c(2, 2, 1, 1))
sapply(l, function(x) plot(x$timestamp, x$provisional/1e+06,
    xlim = xlim, ylim = ylim, pch = 16))
```

```
## $`race 1`
## NULL
##
## $`race 2`
## NULL
##
## $`race 3`
## NULL
##
## $`race 4`
## NULL
##
```

```
## $`race 5`
## NULL
##
## $`race 6`
## NULL
##
## $`race 7`
## NULL
##
## $`race 8`
## NULL
##
## $`reserve 1`
## NULL
##
## $`reserve 2`
## NULL
##
## $`reserve 3`
## NULL
##
## $`reserve 4`
## NULL
##
## $`reserve 5`
## NULL
##
## $`reserve 6`
## NULL
##
## $`reserve 7`
## NULL
##
## $`reserve 8`
## NULL
##
## $`tournament 1`
## NULL
##
## $`tournament 2`
## NULL
##
## $`tournament 3`
## NULL
##
## $`tournament 4`
## NULL
##
## $`tournament 5`
## NULL
##
```

```
## $`tournament 6`
## NULL
##
## $`tournament 7`
## NULL
##
## $`tournament 8`
## NULL
```



5 Identification

5.1 The model

For each room ($l = 1, ..., L$), we observe the count of participants y_l among the n_l registered competitors who have been assigned to some treatment $w_l \in \{0, 1, 2\}$. The count of participating competitors is the realization of a random variable Y_l that depends on a vector of room characteristics $z_l = (x_l, w_l)$ that include the assignment w_l and additional room characteristics x_l. We assume that Y_l is distributed as a binomial variable with parameters n_l and $p(x_l, \theta)$ where $p(x, \theta)$ describes the probability for a competitor to participate in a room described by the vector x. The model is then the following:

$$Y_l \sim \text{Binomial}(N_l, p(x_l, \theta)) \quad \text{for each } l = 1, ..., L. \tag{1}$$

In binomial linear models, the probability function $p(x, \theta)$ is related to a linear prediction through a one-to-one transformation called the link function (see xxxx). Following our theoretical approach, participation is one for those with abilities above a given value:

$$Y_l = 1 \iff \text{ability} \geq m_l$$

where m_l is the marginal type for the room. Given $F_l(\cdot)$ the distribution of individual abilities in each room, the probability of entry is:

$$p(m_l, \theta) = 1 - F_l(m_l) \tag{2}$$

The marginal type is determined by the zero profit condition:

$$R_l(m_l) = m_l^{\alpha} y_0^{\beta} t_0^{\gamma} \tag{3}$$

where the $R_l(m_l)$ is the expected revenue of the marginal type in a given room and it must be equal to the costs evaluated at the deadline t_0 and the target y_0.
If we assume one single prize $q = 1$ (and we normalize the cost of meeting the deadline to one $t_0^{\gamma} = 1$) the zero profit condition becomes:

$$F_l(m_l)^{n_l - 1} = m_l^{\alpha} y_0^{\beta} \tag{4}$$

By raising to the power $1/(n_l - 1)$ both sides of the equation, we obtain a structural relationship between the probability of entry (eq. XXX) and the marginal type, the number of registered competitors, and the cost function.

$$p(x_l, \theta) = 1 - \left[m_l^{\alpha} y_0^{\beta} \right]^{1/(n_l - 1)} \qquad \text{with } \alpha \leq -1, \beta \geq 1.$$

As a result, one could use a "method of moments" to estimate the parameters $\theta = (\alpha, \beta, m_l)$ (y_0 and n_l are observed).

5.2 Estimation

First, we rewrite the probability $p(x, \theta)$ in the following way:

$$p(x_l, \theta) = 1 - \theta_1 x_l^{\theta_2}$$

where we denote the (observed) target by x_l and the vector of parameters by $\theta = (\theta_1, \theta_2)$ where $\theta_1 \equiv m_l^{\alpha/(n_l - 1)}$ and $\theta_2 \equiv \beta/(n_l - 1)$. Assume that $n_l = n$ for each $l = 1, ..., L$.
Let $Y_1, Y_2, ..., Y_L$ be the count of participants generated by our binomial-distribution model. The likelihood is:

$$\mathcal{L} = \prod_{l=1}^{L} \binom{n_l}{y_l} p(x_l, \theta)^{y_l} [1 - p(x_l, \theta)]^{n_l - y_l}$$

The log-likelihood is:

$$ll = \sum_{l=1}^{L} y_l \log(p(x_l, \theta)) + (n_l - y_l) \log(1 - p(x_l, \theta))$$

where we omit the binomial coefficient as it is a constant that does not affect estimation. Using our model into the aboev eqution:

$$ll = \sum_{l=1}^{L} y_l \log(1 - \theta_1 x_l^{\theta_2}) + (n - y_l)[\log(\theta_1) + \theta_2 \log(x_l)]$$

Let the summation be denoted by $\sum y_l = \bar{y}$. We can rewrite the ll expression:

$$ll = (Ln - \bar{y}) \log(\theta_1) + \sum_{l=1}^{L} y_l \log(1 - \theta_1 x_l^{\theta_2}) + (n - y_l)\theta_2 \log(x_l)$$

First order conditions are:

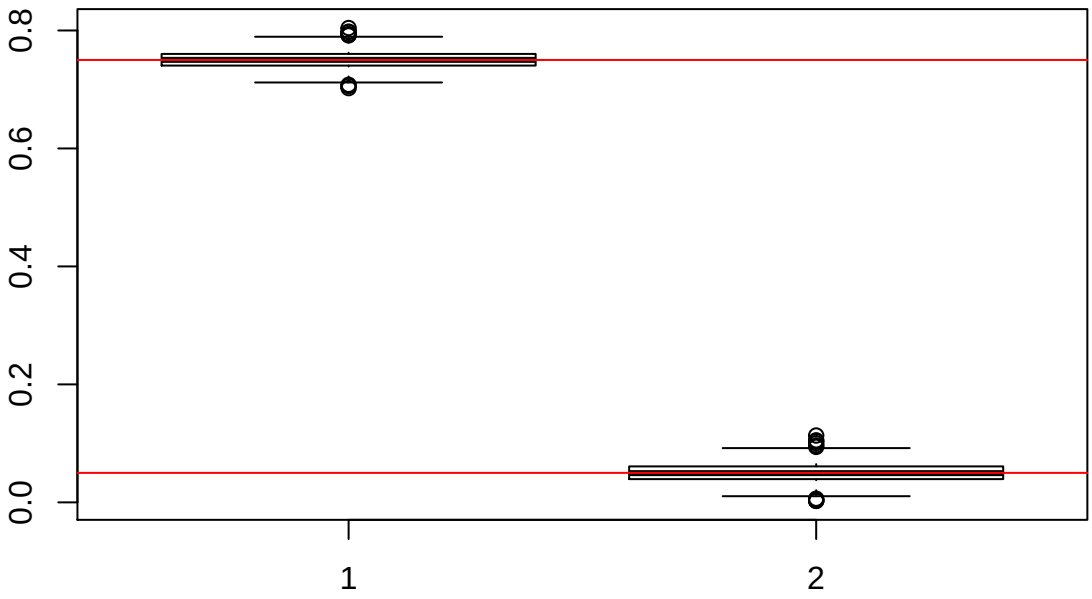
$$\frac{\partial ll}{\partial \theta_1} = \frac{(Ln - \bar{y})}{\theta_1} - \sum_{l=1}^{L} y_l \frac{x_l^{\theta_2}}{1 - \theta_1 x_l^{\theta_2}} = 0$$

$$\frac{\partial ll}{\partial \theta_2} = \sum_{l=1}^{L} \left[-y_l \frac{x_l^{\theta_2} \theta_1 \log(x_l)}{1 - \theta_1 x_l^{\theta_2}} + (n - y_l) \log(x_l) \right] = 0.$$

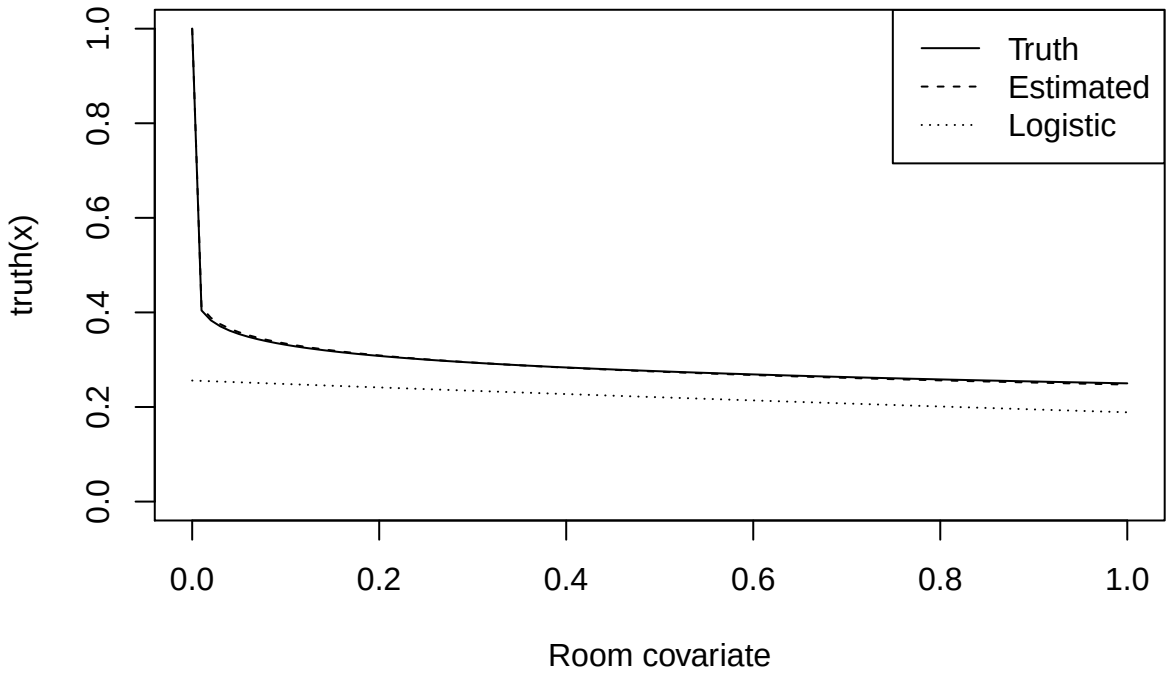
Try with simulations

```
n <- 10
nsize <- 199
theta1 <- 0.75
theta2 <- 0.05
params <- c(theta1, theta2)
x <- runif(nsize)
p <- 1 - theta1 * x^theta2
# hist(prob, 100)

loglik <- function(theta, data) {
    theta1 <- theta[1]
    theta2 <- theta[2]
    x <- data$x
    y <- data$y
    n <- data$n
    p <- 1 - theta1 * x^theta2
    out <- dbinom(y, n, p, log = T)
    return(-sum(out))
}

# Simulate data
y <- rbinom(nsize, size = n, prob = p)

# Example: loglik(c(0.5, 0.9), data.frame(y, x, n))

# MLE estimation
dat <- data.frame(y, x, n)
theta.start <- runif(2)
```

```r
mle.est <- optim(theta.start, loglik, data = dat)
thetahat <- mle.est$par

# Replications
out <- replicate(1000, {
    y <- rbinom(nsize, size = n, prob = p)
    dat <- data.frame(y, x, n)
    mle.est <- optim(theta.start, loglik, data = dat)$par
})
boxplot(t(out))
abline(h = params, col = 2)
```




```r
truth <- function(x) 1 - params[1] * x^params[2]

# Now I can 'predictions' on what would happen with different
# 'x' distribution
prediction <- function(x) 1 - thetahat[1] * x^thetahat[2]

# Compare with logistic regression
N <- rep(n, nsize)
logistic <- glm(cbind(y, N) ~ x, family = binomial(logit))
prediction.logistic <- function(x) {
    yhat <- coef(logistic)[1] + coef(logistic)[2] * x
    exp(yhat)/(1 + exp(yhat))
}

# FIGURE
curve(truth, xlab = "Room covariate", ylim = c(0, 1))
curve(prediction, add = TRUE, lty = 2)
curve(prediction.logistic(x), add = TRUE, lty = 3)
legend("topright", c("Truth", "Estimated", "Logistic"), lty = 1:3)
```


truth(x)
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
Room covariate
Truth
Estimated
Logistic

6 Identification (old)

6.1 Example with the Uniform distribution

Imagine that the distribution F_θ is uniform on $(0, \theta)$ and the cost ability $c_a(x) = 1/x$. Then the zero profit condition (3) becomes

$$\left(\frac{m}{\theta}\right)^{n-1} m = \beta x_l \tag{5}$$

Solving for the marginal type gives:

$$m = \left(\beta x_l \theta^{n-1}\right)^{1/n} \tag{6}$$

So, the probability of participation (**??**) becomes

$$\begin{aligned} p(x_l, \theta) &= 1 - F_\theta(m) && (7)\\ &= 1 - \frac{\left(\beta x_l \theta^{n-1}\right)^{1/n}}{\theta} && (8)\\ &= 1 - \left(\frac{\beta x_l}{\theta}\right)^{1/n}. && (9) \end{aligned}$$

Because it is non-linear in its parameters, the model is not identifiable [need proof]. However, we can re-parametrize the model with $\beta/\theta \equiv \eta$ to make the parameter η identifiable.

6.2 Example with Beta distribution

In this example, we assume abilities are drawn from a lognormal distribution with parameters μ and σ.

6.3 Estimation

Our aim is to estimate the parameters θ and β and to evaluate whether costs are different between the three competition modes under study: race, tournament, tournament + requirement.
It is natural to estimate the parameters θ by maximum likelihood because the ability distribution (and therefore the distribution of the outcomes) is known. The estimation criterion used here is the maximization of the *deviance*.
The deviance is:

$$D(\theta) = -2 \sum Y \log(\frac{Np(x, \theta)}{Y}) + (N - Y) \log(\frac{N - Np}{N - Y}). \tag{10}$$

Note that the deviance is a function of the likelihood (see xxx pg. xxx).

```
# Define likelihood
structreg <- function(x, y, wt = rep(1, length(y)), intercept = TRUE,
    start = rep(0, p), ...) {
    fmin <- function(beta, X, y, w) {
        p <- 1 - (beta %*% X)^(1/n)  # Function of parameters
        -sum(2 * w * ifelse(y, log(p), log(1 - p)))
    }
    # gmin <- function(beta, X, y, w) { # Gradient here!  }
    if (is.null(dim(x)))
        dim(x) <- c(length(x), 1)
```

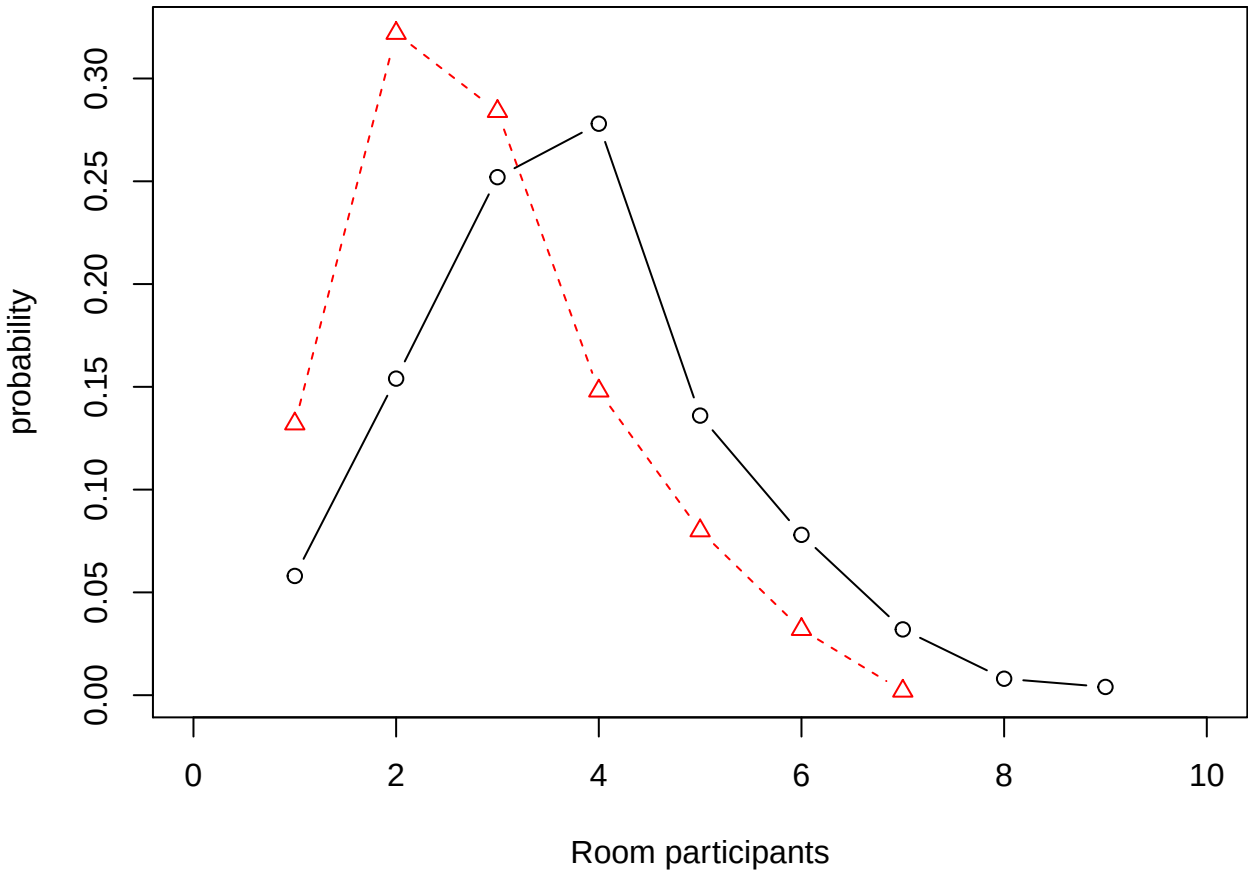


Figure 1: Simulated distribution of participants with uniform distribution. Parameters are $\eta = 0.1$ and $n = 15$. Data simulated 500 times. Dashed curve had higher costs ($x_l = 1.5$) than the solid curve ($x_l = 0.5$).

```
    dn <- dimnames(x)[[2]]
    if (!length(dn))
        dn <- paste("Var", 1:ncol(x), sep = "")
    p <- ncol(x) + intercept
    if (intercept) {
        x <- cbind(1, x)
        dn <- c("(Intercept)", dn)
    }
    if (is.factor(y))
        y <- (unclass(y) != 1)
    # fit <- optim(start, fmin, gmin, X = x, y = y, w = wt,
    # method = 'BFGS', ...)
    fit <- optim(0.5, fmin, X = x, y = y, w = wt, method = "BFGS",
        ...)
    names(fit$par) <- dn
    cat("\nCoefficients:\n")
    print(fit$par)
    cat("\nResidual Deviance:", format(fit$value), "\n")
    cat("\nConvergence message:", fit$convergence, "\n")
    invisible(fit)
}

# Create data
ilogit <- function(x) exp(x)/(1 + exp(x))
n <- 50000
competitors <- 5
x1 <- rchisq(competitors, df = 3)
x2 <- runif(n)
x3 <- rnorm(n)
X <- cbind(rep(1, length(x1)), x1, x2, x3)
betas <- c(-2.5, 0.1, 0.25, 0.5)
fc <- ilogit(X %*% betas)  ## This the fixed cost
p <- 1 - fc^(1/competitors)
y <- rbinom(n = n, size = competitors, prob = p)

# Objective function
fmin <- function(beta, X, y, w, competitors) {
    p <- 1 - ilogit(X %*% beta)^(1/competitors)  # Function of parameters
    -sum(2 * w * ifelse(y, log(p), log(1 - p)))
}

b.start <- runif(length(betas))
(out <- optim(b.start, fmin, X = X, y = y, competitors = competitors,
    w = rep(1, length(y)))$par)
```

```
## [1] -12.6957829   0.4343465   0.9204423   2.0942530
```

```
rbind(out/competitors, betas)
```

```
##             [,1]      [,2]      [,3]      [,4]
##       -2.539157 0.0868693 0.1840885 0.4188506
## betas -2.500000 0.1000000 0.2500000 0.5000000
```

6.4 More examples

Suppose F is lognormal(μ, σ), then the zero profit condition is:

$$\left[\frac{1}{2} + \frac{1}{2}\left(\frac{\log m - \mu}{\sqrt{2\sigma}}\right)\right] = a^{-1/(n-1)} K_l \tag{11}$$

```
# xxx
zeroprofit <- function(x, mu, sigma, n) {
    x * plnorm(x, meanlog=mu, sdlog=sigma)^(n-1)
}
for (i in 3:10)
    curve(zeroprofit(x, i, 10, 10), add=TRUE, lty=i)

# Clear
rm(list = ls())

# Libraries
require(xtable)
require(moments)
require(races)

# Data
data(races)
attach(races)

# TABLE 1.
size <- ave(submit, paste(treatment, room), FUN = length)
(m <- aggregate(submit ~ room + size + treatment, FUN = sum))
```

```
##    room size  treatment submit
## 1     1    9       race      2
## 2     2   10       race      2
## 3     3   10       race      5
## 4     4   10       race      1
## 5     5   15       race      4
## 6     6   15       race      3
## 7     7   15       race      4
## 8     8   15       race      5
## 9     1   10 tournament      3
## 10    2   10 tournament      5
## 11    3   10 tournament      5
## 12    4   10 tournament      2
## 13    5   15 tournament      5
## 14    6   15 tournament      4
## 15    7   15 tournament      4
## 16    8   15 tournament      5
## 17    1   10    reserve      1
## 18    2   10    reserve      3
## 19    3   10    reserve      3
## 20    4   10    reserve      2
## 21    5   15    reserve      4
## 22    6   15    reserve      6
## 23    7   15    reserve      3
## 24    8   15    reserve      5
```

7 The model

We now generalize the contest game introduced by @moldovanu2001optimal to a situation where players simultaneously decide *i*) the quality and *ii*) how fast to produce a given output. Then we explore the problem of revenue maximization faced by a contest designer with preferences for both quality and time.

7.1 Basic setup

There are $i = 1, ..., n$ players willing to compete for $k = 1, ..., q$ prizes of values $v_1 \geq v_2 \geq ... \geq v_q \geq 0$ and total value normalized to one, $\sum_{k=1}^{q} v_k = 1$.
Players observe privately an individual ability a_i, reflecting differences in skills, time constraints, and other elements affecting their quality and time in production. Then they simultaneously decide on how much time t_i to spend on a given production task, and a nonnegative quality level y_i of production.
A function $C(\cdot)$ determines the cost they incur:

$$C(a, y, t) = a^{\alpha} y^{\beta} t^{\gamma} \qquad \text{with } \alpha, \gamma < 0, \beta > 1. \tag{12}$$

Hence, costs are decreasing in a_i, increasing in q_i, and decreasing in t_i, with constant elasticities α, β, and γ, respectively. Although, alternative parametrization of the cost function (e.g., replacing time t_i with speed s_i, quality over time) can be considered as well.
Abilities are drawn at random from a common distribution function $F(\cdot)$ with density $f(\cdot)$ on a bounded interval $[\underline{a}, \bar{a}]$ with $\underline{a} > 0$.
Players are risk neutral and maximize their expected payoffs

$$\pi_i = \sum_{k=1}^{q} p_k(q_i, t_i) v_k - C(a_i, y_i, t_i) \tag{13}$$

where $p_k(\cdot)$ denotes the conditional probability of winning a prize given player i's quality y_i and time t_i.
For a given distribution of opponents' actions $(Y_1, ..., Y_{n-1}; T_1, ..., T_{n-1})$, the probability $p_k(\cdot)$ is completely specified by the rules of the contest. Here, we consider two kinds of rules: the tournament and the race competition.
Let denote a deadline by t_0 and a minimum quality target by y_0; Let denote the k'th smallest of the Y's by $Y_{(k)}$ ($Y_{(1)}$ being the smallest, $Y_{(2)}$ being the second smallest, and so on); and the k'th smallest of the T's by $T_{(k)}$ ($T_{(1)}$ being the smallest, $T_{(2)}$ being the second smallest, and so on).
A tournament competition is a contest where the player having achieved the highest quality before the deadline ($t_i \leq t_0$) gets the first prize, the player having achieved the second highest output quality before the deadline gets the second prize, and so on. Hence, using the convention that, if a player's time is above the deadline, the corresponding quality is zero, the conditional probability of winning the k'th prize in a tournament is

$$p_k(y_i, t_i \mid \text{tournament}) = p_k(y_i \mid \text{tournament}) = \Pr\left\{Y_{(n-k)} \leq y_i \leq Y_{(k)}\right\}.$$

A race competition is a contest where the first player to achieve a given minimum quality target ($y_i \geq y_0$) gets the first prize, the player being the second to achieve the target gets the second prize, and so on. Hence, using the convention that, if a player's quality is below the target, the corresponding time is infinity, player i's conditional probability of winning the k'th prize in a race is:

$$p_k(y_i, t_i \mid \text{race}) = p_k(y_i \mid \text{race}) = \Pr\left\{T_{(k)} \leq t_i \leq T_{(n-k)}\right\}.$$

In other words, races and tournaments are a special case of a general contest game but with different probabilities of winning.
The contest designer is risk neutral and cares only about the output of the winner. The contest designer wants to maximize the quality of the winner's output while keeping at a minimum the time spent by the

winner on the task (alternatively, maximizing the speed of production). Let denote the actions of the contest winner by (Y^w, T^w). The contest designer's expected payoff (net of payments to competitors) is:

$$\pi_{cd} = E[Y^w + \tau T^w \mid Y^w \geq y_0, T^w \leq t_0, \{\text{race, tournament}\}]$$

where $\tau \leq 0$ denotes the contest designer's negative preferences towards time.

7.2 Equilibrium

In this section, we solve the model for the unique symmetric Bayesian Nash equilibrium of players.

7.2.1 Tournament

At equilibrium, players choose q_i and t_i by maximizing (13) given their beliefs about the equilibrium actions of the other players.
The key to characterization of the equilibrium is that $t_i = t_0$ is a (weakly) dominant strategy for each player. Indeed, any time strictly below the deadline does not affect the probability of winning but is costly in terms of effort, and any time strictly above the deadline gives a negative payoff.
Since players should avoid playing weakly dominated strategies, we have that $t_i = t_0$ for every $i = 1, ..., n$. Then it can be shown that, at equilibrium, the optimal quality is a one-to-one transformation of a player's ability according to some "bidding" function $b(\cdot)$. Hence, $Y_i = b(A_i)$. Since the distribution of A_i is known, one can use a change of variables formula to express the probability of winning in (13) in terms of the distribution F. Then, the first order condition of π with respect to quality gives the following differential equation in a_i (for every $i = 1, ..., n$):

$$\sum_{k=1}^{q} \hat{p}'_k(\phi(y_i) \mid \text{tournament})\phi'(y_i)v_k = \beta y_i^{\beta-1} a^{\alpha} t^{\gamma}$$

with the boundary condition $y(\underline{a}) = 0$.
At equilibrium we have $a = \phi(y)$. Thus, the differential equation is separable. After rearranging, we have:

$$\sum_{k=1}^{q} \phi(y_i)^{-\alpha} \hat{p}'_k(\phi(y_i) \mid \text{tournament})\phi'(y_i)v_k = \beta y_i^{\beta-1} t^{\gamma}.$$

Integration of both sides with respect to ϕ gives:

$$\int_0^y \sum_{k=1}^{q} t^{-\gamma} v_k \phi(x)^{-\alpha} \hat{p}'_k(\phi(x) \mid \text{tournament})\phi'(x)dx = \int_0^y \beta x^{\beta-1} dx$$

Using the chain rule of derivatives

$$\int_a^b f(g(x))g'(x)dx = \int_{g(a)}^{g(b)} f(x)dx \qquad \text{(chain rule)}$$

the solution is

$$y(a_i) = \left[t_0^{-\gamma} \sum_{k=1}^{q} v_k \int x^{-\alpha} \hat{p}'_k(x)dx \right]^{1/\beta} \tag{14}$$

An important property of (14) is that $y(a_i)$ has its lower bound in zero and its upper bound in

$$y(\bar{a}) = \left[t_0^{-\gamma} v_1 \left(\frac{\bar{a}^{-\alpha+1} - \underline{a}^{-\alpha+1}}{-\alpha+1} \right) \right]^{1/\beta}.$$

- Monotonicity of the equilibrium output quality implies that, for every $i = 1, ..., n$, the equilibrium expected payoff from the contest π_i^* depends on the rank of the player's ability relative to the others.
- Equilibrium payoffs do not depend on the elasticity β or γ, but only on α.
- When abilities are distributed over the unit interval, the equilibrium payoff for the highest ability player is $\pi(\bar{a} = 1) = v_1(-\alpha)/(1 - \alpha)$.

7.2.2 Equilibrium in a race

In a similar way, one can derive the equilibrium strategy in a race. Again the key observation is that any quality below the target gives a zero probability of winning and any quality above the target gives a constant probability of winning. Thus, player i's choice of quality is either zero (with $t_i = t_0$ by convention) or $y_i = y_0$. Then, the equilibrium xxx for player i is

$$t^*(a_i) = c_\tau^{-1}\left[c_\tau(t_0) + \frac{1}{c_y(y_0)}\left(\alpha\int_{a_i}^{\bar{a}} A'(z)dz + (1-\alpha)\int_{a_i}^{\bar{a}} B'(z)dz\right)\right] \tag{15}$$

where

$$A(x) = \frac{1}{c_a(x)} f_{(n-1:n-1)}(x) \tag{16}$$

and

$$B(x) = \frac{1}{c_a(x)}\left\{\left[1 - F_{(n-1:n-1)}(x)\right] f_{(n-1:n-2)}(x) + f_{(n-1:n-1)}(x) F_{(n-1:n-2)}(x)\right\}. \tag{17}$$

An important property of XX is that $y^*(a_i)$ has its upper bound in XX and lower bound in XX. Again payoffs are xxxx. Hence, by setting to zero and solving for the ability, gives the marginal ability $\underline{a}$ as

$$\underline{a} = h(n, V, F_A, C, d). \tag{18}$$

7.2.3 Tournament vs races

By comparing equilibrium xxx and xxx, we find that the race and the tournament do not (ex-post) dominate one another with respect to output quality. Whereas the race always dominates the tournament with respect to completion time. [This is only when the deadline is the same. Otherwise, there's always xxxx.] This result is stated below.

Proposition 1. *There always exist an interval of abilities where the output quality is higher in the race than in the tournament. By contrast, every player takes less completion time in the race than in the tournament.*

Proof. Marginal type has utility zero in a race but the same type has a strictly positive utility in the tournament. Since probability of winning is not different in the race or the tournament (the bid is a monotonic transformation of the individual ability or, in other words, rankings are virtually the same), expected payoffs in equilibrium differ only in the cost functions. Hence, to be an equilibrium, the player in the tournament should bid less than the player in the race to earn a strictly positive expected payoff. □

Let's make an example.

```
p <- plnorm   # pdf individual abilities
r <- rlnorm   # Simulate individual abilities
cy <- function(x) x^2 # Cost function performance
ct <- function(x) 2*exp(1-x)  # Cost function timing
```

FIGURE 1. Equilibrium bids in a race and a tournament.
Implications. The above proposition applies only if the target is higher in a race than in a tournament. But what if the two competitions had the same target ? In that case, tournaments and races have the same marginal type. Therefore, the performance of players in the tournament with reserve are always non-lower than those in the race. This does not imply that it is optimal to set the target. On the contrary, we will show that it is optimal to set an optimal target in a tournament that is below the optimal target in a race. Next section.

7.3 The contest designer's problem

Let us now focus on the contest designer's problem. Imagine the contest designer can choose the competition format to be either the race or the tournament. Imagine all other aspects of design are given. The prize structure α has been already chosen. There is a deadline t_0, which is the same in both competition formats. [The quality requirement y_0^c in the tournament is smalle than that in the race $y_0^{\text{race}} > y_0^{\text{tournament}}$)] We will relax this assumption later to consider a more general setting where these variables are also part of the contest designer's problem.
The contest designer has an objective function that is increasing in the expected quality of the winning solution and decreasing in the corresponding time to completion. Here, to do not complicate exposition, we assume that the contest designer cares about the winning submission only: second placed efforts are not considered. [If the principal values the diversity of the solutions ... but we assume it does not.]
XXX EQUATION XXXX
The optimal choice involves a comparison of the expected profits between the race and the tournament. Given xxxx, we can show that there will be a threshold on the cost of completion time $\hat{\tau}$ above which the race is a better choice than the tournament, and vice versa.

Proposition 2. *There's a tau above which ...*

Proof. In a tournament, the objective function is

$$\begin{aligned} R_{\text{tournament}} &= \Pr(t_{(1:n)} \leq t_0) \left\{ \int y^*(x \mid t_{(1:n)} \leq t_0) dF_{n:n}(x) - \tau t_0 - 1 \right\} \\ &= \int_{\hat{a}}^{\bar{a}} y^*(x) dF_{n:n}(x) - \tau t_0 - 1. \end{aligned} \tag{19}$$

That is, the contest designer's objective function is the sum of the expected output quality for a given deadline, minus the cost τ of having the winner working on the task until completion (i.e., until the deadline), and the cost of the prize pool (recall the prize pool is normalized to one).
[Implicitly, you're assuming that the prize is always large enough to ensure positive effort.] [Second prize too is stochastic!!!!]
In a race, the objective function is

$$\begin{aligned} R_{\text{race}} &= \Pr(a_{(N)} \geq \hat{a}) \left\{ y_0 - \alpha - \Pr(a_{(N-1)} \geq \hat{a})(1-\alpha) \right\} - \tau \int_{\hat{a}}^{\infty} t^*(x) dF_{N:N}(x) \\ &= [1 - F_{N:N}(\hat{a})] \left\{ y_0 - \alpha - [1 - F_{N-1:N}(\hat{a})](1-\alpha) \right\} - \tau \int_{\hat{a}}^{\infty} t^*(x) dF_{N:N}(x). \end{aligned} \tag{20}$$

Note. $t^*(x) \leq t_0$ for all x's. Thus, a lower bound for the above objective function can be computed:

$$\underline{R_{\text{race}}} = [1 - F_{N:N}(\hat{a})] \left\{ y_0 - \alpha - [1 - F_{N-1:N}(\hat{a})](1-\alpha) - \tau t_0 \right\} \tag{21}$$

An even simpler lower bound is rewriting the above expression as if $\alpha = 1$ (note if the real alpha was set 1 then also mtype would change and therefore setting alpha hits a lower bound only when mtype does xxxx when alpha is 1).
Note. $y^*(x)$ is lower than y_0 for all $a < \hat{a}$. Thus, a lower bound of the tournament's expression is

$$\overline{R_{\text{tournament}}} = [1 - F_{N:N}(\hat{a})]y_0 + \int_{\hat{a}}^{\infty} y^*(x) dF_{N:N}(x) - \tau t_0 - 1. \tag{22}$$

$$\begin{aligned}
\underline{R_{\text{race}}} \geq & \overline{R_{\text{tournament}}} \\
[1 - F_{N:N}(\hat{a})](y_0 - 1 - \tau t_0) \geq & [1 - F_{N:N}(\hat{a})]y_0 + \int_{\hat{a}}^{\infty} y^*(x) dF_{N:N}(x) - \tau t_0 - 1 \\
-[1 - F_{N:N}(\hat{a})](\tau t_0 + 1) \geq & \int_{\hat{a}}^{\infty} y^*(x) dF_{N:N}(x) - (\tau t_0 + 1) \\
F_{N:N}(\hat{a})(\tau t_0 + 1) \geq & \int_{\hat{a}}^{\infty} y^*(x) dF_{N:N}(x) \\
\tau \geq & \left[\frac{\int_{\hat{a}}^{\infty} y^*(x) dF_{N:N}(x)}{F_{N:N}(\hat{a})} - 1 \right] \frac{1}{t_0}
\end{aligned} \tag{23}$$

End proof.
When the cost of time τ is sufficiently high, the race is preferred. Interestingly, the threshold is a function of the deadline to complete the job, as xxx. It also depends on the shape of xxxx.

7.3.1 Optimal minimum-entry

Now we turn to discuss the contest designer's choice of an optimal minimum requirement y_0. So far, we have assumed that $y_0^{\text{race}} > y_0^{\text{tournament}}$. Now, we show that the assumption that xxxx is indeed an optimal choice of the contest designer. This is summarized in the next proposition.

Proposition 3. *Suppose the contest designer can choose the target that max profits in both the race and the tournament. Then, the optimal y_0 in tournament is generally lower than that in a race.*

To prove that it is indeed the case. We proceed in two steps. First, we assume that the contest designer does not care about minimizing the timing of the innovation by imposing $\tau = 0$. For simplicity, assume that $\alpha = 1$ (winner-takes-all). In a race, this means that the optimal target will be a value that makes equal the costs in terms of less participation versus the gains in terms of higher values of the winning solutions. Formally, the contest designer's problem in a race is

$$\text{maximize } R^{\text{race}} = [1 - F_{N:N}(\hat{a})](y_0^{\text{race}} - 1). \tag{24}$$

Note that $\hat{a}$ depends on the target. This is clearly concave in y_0^{race}. Thus, the first order condition is also sufficient.

$$\text{FOC } \Rightarrow -F'_{N:N}(\hat{a})\hat{a}'(y_0^{\text{race}} - 1) + [1 - F_{N:N}(\hat{a})] = 0. \tag{25}$$

In a tournament, ...

$$\text{maximize } R^{\text{race}} = \int_{\hat{a}}^{\infty} y^*(x, y_0) dF_{N:N}(x) - [1 - F_{N:N}(\hat{a})]. \tag{26}$$

Convexity is not sure. If not, then the optimal target is zero. Which is lower than the optimal target in a race.
Instead. If the objective function is (strictly) concave then there's an internal solution.

$$\begin{aligned}\text{FOC } \Rightarrow & \frac{d\int_{\hat{a}}^{\infty} y^*(x,y_0)dF_{N:N}(x))}{dy_0} + F'_{N:N}(\hat{a})\hat{a}' = 0 \\ & \text{(by using Leibniz rule)} \\ \Rightarrow & -y^*(\hat{a},y_0)\hat{a}'F'_{N:N}(\hat{a}) + \int_{\hat{a}}^{\infty} \frac{\partial y^*(x,y_0)}{\partial y_0} dF_{N:N}(x) - F'_{N:N}(\hat{a})\hat{a}' = 0 \\ \Rightarrow & -y_0\hat{a}'F'_{N:N}(\hat{a}) + \int_{\hat{a}}^{\infty} \frac{\partial y^*(x,y_0)}{\partial y_0} dF_{N:N}(x) - F'_{N:N}(\hat{a})\hat{a}' = 0. \end{aligned} \tag{27}$$

Using (25) with (27), the optimal target is the same in the race and the tournament only if

$$\int_{\hat{a}}^{\infty} \frac{\partial y^*(x,y_0)}{\partial y_0} dF_{N:N}(x) = [1 - F_{N:N}(\hat{a})]. \tag{28}$$

$$\frac{\partial y^*(x,y_0)}{\partial y_0} = \frac{c'_y(y_0)}{c'_y(y^*(x,y_0))}.$$

Then.

- If $c_y(\cdot)$ is linear, we have that the ratio is one for all x.
- If $c_y(\cdot)$ is convex, then we have that it is less than one. If
- If $c_y(\cdot)$ is concave, then we have that it is higher than one.

As a result, if linear or convex the first order condition is lower than that in the race. Since the obj. function is concave (second order is decreasing), the target should be lower in a tournament than in a race to satisfy the first order condition. (a lower target increases the focs.).
Conjecture. If $\tau > 0$, the y_0 in the race is higher.

7.4 Structural econometric model

Readings:

- The winner's curse, reserve prices, and endogenous entry: Empirical insights from eBay auctions
- Entry and competition effects in first-price auctions: theory and evidence from procurement auctions
- Auctions with entry

General two-step strategy:

- First step. Identify the marginal type from the data and the distribution of types.
- Second step. Using the estimated distribution of types.

Basic idea. Equilibrium condition gives:

$$y_i^* = y^*(a_i; F_{\mathcal{A}}). \tag{29}$$

with $y^*(\cdot)$ being an invertible function with ϕ denoting the inverse.

Hence the distribution of bids is

$$F_Y(y) = \Pr(y_i^* \leq y) = \Pr(y^*(a_i) \leq y) = \Pr(a_i \leq \phi(y)) = F_{\mathcal{A}}(\phi(y)). \tag{30}$$

Identification of the model. suggest
data(scores) dat <- merge(scores, races[, c("coder_id", "treatment")], by='coder_id') attach(dat)
time_l <- split(timestamp, coder_id)
difftime2 <- function(x) { n <- length(x) init <- as.POSIXct("2015-03-08 12:00:00 EDT") y <- x - c(init, x[-n]) units(y) <- "hours" return(as.numeric(y)) }
interval <- unlist(tapply(timestamp, coder_id, FUN=difftime2)) boxplot(interval ~ treatment, outline=FALSE)

8 difference of location

sapply(interval_l, mean) sapply(interval_l, median)

9 test difference

interval_l <- split(interval, treatment) kruskal.test(interval_l) # Significant!!!!

10 Tobit

10.1 Theory

In the standard tobit model [@amemiya1985advanced], we have:

$$y_i = \begin{cases} y_l & \text{if } y_i^* < y_l \\ y_i^* & \text{if } y_i^* \geq y_l \end{cases}$$

Where $y^* = x\beta + \epsilon$, assuming disturbance $\epsilon \sim N(0, \sigma^2)$
Let denote the standard normal distribution by $\Phi(\cdot)$ and the density function by $\phi(\cdot)$. The likelihood is:

$$\mathcal{L} = \prod_i^n \left(\sigma^{-1}\phi\left(\frac{y_i - X_i\beta}{\sigma}\right)\right)^{I(y_i)} \left(1 - \Phi\left(\frac{X_i\beta - y_l}{\sigma}\right)\right)^{1-I(y_i)}$$

where $I(\cdot)$ is an indicator function that is 1 when $y_i > y_l$ and 0 otherwise.

10.2 Application

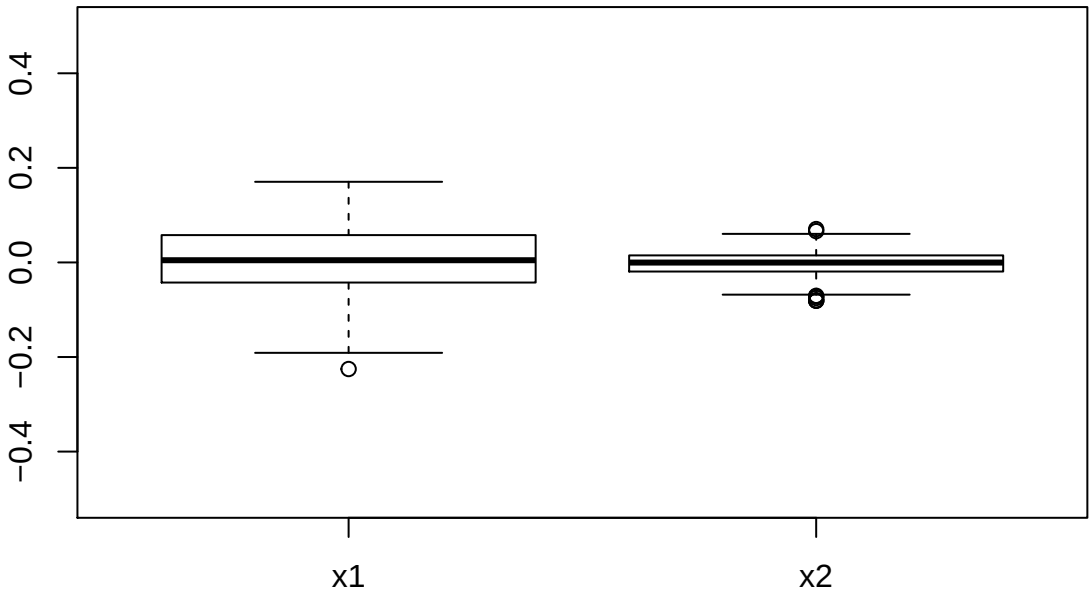
```
require(AER)

# Example with left censoring
n <- 1000
yl <- 0.1
x1 <- runif(n)
x2 <- rchisq(n, 1)
betas <- c(0.25, -0.1)
simulate.tobit <- function(n, betas, yl, x1, x2) {
    ystar <- rnorm(n, mean = cbind(x1, x2) %*% betas)
    y <- ifelse(ystar < yl, yl, ystar)
    return(y)
}

# Try
y <- simulate.tobit(n, betas, yl = 0.1, x1, x2)
summary(tobit(y ~ x1 + x2 - 1, left = yl))
```

```
##
## Call:
## tobit(formula = y ~ x1 + x2 - 1, left = yl)
##
## Observations:
##          Total  Left-censored     Uncensored Right-censored
##           1000            540            460              0
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## x1           0.13607    0.07625   1.785  0.07432 .
## x2          -0.07063    0.02374  -2.975  0.00293 **
## Log(scale)  0.03199    0.03575   0.895  0.37082
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Scale: 1.033
##
```

```
## Gaussian distribution
## Number of Newton-Raphson Iterations: 3
## Log-likelihood: -1018 on 3 Df
## Wald-statistic: 9.351 on 1 Df, p-value: 0.0022287
```

```
# Simulations
coef.sim <- replicate(499, {
    y <- simulate.tobit(n, betas, yl, x1, x2)
    coef(tobit(y ~ x1 + x2 - 1, left = yl)) - betas
})
```

```
# Consistency looks good
boxplot(t(coef.sim), ylim = 0.5 * c(-1, 1))
```




```
# Inference looks good
apply(coef.sim, 1, sd)
```

```
##         x1         x2
## 0.07102170 0.02557579
```