[gtree](http://feedproxy.google.com/~r/skranz_R/~3/sgXZRuJZC8g/skranz.github.io/gtree" \t "_blank) is a new R package that allows to specify extensive form games using stages, similar as one specifies economic experiments with [ztree](https://www.ztree.uzh.ch/en.html) or [otree](https://otree.readthedocs.io/en/latest/).

Let us specify a simple ultimatum game in gtree. A proposer can offer the responder between 0 and 10 Euros. If the responder accepts the offer, the proposer keeps 10 Euro minus the proposed amount. If the responder rejects, both get nothing.

library(gtree)

game = new\_game(

gameId = "Ultimatum\_Dictator",

options = make\_game\_options(verbose=FALSE),

params = list(numPlayers=2,cake=10),

stages = list(

stage("proposerStage",

player=1,

actions = list(

action("offer",~0:cake)

)

),

stage("responderStage",

player=2,

observe = c("offer"),

actions = list(

action("accept",c(FALSE,TRUE))

)

),

stage("payoffStage",

compute=list(

payoff\_1 ~ ifelse(accept, cake-offer,0),

payoff\_2 ~ ifelse(accept, offer,0)

)

)

)

)

The following code solves for all pure strategy subgame perfect equilibria and shows the outcomes.

game %>%

game\_solve() %>%

eq\_outcomes() %>%

select(eqo.ind,offer, accept, payoff\_1, payoff\_2)

## # A tibble: 2 x 5

## eqo.ind offer accept payoff\_1 payoff\_2

##

## 1 1 1 TRUE 9 1

## 2 2 0 TRUE 10 0

We have two different equilibrium outcomes: the proposer either offers 0 or 1 and in both equilibrium outcomes the offer will be accepted.

The function eq\_tables() also shows us the behavior off the equilibrium path:

game %>% eq\_tables()

## $offer

## # A tibble: 2 x 2

## offer eq.inds

##

## 1 0 2

## 2 1 1

##

## $accept

## # A tibble: 3 x 2

## accept eq.inds

##

## 1 FALSE 1

## 2 TRUE 2

## 3 TRUE 1,2

We see that all offers above 0 are always accepted, and the two equilibria differ by whether an offer of 0 is accepted or rejected.

In behavioral economics the ultimatum game is often used to illustrate that many people have *social preferences* that go beyond the maximization of own payoffs only. One simple form of social preferences is [inequality aversion](https://en.wikipedia.org/wiki/Inequity_aversion) (Fehr & Schmidt, 1999) The following code solves the game assuming inequality averse preferences with an envy parameter of alpha=0.5 but no guilt (beta=0). This means the effective utility function of both players are given by

`u\_1 = payoff\_1 - 0.5\*max(payoff\_2-payoff\_2,0)`

`u\_2 = payoff\_2 - 0.5\*max(payoff\_1-payoff\_1,0)`

game %>%

game\_set\_preferences(pref\_ineqAv(alpha=0.5, beta=0)) %>%

game\_solve() %>%

eq\_tables()

## $offer

## # A tibble: 1 x 2

## offer eq.inds

##

## 1 3 1

##

## $accept

## # A tibble: 11 x 3

## # Groups: offer [11]

## offer accept eq.inds

##

## 1 0 FALSE 1

## 2 1 FALSE 1

## 3 2 FALSE 1

## 4 3 TRUE 1

## 5 4 TRUE 1

## 6 5 TRUE 1

## 7 6 TRUE 1

## 8 7 TRUE 1

## 9 8 TRUE 1

## 10 9 TRUE 1

## 11 10 TRUE 1

Now we have a unique equilibrium in which the proposer offers 3 and every offer below 3 would be rejected.

The internal gtree solver used above is quite limited. It can only compute pure strategy subgame perfect equilibria. To access a much larger set of game theoretic solvers, one can use gtree together with the [Gambit](http://www.gambit-project.org/) command line solvers. The following code uses the [gambit-logit](https://gambitproject.readthedocs.io/en/latest/tools.html#gambit-logit-compute-quantal-response-equilbria) solver to compute a [logit quantal response equilibrium](https://en.wikipedia.org/wiki/Quantal_response_equilibrium) (QRE) for the ultimatum game using the previously set inequality aversion preferences:

game %>%

game\_gambit\_solve(qre.lambda=2) %>%

eq\_tables()

## $offer

## # A tibble: 9 x 3

## # Groups: offer [9]

## offer .prob eq.inds

##

## 1 0 0.00000116 1

## 2 1 0.00000120 1

## 3 2 0.00000718 1

## 4 3 0.602 1

## 5 4 0.353 1

## 6 5 0.0438 1

## 7 6 0.000646 1

## 8 7 0.00000952 1

## 9 8 0.000000140 1

##

## $accept

## # A tibble: 20 x 4

## # Groups: offer, accept [20]

## offer accept .prob eq.inds

##

## 1 0 FALSE 1.000 1

## 2 0 TRUE 0.0000264 1

## 3 1 FALSE 0.998 1

## 4 1 TRUE 0.00179 1

## 5 2 FALSE 0.892 1

## 6 2 TRUE 0.108 1

## 7 3 FALSE 0.108 1

## 8 3 TRUE 0.892 1

## 9 4 FALSE 0.00179 1

## 10 4 TRUE 0.998 1

## 11 5 FALSE 0.0000264 1

## 12 5 TRUE 1.000 1

## 13 6 FALSE 0.00000320 1

## 14 6 TRUE 1.000 1

## 15 7 FALSE 0.000000388 1

## 16 7 TRUE 1 1

## 17 8 FALSE 0.0000000471 1

## 18 8 TRUE 1 1

## 19 9 TRUE 1 1

## 20 10 TRUE 1 1

We see how player 1 now makes all offers with positive probabilities but mainly concentrates on offers 3,4 and 5. Interestingly, an offer of 2 is still very unlikely and will be rejected with very high probability.

Gambit has an own [Python interface](https://gambitproject.readthedocs.io/en/latest/pyapi.html) to directly construct the game tree of extensive form games. While it is a matter of taste, I think for many games studied in economic experiments it is simpler to specify stages instead of directly constructing a game tree. For a comparision you can look at a [gtree implementation](https://skranz.github.io/gtree/articles/t04_sender_receiver.html) of a sender-receiver game whose [Gambit Python interface implementation](https://nbviewer.jupyter.org/github/gambitproject/gambit/blob/master/contrib/samples/sendrecv.ipynb) is linked as example on the Gambit main page.

Take a look at the gtree page <https://skranz.github.io/gtree> for more examples and features. On Github you can find gtree here: <https://github.com/skranz/gtree>. Please open an issue if you have some feature request, bug report or question concerning gtree.

**Playing games in the web: Rock Paper Scissors Laser Mirror**

[gtreeWebPlay](https://skranz.github.io/gtreeWebPlay/) is a companion package to gtree that helps building simply shiny apps that a allow to play a game. Below I embedded an app to play Rock-Paper-Scissors-Laser-Mirror:

* Laser beats rock, paper and scissors.
* Mirror beats laser but is beaten by rock, paper and scissors.
* Otherwise as usual.

You play agains the population of all previous users. (And the first users play against the equilibrium strategies.) Let us see whether you can beat the crowd:

**Kuhn Poker**

Below is a gtreeWebPlay app that allows you to play a round of [Kuhn Poker](https://en.wikipedia.org/wiki/Kuhn_poker).  
(<https://skranz.github.io/gtree/articles/t03_kuhn_poker.html>). The deck has only three cards: *Jack*, *Queen* and *Ace* and two players who each get one card after they both put one dollar into the blind. Below is an embedded app that allows you to play it against a randomly chosen earlier player:

A game theoretic analysis of Kuhn poker using gtree can be found in [this tutorial](https://skranz.github.io/gtree/articles/t03_kuhn_poker.html)

