DFG Research Group 2104
Need-Based Justice and Distribution Procedures

Ein Programm zum Studium des Baron-Ferejohn-Modells, insbesondere unter Berücksichtigung von Bedarfen

Nils Springhorn

Working Paper Nr. 2025-01

July 2025

DFG Research Group 2104 at Helmut Schmidt University Hamburg
https://www.hsu-hh.de/bedarfsgerechtigkeit/

Ein Programm zum Studium des Baron-Ferejohn-Modells, insbesondere unter Berücksichtigung von Bedarfen

Nils Springhorn*

Abstract: Baron und Ferejohn haben in *Bargaining in Legislature* (1989) theoretische Lösungen für bestimmte Verhandlungssituation bestimmt. Ihr Modell ist weithin akzeptiert, wurde mannigfaltig untersucht und weiterentwickelt und hat einen großen Einfluss auf die Forschung. Bisher wird aber nicht berücksichtigt, dass Verhandlungsteilnehmer:innen in bestimmten Situationen den theoretisch bestimmten Lösungen nicht zustimmen können, weil sie einen Bedarf an der zu verteilenden Ressource haben, der größer ist als die für sie theoretisch bestimmten Zuteilung (besonders deutlich wird das im Fall von Bedarfen, deren Deckung überlebensnotwendig ist). Ich habe mit entsprechenden Untersuchungen begonnen (Springhorn 2021). In diesem Zusammenhang ist das Programm entstanden, das hier vorgestellt wird. Es dient der Darstellung und Untersuchung von Verhandlungsverläufen, die auf dem Baron-Ferejohn- Modell basieren und Bedarfe berücksichtigen.

---

*Carl von Ossietzky University Oldenburg, Department of Social Sciences, Uhlhornsweg 82, 26129 Oldenburg, Germany, n.springhorn@uol.de. The author is member of the research group “Need-Based Justice and Distributive Procedures” (FOR 2104) funded by the German Research Foundation (DFG Grant TE 1022/2-2).

Einleitung

Das Programm dient dazu, Verhandlungsverläufe, die dem Baron-Ferejohn-Modell (kurz: BF-Modell; Baron and Ferejohn 1989) oder einer seiner Erweiterungen oder Spielarten folgen, zu simulieren und zu untersuchen.

Dieser Artikel dient primär der Beschreibung des Programms. Dem vorangestellt ist ein Kapitel mit einer sehr kurzen Zusammenfassung des ursprünglichen BF-Modells und der Erweiterungen durch Springhorn (2021). Für weitere Informationen, den theoretischen Hintergrund, die Relevanz, die Motivation und mögliche Anwendungsgebiete sei der Leser neben dem Artikel von Baron und Ferejohn, in dem das ursprüngliche Modell vorgestellte wird, auf Artikel verwiesen, die die mannigfaltigen Weiterentwicklungen und die vielen empirischen Studien, die in diesem Zusammenhang durchgeführt wurden, zusammenfassen (wie zum Beispiel von Baranski und Morton (2020) oder Miller, Montero, and Vanberg (2018)) und insbesondere auf die Artikel von Springhorn, die den Anlass gegeben haben, dieses Programm zu schreiben (Springhorn 2021 und 2024).

Das Programm liefert zwar Lösungen für Verhandlungssituation, im Allgemeinen wird es aber nicht notwendig sein, dafür das Programm zu nutzen. Die Stärke des Programms liegt in der theoretischen Untersuchung von Verhandlungsverläufen: Es kann einem besseren Verständnis der bestehenden Strategien dienen, vor allem aber hilft es bei dem Verständnis und der Untersuchung von Weiter- und Neuentwicklungen. Dabei helfen die eingebauten Parametrisierungsmöglichkeiten; das Programm ist aber auch so angelegt, dass weitere, neue Strategien mit geringen Programmierkenntnissen implementiert werden können.

Das Baron-Ferejohn-Modell und zwei Modelle, die Bedarfe im BF-Modell berücksichtigen

Das Baron-Ferejohn-Modell

Das grundsätzliche Setting des BF-Modells sieht so aus, dass mehrere Spieler[1] in potenziell mehreren Runden über die Verteilung einer Ressource verhandeln. Einer der Spieler wird zufällig und mit gleicher Wahrscheinlichkeit gewählt, um einen Verteilungsvorschlag zu machen. Dieser Vorschlag wird den anderen Spielern zur Abstimmung vorgelegt (die Spieler stimmen über den Verteilungsvorschlag als Ganzes ab, nicht nur über die Zuteilung, die der Vorschlag für den stimmberechtigten Teilnehmer vorsieht). Stimmt eine einfache Mehrheit für den Vorschlag (die Zustimmung des Vorschlagenden wird vorausgesetzt), kommt eine Einigung zustande, die Ressource wird entsprechend des Vorschlags verteilt und die Verhandlung endet. Andernfalls geht es in die nächste Runde, die genauso verläuft. Wieder wird zufällig und mit gleicher Wahrscheinlichkeit ein Spieler gewählt, der einen Vorschlag macht und es wird darüber abgestimmt. Wenn die Anzahl der Runden endlich ist und es in der letzten Runde keine Mehrheit gibt, gehen alle Spieler leer aus. Baron und Ferejohn nehmen an, dass die Spieler risikoneutral sind und daher einem Verteilungsvorschlag – qua Definition – zustimmen, wenn dieser für sie eine Zuteilung vorsieht, die mindestens so groß ist wie ihr Erwartungswert[2]. Sie zeigen, dass die folgende

[1] In den (semi-)formalen Beschreibungen wird die männliche Form generisch verwendet, um die Lesbarkeit zu erleichtert. Der/die Leser:in möge die Bezeichnungen als Variablen verstehen.

[2] Baron und Ferejohn nutzen das als Argument dafür, dass Spieler in der letzten Runde einem Verteilungsvorschlag zustimmen würden, der dem Vorschlagenden alles zuspricht und allen anderen nicht. Springhorn kritisiert das in „Capitulate for Nothing? Does Baron and Ferejohn's Bargaining Model Fail Because No One Would Give Everything for Nothing?“ (Springhorn 2024), zeigt aber auch, dass die Lösung von Baron und Ferejohn auch unter plausiblen Annahmen hält.

Verhandlungslösung bereits in der ersten Verhandlungsrunde eine Mehrheit erreicht: Der Vorschlagende bietet genau so vielen Spielern, wie er für eine Mehrheit benötig, ein n-tel der Ressource an und behält den Rest für sich.

Das Baron-Ferejohn-Modell unter Berücksichtigung von Bedarfen

Springhorn stellt die Frage, wie die Verhandlungen aussehen, wenn davon ausgegangen wird, dass die Spieler individuell unterschiedliche Zustimmungsuntergrenzen haben, insbesondere, wenn sie einen Bedarf an der Ressource haben (im Extremfall einen Bedarf, dessen Deckung überlebensnotwendig ist), so dass sie nicht zustimmen können, wenn die von Baron und Ferejohn bestimmte Zuteilung an sie diese Grenze unterschreitet. Zu diesem Zweck wird für jeden Spieler ein „minimaler Zustimmungswert" (minimum approval value) eingeführt: Ein Wert, der angibt, wie klein die Zuteilung an einen Spieler ausfallen darf, damit der Spieler einem Verteilungsvorschlag zustimmt (im BF-Modell ist das der Erwartungswert).

Der erste diesbezügliche Ansatz – der genauere Überlegungen vorwegnehmend als „naiv" bezeichnet wird – sieht so aus, dass das Maximum von Erwartungswert (wie in dem Ansatz von Baron und Ferejohn) und dem (individuellen) Bedarf als minimaler Zustimmungswert herangezogen wird. Ein Spieler stimmt einem Verteilungsvorschlag demnach zu, wenn ihm mindestens sein Erfahrungswert und mindestens sein Bedarf angeboten wird. Auf den ersten Blick scheint es plausibel, anzunehmen, dass damit Spieler mit großem Bedarf tendenziell benachteiligt sind: Ihr minimaler Zustimmungswert ist aufgrund des großen Bedarfs groß – ihre Zustimmung ist für den Vorschlagenden „teuer", so dass dieser es vorzieht, Spielern mit kleinem Bedarf Angebote zu machen. Es ist eines der interessantesten Ergebnisse der Untersuchungen (und an dieser Stelle wurde das Programm geboren), dass dem nur bedingt so ist: Denn genau diese vermeintlich plausible Annahme spiegelt sich für Spieler mit kleinem Bedarf in ihren Erwartungswerten wider – sie erwarten, in der nächsten Runde bevorzugt zu werden. Der Erwartungswert von Spielern mit kleinem Bedarf kann daher so groß sein, dass er das Maximum von Bedarf und Erwartungswert von Spielern mit großem Bedarf übertrifft – zumal der Erwartungswert von Spielern mit großem Bedarf aus dem gleichen Grund klein ist. Die Zustimmung eines Spielers mit großem Bedarf kann daher für den Vorschlagenden „günstiger" sein als die Zustimmung eines Spielers mit kleinem Bedarf.

Dabei wird allerdings nicht berücksichtigt, dass ein Spieler mit kleinem Bedarf prinzipiell die Möglichkeit hat, Angebote, die unter seinem Erwartungswert liegen, zu akzeptieren (sofern sie über seinem Bedarf liegen). Vor die Wahl gestellt, nur Angebote zu akzeptieren, die mindestens so groß sind, wie der eigene Erwartungswert und damit leer auszugehen und den eigenen Bedarf nicht zu decken oder auch Angebote zu akzeptieren, die kleiner sind als der eigene Erwartungswert – aber mindestens so groß wie der eigene Bedarf – und damit nicht nur nicht leer auszugehen, sondern auch den eigenen Bedarf zu decken, dürfe die Wahl auf letztere Strategie fallen. Damit hängt der minimale Zustimmungswert aber nicht mehr nur vom eigenen Erwartungswert (wie im BF-Modell) und nicht mehr nur vom eigenen Erwartungswert und dem eigenen Bedarf (wie im naive approach) ab, sondern auch von den Erwartungswerten und Bedarfen aller konkurrierender Spieler: Es gilt – sofern der eigene Bedarf das zulässt – gegebenenfalls so viele konkurrierende Spieler (minimal) zu „unterbieten", dass man in der Gruppe der „günstigsten" Spieler ist. Damit nimmt die Komplexität gegenüber dem ursprünglichen BF-Modell und dem naiven Ansatz deutlich zu, zumal ähnliche Überlegungen auch für den Vorschlagenden gelten. Die entsprechenden Strategien und Lösungen werden von Springhorn unter dem Namen „improved approach" behandelt.

Im folgenden Kapitel finden sich nummerische Beispiele für die verschiedenen Ansätze.

Beschreibung des Programms – Input und Output

Das Programm ist in R geschrieben (R Core Team 2025), es nutzt die packages tidyverse (Wickham et al. 2019) und openxlsx (Schauberger and Walker 2025). Es gliedert sich in die Abschnitte "TECHNICAL STUFF", "ADMINISTRATION", "CHECKS", "PREPARATIONS", "FUNCTIONS", "PROGRAM", "EXAMPLES". Für den/die Nutzer:in relevant sind die Abschnitte „ADMINISTRATION“ und „PROGRAM“. Unter „ADMINISTRATION“ gibt es die Unterabschnitte „technical“ und „concerning the bargaining“.

Unter „technical“ gibt der/die Nutzer:in die folgenden Parameter ein:

- sein/ihr Arbeitsverzeichnis.

Unter „concerning the bargaining“ gibt der/die Nutzer:in die folgenden Parameter der Verhandlung ein:

- number_of_players (Anzahl der Spieler),
- needs (ein Vektor mit der Größe der Bedarfe der Spieler]),
- resource (die Größe der Ressource),
- maximum_number_of_sessions (Anzahl der Runden, über die maximal verhandelt werden kann),
- delta (ein Faktor, der weitere Verhandlungsrunden verteuert – siehe Baron und Ferejohn 1989),
- die Anzahl der Nachkommastellen (hiermit wird die Größe gesteuert, um die Abstimmende konkurrierende Abstimmende unterbieten (siehe Springhorn 2021)).

Unter „PROGRAM“ steht die Funktion „execute_program” zur Verfügung, die mit den Parametern „strategy“, „overview“ und „write_excel_file“ aufgerufen werden kann.

- „strategy“ erlaubt die Auswahl zwischen „BF“ (Baron-Ferejohn-Modell), „naive_approach" und „improved_approach" (siehe oben; außerdem ist die Strategie „need“ implementiert, die die Zustimmung davon abhängig macht, ob der Bedarf eines Abstimmenden erfüllt wird; weitere Strategien lassen sich in der Funktion „calculate_and_add_minimum_approval_values_for_current_session“ (Abschnitt „FUNCTIONS“) vergleichsweise einfach nach dem Muster der dort hinterlegten Strategien implementieren),
- „overview“ erlaubt die Auswahl „TRUE“ and „FALSE“; „TRUE“ liefert einen Output mit den wichtigsten In- und Output-Größen, „FALSE“ liefert eine vollständige Übersicht über alle Berechnungsschritte (siehe unten),
- „write_excel_file“erlaubt die Auswahl „TRUE“ and „FALSE“; im Falle von „TRUE“ werden die Ergebnisse im Arbeitsverzeichnis in eine Excel-Datei geschrieben.

Output - „overview = TRUE“

Mit der Parametrisierung „overview = TRUE“ erhält man einen Überblick über die wichtigsten Größen. Unter Verwendung der Input-Parametern

```
number_of_players = 3
needs = c(15, 20, 30)
resource = 100
maximum_number_of_sessions = 2
delta = 0
```

ergeben sich die folgenden Outputs:

Die Parametrisierung „strategy = BF“

session	proposer	player	need	expected_value	minimum_approval_value	probability_for_offer	offer
1	i	i	15	33.33333	66.66667	1	66.66667
1	i	j	20	33.33333	33.33333	0.5	33.33333
1	i	k	30	33.33333	33.33333	0.5	33.33333
1	j	i	15	33.33333	33.33333	0.5	33.33333
1	j	j	20	33.33333	66.66667	1	66.66667
1	j	k	30	33.33333	33.33333	0.5	33.33333
1	k	i	15	33.33333	33.33333	0.5	33.33333
1	k	j	20	33.33333	33.33333	0.5	33.33333
1	k	k	30	33.33333	66.66667	1	66.66667
2	i	i	15	0	100	1	100
2	i	j	20	0	0	0.5	0
2	i	k	30	0	0	0.5	0
2	j	i	15	0	0	0.5	0
2	j	j	20	0	100	1	100
2	j	k	30	0	0	0.5	0
2	k	i	15	0	0	0.5	0
2	k	j	20	0	0	0.5	0
2	k	k	30	0	100	1	100

Tabelle 1: Die Parametrisierungen „strategy = BF“ und „overview = TRUE“

Die Spalten „session”, „proposer” und „player“ ergeben sich (grundsätzlich) als Kreuzprodukt der Input-Parameter „session”, „proposer” und „player“; das heißt für jede Runde wird für jede Auswahl eines Spielers als Vorschlagender und jeden Spieler eine eigene Zeile gebildet. In der Spalte „need“ wird der Bedarf des Spielers ergänzt (der Im Fall „strategy = BF“ keine Rolle spielt). Die Spalte „expected_value“ ergibt sich aus dem Durchschnitt der mit ihren Wahrscheinlichkeiten gewichteten Angeboten, die ein Spieler in der folgenden Runde erhält. Die Spalte „minimum_approval_value“ ist strategieabhängig; im Fall „strategy = BF“ ist er gleich dem Erwartungswert. Das Ergebnis der Verhandlung lässt sich in den Spalten „probability_for_offer“ und „offer“ ablesen: Mit einer Wahrscheinlichkeit von 1 erhält zum Beispiel i – wenn i als Vorschlagender gewählt wird – zwei Drittel der Ressource mit der Größe 100 und die beiden anderen Spieler erhalten mit einer Wahrscheinlichkeit von 0,5 ein Drittel der Ressource mit der Größe 100. Das entspricht der Lösung von Baron und Ferejohn.

Die Parametrisierung „strategy = naive_approach“

session	proposer	player	need	expected_value	minimum_approval_value	probability_for_offer	offer
1	i	i	15	36.66667	36.66667	1	70
1	i	j	20	35	35	0	35
1	i	k	30	28.33333	30	1	30

1	j	i	15	36.66667	36.66667	0	36.66667
1	j	j	20	35	35	1	70
1	j	k	30	28.33333	30	1	30
1	k	i	15	36.66667	36.66667	0	36.66667
1	k	j	20	35	35	1	35
1	k	k	30	28.33333	30	1	65
2	i	i	15	0	15	1	80
2	i	j	20	0	20	1	20
2	i	k	30	0	30	0	30
2	j	i	15	0	15	1	15
2	j	j	20	0	20	1	85
2	j	k	30	0	30	0	30
2	k	i	15	0	15	1	15
2	k	j	20	0	20	0	20
2	k	k	30	0	30	1	85

Tabelle 2: Die Parametrisierungen „strategy = naive_approach“ und „overview = TRUE“

Die ersten vier Spalten sind nicht strategieabhängig und unterscheiden sich dementsprechend nicht von dem Fall mit der Wahl „strategy = BF“. Die Erwartungswerte weichen in Ihre Größe ab. i hat zum Beispiel in der ersten Runde einen Erwartungswert von zirka 36,66. Dieser Wert ergibt sich wie folgt aus der zweiten Runde: Wird i als Vorschlagender gewählt, macht er j ein Angebot in Höhe von 20 (was j akzeptiert, da es das Maximum von seinem Bedarf und seinem Erwartungswert ist) und behält 80 für sich. Wird j oder k als Vorschlagender gewählt, erhält i mit einer Wahrscheinlichkeit von 1 ein Angebot in Höhe von 15, da er der günstigste der konkurrierenden Spieler ist. In der zweiten Runde bestätigt sich damit die plausible Annahme, dass Spieler mit kleinem Bedarf einen Vorteil haben. Die Verhandlung geht allerdings gar nicht in die zweite Runde, da es bereits in der ersten Runde eine Lösung gibt: Wird i als Vorschlagender gewählt, erhält k ein Angebot in Höhe von 30, wird j als Vorschlagender gewählt, erhält k ein Angebot in Höhe von 30 und wird k als Vorschlagender gewählt, erhält j ein Angebot in Höhe von 35. Die Angebote werden jeweils akzeptiert, weil sie das Maximum von Bedarf und Erwartungswert sind. Die Verhandlung endet. Hier zeigt sich, dass der kleine Bedarf von i zu einem großen Erwartungswert führt und i in der ersten Runde der teuerste Spieler ist.

Die Parametrisierung „strategy = improved _approach“

session	proposer	player	need	expected_value	minimum_approval_value	probability_for_offer	offer
1	i	i	15	36.66667	36.66667	1	70.1
1	i	j	20	35	29.9	1	29.9
1	i	k	30	28.33333	30	0	30
1	j	i	15	36.66667	29.9	1	29.9
1	j	j	20	35	35	1	70.1
1	j	k	30	28.33333	30	0	30
1	k	i	15	36.66667	19.9	1	19.9
1	k	j	20	35	20	0	20
1	k	k	30	28.33333	30	1	80.1
2	i	i	15	0	15	1	80

2	i	j	20	0	20	1	20
2	i	k	30	0	30	0	30
2	j	i	15	0	15	1	15
2	j	j	20	0	20	1	85
2	j	k	30	0	30	0	30
2	k	i	15	0	15	1	15
2	k	j	20	0	20	0	20
2	k	k	30	0	30	1	85

Tabelle 3: Die Parametrisierungen „strategy = improved _approach“ und „overview = TRUE“

Wiederum unterscheiden sich die ersten vier Spalten nicht von den vorangegangenen Beispielen und auch die Erwartungswerte sind gleich denen im vorangegangenen Beispiel. Die minimalen Zustimmungswerte folgen nun aber der Strategie, den konkurrierenden Spieler wenn nötig und möglich zu unterbieten. Wird i als Vorschlagender gewählt, akzeptiert j ein Angebot von 29,9 (und erhält ein solches entsprechend mit einer Wahrscheinlichkeit von 1), um – anders als im naiven Ansatz – günstiger zu sein als k und nicht mit leeren Händen nach Hause zu gehen. Werden j oder k als Vorschlagender gewählt, unterbietet i k beziehungsweise j und akzeptiert Angebote in Höhe von 29,9, um den minimalen Zustimmungswert von k in Höhe von 30 zu unterbieten, beziehungsweise in Höhe von 19,9, um den minimalen Zustimmungswert von j in Höhe von 20 zu unterbieten.

Output – „overview = FALSE“

Mit der Parametrisierung „overview = FALSE“ werden alle Größen, die zur Berechnung der Ergebnisse notwendig sind, angezeigt, so dass sich diese Schritt für Schritt nachvollziehen lassen. Im Fall von drei Spielern und zwei Runden wird das im Allgemeinen für die bisher untersuchten Strategien nicht notwendig sein, aber in Fällen mit mehr Spielern und/oder mehr Runden und gegebenenfalls anderen Strategien erweist es sich als sehr hilfreich.

Mit den gleichen Inputparametern wie oben ergibt sich für die Wahl „strategy = improved _approach“ der folgende Output:

session	proposer	player	is_respondent	need	expected_value	need_that_has_to_be_underbidden	minimum_approval_value	rank_of_minimum_approval_value_of_respondents	number_of_respondents_with_given_rank_of_minimum_approval_value	number_of_respondents_with_lower_rank_of_minimum_approval_value	number_of_respondents_with_given_rank_of_minimum_approval_value_needed_for_majority	probability_for_offer	resource	needed_resource	condition_for_majority	offer	probability_for_offer_times_offer	probability_for_offer_times_offer_in_next_session	difference_of_probability_for_offer_times_offer_in_current_and_next_session
1	i	i	FALSE	15	36.66667		36.66667					1	100	66.56667	TRUE	70.1	70.1	80	9.9
1	i	j	TRUE	20	35	30	29.9	1	1	0	1	1	100	66.56667	TRUE	29.9	29.9	20	-9.9
1	i	k	TRUE	30	28.33333	30	30	2	1	1	0	0	100	66.56667	TRUE	30	0	0	0
1	j	i	TRUE	15	36.66667	30	29.9	1	1	0	1	1	100	64.9	TRUE	29.9	29.9	15	-14.9
1	j	j	FALSE	20	35		35					1	100	64.9	TRUE	70.1	70.1	85	14.9
1	j	k	TRUE	30	28.33333	30	30	2	1	1	0	0	100	64.9	TRUE	30	0	0	0
1	k	i	TRUE	15	36.66667	20	19.9	1	1	0	1	1	100	49.9	TRUE	19.9	19.9	15	-4.9
1	k	j	TRUE	20	35	20	20	2	1	1	0	0	100	49.9	TRUE	20	0	0	0
1	k	k	FALSE	30	28.33333		30					1	100	49.9	TRUE	80.1	80.1	85	4.9
2	i	i	FALSE	15	0		15					1	100	35	TRUE	80	80		
2	i	j	TRUE	20	0	30	20	1	1	0	1	1	100	35	TRUE	20	20		
2	i	k	TRUE	30	0	30	30	2	1	1	0	0	100	35	TRUE	30	0		
2	j	i	TRUE	15	0	30	15	1	1	0	1	1	100	35	TRUE	15	15		
2	j	j	FALSE	20	0		20					1	100	35	TRUE	85	85		
2	j	k	TRUE	30	0	30	30	2	1	1	0	0	100	35	TRUE	30	0		
2	k	i	TRUE	15	0	20	15	1	1	0	1	1	100	45	TRUE	15	15		
2	k	j	TRUE	20	0	20	20	2	1	1	0	0	100	45	TRUE	20	0		
2	k	k	FALSE	30	0		30					1	100	45	TRUE	85	85		

Tabelle 4: Die Parametrisierungen „strategy = improved _approach “ und „overview = FALSE “

Die Spalte „is_respondent“ ist rein technischer Natur – sie erlaubt es, in der Excel- Datei auf den Vorschlagenden und die „abstimmenden“ Spieler („respondents“) zu filtern.

„need_that_has_to_be_underbidden“ ist der Bedarf, der – im improved approach oder ähnlichen Ansätzen – unterboten werden muss. Der Vorschlagenden ist davon nicht tangiert, weshalb sich die Angabe auf die „abstimmenden“ Spieler beschränkt. Der Wert ergibt sich als der Wert der geordneten Bedarfe der respondents, der eine Stelle über der aufgerundeten Mitte dieser Liste steht (wenn es drei respondents gibt, ist das die zweite Stelle, genauso in dem Fall, dass es vier respondents gibt, weil jeweils zwei respondents für eine Mehrheit benötigt werden – das ist ein einfaches Vorgehen auch in dem Fall, dass mehrere respondents den gleichen Bedarf haben). Zu beachten ist, dass dieser Wert nur Berücksichtigung findet, wenn er größer als der Bedarf ist.

Vergleichsweise kompliziert gestaltet sich die Berechnung der Wahrscheinlichkeit für ein Angebot (probability_for_offer), weil es vorkommen kann, dass nicht alle respondents mit gleich großem minimalen Zustimmungswert für eine Mehrheit benötigt werden und unter ihnen eine hinreichend große Anzahl zufällig und mit gleicher Wahrscheinlichkeit ausgewählt werden muss. Zu diesem Zweck werden die disjunkten minimalen Zustimmungswerte der Größe nach geordnet. Jedem minimalen Zustimmungswert wird ein dieser Ordnung folgender Rang zugewiesen (rank_of_minimum_approval_value_of_respondents). Es wird bestimmt, wie viele respondents mit einem jeden dieser Ränge existieren (number_of_respondents_with_given_rank_of_minimum_approval_value) und wie viele respondents einen kleineren Rang haben (number_of_respondents_with_lower_rank_of_minimum_approval_value). Daraus wird abgeleitet, wie viele respondents mit dem gegebenen Rang für eine Mehrheit benötigt werden (number_of_respondents_with_given_rank_of_minimum_approval_value_needed_for_majority). Werden alle repondents mit dem gegebenen Rang benötigt, wird für sie die Wahrscheinlichkeit für ein Angebot auf 1 gesetzt. Wird keiner der respondents mit dem gegebenen Rang benötigt, wird sie auf 0 gesetzt. Andernfalls beträgt sie 1/ number_of_respondents_with_given_rank_of_minimum_approval_value_needed_for_majority.

In den Spalten “resource”, “needed_resource” und “condition_for_majority” wird überprüft, ob die Ressource groß genug ist, um unter den Bedingungen der gegebenen Runde die minimalen Zustimmungswerte der für eine Mehrheit notwendigen Spieler in Summe zu decken (eine Randbedingung, die die Bestimmung von Lösungen unter Berücksichtigung von Bedarfen gegenüber dem BF-Modell deutlich erschwert – siehe Springhorn 2021).

Abschließend wird in den Spalten „probability_for_offer_times_offer”, “probability_for_offer_times_offer_in_next_session” und “difference_of_probability_for_offer_times_offer_in_current_and_next_session” quantifiziert, ob und welche Veränderungen es zwischen der gegebenen Runde und der vorherigen Runde gibt. Damit ist es möglich, im Fall von vielen und potenziell unendlich vielen Runden Muster und gegebenenfalls Grenzwerte zu erkennen (ein Beispiel dafür findet sich im nächsten Kapitel).

Ausblick

Für den folgenden Output wurden folgende Input-Parameter

```
number_of_players = 3
needs = c(12, 15, 18)
resource = 60
maximum_number_of_sessions = 20
delta = 0
```

und die Parametrisierung „strategy = naive_approach“ genutzt.

session	proposer	player	is_respondent	need	expected_value	minimum_approval_value	probability_for_offer	offer	probability_for_offer_times_offer	probability_for_offer_times_offer_in_next_session	difference_of_probability_for_offer_times_offer_in_current_and_next_session
1	i	i	FALSE	12	22.97143	22.97143	1	42	42	42.17143	0.17143
1	i	j	TRUE	15	22.97143	22.97143	0	22.97143	0	17.82857	17.82857
1	i	k	TRUE	18	14.05714	18	1	18	18	0	-18
1	j	i	TRUE	12	22.97143	22.97143	0	22.97143	0	17.82857	17.82857
1	j	j	FALSE	15	22.97143	22.97143	1	42	42	42.17143	0.17143
1	j	k	TRUE	18	14.05714	18	1	18	18	0	-18
1	k	i	TRUE	12	22.97143	22.97143	0.5	22.97143	11.48571	8.91429	-2.57143
1	k	j	TRUE	15	22.97143	22.97143	0.5	22.97143	11.48571	8.91429	-2.57143
1	k	k	FALSE	18	14.05714	18	1	37.02857	37.02857	42.17143	5.14286
2	i	i	FALSE	12	17.82857	17.82857	1	42.17143	42.17143	42	-0.17143
2	i	j	TRUE	15	17.82857	17.82857	1	17.82857	17.82857	0	-17.82857
2	i	k	TRUE	18	24.34286	24.34286	0	24.34286	0	18	18
2	j	i	TRUE	12	17.82857	17.82857	1	17.82857	17.82857	0	-17.82857
2	j	j	FALSE	15	17.82857	17.82857	1	42.17143	42.17143	42	-0.17143
2	j	k	TRUE	18	24.34286	24.34286	0	24.34286	0	18	18
2	k	i	TRUE	12	17.82857	17.82857	0.5	17.82857	8.91429	11.48571	2.57143
2	k	j	TRUE	15	17.82857	17.82857	0.5	17.82857	8.91429	11.48571	2.57143
2	k	k	FALSE	18	24.34286	24.34286	1	42.17143	42.17143	37.02857	-5.14286
3	i	i	FALSE	12	22.97143	22.97143	1	42	42	42.17143	0.17143
3	i	j	TRUE	15	22.97143	22.97143	0	22.97143	0	17.82857	17.82857
3	i	k	TRUE	18	14.05714	18	1	18	18	0	-18
3	j	i	TRUE	12	22.97143	22.97143	0	22.97143	0	17.82857	17.82857
3	j	j	FALSE	15	22.97143	22.97143	1	42	42	42.17143	0.17143
3	j	k	TRUE	18	14.05714	18	1	18	18	0	-18
3	k	i	TRUE	12	22.97143	22.97143	0.5	22.97143	11.48571	8.91429	-2.57143
3	k	j	TRUE	15	22.97143	22.97143	0.5	22.97143	11.48571	8.91429	-2.57143
3	k	k	FALSE	18	14.05714	18	1	37.02857	37.02857	42.17143	5.14286
4	i	i	FALSE	12	17.82857	17.82857	1	42.17143	42.17143	42	-0.17143
4	i	j	TRUE	15	17.82857	17.82857	1	17.82857	17.82857	0	-17.82857
4	i	k	TRUE	18	24.34286	24.34286	0	24.34286	0	18	18
4	j	i	TRUE	12	17.82857	17.82857	1	17.82857	17.82857	0	-17.82857
4	j	j	FALSE	15	17.82857	17.82857	1	42.17143	42.17143	42	-0.17143
4	j	k	TRUE	18	24.34286	24.34286	0	24.34286	0	18	18
4	k	i	TRUE	12	17.82857	17.82857	0.5	17.82857	8.91429	11.48571	2.57143
4	k	j	TRUE	15	17.82857	17.82857	0.5	17.82857	8.91429	11.48571	2.57143
4	k	k	FALSE	18	24.34286	24.34286	1	42.17143	42.17143	37.02857	-5.14286

Tabelle 5: Outlook - Die Parametrisierungen „strategy = naive _approach“ und „overview = FALSE“ (Ausschnitt)

Zu beachten ist, dass hier nur die ersten vier Runden von insgesamt 20 Runden (die alle im Output stehen) dargestellt werden (außerdem wurden der Übersicht halber einige Spalten entfernt).

Interessant ist das Beispiel, weil sich die Lösungen wiederholen. Die Lösungen der ersten Runde finden sich genauso in der dritten Runde, der fünften Runde usw. (genaugenommen konvergieren die Lösungen von der 20. Runde zurückgerechnet gegen zwei alternierende Grenzwerte). Ursache dafür ist – vereinfacht gesagt – wiederum der bereits beschrieben Effekt, dass kleine/große Bedarfe (initial) zu großen/kleinen Erwartungswerten und damit zu großen/kleinen minimalen Zustimmungswerten führen können. Gibt es mehr als zwei Runden, kann der Effekt eintreten, dass der minimale Zustimmungswert eines Spielers (von der letzten Runde zurückgerechnet) zunächst wächst/sinkt, bis er über/unter den minimale Zustimmungswert eines anderen Spielers wächst/sinkt, womit sich die Wahl eines Spielers durch den Vorschlagenden von dem einen Spieler auf den anderen verlagert und sich das Wachstumsverhalten umkehrt.

Auch wenn der naive approach bereits (aus anderen Gründen) verworfen wurde, zeigt sich damit eine Herausforderung für den improved approach, denn es könnte für Spieler vorteilhaft sein, eine Lösung in einer Runde zu verhindern, weil sie in einer späteren Runde ein besseres Ergebnis erzielen (im Fall von zwei Runden ist das nicht der Fall – siehe Springhorn 2021). Ein Spieler würde demnach einen Verteilungsvorschlag nicht zustimmen, obwohl er seinen dem improved approach folgenden minimalen Zustimmungswert angeboten bekommt (und damit der Strategie des improved approach folgend zustimmen sollte), weil er durch eine Ablehnung verhindern kann, dass es zu einer Mehrheit kommt und (eine) weitere (für ihn vorteilhafte) Runde(n) verhandelt werden muss. Damit würde sich auch der improved approach für Verhandlungen mit mehr als zwei Runden ebenfalls als „naiv“ erweisen.

Bei der Untersuchung dieser Fragestellung und einer gegebenenfalls daraus resultierenden Weiterentwicklung der Strategien kann das Programm eine große Hilfe sein, ebenso in dem Fall, dass sich eine theoretische Bestimmung der Lösungen als zu komplex erweisen sollte (angesichts der Komplexität, die der improved approach bereits im Fall von zwei Runden aufweist, bin ich skeptisch, dass sich analytische Lösungen bestimmen lassen, wenn sich bestätigen sollte, dass nicht nur die folgende Runde, sondern potenziell alle Runden berücksichtigt werden müssen). In letzterem Fall lässt sich aber auf Basis des Programms ein algorithmisches Vorgehen etablieren, als das sich aus dem vollständigen Verhandlungsverlauf ablesen lässt, ob es sich für einen Spieler – sofern die Möglichkeit besteht, eine Mehrheit zu verhindern – lohnt, eine Mehrheit zu verhindern.

Zusammenfassung

Mit dem Programm wird die Möglichkeit geschaffen, vollständige Verhandlungsverläufe darzustellen und damit die Verhandlungslösungen Schritt für Schritt nachzuvollziehen und zu untersuchen. Die Anzahl der Spieler, die Größe ihrer Bedarfe, die Größe der Ressource und die Anzahl der Runden, über die maximal verhandelt werden kann sind dabei frei parametrisierbar. Als Strategien sind die Strategien des Baron-Ferejohn-Modells, des naive approach und des improved approach bereits implementiert. Das Programm ist so angelegt, dass sich weitere Strategien mit geringen Programmierkenntnissen vergleichsweise einfach hinzufügen lassen. Das Programm kann damit zwar Lösungen für Verhandlungssituation liefern, sein Hauptzweck liegt aber in der theoretischen Untersuchung von Verhandlungsverläufen: Es kann einem besseren Verständnis der bestehenden Strategien dienen, vor allem aber hilft es bei dem Verständnis und der Untersuchung von Weiter- und Neuentwicklungen. In dem Kapitel „Ausblick“ ist ein solcher Anwendungsfall – der eine gewissermaßen natürliche Fortsetzung der bereits getanen Arbeit ist – skizziert.

Literatur

Agranov, M. and C. Tergiman 2014. Communication in multilateral bargaining. *Journal of Public Economics* 118 (October): 75-85.

Baranski, A. and R. Morton 2020. The Determinants of Multilateral Bargaining: A Comprehensive Analysis of Baron and Ferejohn Majoritarian Bargaining Experiments. *Division of Social Science Working Paper Series*, Working Paper 0037, New York University Abu Dhabi, Abu Dhabi.

Baron, D.P. and J.A. Ferejohn 1989. Bargaining in Legislatures. *The American Political Science Review* 83 (4): 1181–1206.

Miller, L., M. Montero and C. Vanberg 2018. Legislative Bargaining with Heterogeneous Disagreement Values: Theory and Experiments. *Games and Economic Behavior* 107: 60–92.

R Core Team 2025. _R: A Language and Environment for Statistical Computing_. R Foundation for Statistical Computing, Vienna, Austria. <https://www.R-project.org/>.

Schauberger P. and A. Walker 2025. _openxlsx: Read, Write and Edit xlsx Files_. doi:10.32614/CRAN.package.openxlsx <https://doi.org/10.32614/CRAN.package.openxlsx>, R package version 4.2.8, <https://CRAN.R-project.org/package=openxlsx>.

Springhorn, N. 2021. Bargaining According to the Baron-Ferejohn Model, Taking into Account Need. FOR2104 Working Paper 2021-06. Hamburg: Helmut Schmidt University.

Springhorn, N. 2024. Capitulate for Nothing? Does Baron and Ferejohn's Bargaining Model Fail Because No One Would Give Everything for Nothing?. In *Oldenburger Jahrbuch für Philosophie 2021/2022*, ed. A. Bauer und H. Grass, 227-236. Oldenburg: University of Oldenburg Press.

Wickham H., M. Averick, J. Bryan, W. Chang, L.D. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T.L. Pedersen, E. Miller, S.M. Bache, K. Müller, J. Ooms, D. Robinson, D.P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, H. Yutani 2019. Welcome to the tidyverse. *_Journal of Open Source Software_,* *4*(43), 1686. doi:10.21105/joss.01686 <https://doi.org/10.21105/joss.01686>.

Anhang

Code

```
################################################################################
### TECHNICAL STUFF
################################################################################

# Remove old data
rm(list = ls())

# Install, if necessary, required packages
# install.packages(c("tidyverse", "openxlsx"))

# Open required packages
invisible(lapply(c("tidyverse", "openxlsx"), library, character.only = T))

################################################################################
### ADMINISTRATION
################################################################################

# technical ####
my_working_directory = "~/."
```

```
setwd(my_working_directory)

# concerning the bargaining ####
number_of_players = 3
needs = c(15, 20, 30)
resource = 100
maximum_number_of_sessions = 2
delta = 0
decimal_digits = 1


################################################################################
### CHECKS
################################################################################

if (length(needs) != number_of_players) stop("The number of players does not match the number of needs.")

################################################################################
### PREPARATIONS
################################################################################

players =
 letters[9:(8+number_of_players)]

number_of_players_needed_for_majority =
 floor(number_of_players/2) + 1

number_of_respondents_needed_for_majority =
 number_of_players_needed_for_majority - 1

data_for_all_sessions =
 tibble()

################################################################################
### FUNCTIONS
################################################################################

calculate_and_add_expected_values_for_next_session = function(data_for_current_session, data_for_next_session) {

 print("   calculate_and_add_expected_values_for_next_session")

 number_of_current_session =
  data_for_current_session %>%
  pull(session) %>%
  unique()

 if (number_of_current_session == maximum_number_of_sessions) {
```

```
  data_for_current_session =
    data_for_current_session %>%
    mutate(expected_value = 0)

 } else {

  data_for_current_session =
    data_for_current_session %>%
    left_join(data_for_next_session %>%
                group_by(player) %>%
                mutate(expected_value = mean(offer * probability_for_offer)) %>%
                select(player, expected_value) %>%
                ungroup() %>%
                unique(),
              by = c("player"))

 }

 return(data_for_current_session)

}

calculate_and_add_minimum_approval_values_for_current_session = function(data_for_cur-
rent_session, strategy) {

 print("   calculate_and_add_minimum_approval_values_for_current_session")

 if (strategy == "BF") {

  data_for_current_session =
    data_for_current_session %>%
    mutate(minimum_approval_value =
           case_when(is_respondent ~ expected_value,
                     T ~ resource - number_of_respondents_needed_for_majority * expected_value))

 } else if (strategy == "need") {

  data_for_current_session =
    data_for_current_session %>%
    mutate(minimum_approval_value = need)

 } else if (strategy == "naive_approach") {

  data_for_current_session =
    data_for_current_session %>%
    mutate(minimum_approval_value =
           pmax(need, expected_value))
```

```
  } else if (strategy == "improved_approach") {

    # data_for_respondents ####
    data_of_respondents =
      data_for_current_session %>%
      filter(is_respondent)

    # rank_of_need_of_respondents ####
    rank_of_need_of_respondents =
      data_of_respondents %>%
      group_by(session, proposer) %>%
      select(session, proposer, need) %>%
      unique() %>%
      ungroup() %>%
      group_by(session, proposer) %>%
      mutate(rank_of_need_of_respondents = rank(need, ties.method = "first")) %>%
      ungroup()
    rank_of_need_of_respondents

    data_of_respondents =
      data_of_respondents %>%
      left_join(rank_of_need_of_respondents,
                by = c("session", "proposer", "need"))
    data_of_respondents

    # number_of_respondents_with_given_rank_of_need ####
    data_of_respondents =
      data_of_respondents %>%
      group_by(session, proposer, rank_of_need_of_respondents) %>%
      mutate(number_of_respondents_with_given_rank_of_need = n()) %>%
      ungroup()
    data_of_respondents

    # number_of_respondents_with_lower_rank_of_need ####
    number_of_respondents_with_lower_rank_of_need =
      data_of_respondents %>%
      select(session, proposer, need, rank_of_need_of_respondents, number_of_respond-
ents_with_given_rank_of_need) %>%
      unique() %>%
      group_by(session, proposer) %>%
      arrange(session, proposer, rank_of_need_of_respondents) %>%
      mutate(number_of_respondents_with_lower_rank_of_need = cumsum(number_of_respond-
ents_with_given_rank_of_need) - number_of_respondents_with_given_rank_of_need) %>%
      ungroup()
    number_of_respondents_with_lower_rank_of_need

    data_of_respondents =
      data_of_respondents %>%
      left_join(number_of_respondents_with_lower_rank_of_need,
```

```
          by = c("session", "proposer", "need", "rank_of_need_of_respondents", "number_of_respondents_with_given_rank_of_need"))
  data_of_respondents

  # number_of_respondents_with_given_rank_of_need_needed_for_majority ####
  number_of_respondents_with_given_rank_of_need_needed_for_majority =
    number_of_respondents_with_lower_rank_of_need %>%
    mutate(number_of_respondents_with_given_rank_of_need_needed_for_majority =
          case_when(number_of_respondents_with_lower_rank_of_need >= number_of_respondents_needed_for_majority ~ 0,
                    T ~ pmin(number_of_respondents_with_given_rank_of_need, number_of_respondents_needed_for_majority - number_of_respondents_with_lower_rank_of_need)))
  number_of_respondents_with_given_rank_of_need_needed_for_majority

  data_of_respondents =
    data_of_respondents %>%
    left_join(number_of_respondents_with_given_rank_of_need_needed_for_majority,
          by = c("session", "proposer", "need", "rank_of_need_of_respondents", "number_of_respondents_with_given_rank_of_need", "number_of_respondents_with_lower_rank_of_need"))
  data_of_respondents

  # need_that_has_to_be_underbidden
  need_that_has_to_be_underbidden =
    number_of_respondents_with_given_rank_of_need_needed_for_majority %>%
    group_by(session, proposer) %>%
    mutate(need_that_has_to_be_underbidden = need[match(0, number_of_respondents_with_given_rank_of_need_needed_for_majority)])
  need_that_has_to_be_underbidden

  data_of_respondents =
    data_of_respondents %>%
    left_join(need_that_has_to_be_underbidden,
          by = c("session", "proposer", "need", "rank_of_need_of_respondents", "number_of_respondents_with_given_rank_of_need", "number_of_respondents_with_lower_rank_of_need", "number_of_respondents_with_given_rank_of_need_needed_for_majority"))
  data_of_respondents

  # minimum_approval_value for all players (proposer and respondents) of the current session
  data_for_current_session =
    data_of_respondents %>%
    bind_rows(data_for_current_session %>%
            filter(!is_respondent)) %>%
    mutate(minimum_approval_value =
          case_when(is_respondent & !is.na(need_that_has_to_be_underbidden) ~ pmax(need, pmin(expected_value, need_that_has_to_be_underbidden - 10^-decimal_digits)),
                    is_respondent & is.na(need_that_has_to_be_underbidden) ~ pmax(need, expected_value),
                    T ~ pmax(need, expected_value))) %>%
    arrange(session, proposer, player)
```

```
  data_for_current_session

 } else {

  stop("The chosen strategy is not known.")

 }

 return(data_for_current_session)

}

calculate_and_add_probability_of_offers_for_current_session = function(data_for_current_session) {

 print("   calculate_and_add_probability_of_offers_for_current_session")

 # data_for_respondents ####
 data_of_respondents =
  data_for_current_session %>%
  filter(is_respondent)

 # rank_of_minimum_approval_value_of_respondents ####
 ranks_of_minimum_approval_value_of_respondents =
  data_of_respondents %>%
  group_by(session, proposer) %>%
  select(session, proposer, minimum_approval_value) %>%
  unique() %>%
  ungroup() %>%
  group_by(session, proposer) %>%
  mutate(rank_of_minimum_approval_value_of_respondents = rank(minimum_approval_value,
ties.method = "first")) %>%
  ungroup()
 ranks_of_minimum_approval_value_of_respondents

 data_of_respondents =
  data_of_respondents %>%
  left_join(ranks_of_minimum_approval_value_of_respondents,
        by = c("session",
            "proposer",
            "minimum_approval_value"))
 data_of_respondents

 # number_of_respondents_with_given_rank_of_minimum_approval_value ####
 data_of_respondents =
  data_of_respondents %>%
  group_by(session, proposer, rank_of_minimum_approval_value_of_respondents) %>%
  mutate(number_of_respondents_with_given_rank_of_minimum_approval_value = n()) %>%
  ungroup()
 data_of_respondents
```

```
 # number_of_respondents_with_lower_ranks_of_minimum_approval_value ####
 number_of_respondents_with_lower_rank_of_minimum_approval_value =
   data_of_respondents %>%
   select(session, proposer, rank_of_minimum_approval_value_of_respondents, number_of_respondents_with_given_rank_of_minimum_approval_value) %>%
   unique() %>%
   group_by(session, proposer) %>%
   arrange(session, proposer, rank_of_minimum_approval_value_of_respondents) %>%
   mutate(number_of_respondents_with_lower_rank_of_minimum_approval_value =
          cumsum(number_of_respondents_with_given_rank_of_minimum_approval_value) - number_of_respondents_with_given_rank_of_minimum_approval_value) %>%
   ungroup()
 number_of_respondents_with_lower_rank_of_minimum_approval_value

 data_of_respondents =
   data_of_respondents %>%
   left_join(number_of_respondents_with_lower_rank_of_minimum_approval_value,
           by = c("session",
                "proposer",
                "rank_of_minimum_approval_value_of_respondents",
                "number_of_respondents_with_given_rank_of_minimum_approval_value"))
 data_of_respondents

 # number_of_respondents_with_given_rank_of_minimum_approval_value_needed_for_majority_and_probably_for_offer ####
 number_of_respondents_with_given_rank_of_minimum_approval_value_needed_for_majority_and_probably_for_offer =
   number_of_respondents_with_lower_rank_of_minimum_approval_value %>%
   mutate(number_of_respondents_with_given_rank_of_minimum_approval_value_needed_for_majority =
          case_when(number_of_respondents_with_lower_rank_of_minimum_approval_value >= number_of_respondents_needed_for_majority ~ 0,
                    T ~ number_of_respondents_needed_for_majority - number_of_respondents_with_lower_rank_of_minimum_approval_value))
 number_of_respondents_with_given_rank_of_minimum_approval_value_needed_for_majority_and_probably_for_offer

 data_of_respondents =
   data_of_respondents %>%
   left_join(number_of_respondents_with_given_rank_of_minimum_approval_value_needed_for_majority_and_probably_for_offer,
           by = c("session",
                "proposer",
                "rank_of_minimum_approval_value_of_respondents",
                "number_of_respondents_with_given_rank_of_minimum_approval_value",
                "number_of_respondents_with_lower_rank_of_minimum_approval_value"))
 data_of_respondents
```

```
  # probability_for_offer for all players (proposer and respondents) of the current session ####
  data_for_current_session =
    data_of_respondents %>%
    bind_rows(data_for_current_session %>%
                filter(!is_respondent)) %>%
    mutate(probability_for_offer =
             case_when(!is_respondent ~ 1,
                       T ~ number_of_respondents_with_given_rank_of_minimum_ap-
proval_value_needed_for_majority/number_of_respondents_with_given_rank_of_minimum_ap-
proval_value)) %>%
    arrange(session, proposer, player)
  data_for_current_session

  return(data_for_current_session)

}

check_on_condition_for_majority_for_current_session = function(data_for_current_session) {

  print("   check_on_condition_for_majority_for_current_session")

  conditions_for_majority =
    data_for_current_session %>%
    select(session, proposer, player, minimum_approval_value, probability_for_offer) %>%
    group_by(session, proposer) %>%
    mutate(resource = resource,
           needed_resource = sum(probability_for_offer * minimum_approval_value),
           condition_for_majority =
             case_when(needed_resource <= resource ~ T,
                       T ~ F)) %>%
    ungroup()

  data_for_current_session =
    data_for_current_session %>%
    left_join(conditions_for_majority,
              by = c("session", "proposer", "player", "minimum_approval_value", "probability_for_offer"))

  return(data_for_current_session)

}

calculate_and_add_offers_for_current_session = function(data_for_current_session) {

  print("   calculate_and_add_offers_for_current_session")

  offers =
    data_for_current_session %>%
    group_by(session, proposer) %>%
```

```
  mutate(offer = case_when(condition_for_majority & !is_respondent  ~ resource - sum(probabil-
ity_for_offer * minimum_approval_value ) + minimum_approval_value,
                   condition_for_majority & is_respondent ~ minimum_approval_value,
                   T ~ 0)) %>%
  ungroup()

 data_for_current_session =
  data_for_current_session %>%
  left_join(offers,
        by = colnames(data_for_current_session)) %>%
  mutate(probability_for_offer_times_offer = probability_for_offer * offer)

 return(data_for_current_session)

}

calculate_data_for_current_session = function(session, data_for_next_session, strategy) {

 print("   calculate_data_for_current_session")

 # initialize dataframe for current session
 data_for_current_session =
  tibble(session = session) %>%
  cross_join(tibble(proposer = players)) %>%
  cross_join(tibble(player = players)) %>%
  mutate(is_respondent = ifelse(proposer == player, F, T)) %>%
  left_join(tibble(player = players, need = needs),
        by = c("player"))

 # add data for current session
 data_for_current_session =
  data_for_current_session %>%
  calculate_and_add_expected_values_for_next_session(., data_for_next_session)

 data_for_current_session =
  data_for_current_session %>%
  calculate_and_add_minimum_approval_values_for_current_session(., strategy)

 data_for_current_session =
  data_for_current_session %>%
  calculate_and_add_probability_of_offers_for_current_session(.)

 data_for_current_session =
  data_for_current_session %>%
  check_on_condition_for_majority_for_current_session(.)

 data_for_current_session =
  data_for_current_session %>%
  calculate_and_add_offers_for_current_session(.)
```

```
  return(data_for_current_session)

}

calculate_data_for_all_sessions = function(strategy) {

  # execute backward induction ####
  print("execute backward induction")

  for (session in maximum_number_of_sessions:1) {

    print(paste0("backward, session = ", session))

    if (session == maximum_number_of_sessions) {

      data_for_next_session = tibble()

    } else {

      next_session = session + 1

      data_for_next_session =
        data_for_all_sessions %>%
        filter(session == next_session)

    }

    data_for_current_session =
      calculate_data_for_current_session(session, data_for_next_session, strategy)

    if (session == maximum_number_of_sessions) {

      data_for_all_sessions =
        data_for_current_session

    } else {

      data_for_all_sessions =
        rbind(data_for_current_session, data_for_all_sessions)

    }

  }

  # check for differences between the offers from the previous session and the current session
  print("check for differences between the offers from the previous session and the current session")

  probability_for_offer_times_offer_in_next_session =
```

```
  data_for_all_sessions %>%
  select(next_session = session,
         proposer,
         player,
         probability_for_offer_times_offer_in_next_session = probability_for_offer_times_offer)

 data_for_all_sessions =
  data_for_all_sessions %>%
  mutate(next_session = session + 1) %>%
  left_join(probability_for_offer_times_offer_in_next_session,
            by = c("next_session",
                   "proposer",
                   "player")) %>%
  select(-next_session) %>%
  mutate(difference_of_probability_for_offer_times_offer_in_current_and_next_session =
           probability_for_offer_times_offer_in_next_session - probability_for_offer_times_offer)

 return(data_for_all_sessions)

}

##############################################################################################
### PROGRAM
##############################################################################################

execute_program = function(strategy, overview = T, write_excel_file = F) {

 all_data =
  calculate_data_for_all_sessions(strategy) %>%
  mutate_if(is.numeric, list(~round(., digits = 5))) %>%
  mutate_all(list(~str_replace_all(., c("WAHR" = "T", "FALSCH" = "F"))))
 # %>%
 #   select(-c(rank_of_need_of_respondents,
 #             number_of_respondents_with_given_rank_of_need,
 #             number_of_respondents_with_lower_rank_of_need,
 #             number_of_respondents_with_given_rank_of_need_needed_for_majority))

 overview_data =
  all_data %>%
  select(session,
         proposer,
         player,
         need,
         expected_value,
         minimum_approval_value,
         probability_for_offer,
         offer)

 if (overview == T) {
```

```
  data = overview_data

 } else {

  data = all_data

 }

 if (write_excel_file == T) {

  wb = createWorkbook()

  addWorksheet(wb, "overview")
  writeDataTable(wb, sheet = "overview", x = overview_data, startCol = 1, startRow = 1, colNames = T, tableStyle = "TableStyleLight1", headerStyle = createStyle(textRotation = 90, valign = "bottom", halign = "left"))
  freezePane(wb, sheet = "overview", firstActiveRow = 2)
  setColWidths(wb, sheet = "overview", cols = c(1:length(colnames(overview_data))), widths = 6)

  addWorksheet(wb, "all data")
  writeDataTable(wb, sheet = "all data", x = all_data, startCol = 1, startRow = 1, colNames = T, tableStyle = "TableStyleLight1", headerStyle = createStyle(textRotation = 90, valign = "bottom", halign = "left"))
  freezePane(wb, sheet = "all data", firstActiveRow = 2)
  setColWidths(wb, sheet = "all data", cols = c(1:length(colnames(all_data))), widths = 6)

  foldername = my_working_directory; if(!dir.exists(foldername)){dir.create(foldername)}
  filename = paste0("bargaining_", format(Sys.time(), "timestamp_%Y%m%d_%H%M%S"), ".xlsx")
  saveWorkbook(wb, file = paste0(foldername, "/", filename), overwrite = F)

 }

 return(data)

}

################################################################################
### EXAMPLES
################################################################################

execute_program("BF", T, F)
execute_program("BF", F, F)
execute_program("BF", F, F) %>% view() # view allows a better view in a separate window
execute_program("naive_approach", T, F)
execute_program("naive_approach", T, F) %>% view()
execute_program("improved_approach", T, F)
execute_program("improved_approach", T, F) %>% view()
execute_program("improved_approach", F, T)
```

DFG Research Group 2104
– Latest Contributions

https://www.hsu-hh.de/bedarfsgerechtigkeit/publikationen/

Springhorn, Nils: On the Measurement of Need-based Justice II. Working Paper Nr. 2024-01.

Krügel, Jan Philipp and Traub, Stefan: Heterogeneity, Risk-taking and Discrimination: An Experimental Study. Working Paper Nr. 2023-03

Dietrich, Brian: Public Service Motivation in Germany. An Empirical Assessment of the Adaptability of PSM to Germany. Working Paper Nr. 2023-02.

Dietrich, Brian, Jankowski, Michael, Schnapp, Kai-Uwe and Tepe, Markus: Differentiation or Discrimination? Discretionary Decision-making of Street-level Bureaucrats. Working Paper Nr. 2023-01.

Kittel, Bernhard, Schwaninger, Manuel and Szendrö Réka: Need-based Justice beyond Close Social Ties: Experimental Evidence on the Scope of the Need Principle. Working Paper Nr. 2022-03.

Wyszynski, Marc and Diederich, Adele: Individual differences moderate effects in an Unusual Disease paradigm: A within-subjects study using a psychophysical data collection approach. Working Paper Nr. 2022-02.

Wyszynski, Marc and Bauer, Alexander Max: Give what you can, take what you need – The effect of framing on rule-breaking behavior in social dilemmas. Working Paper Nr. 2022-01.

Springhorn, Nils: Bargaining According to the Baron-Ferejohn Model, Taking into Account Need. Working Paper Nr. 2021-06.

Springhorn, Nils: Capitulate for Nothing? Does Baron and Ferejohn's Bargaining Model Fail Because No One Would Give Everything for Nothing? Working Paper Nr. 2021-05.

Kittel, Bernhard, Neuhofer, Sabine and Schwaninger, Manuel: The Dark Side of Transparent Needs. An Experiment on Information and Need-based Justice. Working Paper Nr. 2021-04.

Neuhofer, Sabine: Let's chat about justice in a fair distribution experiment. Working Paper Nr. 2021-03.

Schwaninger, Manuel: Sharing with the Powerless Third: Other-regarding Preferences in Dynamic Bargaining. Working Paper Nr. 2021-02.

Traub, Stefan, Schwaninger, Manuel, Paetzel, Fabian and Neuhofer, Sabine: Evidence on Need-sensitive Giving Behavior: An Experimental Approach to the Acknowledgment of Needs. Working Paper Nr. 2021-01.

Bauer, Alexander Max: Sated but Thirsty. Towards a Multidimensional Measure of Need-Based Justice. Working Paper Nr. 2018-03.

Khadjavi, Menusch and Nicklisch, Andreas: Parent's Ambitions and Children's Competitiveness. Working Paper Nr. 2018-02.

Bauer, Alexander Max: Monotonie und Monotoniesensitivität als Desiderata für Maße der Bedarfsgerechtigkeit. Working Paper Nr. 2018-01.

Schramme, Thomas: Mill and Miller: Some thoughts on the methodology of political theory. Working Paper Nr. 2017-25.

Kittel, Bernhard, Tepe, Markus and Lutz, Maximilian: Expert Advice in Need-based Allocations. Working Paper Nr. 2017-24.