DyPy:用于模拟矩阵形式游戏的Python库

Anjalika Nande[1], Andrew Ferdowsian[3], Eric Lubin[4], Erez Yoeli[5], and Martin Nowak[2,1]哈佛大学
物理系。
2. 哈佛大学数学系和有机体与进化生物学系
大学
3. 普林斯顿大学经济系
4麻省理工学院计算机科学系
5麻省理工学院斯隆管理学院

摘要

进化博弈论（EGT）模拟被用来模拟从生物学到经济学到语言学等一系列领域中经历生物和文化进化的人群。在本文中，我们介绍了DyPy，这是一个开源的Python软件包，可以对任何矩阵形式的博弈进行进化模拟，用于三种常见的进化动力学。Moran, Wright-Fisher和Replicator。我们讨论了这个包的基本组成部分，并说明了如何使用它来运行各种模拟。我们的软件包允许用户在没有太多Python知识的情况下相当容易地运行这种模拟。我们希望这将是许多不同领域的研究人员的巨大财富。

关键字：进化博弈论、Python、矩阵形式游戏、群体选择、频率偏向模仿

简介

进化博弈论（EGT）被用来模拟经历达尔文或文化进化的种群；生物学、经济学、社会学、人类学、语言学和哲学的研究人员已经用它来探索诸如多细胞性、人类合作和语法进化等课题（Smith和Price，1973；Smith，1982；Matsui，1996；Traulsen等人，2010；Nowak，2006a；Jaeger，2008；Bruin，2005；Friedman，1998；Johnson等人，2003）。虽然研究人员通常可以分析解决简单游戏的进化动力学问题，但对于许多游戏来说，分析解决是困难的或不可能的。在这种情况下，研究人员通常依靠数值模拟来探索一个游戏的动态（回顾见Nowak, 2006a）。

在本文中，我们介绍了DyPy，这是一个开源的Python软件包，可以用来轻松地运行各种有用的EGT模拟，这些模拟到目前为止还需要研究人员进行重复的、耗费时间和精力的编码。该软件包提供了大多数简单的EGT模拟所需的功能，如允许用户分析任何矩阵形式的游戏，无论玩家和策略的数量如何，也无论游戏是否对称。用户可以从三种最常用的进化动力学中进行选择。Moran, Wright-Fisher和Replicator。他们可以指定人口是否只受制于个人层面的选择，或者是否也有群体层面的选择。而且，他们可以考虑到频率偏向的模仿。用户可以分析进化动态的一次运行，或多次运行。他们还可以在一定范围内改变模型参数，以探索参数值的选择对其游戏动态的影响。任何模拟的结果都可以保存为一个数据集，其中包含策略随时间变化的频率和它们的报酬。这些结果也可以用一些图形功能直观地呈现出来。大多数模拟只需要20-100行代码，只要有一点编程经验就可以编写。DyPy已被并行化和优化，以确保模拟运行速度。据我们所知，DyPy在其范围和实施难度方面是独一无二的。

*通讯作者。电子邮件：anande01@g.harvard.edu

DyPy的可用性

DyPy是一个开源的Python软件库，托管在Github上，网址是 https://github.com/anjalika-nande/dynamics 模拟。它在Python 3.0或更高版本上运行。在Github资源库中提供了库中每个命令的详细文档、示范性模拟的示例代码和Wiki。我们鼓励通过拉动请求和对额外功能的建议进行改进。

DyPy输出

每次模拟的结果都是一个数据集，由不同时期的策略频率及其相关的报酬组成。为了便于解释，这些结果可以通过一些不同的图形选项来直观地呈现。这些选项包括从单一模拟结果的二维图到三维线图和等高线图，这些图是在模拟的多次迭代中改变两个参数的结果（关于三维绘图选项的细节，见支持信息）。所有这些图都可以保存为matplotlib支持的任何文件格式，默认为.png。

定义一个游戏

EGT模拟有两个主要组成部分：正在进行的游戏和策略演变的动态。DyPy允许用户通过对游戏类进行子类化并适当定义报酬矩阵来创建任何想要的游戏。在创建游戏时，用户还可以定义感兴趣的状态（即玩每种策略的玩家的分布）--通常是为了分析游戏的可能均衡，这些均衡已经被分析确定。

为了说明如何做到这一点，并展示该库的功能，我们在本文中转载了一些与合作演变有关的文献中众所周知的结果。我们首先定义了囚徒困境。

囚徒的窘境

在所有规模的生物系统中都可以看到合作；从多细胞的形成到大规模的人类合作（Michod和Roze，2001；Nowak，2006b）。然而，为了他人的利益而付出个人代价的合作总是会被叛逃者所利用。著名的双人游戏 "囚徒困境"（Axelrod, 1980; Smith, 1982; Nowak, 2006a）反映了这一现象。与该博弈相关的回报矩阵是：。

$$\begin{array}{c} \\ C \\ D \end{array} \begin{bmatrix} C & D \\ R & S \\ T & P \end{bmatrix} \tag{1}$$

其中$T>R>P>S$。

像这样的游戏通常是通过寻找游戏的纳什均衡来解决的，纳什均衡是指没有任何一方可以通过单方面的偏离而获益的策略组合。这个游戏的唯一纳什均衡是双方都叛变，尽管如果双方合作，他们会得到更高的报酬。

在进化博弈论中，有一整个玩家群体，在每一轮中，他们被分配到彼此的游戏中--他们如何被分配到游戏中的细节取决于特定的动态。EGT中最基本的均衡概念是进化稳定策略（ESS）。如果当人口中的每个人都玩这个策略时，玩其他策略的突变体将获得较低的回报并死亡，那么这个策略就是ESS。ESS与纳什均衡紧密对应（事实上，所有ESS必须是纳什均衡，但反之则不然）。正如囚徒困境的唯一纳什均衡是（D，D），囚徒困境的唯一ESS是人口中的所有人都玩D（Smith，1982）。

现在，我们将使用DyPy模拟一群玩家玩囚徒困境，看看自然选择是否有利于叛逃。要做到这一点，我们首先设置游戏。这涉及到指定策略、平衡点和与之相关的报酬矩阵。以下16行代码被用来准备游戏。

```
从games.game导入SymmetricNPlayerGame

# 定义囚徒困境游戏的类。 class
PrisonersDilemma(SymmetricNPlayerGame):
    DEFAULT_PARAMS = dict(R=3,S=0,T=5,P=1,bias_strength=0)
    # 策略的列表
    STRATEGY_LABELS = ('Cooperate', 'Defect')
    # 平衡状态的列表
    EQUILIBRIA_LABELS=('合作', '变节')

   捍卫   init (self,R,S,T,P,bias_strength)。
        # 定义报酬矩阵 payoff_matrix
        = ((R,S),(T,P))
        super(PrisonersDilemma, self). init (payoff_matrix,1,bias_strength)

    @classmethod
    # 定义平衡的函数
    def classify(cls, params, state, tolerance):

        阈值=1-公差 如果
        state[0][0]>阈值。
            返回 0 # 合作 elif
        state[0][1] > threshold:
            return 1 # Defect
        else:
            return super(PrisonersDilemma, cls).classify(params, state, tolerance)
```

囚徒困境是一个对称的游戏，即所有玩家都有相同的策略和回报。
所以我们使用游戏类的SymmetricNPlayerGame子类。

类方法，'分类'被用来识别感兴趣的状态，在本例中，所有玩家合作的状态，以及所有玩家叛逃的状态。分类命令确定人口在一轮结束时是否 "处于 "这些状态。从技术上讲，它测试人口中的 "1-公差 "是否处于该状态，其中 "公差 "参数由用户选择（见SI中一个更复杂的例子，它使用游戏的参数来得出所需的公差）。如果系统处于用户定义以外的状态，分类命令将该状态分配为 "未分类"。这些状态不一定只与纯策略相关，可以包括由混合策略组成的稳定状态。

模拟的类型

一旦游戏被创建，用户可以选择随机的（Moran, Wright-Fisher）或决定性的（Replicator）动力学。GameDynamicsWrapper和VariedGame类负责结合所选择的游戏和动力学，并在给定的世代数和人口规模下运行所需的模拟。用户还可以指定一个 "起始状态"，它给出了初始策略频率（详见SI）。它默认为一个初始频率的随机列表，人口大致平均分配。
在所有的策略中。**表1**描述了DyPy中的一些重要类。我们提供了一些
以下是软件包中常用的模拟方法，并列举了一些例子。

对游戏动态的单一模拟

当用户希望在一次EGT模拟中分析指定代数（整个种群的更新步骤数）的策略动态时，可以使用 "模拟 "方法。在这两种随机动态中，用户还可以指定一个突变率，μ（更多细节见SI）。结果是每个玩家的策略随时间变化的动态图。

例如，我们模拟了囚禁者的策略。

级别	描述
游戏	囊括了游戏的理念，即是要 被模拟。用户可以创建这个类的一个子类，并通过指定回报矩阵、玩家数量和策略以及感兴趣的均衡来定义游戏。
动态性	动态管理的更新规则，从一个 辈到下一代。我们提供三种常用的动力学。Moran, Wright-Fisher和Replicator。用户也可以定义他们自己的。
游戏动力学封装器（ GameDynamicsWrapper	一个包装动态类的辅助类 和一个游戏类。它为具有固定的游戏和动力学参数集的模拟提供了辅助方法。
变化的游戏	一个帮助类，它包装了一个动态类和 一个游戏类。它在改变一个或多个参数的同时为模拟提供了辅助方法。

表1：重要的DyPy类的简要描述

在没有突变的情况下，使用莫兰动力学的两难游戏。输出是一个图，**图1A**。我们可以看到，动力学的结果趋向于叛逃，这是纳什均衡。示例代码。

```
from wrapper import
GameDynamicsWrapper from
dynamics.Moran import Moran
s = GameDynamicsWrapper(PrisonersDilemma,Moran,dynamics_kwargs={'mu':0})
s.simulate(num_gens=2000,pop_size=100,graph=dict( area=True,options=['smallfont'])
```

我们在补充资料中提供了一个使用确定性的Replicator动态的例子。

多次迭代模拟游戏的动态

在许多情况下，我们希望多次运行模拟，以检查在初始条件发生变化和/或由于动态的固有随机性而产生的结果的稳健性。游戏也可能有多个均衡，在这种情况下，用户可能会对人口在每个均衡中度过的世代的比例感兴趣。由于这些原因，我们提供了 "模拟多 "的方法，在这个方法中，模拟的多次迭代被运行，并返回每个结果均衡的频率。如果系统不在使用分类命令定义的状态的 "容忍度 "之内，模拟会返回 "未分类"。当进行多次模拟时，它们会自动在所有可用的内核上并行化。

我们现在使用 "模拟许多 "和随机的莫兰动态来证明，即使在存在随机性的情况下，玩 "囚徒困境 "的人群也会稳定在叛逃状态。

```
from wrapper import
GameDynamicsWrapper from
dynamics.Moran import Moran
s = GameDynamicsWrapper(PrisonersDilemma,Moran)
s.simulate_many(num_iterations=100, num_gens=2000,pop_size=100,/
graph=dict(area=True, options=['smallfont'])
```

该命令返回每个平衡点的频率的文本输出。

{'变节':1.0}。

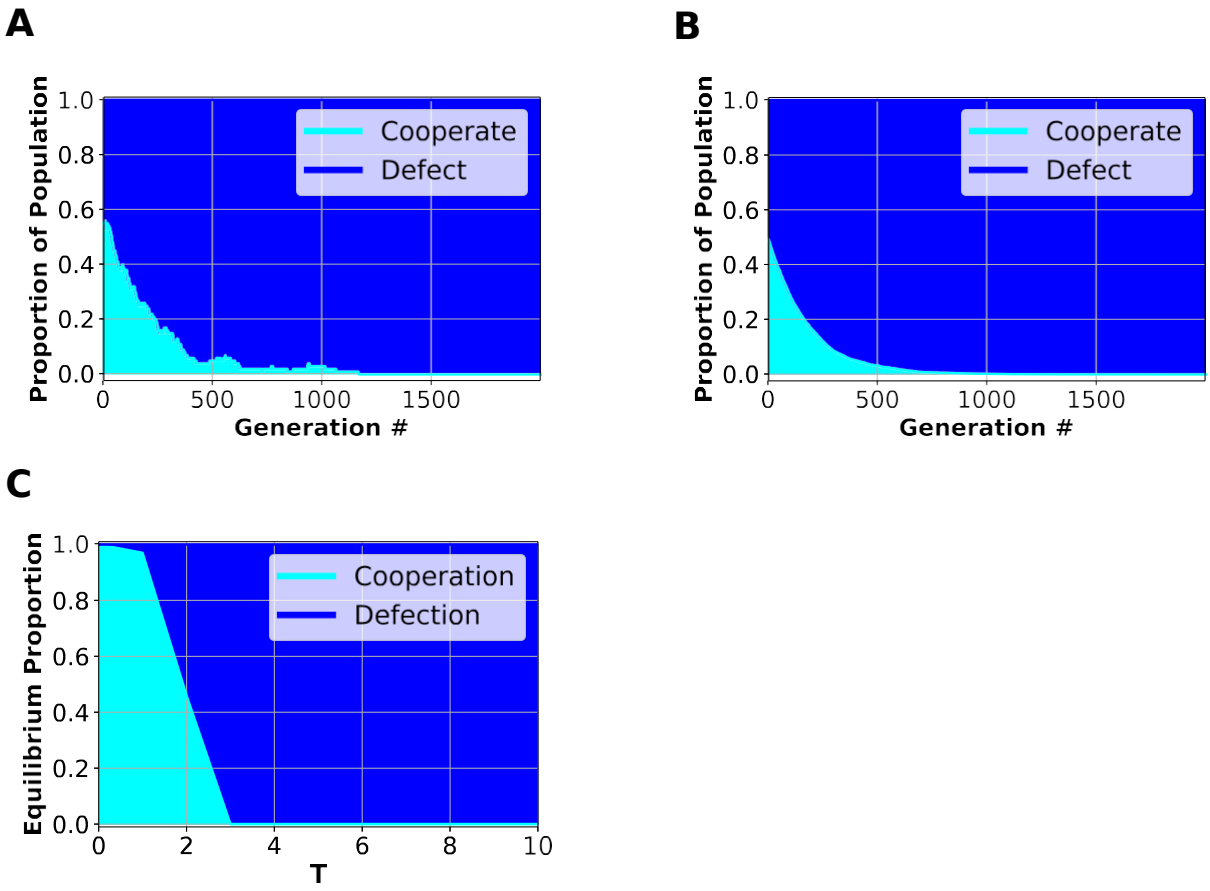

图1：使用DyPy中的三种主要方法对玩囚徒困境的群体进行模拟的结果。A）通过Moran过程演化2000代的策略。B) 通过Moran过程进行的2000代战略演变，平均100次迭代。C) 变化报酬T时的策略均衡比例

在莫兰动力学下的囚徒困境中。每个模拟运行了4000代，每个T值（1，2，...，10）平均100次迭代。我们在所有的模拟中使用了100个种群规模，囚徒困境中的报酬设定为R=3，S=0，T=5和P=1。

该模拟从未返回 "未分类"，并且在每次迭代中都能可靠地达到叛逃平衡。它还会返回一个图表（**图1B**），这是每个玩家的策略在一段时间内的动态迭代的平均值。在SI中，我们提供了一个例子，"模拟许多 "被用于一个游戏
(Repeated Prisoners' Dilemma)，表现出多个稳定的均衡。

近似的固定概率

frac invasions "方法计算了在一个种群中引入的策略在指定代数后支配其他策略的迭代比例。用户可以通过 "容忍度 "参数定义一个策略被认为是 "支配性 "的所需的普遍性数量。

我们包括这种方法，因为它可以用来近似计算策略的固定概率。在一个有两个策略A和B的游戏中，A的固定概率是指它在只有B玩家的群体中被引入时在群体中固定的概率。这一概念被广泛用于群体遗传学领域，例如，研究某一特定等位基因如何在群体中固定下来（Patwa and Wahl, 2008; de Oliveira and Campos, 2004; Lambert, 2006）。

一个策略的固定概率可以通过设置容忍度=0和运行适当代数的模拟来近似计算。这是为了确保模拟运行的时间足够长，使策略在种群中固定下来，或者种群达到其他的平衡状态。Hindersin等人（2019）对计算固定化概率的分析、数值和模拟方法进行了深入研究。所述技术可用于决定模拟的长度。以下几行代码可用于近似计算囚徒困境中 "缺陷 "的固定概率，公差设置为0。

```
from wrapper import
GameDynamicsWrapper from
dynamics.Moran import Moran
s = GameDynamicsWrapper(PrisonersDilemma,Moran)
s.frac_invasions(num_iterations = 1000, num_gens = 2000, \
pop_size = 100, strategy_indx = 1) # 策略 "defect "的索引
```

这将返回一个文本输出，其形式为:

```
所需策略在人口中占主导地位的运行比例=0.79
```

囚徒困境中策略的固定概率可以通过分析计算出来（Hin- dersin等人（2019））。对于我们模拟中使用的报酬和选择强度，"缺陷 "的固定概率为0.8，这与通过模拟得到的近似值非常接近。

改变参数的影响

我们提供 "变化 "方法来分析改变一个或多个与动力学或游戏相关的参数的效果。对于每一个被改变的参数值，都要进行多次迭代模拟，并记录每个平衡的最终频率。输出是这些最终频率与不同参数的函数关系图。作为一个例子，我们改变了囚徒困境游戏的报酬值T（1，2，...，10）。

```
from wrapper import VariedGame
from dynamics.Moran import Moran
s = VariedGame(PrisonersDilemma, Moran)
s.vari(game_kwargs={'T':[0,10,10]},
num_gens=4000,num_iterations=100,
graph=dict(area=True,options=['smallfont'])
```

输出，**图1C**显示了在固定的R=3的情况下，改变T的策略的均衡比例。当T>R时，该游戏是囚徒困境，因此，我们看到一旦如此，动态就会收敛为叛逃。

除了在具有基于报酬的学习的良好混合种群中进行EGT模拟外，我们还包括模拟频率偏向的模仿和群体选择的影响

频率偏向的模仿

使用进化动力学的动机之一是，人类会学习或模仿，而且，关键是，他们会优先学习或模仿成功的策略（M.Chudek 等 人 ， 2012 ； P.L.Harris 和 K.H.Corriveau ， 2011 ； G.Stenberg ， 2009 ； B.Galef ， 2008 ； K.Laland，2011），就像在生物进化中，自然选择偏爱成功策略。因此，我们到目前为止所关注的动力学（Replicator、Wright-Fisher和Moran）可以用来描述学习和模仿，就像它们被用来描述生物进化一样。

然而，人类有时会模仿*常见的*策略，这与它们是否成功有些无关（R.Boyd和P.J.Richerson，1985；Chudek等人，2015）。这被称为*频率偏向性模仿*。如果频率偏向是重要的，相对于成功偏向，这可以改变一个动态的行为方式，以及它将在哪里稳定下来。

由于这些原因，DyPy在分析游戏的动态时，有可能包括频率偏向的模仿。我们通过在游戏的平均回报中加入频率偏向的模仿

一个策略，一个用户指定的策略频率函数： $u^{FBI}(\sigma, \sigma', \ldots, \sigma') = u_i(\sigma, \sigma', \ldots, \sigma')(1 - \gamma) + \phi(\sigma)\gamma$ 其中$u_i(\sigma, \sigma', \ldots, \sigma')$是玩家$i$在群体中玩$\sigma$时的平均报酬率。对手的策略集$\{\sigma', \ldots, \sigma'\}$，$\gamma$是个体与频率偏向的相对强度。

(顺应性）模仿，ϕ（σ）是用户指定的σ的频率函数。这个函数默认为常用的函数（W.Nakahashi，2007）。

$$\phi(\sigma) = \frac{x_\sigma^a}{\sum_{j=1}^{M} x_j^a} \times s \tag{2}$$

其中，x_σ 是策略σ的频率，总和是玩家i可以采用的所有M个策略的频率，s是一个比例系数，a是Nakahashi（W.Nakahashi, 2007）定义的 "顺应性 "参数。a = 1是软件包中的默认值。在代码中，我们把s称为 "偏置比例"，γ称为 "偏置强度"。这种形式主义是(Molleman等人，2013)中所使用的形式主义的一个更一般的版本，在那里，一致性的影响被认为是一种协调游戏。

囚徒困境中的顺从主义

顺应主义是一种社会学习策略，参与者通过模仿大多数人学习策略（R.Boyd和P.J.Richerson，1985）。将顺应主义的影响加入到迄今为止所考虑的基于报酬的学习中，可以帮助稳定某些条件下的合作。我们可以通过VariedGame类来模拟囚徒困境游戏的这个系统，并改变顺从主义的强度来重现（Molleman等人，2013）的结果。模拟中使用的所有参数见SI。

```
from wrapper import VariedGame
from dynamics.wright_fisher import WrightFisher
s = VariedGame(PrisonersDilemma,WrightFisher)
s.variety(game_kwargs={'bias_strength':[0,1,10]},num_gens=100,num_iterations=100,/
parallelize=True,graph=dict(area=True,options=['smallfont']))
```

在**图**2A中，随着顺应性强度的增加，群体中出现了合作。当学习完全是顺应性的（γ=1），种群收敛到合作或叛逃完全取决于初始状态，因此我们看到两种策略在50%的时间内都固定下来。

组别选择

群体选择是指有时进化是通过自然选择在个体层面和群体层面发生的（Luo，2014），这导致了对群体整体有利的特征（策略）的青睐。在这种多层次（群体和个体）选择的情况下，一个游戏的动态可能不再收敛到纳什均衡。例如，（Traulsen和Nowak，2006）表明，在存在群体选择的情况下，在某些条件下合作比叛逃更有利。考虑到这一点，我们在DyPy中加入了模拟这种多层次选择动态的工具。

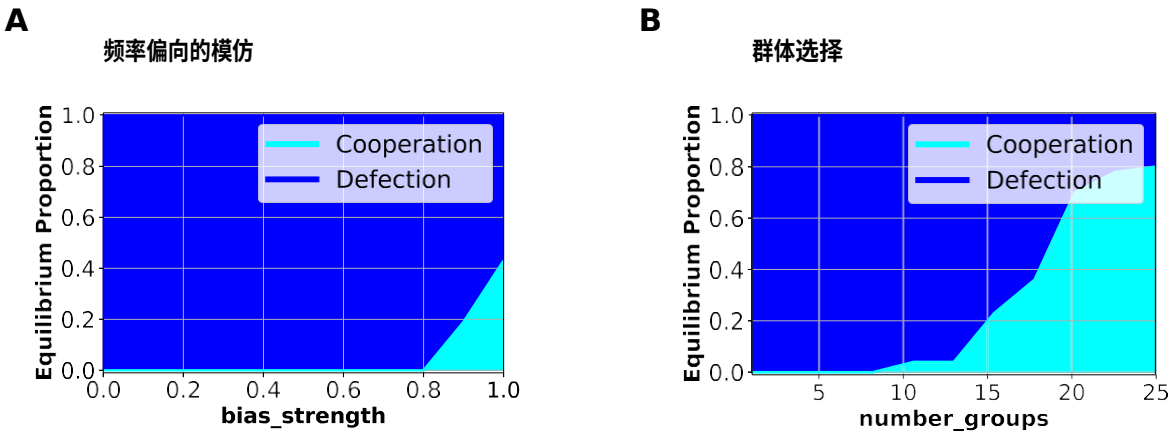

图2：在存在频率偏向的模仿和群体选择的情况下，策略的均衡比例。左图：在 Wright-Fisher动力学下的囚徒困境游戏中，改变顺应性的强度时，人口中出现均衡的比例。γ（偏差强度）从0（无顺应性）到1（完全顺应性学习）变化。每个模拟运行了100代，每个γ值平均进行了100次迭代。**右图。在**玩囚徒困境（赖特-菲舍尔动力学）的群体中，平衡出现的比例，在改变数量的时候
在总人口固定的情况下，小组的数量和小组的大小。组的数量从1到25不等。在每个组的数量上，模拟运行了100代，每代平均100次迭代。群体选择的 "比率 "为0.2。两次模拟的种群大小都被设定为100。

我们使用群体选择的形式主义（Luo，2014），包括固定数量的群体m，每个群体由固定的n大小的种群组成。在每个时间步，系统可以通过Moran或Wright-Fisher过程进行群体或个体水平的选择。这两个层次的选择之间的对立是通过 "速率 "来实现的，"速率 "是指在每个时间步长发生群体选择的概率。在个体层面上，繁殖与个体的体能成正比，而群体的繁殖则与他们的平均体能成正比。具体来说，在莫兰过程中的每一轮复制中，从整个种群中选择一个个体（群体）进行繁殖，与他们的体能成正比。另一方面，在Wright-Fisher过程中的每一轮复制中，所有个体（群体）都按其适配性比例进行繁殖。每当繁殖发生时，通过随机选择个体（群体）死亡来保持群体（群体）数量的固定。

囚徒困境中的群体选择

群体选择是另一种机制，通过这种机制合作可能在种群中得到稳定。在具有多级选择的种群中，可以有利于合作操作（Luo，2014；Traulsen和Nowak，2006），在某些条件下，群体之间的竞争会导致合作行为。据观察，较小的群体规模和大量的群体有利于合作者（Traulsen和Nowak，2006；Traulsen等人，2008）。我们可以通过使用VariedGame类和囚徒困境游戏在我们的软件包中模拟这一点。这种模拟改变了种群中的群体数量，保持了总种群规模的固定。我们在固定组数的情况下进行多次迭代模拟。

```
from wrapper import VariedGame
from dynamics.wright_fisher import WrightFisher
s = VariedGame(PrisonersDilemma, WrightFisher,dynamics_kwargs={'rate':0.2})
s.vari(dynamics_kwargs={'number_groups':[1,25,10]}, num_gens=100,\num_iterations=100,
graph=dict( area=True, options=['smallfont'] )
```

输出是一个图表（**图2B**），显示了模拟中达到稳定状态的次数比例。随着群体数量的增加和群体规模的减少，我们看到群体中出现了合作。

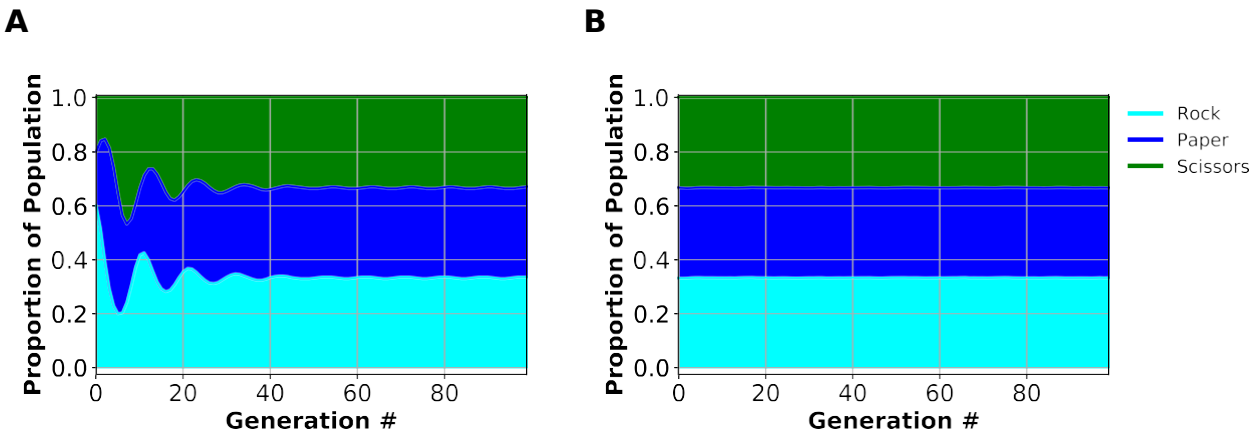

图3：使用Wright-Fisher动态方法玩石头剪刀布游戏的群体（n=500）的策略演变。(A)单次运行100代模拟的结果。(b)在3%的突变率下进行的500次模拟的平均结果。'simulate many'也返回文本输出，{'Nash':1.0}（见SI
如何定义），这意味着在每个迭代结束时都达到了纳什均衡。

剪刀石头布 "游戏

到目前为止，我们已经分析了囚徒困境的动态，在均衡状态下，所有玩家都玩*D*。现在我们简要介绍一下石头-剪子-布（RPS）游戏的动态，对于这种游戏，唯一的均衡状态是三分之一的玩家玩石头，三分之一玩纸，三分之一玩剪子。对游戏动态的分析揭示了一些有趣的东西，而均衡分析则无法做到这一点：人口经常循环一段时间，甚至永远循环，这主要取决于以下几个方面的报酬的相对大小
赢得与输掉一轮游戏（Hoffman等人，2015）。我们在**图3**中用一个
包括突变在内的随机过程，以及 "模拟 "和 "多次模拟 "方法（代码见SI
和使用的参数）。

讨论

本文介绍了DyPy，一个用于促进各种有用的EGT模拟的Python库，以及一些简单的例子来说明该库的功能。我们希望DyPy将被证明是有经验的和新手研究人员以及教师的一种资产。我们计划继续维护和扩展DyPy的功能，例如，增加额外的动力学，如强化学习和Fudenberg和Imhof（2006）的稀有突变的模仿过程，以及允许结构化种群。我们鼓励用户与我们联系，提出更多的功能建议，同时也鼓励用户在GitHub上提出拉动请求，以便帮助图书馆的发展。

作者投稿

DyPy是由EY和EL构思的。它最初是由EL用Python 2编写的，AF将其更新为Python 3并增加了一些功能。AN扩展了该软件包，以包括频率偏向的模仿、群体选择和其他功能。在EY和MN的指导下，AN领导了手稿的撰写。所有作者都对草案做出了贡献并同意发表。

鸣谢

我们要感谢Alexander Heyde、Christian Hilbe和Laura Schmid对稿件的有益讨论和编辑。这项工作得到了美国国立卫生研究院DP5OD019851（Anjalika Nande）的部分资助。

参考文献

Robert Axelrod.囚徒困境中的有效选择.*The Journal of Conflict Resolution*, 24(1):3-25, 1980.ISN 00220027, 15528766.URL http://www.jstor.org/stable/173932。

E.Whiskin B.Galef.钠和蛋白质缺乏率对社会信息的使用：对一个预测的检验（Boyd和Richerson 1988）。*动物行为学*，75：627-630，2008。

Boudewijn de Bruin. 哲学中的博弈论.*Topoi*, 24(2):197-208, Sep 2005.ISSN 1572-8749. doi: 10.1007/s11245-005-5055-3.URL https://doi.org/10.1007/s11245-005-5055-3.

M Chudek, M Muthukrishna, and J Henrich.文化进化。*载于《进化心理学手册》，第二版*，2015年。

Viviane M. de Oliveira and Paulo R.A. Campos.优势突变固定的动力学。*Physica A: Statistical Mechanics and its Applications*, 337(3):546 - 554, 2004.ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2004.02.007.url http://www.sciencedirect.com/science/article/pii/s0378437104002080。

Daniel Friedman.论进化博弈论的经济应用.*进化经济学杂志*，8（1）：15-43，1998年3月。ISSN 1432-1386. doi: 10.1007/s001910050054.URL https://doi.org/10.1007/ s001910050054。

Drew Fudenberg 和 Lorens A Imhof. 具有小规模突变的模仿过程.*Journal of Economic Theory*, 131(1):251-262, 2006.

G.Stenberg.婴儿社会指称的选择性。*Infancy*, 14:457-473, 2009.

Laura Hindersin, Bin Wu, Arne Traulsen, and Julian Garc´ıa.有限种群中进化游戏动力学的计算和模拟。*Scientific Reports*, 9(1):6946, May 2019.ISSN 2045-2322. doi: 10.1038/s41598-019-43102-z.URL https://doi.org/10.1038/s41598-019-43102-z.

Moshe Hoffman, Sigrid Suetens, Uri Gneezy, and Martin A Nowak.石头剪刀布游戏中进化动力学的实验调查。*科学报告*，5:8817，2015。

Gerhard Jaeger.博弈论在语言学中的应用。*Language and Linguistics Compass*, 2(3):406- 421, 2008. doi: 10.1111/j.1749-818X.2008.00053.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1749-818X.2008.00053.x.

Dominic D. P. Johnson, Pavel Stopka, and Stephen Knights.人类合作之谜。*自然》*, 421 (6926):911-912, 2003. doi: 10.1038/421911b.URL https://doi.org/10.1038/421911b.

M.Webster K.Laland, N.Atton.从鱼到时尚：对文化进化的实验和理论见解。*英国皇家学会哲学期刊B：生物科学》*，366：958-968，2011。

Amaury Lambert. 弱选择下的固定化概率。A branching process unifying approach.*Theoretical Population Biology*, 69(4):419 - 441, 2006.ISSN 0040-5809. doi: https://doi.org/10.1016/j.tpb.2006.01.002.URL http://www.sciencedirect.com/science/article/pii/S0040580906000116

。

Shishi Luo.一个统一的框架揭示了多层次选择的关键属性。*理论生物学杂志*, 341:41 - 52, 2014.ISSN 0022-5193. doi: https://doi.org/10.1016/j.jtbi.2013.09.024.URL http://www.sciencedirect.com/science/article/pii/S0022519313004542。

Akihiko Matsui.On cultural evolution:社会规范、理性行为和进化博弈论.*日本和国际经济杂志*，10（3）：262 - 294 ， 1996 。 ISSN 0889-1583. doi: https://doi.org/10.1006/jjie.1996.0015.url http://www.sciencedirect.com/science/article/pii/ s0889158396900155。

M.Chudek, S.Heller, S.Birch, and J.Heinrich.有名望偏见的文化学习：旁观者对潜在模型的不同关注影响了儿童的学习。*Evolution and Human Behavior*, 33:46-56, 2012.

Richard E. Michod and Denis Roze.多细胞性进化中的合作与冲突。*Heredity*, 86(1):1-7, 2001. doi: 10.1046/j.1365-2540.2001.00808.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1046/j.1365-2540.2001.00808.x.

L Molleman, I Pen, and FJ Weissing.顺应性对社会行为的文化演变的影响。*PLoS ONE*, 8(7):e68153, 2013. doi: 10.1371/journal.pone.0068153.

马丁-诺瓦克。*Evolutionary Dynamics*.哈佛大学出版社，2006a.

Martin A. Nowak.合作进化的五个规则。*Science*, 314(5805):1560-1563, 2006b.ISSN 0036-8075. doi: 10.1126/science.1133755.URL https://science.sciencemag.org/content/314/5805/ 1560.

Z Patwa and L.M Wahl.有益突变的固定概率.*Journal of The Royal Society Interface*, 5(28):1279-1289, 2008. doi: 10.1098/rsif.2008.0248.URL https://royalsocietypublishing.org/doi/abs/10.1098/rsif.2008.0248.

P.L.Harris和K.H.Corriveau.幼儿对线人的选择性信任。*皇家学会的哲学交易B：生物科学*，366：1179-1187，2011。

R.Boyd和P.J.Richerson.文化和进化过程。*芝加哥大学出版社*，1985年。

J.Maynard Smith和G. R. Price。动物冲突的逻辑。*自然》*, 246(5427):15-18, 1973. doi: 10.1038/246015a0.URL https://doi.org/10.1038/246015a0.

John Maynard Smith. *进化与游戏理论》* 。 Cambridge University Press, 1982. doi: 10.1017/CBO9780511806292.

Arne Traulsen and Martin A. Nowak.多层次选择的合作进化。*国家科学院院刊》*，103（29）：10952-10955，2006。URL https://www.pnas.org/content/103/29/10952。

Arne Traulsen, Noam Shoresh, and Martin A. Nowak.任何强度的个体和群体选择的分析结果。*数学生物学公报*，70(5):1410, 2008 年 4 月 .ISSN 1522-9602. doi: 10.1007/ s11538-008-9305-6.URL https://doi.org/10.1007/s11538-008-9305-6.

Arne Traulsen, Dirk Semmann, Ralf D. Sommerfeld, Hans-Ju¨rgen Krambeck, and Manfred Milinski.进化游戏中的人类策略更新。*美国国家科学院院刊》*，107（7）：2962-2966，2010。ISSN 0027-8424. doi: 10.1073/pnas.0912515107.URL https://www.pnas.org/content/107/ 7/2962.

W.Nakahashi.环境波动时社会学习中顺应性传递的演变。博士论文，2007年。

支持信息
DyPy:用于模拟矩阵形式游戏的Python库

Anjalika Nande, Andrew Ferdowsian, Eric Lubin, Erez Yoeli, Martin Nowak

1 简介

这个SI包括关于如何使用软件包的额外有用信息，并提供了比正文中的囚徒困境稍微复杂的游戏--重复囚徒困境的示例代码。此外，我们还提供了在正文中的一致性和石头剪刀布的例子中所使用的代码。我们还着重介绍了如何定义起始状态，纳入突变的影响，并列出了软件包中提供的其他图形选项。

2 定义起始状态

开始状态 "对应于用户可以指定的策略的初始频率。它是一个大小为$m \times n \times l$的多维列表，其中m是组的数量，n是玩家类型的数量，l是玩家策略的数量。例如，在正文中定义的囚徒困境游戏中，在没有分组选择的情况下，开始状态将是以下形式。

```
start_state = [[[60,40]]] 。
```

这意味着，在100个个体的总人口中，开始时有60个个体在玩 "合作"，而40个个体在玩 "叛逃"。这两种策略的顺序取决于游戏中的定义方式。在使用任何一种模拟方法时，都可以包括起始状态，例如：。

```
s.simulate(num_gens=100,pop_size=100,start_state=[[[60,40]]])
```

3 突变

我们提供了将突变的影响纳入两个随机动态的功能--赖特-菲舍尔和莫兰。用户可以提供一个适用于每个策略的通用突变率，或者提供一个突变率列表，每个玩家的每个策略都有自己的突变率。例如，在一个2人游戏中，玩家1有2个策略，玩家2有3个策略，突变矩阵的形式为。

```
mu = [[0.1,0.2],[0.1,0.2,0.3]]
```

并可以在初始化GameDynamicWrapper或VariedGame类时指定，例如。

```
s = GameDynamicsWrapper(PrisonersDilemma,Moran,dynamics_kwargs={'mu':[0.1,0.2],
[0.1,0.2,0.3]})
```

4 反复的囚徒困境

人们利用不同的方法来稳定标准囚徒困境中的合作（Nowak，2006）。重复 "游戏是指游戏不是只玩一次，而是在两个玩家之间重复多次。对于这样的游戏，有一些策略，如针锋相对（Tit-for-tat，TFT）策略，可以稳定地抵御总是缺陷（ALLD）的入侵（Axelrod，1980a，b）。TFT从合作开始，然后在随后的几轮中，无论对手在前一轮中采取什么策略，都会进行比赛。我们可以模拟一个玩TFT、ALLD和总是合作（ALLC）的群体，并表明TFT有助于在没有错误的情况下保持合作。在本节中，我们说明了模拟的结果。关于游戏（RepeatedPD）是如何编码的，以及我们如何定义预期均衡状态，请见下一节。我们首先用Replicator动态模拟人口。

```
s = GameDynamicsWrapper(RepeatedPD,Replicator)
s.simulate(num_gens = 100, pop_size = 100, graph=dict(area=True,options=['smallfont'])
```

这个模拟的结果是**图1A**，显示TFT帮助ALLC支配ALLD。接下来，我们检查这个最终状态在随机性存在下的稳健性。

```
s = GameDynamicsWrapper(RepeatedPD,Moran)
s.simulate_many(num_iterations = 100 , num_gens = 2000, pop_size =
100,\ class_end = True, graph=dict( area=True,options=['smallfont'])
```

该命令返回一个文本输出，包括每个平衡的比例。

{'合作平衡': 0.98，'非合作平衡': 0.02}。

图1B中的图是每一代的所有迭代的平均值。

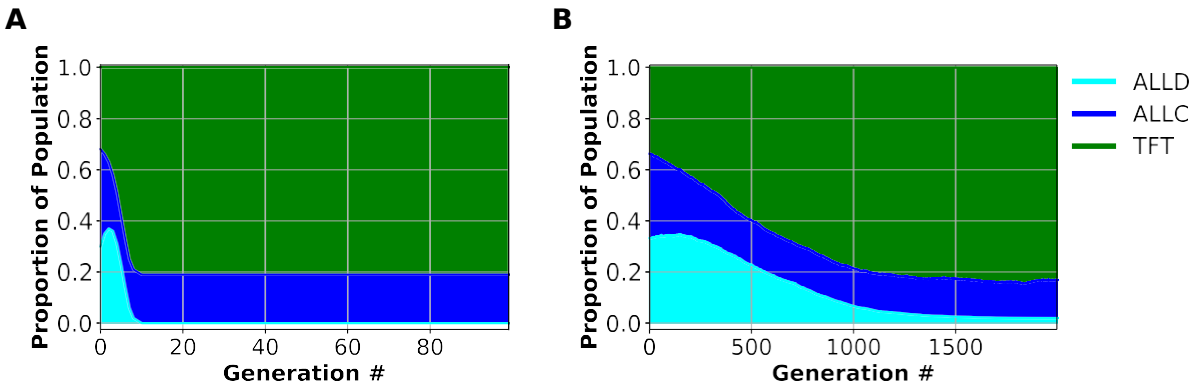

图1：重复玩囚徒困境策略的人群的策略演变。ALLD、ALLC和TFT。A）在Replicator动态下，100代。B）在Moran动态下进行了2000代，100次迭代的平均数。两次模拟的种群规模都设定为100。

5 反复囚徒困境的代码

总是缺陷（ALLD）、总是合作（ALLC）和针锋相对（TFT）属于被动策略集，这三种策略相互博弈的回报矩阵中的条目，公式（1）可以通过成熟的方法计算出来（Nowak，2006）。

$$\begin{array}{c} \\ ALLD \\ ALLC \end{array} \begin{array}{c} \begin{array}{ccc} 辽宁省 & 辽宁省 & TFT \end{array} \\ \begin{array}{ccc} P & T & P \\ S & R & R \end{array} \end{array} \tag{1}$$

触摸屏
P
R
R

这里$T>R>P>S$是与正文中公式（2）中的基本囚徒困境博弈相关的标准报酬。我们把 "合作 "均衡定义为系统的状态，即ALLD不能入侵TFT。如果ALLD玩家的体能高于TFT，ALLD就可以入侵TFT。在进化博弈论中，健身性被定义为与报酬成正比。因此，如果x_{ALLD}，x_{ALLC}和x_{TFT}分别是ALLD、ALLC和TFT玩家的频率，每个玩家的适配度由以下公式给出。

$$f_{ALLD} = P x_{ALLD} + T x_{ALLC} + P x_{TFT} \quad (2)$$

$$f_{ALLC} = S x_{ALLD} + R x_{ALLC} + R x_{TFT} \quad (3)$$

$$f_{TFT} = P x_{ALLD} + R x_{ALLC} + R x_{TFT} \quad (4)$$

只要$f_{ALLD} < f_{TFT}$，系统的状态就可以是 "合作的"。解答这个方程，我们发现系统在以下情况下处于合作平衡状态。

$$x_{ALLC} < \frac{(R - P)}{(T - R)} x_{TFT} \quad (5)$$

当这个条件没有得到满足时，ALLD可以入侵TFT，这导致了一个由叛逃者主导的 "非合作 "均衡。一旦报酬矩阵和均衡被定义，我们就可以对游戏进行如下编码。

```
from games.game import SymmetricNPlayerGame
class RepeatedPD(SymmetricNPlayerGame):

    DEFAULT_PARAMS = dict(R=3, T=5, S=0, P=1,
    bias_strength=0) STRATEGY_LABELS = ( 'ALLD', 'ALLC',
    'TFT')
    EQUILIBRIA_LABELS = ('Cooperative Equilibrium', 'Non Cooperative Equilibirum' )

    捍    init (self, R, T, S, P, bias_strength):

    卫
        payoff_matrix = ((P, T, P),
                         (S, R, R)。
                         (P, R, R))


        super(RepeatedPD, self). init (payoff_matrix, 1, bias_strength)


    @classmethod
    def classify(cls, params, state, tolerance):

        R   =   getattr(params,
        "R")        T        =
        getattr(params, "T") S
        = getattr(params, "S")
        P   =   getattr(params,
        "P")

        公差 = (R-P)/(T-R)
        如果state[0][1] <= tolerance * state[0][2] :
            返回0 # 合作平衡点
        elif state[0][1] > tolerance * state[0][2] :
```

```
    return 1 # Non-Cooperative Equilibrium
否则。
    return super(RepeatedPD, cls).classify(params, state, tolerance)
```

6 合规性示例的代码

为了将我们的结果与（Molleman等人，2013）进行定性比较，我们选择 "偏差函数 "在频率上是线性的，ϕ（x）=$2x$（"偏差尺度 "被设定为2），这样，¯ 一致性被模拟为一个协调游戏。

$$f_C \square (Rx_C + Sx_D)(1 - \gamma) + 2\gamma x_C \quad (6)$$
$$f_D \square (Tx_C + Px_D)(1 - \gamma) + 2\gamma x_D \quad (7)$$

$T>R>P>S$是与囚徒困境相关的通常报酬。与前面的例子一样，我们将系统的 "合作 "均衡定义为当叛逃者不能入侵合作者时，即$f_C > f_D$。求解公式（6），可以得出以下两个策略频率的合作均衡的条件。

$$x_C > x_D \frac{(P - S)(1 - \gamma) + 2\gamma}{(R - T)(1 - \gamma) + 2\gamma} \quad (8)$$

需要注意的是，只有当$(R - T)(1 - \gamma) + 2\gamma > 0$时，这样的$x_C$才存在，因为$x_C$，$x_D > 0$。用于生成游戏的精确代码。

```
from games.game import SymmetricNPlayerGame
class PrisonersDilemma(SymmetricNPlayerGame):
    DEFAULT_PARAMS = dict(R=3, S=0, T=5, P=1, bias_strength=0.0, bias_scale=2)
    STRATEGY_LABELS = （'合作', '缺陷'）。
    EQUILIBRIA_LABELS = ('合作', '变节')
    捍  init (self,R,S,T,P,bias_strength,bias_scale): payoff_matrix
    卫  = ((R,S), (T,P)
        super(PrisonersDilemma, self). init (payoff_matrix, 1, bias_strength,\
        bias_scale)

    @classmethod
    def classify(cls, params, state, tolerance):

        R   =   getattr(params,
        "R")        T        =
        getattr(params, "T") S
        = getattr(params, "S")
        P   =   getattr(params,
        "P")
        bias = getattr(params, "bias_strength")

        # 避免除以零，如果偏移量
        !=0.5。
            比率=((P-S)*(1-bias)+2*bias)/((R-T)*(1-bias)+2*bias) 否则。
            比例=1

        如果比率>0。
            如果state[0][0] > state[0][1]*ratio:
                返回0 # 合作
            elif state[0][0] <= state[0][1] *ratio:
                return 1 # Defection
            否则。
                return super(PrisonersDilemma, cls).classify(params, state, tolerance)
```

[1]'偏置函数'是在'PayoffMatrix'类的'获取预期报酬'方法中定义的。

如果比率<0。
返回1

7 剪刀石头布 "游戏

石头-剪子-布(RPS)是一种三段式策略游戏，其中存在着策略的循环支配。石头打败剪刀，剪刀打败纸，纸又打败石头。任何具有循环支配的对称三策略博弈都是RPS博弈的特征（Nowak（2006））。我们在模拟中使用了以下报酬矩阵。

$$\begin{array}{c} \\ R \\ P \\ S \end{array} \begin{array}{c} \begin{array}{ccc} R & P & S \end{array} \\ \begin{bmatrix} 0 & -0.2 & 2 \\ 2 & 0 & -0.2 \\ -0.2 & 2 & 0 \end{bmatrix} \end{array} \tag{9}$$

这个游戏的纳什均衡是一个混合策略，在纯策略上具有均匀分布，($\frac{1}{3}, \frac{1}{3}, \frac{1}{3}$)。用来生成该游戏的代码。

```
class RPS(SymmetricNPlayerGame):
    DEFAULT_PARAMS = dict(a1=0.2, a2=0.2, a3=0.2, b1=2, b2=2, b3=2)
    STRATEGY_LABELS = ('Rock', 'Paper', 'Scissors')
    EQUILIBRIA_LABELS = ('Nash', )
    捍    init (self, a1, a2, a3, b1, b2, b3):
    卫  payoff_matrix = ((0, -a2, b3), (b1, 0, -a3), (-a1, b2, 0))
        super(RPS, self). init (payoff_matrix,1)

    @classmethod
    def classify(cls, params, state, tolerance):

        阈值 = 1/3 pop_rock =
        state[0][0] pop_paper
        = state[0][1]
        pop_scissors = state[0][2]

        if abs(pop_rock-threshold) <= tolerance and
        \ abs(pop_paper-threshold) <= tolerance and
        \ abs(pop_scissors-threshold) <= tolerance。
            返回0 #Nash else:
            return super(RSP, cls).classify(params, state, tolerance)
```

如果人口收敛到纳什均衡，"分类 "方法返回 "纳什"，否则返回 "未分类"。我们将公差设置为0.03。用于生成模拟的单次迭代的代码。

```
s = GameDynamicsWrapper(RSP,WrightFisher)
s.simulate(num_gens=100, pop_size=500, start_state = [[[300,100,100]]],
graph = dict(area=True, options=['smallfont'])
```

我们用这个起始状态来说明，即使动力学开始时离平衡点更远，固定点也是稳定的。在有突变的情况下，用于生成模拟的多次迭代的代码。

```
s = GameDynamicsWrapper(RSP, WrightFisher,
dynamics_kwargs={'mu':0.03}) s.simulate_many(num_iterations = 500,
num_gens = 100, pop_size=500, graph =
dict(area=True,option=['smallfont'])
```

8 其他图表选项

该库提供了其他的绘图方法，这些方法并没有包括在正文中。我们用基本的囚徒困境游戏来展示如何使用这些方法。

8.1 柱状图

我们在 "模拟许多 "方法中提供了在迭代结束时绘制每个棋手策略分布直方图的选项。例如，下面的代码给出了**图2**的输出。

```
s = GameDynamicsWrapper(PrisonersDilemma, WrightFisher,dynamics_kwargs={'mu':[0.1,0.1]})
s.simulate_many(num_iterations = 100, num_gens = 100, histogram = True)
```

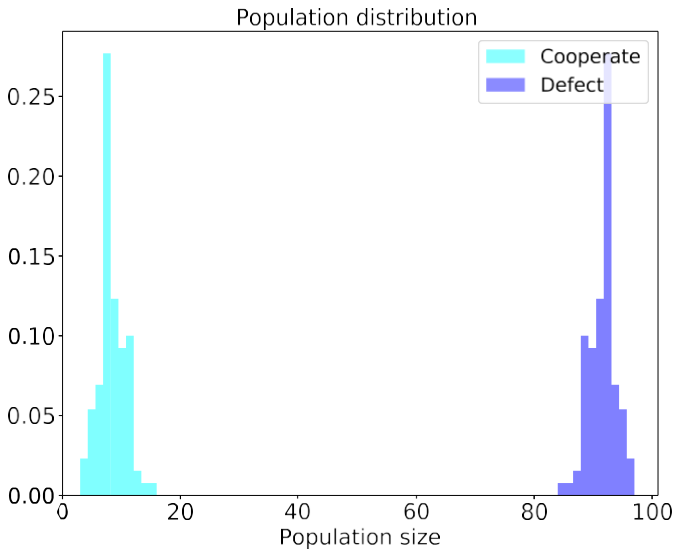

图2：Wright-Fisher动力学下囚徒困境中的策略的最终种群数量分布。结果是100代的100次迭代，种群规模为100。这是在两种策略都有10%的突变率的情况下。

8.2 等高线图

我们可以通过改变两个报酬，例如囚徒困境中的R和T，来创建一个等高线图。我们使用'vari 2params'辅助函数来改变游戏实例的两个参数。叛逃总是占优势的
只要T>R（根据囚徒困境的定义）就可以合作，所以我们在等高线图（**图3**）中包括*T=R*这条线，以供参考。

```
s = VariedGame(PrisonersDilemma, WrightFisher)
s.vari_2params('R', (0, 10, 20), 'T', (0, 10, 20), num_iterations=50,\)
num_gens=100, graph=dict(type='contour', lineArray=[ (0, 10, 0, 10) ])
```

当*T>R时*，该游戏是囚徒困境，"叛逃 "是唯一的纳什均衡。当*R>T*，因为*S<P*，系统是双稳态的（Nowak，2006）。在这种情况下，根据初始条件，系统要么收敛为合作，要么收敛为叛逃。

8.3 三维导线图

我们还可以在改变游戏的2个参数时产生3D线框图（**图4**）。如上所述，改变R和T，代码中唯一的变化是指定要创建的图的类型。

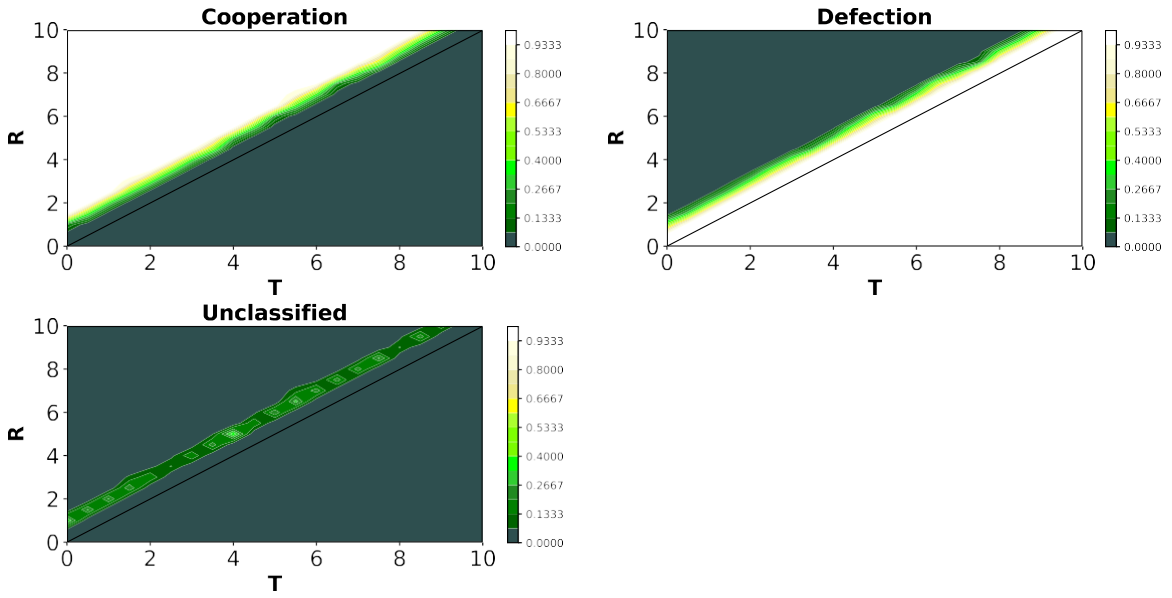

图3：使用Wright-Fisher动力学改变参数*R*和*T*时的均衡分布的等高线图。当*T*>*R*时，游戏是囚徒困境，"叛逃 "是唯一的纳什均衡。未分类 "对应于混合均衡，两种策略在人口中共存。模拟运行了100代
每对*R*[0, 10]和*T*[0, 10]值的结果是50次迭代的平均值。黑色实线对应的是*R*=*T*。

```
s = VariedGame(PrisonersDilemma, WrightFisher)
s.vari_2params('R', (0, 10, 15), 'T', (0, 10, 15),num_iterations=50,//。
num_gens=100, graph=dict(type='3d', lineArray=[ (0, 10, 0, 10) ])
```

x和y轴对应于被改变的参数，而平衡比例则绘制在z轴上。

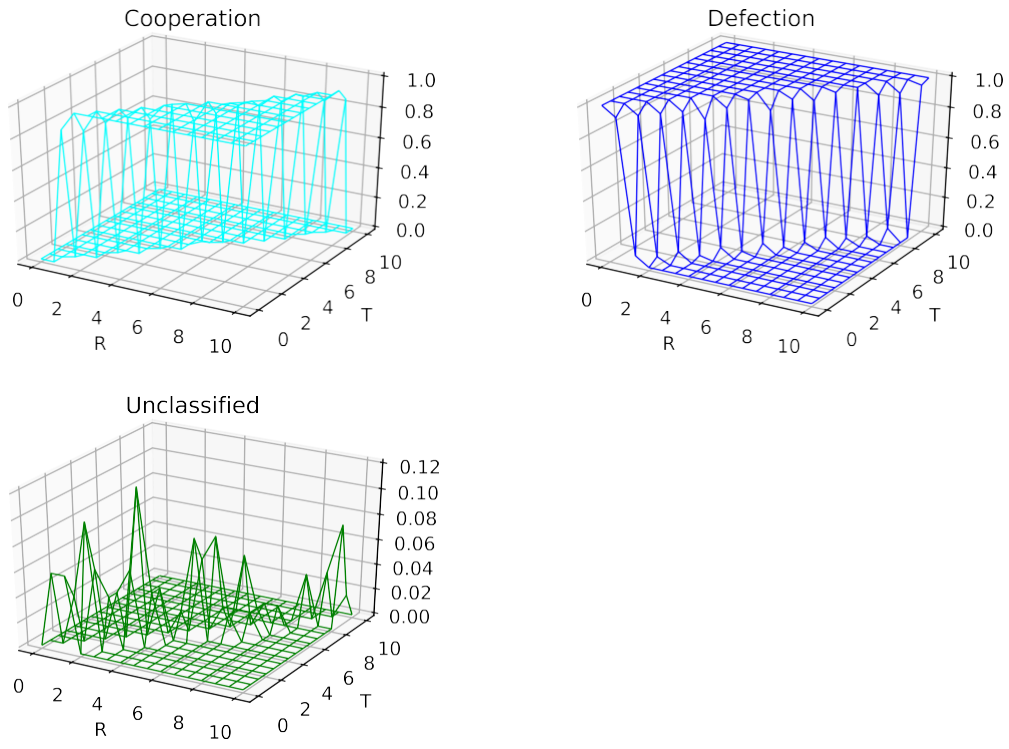

图4：使用Wright-Fisher动力学，在囚徒困境中改变R和T的平衡分布的三维线图。模拟运行了100代，每对R[0, 10]和T[0, 10]值的结果都是50次迭代的平均值。

参考文献

Robert Axelrod.囚徒困境中的有效选择.*The Journal of Conflict Resolution*, 24(1):3-25, 1980a.ISN 00220027, 15528766.URL http://www.jstor.org/stable/173932。

Robert Axelrod.囚徒困境中更有效的选择.*冲突解决杂志*，24（3）。379-403, 1980b. doi: 10.1177/002200278002400301.

L Molleman, I Pen, and FJ Weissing.顺应性对社会行为的文化演变的影响。*PLoS ONE*, 8(7):e68153, 2013. doi: 10.1371/journal.pone.0068153.

马丁-诺瓦克。*Evolutionary Dynamics*.哈佛大学出版社，2006年。