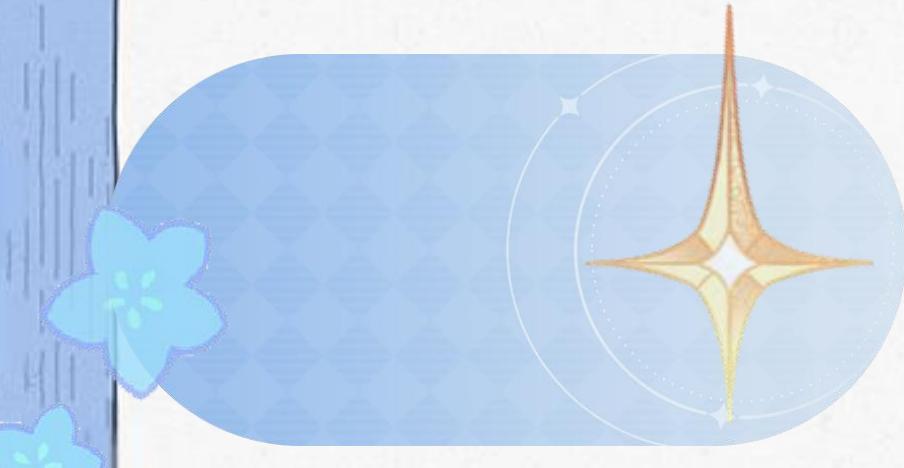




抽卡游戏概率分析

抽卡概率计算代码介绍



目录

抽卡游戏简介

01

抽卡原理

02

抽卡分析

03

代码简介

04



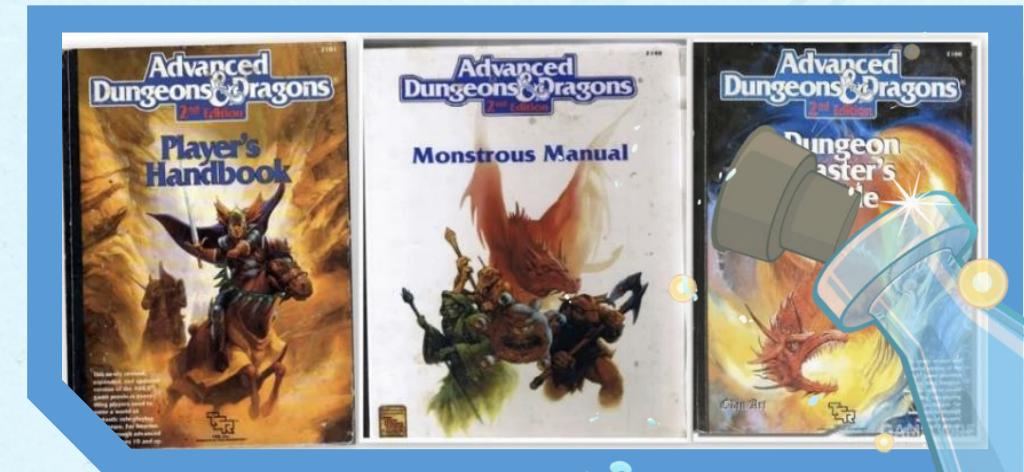
01 抽卡简介

随着设计和技术水平的提升，中国游戏市场空前庞大。今年1-6月，国内游戏市场实际销售收入规模为1442.63亿元，同比下降2.39%，环比增长22.2%，市场回暖趋势明显，庞大的销售收入数字，离不开创新付费模式设计对产品运营的拉动帮助。如今炙手可热的“抽卡”设计，就是很多游戏产品中的一种主要付费模式。

“抽卡”游戏英文翻译为 Gacha game，指有随机机制设计的游戏。通常将游戏内的核心资源，根据不同的等级（稀有度），以不同比例置入一个奖池中，玩家付费随机抽取这些资源的游戏类型。不同游戏的抽卡设计，基本都具有如下特点：有几率通过少量费用获取游戏最高级（稀有）的资源；每次抽卡都具有随机性；低等级（低稀有度）的资源在抽卡系统中的占比比较高。随着游戏长期进行更新，推出越来越多的新角色，玩家也就被驱使着不断进行抽卡乃至付费抽卡，从而构成了游戏完整的付费逻辑。

抽卡机制的最早来源是日本的Gacha，也被成为扭蛋。即是基于颇为流行的自动贩售机原理，当投入钱币后会得到随机的奖励。

最早使用这种机制的是TSR的龙与地下城和威世智的万智牌，这两款游戏将“gacha”融入其中，玩家购买的则是一个不知道其中有什么卡的卡包。每个玩家都需要购买实体卡包，但每个实体卡包未必能实现自己希望的套卡，所以慢慢的很多玩家通过交换来换得自己想要的实体卡片，这也就是TCG（Trading card game）交换式卡牌的由来。



虚拟抽卡游戏最早可以追溯到
2012年的《扩散性百万亚瑟王》，
由史克威尔，在2015年传入中国
后，中国的抽卡游戏呈现了爆发
式增长，出现多个爆款游戏，如
阴阳师，明日方舟，O神，同时大
量主流玩法无关抽卡的游戏也将
稀有道具通过抽卡方式放出



(一) 含有宣扬赌博内容

根据《网络游戏虚拟货币监管和执法要点指引》，目前含宣扬赌博内容的网络游戏的主要形式：

(1) 网络游戏运营企业在用户直接或变相投入法定货币或虚拟货币的前提下，采取抽签、押宝、随机抽取等偶然方式获得游戏道具或网络游戏虚拟货币。

(2) 网络游戏运营企业在用户直接或变相投入法定货币或虚拟货币的前提下赠予积分供用户使用，并利用该积分采取抽签、押宝、随机抽取等偶然方式分配游戏道具或网络游戏虚拟货币。

因此，如果参与游戏抽卡的条件是投入法定货币或虚拟货币，则可能被认定为宣扬赌博内容。

(二) 未公示概率，与公示内容不符

根据文化部要求，网络游戏运营企业应当及时在该游戏的官方网站或者随机抽取页面公示可能抽取或者合成的所有虚拟道具和增值服务的名称、性能、内容、数量及抽取或者合成概率。公示的随机抽取相关信息应当真实有效。若游戏抽卡未公示概率，或抽卡概率与公示内容不符，均会涉嫌违规。

比如北京掌握时代科技有限公司《掌握江湖-仗剑天涯》抽奖活动“关扑”未公示所有可能抽取的概率，胡莱游戏的《妖怪正传》游戏内抽卡概率涉嫌与公示内容不符，均被认定为涉嫌存在通过随机抽取等偶然方式诱导网络游戏用户采取投入法定货币或者网络游戏虚拟货币方式获取网络游戏产品和服务的违法行为，从而受到罚款处罚。

如今炙手可热的“抽卡”设计，就是很多游戏产品中的一种主要付费模式。但“抽卡”模式火爆的表象下，潜藏着机制不合理、概率不透明、监管不完善等问题；有些产品的“抽卡”设计甚至侵害了消费者权益。

玩家告厂商欺诈，厂商反诉恶意诉讼

此次案件被告是话题二次元手游《战双帕弥什》研发及运营商库洛游戏，原告是一名《战双帕弥什》玩家，于2019年12月4日，至2019年12月15日累计充值2000元，购买被告运营的《战双帕弥什》游戏内“虹卡”。原告玩家以“S级构造体”心理预期抽取概率1.9%，但实际为0.5%为由提起诉讼，要求法院判令原告库洛游戏退还人民币2000元，赔偿人民币6000元以及承担诉讼费用。



请输入案件名或案号或法官名

首页 庭审直播 庭审预告 直播回顾 庭审录播 重大案件 热点排行 普法教育 法院导航 数据公开

直播 民事 李根诉广州库洛科技有限公司网络服务合同纠纷
广州互联网法院

关注法院

抽卡原理

0
2

游戏的抽卡原理是一种利用随机数生成器（RNG）来决定玩家能够获得的游戏内容的机制。RNG是一种能够产生不可预测的数字序列的算法，通常基于一些复杂的数学公式或物理现象。游戏中的抽卡机制会根据不同的RNG结果，给予玩家不同的奖励，比如角色、道具、资源等。

这些奖励通常有不同的稀有度和价值，因此玩家会有不同的期望和满意度。游戏开发者会根据游戏的设计目标和玩家的反馈，来调整抽卡的概率和规则，以达到平衡游戏的难度和乐趣，以及提高玩家的投入和留存。

随机数生成器可以分为两大类：真随机数发生器（TRNG）和伪随机数发生器（PRNG）。

真随机数发生器（TRNG）是一种用于生成随机数的设备，其优点是产生的随机数具有高质量和不可预测性，但是缺点是速度较慢、成本较高、难以复制和验证。真随机数发生器一般用于安全敏感的场合，比如加密、认证、抽奖等。

伪随机数发生器（PRNG）是一种用于生成随机数的算法，其输出的随机数是基于一些数学公式或规则产生的，这些公式或规则通常需要一个初始值，称为种子（seed）。伪随机数发生器的优点是产生的随机数具有高速度和低成本，而且可以复制和验证，但是缺点是产生的随机数具有周期性和可预测性，因此不适合用于安全敏感的场合。伪随机数发生器一般用于模拟、游戏、测试等场合。

游戏中大多采用的是伪随机数发生器（PRNG），因为它们能够满足游戏的性能和效果的需求，而且可以通过一些技巧来提高随机数的质量和玩家的体验。

比如，可以通过调整随机数的分布、范围、频率、保底等来平衡游戏的难度和乐趣，也可以通过保存随机数的状态、种子、结果等来保证游戏的可复现性和可调试性。

保底机制

保底机制是一种在游戏中抽奖时，为了保证玩家的抽奖体验和满意度，而设置的一种提高中奖概率的机制。不同的游戏可能有不同的保底规则，但一般都是基于抽奖次数和中奖概率的关系。

如《王者荣耀》，购买次数到达361次时，荣耀水晶产出概率为100%。《剑与远征》两个保底机制，30抽必出紫卡，在同卡池内累计抽30次即可获得出一张紫卡英雄不论是单抽还是连抽，只要数量达到即必出紫卡。可以说，保底机制保证了玩家的最终体验内容。

水位机制

概率递增法，是指抽卡时，抽卡次数越多，爆率越高的抽卡方法。

这个机制作用的原理也很好理解，我们假设一个“水位值” a ，它的初始值是0。每次抽卡计算概率时， a 都会作为一个附加值加进中奖概率中，也即 a 值越大，中奖概率当然就越高。而当一次抽卡没有中奖时，就进行一次类似于 $a=a+1$ 的累计计算。（这里只是用简易模型解释，实际情况中要复杂得多）这样下一次 a 的值必然会比上一次的 a 值大，相应的你中奖概率就会更高，如此循环下去直到你抽中大奖，则 a 值归零，重新计算。

通过水位机制就能确保经常抽卡的玩家不会“一直非酋”下去，从而获得良好的体验。

事先声明:

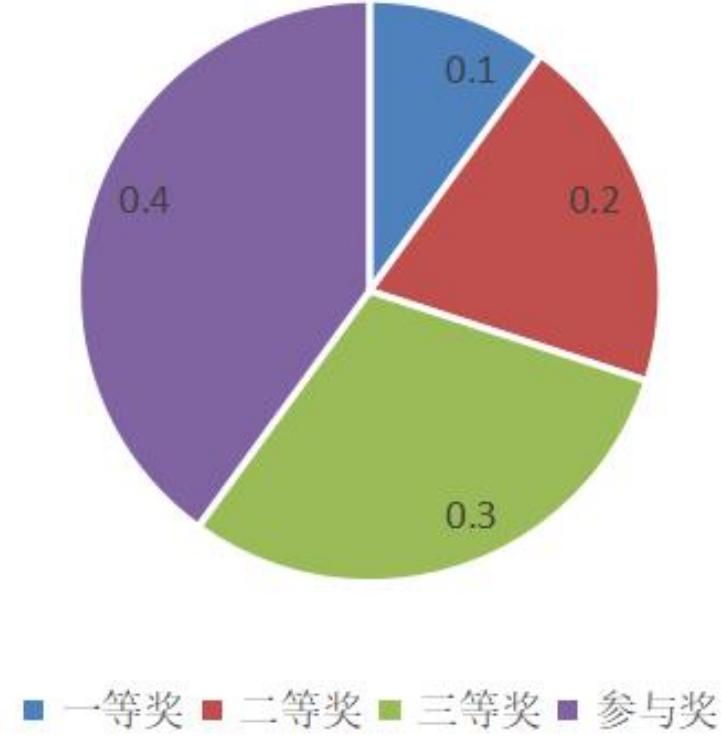
具体的抽卡分析资料均为互联网提供,仿真程度可能略有差错,仅供参考
游戏厂商大部分都是不会公布完整的抽卡算法的,他们会公布出奖的概率
但不会公布背后的机制,游戏厂商确实可能存在欺诈行为.

本部分将以原神为案例,通过游戏内部的相关规则以及一定的数据集
去分析抽卡的整个模型以及抽卡抽中高稀有品质物品的概率.

03

抽卡分析





首先介绍一个简单的抽卡模型:轮盘选择法:
轮盘选择法是经常应用于随机选择权重不同的多类物品的算法。类似于超市中常见的大转盘抽奖，转动转盘，转盘停下的位置是随机的，因此指针指向某类物品的概率正比于某类物品所占面积。

如果完全根据权重来进行选择，实现让玩家必得某类物品的功能会比较繁琐。因此对朴素的轮盘选择法进行一点改变，设置权重和的上限 W_{ceil} ,选取的随机数落在

$[1, \min \sum_{i=1}^n W_i, W_{ceil}]$,并对物品按照从高优先级到低优先级的顺序编号。即是当权重和不超过上限时，概率按照权重划分。当权重和超过上限时，优先级低的物品权重会被截断一部分。这样修改后，想要保证玩家必得某类物品时只需将其优先级设置为最高，并将其权重设置超过 W_{ceil} ,即可实现简洁的获得某类物品的规则

以近期原神4.2版本公布的芙宁娜的人物类卡池中的规则进行详细分析规则并给出相关概率:其中5星中奖率以及是否抽中为5星限定角色会分开介绍

由此分析可知,原神UP池概率机制如下:

在[1,73]区间抽时,每次出五星角色的概率为0.6%,在[74,90]抽每次抽卡概率比上次高6%

当获取到五星角色时,有50%概率是UP角色,若这次五星没有抽到UP角色,下次五星角色必定是UP角色

五星角色保底为90抽,五星UP角色保底为180抽.

■祈愿规则■

【5星物品】

在本期「众水的颂诗」活动祈愿中,5星角色祈愿的基础概率为**0.600%**,综合概率(含保底)为**1.600%**,最多**90**次祈愿必定能通过保底获取5星角色。

当祈愿获取到5星角色时,有**50.000%**的概率为本期5星UP角色「不休独舞·芙宁娜(水)」。如果本次祈愿获取的5星角色非本期5星UP角色,下次祈愿获取的5星角色**必定**为本期5星UP角色。

【4星物品】

在本期「众水的颂诗」活动祈愿中,4星物品祈愿的基础概率为**5.100%**,4星角色祈愿的基础概率为**2.550%**,4星武器祈愿的基础概率为**2.550%**,4星物品祈愿的综合概率(含保底)为**13.000%**。最多**10**次祈愿必定能通过保底获取4星或以上物品,通过保底获取4星物品的概率为**99.400%**,获取5星物品的概率为**0.600%**。

当祈愿获取到4星物品时,有**50.000%**的概率为本期4星UP角色「朗镜索真·夏洛蒂(冰)」、「萃念初蘖·柯莱(草)」、「无冕的龙王·北斗(雷)」中的一个。如果本次祈愿获取的4星物品非本期4星UP角色,下次祈愿获取的4星物品**必定**为本期4星UP角色。当祈愿获取到4星UP物品时,每个本期4星UP角色的获取概率均等。



决定抽到物品等级的机制

“保底”机制

每抽卡一次进行一次轮盘选以确定抽到什么等级的物品，常驻祈愿和角色活动

祈愿各类物品的权重如下：

$$W_{\text{常驻和角色祈愿五星}}[i] = \begin{cases} 60 & (i \leq 73) \\ 60 + 600 \cdot (i - 73) & (i \geq 74) \end{cases}$$

$$W_{\text{常驻和角色祈愿四星}}[j] = \begin{cases} 510 & (j \leq 8) \\ 510 + 5100 \cdot (j - 8) & (j \geq 9) \end{cases}$$

$$W_{\text{三星物品}} = 9430$$

模型中有两个计数器*i*和*j*，其中*i*表示此前已经连续*i*-1抽没有抽到五星物品，*j*表示此前已经连续*j*-1抽没有抽到四星物品。需要注意，每种类型的祈愿都有独立的计数器，且四星五星计数器独立。保底机制中，五星优先于四星，四星优先于三星。或许是为了程序实现方便，四星和五星物品的实际综合概率均略高于公示概率。



假设抽卡时的概率是这样变化的：

在抽数 t 小于等于某个固定的整数 M 时，概率保持基础概率不变。

在抽数 t 大于 M 时，概率呈线性上升直至在 $t = W$ (保底) 时达到 1。

于是，对于常驻池与角色UP池，我们有如下函数：

$$P(t) = \begin{cases} P_c & t \leq M \\ 1 - P_c * (t - M) / (W - M) + P_c & t > M \end{cases}$$

其中 P_c 为基础概率, W 为保底次数

对于第 t 次抽取，我们可以计算出在此次抽奖中抽中五星的概率：

第 t 次抽到 = 之前抽不到 * 本次抽到

$$Q(t) = \left(\prod_{i=1}^{t-1} 1 - P(i) \right) * P(t)$$

抽到的五星期望概率 E 以及综合概率 P_v 为：

(白图为计算后的相关数据)

$$E = \sum_{i=1}^W (Q(i) * i) \quad P_v = 1/E$$

1. UP角色池与常驻池

$$P_c = 0.006, M = 73, W = 90$$

$$P(t) = \begin{cases} 0.006, & t \leq 73 \\ 0.994 \times \frac{t - 73}{90 - 73} + 0.006. & t > 73 \end{cases}$$

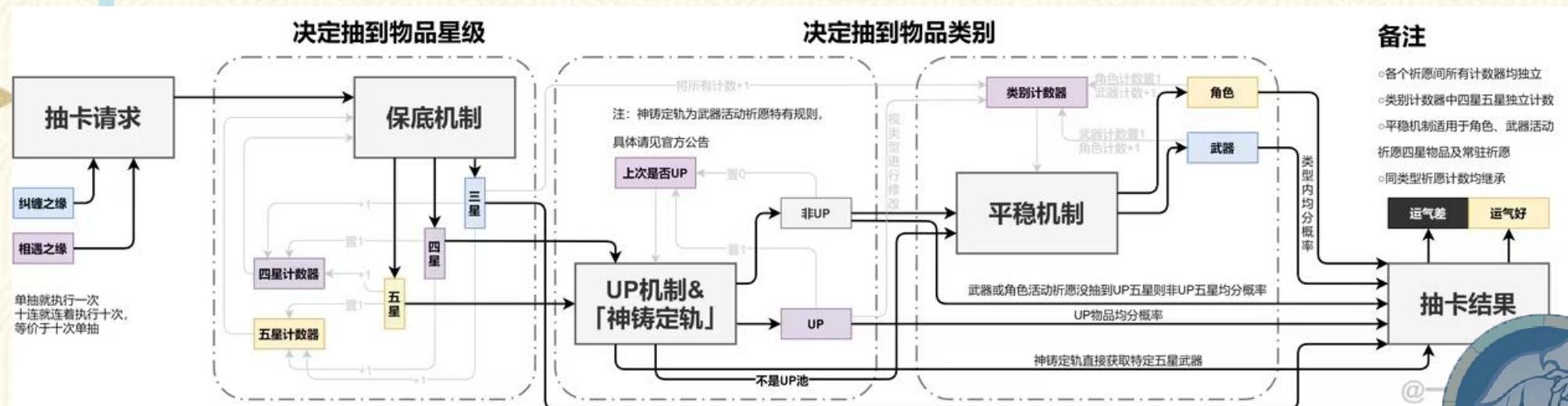
$$E = 62.34$$

$$P_v = 1.604\%$$



机制总结

原神的抽卡机制是一个概率有限状态机，确定当前状态则确定当前抽卡概率，状态之间依照概率进行转移。抽卡机制可总结为下图。注：十连等价于连着单抽十次。



在一个随机过程 (Random process) 中，只需要知道当前状态，就可以计算出未来状态，而不需要知道所有的历史路径。

比如，我只需要知道你这抽是第75抽，没抽中。而我不需要知道你所有的抽卡记录，就可以算出你下抽中5星物品的概率。

马尔可夫链（英语：Markov chain），又称离散时间马尔可夫链（discrete-time Markov chain，缩写为DTMC^[1]），因俄国数学家安德烈·马尔可夫得名，为状态空间中经过从一个状态到另一个状态的转换的随机过程。该过程要求具备“无记忆”的性质：下一状态的概率分布只能由当前状态决定，在时间序列中它前面的事件均与之无关。这种特定类型的“无记忆性”称作马尔可夫性质。马尔科夫链作为实际过程的统计模型具有许多应用。

在马尔可夫链的每一步，系统根据概率分布，可以从一个状态变到另一个状态，也可以保持当前状态。状态的改变叫做转移，与不同的状态改变相关的概率叫做转移概率。随机漫步就是马尔可夫链的例子。随机漫步中每一步的状态是在图形中的点，每一步可以移动到任何一个相邻的点，在这移动到每一个点的概率都是相同的（无论之前漫步路径是如何的）。



关于极端情况下的解释(例如十连抽卡出多个5星角色的情况/连续多次抽到极限情况才能出up五星角色的情况的可能性):

如前面所说,抽卡每一抽是独立计算的,十连抽理论上是和单抽没区别的,一旦在某一次抽出一个五星,将立即重置概率. 所以不存在十连比十次单抽更容易出双金的说法.

设有相互独立的随机变量序列 $\{X_i\}$, 期望存在, $E(X_i) = \mu_i$, 方差存在且有共同上界,

$$D(X_i) = \sigma_i^2 < M, \text{ 则 } \forall \varepsilon > 0, \text{ 有 } \lim_{n \rightarrow \infty} P \left\{ \left| \frac{1}{n} \sum_{i=1}^n X_i - \frac{1}{n} \sum_{i=1}^n \mu_i \right| < \varepsilon \right\} = 1$$

切比雪夫大数定律揭示了随着样本容量的增加, 样本平均数将接近于总体平均数的规律. 因此越抽越非的问题, 实质上是随着抽数的不断增加, 样本平均数向总体平均数回归的过程. 而根据切比雪夫大数定律这里的总体平均数可以由实验次数趋于无穷时的样本平均数来计算出.



大数定律

代码实现

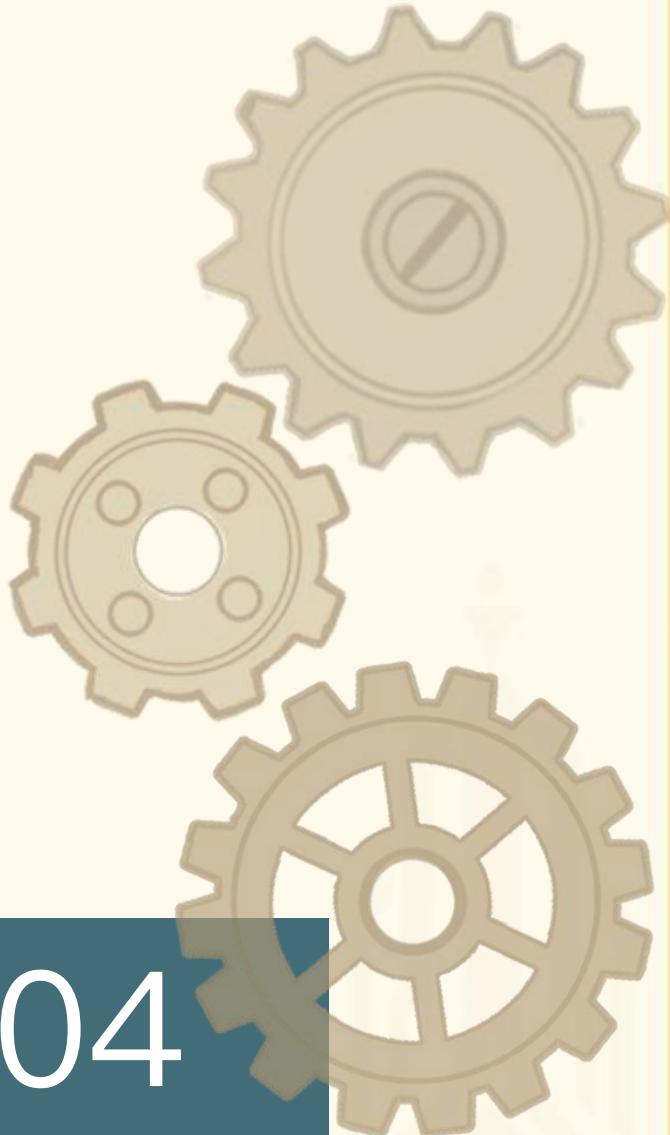


研究方法

概率分布

代码简介

04



辛钦大数定律

设 $X_1, X_2, \dots, X_n, \dots$ 是相互独立的随机变量序列，服从相同分布，且具有有限的数学期望，则对于任意的 $\varepsilon > 0$ ，有：

$$\lim_{n \rightarrow \infty} P\left\{ \left| \frac{1}{n} \sum_{i=1}^n X_i - u \right| < \varepsilon \right\} = 1$$



辛钦大数定律从理论上指出：用算术平均值来近似实际真值是合理的。而在数理统计中，这一定律使得用算术平均值来估计数学期望有了理论依据。

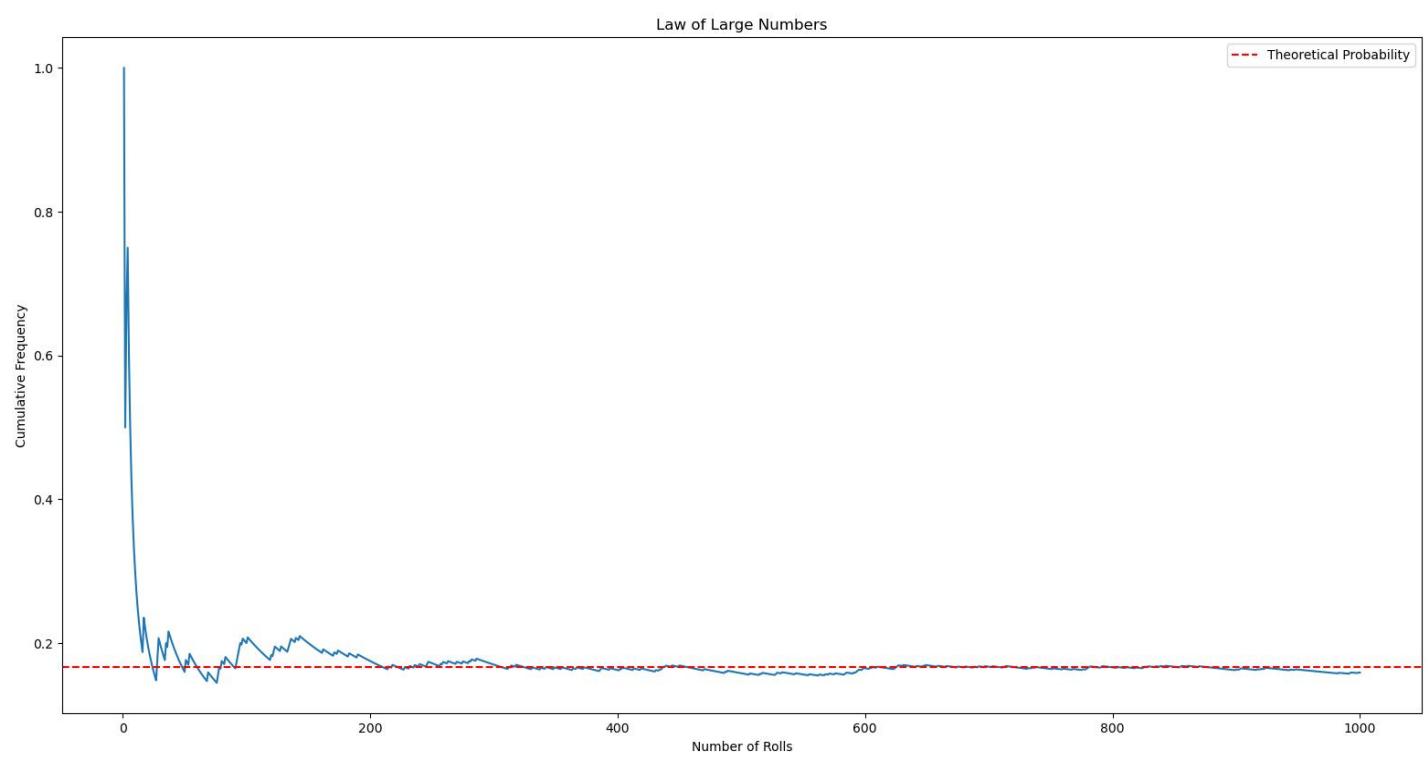






小展台

掷色子模拟小实验



随着次数的增加，频率会不断向理论期望值靠拢！





原神抽卡分布 ——— 研究方法

基于辛钦大数定律，采用简单大数据模型对数据解读，并分析基本信息，如最大值，期望值，平均数。



——抽卡资料截止至2.0版本

本次实验针对《原神》中20万抽卡数据资料，对角色池，武器池进行解析。

期望值与概率

若根据官方抽卡概率来看，《原神》抽卡的期望值是可计算的

$$P(\text{bottom}) + 0.006(1 - P(\text{bottom})) = P(\text{success})$$

$$P(\text{success})(1 - 0.006)89 = P(\text{bottom})$$

$$\text{解得 } P(\text{success}) = 0.016 + -0.000005$$



所以理论概率P为0.016。
经过验证，与官方所给定的数据相符。

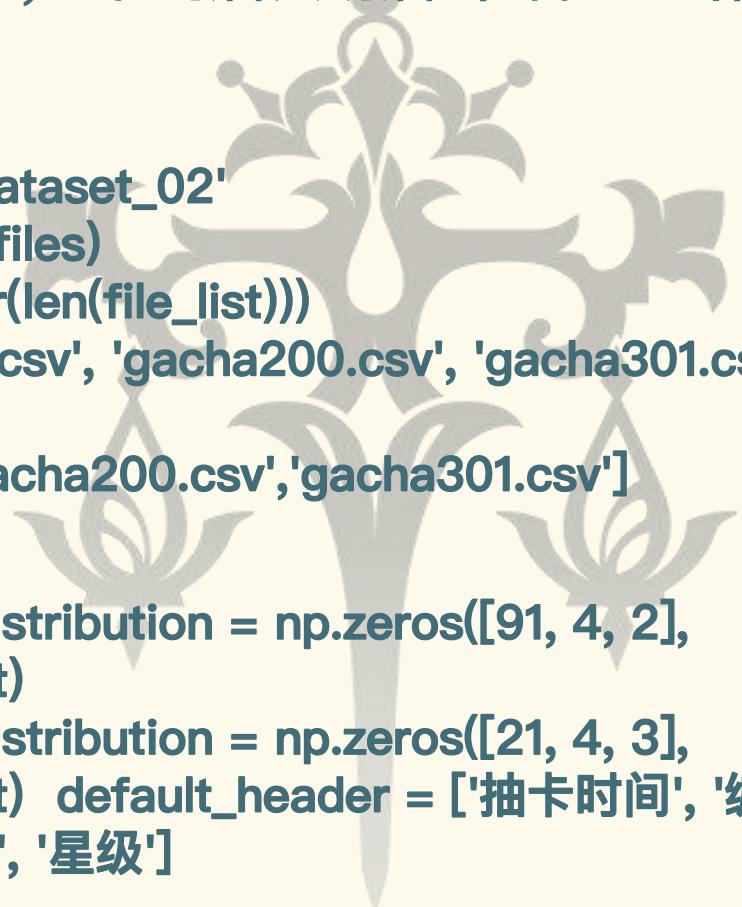
代码实现

选取数据集，根据数据集配置相应
函数读取文件

```
total_files = 'GI_gacha_dataset_02'  
file_list = os.listdir(total_files)  
print('分析样本数量:' + str(len(file_list)))  
file_names = ['gacha100.csv', 'gacha200.csv', 'gacha301.csv',  
'gacha302.csv']  
file_c=['gacha100.csv','gacha200.csv','gacha301.csv']
```



```
star_5_distribution = np.zeros([91, 4, 2],  
dtype=int)  
star_4_distribution = np.zeros([21, 4, 3],  
dtype=int) default_header = ['抽卡时间', '编号', '名  
称', '类别', '星级']
```



记录数据，并计算出相应的统计数值：

```
def produce_var(star, gacha_data, check_p):
    tot = 0
    for k in range(1, len(gacha_data)):
        tot += k * gacha_data[k]
    data_mean = tot / sum(gacha_data)
    s_2 = 0
    for k in range(1, len(gacha_data)):
        s_2 += gacha_data[k] * (k - data_mean) ** 2
    s_2 = s_2 / (sum(gacha_data) - 1)
    stander_check = (data_mean - 1 / check_p)/math.sqrt(s_2/sum(gacha_data))

    print('====' + str(star) + '星分析====')
    print('样本量' + str(sum(gacha_data)))
    print('样本均值' + str(data_mean))
    print('样本平均概率' + str(1 / data_mean))
    print('样本方差' + str(s_2))
    print('转为标准正态的参考值'+str(stander_check))
```



绘制出图表数据：

```
plt.plot(range(1, guarantee_pull+1), Expect_distribution_5[1:guarantee_pull+1],  
label='theory')  
plt.plot(range(1, guarantee_pull+1), x[1:guarantee_pull+1] /  
sum(x[1:guarantee_pull+1]),  
label='actual situation in dataset_02')  
plt.title(file_text+' 5 star distribution')  
plt.legend(loc="upper left")  
plt.text(15, 0.06, '5star sample number:' + str(sum(x[1:guarantee_pull+1])) +  
\n' +  
    'theory probability:' + str(round(100/expect_pull_time, 4)) + '%' + '\n' +  
    'sample probability:' + str(round(100/data_mean, 4)) + '%' + '\n' +  
    'max pull:' + str(max_pull) + '\n' +  
    'plot time:' + time.asctime(time.localtime(time.time())),  
verticalalignment="top", horizontalalignment="left")
```



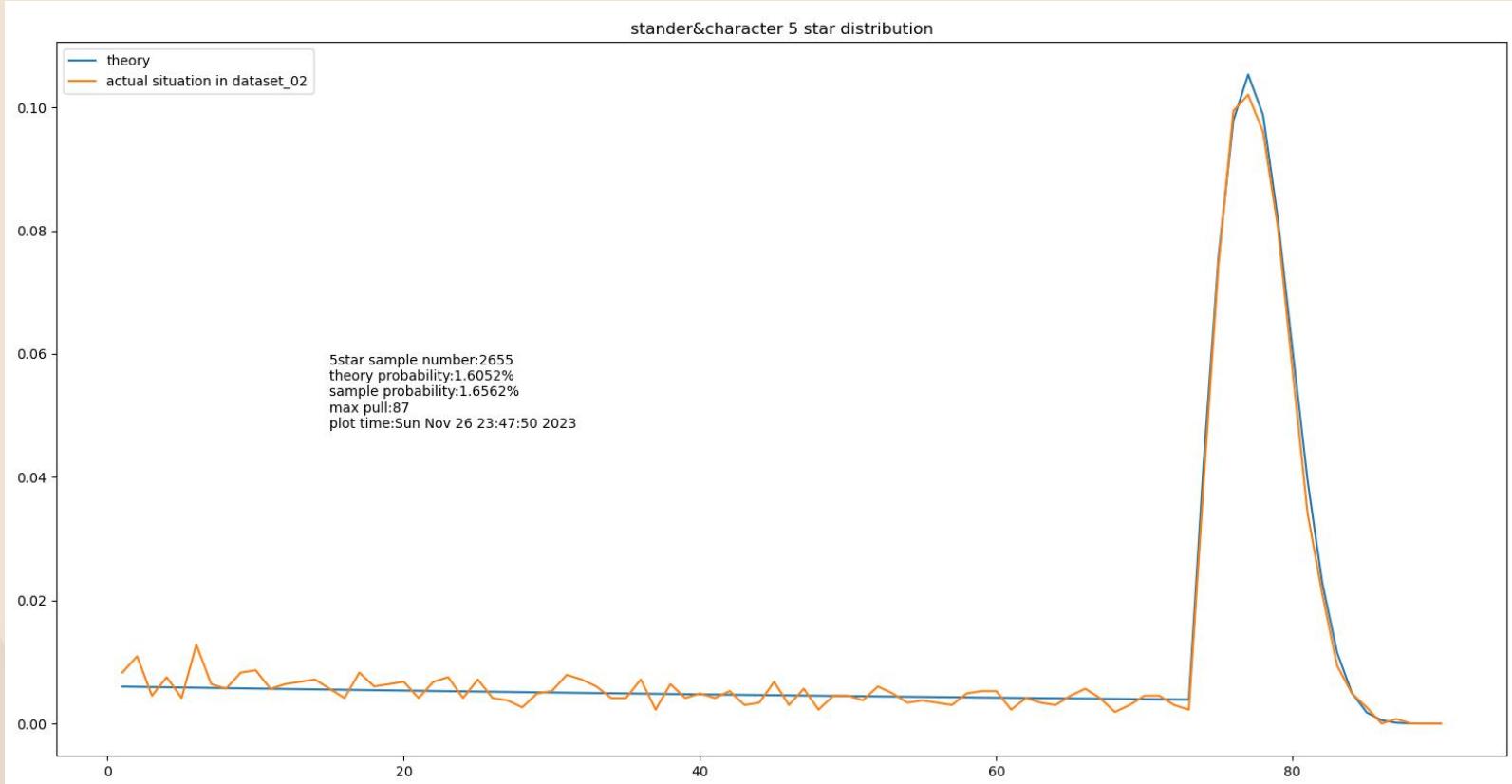
```
need_5 = np.sum(np.sum(star_5_distribution[0:91, 1:3, :], axis=2),  
axis=1) plot_5_star_compare_graph(need_5, 0) need_5 =  
np.sum(np.sum(star_5_distribution[0:91, 3:4, :], axis=2),  
axis=1) plot_5_star_compare_graph(need_5, 1)
```





成果图——角色部分

五星角色频率分布直方图







概率分布-正态分布及标准正态分布

标准正态分布 (standard normal distribution) , 是一个在数学、物理及工程等领域都非常重要的概率分布，在统计学的许多方面有着重大的影响力。期望值 $\mu=0$, 即曲线图象对称轴为Y轴, 标准差 $\sigma=1$ 条件下的正态分布, 记为 $N(0, 1)$ 。

$$\sigma = \sqrt{(\Sigma ((x - \mu)^2) / N)}$$

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$$

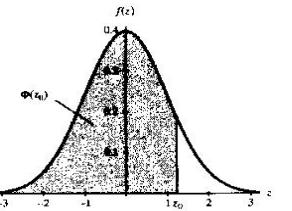
通过标准正态分布表，我们可以对符合正态分布的函数进行分析起每个点所对应的概率。



如，当在触发保底机制的前提下，我们若想研究在第几抽可以得到抽到（想要的角色/武器）时，就可以了解到它的概率是多少。



标准正态分布及参考值



$$P(Z \leq z) = \Phi(z) = \int_{-\infty}^z \frac{1}{\sqrt{2\pi}} e^{-w^2/2} dw$$

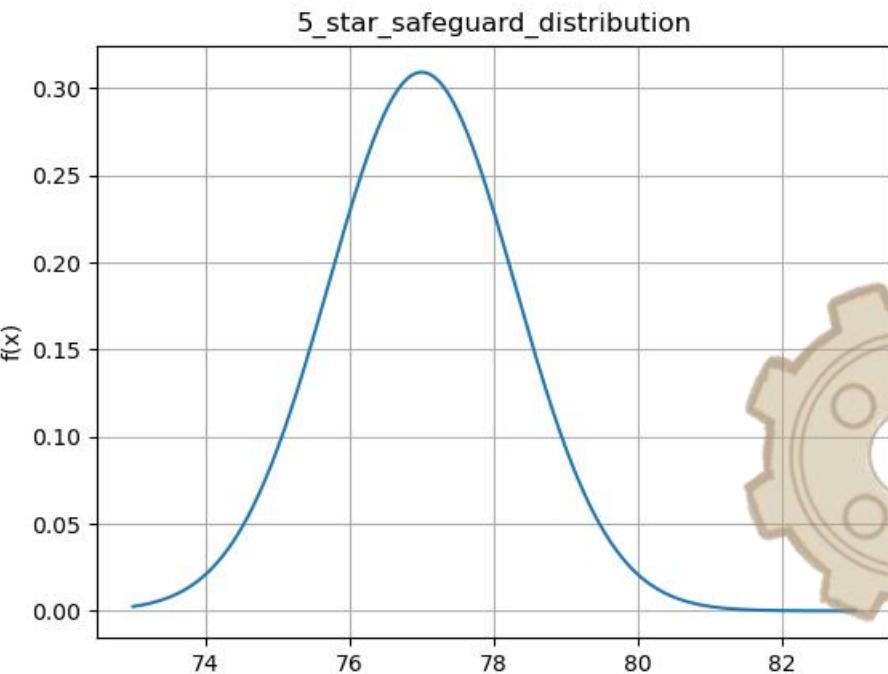
$$\Phi(-z) = 1 - \Phi(z)$$

| z | 0.00 | 0.01 | 0.02 | 0.03 | 0.04 | 0.05 | 0.06 | 0.07 | 0.08 | 0.09 |
|----------------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 0.0 | 0.5000 | 0.5040 | 0.5080 | 0.5120 | 0.5160 | 0.5199 | 0.5239 | 0.5279 | 0.5319 | 0.5359 |
| 0.1 | 0.5398 | 0.5438 | 0.5478 | 0.5517 | 0.5557 | 0.5596 | 0.5636 | 0.5675 | 0.5714 | 0.5753 |
| 0.2 | 0.5793 | 0.5832 | 0.5871 | 0.5910 | 0.5948 | 0.5987 | 0.6026 | 0.6064 | 0.6103 | 0.6141 |
| 0.3 | 0.6179 | 0.6217 | 0.6255 | 0.6293 | 0.6331 | 0.6368 | 0.6406 | 0.6443 | 0.6480 | 0.6517 |
| 0.4 | 0.6554 | 0.6591 | 0.6628 | 0.6664 | 0.6700 | 0.6736 | 0.6772 | 0.6808 | 0.6844 | 0.6879 |
| 0.5 | 0.6915 | 0.6950 | 0.6985 | 0.7019 | 0.7054 | 0.7088 | 0.7123 | 0.7157 | 0.7190 | 0.7224 |
| 0.6 | 0.7257 | 0.7291 | 0.7324 | 0.7357 | 0.7389 | 0.7422 | 0.7454 | 0.7486 | 0.7517 | 0.7549 |
| 0.7 | 0.7580 | 0.7611 | 0.7642 | 0.7673 | 0.7703 | 0.7734 | 0.7764 | 0.7794 | 0.7823 | 0.7852 |
| 0.8 | 0.7881 | 0.7910 | 0.7939 | 0.7967 | 0.7995 | 0.8023 | 0.8051 | 0.8078 | 0.8106 | 0.8133 |
| 0.9 | 0.8159 | 0.8186 | 0.8212 | 0.8238 | 0.8264 | 0.8289 | 0.8315 | 0.8340 | 0.8365 | 0.8389 |
| 1.0 | 0.8413 | 0.8438 | 0.8461 | 0.8485 | 0.8508 | 0.8531 | 0.8554 | 0.8577 | 0.8599 | 0.8621 |
| 1.1 | 0.8643 | 0.8665 | 0.8686 | 0.8708 | 0.8729 | 0.8749 | 0.8770 | 0.8790 | 0.8810 | 0.8830 |
| 1.2 | 0.8849 | 0.8869 | 0.8888 | 0.8907 | 0.8925 | 0.8944 | 0.8962 | 0.8980 | 0.8997 | 0.9015 |
| 1.3 | 0.9032 | 0.9049 | 0.9066 | 0.9082 | 0.9099 | 0.9115 | 0.9131 | 0.9147 | 0.9162 | 0.9177 |
| 1.4 | 0.9192 | 0.9207 | 0.9222 | 0.9236 | 0.9251 | 0.9265 | 0.9279 | 0.9292 | 0.9306 | 0.9319 |
| 1.5 | 0.9332 | 0.9345 | 0.9357 | 0.9370 | 0.9382 | 0.9394 | 0.9406 | 0.9418 | 0.9429 | 0.9441 |
| 1.6 | 0.9452 | 0.9463 | 0.9474 | 0.9484 | 0.9495 | 0.9505 | 0.9515 | 0.9525 | 0.9535 | 0.9545 |
| 1.7 | 0.9554 | 0.9564 | 0.9573 | 0.9582 | 0.9591 | 0.9599 | 0.9608 | 0.9616 | 0.9625 | 0.9633 |
| 1.8 | 0.9641 | 0.9649 | 0.9656 | 0.9664 | 0.9671 | 0.9678 | 0.9686 | 0.9693 | 0.9699 | 0.9706 |
| 1.9 | 0.9713 | 0.9719 | 0.9726 | 0.9732 | 0.9738 | 0.9744 | 0.9750 | 0.9756 | 0.9761 | 0.9767 |
| 2.0 | 0.9772 | 0.9778 | 0.9783 | 0.9788 | 0.9793 | 0.9798 | 0.9803 | 0.9808 | 0.9812 | 0.9817 |
| 2.1 | 0.9821 | 0.9826 | 0.9830 | 0.9834 | 0.9838 | 0.9842 | 0.9846 | 0.9850 | 0.9854 | 0.9857 |
| 2.2 | 0.9861 | 0.9864 | 0.9868 | 0.9871 | 0.9875 | 0.9878 | 0.9881 | 0.9884 | 0.9887 | 0.9890 |
| 2.3 | 0.9893 | 0.9896 | 0.9898 | 0.9901 | 0.9904 | 0.9906 | 0.9909 | 0.9911 | 0.9913 | 0.9916 |
| 2.4 | 0.9918 | 0.9920 | 0.9922 | 0.9925 | 0.9927 | 0.9929 | 0.9931 | 0.9932 | 0.9934 | 0.9936 |
| 2.5 | 0.9938 | 0.9940 | 0.9941 | 0.9943 | 0.9945 | 0.9946 | 0.9948 | 0.9949 | 0.9951 | 0.9952 |
| 2.6 | 0.9953 | 0.9955 | 0.9956 | 0.9957 | 0.9959 | 0.9960 | 0.9961 | 0.9962 | 0.9963 | 0.9964 |
| 2.7 | 0.9965 | 0.9966 | 0.9967 | 0.9968 | 0.9969 | 0.9970 | 0.9971 | 0.9972 | 0.9973 | 0.9974 |
| 2.8 | 0.9974 | 0.9975 | 0.9976 | 0.9977 | 0.9977 | 0.9978 | 0.9979 | 0.9979 | 0.9980 | 0.9981 |
| 2.9 | 0.9981 | 0.9982 | 0.9982 | 0.9983 | 0.9984 | 0.9984 | 0.9985 | 0.9985 | 0.9986 | 0.9986 |
| 3.0 | 0.9987 | 0.9987 | 0.9987 | 0.9988 | 0.9988 | 0.9989 | 0.9989 | 0.9989 | 0.9990 | 0.9990 |
| α | 0.400 | 0.300 | 0.200 | 0.100 | 0.050 | 0.025 | 0.020 | 0.010 | 0.005 | 0.001 |
| z_α | 0.253 | 0.524 | 0.842 | 1.282 | 1.645 | 1.960 | 2.054 | 2.326 | 2.576 | 3.090 |
| $z_{\alpha/2}$ | 0.842 | 1.036 | 1.282 | 1.645 | 1.960 | 2.240 | 2.326 | 2.576 | 2.807 | 3.291 |



模拟正态分布图

```
def normal_distribution(x, mu,  
sigma):  
    return np.exp(-(x - mu)**2 /  
(2 * sigma**2)) / (sigma *  
np.sqrt(2 * np.pi)) mu = 77  
sigma = 1.29  
x = np.linspace(73, 83, 1000)  
y = normal_distribution(x, mu,  
sigma)
```



配合标准正态分布参考值，我们可以快速得出所需要的（某一抽）概率

模拟73-83抽的正态分布（其中，服从平均值为77， σ 为1.29）





感谢观看

原神！启动！

