

高校数学とJulia言語 Day 5

確率とシミュレーション

～最終日・5日間の集大成～

城北中学校・高等学校 中学3年・高校1年

夏期講習会III 2025/8/24～2025/8/28

担当：清水団



5日間の学習予定







- **Day 1** : Google Colabの紹介・基本計算 ✓
- **Day 2** : 関数のグラフの描画 ✓
- **Day 3** : 最適化（最大・最小） ✓
- **Day 4** : データの分析 ✓
- **Day 5** : 確率・シミュレーション ← 今日はここ！最終日

今日のゴール：確率をシミュレーションで体験し、理論と実践を結びつけよう！

確率とシミュレーションの重要性

現代社会のあらゆる場面で活用されています！

実社会での応用例

-  天気予報：降水確率の計算
-  保険：事故や病気のリスク評価
-  スポーツ：勝率の予測、戦略の最適化
-  経済：株価の変動予測、リスク管理
-  ゲーム：確率の計算、戦略の分析
-  医療：薬の効果や副作用の評価

シミュレーションで実験を何度もしなくても確率的現象を理解できます！

基本的な確率：コイン投げ

理論 vs 実践

理論：表が出る確率は0.5

実践：本当にそうなの？

```
# コイン投げの関数
function coin_flip()
    return rand() < 0.5 ? "表" : "裏"
end

# 大量のシミュレーション
function simulate_coin_flips(n)
    heads_count = sum([rand() < 0.5 for _ in 1:n])
    return heads_count / n
end

# 異なる回数でテスト
for n in [10, 100, 1000, 10000, 100000]
    prob = simulate_coin_flips(n)
    println("$n回: 確率 = $(round(prob, digits=4))")
end
```

大数の法則を視覚化

試行回数が増えるほど理論値に近づく

```
# 確率の収束を可視化
n_max = 1000
cumulative_prob = []
heads_count = 0

for i in 1:n_max
    if rand() < 0.5
        heads_count += 1
    end
    push!(cumulative_prob, heads_count / i)
end

# グラフで収束を確認
plot(1:n_max, cumulative_prob,
     title="コイン投げの確率の収束",
     xlabel="試行回数", ylabel="表が出る確率")
hline!([0.5], color=:red, label="理論値")
```

これが「大数の法則」です！

サイコロの確率分布

1つのサイコロ：各目の確率は $1/6$

```
# サイコロシミュレーション
n_rolls = 6000
dice_results = [rand(1:6) for _ in 1:n_rolls]

# ヒストグラムで分布を確認
histogram(dice_results, bins=0.5:1:6.5,
          title="サイコロの目の分布",
          xlabel="サイコロの目", ylabel="出現回数")

# 理論値の線を追加
hline!([n_rolls/6], color=:red, label="理論値")
```

公正なサイコロなら、各目がほぼ同じ回数出るはず！

2つのサイコロの和

どの数字が最も出やすい？

和	出現パターン	確率
2	(1,1)	1/36
3	(1,2), (2,1)	2/36
4	(1,3), (2,2), (3,1)	3/36
...
7	(1,6), (2,5), ..., (6,1)	6/36

7が最も出やすい！これは実際のカジノゲームでも重要な知識です

🎯 2つのサイコロのシミュレーション

```
# 2つのサイコロの和をシミュレーション
n_rolls = 10000
sums = [rand(1:6) + rand(1:6) for _ in 1:n_rolls]

# 理論的確率
theoretical_probs = [1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1] ./ 36

# ヒストグラムと理論値を比較
histogram(sums, bins=1.5:1:12.5, normalize=true,
          title="2つのサイコロの和の分布",
          xlabel="サイコロの和", ylabel="確率")

plot!(2:12, theoretical_probs,
      color=:red, linewidth=3, marker=:circle,
      label="理論値")
```


誕生日のパラドックス

驚きの確率現象！

問題：30人のクラスで、同じ誕生日の人が2人以上いる確率は？

直感：365日もあるし、30人くらいなら確率は低そう... 現実：なんと約70%の確率で同じ誕生日の人がいる！

なぜこんなに高い？

- 比較する組み合わせが多い（30人なら435通り！）
- 「特定の日」ではなく「どこかで重複」を調べている

誕生日パラドックスのシミュレーション

```
# 誕生日の重複をチェック
function has_birthday_collision(n_people)
    birthdays = rand(1:365, n_people)
    return length(unique(birthdays)) < n_people
end

# シミュレーション実行
function birthday_simulation(n_people, n_trials=10000)
    collisions = sum([has_birthday_collision(n_people)
                     for _ in 1:n_trials])
    return collisions / n_trials
end

# 様々な人数で実験
for n in [10, 15, 20, 23, 25, 30, 35, 40, 50]
    prob = birthday_simulation(n)
    println("$n人: $(round(prob*100, digits=1))%")
end
```

23人で50%を超える！

誕生日パラドックスの可視化

```
people_counts = 5:5:60
probabilities = [birthday_simulation(n) for n in people_counts]

plot(people_counts, probabilities,
      marker=:circle, linewidth=2,
      title="誕生日パラドックス",
      xlabel="クラスの数",
      ylabel="同じ誕生日の人がいる確率")

# 50%ラインを追加
hline!([0.5], color=:red, linestyle=:dash, label="50%")

# 23人のポイントを強調
scatter!([23], [birthday_theory(23)],
         markersize=10, color=:orange,
         label="23人")
```

グラフで見ると急激に確率が上がることがわかります！

🎯 モンテカルロ法： π を求めよう

ランダムな点で円周率を計算！

アイデア：

1. 正方形の中にランダムに点を打つ
2. 円の内側に入る点の割合を数える
3. $\pi \approx 4 \times (\text{円内の点数}) / (\text{全体の点数})$

```
function estimate_pi(n_points)
    inside_circle = 0
    for _ in 1:n_points
        x, y = rand() * 2 - 1, rand() * 2 - 1 # -1~1の範囲
        if x^2 + y^2 <= 1 # 単位円の内部
            inside_circle += 1
        end
    end
    return 4 * inside_circle / n_points
end
```

✓ モンテカルロ法の収束

```
# 異なる点数で $\pi$ を推定
point_counts = [100, 1000, 10000, 100000, 1000000]
pi_estimates = [estimate_pi(n) for n in point_counts]

println("モンテカルロ法による円周率の推定:")
for i in 1:length(point_counts)
    n = point_counts[i]
    pi_est = pi_estimates[i]
    error = abs(pi_est -  $\pi$ )
    println("$n点:  $\pi \approx$  $(round(pi_est, digits=4))")
    println("        誤差: $(round(error, digits=4))")
end

println("真の値:  $\pi =$  $(round( $\pi$ , digits=6))")
```

点数が増えるほど正確になります！

モンテカルロ法の可視化

```
# 少数の点で可視化
n_points = 1000
points_inside = []
points_outside = []

for _ in 1:n_points
    x, y = rand() * 2 - 1, rand() * 2 - 1
    if x^2 + y^2 <= 1
        push!(points_inside, (x, y))
    else
        push!(points_outside, (x, y))
    end
end

# 散布図で表示 (円の内側は青、外側は赤)
scatter([p[1] for p in points_inside], [p[2] for p in points_inside],
        color=:blue, markersize=2, label="円の内側")
scatter!([p[1] for p in points_outside], [p[2] for p in points_outside],
        color=:red, markersize=2, label="円の外側")
```

実習：確率を体験しよう

実際に**Google Colab**で以下を試してみましょう

1. コイン投げ：大数の法則を体験
2. サイコロ：等確率の検証
3. 誕生日パラドックス：直感に反する確率
4. モンテカルロ法： π の近似計算
5. 自分だけの確率実験：オリジナル問題に挑戦

理論だけでなく、実際にシミュレーションして確率を体感しましょう！

本日の演習問題

問題1: ジャンケンの確率

ジャンケンで「あいこ」になる確率をシミュレーションで求める（理論値 $1/3$ と比較）

問題2: 3つのサイコロの最大値

3つのサイコロの最大値が6になる確率を求める

問題3: 自由課題（以下から選択）

A. モンティ・ホール問題：ドアを変更する戦略の効果

B. ランダムウォーク：酔歩問題のシミュレーション

C. オリジナル問題：自分で考えた確率問題

創造性を発揮して、面白い確率問題に挑戦してみましょう！

🌟 5日間の総復習

Day 1: Julia言語の基礎

- ✓ Google Colabの使い方
- ✓ 基本計算と数学関数
- ✓ 変数と数式の処理

Day 2: 関数とグラフ

- ✓ 関数の定義方法
- ✓ 美しいグラフの描画
- ✓ 複数関数の比較分析

Day 3: 最適化

- ✓ 最大・最小値の数値探索
- ✓ グラフによる視覚的理解
- ✓ 制約条件付き最適化

🌟 5日間の総復習（続き）

Day 4: データ分析

- ✓ 統計量の計算と解釈
- ✓ ヒストグラムと散布図
- ✓ 相関分析と回帰直線

Day 5: 確率とシミュレーション

- ✓ 確率現象のシミュレーション
- ✓ 理論値と実験値の比較
- ✓ モンテカルロ法による数値計算

5日間で、数学とプログラミングの強力な組み合わせを体験しました！

身につけたスキル

プログラミングスキル

- **Julia**言語の基本的な使い方
- 関数定義とグラフ描画
- データ処理と統計分析
- シミュレーション技術

数学的思考力

- 視覚化による直感的理解
- 数値計算による問題解決
- 統計的な現象の理解
- 確率的な思考方法

問題解決能力

- 仮説→実験→検証のサイクル
- 理論と実践の橋渡し

今後の学習に向けて

大学での学習

- 微積分：関数の解析がより深く理解できる
- 線形代数：行列計算をJuliaで実践
- 統計学：今日学んだ基礎をさらに発展
- 物理・化学：数値実験とシミュレーション

将来のキャリア

- データサイエンティスト：今日学んだスキルが基礎
- **AI・機械学習エンジニア**：確率とプログラミングが必須
- 金融工学：リスク分析にモンテカルロ法
- 研究者：あらゆる分野で数値計算が重要

演習問題を解いてみよう！

Google Colabを開いて、最後の実習に挑戦しましょう

取り組み方

1. 問題を理解する
2. シミュレーションを設計する
3. 理論値を計算する（可能であれば）
4. 結果を比較・検証する
5. グラフで可視化する
6. 考察をまとめる

5日間の学習の集大成です。全力で取り組みましょう！

🎉 お疲れさまでした！ 🎉

5日間の「高校数学とJulia言語」講習会

完走おめでとうございます！

皆さんは今、数学とプログラミングという
2つの強力な武器を手に入れました

この経験が、皆さんの未来を
より豊かで創造的なものにしてくれることを
心から願っています

最終メッセージ

数学 × プログラミング = 無限の可能性

- 数学は「考える力」を育てます
- プログラミングは「実現する力」を育てます
- この2つの組み合わせで、皆さんの可能性は無限に広がります

今後への提案

1. 続けること：学んだスキルを忘れないよう定期的に使う
2. 深めること：興味を持った分野をさらに探求する
3. 応用すること：他の教科や日常生活でも活用する
4. 創造すること：自分だけのプロジェクトに挑戦する

今回学んだ技術を使って、ぜひ自分だけの面白いプロジェクトに挑戦してください！

参考資料・今後の学習

公式ドキュメント

- Julia Documentation: <https://docs.julialang.org/>
- Plots.jl: <https://docs.juliaplots.org/>
- Statistics.jl: <https://docs.julialang.org/en/v1/stdlib/Statistics/>

学習リソース

- オンライン学習サイト
- 確率・統計の参考書
- データサイエンス入門書
- プログラミング関連書籍

コミュニティ

- Julia言語のコミュニティ
- 数学・統計学の勉強会
- プログラミング学習グループ

? 最後の質問タイム

何か質問はありませんか？

- 今日学んだ確率・シミュレーションについて
- 5日間の内容全般について
- 今後の学習方法について
- プログラミングと数学の関係について
- 将来のキャリアについて
- その他、何でも！

遠慮なく質問してください。皆さんの学習を最後まで全力でサポートします！

🎉 本当にお疲れさまでした！ 🎉

5日間、よく頑張りました！

**皆さんの今後の成長と活躍を
心から応援しています！**

Keep coding, keep learning!

宿題

最後の演習問題を完成させて、Google Classroomに提出してください。

また会える日を楽しみにしています！ 🙌