7章
仮説と推定

統計に従った振る舞いが、本当に知的な人間の証である。

── ジョージ・バーナード・ショー ● 劇作家

統計と確率の理論を何に用いるのでしょうか。データサイエンスのサイエンスの部分には、データとそれを生成するプロセスに関する**仮説**を立て、検定を行う作業が含まれます。

7.1 統計的仮説検定

データサイエンスでは、ある仮説が真である可能性が高いか否かを示さなければならない場面がしばしば登場します。ここで言う仮説とは「このコインには歪みがない」、「データサイエンティストはRよりもPythonを好む」、「うっとうしい広告がポップアップされる上に、広告のクローズボタンが小さくて見えにくいようなWebページは、内容が全く読まれることもなくページが閉じられてしまう可能性が高い」など、データに関する統計量に言い換えられる、ある種の主張のことです。統計量とは、さまざまな仮定のもとで既知の分散に従う確率変数の観測結果であると考えられ、それらの仮定がどの程度確からしいかを提示できます。

古典的な設定では、**帰無仮説** H_0 が基本的な立ち位置を表し、対立仮説 H_1 と比較されます。統計量を用いて、この帰無仮説 H_0 を棄却できるか否かを決定します。例を見れば、この考え方が理解できるでしょう。

7.2 事例：コイン投げ

ここにコインがあるとしましょう。このコインに歪みがあるか否かを確認します。コインを投げて表が出る確率を p とした場合、コインに歪みがないことを示す帰無仮説は、$p = 0.5$ となります。この仮説と対立仮説である $p \neq 0.5$ と比較して検定を行います。

この検定ではコインをn回投げて表が出た回数Xを数えます。各コイン投げはベルヌーイ試行に当たり、XはBinomial(n,p)の確率変数となります。これは（第6章で見たように）正規分布で近似可能です。

```
def normal_approximation_to_binomial(n, p):
    """Binomial(n, p)に相当するμとσを計算する"""
    mu = p * n
    sigma = math.sqrt(p * (1 - p) * n)
    return mu, sigma
```

確率変数が正規分布に従う限り、実際の値が特定の範囲内に入る（もしくは入らない）確率はnormal_cdfを使って把握できます。

```
# 変数が閾値を下回る確率はnormal cdfで表せる
normal_probability_below = normal_cdf

# 閾値を下回っていなければ、閾値より上にある
def normal_probability_above(lo, mu=0, sigma=1):
    return 1 - normal_cdf(lo, mu, sigma)

# hiより小さく、loより大きければ、値はその間にある
def normal_probability_between(lo, hi, mu=0, sigma=1):
    return normal_cdf(hi, mu, sigma) - normal_cdf(lo, mu, sigma)

# 間になければ、範囲外にある
def normal_probability_outside(lo, hi, mu=0, sigma=1):
    return 1 - normal_probability_between(lo, hi, mu, sigma)
```

同じことは逆の手順でも可能です。あるレベルまでの可能性に相当する区間または平均を中心とした左右対称な領域を求めます。例えば、平均を中心として60%の確率で発生する領域を求めたければ、上下それぞれの確率が20%の分を取り除けば良いのです（残りは60%となります）。

```
def normal_upper_bound(probability, mu=0, sigma=1):
    """確率P(Z <= z)となるzを返す"""
    return inverse_normal_cdf(probability, mu, sigma)

def normal_lower_bound(probability, mu=0, sigma=1):
    """確率P(Z >= z)となるzを返す"""
    return inverse_normal_cdf(1 - probability, mu, sigma)
```

```
def normal_two_sided_bounds(probability, mu=0, sigma=1):
    """指定された確率を包含する（平均を中心に）対称な境界を返す"""
    that contain the specified probability"""
    tail_probability = (1 - probability) / 2

    # 上側の境界はテイル確率（tail_probability）分上に
    upper_bound = normal_lower_bound(tail_probability, mu, sigma)

    # 下側の境界はテイル確率（tail_probability）分下に
    lower_bound = normal_upper_bound(tail_probability, mu, sigma)

    return lower_bound, upper_bound
```

コイン投げの回数を$n = 1000$としましょう。コインに歪みはないという仮説が真であるなら、Xは平均が500で分散15.8の正規分布で近似できます。

```
mu_0, sigma_0 = normal_approximation_to_binomial(1000, 0.5)
```

実際には真であるにもかかわらずH_0を棄却してしまうという**第一種の過誤**（偽陽性）をどの程度受け入れるか、いわゆる**有意性**について決めておかなければなりません。失われた年代記によると、多くの場合で有意水準を5%か1%に設定しています。ここでは5%を使用しましょう。

Xが以下で与えられる区間外になってしまったため、H_0が棄却されるという状況について考えてみましょう。

```
normal_two_sided_bounds(0.95, mu_0, sigma_0) # (469, 531)
```

pが実際に0.5に等しい（つまり、H_0が真である）場合を考えると、Xがこの区間外となる可能性は5%であり、これは当初考えた有意性と正確に等しい値です。別の表現をしてみましょう。もしH_0が真である場合、おおよそ20回のうち19回はこの検定が正しい結論を導くことになります。

検定力、つまり実際にはH_0が偽であるにもかかわらずH_0を棄却しないという**第二種の過誤**が起きない確率についても考えてみましょう。検定力を測るために、H_0が偽であるとはどういうことなのかを正確に定義しなくてはなりません（pが0.5ではないと知っていることは、Xの分布に関する情報をほとんどもたらしません）。コインの表が出やすいように少しだけ歪んでいて、pが実際には0.55であった場合に何が起きる

のかを確認しましょう。

この場合、検定力は次のように計算できます。

```
# pが0.5であると想定の元で、95%の境界を確認する
lo, hi = normal_two_sided_bounds(0.95, mu_0, sigma_0)

# p = 0.55であった場合の、μとσを計算する
mu_1, sigma_1 = normal_approximation_to_binomial(1000, 0.55)

# 第二種過誤とは、帰無仮説を棄却しないという誤りがあり、Xが
# 当初想定の領域に入っている場合に生じる
type_2_probability = normal_probability_between(lo, hi, mu_1, sigma_1)
power = 1 - type_2_probability # 0.887
```

代わりに帰無仮説が、コインに歪みがない、もしくは$p \leq 0.5$であると仮定してみましょう。この場合片側検定を使います。Xが500よりずっと大きければ帰無仮説を棄却し、500よりも小さければ棄却しません。つまり、5%の有意性で検定を行うには、normal_probability_belowを使って確率が95%となるカットオフ値を求めることになります。

```
hi = normal_upper_bound(0.95, mu_0, sigma_0)
# 526（< 531, 上側のテイル部分が少し大きくなる）

type_2_probability = normal_probability_below(hi, mu_1, sigma_1)
power = 1 - type_2_probability # 0.936
```

Xが469より小さい（H_1が真であるなら、ほとんど起こりえない値）場合にはH_0を棄却せず、一方でXが526と531の間（H_1が真であるなら、多少起こりえる可能性がある値）の場合にはH_0を棄却することになるため、より強い検定であると言えます。

この検定を測る別の尺度がp値です。特定の確率でのカットオフを選ぶ代わりに、H_0が真であると仮定して、実際に観測された値と少なくとも同等に極端な値が生じる確率を計算します。

コインの歪みに関する両側検定は次のように計算します。

```
def two_sided_p_value(x, mu=0, sigma=1):
    if x >= mu:
        # xが平均より大きい場合、テイル確率はxより大きい分
        return 2 * normal_probability_above(x, mu, sigma)
```

```
    else:
        # xが平均より小さい場合、テイル確率はxより小さい分
        return 2 * normal_probability_below(x, mu, sigma)
```

表が530回出た場合は次のように計算できます。

```
two_sided_p_value(529.5, mu_0, sigma_0) # 0.062
```

連続性補正（https://ja.wikipedia.org/wiki/連続性補正）により、ここでは530ではなく、529.5を使いました。530回の表が出る確率は`normal_probability_between(530, 531, mu_0, sigma_0)`よりも`normal_probability_between(529.5, 530.5, mu_0, sigma_0)`の方が良い推定となるという事実を反映しています。それに対応して、少なくとも530回の表が出る確率の推定は、`normal_probability_above(529.5, mu_0,sigma_0)`を使う方が良い値となります。これは**図6-4**を作るコードですでに使われています。

これが理にかなった推定であることを納得するために、シミュレーションを行いましょう。

```
extreme_value_count = 0
for _ in range(100000):
    num_heads = sum(1 if random.random() < 0.5 else 0 # 1,000回のコイン投げを行い、
                    for _ in range(1000))             # 表が出る回数を数える。
    if num_heads >= 530 or num_heads <= 470:          # そのうち極端な回数はどれだけ
        extreme_value_count += 1                      # 出たかを数える。

print extreme_value_count / 100000 # 0.062
```

p値は有意性の5%よりも大きいため、帰無仮説は棄却されません。それでは、表が532回出た場合はどうでしょうか。

```
two_sided_p_value(531.5, mu_0, sigma_0) # 0.0463
```

5%よりも小さい値となったので、帰無仮説を棄却します。これは先に行った検定と全く同じものですが、統計的に異なるアプローチを取っています。

次も同様です。

```
upper_p_value = normal_probability_above
lower_p_value = normal_probability_below
```

片側検定の場合、525回の表が出れば、帰無仮説を棄却しませんが、

```
upper_p_value(524.5, mu_0, sigma_0) # 0.061
```

527回ならば、棄却することになります。

```
upper_p_value(526.5, mu_0, sigma_0) # 0.047
```

normal_probability_aboveを使ってp値を計算する前に、データがおおよそ正規分布に従っていることを確かめる必要があります。データサイエンスの黒歴史には、観測されたデータが無作為に発生する可能性は100万分の1だ、と主張するような事例が山ほどあります。正しくは、「データが正規分布に従うなら、その可能性がある」のでありデータがその前提とは異なる場合には、意味がありません。

正規分布であるかどうかを調べる統計手法は数多く存在しますが、データをグラフ化するのが、簡単で優れています。

7.3 信頼区間

分布を未知のパラメータとして、コインの表が出る確率に関する仮説検定を行ってきました。もしこれが本当であるなら、観測値の周辺の**信頼区間**を求めるのが3番目の手法となります。

例えば、表を1、裏を0とするベルヌーイ変数の平均を見ることで歪みのあるコインに対する確率を推定できます。1,000回の試行で525回の表が出たとすると、pの推定値は0.525です。

この推定値はどの程度信頼できるでしょうか。pの正確な値を知っているなら、中心極限定理により（「**6.7 中心極限定理**」を思い出してください）、このベルヌーイ変数の平均値は、近似的に平均pおよび次の標準偏差の正規分布に従います。

```
math.sqrt(p * (1 - p) / 1000)
```

ここではpが未知となっているので、先の推定値を使います。

```
p_hat = 525 / 1000
mu = p_hat
sigma = math.sqrt(p_hat * (1 - p_hat) / 1000) # 0.0158
```