Peña530Week8

February 4, 2024

Chapters 9 & 10

Miles A. Peña

DSC 530

02/04/24

Chapter 9

As sample size increases, the power of a hypothesis test increases, which means it is more likely to be positive if the effect is real. Conversely, as sample size decreases, the test is less likely to be positive even if the effect is real. To investigate this behavior, run the tests in this chapter with different subsets of the NSFG data. You can use thinkstats2.SampleRows to select a random subset of the rows in a DataFrame. What happens to the p-values of these tests as sample size decreases? What is the smallest sample size that yields a positive test?

[25]:
```
import numpy as np

import first
import hypothesis
import thinkstats2
```

[26]:
```
class DiffMeans(hypothesis.DiffMeansPermute):

    def RunModel(self):

        one = np.random.choice(self.pool, self.n, replace = True)
        two = np.random.choice(self.pool, self.m, replace = True)

        return one, two

def RunSampleTest(first, others):

    # baby length
    data = firsts.prglngth.values, others.prglngth.values
    ht = DiffMeans(data)
```

```
    pvalue = ht.PValue(iters = 10000)

    print("\nMeans Permute Pregnancy Length")
    print("P-Value: {:.3f}".format(pvalue))
    print("Actual: {:.3f}".format(ht.actual))
    print("T-test Max: {:.3f}".format(ht.MaxTestStat()))

    # baby weight
    data = (firsts.totalwgt_lb.dropna().values,
            others.totalwgt_lb.dropna().values)
    ht = hypothesis.DiffMeansPermute(data)
    pvalue = ht.PValue(iters = 10000)
    print("\nMeans Permute Birthweight")
    print("P-Value: {:.3f}".format(pvalue))
    print("Actual: {:.3f}".format(ht.actual))
    print("T-test Max: {:.3f}".format(ht.MaxTestStat()))

def RunTests(live, iters=1000):

    n = len(live)
    firsts = live[live.birthord == 1]
    others = live[live.birthord != 1]

    # diff in mean prglngth
    data = firsts.prglngth.values, others.prglngth.values
    ht = hypothesis.DiffMeansPermute(data)
    pvalue1 = ht.PValue()

    # diff in mean brthwgt
    data = (firsts.totalwgt_lb.dropna().values,
            others.totalwgt_lb.dropna().values)
    ht = hypothesis.DiffMeansPermute(data)
    pvalue2 = ht.PValue()

    # correlation test
    live2 = live.dropna(subset=['agepreg', 'totalwgt_lb'])
    data = live2.agepreg.values, live2.totalwgt_lb.values
    ht = hypothesis.CorrelationPermute(data)
    pvalue3 = ht.PValue()

    # chi-square test of pregnancy length
    data = firsts.prglngth.values, others.prglngth.values
    ht = hypothesis.PregLengthTest(data)
    pvalue4 = ht.PValue()

    print("{}\t{:.3f}\t{:.3f}\t{:.3f}\t{:.3f}".format(n, pvalue1, pvalue2,␣
 ↪pvalue3, pvalue4))
```

[27]:
```
thinkstats2.RandomSeed(10)

live, firsts, others = first.MakeFrames()
RunSampleTest(first, others)
```

```
Means Permute Pregnancy Length
P-Value: 0.161
Actual: 0.078
T-test Max: 0.212

Means Permute Birthweight
P-Value: 0.000
Actual: 0.125
T-test Max: 0.123
```

[28]:
```
n = len(live)
print("n\t Test1\t  Test2\t  Test3\t  Test4\t")
for i in range(1):
    sample = thinkstats2.SampleRows(live, n)
    RunTests(sample)
    n //= 2
```

```
n        Test1    Test2   Test3   Test4
9148    0.183   0.000   0.000   0.000
```

[29]:
```
n = len(live)
print("n\t Test1\t  Test2\t  Test3\t  Test4\t")
for i in range(2):
    sample = thinkstats2.SampleRows(live, n)
    RunTests(sample)
    n //= 2
```

```
n        Test1    Test2   Test3   Test4
9148    0.153   0.000   0.000   0.000
4574    0.225   0.000   0.000   0.000
```

[30]:
```
n = len(live)
print("n\t Test1\t  Test2\t  Test3\t  Test4\t")
for i in range(3):
    sample = thinkstats2.SampleRows(live, n)
    RunTests(sample)
    n //= 2
```

```
n        Test1    Test2   Test3   Test4
9148    0.151   0.000   0.000   0.000
4574    0.167   0.016   0.000   0.000
2287    0.981   0.010   0.001   0.001
```

[31]:
```
n = len(live)
print("n\t Test1\t  Test2\t  Test3\t  Test4\t")
for i in range(4):
    sample = thinkstats2.SampleRows(live, n)
    RunTests(sample)
    n //= 2
```

```
n        Test1    Test2   Test3   Test4
9148    0.172   0.000   0.000   0.000
4574    0.487   0.003   0.000   0.000
2287    0.057   0.001   0.005   0.000
1143    0.587   0.392   0.176   0.860
```

[32]:
```
n = len(live)
print("n\t Test1\t  Test2\t  Test3\t  Test4\t")
for i in range(5):
    sample = thinkstats2.SampleRows(live, n)
    RunTests(sample)
    n //= 2
```

```
n        Test1    Test2   Test3   Test4
9148    0.177   0.000   0.000   0.000
4574    0.263   0.000   0.000   0.000
2287    0.724   0.008   0.000   0.000
1143    0.886   0.030   0.103   0.006
571     0.609   0.135   0.043   0.336
```

[33]:
```
n = len(live)
print("n\t Test1\t  Test2\t  Test3\t  Test4\t")
for i in range(6):
    sample = thinkstats2.SampleRows(live, n)
    RunTests(sample)
    n //= 2
```

```
n        Test1    Test2   Test3   Test4
9148    0.181   0.000   0.000   0.000
4574    0.010   0.294   0.004   0.000
2287    0.819   0.002   0.000   0.000
1143    0.419   0.023   0.032   0.001
571     0.221   0.053   0.246   0.000
285     0.048   0.207   0.950   0.028
```

[36]:
```
n = len(live)
print("n\t Test1\t  Test2\t  Test3\t  Test4\t")
for i in range(7):
    sample = thinkstats2.SampleRows(live, n)
    RunTests(sample)
    n //= 2
```

```
n        Test1    Test2   Test3   Test4
9148     0.191    0.000   0.000   0.000
4574     0.323    0.005   0.000   0.000
2287     0.168    0.026   0.001   0.000
1143     0.279    0.321   0.004   0.011
571      0.168    0.801   0.554   0.021
285      0.533    0.392   0.001   0.753
142      0.430    0.495   0.205   0.094
```

In general, p-values become negative as the sample size decreases.

Chapter 10

Using the data from the BRFSS, compute the linear least squares fit for log(weight) versus height. How would you best present the estimated parameters for a model like this where one of the variables is log-transformed? If you were trying to guess someone's weight, how much would it help to know their height? Like the NSFG, the BRFSS oversamples some groups and provides a sampling weight for each respondent. In the BRFSS data, the variable name for these weights is totalwt. Use resampling, with and without weights, to estimate the mean height of respondents in the BRFSS, the standard error of the mean, and a 90% confidence interval. How much does correct weighting affect the estimates?

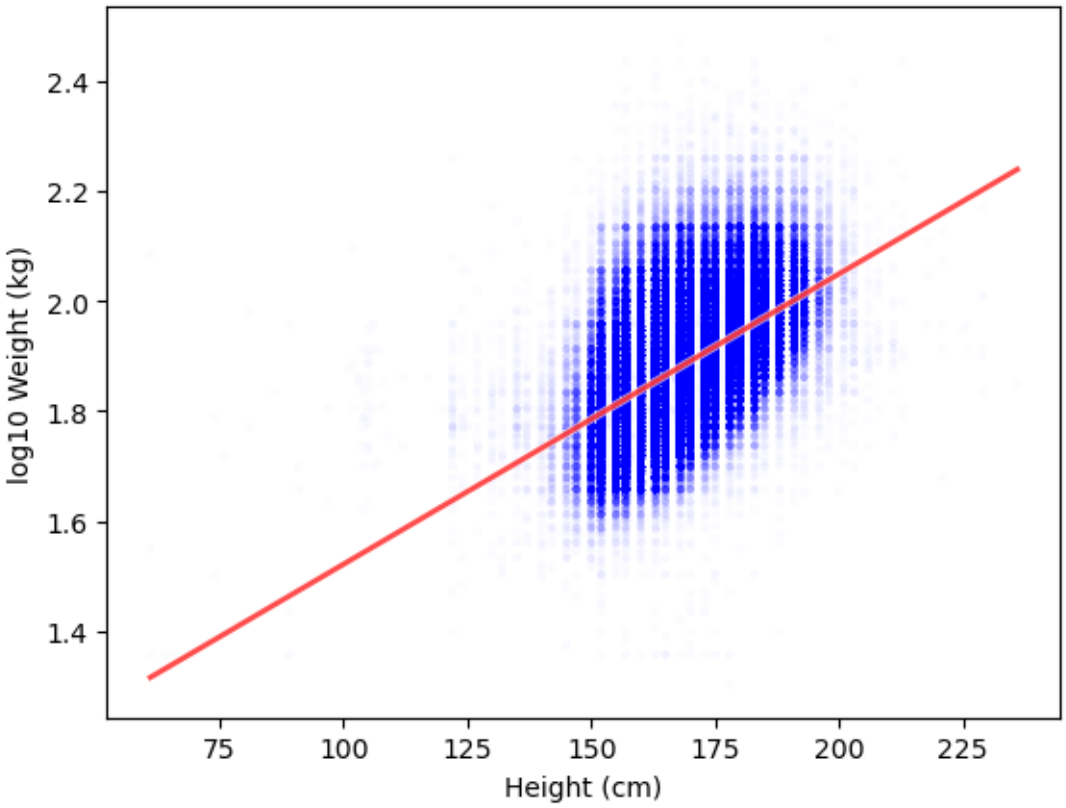
[37]:
```
import pandas as pd

import scatter
import brfss
import thinkplot
```

[41]:
```
df = brfss.ReadBrfss(nrows = None)
df = df.dropna(subset=['htm3', 'wtkg2'])
heights, weights = df.htm3, df.wtkg2
log_weights = np.log10(weights)
```

[42]:
```
inter, slope = thinkstats2.LeastSquares(heights, log_weights)
inter, slope
```

[42]:
```
(0.9930804163932496, 0.005281454169418002)
```

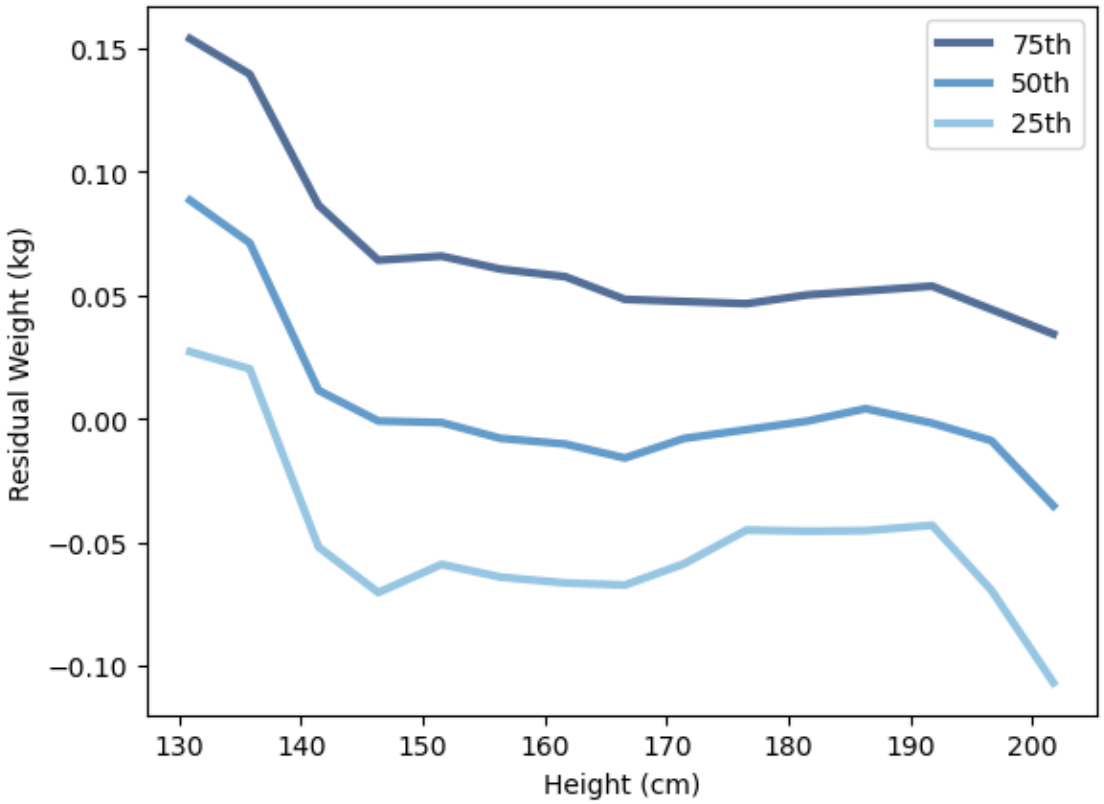
[45]:
```
thinkplot.Scatter(heights, log_weights, color = 'blue', alpha = 0.01, s = 10)
thinkplot.Plot(fxs, fys, color = 'white', linewidth = 3)
thinkplot.Plot(fxs, fys, color = 'red', linewidth = 2)
thinkplot.Config(xlabel = 'Height (cm)', ylabel = 'log10 Weight (kg)', legend =␣
↪False)
```



[50]:
```
res = thinkstats2.Residuals(heights, log_weights, inter, slope)
df['residual'] = res
```

[51]:
```
bins = np.arange(130, 210, 5)
indices = np.digitize(df.htm3, bins)
groups = df.groupby(indices)

means = [group.htm3.mean() for i, group in groups][1:-1]
cdfs = [thinkstats2.Cdf(group.residual) for i, group in groups][1:-1]
```

[57]:
```
thinkplot.PrePlot(3)
for percent in [75, 50, 25]:
    ys = [cdf.Percentile(percent) for cdf in cdfs]
    label = '%dth' % percent
    thinkplot.Plot(means, ys, label = label)

thinkplot.Config(xlabel = 'Height (cm)', ylabel = 'Residual Weight (kg)',␣
↪legend = True)
```



[61]:
```
# compute correlation and coefficient of determination

rho = thinkstats2.Corr(heights, log_weights)
r2 = thinkstats2.CoefDetermination(log_weights, res)

print("Correlation: {:.3f}".format(rho))
print("Coefficent of Determination: {:.3f}".format(r2))
```

```
Correlation: 0.532
Coefficent of Determination: 0.283
```

[62]:
```
rho**2 - r2
```

[62]:
```
2.1149748619109232e-14
```

[63]:
```
# stdev w/o height

std_ys = thinkstats2.Std(log_weights)
print("Standard Deviation w/o Height: {:.3f}".format(std_ys))
```

```
Standard Deviation w/o Height: 0.103
```

[64]:
```
# stdev w/height

std_res = thinkstats2.Std(res)
print("Standard deviation w/ height: {:.3f}".format(std_res))
```

```
Standard deviation w/ height: 0.087
```

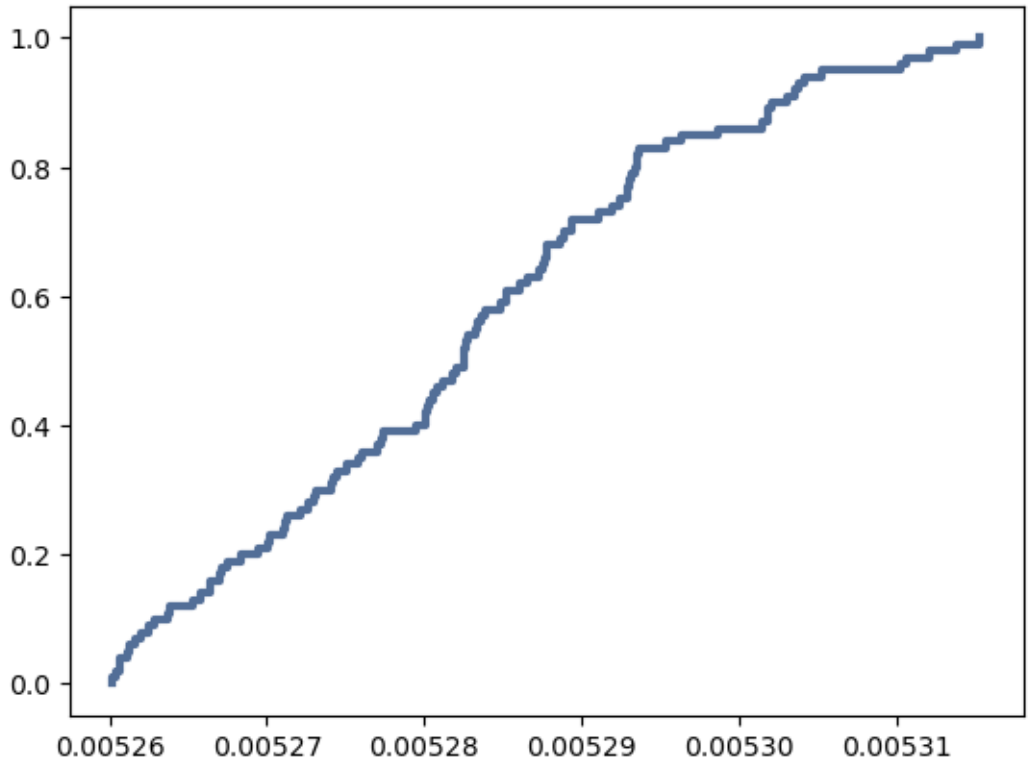
[67]:
```
# height impact

print("Height Impact: {:.3f}".format(1 - (std_res/std_ys)))
```

```
Height Impact: 0.153
```

[69]:
```
t = []
for _ in range(100):
    sample = thinkstats2.ResampleRows(df)
    estimates = thinkstats2.LeastSquares(sample.htm3, np.log10(sample.wtkg2))
    t.append(estimates)

inters, slopes = zip(*t)
```

[70]:
```
cdf = thinkstats2.Cdf(slopes)
thinkplot.Cdf(cdf)
```

[70]:
```
{'xscale': 'linear', 'yscale': 'linear'}
```



[86]:
```
pvalue = cdf[0]
print("P-Value: {:.3f}".format(pvalue))
ci = cdf.Percentile(5), cdf.Percentile(95)
print("Confidence Interval:", ci)
mean = thinkstats2.Mean(slopes)
print("Mean: {:.3f}".format(mean))
stderr = thinkstats2.Std(slopes)
print("Standard Error:", stderr)
```

```
P-Value: 0.000
Confidence Interval: (0.005261086258057469, 0.005305302252469715)
Mean: 0.005
Standard Error: 1.392178520863004e-05
```

[89]:
```
def Summarize(estimates, actual = None):
    mean = thinkstats2.Mean(estimates)
    stderr = thinkstats2.Std(estimates, mu = actual)
    cdf = thinkstats2.Cdf(estimates)
    ci = cdf.ConfidenceInterval(90)
    print('Mean: {:.3f} SE: {:.3f} CI: {}'.format(mean, stderr, ci))
```

[90]:
```
estimates_unweighted = [thinkstats2.ResampleRows(df).htm3.mean() for _ in␣
 ↪range(100)]
Summarize(estimates_unweighted)
```

```
Mean: 168.955 SE: 0.018 CI: (168.92637280462418, 168.98264415206452)
```

[94]:
```
def ResampleRowsWeighted(df, column = 'finalwgt'):
    weights = df[column]
    cdf = thinkstats2.Cdf(dict(weights))
    indices = cdf.Sample(len(weights))
    sample = df.loc[indices]
    return sample
```

[96]:
```
estimates_weighted = [ResampleRowsWeighted(df, 'finalwt').htm3.mean() for _ in␣
 ↪range(100)]
Summarize(estimates_weighted)
```

```
Mean: 170.496 SE: 0.019 CI: (170.465119040401, 170.52218617999554)
```

Using weight yields about a 1.5 cm increase in mean heights. The standard error of the mean is nearly identical. And the confidence interval has a difference of (1.54, 1.54).