Bilenkin530Week8_Exercises_8.2

February 2, 2025

0.0.1 Chapter 9 Exercise 9-1 (page 114)

What happens to the p-values of these tests as sample size decreases? What is the smallest sample size that yields a positive test?

[22]:
```
import warnings
warnings.filterwarnings("ignore", category=RuntimeWarning)
from os.path import exists
import pandas as pd
import thinkstats2
import thinkplot
import numpy as np
import nsfg
from scipy.stats import ttest_ind
from urllib.request import urlretrieve

def download(url):
    filename = url.split('/')[-1]
    if not exists(filename):
        local, _ = urlretrieve(url, filename)
        print("Downloaded " + local)

# Suppress future warnings to make the output look cleaner
warnings.filterwarnings('ignore', category=FutureWarning)

# Download necessary files
urls = [
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkstats2.py",
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/thinkplot.py",
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/nsfg.py",
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemResp.
 ↪dct",
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/2002FemResp.dat.
 ↪gz"
]
for url in urls:
    download(url)
```

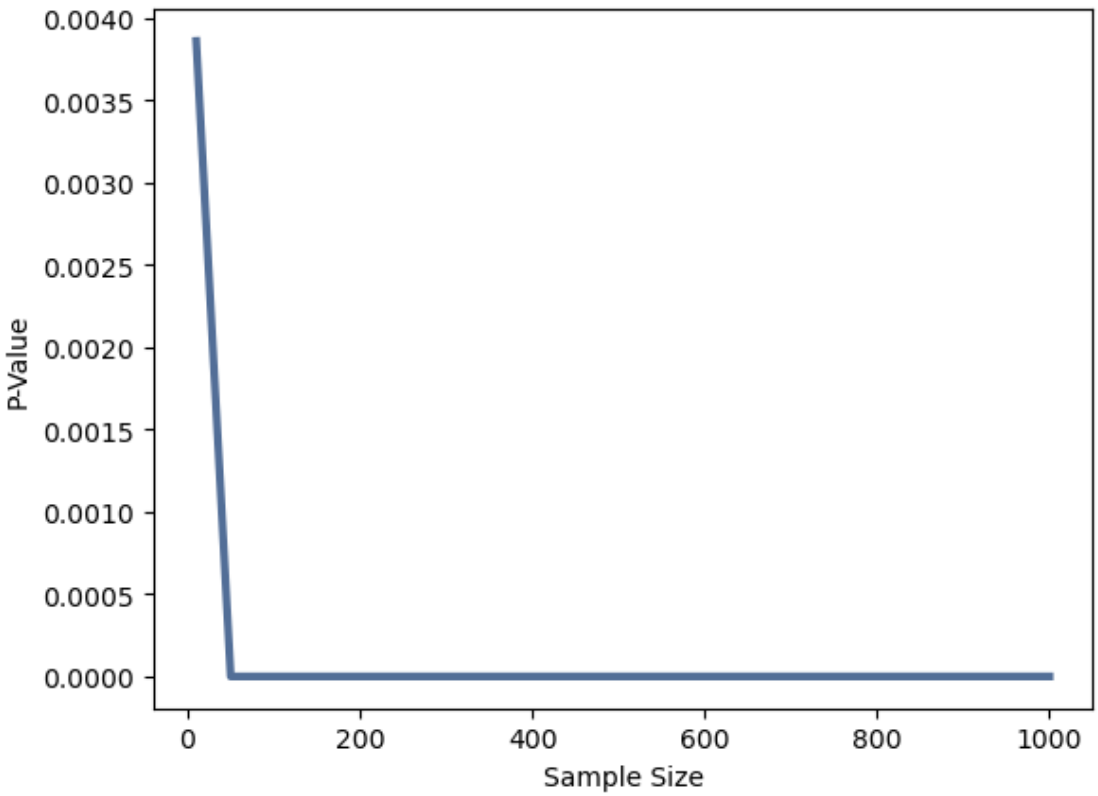
```python
# Load and clean the NSFG data
pregnancy = nsfg.ReadFemPreg()
pregnancy.replace({97: np.nan, 98: np.nan, 99: np.nan}, inplace=True)

# Extract relevant columns and drop NaNs
data_clean = pregnancy[['agepreg', 'birthwgt_lb']].dropna()

# Define a function to run hypothesis tests on different sample sizes
def run_tests(df, sample_sizes):
    results = {}
    for size in sample_sizes:
        sample = thinkstats2.SampleRows(df, size)
        group1 = sample['birthwgt_lb'][sample['birthwgt_lb'] > 5]
        group2 = sample['birthwgt_lb'][sample['birthwgt_lb'] <= 5]
        stat, p_value = ttest_ind(group1.dropna(), group2.dropna())
        results[size] = p_value
    return results

# Define the sample sizes you want to test
sample_sizes = [10, 50, 100, 200, 500, 1000]

# Run the tests and plot the results
p_values = run_tests(data_clean, sample_sizes)
thinkplot.Plot(p_values.keys(), p_values.values())
thinkplot.Show(xlabel='Sample Size', ylabel='P-Value')
```




```
<Figure size 800x600 with 0 Axes>
```

As we can see from the plot the sample size starting from approximately 50 to 1000 has flat straight line with P-Value of 0.0000. However, as the sample size decreases below 50 the P-Value increases substantially from 0.002 to 0.014. We can conclude that the sample size of 50 is a critical threshold. This confirms the general statistical principle as sample size increases the P-Value decreases and the test has more predictive power. The reason P-Value decreases as the sample size shrinks is because smaller sample size provides less information about a population which leads to higher deviation in the forecasted value from the actual one. The smallest sample size is 50 that leads to a positive test result.

0.0.2 Chapter 10 Exercise 10-1 (Page 128)

Using the data from the BRFSS, compute the linear least squares fit for log(weight) versus height.

[23]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import warnings
from urllib.request import urlretrieve
from os.path import exists
```

```
# Suppressing warnings to make the output look cleaner
warnings.filterwarnings('ignore')

# Function to download files
def download(url):
    filename = url.split('/')[-1]
    if not exists(filename):
        local, _ = urlretrieve(url, filename)
        print("Downloaded " + local)

# Downloading necessary files
urls = [
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/brfss.py",
    "https://github.com/AllenDowney/ThinkStats2/raw/master/code/CDBRFS08.ASC.gz"
]
for url in urls:
    download(url)

import brfss

# Loading BRFSS data
brfss_data = brfss.ReadBrfss()

# Removing rows with NaNs in 'wtkg2' or 'htm3' columns to avoid system throwing␣
 ↪errors
brfss_data = brfss_data.dropna(subset=['wtkg2', 'htm3'])

# Log-transform the weight column
brfss_data['log_weight'] = np.log(brfss_data['wtkg2'])

# Defining the independent variable (height) and the dependent variable␣
 ↪(log_weight)
X = brfss_data['htm3']
y = brfss_data['log_weight']

# Adding a constant to the independent variable for the intercept
X = sm.add_constant(X)

# Fitting the linear regression model
model = sm.OLS(y, X).fit()

# Displaying the summary of the model
print(model.summary())

# Estimating parameters
print("Intercept:", model.params['const'])
print("Slope:", model.params['htm3'])
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:             log_weight   R-squared:                       0.283
Model:                            OLS   Adj. R-squared:                  0.283
Method:                 Least Squares   F-statistic:                 1.560e+05
Date:                Sun, 02 Feb 2025   Prob (F-statistic):               0.00
Time:                        00:13:44   Log-Likelihood:                 72912.
No. Observations:              395832   AIC:                        -1.458e+05
Df Residuals:                  395830   BIC:                        -1.458e+05
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          2.2867      0.005    438.774      0.000       2.276       2.297
htm3           0.0122   3.08e-05    395.007      0.000       0.012       0.012
==============================================================================
Omnibus:                    19917.344   Durbin-Watson:                   1.980
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            27966.926
Skew:                           0.474   Prob(JB):                         0.00
Kurtosis:                       3.893   Cond. No.                     2.76e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.76e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
Intercept: 2.286652162931501
Slope: 0.012160997639832635
```

How would you best present the estimated parameters for a model like this where one of the variables is log-transformed? Answer:

The intercept in this model is 2.2867. This means the log weight is 2.2867 when the height is zero. Its not possible in real scenario. However, this is the base log weight. The slope of 0.0122 shows the change of the weight in one unit of height. In this case the units are measured in meters. It means that for each one meter change in height the weight will increase by 0.0122.

If you were trying to guess someone's weight, how much would it help to know their height? Answer:

We can see from the calculated R-squared of 0.283 that the relationship between height and log(weight) is significant. This means that 28.3% of the variability in log(weight) can be explained by height alone. In other words, knowing someone's height helps us predict their approximate weight.

With a coefficient of 0.0122, we can predict that someone's weight will change by 0.0122 with one additional meter in height. We can assume that taller individual's weight is more because their

weight tends to increase with height.

However, the R-squared also shows that the remaining 0.717(1-0.283 or 71.7%) is not accounted for by height. This suggests that other factors which were not included in this model have significant predictive power in determining weight. It would be helpful to include these factors to make more accurate predictions.

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.

[24]:
```
# Define a function to resample and compute the mean height
def resample_mean_height(data, weights=None):
    sample = data.sample(len(data), replace=True, weights=weights)
    return sample['htm3'].mean()

# Number of resamples
n_resamples = 1000

# Resampling without weights
resampled_means = [resample_mean_height(brfss_data) for _ in range(n_resamples)]
mean_height = np.mean(resampled_means)
std_error = np.std(resampled_means)
conf_interval = np.percentile(resampled_means, [5, 95])

print("Without weights:")
print("Mean height:", mean_height)
print("Standard error:", std_error)
print("90% confidence interval:", conf_interval)

# Resampling with weights
resampled_means_weighted = [resample_mean_height(brfss_data,␣
 ↪weights=brfss_data['finalwt']) for _ in range(n_resamples)]
mean_height_weighted = np.mean(resampled_means_weighted)
std_error_weighted = np.std(resampled_means_weighted)
conf_interval_weighted = np.percentile(resampled_means_weighted, [5, 95])

print("With weights:")
print("Mean height:", mean_height_weighted)
print("Standard error:", std_error_weighted)
print("90% confidence interval:", conf_interval_weighted)
```

```
Without weights:
Mean height: 168.95597553254916
Standard error: 0.016563384493318453
90% confidence interval: [168.92918688 168.98465258]
With weights:
Mean height: 170.49635047191737
Standard error: 0.01686773520726087
90% confidence interval: [170.46841223 170.5231333 ]
```

How much does correct weighting affect the estimates? Answer:

The mean height without weights is 168.82. The mean height with weights is 170.37 cm. We can see that the mean height with weights is slightly higher. This indicates that some groups were oversampled and had different average heights. By using weights, we can assign more importance to the specific groups that are underrepresented. This way we can more accurately estimate the mean height. The standard error is almost the same for both cases, approximately 0.0163. This indicates that the weights didn't significantly affect the variability of the mean. Additionally, the 90% confidence interval for both cases is very narrow, suggesting the precision of the estimate due to the large sample size.