Multiple Testing Lab

Last modified: 2026-11-19

```
# The Python chunks on this page import matplotlib, pandas and scipy;
# reticulate provisions them.
py_packages <- c("matplotlib", "pandas", "scipy")
library(reticulate) # nolint: undesirable_function_linter
library(tidymodels) # nolint: undesirable_function_linter
py_require(py_packages)
```

1 The book's Chapter 13 lab

This page holds the code lab for the multiple testing[1] notes.

To work the lab without the solutions, download a stub with only the exercises:

- Quarto
- Jupyter notebook[2]
- Python[3]
- R[4]

Each solution shows the code in Python and in R.

Section 13.6 of the book [1] is the source of these exercises. The Python code uses pandas, matplotlib, plus SciPy, while the R code uses the tidyverse.

2 Exploring the Fund data

Monthly excess returns of 2000 fund managers over 50 months make up the `Fund` data [1, Sec. 13.6]. The data are in `data/fund.csv`[5], one column per manager and one row per month, copied from the `ISLP` Python package. A manager's excess return is the manager's return minus a benchmark's, so a manager with no skill has an excess return of zero on average.

Dataset 1: The fund manager (`Fund`) data

Exercise 2.1 (Lab: the first manager's returns)**.** Use the `Fund` data. What are the sample mean[a] and sample standard deviation[b] of the excess returns of `Manager1`?

Hint

Read the file into a table, then apply a summary function to one column.

[a] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-sample-mean
[b] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-sample-sd

[1] https://morrison-lab.github.io/sds/multiple-testing.html
[2] https://morrison-lab.github.io/lds/chapters/lab-stubs/multiple-testing-lab-stub.ipynb
[3] https://morrison-lab.github.io/lds/chapters/lab-stubs/multiple-testing-lab-stub.py
[4] https://morrison-lab.github.io/lds/chapters/lab-stubs/multiple-testing-lab-stub.R

Solution

Solution 2.1.

2.0.1 Python

```
import pandas as pd

fund_py = pd.read_csv("data/fund.csv")
fund_py_mean = float(fund_py["Manager1"].mean())
fund_py_sd = float(fund_py["Manager1"].std())
round(fund_py_mean, 2), round(fund_py_sd, 2)
```

```
(3.0, 7.42)
```

2.0.2 R

```
fund_r <- readr::read_csv("data/fund.csv", show_col_types = FALSE)
fund_r_mean <- mean(fund_r$Manager1)
fund_r_sd <- sd(fund_r$Manager1)
c(mean = fund_r_mean, sd = fund_r_sd)
```

```
    mean       sd
3.000000 7.416198
```

Over 50 months, `Manager1` has a mean[a] excess return of 3 and a standard deviation[b] of 7.42.

[a] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-sample-mean
[b] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-sample-sd

Exercise 2.2 (Lab: a histogram of the managers' mean returns)**.** Use the Fund data. Draw a histogram of the sample mean[a] excess return of each of the managers.

> **Hint**
> Summarize each column of the table to get one value per manager, then draw a histogram of those values.

[a] https://morrison-lab.github.io/sds/exploratory-descriptive.html#def-sample-mean

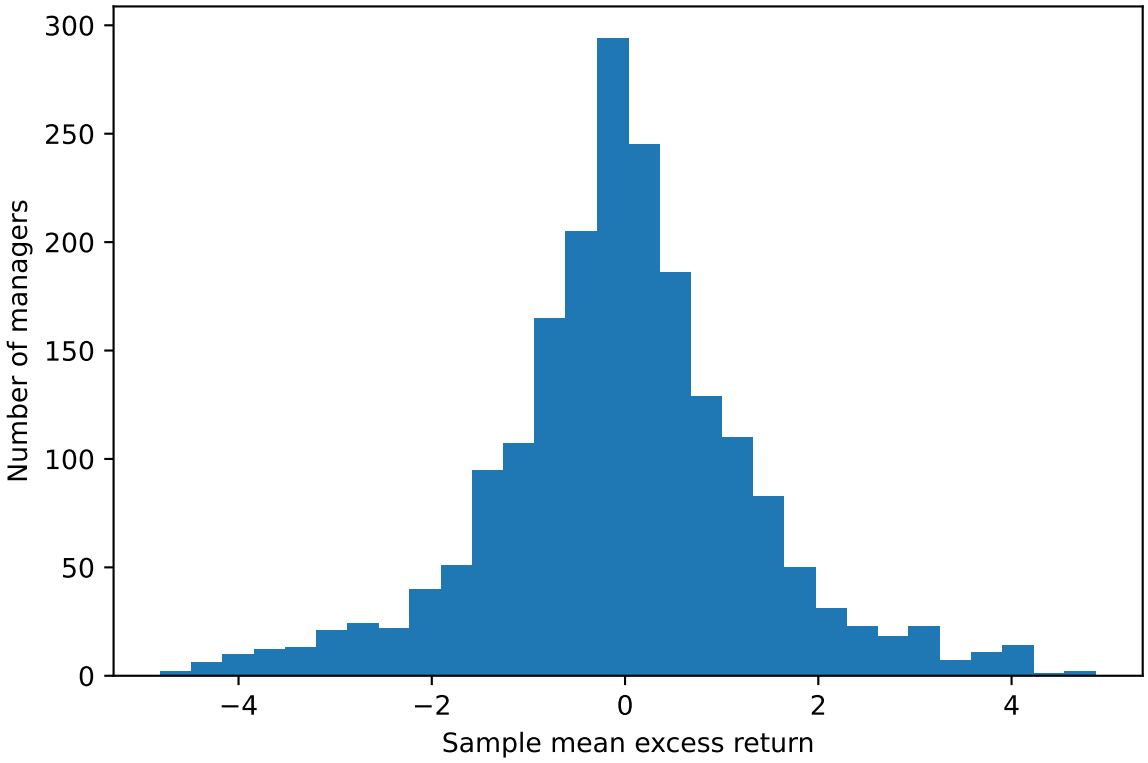
Solution

Solution 2.2. This solution continues Solution 2.0.2.

2.0.3 Python

```
import matplotlib.pyplot as plt

fund_py_means = fund_py.mean()
fig, ax = plt.subplots(figsize=(6, 4), layout="constrained")
ax.hist(fund_py_means, bins=30);
ax.set_xlabel("Sample mean excess return");
ax.set_ylabel("Number of managers");
plt.show()
```



2.0.4 R

```
library(ggplot2) # nolint: undesirable_function_linter

fund_r_means <- map_dbl(fund_r, mean)
fund_r_means_tbl <- tibble(mean = fund_r_means)
means_histogram <- fund_r_means_tbl |>
  ggplot() +
  aes(mean) +
  geom_histogram(bins = 30) +
  labs(x = "Sample mean excess return", y = "Number of managers")
means_histogram
```

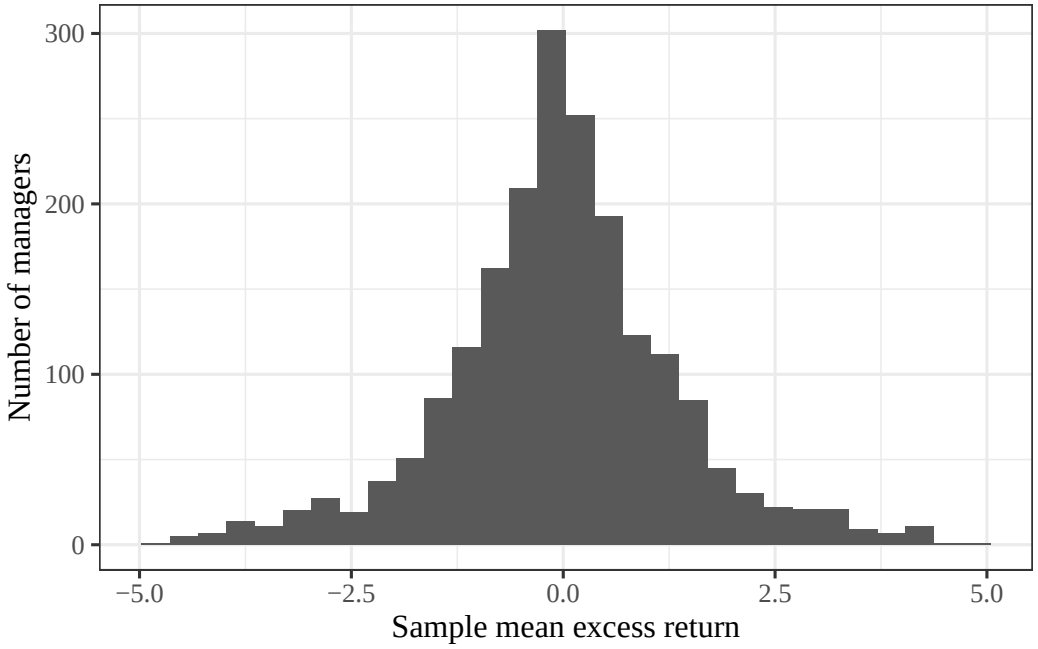


Figure 1: Bars counting the fund managers at each level of excess return, averaged over the months.

```
fund_r_per_bin <- length(fund_r_means) / 30
```

Figure 1 shows a sharp peak at zero and heavy tails on both sides.
I chose 30 bins, which is 67 managers per bar on average. Far fewer bins would hide the shape inside a few wide bars, and far more would leave most bars with only a few managers, so the outline would look ragged. That is why 30 is a compromise between the two.

3 Testing the managers

Exercise 3.1 (Lab: test the first manager)**.** Use the `Fund` data. Run a one-sample t-test[a] of the null hypothesis[b] that `Manager1` has no skill. What is the p-value[c]?

Hint

The `scipy.stats` module has a t-test in Python, while R has `t.test()`.

[a] https://morrison-lab.github.io/sds/basic-statistical-methods.html#def-one-sample-t-test
[b] https://morrison-lab.github.io/sds/inference.html#def-null-hypothesis
[c] https://morrison-lab.github.io/sds/inference.html#def-p-value

Solution

Solution 3.1. This solution continues Solution 2.2.

3.0.1 Python

```
from scipy import stats

fund_py_test = stats.ttest_1samp(fund_py["Manager1"], 0)
fund_py_p = float(fund_py_test.pvalue)
round(fund_py_p, 4)
```

```
0.0062
```

3.0.2 R

```
fund_r_test <- t.test(fund_r$Manager1)
fund_r_p <- fund_r_test$p.value
round(fund_r_p, 4)
```

```
[1] 0.0062
```

The p-value[a] is 0.0062. At significance level[b] 0.05, the test rejects the null hypothesis[c] for this one manager.

[a] https://morrison-lab.github.io/sds/inference.html#def-p-value
[b] https://morrison-lab.github.io/sds/inference.html#def-significance-level
[c] https://morrison-lab.github.io/sds/inference.html#def-null-hypothesis

Exercise 3.2 (Lab: test every manager)**.** Use the Fund data. Run the test of Solution 3.0.2 for each of the managers, then count the managers whose p-value[a] is below 0.05.

Hint

The test routine in Python takes a whole table plus an axis, while in R you can apply the test to each column.

[a] https://morrison-lab.github.io/sds/inference.html#def-p-value

Solution

Solution 3.2. This solution continues Solution 3.0.2.

3.0.3 Python

```
fund_py_all = stats.ttest_1samp(fund_py, 0)
fund_py_ps = fund_py_all.pvalue
int((fund_py_ps < 0.05).sum())
```

```
289
```

3.0.4 R

```
fund_r_pvalues <- map_dbl(fund_r, \(x) t.test(x)$p.value)
sum(fund_r_pvalues < 0.05)
```

```
[1] 289
```

```
fund_r_rejected <- sum(fund_r_pvalues < 0.05)
fund_r_expected <- 0.05 * length(fund_r_pvalues)
```

The test rejects for 289 of the 2000 managers. If no manager had any skill, each test would reject 5%

of the time, so about 100 managers would be rejected by chance alone. This 5% figure is why a count of rejected managers at level 0.05 cannot by itself identify skilled managers.

Exercise 3.3 (Lab: the Bonferroni correction)**.** Use the p-values[a] of Solution 3.0.4. Apply the Bonferroni correction[b] to hold the family-wise error rate[c] at 0.05. How many managers does it reject?

Hint

Multiply each p-value by the count of tests, then compare it with 0.05. In R, `p.adjust()` does the multiplication.

[a] https://morrison-lab.github.io/sds/inference.html#def-p-value
[b] https://morrison-lab.github.io/sds/multiple-testing.html#def-bonferroni
[c] https://morrison-lab.github.io/sds/multiple-testing.html#def-fwer

Solution

Solution 3.3. This solution continues Solution 3.0.4.

3.0.5 Python

```
fund_py_bonf = fund_py_ps * len(fund_py_ps)
int((fund_py_bonf < 0.05).sum())
```

```
0
```

3.0.6 R

```
fund_r_bonf <- p.adjust(fund_r_pvalues, method = "bonferroni")
sum(fund_r_bonf < 0.05)
```

```
[1] 0
```

The correction rejects 0 of the 2000 managers. It rejects far fewer managers than the uncorrected test did, because it guards against any wrong call across all the tests.

Exercise 3.4 (Lab: the Benjamini–Hochberg procedure)**.** Use the p-values[a] of Solution 3.0.4. Apply the Benjamini–Hochberg procedure[b] to hold the false discovery rate[c] at 0.1. How many managers does it reject?

Hint

The `scipy.stats` module adjusts p-values in Python, while R's `p.adjust()` takes the method `"BH"`.

[a] https://morrison-lab.github.io/sds/inference.html#def-p-value
[b] https://morrison-lab.github.io/sds/multiple-testing.html#thm-bh-fdr
[c] https://morrison-lab.github.io/sds/multiple-testing.html#def-fdr

Solution

Solution 3.4. This solution continues Solution 3.0.6.

3.0.7 Python

```
fund_py_bh = stats.false_discovery_control(fund_py_ps)
int((fund_py_bh < 0.1).sum())
```

```
146
```

3.0.8 R

```
fund_r_bh <- p.adjust(fund_r_pvalues, method = "BH")
sum(fund_r_bh < 0.1)
```

```
[1] 146
```

The method rejects 146 managers, more than the correction of Solution 3.0.6 does, because allowing a small share of wrong calls is a weaker aim than allowing none.

[1] G. James, D. Witten, T. Hastie, R. Tibshirani, and J. Taylor, *An introduction to statistical learning: With applications in Python*. Springer, 2023. doi: 10.1007/978-3-031-38747-0[6].

[6] https://doi.org/10.1007/978-3-031-38747-0