

Exercises

Here are some challenges based on everything you've studied so far. Good luck!

We'll cover the following



- Question 1
- Solution 1
- Question 2
- Solution 2
- Question 3
- Solution 3
- Question 4
- Solution 4
- Question 5
- Solution 5

Here is a [data file](#) containing two columns of comma-separated data.

```
100,111
93,103
115,119
97,117
106,116
111,116
111,119
100,103
126,118
93,119
```



Question 1

Write a program to read in the data file into one or more data structures, and print the values out to the screen. You can assume in your program that you know the number of rows of data (10).

main.c

All code files are copied to end of the page...

data.csv

▶

📄

↶

⌕

Solution 1

main.c

data.csv

All code files are copied to end of the page...

▶

📄

↶

⌕

Question 2

Rewrite your program above assuming you don't know in advance how many rows of data you have.

main.c

data.csv

All code files are copied to end of the page...

▶

📄

↶

⌕

Solution 2

main.c

data.csv

All code files are copied to end of the page...

▶

📄

↶





⌕

Question 3





Add to your program a function that computes the value of a t statistic for the difference between means of the two columns of data. Assume it's an unpaired t-test and you can compute t using the following equation:

$$t = \frac{\bar{X}_2 - \bar{X}_1}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

You are given the solution from Question 2.

main.c	All code files are copied to end of the page...
data.csv	
	  





Solution 3

main.c	All code files are copied to end of the page...
data.csv	
	  

Question 4

Implement a bootstrapping test of the t statistic you get above. Iterate `nboot` times, each time taking a random sample (with replacement) from the set of 20 observations, and assigning them to each group, then re-do the t-test. Count up how many times out of `nboot` you get a t value as large or larger as the one you computed above (so this is a one-tailed test). Set `nboot` to 1 million and report execution time. If you have a fast machine set `nboot` to 10 million so you have some dynamic range. If you have a slow machine set `nboot` to $1e5$ (or $1e4$ if it's really slow).

You will continue from Solution 3.

main.c	All code files are copied to end of the page...
data.csv	
	  

Solution 4

main.c	All code files are copied to end of the page...
data.csv	

data.csv

▶

💾

↶

⌵

Question 5

Parallelize the bootstrap loop to make use of multiple CPU cores. Report execution time.

main.c

data.csv

All code files are copied to end of the page...

▶

💾

↶

⌵

Solution 5

main.c

data.csv

All code files are copied to end of the page...

▶

💾

↶

⌵

Code Files Content !!!

⌵⌵⌵ ⌵⌵⌵

| main.c [1]

#include

int main() {

// your code goes here

return 0;

}

```
-----  
| data.csv [1]  
-----
```

```
100,111  
93,103  
115,119  
97,117  
106,116  
111,116  
111,119  
100,103  
126,118  
93,119
```

```
*****
```

```
-----  
| main.c [2]  
-----
```

```
#include  
  
int main(int argc, char *argv[])  
{  
    FILE *fid;  
    int nr = 10;  
  
    // open the file  
    fid = fopen("data.csv", "r");  
    printf("FILE OPENNEED!!!");  
    // load in the data  
    double c1[nr], c2[nr];  
    int i;  
    for (i=0; i  
  
int main() {  
    // your code goes here  
  
    return 0;  
}
```

```
-----  
| data.csv [3]  
-----
```

```
100,111  
93,103  
115,119  
97,117  
106,116  
111,116
```

111,119
100,103
126,118
93,119

main.c [4]

#include

int main(int argc, char *argv[])

```
{
    FILE *fid;
    int nr = 0;
    char buf[255];

    // open the file
    fid = fopen("data.csv", "r");

    // count the number of lines
    while (!feof(fid)) {
        fgets(buf, 255, fid);
        nr++;
    }
    nr--;

    // go back to beginning of file
    fseek(fid, 0, SEEK_SET);

    // load in the data
    double c1[nr], c2[nr];
    int i;
    for (i=0; i
```

int main(int argc, char *argv[])

```
{
    FILE *fid;
    int nr = 0;
    char buf[255];

    // open the file
    fid = fopen("data.csv", "r");

    // count the number of lines
    while (!feof(fid)) {
        fgets(buf, 255, fid);
        nr++;
    }
    nr--;

    // go back to beginning of file
    fseek(fid, 0, SEEK_SET);
```

```

// load in the data
double c1[nr], c2[nr];
int i;
for (i=0; i
#include

double mean(int n, double v[n]);
double sd(int n, double v[n]);
double ttest(int n1, int n2, double v1[n1], double v2[n2]);

int main(int argc, char *argv[])
{
    FILE *fid;
    int nr = 0;
    char buf[255];

    // open the file
    fid = fopen("data.csv", "r");

    // count the number of lines
    while (!feof(fid)) {
        fgets(buf, 255, fid);
        nr++;
    }
    nr--;

    // go back to beginning of file
    fseek(fid, 0, SEEK_SET);

    // load in the data
    double c1[nr], c2[nr];
    int i;
    for (i=0; i
#include

double mean(int n, double v[n]);
double sd(int n, double v[n]);
double ttest(int n1, int n2, double v1[n1], double v2[n2]);

int main(int argc, char *argv[])
{
    FILE *fid;
    int nr = 0;
    char buf[255];

    // open the file
    fid = fopen("data.csv", "r");

    // count the number of lines
    while (!feof(fid)) {
        fgets(buf, 255, fid);
        nr++;
    }
    nr--;

    // go back to beginning of file
    fseek(fid, 0, SEEK_SET);

    // load in the data
    double c1[nr], c2[nr];
    int i;
    for (i=0; i

```

```

#include
#include
#include

double mean(int n, double v[n]);
double sd(int n, double v[n]);
double ttest(int n1, int n2, double v1[n1], double v2[n2]);

int main(int argc, char *argv[])
{
    FILE *fid;
    int nr = 0;
    char buf[255];

    // open the file
    fid = fopen("data.csv", "r");

    // count the number of lines
    while (!feof(fid)) {
        fgets(buf, 255, fid);
        nr++;
    }
    nr--;

    // go back to beginning of file
    fseek(fid, 0, SEEK_SET);

    // load in the data
    double c1[nr], c2[nr];
    int i;
    for (i=0; i< nr; i++) tcount++;
    }

    printf("%d/%d, p=%.5f\n", tcount, nboot, (double)tcount/nboot);

    return 0;
}

// helper functions

double mean(int n, double v[n])
{
    double sum = 0.0;
    int i;
    for (i=0; i< n; i++)
        sum += v[i];
    return sum/n;
}

double sd(int n, double v[n])
{
    double sum = 0.0;
    int i;
    for (i=0; i< n; i++)
        sum += (v[i] - mean(n, v))^2;
    return sqrt(sum/n);
}

double ttest(int n1, int n2, double v1[n1], double v2[n2]);

int main(int argc, char *argv[])
{
    FILE *fid;
    int nr = 0;
    char buf[255];

    // open the file
    fid = fopen("data.csv", "r");

```



```

// count the number of lines
while (!feof(fid)) {
    fgets(buf, 255, fid);
    nr++;
}
nr--;

// go back to beginning of file
fseek(fid, 0, SEEK_SET);

// load in the data
double c1[nr], c2[nr];
int i;
for (i=0; i= tobs) tcount++;
}

printf("%d/%d, p=%.5f\n", tcount, nboot, (double)tcount/nboot);

return 0;
}

```

// helper functions

```
double mean(int n, double v[n])
```

```
{
    double sum = 0.0;
    int i;
    for (i=0; i
```

```
#include
#include
#include
#include

```

```
double mean(int n, double v[n]);
```

```
double sd(int n, double v[n]);
```

```
double ttest(int n1, int n2, double v1[n1], double v2[n2]);
```

```
int main(int argc, char *argv[])
```

```
{
    FILE *fid;
    int nr = 0;
    char buf[255];

```

```
// open the file
fid = fopen("data.csv", "r");

```

```
// count the number of lines
while (!feof(fid)) {
    fgets(buf, 255, fid);
    nr++;
}
nr--;

```

```
// go back to beginning of file
fseek(fid, 0, SEEK_SET);

```

```
// load in the data
double c1[nr], c2[nr];
int i;
for (i=0; i= tobs) tcount++;

```

```
}

printf("%d/%d, p=%.5f\n", tcount, nboot, (double)tcount/nboot);

return 0;
}

// helper functions

double mean(int n, double v[n])
{
    double sum = 0.0;
    int i;
    for (i=0; i
```