Part 2. Analyze mapped BS-seq data

1. Basic MATLAB

2. Compare one sample to another sample using a Binomial test

3. Compare groups of samples to each other using a *t*-test

Part 2. Analyze mapped BS-seq data

1. Basic MATLAB
    - Matrix and cell objects
    - Objects and functions
    - Logicals and logical operations
    - Loops
    - Load data (workspace or data tables)
    - Save data (workspace or variables)
2. Compare one sample to another sample using a Binomial test
3. Compare groups of samples to each other using a *t*-test

Basic MATLAB: Matrix objects

- Matrix of numbers

```
a = [1, 2, 2, 3, 5];    % Makes a 1x5 matrix
```

a is the object/variable created

= anything after the equal sign is assigned to a. If a exists, it will be replaced

[ ] inside the brackets are the contents of the matrix. Can also use ( ) if concatenation is not necessary

; suppress output. e.g. if you don't want to print 1 million rows on the screen

% comment anything after this sign

- Examples

```
b = [1; 2; 2; 3; 5];   % Makes a 5x1 matrix

c = zeros(10,1);       % Makes a 10x1 matrix of zeros

d = [ ];               % Makes an empty matrix

e(10,2) = 3;    % assign '3' to row 10 and column 2 of matrix e

% if e doesn't exist, it will create it, if it exists but is
smaller, it will extend it
```

- *Note: spaces don't matter so much in Matlab matrices, and matrix operations. Try it*

Basic MATLAB: Cell objects

- Cell arrays:

```
f = {'Hello world'};    % Makes a 1x1 cell array, with a defined string

g = cell(20,1);         % Makes a 20x1 empty cell array
```

- { } curly brackets make a cell array
- ' ' single quotes enclose a string
- Cell arrays can be multidimensional arrays of cells or matrices. Use { } when assigning cells/ matrices to a multidimensional array:

```
h(1,1) = {cell(10,1)};

h(2,1) = {cell(200,2)};

h(3,1) = {ones(20,1)};
```

- If you are storing a multi-dimensional cell array, you must enclose it with { }

```
h(3,1) = {zeros(20,1)};    % store a matrix of ones in the array
```

- To access the contents:

```
h(3,1)

h{3,1}
```

Basic MATLAB: Objects and functions

- Functions can perform an operation on objects, or defined input.
- Make a matrix `j`. Note that `,` concatenates horizontally, and `;` concatenates vertically

```
j = [1,2,3; 4,5,6;7,8,9]

k = mean(j)     %By default mean along first dimension (mean of columns)

l = mean(j,2)   %Mean along second dimension (mean of rows)
```

- mean is the function. The input for the function must be enclosed in ( ). The input can be an object, as above, or also defined input:

```
m = mean([1,2,3])
```

- The function usage and input allowed will vary depending on the function. But it is very easy to see this using the Matlab HELP menu, which also has examples.
- Example of *ttest2* function. This particular function has a variable number of outputs, specified before the = sign. You can request to get the first output only (in this ex it is stored in h), or both in h and p, as [h,p]

```
n = [2,2,3,3,4,4];  o = [5,5,6,6,7];

[h,p] = ttest2(n,o)
```

- Note that the labels for the output objects are arbitrary. You can name objects whatever you want, as long as you don't give them reserved keywords of the Matlab language (ex: if, else,end, for)

```
[orange, apple] = ttest2(j(1,:), j(2,:));
```

- `j(1,:)` stands for contents of matrix j, row 1, all columns. Alternatively, `j(:,3)` stands for all rows, column 3 of matrix j

Basic MATLAB: Logicals and logical operations

- Logical data types represent the logical state true or false, represented by 1 and 0, respectively
- Examples:

```
10 > 40                  %is 10 greater than 40?

r = [30, 40, 50, 60, 70]

r > 40                   %which elements of r are greater than 40?

ans =

     0     0     1     1     1

s = [16,2,3; 5,11,10] % a 2x3 matrix

t = s<=10              % returns a 2x3 logical matrix, each entry is true or
false for the logical test
```

- Logical operators **==** (is equal), **~=** (is not equal), **&** (AND), **|** (OR)

```
a = [0.1; 0.5; 1]; b = [0.01; 0.04; 0.6];

c = a==0.5    % which rows of a are equal to 0.5

d = b <0.05   % which rows of b are less than 0.05

e = c&d       % which rows are true in both c and d

f = c|d       % which rows are true in either c or d

g = [a,b,c,d,e,f]      % let's look at the outputs in one table
```

Basic MATLAB: Loops

- Useful if you want to repeat an operation, or series of operations, many times. ex: for sequencing data with millions of rows
- Simple loop example. Note that every for or if statement must close with an end.

```
for i=1:10          % i=1:10 is same as i=[1,2,3,4,5,6,7,8,9,10]
    i               % print i on screen
end                 % end loop
```

- Loop performing a function

```
a = [2,4,3,4,3; 4,2,4,5,5; 0,0,0.2,0.2,0.1];    %3x5 matrix with some data
b = zeros(3,1);                 %pre-allocate a matrix for the output
for i=1:3                       %for i=1 or 2 or 3
    b(i) = std(a(i,:));         %calculate standard deviation of row i of a
                                %Store the output in the i-th row of b
end
```

Basic MATLAB: Loops

- Loop using relational operators (>, <, >=, <=)

```
c = [1,4,5];  d = [3,3,5];

e = zeros(3,1);

for i=1:3

    if c(i) > d(i)          % if this statement is true

        e(i) = c(i);        % do this

    elseif c(i) < d(i)      % otherwise if this is true

        e(i) = d(i);        % do this

    else                    % if none of the above apply

        e(i) = nan;         % assign missing value 'nan'

    end

end
```

Basic MATLAB: Load data

- Load workspace/data using the GUI:

  `File -> Import data`

- Load a Matlab workspace using the command line:

```
load BSseq_data.mat  %loads workspace from current directory

load('~/Workshop6/BSseq_data.mat')  %loads from a specific directory
```

- Copy/paste a data table into the workspace
- Load a data table by dragging and dropping into the workspace
- Load a data table using the command line. There are many ways of doing this. Note that the file name needs to be enclosed in quotes, otherwise Matlab will look for an object in the workspace with this name. Ex:

```
x = load('test_data.txt');      %this works if you have a data table with
numbers only

y = importdata('file1.wig');   %will load data with strings and numbers. If
both are present, it will create a structure with a cell array (for strings)
and a matrix (for numeric data)
```

- Since memory is limited, for very large data files I typically make a file(s) with only the fields that I need using Unix or Python, then import these into Matlab for downstream analysis.

Basic MATLAB: Save data

- Save Matlab workspace using the GUI:

  `File -> Save Workspace As`

- Save workspace using the command line:

```
save BSseq_data.mat  %saves workspace (all variables)to current
directory

save('~/Desktop/Workshop6/BSseq_data.mat')  % saves workspace (all
variables) variable specific directory
```

- Save a specific variable/object to a file. There are a lot of options, see Help for specific format/delimiter, etc.

```
save('varX.txt', 'x', '-ascii', '-tabs')
```

save to this file

this variable (s) 'x', 'y', etc.

format

delimiter

Part 2. Analyze mapped BS-seq data

1. Basic MATLAB

2. Compare one sample to another sample using a Binomial test

3. Compare groups of samples to each other using a *t*-test

Compare individual samples: Motivation and approach

- **Motivation:** Researchers often wish to compare samples in different conditions. However, due to sequencing costs, they often start by sequencing only one sample from each. For example, samples could be:
    - Control vs treated
    - Wild-type vs knock-out
- **Approach:** Use the Binomial distribution. Matlab function `binofit`
    - Compare methylation level of the two samples at individual cytosines. i.e. compare cytosine %methylation at chrN bpX in the two samples
    - Use the average methylation (p) and the number of counts at each site (n) as parameters for the binomial distribution
    - Construct a distribution for each sample, and calculate confidence intervals to determine if they are differentially methylated
    - Repeat for all cytosines

Compare individual samples: Binomial test approach

Binomial distribution is defined by parameters p and n

$$P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}, \quad x = 0, 1, 2, 3, \cdots, n$$

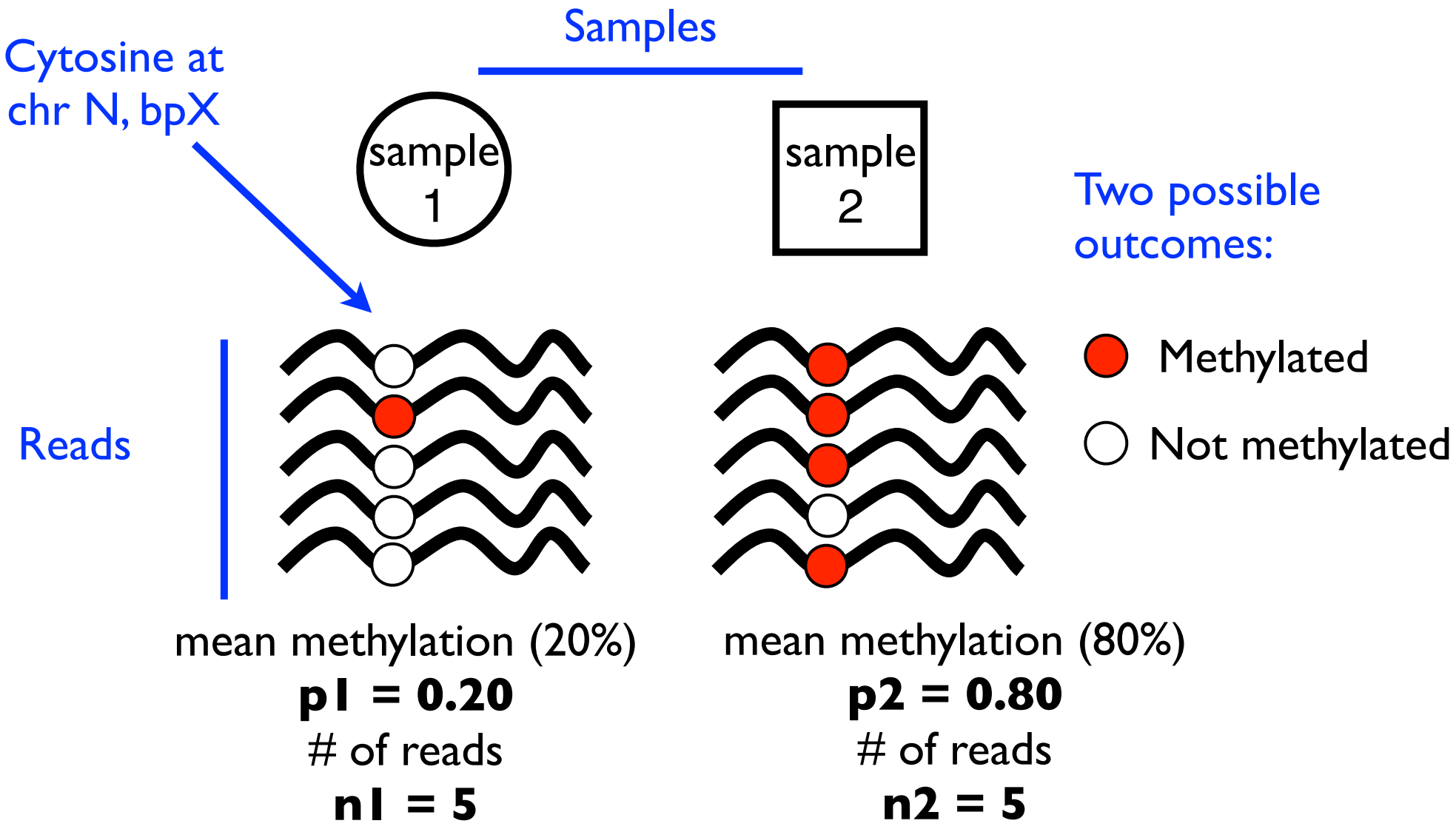
Suppose that n independent Bernoulli trials are performed (i.e. a trial with two possible outcomes), each trial having probability of success p.

Let **X** be the number of successes among the n trials.

We say that **X** follows the binomial probability distribution with parameters n, p.

It gives the probability of successes *x*, in n trials

Compare individual samples: Binomial test approach



p and **n** are the parameters for the binomial distribution
probability of outcome **p** and number of trials **n**

Compare individual samples: Binomial test approach

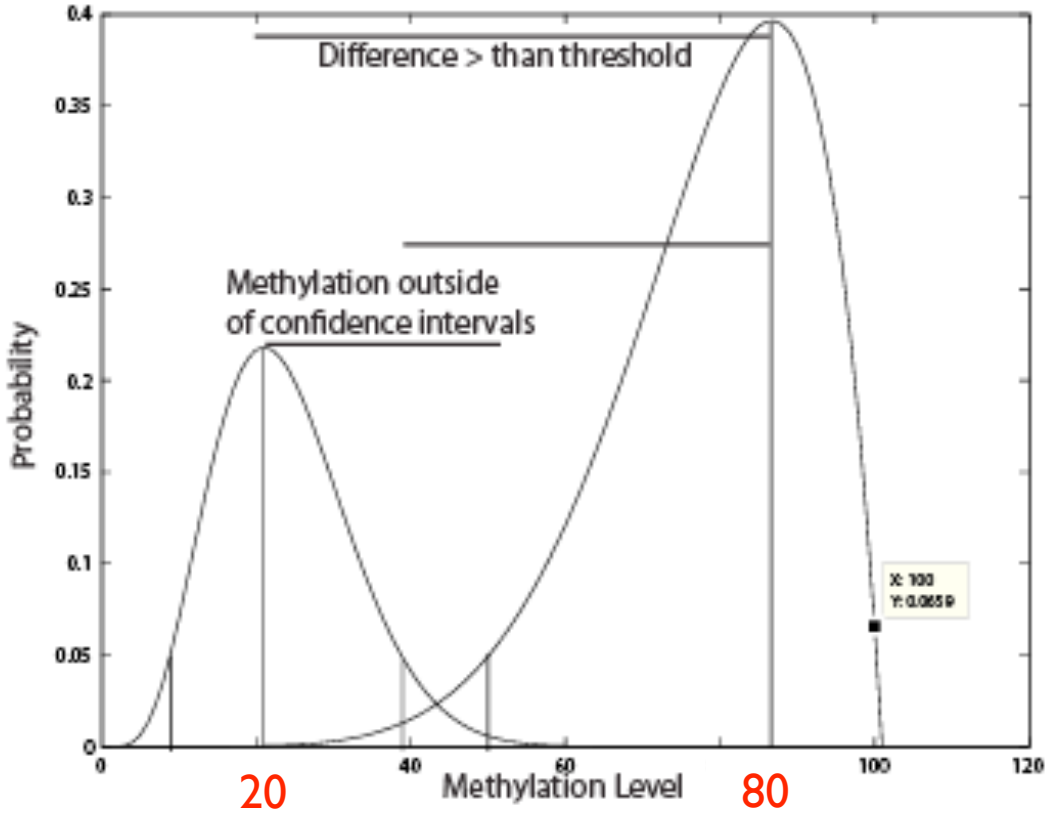
1. Select data to do analysis on. ex: select cytosines covered by at least 10 reads (arbitrary)

2. Determine n and p for each sample (n1, p1, n2, p2)

3. For each cytosine, construct a binomial distribution for each sample, and determine the confidence intervals of each using function binofit

4. Determine the difference in methylation between the two samples, at that cytosine. **delta** = abs(p1-p2)

5. Call a cytosine differentially methylated if it passes **binofit** confidence intervals and **delta**>threshold (ex: 50%)



Compare individual samples: Binomial test approach

Load sample data

- Change Matlab current directory to the directory where you stored the file BSseq_data.mat

```
load BSseq_data.mat
```

- You should have two matrix variables in the workspace called Sample1_data and Sample2_data, and a cell called Headers_data

- Columns of these data matrices are:

1) chromosome
2) base pair position
3) %Methylation level
4) number or reads covering that position methylated
5) total number or reads covering that position

Compare individual samples: Binomial test approach

1. Select data to do analysis on. Select cytosines covered by at least 10 reads, and less than 200 reads

```
a = Sample1_data(:,5);  %number or reads covering each site

b = a>=10;  %logical, which rows are >=10

c = a<200;  %logical, which rows are <200

d = b&c;    %rows which meet both criteria

Sample1_data= Sample1_data(d,:);  %keep data for rows which meet this criteria
```

Compare individual samples: Binomial test approach

1. Select data to do analysis on. Select cytosines covered by at least 10 reads, and less than 200 reads

```
%Repeat for Sample2
a = Sample2_data(:,5);   %number or reads covering each
site

b = a>=10;  %logical, which rows are >=10

c = a<200;  %logical, which rows are <200

d = b&c;    % rows which meet both criteria

Sample2_data = Sample2_data(d,:);  %keep data for rows
which meet this criteria
```

Compare individual samples: Binomial test approach

1. Select data to do analysis on. Select cytosines covered by at least 10 reads, and less than 200 reads

```
% Select cytosines with data in both samples (since we
compare site by site)

[in,ina,inb] = intersect(Sample1_data(:,1:2), Sample2_data(:,1:2), 'rows');

Sample1_data = Sample1_data(ina,:);  %keep intersecting
rows

Sample2_data = Sample2_data(inb,:);
```

Compare individual samples: Binomial test approach

2. Determine n and p for each sample (n1, p1, n2, p2)

```
% n is already given in the output, total number of counts
n1 = Sample1_data(:,5);

n2 = Sample2_data(:,5);

% p, or %methylation level, is already given in column 3
p1 = Sample1_data(:,3);

p2 = Sample2_data(:,3);
```

Compare individual samples: Binomial test approach

3. For each C, construct a binomial distribution for each sample, and determine the confidence intervals of each using binofit

```
% You select a threshold,such that binofit returns a
% confidence interval of 100(1-alpha)
% For a 95% confidence interval:
alpha = 0.05;

% The confidence interval is returned in pci
% pci has 2 columns, 1)lower bound, 2) upper bound of CI

tic
[phat_1, pci_1] = binofit(p1, n1, alpha);
toc

[phat_2, pci_2] = binofit(p2, n2, alpha);
```

Compare individual samples: Binomial test approach

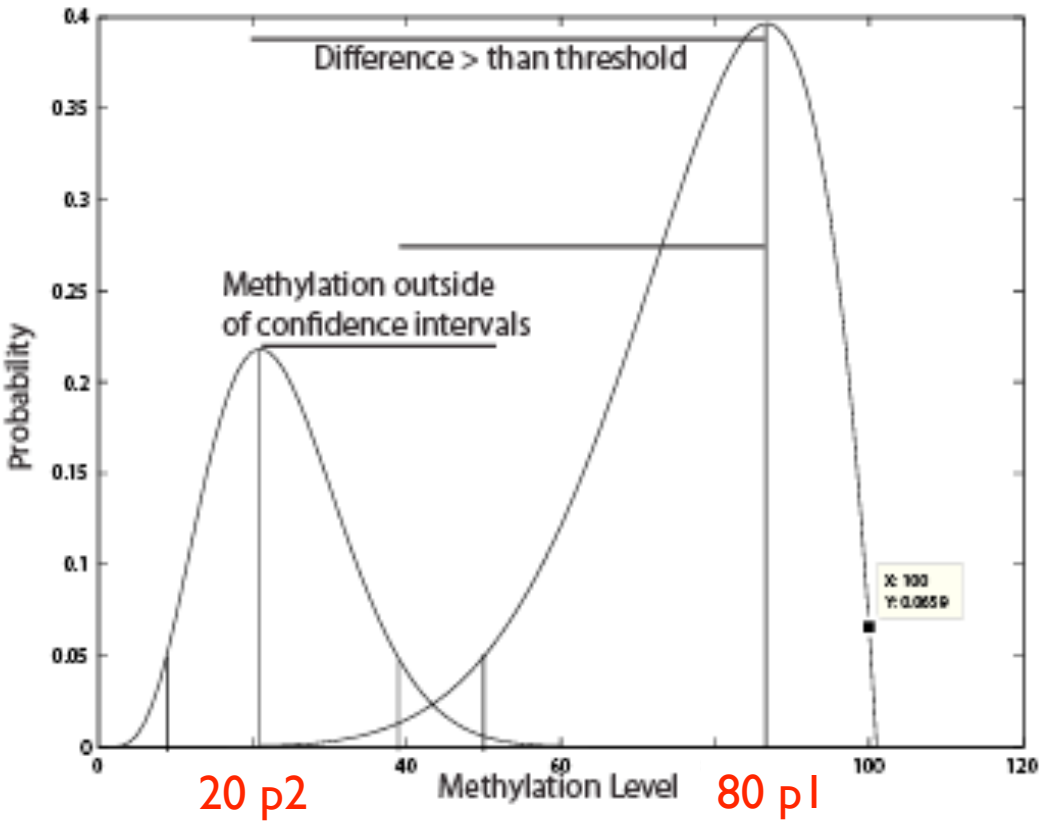
3B. For each C, determine if the mean of one sample (p1) lies outside of the confidence interval of the other sample pci2, and viceversa

```
x = (p1 < pci_2(:,1)) & (p2 > pci_1(:,2));

y = (p1 > pci_2(:,2)) & (p2 < pci_1(:,1));

binomial_pass = x|y;

%logical indicating if
% binomial test passed
```



Compare individual samples: Binomial test approach

4. Determine the difference in methylation between the two samples, at that cytosine. delta = absolute difference between p1 and p2

```
delta = abs(p1-p2);
```

Compare individual samples: Binomial test approach

5. Call a cytosine differentially methylated if it passes binofit and delta>threshold (ex: 50%)

```
% logical for binomial test pass/not pass
binomial_pass;
% difference in %met between samples
delta;

% rows for cytosines which pass both criteria
pass_test  = binomial_pass & (delta>0.50);

% select sites and info for sites
pass_sites_data  = Sample1_data(pass_test,:);

pass_sites_delta = delta(pass_test);

pass_sites_pci   =
[pci_1(pass_test,:),pci_2(pass_test,:)];
```

Compare individual samples: Summarize results

6. Summarize results. Create a matrix with chromosome, bp, mean methylation level for each sample, delta, confidence interval for each sample, and logical column stating if it passed threshold we selected

```
Binomial_summary = [Sample1_data(:,1:2),p1, ...
                    p2, delta, pci_1, pci_2,pass_test];


temp = Binomial_summary(pass_test,:);
```

Part 2. Analyze mapped BS-seq data

1. Basic MATLAB

2. Compare one sample to another sample using a Binomial test

3. Compare groups of samples to each other using a *t*-test

Compare groups of samples: Motivation and approach

- **Motivation:** In moving forward with research projects, investigators will typically examine multiple samples from each condition of interest. In this case a *t*-test is a suitable method for methylation levels between groups of samples. Ex:
    - 4 Control vs 4 treated
    - 5 Wild-type vs 5 knock-out

  As a rule, the more replicates you have per condition, the better. As a minimum, three replicates per condition.

- **Approach:** Test for a difference in means. Matlab function `ttest2`
    - At each cytosine, determine the %methylation level for all samples (output in .CGmap file)
    - At each cytosine, compare the average (mean) methylation level in ConditionA to the average methylation level in ConditionB
    - Select cytosine sites that are significant for *t*-test, show FDR<5% and delta>50%

Compare groups of samples: Load data

1. At each cytosine, determine the %methylation level for all samples (output in .CGmap file)

- Select sites that have coverage (number reads covering each sites) greater than 10 and less than 200. Same as we did before

- For each condition, make a data matrix for cytosines sites x samples, each column is a replicate. The data is the %methylation level from the .CGmap file

- Data variables for this should be on your workspace. Columns are 1)chromosome, 2)bp position, 3:end) data:

```
ConditionA_data;

ConditionB_data;
```

Compare groups of samples: Select data

2. Select sites where we have at least 3 replicates per condition:

```
% Which data rows and columns DO NOT have missing
values
a = ~isnan(ConditionA_data(:,3:end));

% How many total data columns, per site
b = sum(a,2);

% Which sites have at least 3 data columns
c = b>=3;

%how many sites meet criteria?
sum(c)

%keep these sites
ConditionA_data = ConditionA_data(c,:);
```

Compare groups of samples: Select data

2. Select sites where we have at least 3 replicates per condition. Repeat for Condition B:

```
% Which data rows and columns DO NOT have missing
values
a = ~isnan(ConditionB_data(:,3:end));

% How many total data column, per site
b = sum(a,2);

% Which sites have at least 3 data columns
c = b>=3;

%how many sites meet criteria?
sum(c)

%keep these sites
ConditionB_data = ConditionB_data(c,:);
```

Compare groups of samples: Select data

3. Select sites where we have data for both Condition A and B. i.e. intersecting rows:

```
%Intersect chr and bp position of each data set
[in,ina,inb] = intersect(ConditionA_data(:,1:2),ConditionB_data(:,1:2),'rows');

%Keep sites present in both
ConditionA_data = ConditionA_data(ina,:);

ConditionB_data = ConditionB_data(inb,:);
```

Compare groups of samples: t-test

4. For each site (row i), perform t-test using data from Condition A and Condition B:

```
%pre-allocate a matrix for the output p-value
zeros(nrows,ncols)
p = zeros(length(ConditionA_data),1);

%iterate through each row, repeat until all rows are done
for i=1:length(p)

   %select data from condition A, for row i
   dataA = ConditionA_data(i,3:end);

   %select data from condition B, for row i
   dataB = ConditionB_data(i,3:end);

   %t-test. Store result in i-th row of p
   [h,p(i)] = ttest2(dataA,dataB);

end
```

Compare groups of samples: FDR for the *t*-test p-values

5. False discovery rate on the p-values. Note: FDR is calculated based on the Storey et al. method. Other methods of estimating false discovery rate include permutation and simulation.

```
%Output is an FDR for each p-value
%Dimensions of fdr and p are the same
fdr = mafdr(p);

%Which sites show FDR<5%
pass_fdr = fdr<0.05;
```

Compare groups of samples: Delta threshold

6. Determine which sites show difference in average methylation level >50%

```
%Calculate average methylation level for each row, for
both samples
%Note that we use 'nanmean' instead of 'mean'
met_meanA = nanmean(ConditionA_data(:,3:end),2);
met_meanB = nanmean(ConditionB_data(:,3:end),2);

%For each site, calculate the absolute difference in
means, or delta
delta = abs(met_meanA - met_meanB);

%Which sites show delta>50%
pass_delta = delta>0.5;
```

Compare groups of samples: Select sites that pass threshold

6. Determine which sites show FDR<5% and show difference in average methylation level >50%

```
We previous determined which sites show FDR<5%
pass_fdr;    %this is a logical, true or false

%We previous determined which sites show FDR<5%
pass_delta;  %also a logical

%Which sites pass both fdr and delta threshold
pass_fdr_delta = pass_fdr & pass_delta;

(fdr<0.05)& (delta>0.5)
%How many sites
sum(pass_fdr_delta)
```

Compare groups of samples: Summarize results

7. Summarize results. Create a matrix with chromosome, bp, mean methylation level for each condition, delta, p-value for t-test and FDR, and logical column stating if it passed threshold we selected

```
Ttest_summary = [ConditionA_data(:,1:2),met_meanA, ...
                 met_meanB, delta, p, fdr, ...
                 pass_fdr_delta];


temp = Ttest_summary(pass_fdr_delta,:);
```