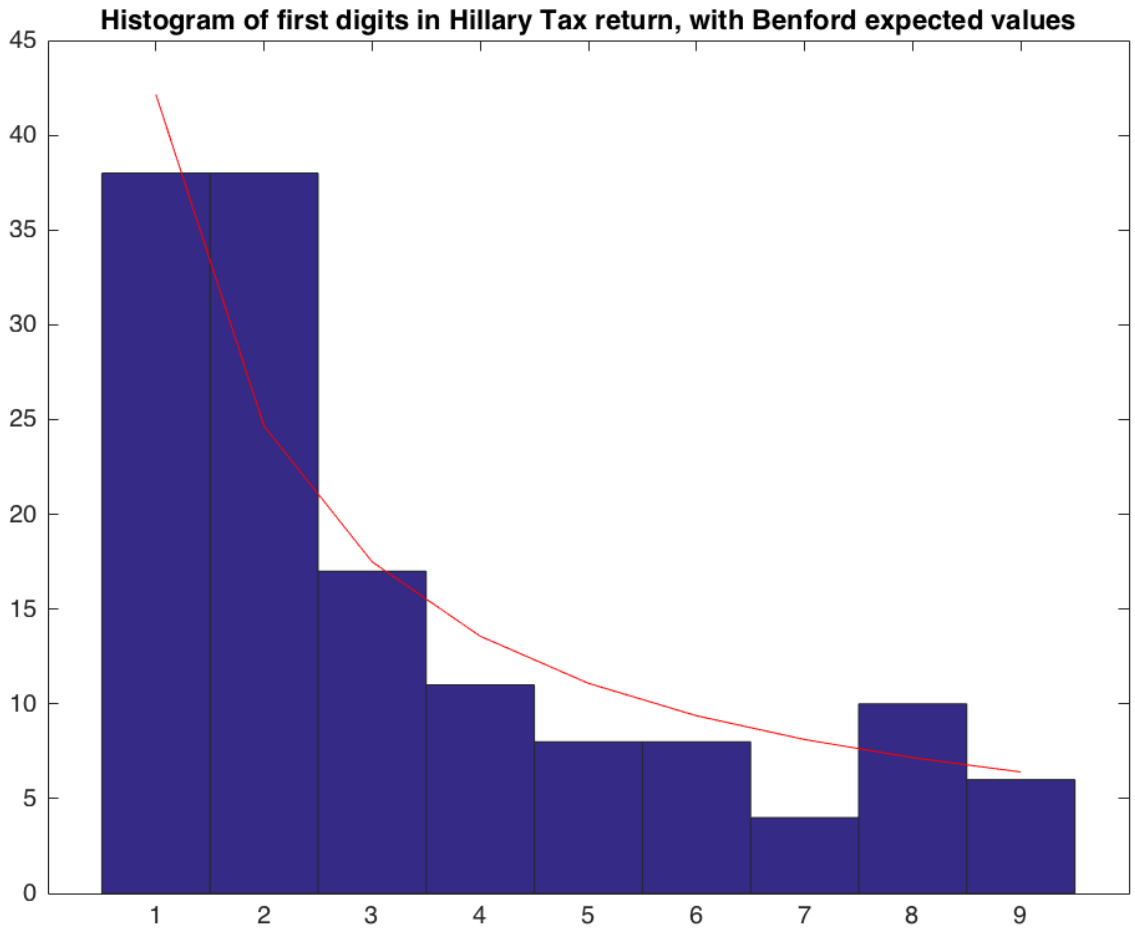
Homework 5

Ruth Johnson - 704 275 412

Problem 1 (Income Tax)

The ChiSquareStatistic = 12.4337 and the ChiSquareProbability = .8671. This says that the probability of getting a chi square statistic that is smaller than this is about 87%, or that about 87% of the time we would expect to get something smaller, and about 13% of the time something larger. Thus, we can conclude that this particluar income tax statement does not violate Benford's Law, and that these statements are true.



```
filename = 'HR_Clinton_2014_tax_return_numbers.txt';
firstdigit = @(x) floor(x ./ (10 .^ floor(log10(x))));
comma_delete(filename, 'data.txt');

digits = importdata('data.txt', '\n');
digits = unique(digits);
digits = firstdigit(digits);

number_of_bins = 9;
nu = number_of_bins - 1;

HillaryHistogram = hist(digits,1:9);
figure
hist(digits,1:9);

BenfordProbilities = diff(log10(1:10));
N = length(digits);
BenfordHistogram = N * BenfordProbilities;
hold on
plot(1:9, BenfordHistogram, 'r');
title('Histogram of first digits in Hillary Tax return, with Benford expected
values')
hold off

ChiSquareStatistic = sum((HillaryHistogram - BenfordHistogram).^2 ./ BenfordHi
stogram);

ChiSquareProbability = cdf('Chisquare', ChiSquareStatistic, nu);
```

Problem 2

1. The simplified function: f = @(t) (20*cos(t) - pX).^2 + (10*sin(t) - pY).^2
2. 1. timestep Δt = 0.01

2. In this case, the only external forces acting on the particle is gravity, which is represented by:

   `gravity = [0 -9.81]` `

3. We know if a particel is outside the ellipse by the following condition. If it is satisfied,then we handle the condition that if'ts outside the ellipse and recalculate it's new trajectory after "bouncing" off of the edge of the ellipse.

   `if( implicitEllipse( pX, pY ) > 1 )`

What is the timestep Δt? Which external forces are acting on the particles and what are their vector representations? How do you know if a particle is outside the ellipse?

3.

```
for I = 1:particlesCount
    particles(I).position = [P(I,1), P(I,2)];       % Position.
    particles(I).velocity = [P(I,3), P(I,4)];        % Velocity.
    particles(I).color = [P(I,5), P(I,6), P(I,7)];          % Color.
    particles(I).mass = [P(I,8)];          % Mass.
end;
```

4.

```
%find new acceleration
        particles(I).force = gravity;
        acc = particles(I).force;
        %keep old velocity
        oldVel = particles(I).velocity;
        %new velocity
        particles(I).velocity = particles(I).velocity + deltaT * acc;
        %new position
        particles(I).position = particles(I).position + deltaT * oldVel;
```

5.

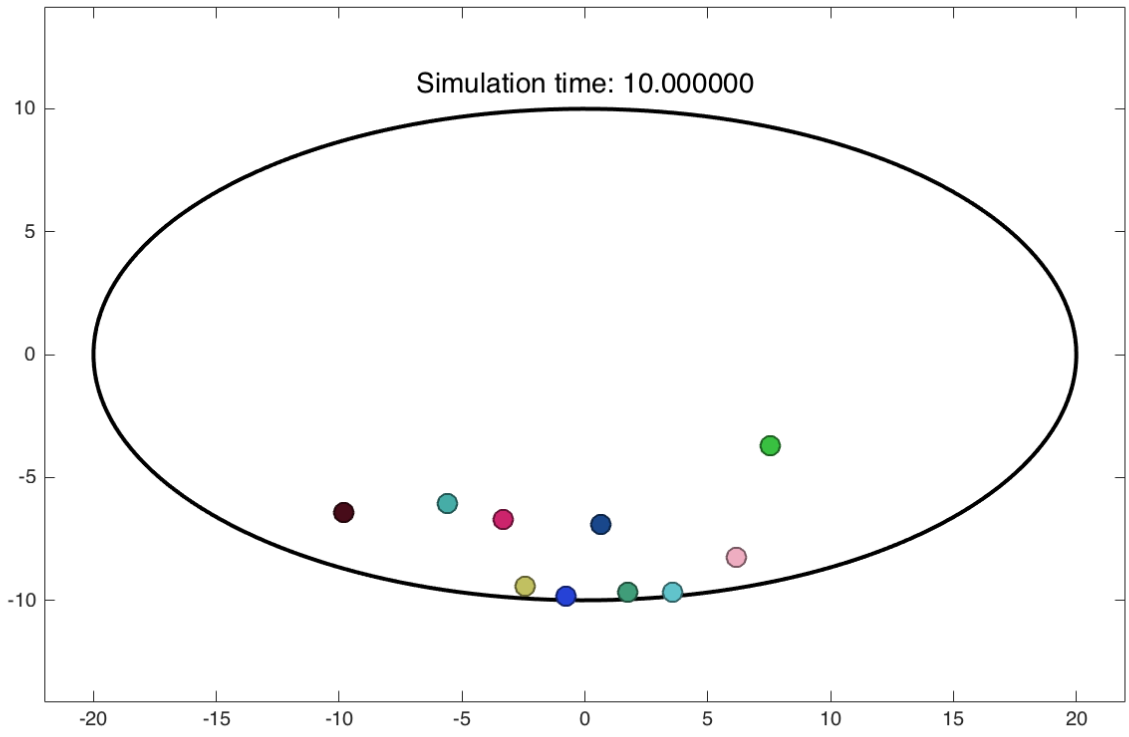
```
  %Function for wich you want to find a root.
            f = @(t) (20*cos(t) - pX).^2 + (10*sin(t) - pY).^2;
            fx = @(t) (20*cos(t));
            y = @(t) 10*sin(t);

            df = @(t) 2*(20*cos(t) - pX) * -20*sin(t) + 2*(10*sin(t) - pY) * +1
0*cos(t);
            %Pick an initial value to launch Newton's method.
            t0 = atan2(pY, pX);

            %t value for which the distance between ellipse and particle is min
imal.
            tStar = fzero(df,t0);

            %Use tStar to relocate particle ON the ellipse
            particles(I).position =[fx(tStar), y(tStar)];
```

6.



Problem 3 (Logistic Regression)

We implemented the objective function as follows:

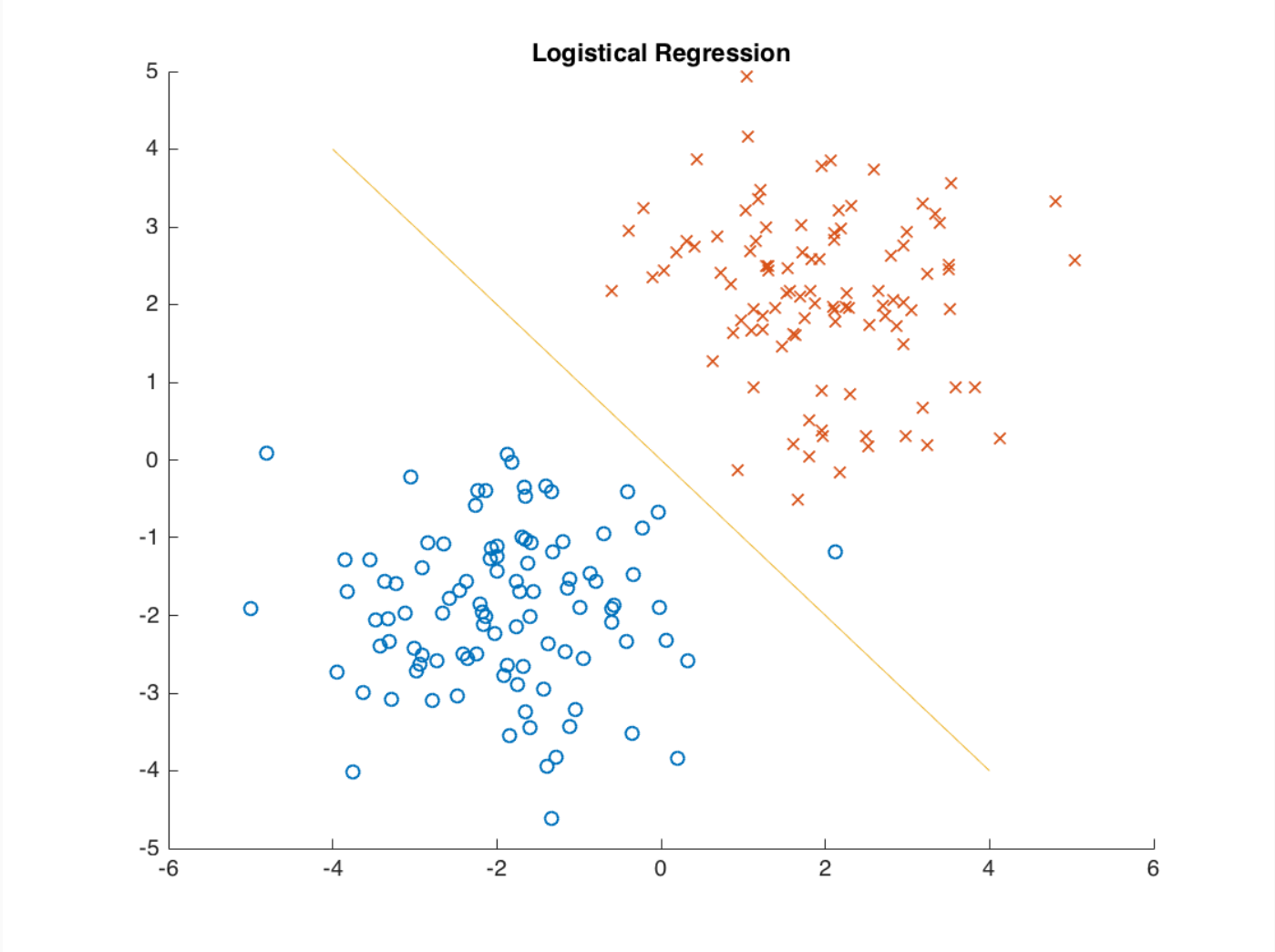
```
function J = cost(X, y, theta)
    %
    % Cost function for logistic regression
    %
    % TO-DO: Implement the cost function for logistic regression.
    % Input:
    %   X: A 200x2 matrix containing all data points
    %   Y: A 200x1 matrix containing labels of data points
    %   theta: the current parameter theta
    % Output:
    %   J: The cost of logistic regression with the current theta
    %
    [m,~] = size(X);
     sum = 0;
    for i=1:m
      y1 = y(i);
      x1 = X(i,1);
      x2 = X(i,2);
      h = 1 /(1 + exp(dot(-theta,[x1,x2,1]')));
      part1 = dot(-y1,log(h));
      part2 = dot((1-y1),log(1-h));
      temp = part1 - part2;
      %temp = -y1*log(1./(1 + exp(dot(-theta,[x1,x2,1])))) - (1-y1)*log(1-(1./
(1 + exp(dot(-theta,[x1,x2,1])))));
      sum = sum + temp;
    end

        J = sum./m; % Replace it!
end
```

We found our best theta to be the following:

```
1.0e-03 * ( 0.0833    0.0833   -0.1250)
```

Below is the plot with the found decision boundary for the data of faces.



MATLAB Script

```
f = @(theta) cost(data(:,1:2),data(:,3),[0,0,0]);

best_theta = [0, 0, 0]; % Replace it!
best_theta = fminsearch(f,[0,0,0]);

% Plot decision boundary
figure, hold on;

scatter(data(1:(m/2), 1), data(1:(m/2), 2));
scatter(data((m/2+1):m, 1), data((m/2+1):m, 2), 'x');
line_eq = @(x) -best_theta(3) - best_theta(2)*x/best_theta(1);
plot([-4, 4], [line_eq(-4), line_eq(4)]);
```