Lecture 9 Exercise

Question 1

Logistic regression using two independent variables. Use the 'iris' dataset from the package RDatasets to answer the following questions.

Part a

Using Julia, select the samples for species 'versicolor' and 'virginica'. Hint: Load data and select relevant samples using the code:

In [2]:
```
using RDatasets;
iris = dataset("datasets", "iris");
groups = groupby(iris,:Species);
new_iris = vcat(groups[2],groups[3]);
```

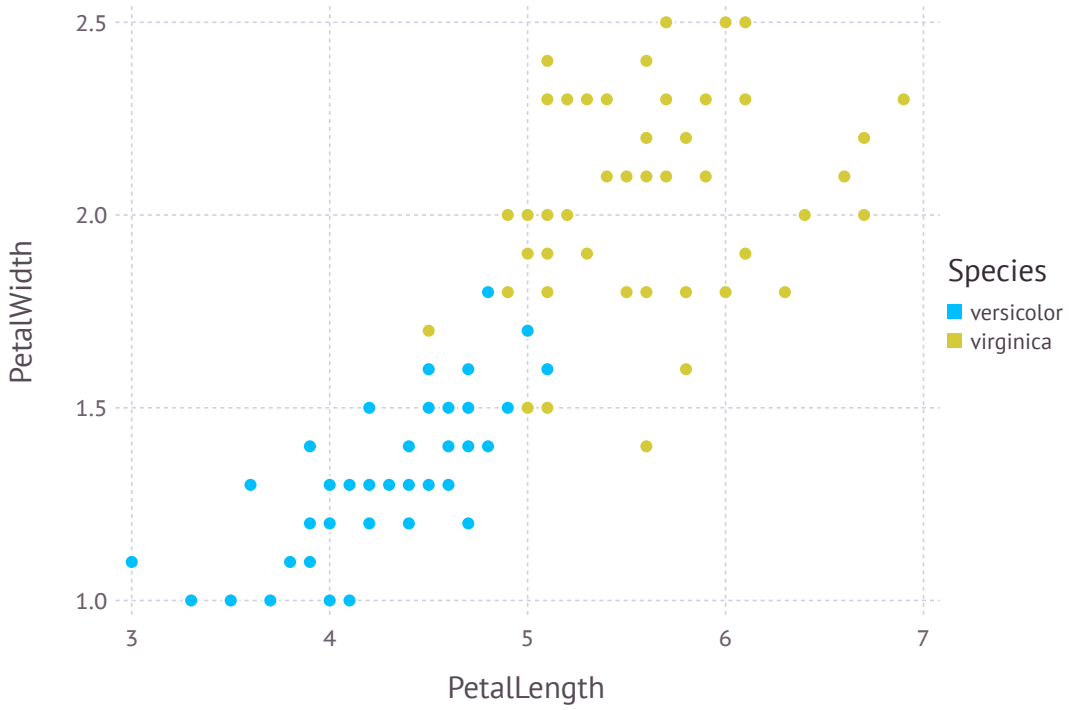
```
WARNING: Method definition unix2zdt(Real) in module TimeZones at /Us
ers/vulcan/.julia/v0.6/TimeZones/src/conversions.jl:122 overwritten
in module RData at /Users/vulcan/.julia/v0.6/RData/src/convert.jl:20
1.
```

Part b

Generate a scatter plot between variables 'PetalLength' and 'PetalWidth', coloring each point based on 'Species'. (Use Gadfly.plot() with Geom.point option)

In [6]:
```
using Gadfly;
Gadfly.plot(new_iris, x=:PetalLength, y=:PetalWidth,color=:Species,Geom.point)
```

Out[6]:



Part c

Considering the goal of performing logistic regression using 'PetalLength' and 'PetalWidth' as independent variables and 'Species' as a dependent variable, write the functional form for the conditional probability f(y|x).

$$f(y|x) = \sigma(\beta_0 + \beta_1 x_1 + \beta_2 x_2) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2)}}$$

Part d

Write equations for conditional likelihood and log conditional likelihood.

Conditional likelihood:

$$L = \prod_i f(y_i|x_i;\theta)$$

$$L = \prod_i \frac{1}{1+e^{-(\beta_0+\beta_1 x_{i1}+\beta_2 x_{i2})}}$$

$$L = \frac{1}{1+e^{-(\beta_0+\beta_1 x_{11}+\beta_2 x_{12})}} \times \frac{1}{1+e^{-(\beta_0+\beta_1 x_{21}+\beta_2 x_{22})}}$$

Log likelihood:

$$l = \sum_{i;y_i=1} log(p_i) + \sum_{i;y_i=0} log(1-p_i)$$

Where $p = \frac{1}{1+e}$ and $1-p = \frac{e}{1+e}$ with $e = exp(1(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}))$

$$l = \sum_{i;y_i=1} log(p_i) + \sum_{i;y_i=0} log(1-p_i)$$

Part e

Write equations for the first gradient with respect to parameters $\beta_0, \beta_1, \beta_2$.

$$\frac{\partial l}{\partial \beta_0} = \sum_i (y_i - p_i)$$

$$\frac{\partial l}{\partial \beta_1} = \sum_i (y_i - p_i) x_{i1}$$

$$\frac{\partial l}{\partial \beta_2} = \sum_i (y_i - p_i) x_{i2}$$

Part f

Write update equations for second derivatives with respect to parameters $\beta_0, \beta_1, \beta_2$.

$$\frac{\partial^2 l}{\partial \beta_0^2} = -\sum_{i=1}^{n} p_i(1-p_i)$$

$$\frac{\partial^2 l}{\partial \beta_1^2} = -\sum_{i=1}^{n} x_{1i}^2 p_i(1-p_i)$$

$$\frac{\partial^2 l}{\partial \beta_2^2} = -\sum_{i=1}^{n} x_{2i}^2 p_i(1-p_i)$$

$$\frac{\partial^2 l}{\partial \beta_0 \beta_1} = -\sum_{i=1}^{n} x_{1i} p_i(1-p_i)$$

$$\frac{\partial^2 l}{\partial \beta_0 \beta_2} = -\sum_{i=1}^{n} x_{2i} p_i(1-p_i)$$

$$\frac{\partial^2 l}{\partial \beta_1 \beta_2} = -\sum_{i=1}^{n} x_{1i} x_{2i} p_i(1-p_i)$$

Part g

Write Newton's algorithm for parameter estimation (with update equations for $\beta = [\beta_0, \beta_1, \beta_2]$.

1. Pick initial value for β.
2. maxIter = 10000
3. for i = 2: maxIter
4. $\widehat{\beta} \leftarrow \beta_{i-1} - \frac{\nabla E(\beta)}{\nabla^2 E(\beta)}$
5. if $|l_i - l_{i-1}| < \epsilon$ terminate; end
6. end for

Part h

Implement Newton's algorithm in Julia and report the estimated parameters $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2$

In [52]:
```
x = convert(Array, new_iris[[:PetalLength, :PetalWidth]]);
x = [ones(size(x,1)) x];
y = Int.(new_iris[:Species].=="virginica");
```

In [47]:

```
function compute_first_derivatives(x,y,p)
    d1 = zeros(3);
    d1[1] = sum(y.-p);
    d1[2] = sum((y.-p).*x[:,2]);
    d1[3] = sum((y.-p).*x[:,3]);
    return d1;
end

function compute_second_derivatives(x,y,p)
    d2 = zeros(3,3);
    d2[1,1] = sum(p.*(1.-p));
    d2[2,2] = sum((x[:,2].^2).*p.*(1.-p));
    d2[1,2] = sum(x[:,2].*p.*(1.-p));
    d2[2,1] = d2[1,2];
    d2[1,3] = sum(x[:,3].*p.*(1.-p));
    d2[2,3] = sum((x[:,2].*x[:,3]).*p.*(1.-p));
    d2[3,1] = d2[1,3];
    d2[3,2] = d2[2,3];
    d2[3,3] = sum((x[:,3].^2).*p.*(1.-p));
    return d2;
end

function compute_p(x,b)
    p = 1./(1.+e.^(-x*b));
    return p;
end

function compute_l(x,y,b)
    p = compute_p(x,b);
    prob = y.*log.(p) + (1-y).*log.(1-p);
    l = sum(prob[.~isnan.(prob)]);
    return l;
end

function newtons_lr(x,y,b)
    max_itr = 20; # maximum num. iterations
    #b = [-4 1]'; # random initialization
    for i=1:max_itr
        l = compute_l(x,y,b);
        print(i,"\n");
        p = compute_p(x,b); print("p: ", p,"\n");
        d1 = compute_first_derivatives(x,y,p); print("d1: ", d1,"\n");
        d2 = compute_second_derivatives(x,y,p); print("d2: ", d2,"\n")
;
        b_new = b.+inv(d2)*d1; print("b_new: ", b_new,"\n");
        l_new = compute_l(x,y,b_new); print("l: ", l,"\n");
        if(abs(l-l_new)<0.0000001) break; end;
        b = b_new;
        print("\n");
    end
    return b;
end
```

Out[47]: newtons_lr (generic function with 1 method)

In [49]:
```
newtons_lr(x, y, [-4; 1; 1])
```

```
1
p: [0.890903, 0.880797, 0.916827, 0.785835, 0.890903, 0.858149, 0.90
8877, 0.574443, 0.869892, 0.785835, 0.622459, 0.845535, 0.731059, 0.
890903, 0.71095, 0.858149, 0.880797, 0.75026, 0.880797, 0.731059, 0.
930862, 0.785835, 0.916827, 0.869892, 0.832018, 0.858149, 0.90025, 0
.937027, 0.880797, 0.622459, 0.71095, 0.668188, 0.75026, 0.937027, 0
.880797, 0.890903, 0.90025, 0.845535, 0.802184, 0.785835, 0.832018,
0.880797, 0.768525, 0.574443, 0.817574, 0.802184, 0.817574, 0.832018
, 0.524979, 0.802184, 0.989013, 0.952574, 0.982014, 0.967705, 0.9820
14, 0.990987, 0.90025, 0.983698, 0.973403, 0.990048, 0.956893, 0.960
834, 0.973403, 0.952574, 0.970688, 0.973403, 0.964429, 0.992608, 0.9
94514, 0.924142, 0.982014, 0.947846, 0.990987, 0.937027, 0.978119, 0
.978119, 0.930862, 0.937027, 0.975873, 0.967705, 0.982014, 0.987872,
0.978119, 0.930862, 0.952574, 0.987872, 0.982014, 0.964429, 0.930862
, 0.970688, 0.982014, 0.967705, 0.952574, 0.985226, 0.985226, 0.9706
88, 0.947846, 0.960834, 0.975873, 0.947846]
d1: [-38.9667, -166.586, -51.5274]
d2: [8.70218 37.4854 12.0153; 37.4854 165.233 53.3226; 12.0153 53.32
26 17.5109]
b_new: [-7.25281, -1.36412, 7.48837]
l: -91.80494266534296

2
p: [0.0399077, 0.103512, 0.0627114, 0.0485953, 0.0915203, 0.0251734,
0.156727, 0.0138429, 0.0220341, 0.110154, 0.0105726, 0.148102, 0.005
37331, 0.0399077, 0.0810053, 0.0588991, 0.103512, 0.00469134, 0.1035
12, 0.0129238, 0.420304, 0.0485953, 0.0627114, 0.00921072, 0.0328104
, 0.0588991, 0.0349969, 0.206987, 0.103512, 0.0105726, 0.0147848, 0.
00806849, 0.0269399, 0.0972259, 0.103512, 0.19624, 0.0807928, 0.0287
468, 0.0426629, 0.0485953, 0.013804, 0.0454751, 0.0235856, 0.0138429
, 0.0374263, 0.0180557, 0.0374263, 0.0328104, 0.0427798, 0.0426629,
0.963855, 0.504519, 0.604548, 0.195789, 0.787463, 0.370416, 0.340487
, 0.0856687, 0.156349, 0.958789, 0.682853, 0.436655, 0.72514, 0.7116
32, 0.977296, 0.93938, 0.218162, 0.520487, 0.635991, 0.0551557, 0.89
9795, 0.738799, 0.19534, 0.387475, 0.667584, 0.123633, 0.420304, 0.3
87475, 0.697134, 0.0397983, 0.206518, 0.267676, 0.829562, 0.0484633,
0.012031, 0.838797, 0.956069, 0.218162, 0.420304, 0.751478, 0.956069
, 0.953177, 0.504519, 0.872375, 0.975699, 0.946698, 0.538545, 0.6526
03, 0.931131, 0.325028]
d1: [19.1804, 110.412, 36.7399]
d2: [10.3046 53.9244 19.0708; 53.9244 287.817 101.561; 19.0708 101.5
61 36.2164]
b_new: [-13.6378, 2.02972, 2.34775]
l: -48.00384346699883

3
p: [0.307621, 0.272409, 0.457467, 0.0782153, 0.314436, 0.189696, 0.4
15393, 0.0100311, 0.222871, 0.0805392, 0.0149787, 0.169194, 0.040265
1, 0.307621, 0.0363079, 0.194633, 0.272409, 0.0488833, 0.272409, 0.0
```

415122, 0.581956, 0.0782153, 0.457467, 0.217412, 0.134945, 0.194633,
0.352449, 0.622924, 0.272409, 0.0149787, 0.0341468, 0.0223115, 0.051
9269, 0.615425, 0.272409, 0.321332, 0.359741, 0.16044, 0.0941594, 0.
0782153, 0.131275, 0.266152, 0.0628777, 0.0100311, 0.112955, 0.09148
16, 0.112955, 0.134945, 0.00692189, 0.0941594, 0.98799, 0.763958, 0.
963309, 0.875946, 0.964416, 0.990885, 0.374518, 0.966927, 0.913768,
0.990174, 0.803652, 0.829268, 0.920995, 0.769644, 0.912805, 0.925501
, 0.852157, 0.994097, 0.996881, 0.508106, 0.965492, 0.731713, 0.9905
93, 0.630365, 0.94593, 0.940837, 0.581956, 0.630365, 0.934558, 0.868
868, 0.960993, 0.982841, 0.947534, 0.55858, 0.734093, 0.984378, 0.96
6536, 0.852157, 0.581956, 0.904906, 0.966536, 0.892218, 0.763958, 0.
976738, 0.97814, 0.910241, 0.725424, 0.833723, 0.938343, 0.719044]
d1: [-2.41215, -7.27374, -0.84193]
d2: [11.644 56.1307 18.9312; 56.1307 273.296 92.431; 18.9312 92.431
31.8088]
b_new: [-20.5344, 2.64483, 4.63842]
l: -20.869838398417382

4
p: [0.166603, 0.157756, 0.350444, 0.0193577, 0.19615, 0.0689647, 0.3
35768, 0.000770248, 0.0880067, 0.0235279, 0.00130655, 0.0780984, 0.0
0488522, 0.166603, 0.00680647, 0.082918, 0.157756, 0.00635486, 0.157
756, 0.00595657, 0.624796, 0.0193577, 0.350444, 0.0732655, 0.0418197
, 0.082918, 0.206622, 0.639935, 0.157756, 0.00130655, 0.00457862, 0.
00221542, 0.00943879, 0.592841, 0.157756, 0.229493, 0.241211, 0.0537
997, 0.0250714, 0.0193577, 0.034522, 0.133036, 0.0122614, 0.00077024
8, 0.0324159, 0.0206334, 0.0324159, 0.0418197, 0.000554087, 0.025071
4, 0.999024, 0.854115, 0.991924, 0.932505, 0.993374, 0.998723, 0.321
402, 0.988763, 0.959097, 0.99925, 0.903007, 0.908563, 0.977087, 0.87
7246, 0.983479, 0.984504, 0.913831, 0.999383, 0.999771, 0.412751, 0.
994565, 0.845812, 0.998442, 0.68448, 0.986371, 0.975488, 0.624796, 0
.68448, 0.982318, 0.902657, 0.988015, 0.996562, 0.988807, 0.477984,
0.683619, 0.998107, 0.995543, 0.913831, 0.624796, 0.970355, 0.995543
, 0.973982, 0.854115, 0.996791, 0.997844, 0.979907, 0.817986, 0.9238
31, 0.988063, 0.786408]
d1: [-0.572573, -0.968459, 0.258022]
d2: [7.48567 36.3569 12.1874; 36.3569 177.673 59.5273; 12.1874 59.52
73 20.2332]
b_new: [-28.9781, 3.6294, 6.84055]
l: -14.316193743741465

5
p: [0.0876129, 0.0843266, 0.282276, 0.00380445, 0.11691, 0.0229085,
0.27388, 3.86657e-5, 0.0326053, 0.00523751, 7.99013e-5, 0.0300675, 0
.000490332, 0.0876129, 0.000893444, 0.0313115, 0.0843266, 0.00070472
6, 0.0843266, 0.000675878, 0.68049, 0.00380445, 0.282276, 0.0238636,
0.011218, 0.0313115, 0.121298, 0.689514, 0.0843266, 7.99013e-5, 0.00
0470256, 0.000165106, 0.00133863, 0.616976, 0.0843266, 0.154347, 0.1
59885, 0.0160477, 0.00545999, 0.00380445, 0.00816203, 0.0626157, 0.0
0192322, 3.86657e-5, 0.00783031, 0.0039663, 0.00783031, 0.011218, 2.
57955e-5, 0.00545999, 0.99995, 0.926143, 0.998888, 0.974902, 0.99919
3, 0.999912, 0.265641, 0.997975, 0.987696, 0.999965, 0.961319, 0.962
844, 0.995268, 0.945319, 0.997399, 0.997505, 0.964312, 0.999969, 0.9

99992, 0.361177, 0.999415, 0.923231, 0.999879, 0.753796, 0.997705, 0
.994008, 0.68049, 0.753796, 0.996704, 0.95335, 0.997889, 0.999641, 0
.998334, 0.448355, 0.715719, 0.999863, 0.999575, 0.964312, 0.68049,
0.993212, 0.999575, 0.994858, 0.926143, 0.999717, 0.999851, 0.996417
, 0.897151, 0.972772, 0.998263, 0.863521]
d1: [-0.143561, 0.00198694, 0.201034]
d2: [5.06854 24.7047 8.25097; 24.7047 120.91 40.2929; 8.25097 40.292
9 13.6084]
b_new: [-37.6268, 4.73739, 8.81854]
l: -11.486430880135309

6
p: [0.046747, 0.0439082, 0.234011, 0.000736286, 0.0686883, 0.0078103
7, 0.222454, 1.89766e-6, 0.0124843, 0.00110695, 4.89441e-6, 0.010965
6, 5.22866e-5, 0.046747, 0.000110751, 0.0117006, 0.0439082, 8.3969e-
5, 0.0439082, 7.8636e-5, 0.728302, 0.000736286, 0.234011, 0.0083357,
0.00304277, 0.0117006, 0.073007, 0.741091, 0.0439082, 4.89441e-6, 4.
89657e-5, 1.26235e-5, 0.000189914, 0.655552, 0.0439082, 0.0998494, 0
.105904, 0.00487766, 0.00118194, 0.000736286, 0.00202521, 0.0296306,
0.000304965, 1.89766e-6, 0.00189682, 0.000786185, 0.00189682, 0.0030
4277, 1.10656e-6, 0.00118194, 0.999997, 0.964053, 0.999856, 0.99164,
0.999904, 0.999995, 0.21131, 0.999694, 0.996742, 0.999998, 0.984797,
0.985749, 0.99904, 0.975807, 0.999547, 0.999575, 0.986642, 0.999999,
1.0, 0.329144, 0.999936, 0.961708, 0.999992, 0.811497, 0.999628, 0.9
98734, 0.728302, 0.811497, 0.999402, 0.981288, 0.999673, 0.999967, 0
.999752, 0.4407, 0.777044, 0.99999, 0.999958, 0.986642, 0.728302, 0.
998459, 0.999958, 0.998906, 0.964053, 0.999975, 0.999989, 0.999318,
0.9435, 0.990479, 0.999736, 0.917378]
d1: [-0.0519313, -0.0306758, 0.0472148]
d2: [3.74888 18.3258 6.10257; 18.3258 89.8655 29.8355; 6.10257 29.83
55 10.0367]
b_new: [-43.4561, 5.50974, 10.0717]
l: -10.482921679059563

7
p: [0.0304729, 0.027794, 0.205731, 0.000242575, 0.0472557, 0.0037994
6, 0.190675, 2.4988e-7, 0.00657349, 0.000382745, 7.52145e-7, 0.00544
462, 1.18228e-5, 0.0304729, 2.67794e-5, 0.00598265, 0.027794, 2.0511
7e-5, 0.027794, 1.8657e-5, 0.75393, 0.000242575, 0.205731, 0.0041756
1, 0.00126548, 0.00598265, 0.0517106, 0.771087, 0.027794, 7.52145e-7
, 1.07538e-5, 2.26397e-6, 5.10782e-5, 0.680977, 0.027794, 0.0725894,
0.079234, 0.0021935, 0.000420779, 0.000242575, 0.000802294, 0.017793
9, 8.86144e-5, 2.4988e-7, 0.0007298, 0.000266684, 0.0007298, 0.00126
548, 1.31004e-7, 0.000420779, 1.0, 0.977682, 0.999963, 0.99604, 0.99
9976, 0.999999, 0.176476, 0.999916, 0.998681, 1.0, 0.991731, 0.99247
3, 0.999664, 0.985741, 0.999852, 0.999865, 0.993149, 1.0, 1.0, 0.310
051, 0.999985, 0.975517, 0.999999, 0.841664, 0.999888, 0.999561, 0.7
5393, 0.841664, 0.999806, 0.990196, 0.999908, 0.999994, 0.999929, 0.
438093, 0.817394, 0.999998, 0.999991, 0.993149, 0.75393, 0.999417, 0
.999991, 0.999594, 0.977682, 0.999995, 0.999998, 0.999766, 0.961904,
0.995217, 0.999922, 0.941177]
d1: [-0.018544, -0.0475437, -0.0057968]
d2: [3.1655 15.5049 5.15512; 15.5049 76.1536 25.2366; 5.15512 25.236

6 8.47324]
b_new: [-45.1641, 5.73972, 10.4252]
l: -10.2914845957235

8
p: [0.0268006, 0.0241838, 0.197546, 0.000174662, 0.0421434, 0.003071
16, 0.181364, 1.37748e-7, 0.00543929, 0.000279023, 4.34142e-7, 0.004
4098, 7.65584e-6, 0.0268006, 1.75863e-5, 0.0048977, 0.0241838, 1.359
14e-5, 0.0241838, 1.22314e-5, 0.759853, 0.000174662, 0.197546, 0.003
41146, 0.000976491, 0.0048977, 0.0466107, 0.778561, 0.0241838, 4.341
42e-7, 6.88981e-6, 1.36829e-6, 3.46918e-5, 0.687564, 0.0241838, 0.06
56769, 0.0724501, 0.00173226, 0.000310036, 0.000174662, 0.000611421,
0.0152751, 6.15869e-5, 1.37748e-7, 0.000550276, 0.000194078, 0.00055
0276, 0.000976491, 6.98277e-8, 0.000310036, 1.0, 0.980474, 0.999975,
0.996807, 0.999984, 1.0, 0.166234, 0.999942, 0.998985, 1.0, 0.993028
, 0.993721, 0.999751, 0.987689, 0.999892, 0.999902, 0.994346, 1.0, 1
.0, 0.304125, 0.99999, 0.97835, 0.999999, 0.84888, 0.999921, 0.99967
8, 0.759853, 0.84888, 0.99986, 0.99189, 0.999936, 0.999996, 0.999951
, 0.436898, 0.828294, 0.999999, 0.999994, 0.994346, 0.759853, 0.9995
58, 0.999994, 0.999692, 0.980474, 0.999997, 0.999999, 0.999827, 0.96
5853, 0.996061, 0.999945, 0.946535]
d1: [-0.00176425, -0.00615655, -0.00156081]
d2: [3.02496 14.8256 4.92782; 14.8256 72.8544 24.1349; 4.92782 24.13
49 8.10035]
b_new: [-45.272, 5.75448, 10.4466]
l: -10.281787113149319

9
p: [0.0265807, 0.0239659, 0.197012, 0.000171042, 0.0418298, 0.003029
91, 0.180745, 1.3265e-7, 0.00537428, 0.000273424, 4.1931e-7, 0.00435
005, 7.44906e-6, 0.0265807, 1.71205e-5, 0.00483526, 0.0239659, 1.324
38e-5, 0.0239659, 1.1909e-5, 0.760144, 0.000171042, 0.197012, 0.0033
6836, 0.000960511, 0.00483526, 0.0463011, 0.778974, 0.0239659, 4.193
1e-7, 6.69831e-6, 1.32545e-6, 3.385e-5, 0.687935, 0.0239659, 0.06524
09, 0.0720265, 0.00170644, 0.00030406, 0.000171042, 0.000601009, 0.0
151263, 6.01812e-5, 1.3265e-7, 0.000540469, 0.000190209, 0.000540469
, 0.000960511, 6.709e-8, 0.00030406, 1.0, 0.98063, 0.999976, 0.99685
, 0.999985, 1.0, 0.165545, 0.999944, 0.999001, 1.0, 0.993099, 0.9937
9, 0.999755, 0.987796, 0.999894, 0.999904, 0.994412, 1.0, 1.0, 0.303
724, 0.99999, 0.978506, 0.999999, 0.849274, 0.999923, 0.999684, 0.76
0144, 0.849274, 0.999862, 0.991987, 0.999937, 0.999996, 0.999952, 0.
436796, 0.828978, 0.999999, 0.999994, 0.994412, 0.760144, 0.999565,
0.999994, 0.999697, 0.98063, 0.999997, 0.999999, 0.99983, 0.966073,
0.996107, 0.999946, 0.946839]
d1: [-8.74454e-6, -3.36475e-5, -9.72268e-6]
d2: [3.01657 14.785 4.91431; 14.785 72.6577 24.0696; 4.91431 24.0696
8.07832]
b_new: [-45.2723, 5.75453, 10.4467]
l: -10.281754052130985

```
Out[49]: 3-element Array{Float64,1}:
          -45.272
            5.75448
           10.4466
```

$$\widehat{\beta_0} = -45.3$$
$$\widehat{\beta_1} = 5.75$$
$$\widehat{\beta_2} = 10.45$$

In [ ]:
```
#### Ex_9_LR_Qts_2 ####
```

In [1]:
```
using RDatasets;
```

In [46]:
```
data = dropmissing(dataset("MASS","cats"));
x = data[:,2];
x = [ones(length(x)) x];
y = Int.(data[:,1].=="M");
```

In [47]:
```
print(size(data));
```

```
(144, 3)
```

In [48]:
```
function compute_p(x,b)
p = 1./(1.+e.^(-x*b));
return p;
end
```

Out[48]:
```
compute_p (generic function with 1 method)
```

In [49]:
```
function compute_l(x,y,b)
p = compute_p(x,b);
prob = y.*log.(p) + (1-y).*log.(1-p);
l = sum(prob[.~isnan.(prob)]);
return l;
end
```

Out[49]:
```
compute_l (generic function with 1 method)
```

In [50]:
```
b0 = collect(-40:40); #select the points along b0
b1 = collect(-40:40); #select the points along b1
heatm = zeros(81,81);
for i=1:length(b0)
for j=1:length(b1)
heatm[i,j]= compute_l(x,y,[b0[i] b1[j]]');
# computing l at each b0, b1 combination
end
end
```

In [51]:
```
function newtons_lr(x,y,b)
max_itr = 20; # maximum num. iterations
#b = [-4 1]'; # random initialization
for i=1:max_itr
l = compute_l(x,y,b);
#print(i,"\n");
p = compute_p(x,b); #print("p: ", p,"\n");
d1 = compute_first_derivatives(x,y,p); #print("d1: ", d1,"\n");
d2 = compute_second_derivatives(x,y,p); #print("d2: ", d2,"\n");
b_new = b.+inv(d2)*d1; #print("b_new: ", b_new,"\n");
l_new = compute_l(x,y,b_new); #print("l: ", l,"\n");
if(abs(l-l_new)<0.0000001) break; end;
b = b_new;
#print("\n");
end
return b;
end
```

Out[51]:
```
newtons_lr (generic function with 1 method)
```

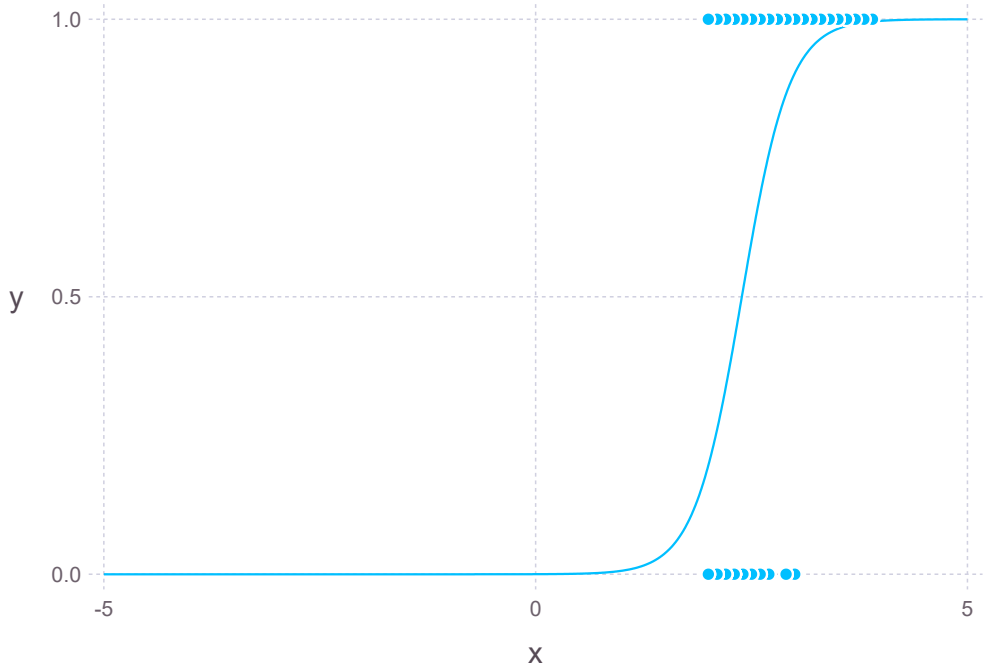
In [52]:
```
Pkg.add("Plots")
using Plots;
Plots.surface(b0,b1,heatm)
```

```
) at /home/jrun/.julia/v0.6/GR/src/gr3.jl:379
 [3] gr_display(::Plots.Subplot{Plots.GRBackend}, ::Measures.Length{:mm,Float64}, ::Measures.Length{:mm,Float64}, ::Array{Float64,1}) at /home/jrun/.julia/v0.6/Plots/src/backends/gr.jl:1097
 [4] gr_display(::Plots.Plot{Plots.GRBackend}, ::String) at /home/jrun/.julia/v0.6/Plots/src/backends/gr.jl:595
 [5] _show(::Base.AbstractIOBuffer{Array{UInt8,1}}, ::MIME{Symbol("image/svg+xml")}, ::Plots.Plot{Plots.GRBackend}) at /home/jrun/.julia/v0.6/Plots/src/backends/gr.jl:1385
 [6] show(::Base.AbstractIOBuffer{Array{UInt8,1}}, ::MIME{Symbol("image/svg+xml")}, ::Plots.Plot{Plots.GRBackend}) at /home/jrun/.julia/v0.6/Plots/src/output.jl:210
 [7] #sprint#228(::Void, ::Function, ::Int64, ::Function, ::MIME{Symbol("image/svg+xml")}, ::Vararg{Any,N} where N) at ./strings/io.jl:66
 [8] display_dict(::Plots.Plot{Plots.GRBackend}) at /home/jrun/.julia/v0.6/Plots/src/output.jl:296
 [9] (::Compat.#inner#14{Array{Any,1},IJulia.#display_dict,Tuple{Plots.Plot{Plots.GRBackend}}})() at /mnt/juliabox/.julia/v0.6/Compat/src/Compat.jl:332
 [10] execute_request(::ZMQ.Socket, ::IJulia.Msg) at /home/jrun/.julia/v0.6/IJulia/src/execute_request.jl:209
```

In [53]:
```
using Gadfly;
```

In [54]:
```
# estimate b using newton method
b = newtons_lr(x,y,[-4 1]')
# compute the sigmoid function
x1 = collect(-5:0.01:5);
p_fit = 1./(1.+e.^(-(b[1].+b[2].*x1)));
# plot the points and overlay the learned sigmoid function
Gadfly.plot(layer(x=x[:,2],y=y,Geom.point),layer(x=x1,y=p_fit, Geom.line))
```

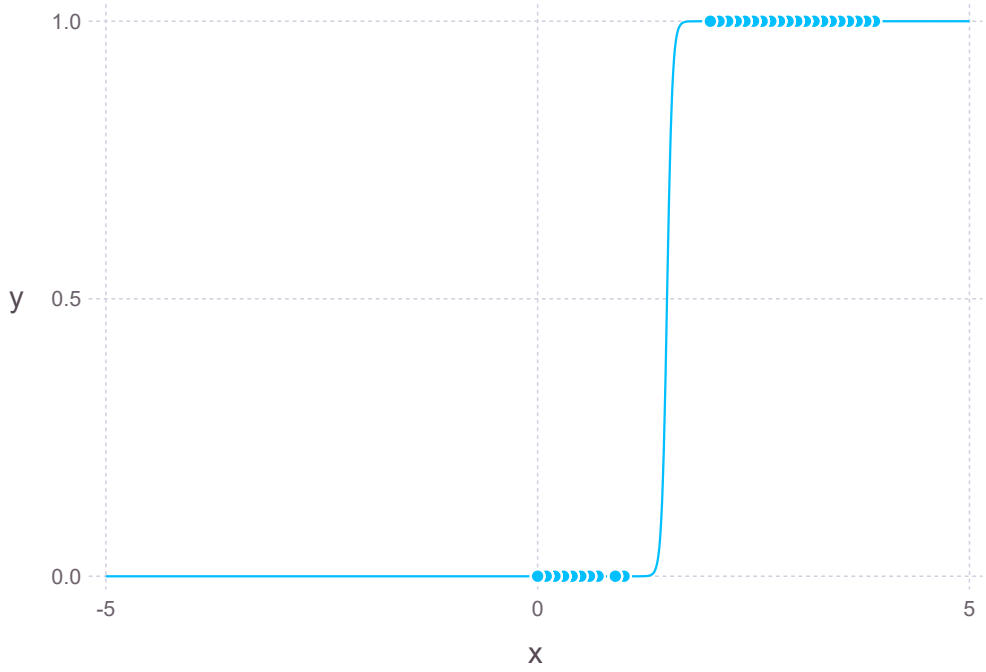
Out[54]:



In [55]:
```
x[y.==0,2] = x[y.==0,2]-2;
```

In [56]:
```
# estimate b using newton method
b = newtons_lr(x,y,[-4 1]')
# compute the sigmoid function
x1 = collect(-5:0.01:5);
p_fit = 1./(1.+e.^(-(b[1].+b[2].*x1)));
# plot the points and overlay the learned sigmoid function
Gadfly.plot(layer(x=x[:,2],y=y,Geom.point),layer(x=x1,y=p_fit, Geom.line))
```

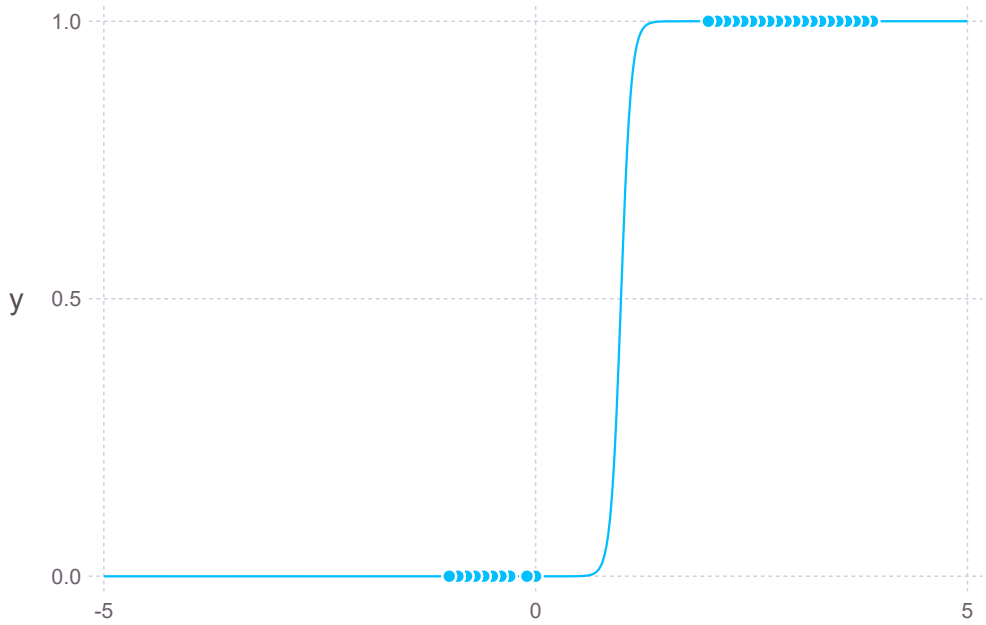
Out[56]:



In [57]:
```
x[y.==0,2] = x[y.==0,2]-1;
```

In [58]:
```
# estimate b using newton method
b = newtons_lr(x,y,[-4 1]')
# compute the sigmoid function
x1 = collect(-5:0.01:5);
p_fit = 1./(1.+e.^(-(b[1].+b[2].*x1)));
# plot the points and overlay the learned sigmoid function
Gadfly.plot(layer(x=x[:,2],y=y,Geom.point),layer(x=x1,y=p_fit, Geom.line))
```

Out[58]:



In [59]:
```
#The sigmoid function seems to fit the data well in part (d) and (e)
# as compared to part(c).
#The sigmoid function is unable to fit the data in part (c) because the
# data shows different values of 'y' for same values of 'x'
```