Exercise 2

a) Generate a simulated data set as follows:

In [207]:
```
srand(1)
x = randn(100)
y = x - 2x.^2 + randn(100)
;
```

In this data set, what is n and what is p? Write out the model used to generate the data in equation form

In this regression model $n = 100$ which is the length of the variables x and y, and $p = 2$, since y is generated with two predictos x and x^2, plus a random error.

(c) Do 5-fold cross-validation to fit polynomial models using least squares, we compute the generalization error

Step 1: Create the folds

In [4]:
```
n = 100
k = 5
s = convert(Integer, n / k)
seq = randperm(n)
fold = [seq[((i-1)*s + 1):(i*s)] for i in 1:k]
```

Out[4]:
```
5-element Array{Array{Int64,1},1}:
 [45, 92, 34, 86, 76, 30, 94, 88, 97, 79, 66, 100, 21, 80, 4, 72, 59, 42, 69, 3
1]
 [14, 85, 5, 19, 74, 6, 48, 89, 60, 75, 87, 83, 81, 15, 71, 23, 27, 40, 41, 29]

 [82, 49, 22, 44, 17, 9, 73, 13, 28, 56, 11, 1, 7, 36, 20, 33, 47, 35, 68, 63]

 [84, 95, 99, 98, 77, 61, 24, 38, 53, 16, 62, 52, 43, 3, 54, 91, 25, 39, 26, 3
2]
 [65, 93, 2, 70, 67, 46, 64, 90, 57, 12, 78, 10, 37, 18, 51, 55, 96, 8, 50, 58]
```

Step 2: fit models with cross-validation

In [208]:
```
# design matrix with polynomial entries
X = [ones(length(y)) x x.^2 x.^3 x.^4]
# store espace for generalization error
gen_error = Array{Float64}(4, k)
βhat_all = [[] for i in 1:k]
# k-fold cross validation
for i in 1:k # outer loop is the cross-validation loop
    # test and train indices
    train_idx = fold[i]
    test_idx = vcat([fold[j] for j in 1:k if j != 3]...)
    # y data
    y_train = y[train_idx]
    y_test = y[test_idx]
    # fit model
    for deg in 1:4 # inner loop fits different polynomial models
        # X data
        X_train = X[train_idx, 1:(1 + deg)]
        X_test = X[test_idx, 1:(1 + deg)]
        # regression
        βhat_train = Symmetric(X_train'*X_train) \ X_train'*y_train
        push!(βhat_all[i], βhat_train)
        yhat_test = X_test*βhat_train
        mse = mean((yhat_test - y_test).^2)
        gen_error[deg, i] = mse # average cross validation error
    end
end
```

We now print the mean squared error (MSE) per degree of fitted polynomial. We see that on average the second degree polynomial works better (as expected from the true parameters), although for **some** fold test sets, higher degree polynomials may work better.

In [49]:
```
mse = mapslices(mean, gen_error, 2)
for deg in 1:4
    @printf("Degree: %i \t MSE: %0.2f \t SE: %s \n", deg, mse[i], round.(gen_erro
end
```

```
Degree: 1        MSE: 8.77       SE: [8.52, 6.96, 13.45, 6.58, 8.36]
Degree: 2        MSE: 1.12       SE: [1.06, 1.31, 1.05, 1.09, 1.08]
Degree: 3        MSE: 2.53       SE: [1.06, 1.35, 8.17, 1.09, 1.0]
Degree: 4        MSE: 2.17       SE: [1.47, 1.8, 5.54, 1.04, 0.99]
```

We now recover the averaged parameters for each degree across all folds.

In [126]:
```
βpooled = [Array{Float64}(deg + 1) for deg in 1:4]
for deg in 1:4
    for i in 1:k
        βpooled[deg] += βhat_all[i][deg] / k
    end
end
for deg in 1:4
   @printf("Degree: %i %25.25s = %35.35s \n", deg, join(["β$i" for i in 0:deg], "
end
```

```
Degree: 1                    β0, β1 =                        [-1.94, 1.12]
Degree: 2                β0, β1, β2 =                  [-0.21, 1.1, -1.85]
Degree: 3            β0, β1, β2, β3 =           [-0.12, 0.82, -2.08, 0.21]
Degree: 4        β0, β1, β2, β3, β4 =   [-0.15, 0.93, -2.06, -0.02, 0.07]
```

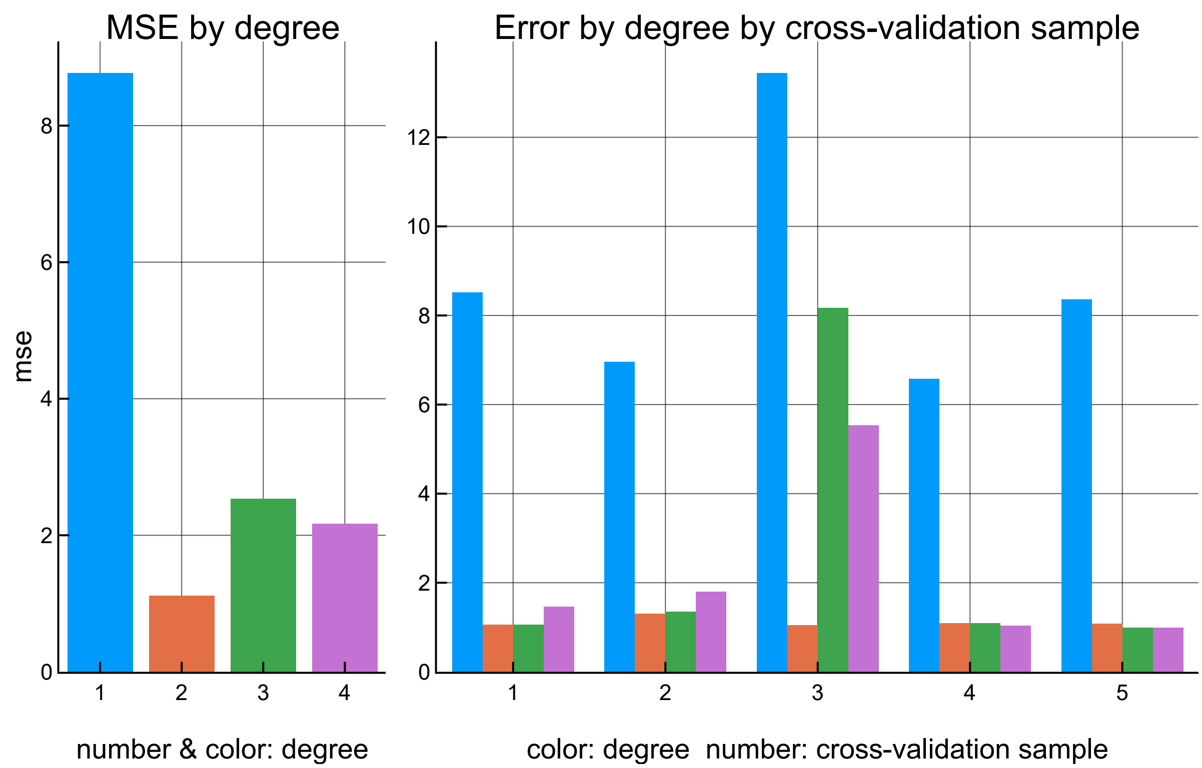
We see above that for degree 2, the pooled coefficients $\hat{\beta}$ do a nice job recovering the true values

$$(\beta_0, \beta_1, \beta_2) = (0, 1, -2) \approx (-0.21, 1.1, -1.85) = (\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2)$$

In [127]:
```
using Plots
```

In [174]:
```
plot(
    plot(mapslices(mean, gen_error, 2),
        seriestype = :bar, ylab = "mse", xlab = "number & color: degree",
        title = "MSE by degree",
        titlefont = font(12),
        guidefont = font(10),
        group = 1:4),
    plot(transpose(gen_error),
        st = :bar, legend = true, xlab = "color: degree  number: cross-validation
        title = "Error by degree by cross-validation sample",
        titlefont = font(12),
        guidefont = font(10)),
    leg = false,
    layout = @layout [a{0.3w} b{0.7w}]
)
```

Out[174]:

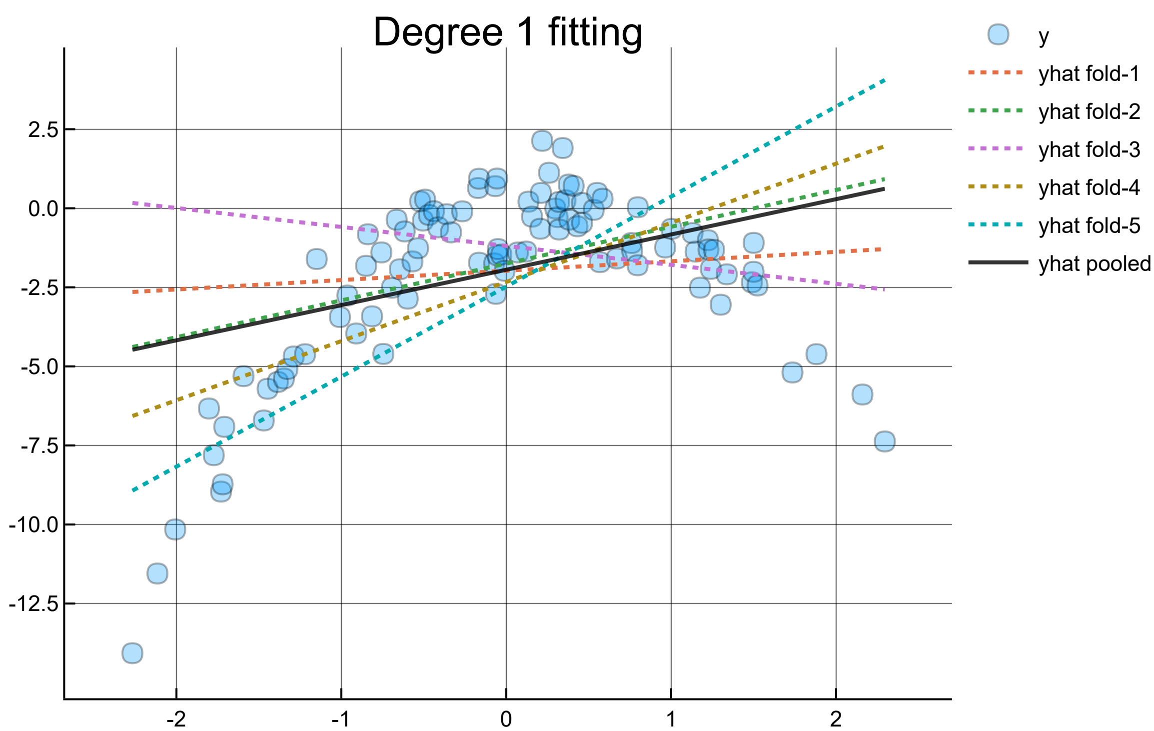


In [225]:
```
order = sortperm(x)
xo = x[order]
Xo = X[order,:]
;
```

In [270]:

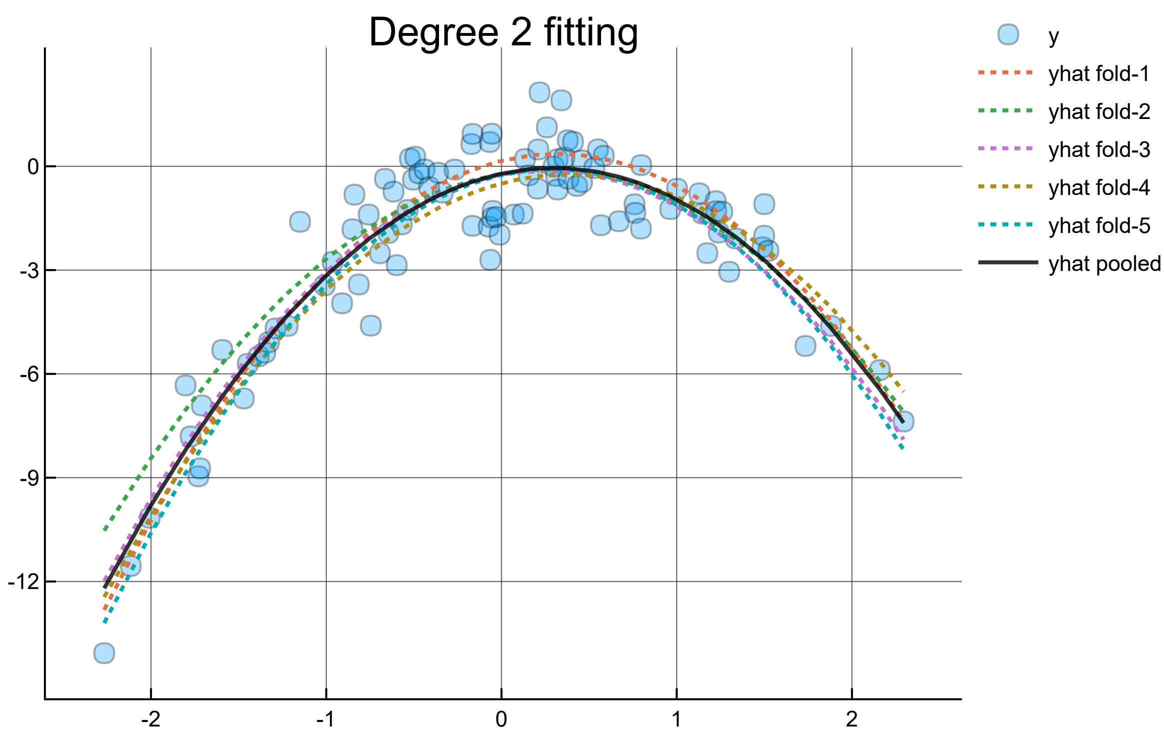
```
deg = 1
Xdeg = Xo[:,1:(deg + 1)]
yhat_folds = hcat([Xdeg*βhat_all[i][deg] for i in 1:k]...)
yhat_pooled = Xdeg*βpooled[deg]
plot(xo, yo, st = :scatter, alpha = 0.3, ms = 5, label = "y", title = "Degree 1 f
plot!(xo, yhat_folds, st = :line, lw = 2, label = hcat(["yhat fold-$i" for i in 1
plot!(xo, yhat_pooled, st = :line, lw = 2, alpha = 0.8, label = "yhat pooled", co
```

Out[270]:



In [267]:
```
deg = 2
Xdeg = Xo[:,1:(deg + 1)]
yhat_folds = hcat([Xdeg*βhat_all[i][deg] for i in 1:k]...)
yhat_pooled = Xdeg*βpooled[deg]
plot(xo, yo, st = :scatter, alpha = 0.3, ms = 5, label = "y", title = "Degree 2 f
plot!(xo, yhat_folds, st = :line, lw = 2, label = hcat(["yhat fold-$i" for i in 1
plot!(xo, yhat_pooled, st = :line, lw = 2, alpha = 0.8, label = "yhat pooled", co
```

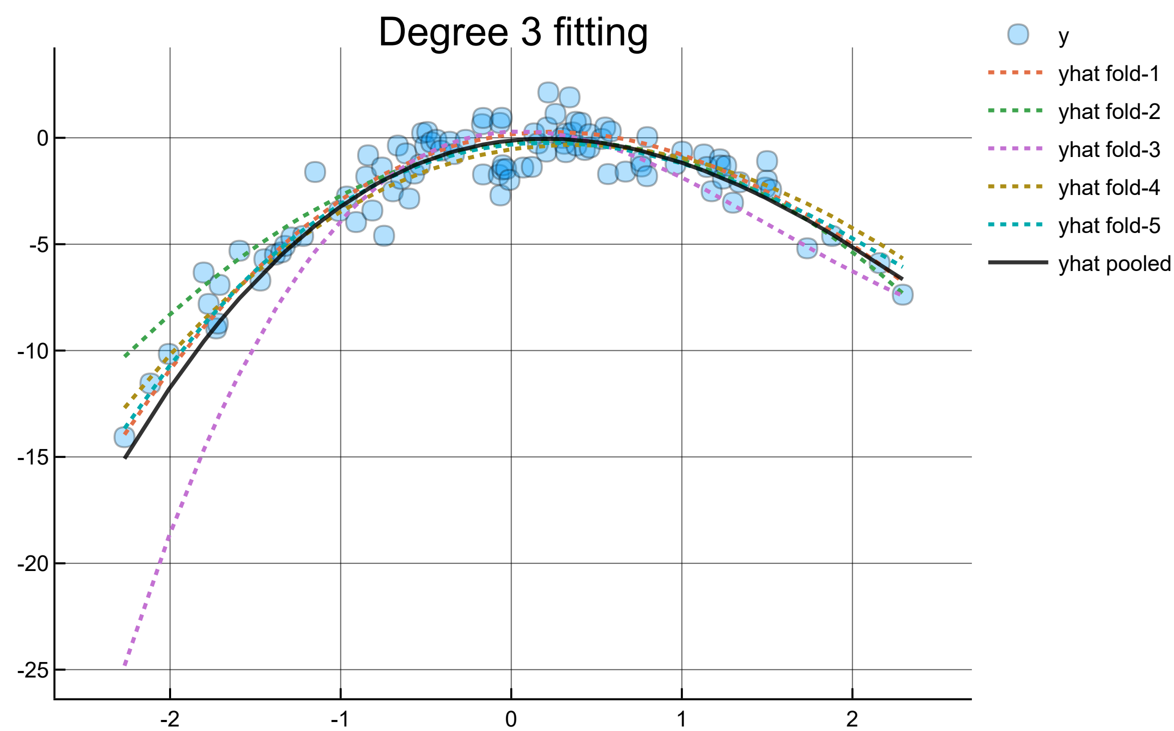
Out[267]:



In [271]:

```
deg = 3
Xdeg = Xo[:,1:(deg + 1)]
yhat_folds = hcat([Xdeg*βhat_all[i][deg] for i in 1:k]...)
yhat_pooled = Xdeg*βpooled[deg]
plot(xo, yo, st = :scatter, alpha = 0.3, ms = 5, label = "y", title = "Degree 3 f
plot!(xo, yhat_folds, st = :line, lw = 2, label = hcat(["yhat fold-$i" for i in 1
plot!(xo, yhat_pooled, st = :line, lw = 2, alpha = 0.8, label = "yhat pooled", co
```

Out[271]:



In [269]:

```
deg = 4
Xdeg = Xo[:,1:(deg + 1)]
yhat_folds = hcat([Xdeg*βhat_all[i][deg] for i in 1:k]...)
yhat_pooled = Xdeg*βpooled[deg]
plot(xo, yo, st = :scatter, alpha = 0.3, ms = 5, label = "y", title = "Degree 4 f
plot!(xo, yhat_folds, st = :line, lw = 2, label = hcat(["yhat fold-$i" for i in 1
plot!(xo, yhat_pooled, st = :line, lw = 2, alpha = 0.8, label = "yhat pooled", co
```

Out[269]: