Problem 3

Chapter 5, Exercise 8 (Sec. 5.4, p. 200)

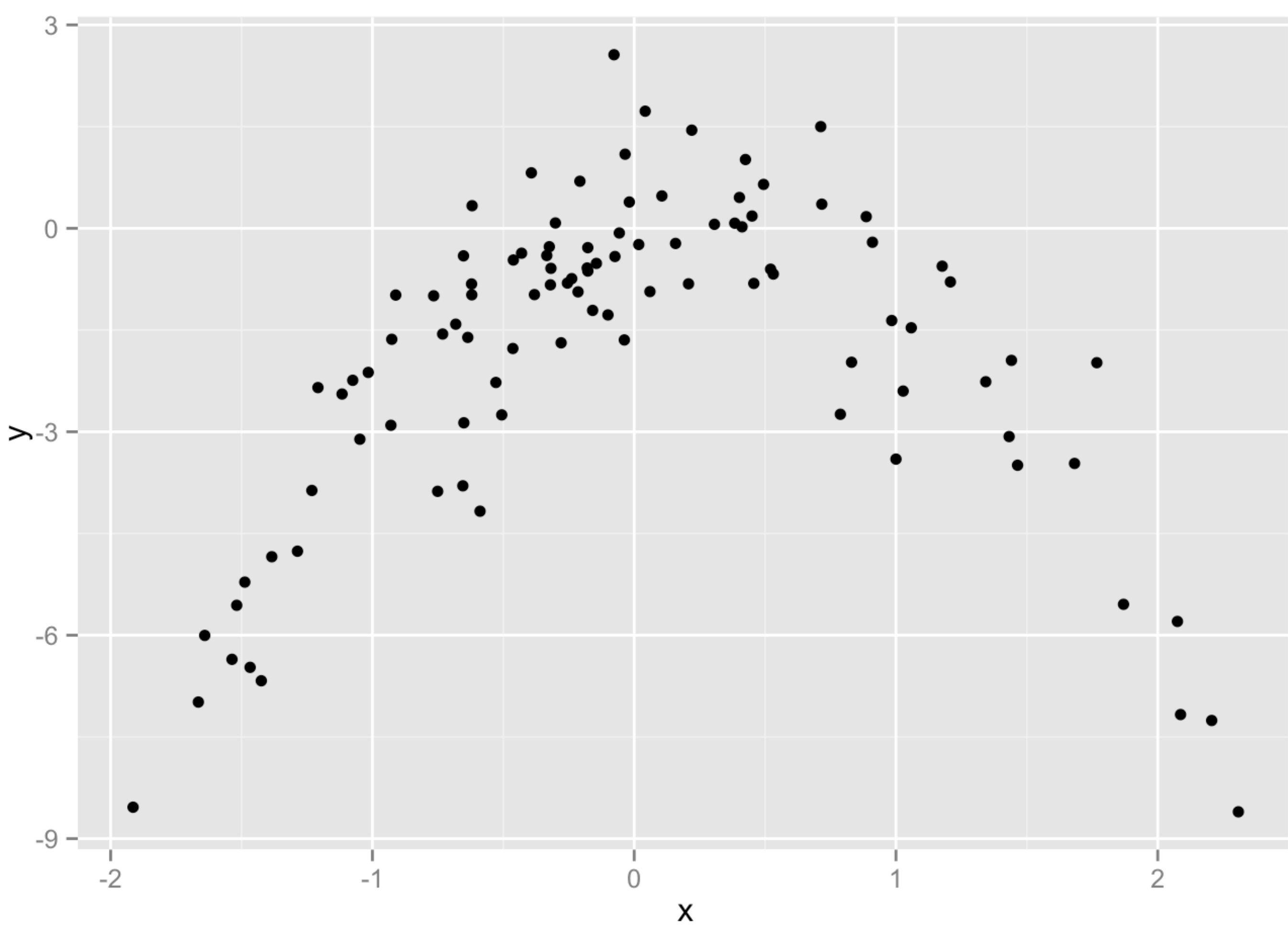
Part A

```
set.seed(1)
y = rnorm(100)
x = rnorm(100)
y = x - 2*x^2 + rnorm(100)
```

```
n = 100  # number of samples
p = 2    # number of dimensions
```

Part B

```
qplot(x, y)
```



There is clearly a non-random relationship between `x` and `y`. In particular, `y` takes on a parabolic shape centered around 0 as `x` varies. However, there is a range of variance of a bit over 1, creating a band of values rather than a neat line of points.

Part C

```
data   = data.frame(y = y, x = x)
```

Linear:

```
model = glm(y ~ x, data = data)
model$coef
```

```
## (Intercept)           x
##  -1.8185184   0.2430443
```

```
## standard k-fold CV estimate = 5.890979
##      bias-corrected version = 5.888812
```

Squared:

```
model = glm(y ~ poly(x, degree = 2), data = data)
model$coef
```

```
##          (Intercept) poly(x, degree = 2)1 poly(x, degree = 2)2
##            -1.827707             2.316401           -21.058587
```

```
## standard k-fold CV estimate = 1.086596
##      bias-corrected version = 1.086326
```

Cubic:

```
model = glm(y ~ poly(x, degree = 3), data = data)
model$coef
```

```
##          (Intercept) poly(x, degree = 3)1 poly(x, degree = 3)2
##           -1.8277074            2.3164010          -21.0585869
## poly(x, degree = 3)3
##           -0.3048398
```

```
## standard k-fold CV estimate = 1.102585
##      bias-corrected version = 1.102227
```

Quadratic:

```
model = glm(y ~ poly(x, degree = 4), data = data)
model$coef
```

```
##          (Intercept) poly(x, degree = 4)1 poly(x, degree = 4)2
##           -1.8277074            2.3164010          -21.0585869
## poly(x, degree = 4)3 poly(x, degree = 4)4
##           -0.3048398           -0.4926249
```

```
## standard k-fold CV estimate = 1.114772
##      bias-corrected version = 1.114334
```

Part D

```
set.seed(5)
y = rnorm(100)
x = rnorm(100)
y = x - 2*x^2 + rnorm(100)

data   = data.frame(y = y, x = x)
```

Linear:

```
model = glm(y ~ x, data = data)
```

```
## standard k-fold CV estimate = 10.32995
##      bias-corrected version = 10.32529
```

Squared:

```
model = glm(y ~ poly(x, degree = 2), data = data)
```

```
## standard k-fold CV estimate = 0.9586209
##      bias-corrected version = 0.9583222
```

Cubic:

```
model = glm(y ~ poly(x, degree = 3), data = data)
```

```
## standard k-fold CV estimate = 0.9867481
##      bias-corrected version = 0.9862594
```

Quadratic:

```
model = glm(y ~ poly(x, degree = 4), data = data)
```

```
## standard k-fold CV estimate = 1.335795
##      bias-corrected version = 1.332331
```

Both runs use the same data generator function, but the `rnorm(...)` creates variance between runs. This variance results in slightly different fits resulting from a linear regression model.

Part E

The squared model had the smallest error. This is what I expected, since the original function is based off the square of `x`.

Part F

Our original function was `y = x - 2*x^2 + rnorm(100)`, which means the correct coefficients ought to be: `B0 = 0, B1 = 1, B2 = -2`.

Instead, we got `[-1.82, 0.24]`, `[-1.83, 2.32, -21.06]`, `[-1.83, 2.31, -21.06, -0.35]`, and `[-1.83, 2.32, -21.06, -0.31, -0.49]` for the linear, squared, cubed, and quadratic fits respectively. These are wayyyy off, even for the best fit (squared, with coefficients `[-1.83, 2.32, -21.06]`). This does not agree with the conclusions drawn based on the cross-validation results, which implied that the squared fit was fairly accurate.