Statistical Computing 1, Stat 590 | Homework 6 | Name: Answer Key
Fall 2015 | Prof. Erik B. Erhardt

**Part I.** (50 points) Do all calculations in LaTeX + R + knitr. Insert computer text output and graphics to support what you are saying. For this assignment, all R code should well commented and be visible (`echo=TRUE`) in the document where you have written it.

I do not want you to use any built-in polynomial functions to solve these problems, you must write your own functions.

Rubric point values are (pts) specified in parenthese before each item to prepare.

(10pts) **1. Cubic function:** Create a function called cubicfn that takes input $(x, a, b, c, d)$ where $x$ is a vector and $a, b, c, d$ are scalars (constants), and returns the cubic polynomial

$$y \quad = \quad a + bx + cx^2 + dx^3$$

where the polynomial is evaluated for each element in $x$.

*Solution*:

```
# cubicfn calculates a cubic fuction of x with coefficients a, b, c, d
#   Default coefficients are 0s.
cubicfn <- function(x=0, a=0, b=0, c=0, d=0) {
  y <- a + b*x + c*x^2 + d*x^3
  return(y)
}
```

**Rubric:** (pts)

**(2)** Function name and arguments correctly specified.

**(2)** Operations within function are correct.

**(4)** Commented lines describing why each command is included.

**(2)** `return()` used to return value `y`.

(20pts) **2. Using our cubic function, plotting:** Create a vector $x$ with 101 elements equally spaced between $-2$ and 3, and compute

```
y1 <- cubicfn(x, 0, 1, 1, 0)
y2 <- cubicfn(x, 1, -3, -1, 3)
```

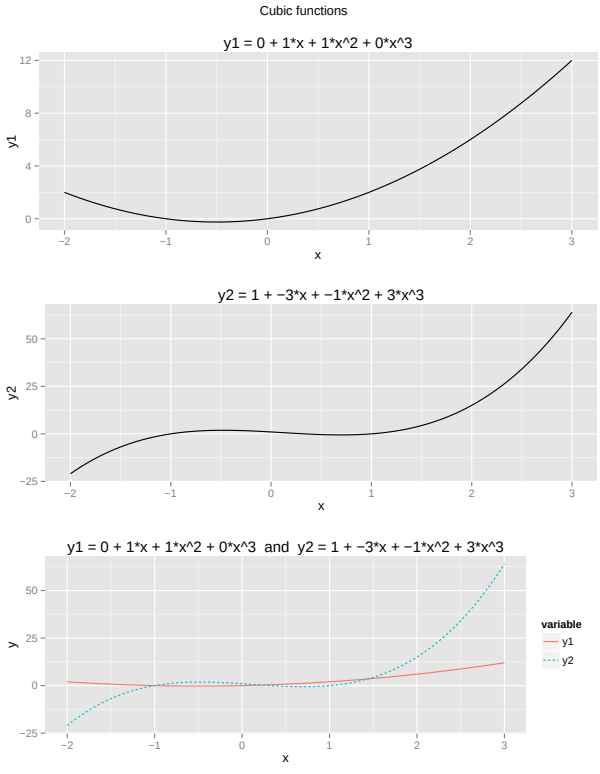
For the plots below, each individually is not very difficult; look at help for plotting functions for axis and title labels. A more challenging task is to figure out how to make multiple plots on the same set of axes. In your writeup, describe how you created the plots.

(a) (5 pts) (2) Make a plot of y1 against x. (1) Label the axes, and (2) provide a title for the plot. Remark: an appropriate title would probably specify the actual function being plotted, ideally specified from the arguments to the function.

*Solution*:

```
# create a sequence of x values
x <- seq(-2, 3, length=101)

y1 <- cubicfn(x, 0, 1, 1, 0)
y2 <- cubicfn(x, 1, -3, -1, 3)

# create a data frame to hold all three vectors
dfcubic <- data.frame(x, y1, y2)

# There are several fine answers to this problem.
# Below I use ggplot2 with geom_line().
library(ggplot2)
```

```
# (a)
p1 <- ggplot(dfcubic, aes(x = x, y = y1))
p1 <- p1 + geom_line()
p1 <- p1 + labs(title = "y1 = 0 + 1*x + 1*x^2 + 0*x^3")

# (b)
p2 <- ggplot(dfcubic, aes(x = x, y = y2))
p2 <- p2 + geom_line()
p2 <- p2 + labs(title = "y2 = 1 + -3*x + -1*x^2 + 3*x^3")

# (c)
# reshape data in long format
library(reshape2)
dfcubic2 <- melt(dfcubic, id.vars="x")

p3 <- ggplot(dfcubic2, aes(x = x, y = value, colour = variable))
p3 <- p3 + geom_line(aes(linetype = variable))
p3 <- p3 + labs(title = "y1 = 0 + 1*x + 1*x^2 + 0*x^3  and  y2 = 1 + -3*x + -1*x^2 + 3*x^3")
p3 <- p3 + ylab("y")

# plot all three together
library(gridExtra)
grid.arrange(p1, p2, p3, ncol = 1, top = "Cubic functions")
```




(b) (5 pts) Repeat part (a) for y2 vs x.

*Solution*: (see above)

(c) (10 pts) On the same set of axes, (10) plot y1 against x and y2 against x. As above, (2) label the axes and (2) provide a title for the plot. In addition, I would like you to (6) label the curves either with a legend or within the plot.

*Solution*: (see above)

(20$^{\text{pts}}$) **3. Polynomial function:**

Create a function called polyfn that takes input $(x, a)$ where $a$ is a vector of arbitrary length and $x$ is a vector, and returns the polynomial

$$y \quad = \quad a[1] + a[2]x + a[3]x^2 + \cdots + a[k]x^{k-1}$$

where the polynomial is evaluated for each element in $x$. Here $k$ is the the number of elements in $a$, or the length of the input vector $a$.

**Rubric:** (pts)

**(2)** Function name and arguments correctly specified.

**(2)** Operations within function are correct.

**(4)** Commented lines describing why each command is included.

**(2)** `return()` used to return value `y`.

Choose a (1) single range for $x$, then $(3 \times 3)$ demonstrate the use of the polynomial with three choices of $a$.

Helpful hint: There are at least two important steps you need to have in your function. First, you need to figure out the length of the input $a$. Then after initializing $y$ to 0, you need to loop over successive powers of $x$, and add the contribution of the given power of $x$ to $y$. If you use a `for()` loop, this part of the function should look something like:

```
y <- 0;
for (some condition) {
  y <- y + contribution of specific power of x,
           controlled by value of FOR variable
}
```

Vectorizing the function is another strategy.

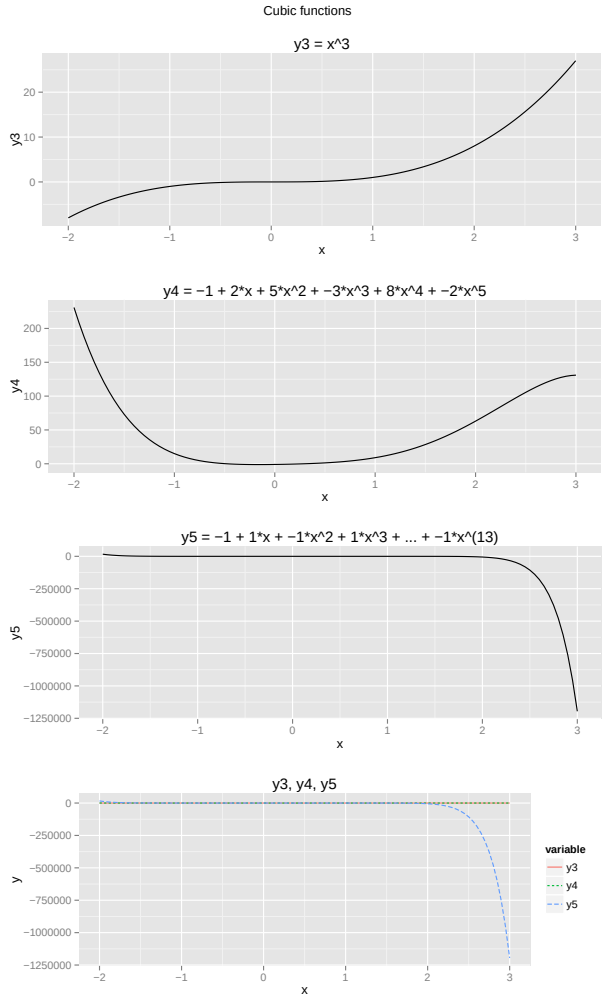
*Solution*: Here's one example of the `polyfn()`

```
# polyfn calculates a polynomial fuction of x with coefficient vector a
#   Default coefficients are 0s.
polyfn <- function(x=0, a=0) {
  # number of coeffcients
  k <- length(a)
  # initialize y to 0 with length of x
  y <- rep(0, length(x))

  # add to y successive coefficients times powers of x
  for (i in 1:k) {
    y <- y + a[i]*x^(i-1)
  }
  return(y)
}
```

Three polynomial functions and their plots.

```
a <- c(0,0,0,1)
y3 <- polyfn(x, a)

a <- c(-1,2,5,-3,8,-2)
y4 <- polyfn(x, a)

a <- c(1,-1,1,-1,1,-1,1,-1,1,-1,1,-1,1,-1)
y5 <- polyfn(x, a)


# create a data frame to hold all three vectors
dfpoly <- data.frame(x, y3, y4, y5)

# There are several fine answers to this problem.
# Below I use ggplot2 with geom_line().
library(ggplot2)
```

```
p1 <- ggplot(dfpoly, aes(x = x, y = y3))
p1 <- p1 + geom_line()
p1 <- p1 + labs(title = "y3 = x^3")

p2 <- ggplot(dfpoly, aes(x = x, y = y4))
p2 <- p2 + geom_line()
p2 <- p2 + labs(title = "y4 = -1 + 2*x + 5*x^2 + -3*x^3 + 8*x^4 + -2*x^5")

p3 <- ggplot(dfpoly, aes(x = x, y = y5))
p3 <- p3 + geom_line()
p3 <- p3 + labs(title = "y5 = -1 + 1*x + -1*x^2 + 1*x^3 + ... + -1*x^(13)")


# (c)
# reshape data in long format
library(reshape2)
dfpoly2 <- melt(dfpoly, id.vars="x")

p4 <- ggplot(dfpoly2, aes(x = x, y = value, colour = variable))
p4 <- p4 + geom_line(aes(linetype = variable))
p4 <- p4 + labs(title = "y3, y4, y5")
p4 <- p4 + ylab("y")


# plot all three together
library(gridExtra)
grid.arrange(p1, p2, p3, p4, ncol=1, top="Cubic functions")
```


Cubic functions
y3 = x^3
y3
x
y4 = −1 + 2*x + 5*x^2 + −3*x^3 + 8*x^4 + −2*x^5
y4
x
y5 = −1 + 1*x + −1*x^2 + 1*x^3 + ... + −1*x^(13)
y5
x
y3, y4, y5
y
x
variable
y3
y4
y5