# Chick Weight Challenge

In this challenge exercise, we're going to explore a dataset from an experiment on the effect of diet on early growth of chicks. At the end, we will use two useful packages: R Color Brewer which creates nice color palettes and R Markdown to turn our analysis into a final report.

**Download and read the comma-separated file "ChickWeight.csv".**

```
# Note: this dataset can also be accessed directly from the ChickWeight package in R
# (see ?ChickWeight)
chick = read.table("../ChickWeight_Challenge_EASY/ChickWeight.csv", header = T, sep = ",")
```

**First, explore the data.**
**How many chicks are in the dataset?**
**How many different diets are in the experiment?**

```
length(unique(chick$Chick))
```

```
## [1] 50
```

```
length(unique(chick$Diet))
```

```
## [1] 4
```

**What are the range of chick weights in the dataset?**

```
range(chick$weight)
```

```
## [1]  35 373
```

**What chick was the heaviest? Lightest?**

```
chick$Chick[which.max(chick$weight)] # heaviest
```

```
## [1] 35
```

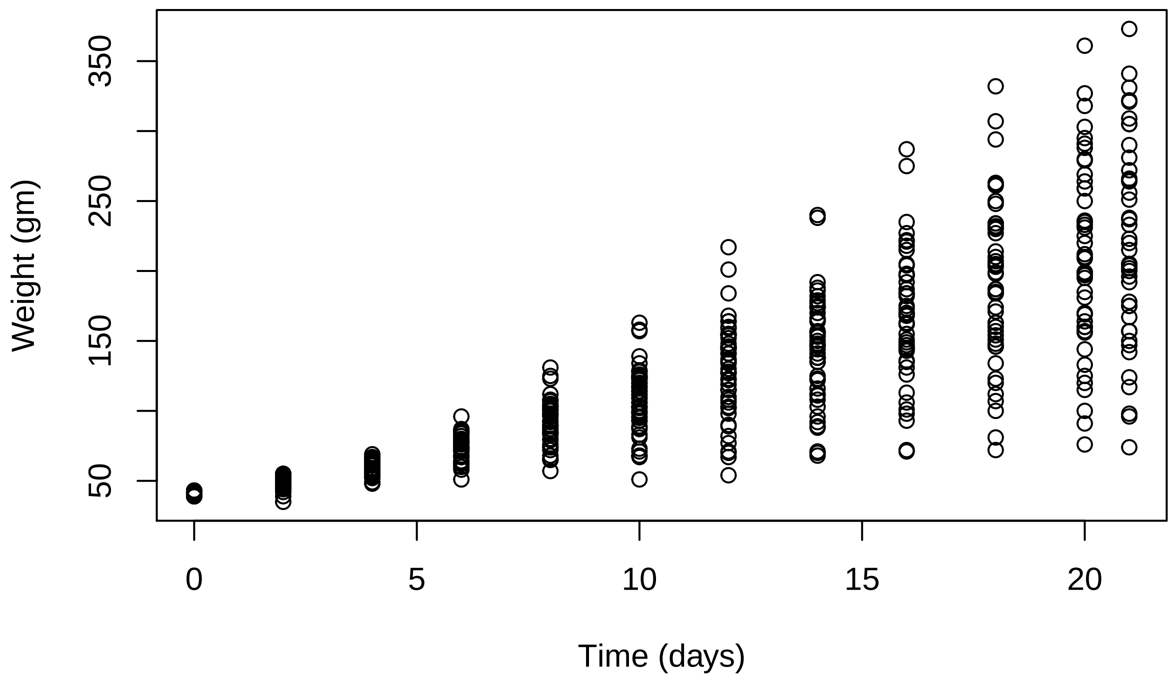
```
chick$Chick[which.min(chick$weight)] # lightest
```

```
## [1] 18
```

```
# Equivalently could do chick$Chick[which(chick$weight == max(chick$weight))]
```

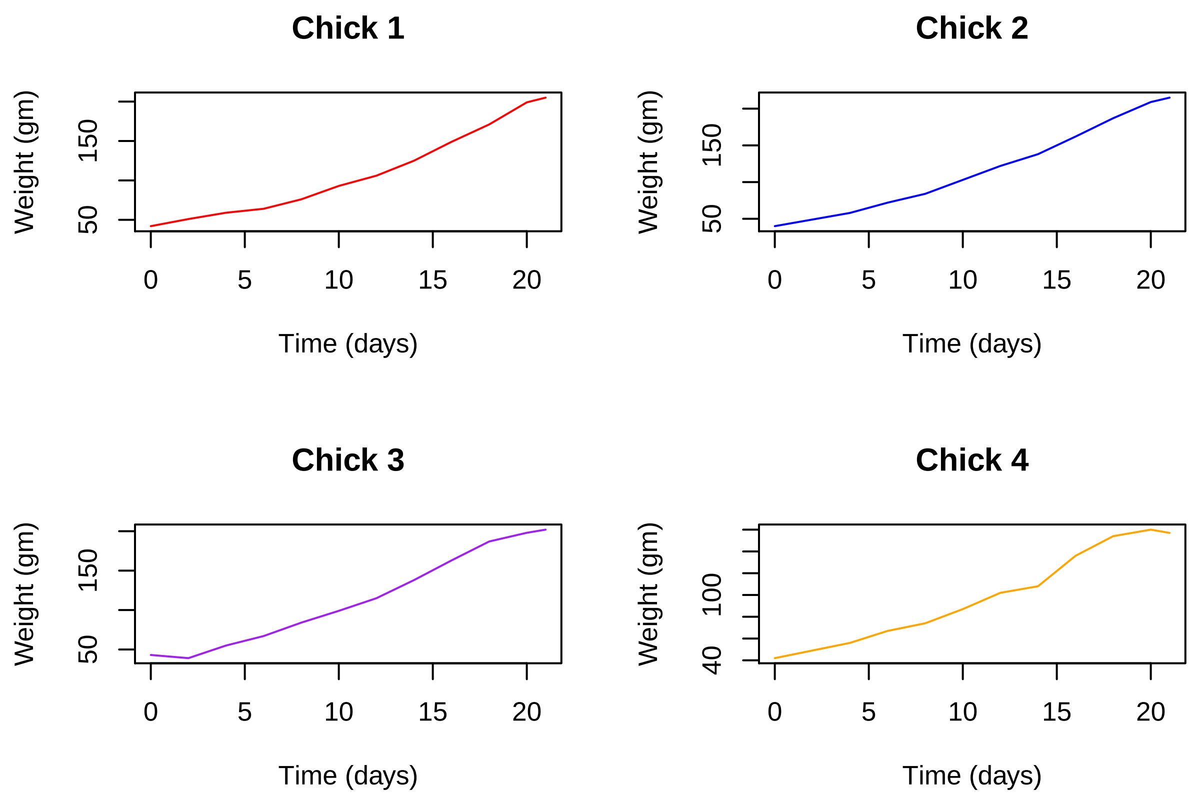
**To vizualize the basics of the data, plot weight versus time**

```r
plot(chick$weight ~ chick$Time,
     xlab = "Time (days)",
     ylab = "Weight (gm)")
```

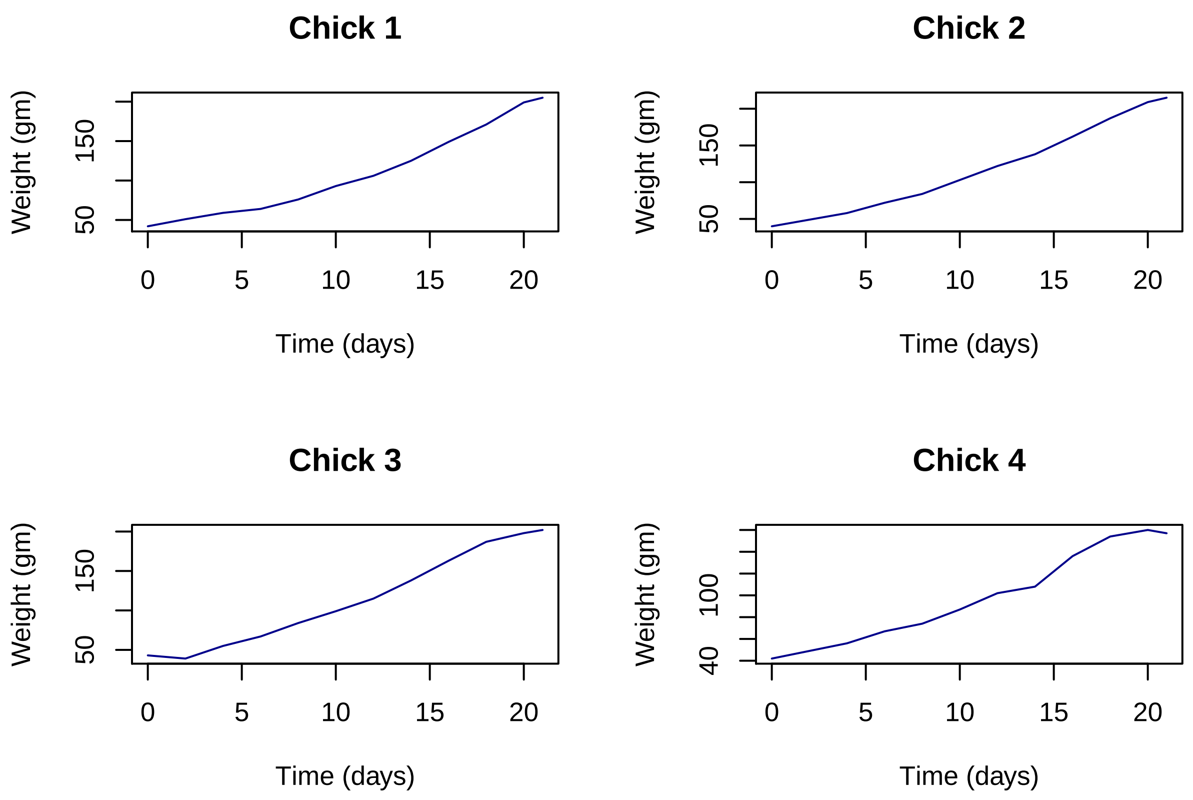


**Create a multipanel plot of the first four chick weight versus time**
**Hint: Use par()**

```r
par(mfrow = c(2,2))
plot(chick$weight[chick$Chick == 1] ~ chick$Time[chick$Chick == 1],
     type = "l", main = "Chick 1", xlab = "Time (days)", ylab = "Weight (gm)", col = "red")
plot(chick$weight[chick$Chick == 2] ~ chick$Time[chick$Chick == 2],
     type = "l", main = "Chick 2", xlab = "Time (days)", ylab = "Weight (gm)", col = "blue")
plot(chick$weight[chick$Chick == 3] ~ chick$Time[chick$Chick == 3],
     type = "l", main = "Chick 3", xlab = "Time (days)", ylab = "Weight (gm)", col = "purple")
plot(chick$weight[chick$Chick == 4] ~ chick$Time[chick$Chick == 4],
     type = "l", main = "Chick 4", xlab = "Time (days)", ylab = "Weight (gm)", col = "orange")
```

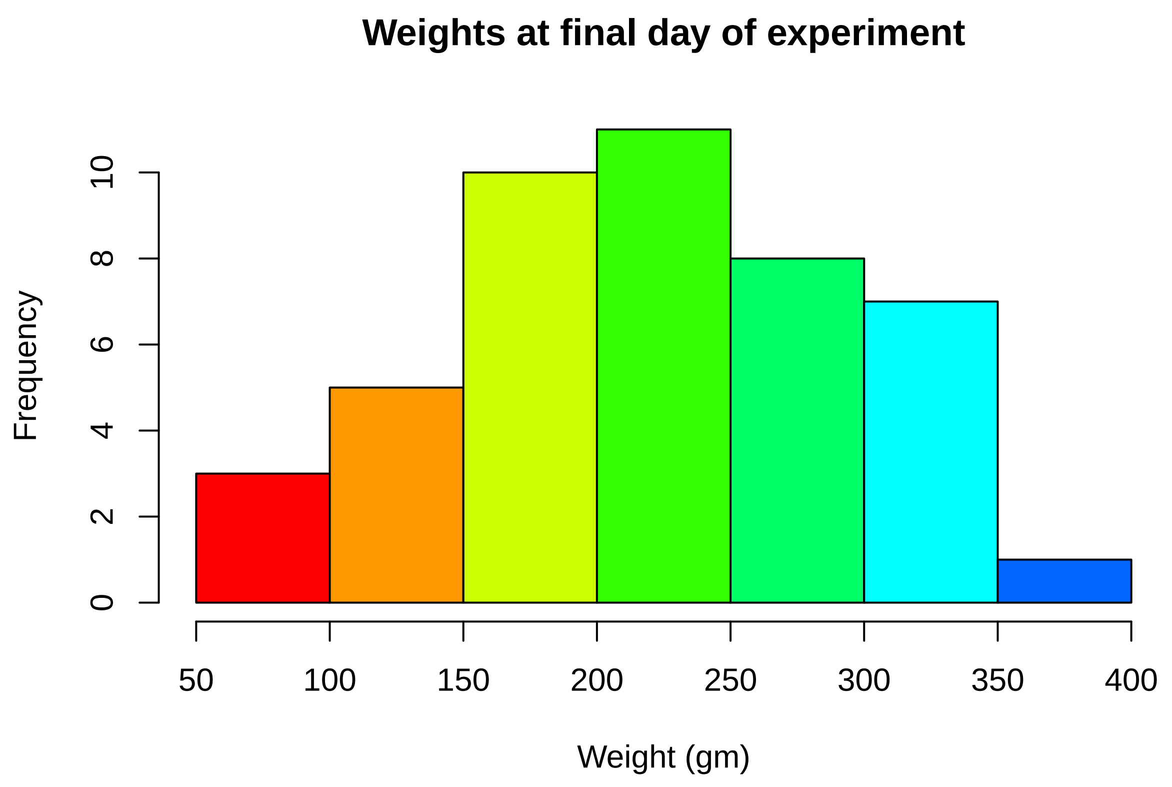


```r
# This can also be accomplished with a for loop:
for (i in 1:4){ # for each chick i
  plot(chick$weight[chick$Chick == i] ~ chick$Time[chick$Chick == i],
       type = "l",
       main = paste("Chick", i), # Chick 1, Chick 2 etc.
       xlab = "Time (days)",
       ylab = "Weight (gm)",
       col = "darkblue")
}
```



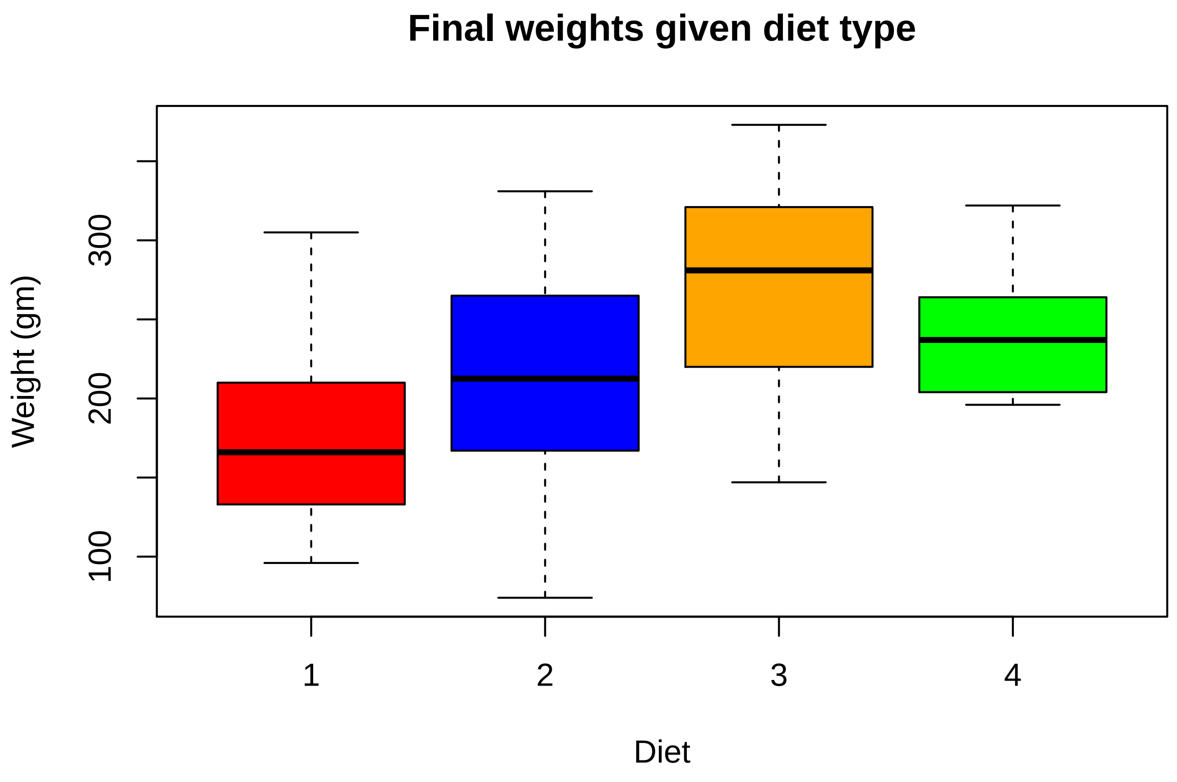
**Plot a histogram of the weights of the chicks at the final day of the experiments (i.e. only the chicks who made it to the last day)**

```r
par(mfrow = c(1,1))
hist(chick$weight[chick$Time == max(chick$Time)],
     xlab = "Weight (gm)",
     main = "Weights at final day of experiment",
     col = rainbow(10))
```



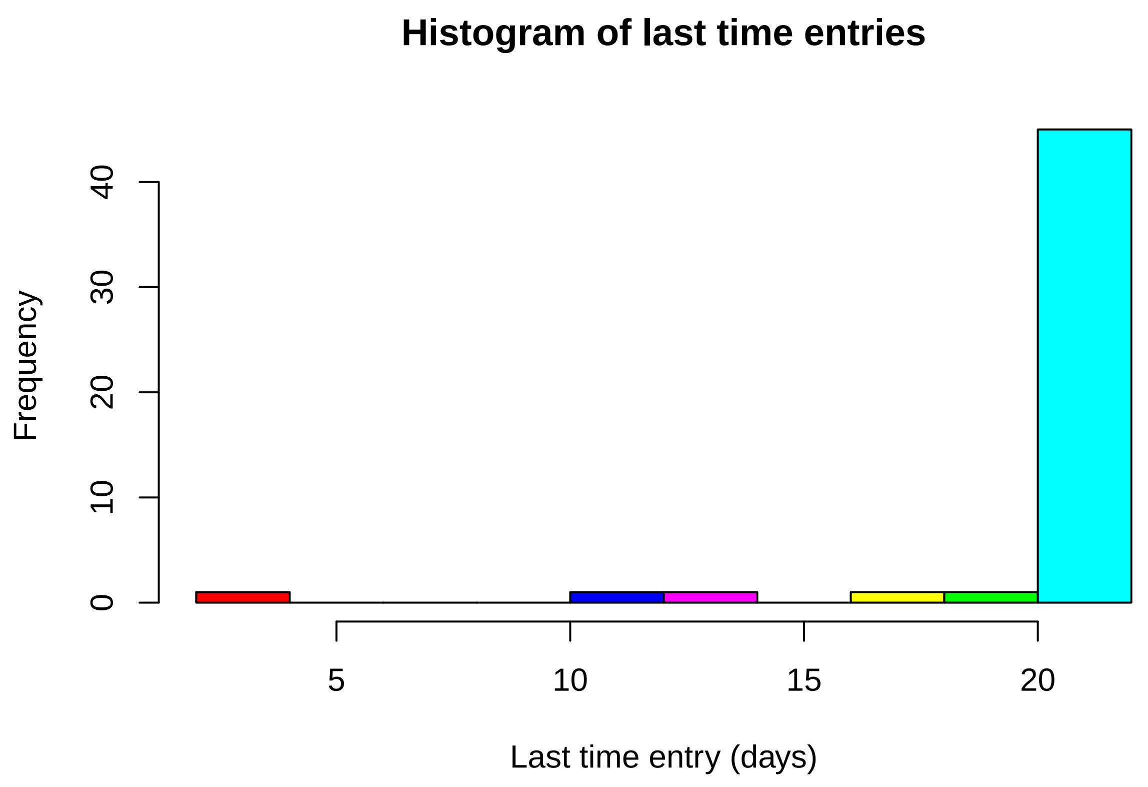
**Create a boxplot where the x-axis represents the different diets and the y-axis is the weights of the chicks at the final day of the experiments**

```
my.new = chick[chick$Time == max(chick$Time), ]
boxplot(weight ~ Diet,
        data = my.new,
        xlab = "Diet",
        ylab = "Weight (gm)",
        main = "Final weights given diet type",
        col = c("red", "blue", "orange", "green"))
```



**Find the last time entry for each chick in the dataset and plot a histogram of these times**

```r
lastTime = sapply(unique(chick$Chick), function(i)
  max(chick$Time[chick$Chick == i]))
hist(lastTime,
     xlab =  "Last time entry (days)",
     main = "Histogram of last time entries",
     col = rainbow(length(unique(lastTime))))
```

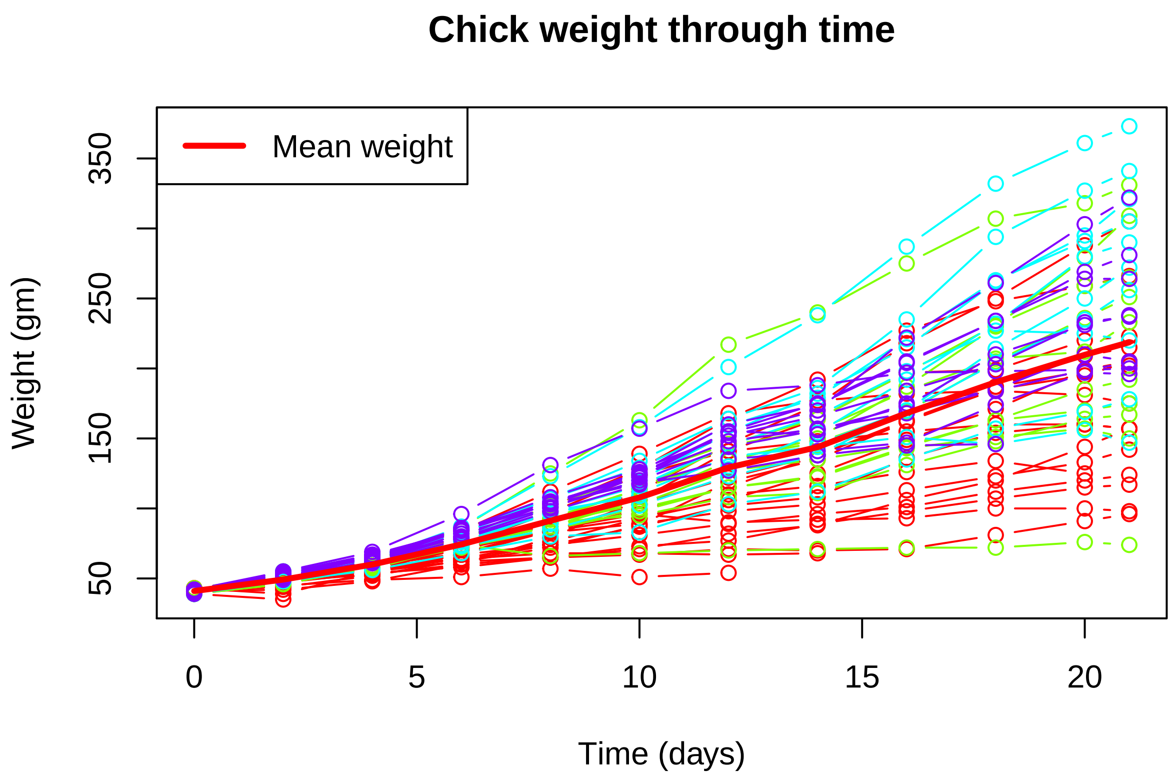


**Create a function that adds points to a existing plot for an individual chick's weight over time**

- As arguments for the function be just the individual chick to be plotted
- In plotting, change "type" to plot both lines and points
- Have the color argument be able to take a vector of colors which length diet type (that you will make later) so that each diet has the same color across chicks Hint: You can update this function later - if you're having trouble with coloring by diet, try plotting everything the same color first

```
plotChickWeight = function(individual) {
  points(chick$Time[chick$Chick == individual],
         chick$weight[chick$Chick == individual],
         type = "b",
         col = colors[chick$Diet[chick$Chick == individual]])
}

# Now, create an empty plot to add these points to
# (hint: can recreate the first plot with "col = NULL")
with(chick, plot(Time, weight,
                 col = NULL,
                 main = "Chick weight through time",
                 ylab = "Weight (gm)",
                 xlab = "Time (days)"))

# Create a vector of color values to pass the function,
# so that each diet type has a different color line/points
# Useful functions: rainbow()
colors = rainbow(length(unique(chick$Diet)))

# Now, use your function to add lines/points to the plot for all chicks
```

```r
my.out = sapply(unique(chick$Chick), plotChickWeight)

# Overlay a solid, thick red line of average weight (across all chicks)
# at each time interval on top of this graph
meanWeight = tapply(chick$weight, chick$Time, FUN = mean)
lines(x = unique(chick$Time), meanWeight,
      type = "l", col  = "red", lwd = 3)

#Add a legend to specify what the solid, thick red line represents
legend("topleft", lty = 1,
       col  = "red", lwd = 3, legend = "Mean weight")
```



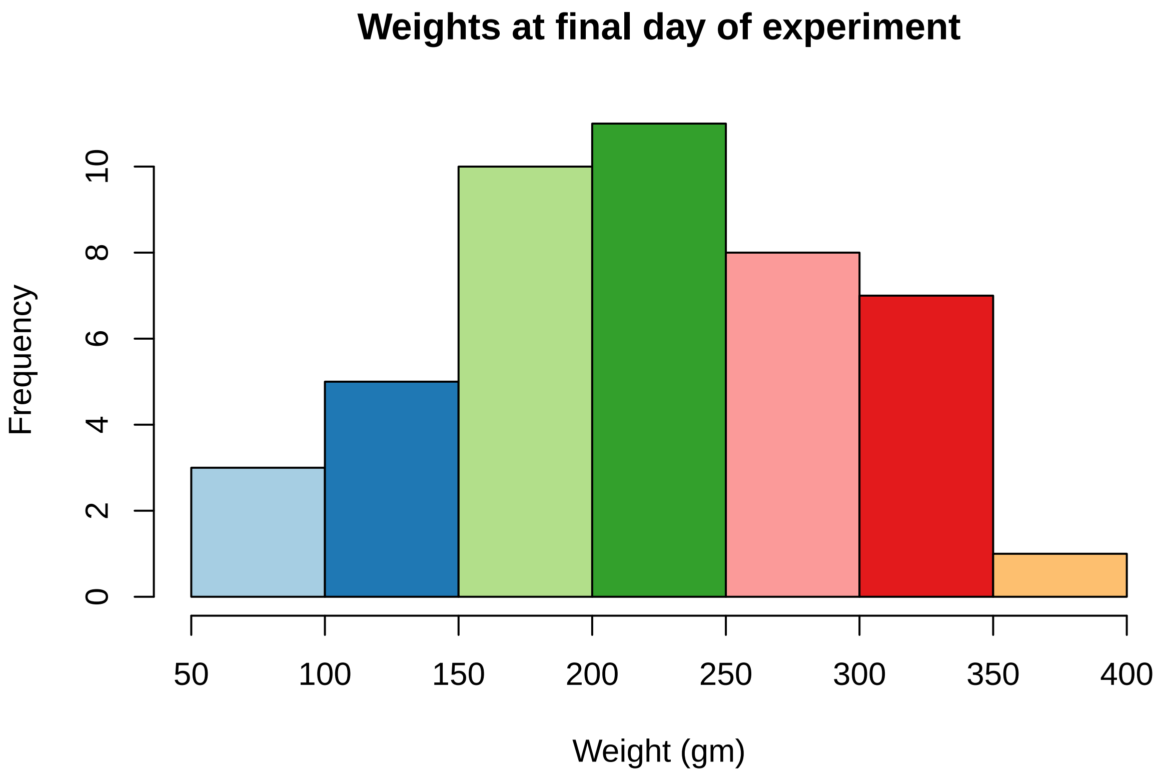
**Try using the package R Color Brewer to generate color palettes. Go to http://colorbrewer2.org/ to vizualize palettes. You can choose palettes that are colorblind safe, print friendly, etc.**

```r
# Install R Color Brewer
# install.packages("RColorBrewer")
library("RColorBrewer")
```

## Examples of R Color Brewer:

**Define color pallete with 10 colors and re-plot histogram of histogram of the weights of the chicks at the final day of the experiments in these colors**
**Note: if histogram has n breaks and n is less than 10, it will just use first n colors. If n is greater than 10, it will reuse colors.**

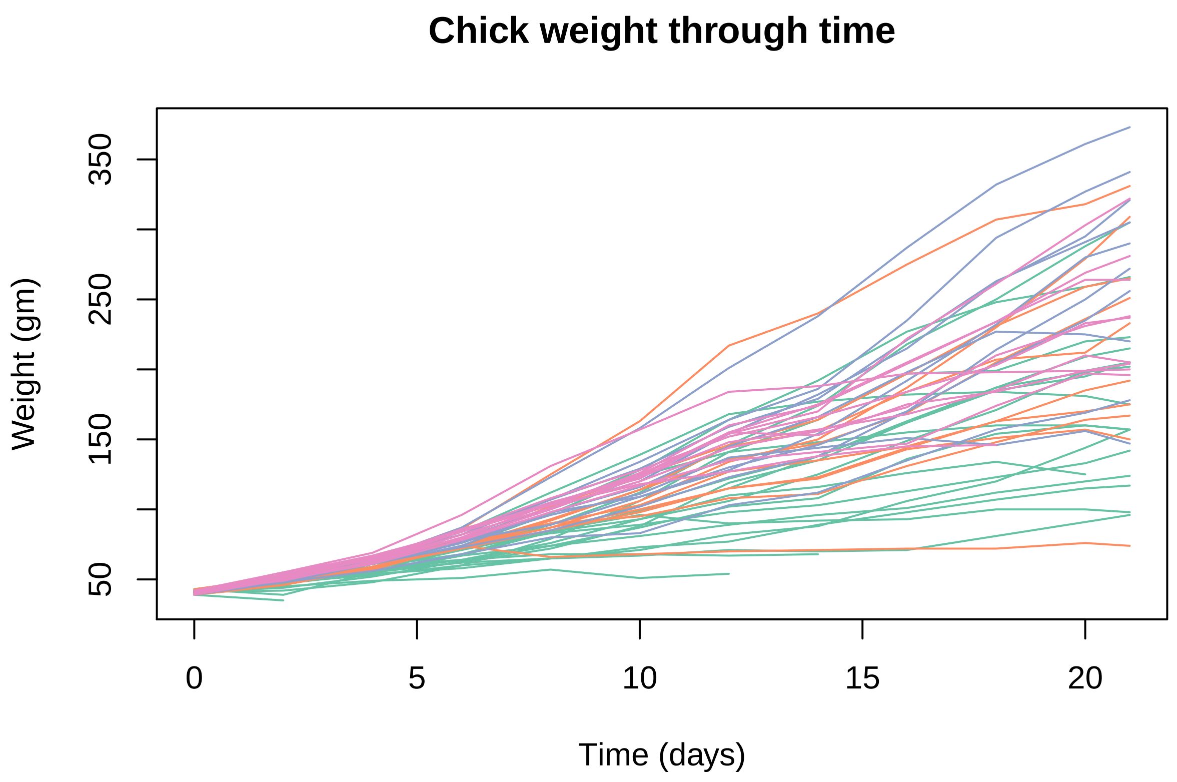
```
my.colors = brewer.pal(10, "Paired")
hist(chick$weight[chick$Time == max(chick$Time)],
     xlab = "Weight (gm)",
     main = "Weights at final day of experiment",
     col = my.colors)
```



**Re-do final plot with new colors and solid lines instead of points and lines**

```
plotChickWeight_newColors = function(individual) {
  lines(chick$Time[chick$Chick == individual],
        chick$weight[chick$Chick == individual],
        col = new.colors[chick$Diet[chick$Chick == individual]])
}

with(chick,
     plot(Time, weight,
          col = NULL,
          main = "Chick weight through time",
          ylab = "Weight (gm)",
          xlab = "Time (days)"))
```

```r
new.colors = brewer.pal(length(unique(chick$Diet)), "Set2")

my.out = sapply(unique(chick$Chick), plotChickWeight_newColors)
```



**Now, let's put this into R Markdown, a package that makes nice final reports of your code and figures**
**Go back and make sure your plots have interpretable axes labels and titles.**
**If you're interested, change the colors and point/line types to make your figures more vizually appealing.**

```r
# Install R Markdown
# install.packages("rmarkdown")
library("rmarkdown")
```

**Use R Studio to create new R Markdown file with the relevant code and output that you generated above. Create an HTML or PDF document.**