Final Project: Analysis

Josh Wilkins

8/02/2018

Abstract

A collection of recipes were recorded and stored in individual files as a means of collaboration amongst the class. The most recent version of this data can be found here. I used a series of functions to help standardize and properly format the recipe tab files so the analysis is kept separate. This analysis includes the comparison between two methods of calculating calories. Atwater factors can be used to compute calories, but a simpler, and more commonly used formula, is CHO=4calories/gm; PRO=4,FAT=9. This analysis also tries to determine the method Wansink used to calculate total calories, but concludes inconclusive on this front.

Minor Adjustments

Read in the formatted data set.

```
master.list <- read.csv("Ingredients.csv")

# Don't care to keep the amount or measure columns
master.list$Amount <- NULL
master.list$Measure <- NULL

print(head(master.list))
```

```
##                                                                  Ingredient
## 1 apples, raw, with skin (includes foods for usda's food distribution program)
## 2                                                          sugar, turbinado
## 3                                                            butter, salted
## 4 apples, raw, with skin (includes foods for usda's food distribution program)
## 5                                                          sugar, turbinado
## 6                                                            butter, salted
##   NDB_No Year         Recipe      Gm_Wgt
## 1   9003 1936 AppleDumplings  182.000000
## 2  19908 1936 AppleDumplings   34.500000
## 3   1001 1936 AppleDumplings    9.466667
## 4   9003 2006 AppleDumplings 1092.000000
## 5  19908 2006 AppleDumplings  101.000000
## 6   1001 2006 AppleDumplings   56.750000
```

For these calculations, the weight column is actually skewed a little high; Sometimes not all of an ingredient can be used. Seeds, pits, and bones are some examples of this. In the food description file, there is a column that accounts for this called "Refuse", which is given as a percentage of the ingredient by weight. So all we have to do is adjust the weight column by this percentage to get better results in later analyses.

For this we are going to need the food description file, let's read that in now.

```
# Reading in the food description table
food_des.dat <- read.table("../SR-Leg_ASC/FOOD_DES.txt", header=FALSE, sep="^", quote="~")
names(food_des.dat) <- c("NDB_No", "FdGrp_Cd", "Long_Desc", "Shrt_Desc", "ComName", "ManufacName", "Sur
```

From which we'll perform the adjustment now to the weights

```
master.list$Adj_Gm_Wgt <- 0  # Weight of fat in ingredient

for(i in 1:nrow(master.list)) {
  master.list[i,]$Adj_Gm_Wgt <- master.list[i,]$Gm_Wgt * (1 - subset(food_des.dat, NDB_No == master.lis
}
```

Weight by Factor

First we need to split the weight of each ingredient into 3 separate weights; protein, carbs, and fat (I'll call these factors to keep things easier to understand from this point forward). To do this, a couple additional tables are needed. Let's read these tables in now.

```
# Reading in the nutrient definition table
nutr_def.dat <- read.table("../SR-Leg_ASC/NUTR_DEF.txt", header=FALSE, sep="^", quote="~")
names(nutr_def.dat) <- c("Nutr_No", "Units", "Tagname", "NutrDesc", "Num_Dec", "SR_Order")

# Reading in the nutrient data table
nut_data.dat <- read.table("../SR-Leg_ASC/NUT_DATA.txt", header=FALSE, sep="^", quote="~")
names(nut_data.dat) <- c("NDB_No", "Nutr_No", "Nutr_Val", "Num_Data_Pts", "Std_Error", "Src_Cd", "Deriv
```

The nutrient definition table is needed for the nutrion number of each factor. Using these numbers in the nutrient data table, we can find the corresponding nutritional values for each ingredient. The nutritional value represents the number of grams of each factor in a 100g sample of the ingredient.

Following that thought, lets first grab the nutrion numbers for each factor.

```
cat("Nutr_No\n")
```

```
## Nutr_No
```

```
cat("Protein:", Nutr.No.Protein <- nutr_def.dat$Nutr_No[grepl('protein', nutr_def.dat$NutrDesc, ignore.
```

```
## Protein: 203
```

```
cat("Fats:\t", Nutr.No.Fat <- nutr_def.dat$Nutr_No[grepl('lipid', nutr_def.dat$NutrDesc, ignore.case=TRU
```

```
## Fats:    204
```

```
cat("Carbs:\t", Nutr.No.Carbs <- nutr_def.dat$Nutr_No[grepl('carb', nutr_def.dat$NutrDesc, ignore.case="
```

```
## Carbs:   205
```

Then we can add weights by factor using these nutrition numbers and the NDB_No of each ingredient.

```
# Create dummy vars to fill
master.list$Gm_Wgt_Pro <- 0  # Weight of protein in ingredient
master.list$Gm_Wgt_CHO <- 0  # Weight of carbs in ingredient
master.list$Gm_Wgt_Fat <- 0  # Weight of fat in ingredient

for(i in 1:nrow(master.list)) {
  # Need the 1/100 factor since Nutr_Val is given by a 100g sample
  master.list[i,]$Gm_Wgt_Pro <-
    (master.list[i,]$Adj_Gm_Wgt/100) * subset(nut_data.dat,
                                          Nutr_No == Nutr.No.Protein &
                                          NDB_No == master.list[i,]$NDB_No)$Nutr_Val
```

```
  master.list[i,]$Gm_Wgt_CHO <-
    (master.list[i,]$Adj_Gm_Wgt/100) * subset(nut_data.dat,
                                          Nutr_No == Nutr.No.Fat &
                                          NDB_No == master.list[i,]$NDB_No)$Nutr_Val

  master.list[i,]$Gm_Wgt_Fat <-
    (master.list[i,]$Adj_Gm_Wgt/100) * subset(nut_data.dat,
                                          Nutr_No == Nutr.No.Carbs &
                                          NDB_No == master.list[i,]$NDB_No)$Nutr_Val
}

print(head(master.list))
```

```
##                                                                  Ingredient
## 1 apples, raw, with skin (includes foods for usda's food distribution program)
## 2                                                          sugar, turbinado
## 3                                                            butter, salted
## 4 apples, raw, with skin (includes foods for usda's food distribution program)
## 5                                                          sugar, turbinado
## 6                                                            butter, salted
##   NDB_No Year         Recipe      Gm_Wgt Adj_Gm_Wgt Gm_Wgt_Pro Gm_Wgt_CHO
## 1   9003 1936 AppleDumplings  182.000000 163.800000 0.42588000   0.278460
## 2  19908 1936 AppleDumplings   34.500000  34.500000 0.00000000   0.000000
## 3   1001 1936 AppleDumplings    9.466667   9.466667 0.08046667   7.678413
## 4   9003 2006 AppleDumplings 1092.000000 982.800000 2.55528000   1.670760
## 5  19908 2006 AppleDumplings  101.000000 101.000000 0.00000000   0.000000
## 6   1001 2006 AppleDumplings   56.750000  56.750000 0.48237500  46.029925
##   Gm_Wgt_Fat
## 1   22.62078
## 2   34.43100
## 3    0.00568
## 4  135.72468
## 5  100.79800
## 6    0.03405
```

This is where the two methods for calculating the calories per factor diverge, but the nutritional value numbers we just grabbed will be used in both calculations.

4-4-9 Calculation

In the simplified method (4-4-9 approach), each factor is simply multiplied by 4 or 9 (4 for carbs and protein, 9 for fat) to get the number of calories for each factor in each ingredient.

```
master.list$ProCalsSimple <- master.list$Gm_Wgt_Pro * 4  # 4-4-9 method for calories from protein
master.list$CHOCalsSimple <- master.list$Gm_Wgt_CHO * 4  # 4-4-9 method for calories from carbs
master.list$FatCalsSimple <- master.list$Gm_Wgt_Fat * 9  # 4-4-9 method for calories from fat
```

Atwater Calculation

In the Atwater method, we have to find the multiplier for each factor. For this we'll need the food description table, which we read in earlier for some modifications to the weight column. We then just simply multiply the

"Pro_Factor", "CHO_Factor", and "Fat_Factor" for each ingredient by the previously calculated weights of protein, carbs, and fat.

```
# Create dummy vars to fill
master.list$ProCalsAtwater <- 0  # Atwater calculated calories from protein
master.list$FatCalsAtwater <- 0  # Atwater calculated calories from fat
master.list$CHOCalsAtwater <- 0  # Atwater calculated calories from carbs

for(i in 1:nrow(master.list)) {
  master.list[i,]$ProCalsAtwater <- master.list[i,]$Gm_Wgt_Pro * subset(food_des.dat, NDB_No == master.l
  master.list[i,]$CHOCalsAtwater <- master.list[i,]$Gm_Wgt_CHO * subset(food_des.dat, NDB_No == master.l
  master.list[i,]$FatCalsAtwater <- master.list[i,]$Gm_Wgt_Fat * subset(food_des.dat, NDB_No == master.l
}
```

Comparison: Total Calorie Calculations

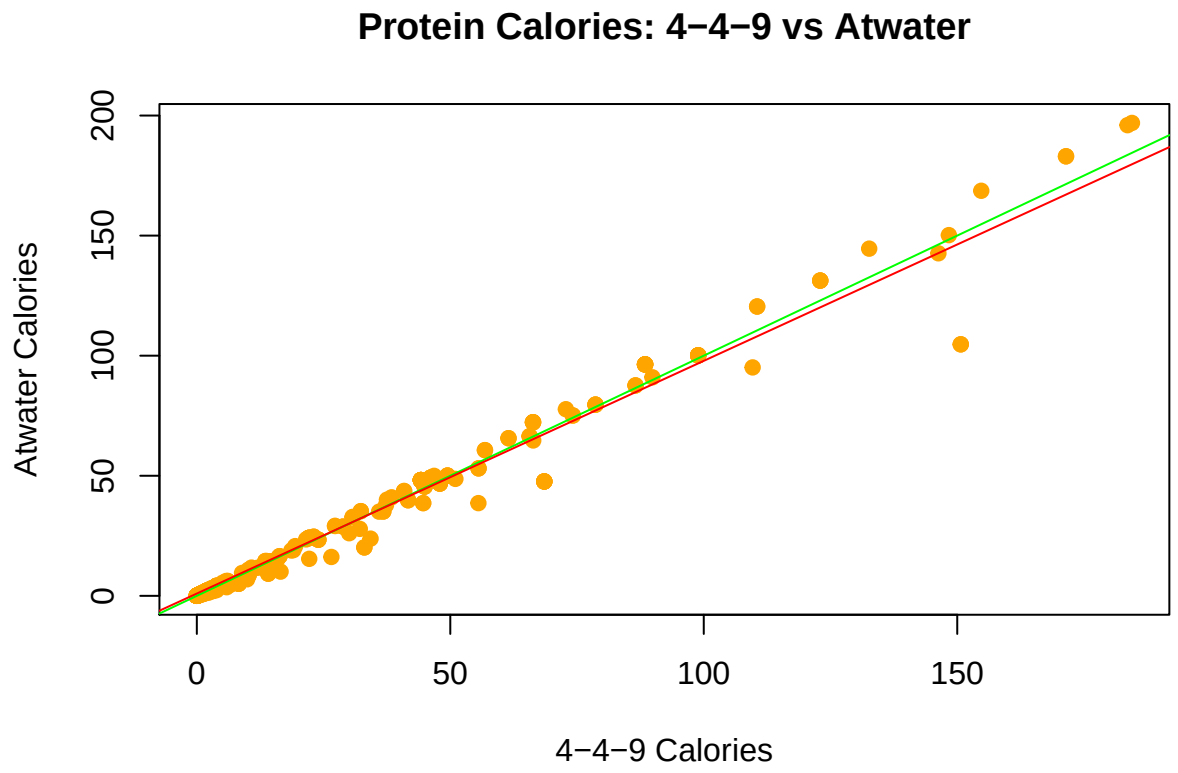
It would be interesting to see what the difference is between these two methods of calculating calories. Let's first look at the data frame to see that it came out correctly.

```
print(head(master.list))
```

```
##                                                                  Ingredient
## 1 apples, raw, with skin (includes foods for usda's food distribution program)
## 2                                                          sugar, turbinado
## 3                                                            butter, salted
## 4 apples, raw, with skin (includes foods for usda's food distribution program)
## 5                                                          sugar, turbinado
## 6                                                            butter, salted
##   NDB_No Year         Recipe      Gm_Wgt Adj_Gm_Wgt Gm_Wgt_Pro Gm_Wgt_CHO
## 1   9003 1936 AppleDumplings  182.000000 163.800000 0.42588000   0.278460
## 2  19908 1936 AppleDumplings   34.500000  34.500000 0.00000000   0.000000
## 3   1001 1936 AppleDumplings    9.466667   9.466667 0.08046667   7.678413
## 4   9003 2006 AppleDumplings 1092.000000 982.800000 2.55528000   1.670760
## 5  19908 2006 AppleDumplings  101.000000 101.000000 0.00000000   0.000000
## 6   1001 2006 AppleDumplings   56.750000  56.750000 0.48237500  46.029925
##   Gm_Wgt_Fat ProCalsSimple CHOCalsSimple FatCalsSimple ProCalsAtwater
## 1   22.62078     1.7035200       1.11384     203.58702      1.4309568
## 2   34.43100     0.0000000       0.00000     309.87900             NA
## 3    0.00568     0.3218667      30.71365       0.05112      0.3435927
## 4  135.72468    10.2211200       6.68304    1221.52212      8.5857408
## 5  100.79800     0.0000000       0.00000     907.18200             NA
## 6    0.03405     1.9295000     184.11970       0.30645      2.0597412
##   FatCalsAtwater CHOCalsAtwater
## 1    189.3359286       1.002456
## 2             NA             NA
## 3      0.0499272      29.715460
## 4   1136.0155716       6.014736
## 5             NA             NA
## 6      0.2992995     178.135810
```

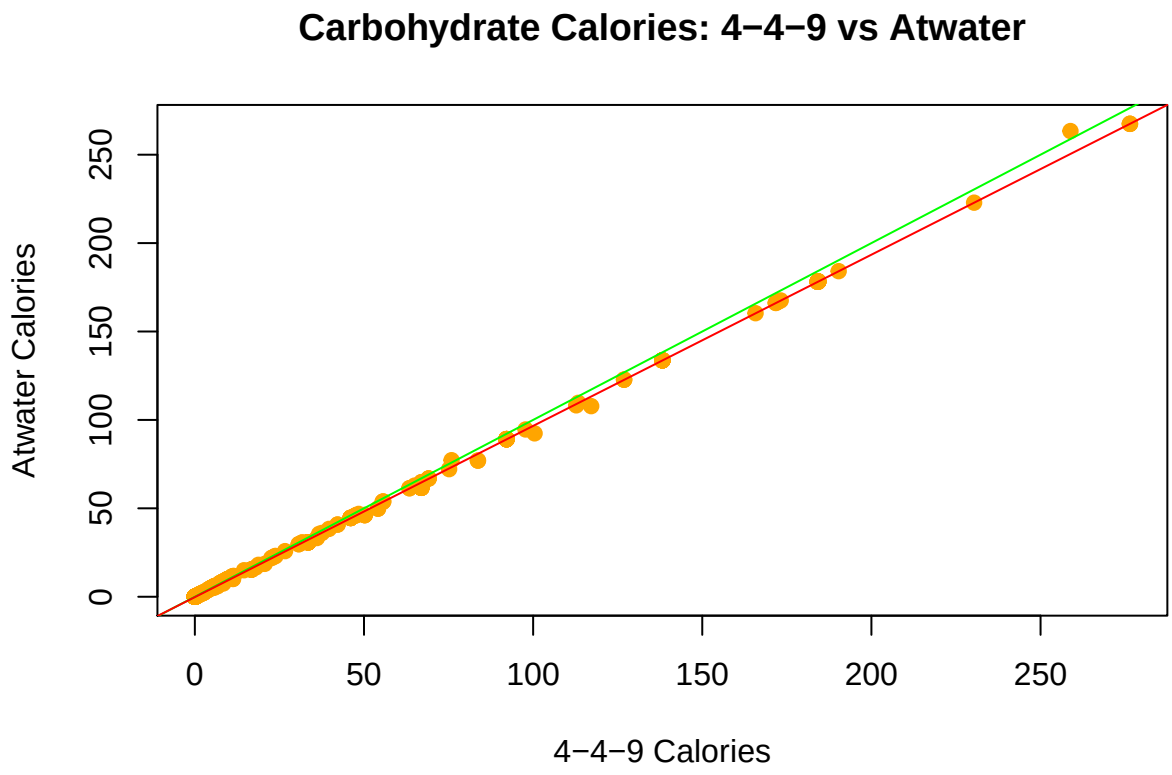
The best way of comparing the two methods would be creating a scatter plot between the points for each of the factors. I added a function to remove some of the larger calorie points since they just detract from the resolution of the scatter plots.

```r
# Simple way of removing outliers from upper bound by removing 3 sigma from mean
remove.outliers <- function(x, y){
  new.data.set <- data.frame(x=x, y=y)
  new.data.set <- subset(new.data.set, !is.na(y))

  right.bound1 <- mean(new.data.set$x) + 3*sd(new.data.set$x)
  right.bound2 <- mean(new.data.set$y) + 3*sd(new.data.set$y)

  new.data.set <- subset(new.data.set, x<right.bound1 & y<right.bound2)
  return(new.data.set)
}

# Plot protein 4-4-9 vs Atwater (first removes outliers)
new.data.set <- remove.outliers(master.list$ProCalsSimple, master.list$ProCalsAtwater)
Pro.Cals.lm <- lm(x~y, data=new.data.set)
plot(new.data.set$x, new.data.set$y,
     main="Protein Calories: 4-4-9 vs Atwater",
     xlab="4-4-9 Calories", ylab="Atwater Calories",
     pch=19, col="orange")
abline(0, 1, col="green")  # What an exact y=x match would be
abline(Pro.Cals.lm, col="red")  # What the slope actually is
```

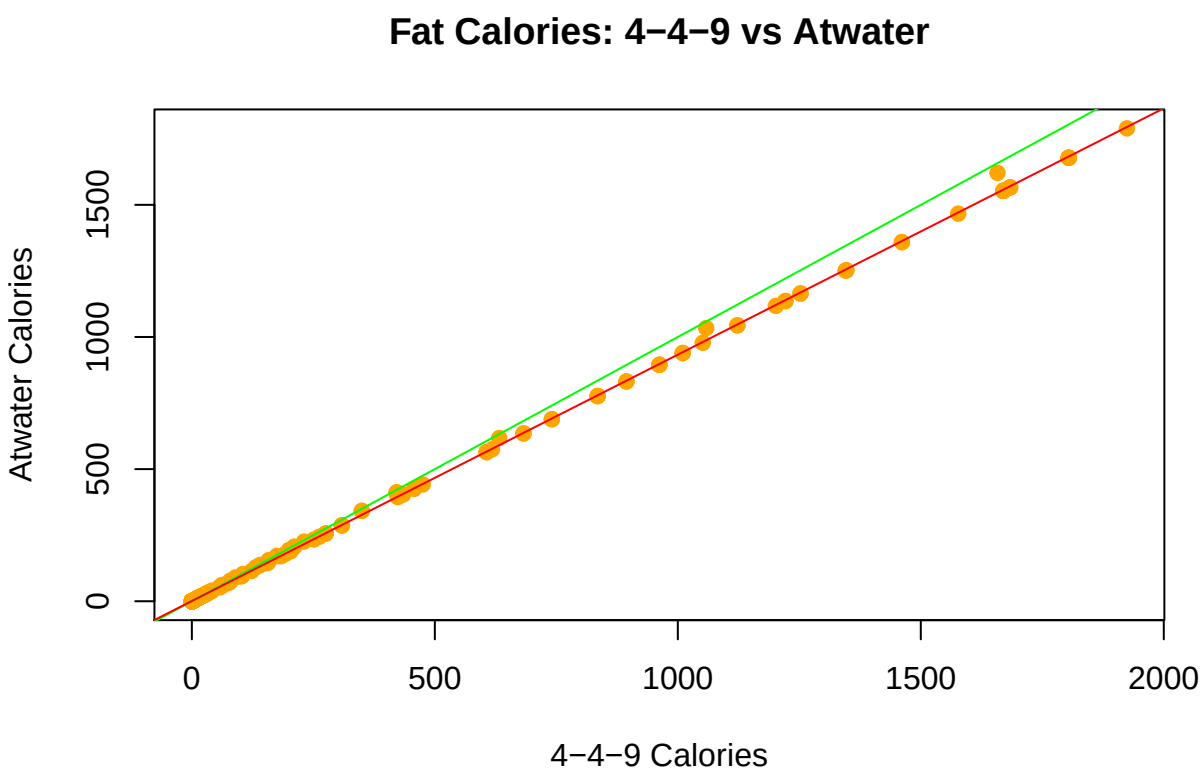



```r
# Plot carbohydrates 4-4-9 vs Atwater (first removes outliers)
new.data.set <- remove.outliers(master.list$CHOCalsSimple, master.list$CHOCalsAtwater)
CHO.Cals.lm <- lm(y~x, data=new.data.set)
plot(new.data.set$x, new.data.set$y,
```

```
     main="Carbohydrate Calories: 4-4-9 vs Atwater",
     xlab="4-4-9 Calories", ylab="Atwater Calories",
     pch=19, col="orange")
abline(0, 1, col="green")  # What an exact y=x match would be
abline(CHO.Cals.lm, col="red")  # What the slope actually is
```




```
# Plot fat 4-4-9 vs Atwater (first removes outliers)
new.data.set <- remove.outliers(master.list$FatCalsSimple, master.list$FatCalsAtwater)
Fat.Cals.lm <- lm(y~x, data=new.data.set)
plot(new.data.set$x, new.data.set$y,
     main="Fat Calories: 4-4-9 vs Atwater",
     xlab="4-4-9 Calories", ylab="Atwater Calories",
     pch=19, col="orange")
abline(0, 1, col="green") # What an exact y=x match would be
abline(Fat.Cals.lm, col="red")  # What the slope actually is
```



In the plots above, the green line represents what would be a one-to-one (y=x) relationship. The red line is the actual slope of the points. What this means is that, if the green line overlapped the red line, the 4-4-9 calculation would be identical to the Atwater calculation. We see that this approximation gets gradually worse for protein, then carbs, then fats. For protein, this 4-4-9 approximation is almost identical to the Atwater calculation. For calories from fat and carbs however, the green line deviates more from the red line indicating the Atwater calculation results in slightly less calories than the 4-4-9 calculation. This can be further illustrated by looking at the slopes of the linear regressions of these points (instead of the comparison to y=x).

```
cat("Protein Slope:\t", Pro.Cals.lm$coefficients[2], "\n")
```

```
## Protein Slope:    0.9690457
```

```
cat("Carb Slope:\t", CHO.Cals.lm$coefficients[2], "\n")
```

```
## Carb Slope:   0.9683092
```

```
cat("Fat Slope:\t", Fat.Cals.lm$coefficients[2], "\n")
```

```
## Fat Slope:    0.9322458
```

This confirms what we saw in the plots. The 4-4-9 calculation matches the Atwater calculation best for protein and gets slightly worse for carbs and fats. Here, a slope less than 1 indicates that the atwater calculation results in less calories than the 4-4-9 calculation.

Wansink Calculation

Another interesting question would be how Wansink came up with his calorie measurements; which method did he use? To do this we first need to read in the recipes that he used.

```
joy.dat <- read.csv("../JoyOfCooking.csv")
```

From this data set, we see that the only way we can find the answer to this question is to sum the calories up per recipe and then compare. The calories per recipe column also does not exist in this data set, so we must calculate that as well, let's do this first.

```
joy.dat$CaloriesperRecipe1936 <- joy.dat$CaloriesperServing1936 * joy.dat$ServingsperRecipe1936
joy.dat$CaloriesperRecipe2006 <- joy.dat$CaloriesperServing2006 * joy.dat$ServingsperRecipe2006
```

Let's get rid of the columns and rows that don't apply to this analysis.

```
joy.dat <- joy.dat[c("RecipeName", "CaloriesperRecipe1936", "CaloriesperRecipe2006")]

# If either calories per recipe column are not null, the row is good to keep
joy.dat <- subset(joy.dat, !is.na(CaloriesperRecipe1936) | !is.na(CaloriesperRecipe1936))
```

And then we can start summing the calories of our own set of recipes, grouped by year. First we'll sum the calories across factors for both the 4-4-9 and Atwater calories.

```
master.list$CalsSimple <- master.list$ProCalsSimple + master.list$CHOCalsSimple + master.list$FatCalsSi
master.list$CalsAtwater <- master.list$ProCalsAtwater + master.list$CHOCalsAtwater + master.list$FatCals
```

And then we can sum the calories by recipe, grouped by year. Its ok if we lose some stuff, the answer we're looking for won't require very many recipes to prove one way or another.

```
cals.per.recipe.long <- aggregate(list(CalsSimple=master.list$CalsSimple, CalsAtwater=master.list$CalsA
                                       by=list(RecipeName=master.list$Recipe, Year=master.list$Year), FUN=su
```

But we'll have to reshape the data now to match the format of the other set of data before we can merge to compare.

```
cals.per.recipe.wide <- reshape(cals.per.recipe.long,
                                timevar = "Year",
                                idvar = c("RecipeName"),
                                direction = "wide")
```

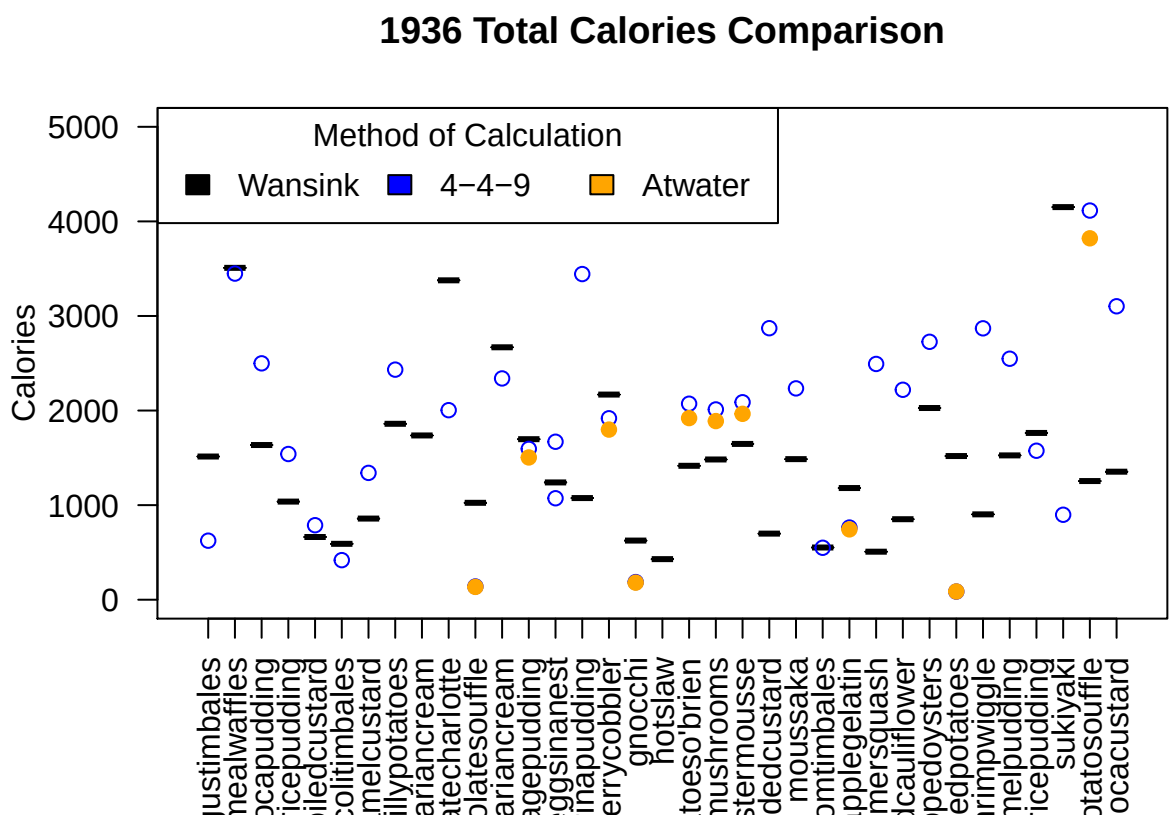
And then do the merge, again ignore the fact we could be losing some data here.

```
# Lowercase and remove white spaces from recipe names in both data sets to reduce data loss
joy.dat$RecipeName <- tolower(gsub(" ", "", joy.dat$RecipeName))
cals.per.recipe.wide$RecipeName <- tolower(gsub(" ", "", cals.per.recipe.wide$RecipeName))

# And then do the merge
recipe.cals <- merge(cals.per.recipe.wide, joy.dat, by="RecipeName")
```

Comparison: Wansink's Total Calorie Calculations

Now we get to the interesting section; more comparisons with some more plots. The first thing we can compare here is simply the three measures we have going for total calories; the Atwater calculated one, the 4-4-9 calculated one, and Wansink's calculated one.

```r
# Plot Wansink's 1936 calculated total calories per recipe
plot(factor(recipe.cals$RecipeName),
     recipe.cals$CaloriesperRecipe1936,
     main="1936 Total Calories Comparison",
     las=2, ylab="Calories", ylim=c(0,5000))

# Plot the 4-4-9 1936 calculated total calories per recipe
points(factor(recipe.cals$RecipeName),
       recipe.cals$CalsSimple.1936,
       las=2, col="blue")

# Plot the Atwater 1936 calculated total calories per recipe
points(factor(recipe.cals$RecipeName),
       recipe.cals$CalsAtwater.1936,
       las=2, col="orange", pch=19)

legend("topleft", title="Method of Calculation",
       c("Wansink", "4-4-9", "Atwater"), fill=c("black", "blue", "orange"), horiz=TRUE)
```




```r
# Plot Wansink's 2006 calculated total calories per recipe
plot(factor(recipe.cals$RecipeName),
     recipe.cals$CaloriesperRecipe2006,
     main="2006 Total Calories Comparison",
     las=2, ylab="Calories", ylim=c(0,5000))

# Plot the 4-4-9 2006 calculated total calories per recipe
```

```
points(factor(recipe.cals$RecipeName),
       recipe.cals$CalsSimple.2006,
       las=2, col="blue")

# Plot the Atwater 2006 calculated total calories per recipe
points(factor(recipe.cals$RecipeName),
       recipe.cals$CalsAtwater.2006,
       las=2, col="orange", pch=19)

legend("topleft", title="Method of Calculation",
       c("Wansink", "4-4-9", "Atwater"), fill=c("black", "blue", "orange"), horiz=TRUE)
```

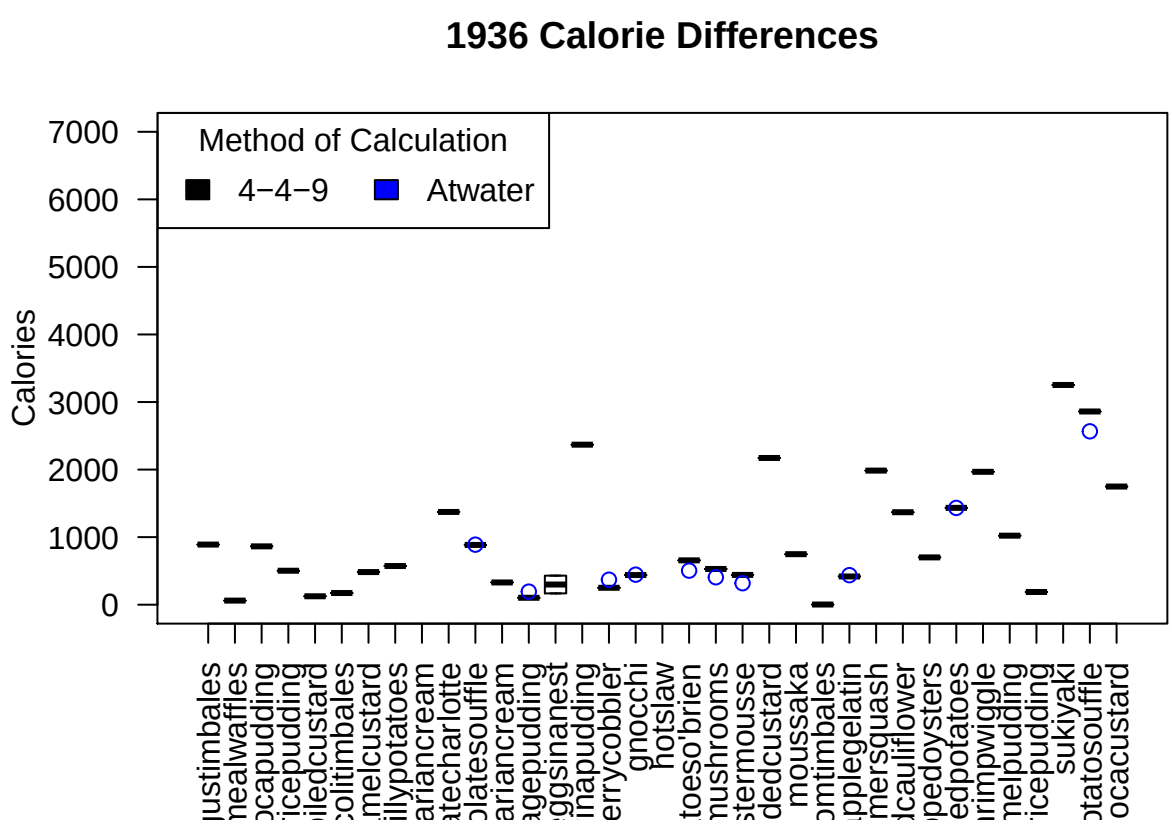
The black lines are Wansink's calculated total recipe values and act kind of as a baseline for comparison between the Atwater and 4-4-9 calculations. From this graph, it is not the easiest to tell if Wansink used the Atwater or the 4-4-9 calculations. And if you look closely at the data, you see that sometimes the 4-4-9 calculation is closer to Wansink's calculation and sometimes the Atwater calculation is closer to Wansink's calculation. It is inconclusive which method Wansink utilized at this point.

A different approach to this that might shed some more light on the question is taking the absolute differences between Wansink's measurements and the two methods.

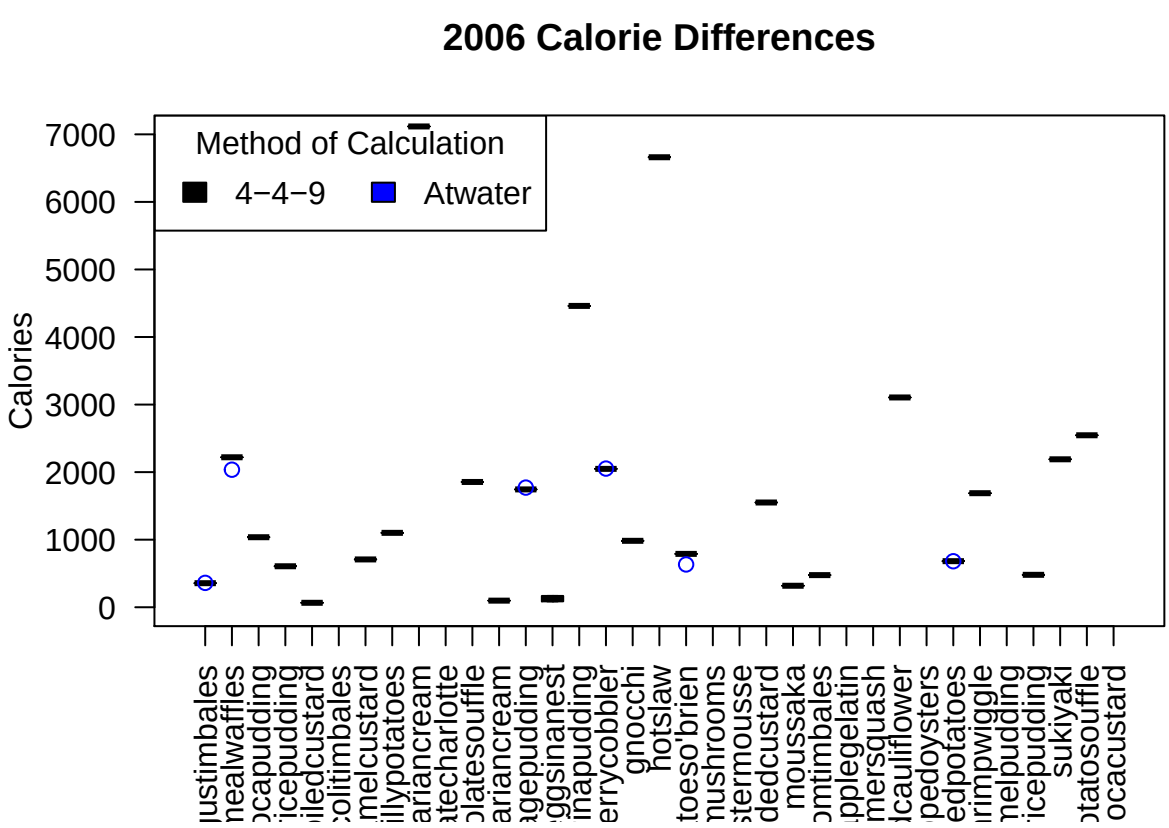
```
recipe.cals$SimpleDiff.1936 <- abs(recipe.cals$CalsSimple.1936 - recipe.cals$CaloriesperRecipe1936)
recipe.cals$SimpleDiff.2006 <- abs(recipe.cals$CalsSimple.2006 - recipe.cals$CaloriesperRecipe2006)
recipe.cals$AtwaterDiff.1936 <- abs(recipe.cals$CalsAtwater.1936 - recipe.cals$CaloriesperRecipe1936)
recipe.cals$AtwaterDiff.2006 <- abs(recipe.cals$CalsAtwater.2006 - recipe.cals$CaloriesperRecipe2006)
```

And we'll create some similar plots to see the differences between Wansink's calculations and each method.

```
# Plot for 1936 recipe calorie differences to joy.dat
plot(factor(recipe.cals$RecipeName),
     recipe.cals$SimpleDiff.1936,
     main="1936 Calorie Differences",
     las=2, ylab="Calories", ylim=c(0, 7000))

points(factor(recipe.cals$RecipeName),
       recipe.cals$AtwaterDiff.1936,
       las=2, col="blue")

legend("topleft", title="Method of Calculation",
    c("4-4-9","Atwater"), fill=c("black", "blue"), horiz=TRUE)
```




```
# Plot for 2006 recipe calorie differences to joy.dat
plot(factor(recipe.cals$RecipeName),
     recipe.cals$SimpleDiff.2006,
     main="2006 Calorie Differences",
     las=2, ylab="Calories", ylim=c(0, 7000))

points(factor(recipe.cals$RecipeName),
       recipe.cals$AtwaterDiff.2006,
       las=2, col="blue")

legend("topleft", title="Method of Calculation",
    c("4-4-9","Atwater"), fill=c("black", "blue"), horiz=TRUE)
```



In this case, Wansink's calculations are used as the baseline (that's why its not in the plot above, it was subtracted out). Turns out the difference between both the 4-4-9 and the Atwater calculations is much less than the difference to the total calorie calculations made by Wansink. It is impossible to distinguish which method he used for his calorie calculations. Probably neither the 4-4-9 or the Atwater calculations since theyre so different, some recipes were even thousands of calories off.

We could do a quick validation of his approach to calculating the ratio of the average total calories per recipe between years. This ratio should be approximately the same using any of the calculation methods.

```
cat("Mean Total Calorie Ratio Between Years\n")
```

```
## Mean Total Calorie Ratio Between Years
```

```
cat("4-4-9:\t", mean(recipe.cals$CalsSimple.1936, na.rm=TRUE)/mean(recipe.cals$CalsSimple.2006, na.rm=T
```

```
## 4-4-9:    0.862029
```

```
cat("Atwater:", mean(recipe.cals$CalsAtwater.1936, na.rm=TRUE)/mean(recipe.cals$CalsAtwater.2006, na.rm
```

```
## Atwater: 1.11232
```

```
cat("Wansink:", mean(recipe.cals$CaloriesperRecipe1936, na.rm=TRUE)/mean(recipe.cals$CaloriesperRecipe2
```

```
## Wansink: 0.9478949
```

They are all relatively the same, which would mean that Wansink did not alter his method of calculating total calories per recipe, although with such high standard deviations in the total calories, this comparison may not be meaningful.