9 Additional Graphs 1,3,4,5

Andrew Marshall

7/23/2018

# General Instructions

There are 5 exercises, each is worth 10 points. As usual, select 4 for grading. One exercise must be compeleted in both languages. There is no SAS template for these exercises, you must create your own SAS file.

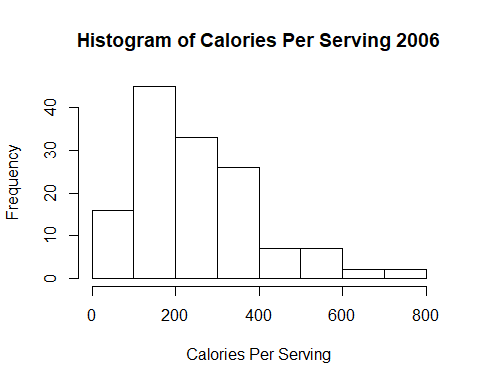
# Exercise 1.

Load the JoyOfCooking data set from lecture and create histograms, QQ-norm and box-whisker plots for calories per serving and servings per recipe. You can use data from either year. Add a title to each plot, identifying the data.

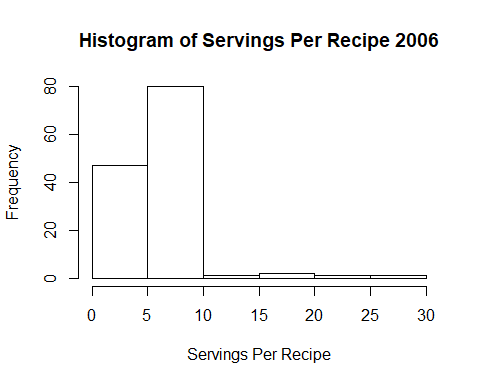
#This section will import the JoyOfCooking data from a CSV file into a dataframe.  
  
PathToJoy <- "C:/Users/drewm/Documents/GitHub/code-stat700/JoyOfCooking.csv"  
JoyOfCooking <- read.csv(PathToJoy,header=TRUE)  
JoyOfCooking

## RecipeName Class TooMuch  
## 1 Blueberry Muffins Per-Piece TRUE  
## 2 Brownies Cockaigne Per-Piece TRUE  
## 3 Corn Bread Muffins Per-Piece TRUE  
## 4 Rolled Biscuits Per-Piece TRUE  
## 5 Sugar Drop Cookies Per-Piece TRUE  
## 6 Waffles Per-Piece TRUE  
## 7 Apple Pie Per-Recipe TRUE  
## 8 Chicken Gumbo Per-Recipe TRUE  
## 9 Chocolate Cake Per-Recipe TRUE  
## 10 Corn Chowder Per-Recipe TRUE  
## 11 Baked Macaroni and Cheese Per-Serving TRUE  
## 12 Baked Pearl Tapioca Pudding Per-Serving TRUE  
## 13 Baked Rice Pudding Per-Serving TRUE  
## 14 Chicken a la King Per-Serving TRUE  
## 15 Chili with Meat and Beans Per-Serving TRUE  
## 16 Firm Omelet Per-Serving TRUE  
## 17 Hungarian Goulash Per-Serving TRUE  
## 18 Irish Stew Per-Serving TRUE  
## 19 Spanish Rice Per-Serving TRUE  
## 20 Stovetop Mac and Cheese Per-Serving TRUE  
## 21 Stovetop Rice Pudding Per-Serving TRUE  
## 22 Apple Dumplings Per-Serving FALSE  
## 23 Apple Tartlets Per-Serving FALSE  
## 24 Asparagus Timbales Per-Serving FALSE  
## 25 Bacon Cornmeal Waffles Per-Serving FALSE  
## 26 Baked Apples Per-Serving FALSE  
## 27 Baked Apples Stuffed with Sausage Per-Serving FALSE  
## 28 Baked Beets Per-Serving FALSE  
## 29 Baked Eggs Per-Serving FALSE  
## 30 Bavarian Berry Cream Per-Serving FALSE  
## 31 Blancmange Per-Serving FALSE  
## 32 Boiled Custard Per-Serving FALSE  
## 33 Boiled Potatoes Per-Serving FALSE  
## 34 Braised Red Cabbage Per-Serving FALSE  
## 35 Bread Pudding Per-Serving FALSE  
## 36 Breaded Fried Shrimp Per-Serving FALSE  
## 37 Broccoli Timbales Per-Serving FALSE  
## 38 Browned Potatoes Per-Serving FALSE  
## 39 Brussels Sprouts with Chestnuts Per-Serving FALSE  
## 40 Caramel Custard Per-Serving FALSE  
## 41 Chantilly Potatoes Per-Serving FALSE  
## 42 Charlotte Russe Per-Serving FALSE  
## 43 Cheese Fondue Per-Serving FALSE  
## 44 Cheese Souffle Cockaigne Per-Serving FALSE  
## 45 Chicken Jambalaya Per-Serving FALSE  
## 46 Chicken Pot Pie Per-Serving FALSE  
## 47 Chicken Salad Per-Serving FALSE  
## 48 Chocolate Bavarian Cream Per-Serving FALSE  
## 49 Chocolate Charlotte Per-Serving FALSE  
## 50 Chocolate Souffle Per-Serving FALSE  
## 51 Chow Mein Per-Serving FALSE  
## 52 Citrus Juice Medley Per-Serving FALSE  
## 53 Coffee Bavarian Cream Per-Serving FALSE  
## 54 Corn Bread Tamale Pie Per-Serving FALSE  
## 55 Cottage Pudding Per-Serving FALSE  
## 56 Crab or Tuna Souffle Per-Serving FALSE  
## 57 Cranberry Juice Per-Serving FALSE  
## 58 Creamed Chicken Per-Serving FALSE  
## 59 Creamed Eggs with Asparagus Tips Cockaigne Per-Serving FALSE  
## 60 Creamed Oysters Per-Serving FALSE  
## 61 Creamed Spinach Per-Serving FALSE  
## 62 Cucumber Mousse Per-Serving FALSE  
## 63 Curried Eggs Per-Serving FALSE  
## 64 Eggs in a Nest Per-Serving FALSE  
## 65 Farina Pudding Per-Serving FALSE  
## 66 Flank Steak with Dressing Per-Serving FALSE  
## 67 Floating Islands Per-Serving FALSE  
## 68 Fresh Fruit Souffle Per-Serving FALSE  
## 69 Fruit and Berry Cobbler Per-Serving FALSE  
## 70 Fruit Gelatin Per-Serving FALSE  
## 71 Fruit Whip Per-Serving FALSE  
## 72 Gelatin Fruit Salad Per-Serving FALSE  
## 73 German Potato Salad Per-Serving FALSE  
## 74 Gnocchi Per-Serving FALSE  
## 75 Golden Glow Salad Per-Serving FALSE  
## 76 Green Bean Casserole Per-Serving FALSE  
## 77 Ham and Cheese Souffle Per-Serving FALSE  
## 78 Ham Cakes with Pineapple and Sweet Potatoes Per-Serving FALSE  
## 79 Ham Loaf Per-Serving FALSE  
## 80 Hamburgers Per-Serving FALSE  
## 81 Herring Salad Per-Serving FALSE  
## 82 Hot Slaw Per-Serving FALSE  
## 83 Indian Pudding Per-Serving FALSE  
## 84 Leftover Potatoes O'Brien Per-Serving FALSE  
## 85 Lemon Gelatin Per-Serving FALSE  
## 86 Lemon Souffle Per-Serving FALSE  
## 87 Lima Beans and Mushrooms Per-Serving FALSE  
## 88 Lobster Mousse Per-Serving FALSE  
## 89 Lobster Newburg Per-Serving FALSE  
## 90 Lobster or Shrimp Salad Per-Serving FALSE  
## 91 Mashed Potatoes Per-Serving FALSE  
## 92 Meat Loaf I Per-Serving FALSE  
## 93 Meat Loaf II Per-Serving FALSE  
## 94 Molded Custard Per-Serving FALSE  
## 95 Moussaka Per-Serving FALSE  
## 96 Mushroom Souffle Per-Serving FALSE  
## 97 Mushroom Timbales Per-Serving FALSE  
## 98 Old-Fashioned Chocolate Pudding Per-Serving FALSE  
## 99 Orange Gelatin Per-Serving FALSE  
## 100 Orange-Tomato Juice Per-Serving FALSE  
## 101 Pineapple Gelatin Per-Serving FALSE  
## 102 Pineapple Snow Per-Serving FALSE  
## 103 Pineapple-Grapefruit Juice Per-Serving FALSE  
## 104 Riced Potatoes Per-Serving FALSE  
## 105 Rombauer Italian Rice Per-Serving FALSE  
## 106 Rote Grutze Per-Serving FALSE  
## 107 Sauerbrauten Per-Serving FALSE  
## 108 Sauteed Apples and Bacon Per-Serving FALSE  
## 109 Sauteed Summer Squash Per-Serving FALSE  
## 110 Scalloped Cauliflower Per-Serving FALSE  
## 111 Scalloped Oysters Per-Serving FALSE  
## 112 Scalloped Potatoes Per-Serving FALSE  
## 113 Scalloped Tomatoes Per-Serving FALSE  
## 114 Scrambled Eggs Per-Serving FALSE  
## 115 Seafood Salad Per-Serving FALSE  
## 116 Shepherd's Pie Per-Serving FALSE  
## 117 Shrimp Wiggle Per-Serving FALSE  
## 118 Snow Pudding Per-Serving FALSE  
## 119 Souffled Omelet Per-Serving FALSE  
## 120 Souffled Potatoes Per-Serving FALSE  
## 121 Sour Cream Apple Cake Souffle Per-Serving FALSE  
## 122 Standing Rib Roast Per-Serving FALSE  
## 123 Steamed Caramel Pudding Per-Serving FALSE  
## 124 Steamed Cauliflower Per-Serving FALSE  
## 125 Steamed Chocolate Feather Pudding Per-Serving FALSE  
## 126 Steamed Plum Pudding Per-Serving FALSE  
## 127 Sticky Toffee Pudding Per-Serving FALSE  
## 128 Sukiyaki Per-Serving FALSE  
## 129 Sweet Potato Souffle Per-Serving FALSE  
## 130 Sweet-and-Sour Beets Per-Serving FALSE  
## 131 Sweet-and-Sour Brisket Per-Serving FALSE  
## 132 Swiss Steak Per-Serving FALSE  
## 133 Tapioca Custard Per-Serving FALSE  
## 134 Tomato Aspic i Per-Serving FALSE  
## 135 Tomato Aspic ii Per-Serving FALSE  
## 136 Tomato Juice Per-Serving FALSE  
## 137 Tuna Salad Per-Serving FALSE  
## 138 Twice-Baked Potatoes Per-Serving FALSE  
## 139 Vanilla Souffle Per-Serving FALSE  
## 140 Waldorf Salad Per-Serving FALSE  
## 141 Welsh Rarebit Per-Serving FALSE  
## 142 White Veal Stew Per-Serving FALSE  
## ServingsperRecipe1936 ServingsperRecipe2006 CaloriesperServing1936  
## 1 NA NA 55.50  
## 2 NA NA 94.60  
## 3 NA NA 80.70  
## 4 NA NA 53.80  
## 5 NA NA 80.80  
## 6 NA NA 271.60  
## 7 NA NA NA  
## 8 NA NA NA  
## 9 NA NA NA  
## 10 NA NA NA  
## 11 6 10 204.70  
## 12 12 8 136.30  
## 13 8 8 129.70  
## 14 6 8 143.40  
## 15 8 8 286.74  
## 16 4 4 109.70  
## 17 6 6 439.50  
## 18 6 6 554.50  
## 19 4 6 195.70  
## 20 4 10 266.00  
## 21 8 8 220.40  
## 22 NA 6 293.60  
## 23 8 8 416.00  
## 24 5 4 302.80  
## 25 6 6 584.80  
## 26 6 6 151.90  
## 27 6 6 159.60  
## 28 6 4 156.40  
## 29 4 1 227.10  
## 30 8 8 322.00  
## 31 6 6 140.20  
## 32 5 4 132.60  
## 33 4 6 223.70  
## 34 4 4 229.60  
## 35 6 8 304.60  
## 36 4 4 80.50  
## 37 6 6 98.50  
## 38 4 4 265.90  
## 39 6 6 119.10  
## 40 4 5 214.50  
## 41 6 8 310.00  
## 42 12 10 284.00  
## 43 10 5 522.30  
## 44 3 6 327.10  
## 45 10 8 526.80  
## 46 6 8 938.20  
## 47 8 4 229.70  
## 48 8 8 217.10  
## 49 6 8 562.80  
## 50 4 4 256.10  
## 51 10 4 168.20  
## 52 6 3 67.80  
## 53 8 8 333.60  
## 54 6 6 203.60  
## 55 6 10 282.90  
## 56 4 4 234.80  
## 57 6 6 58.60  
## 58 6 6 174.00  
## 59 6 4 505.80  
## 60 4 4 238.50  
## 61 4 4 370.70  
## 62 6 4 143.30  
## 63 3 4 209.70  
## 64 4 4 310.00  
## 65 6 6 179.10  
## 66 4 6 429.40  
## 67 5 4 173.40  
## 68 6 4 130.20  
## 69 6 8 361.50  
## 70 12 4 102.40  
## 71 8 8 116.30  
## 72 12 6 105.20  
## 73 6 6 234.50  
## 74 4 18 156.30  
## 75 10 10 61.50  
## 76 6 6 128.60  
## 77 3 6 388.30  
## 78 6 6 405.10  
## 79 8 4 338.20  
## 80 5 4 230.40  
## 81 20 24 573.40  
## 82 6 6 71.50  
## 83 6 8 269.00  
## 84 6 6 236.00  
## 85 4 4 161.80  
## 86 4 8 238.20  
## 87 6 6 247.10  
## 88 8 10 205.90  
## 89 6 4 284.30  
## 90 4 2 248.60  
## 91 6 6 202.80  
## 92 6 8 295.20  
## 93 10 4 371.90  
## 94 6 8 116.50  
## 95 4 8 371.60  
## 96 4 6 221.10  
## 97 6 6 92.00  
## 98 4 4 242.60  
## 99 4 4 157.10  
## 100 3 4 33.90  
## 101 8 8 147.60  
## 102 8 8 336.70  
## 103 8 4 88.10  
## 104 6 6 153.40  
## 105 4 6 199.50  
## 106 4 4 239.70  
## 107 12 6 345.30  
## 108 4 4 314.60  
## 109 4 4 126.90  
## 110 4 4 212.80  
## 111 6 6 337.80  
## 112 8 6 189.90  
## 113 6 10 160.30  
## 114 2 2 158.20  
## 115 4 4 227.50  
## 116 6 4 297.40  
## 117 6 4 150.40  
## 118 6 6 47.20  
## 119 2 4 158.50  
## 120 6 6 224.40  
## 121 12 8 205.70  
## 122 8 30 720.00  
## 123 6 12 254.30  
## 124 4 4 126.40  
## 125 8 10 360.50  
## 126 24 16 375.60  
## 127 8 8 308.40  
## 128 10 8 415.10  
## 129 6 6 209.10  
## 130 6 6 144.20  
## 131 4 8 694.50  
## 132 6 6 418.90  
## 133 8 6 169.20  
## 134 8 10 84.30  
## 135 8 8 77.70  
## 136 6 4 24.80  
## 137 4 4 224.80  
## 138 6 6 455.80  
## 139 8 8 195.70  
## 140 6 4 263.60  
## 141 4 6 268.40  
## 142 6 6 530.30  
## CaloriesperServing2006 CaloricDensity1936 CaloricDensity2006  
## 1 130.00 2.41 2.15  
## 2 128.10 4.20 4.01  
## 3 90.30 2.55 2.28  
## 4 53.90 2.69 2.82  
## 5 62.00 4.19 4.45  
## 6 278.00 1.92 2.07  
## 7 NA 1.77 2.31  
## 8 NA 0.63 1.41  
## 9 NA 3.18 3.18  
## 10 NA 0.71 0.89  
## 11 293.70 1.83 1.99  
## 12 226.90 1.24 1.23  
## 13 157.60 1.28 1.48  
## 14 226.80 1.21 1.11  
## 15 486.50 1.39 0.98  
## 16 119.00 1.56 1.69  
## 17 304.10 1.76 1.24  
## 18 568.80 0.85 0.89  
## 19 212.30 1.20 1.19  
## 20 327.80 3.67 2.98  
## 21 187.20 1.04 1.23  
## 22 217.30 1.30 1.19  
## 23 574.00 2.56 3.03  
## 24 153.30 1.76 1.40  
## 25 363.90 2.80 2.10  
## 26 214.90 0.80 0.90  
## 27 159.60 0.70 0.70  
## 28 99.70 0.78 0.83  
## 29 88.70 2.50 1.60  
## 30 177.30 1.99 1.40  
## 31 202.80 0.97 1.70  
## 32 181.00 1.12 1.19  
## 33 105.80 0.70 0.70  
## 34 173.00 0.57 0.53  
## 35 317.00 1.47 1.65  
## 36 120.80 0.71 0.71  
## 37 144.70 0.74 1.09  
## 38 330.40 1.03 1.58  
## 39 237.70 1.12 1.14  
## 40 221.30 1.31 1.35  
## 41 170.30 1.43 1.28  
## 42 371.70 2.21 3.29  
## 43 443.60 3.84 2.83  
## 44 166.70 1.90 2.05  
## 45 518.15 1.81 1.41  
## 46 559.40 1.98 1.49  
## 47 271.70 3.35 2.48  
## 48 240.40 1.77 2.33  
## 49 379.11 2.89 3.06  
## 50 491.60 1.93 3.79  
## 51 302.70 1.17 1.15  
## 52 113.90 0.93 0.81  
## 53 164.30 2.16 1.84  
## 54 362.90 0.66 1.28  
## 55 247.60 3.15 3.42  
## 56 239.90 1.37 1.49  
## 57 74.50 0.40 0.38  
## 58 444.50 1.41 1.30  
## 59 346.30 2.24 1.83  
## 60 238.50 1.06 1.06  
## 61 267.70 1.18 1.45  
## 62 110.50 1.96 1.42  
## 63 358.00 1.32 2.13  
## 64 310.00 1.50 1.50  
## 65 146.40 1.44 1.38  
## 66 286.30 2.27 2.27  
## 67 327.50 1.19 1.49  
## 68 145.70 1.46 0.92  
## 69 273.50 2.35 1.57  
## 70 82.70 1.01 0.47  
## 71 116.80 1.34 1.35  
## 72 60.80 0.98 0.61  
## 73 193.30 0.87 0.89  
## 74 84.50 1.87 1.36  
## 75 50.50 0.74 0.59  
## 76 151.00 0.65 0.90  
## 77 197.40 1.85 1.97  
## 78 505.50 1.76 1.50  
## 79 212.90 1.55 1.83  
## 80 360.00 2.54 2.54  
## 81 195.90 1.57 1.82  
## 82 176.70 0.83 2.14  
## 83 227.60 1.28 1.34  
## 84 236.00 0.92 0.92  
## 85 155.20 0.79 1.04  
## 86 210.90 2.13 2.16  
## 87 244.30 1.57 1.57  
## 88 160.40 2.44 2.67  
## 89 426.40 2.69 2.69  
## 90 154.90 2.74 1.11  
## 91 257.40 1.08 1.36  
## 92 297.90 3.18 1.75  
## 93 508.20 1.27 2.22  
## 94 135.10 1.15 1.09  
## 95 273.30 1.12 0.83  
## 96 305.60 1.05 1.65  
## 97 92.00 0.74 0.74  
## 98 275.20 1.49 1.63  
## 99 146.80 0.83 0.92  
## 100 49.20 0.32 0.33  
## 101 140.20 0.73 0.73  
## 102 246.50 2.37 1.83  
## 103 120.50 0.73 0.71  
## 104 119.50 0.98 0.79  
## 105 266.00 1.29 1.29  
## 106 199.10 1.40 1.24  
## 107 716.40 1.91 2.36  
## 108 373.90 1.84 1.31  
## 109 105.90 1.15 0.78  
## 110 321.90 0.92 1.03  
## 111 622.70 2.42 2.26  
## 112 149.20 1.18 0.78  
## 113 154.00 1.82 0.80  
## 114 183.60 1.93 2.14  
## 115 189.60 2.61 1.86  
## 116 646.60 1.53 1.60  
## 117 236.80 1.36 1.04  
## 118 79.50 0.55 0.82  
## 119 133.50 1.58 2.12  
## 120 288.70 0.79 0.79  
## 121 308.80 1.78 1.60  
## 122 576.00 2.12 2.12  
## 123 348.80 2.40 4.03  
## 124 183.40 0.79 1.01  
## 125 339.70 3.00 3.50  
## 126 395.70 4.11 3.52  
## 127 437.20 3.57 3.47  
## 128 361.80 1.08 1.12  
## 129 227.60 1.03 1.09  
## 130 141.50 1.13 1.20  
## 131 457.50 1.44 1.39  
## 132 370.60 1.59 1.57  
## 133 127.30 1.09 1.06  
## 134 55.90 0.48 0.36  
## 135 78.20 0.74 0.71  
## 136 28.80 0.15 0.18  
## 137 131.00 2.46 1.69  
## 138 361.60 1.36 1.13  
## 139 226.50 1.51 2.93  
## 140 301.50 3.64 3.18  
## 141 341.60 2.98 2.88  
## 142 793.20 0.80 0.96

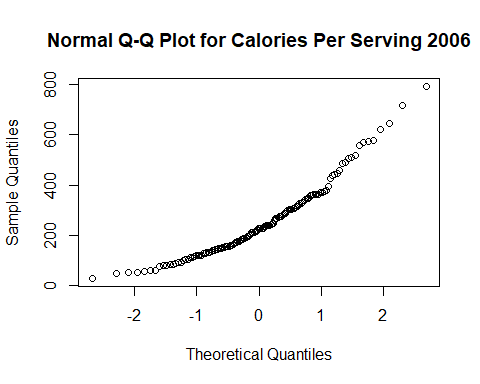
#This section will present a histogram of the JoyOfCooking data (Calories Per Serving) that was recently imported.  
  
hist(JoyOfCooking$CaloriesperServing2006,main = paste("Histogram of Calories Per Serving 2006"),xlab = "Calories Per Serving")



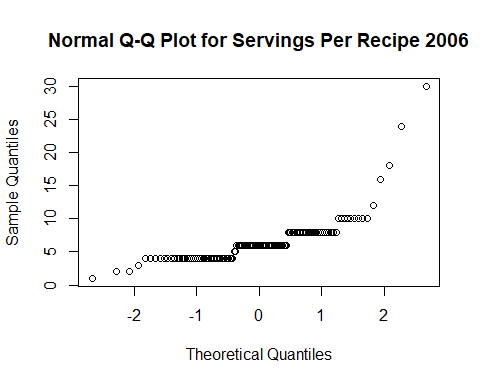
#This section will present a histogram of the JoyOfCooking data (Servings Per Recipe) that was recently imported.  
  
hist(JoyOfCooking$ServingsperRecipe2006,main = paste("Histogram of Servings Per Recipe 2006"),xlab = "Servings Per Recipe")



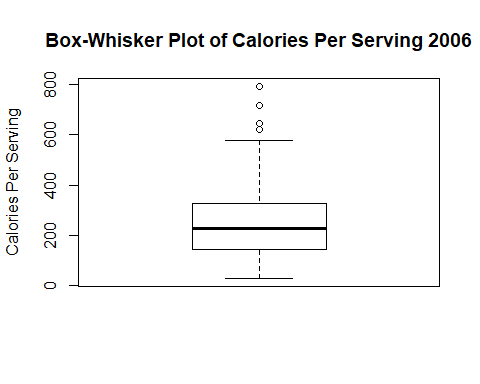
#This section will present a Q-Q Norm graph of the JoyOfCooking data (Calories Per Serving) that was recently imported.  
  
qqnorm(JoyOfCooking$CaloriesperServing2006,main = "Normal Q-Q Plot for Calories Per Serving 2006")



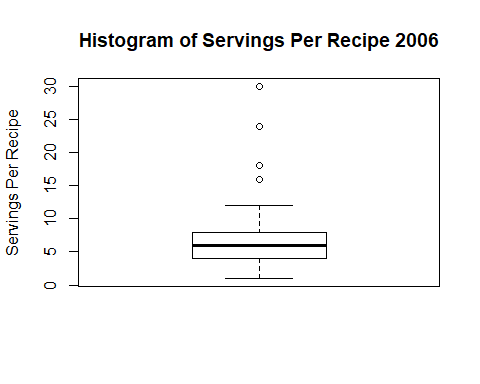
#This section will present a Q-Q Norm graph of the JoyOfCooking data (Calories Per Serving) that was recently imported.  
  
qqnorm(JoyOfCooking$ServingsperRecipe2006,main = "Normal Q-Q Plot for Servings Per Recipe 2006")



#This section will present a box-whisker plot of the JoyOfCooking data (Calories Per Serving) that was recently imported.  
  
boxplot(JoyOfCooking$CaloriesperServing2006,main = paste("Box-Whisker Plot of Calories Per Serving 2006"),ylab = "Calories Per Serving")



#This section will present a box-whisker plot of the JoyOfCooking data (Servings Per Recipe) that was recently imported.  
  
boxplot(JoyOfCooking$ServingsperRecipe2006,main = paste("Histogram of Servings Per Recipe 2006"),ylab = "Servings Per Recipe")



# Exercise 2.

Review Exercise 3, Homework 6, where you calculated skewness and kurtosis. The reference for this exercise, <https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm>, gives four example statistical distributions. We will reproduce the histograms, and add qqnorm and box-whisker plots.

## Part a

Use the code below from lecture to draw 1000 samples from the normal distribution.

norm.sample <- rnorm(1000, mean=0, sd=1)

Look up the corresponding r\* functions in R for the Cauchy distribution (use location=0, scale=1), and the Weibull distribution (use shape = 1.5). For the double exponential, use you can use the \*laplace functions from the rmutil library, or you can use rexp(1000) - rexp(1000)

Draw 1000 samples from each of these distributions. Calculate skewness and kurtosis for each sample. You can use your own functions, or the moments library.

## Part b

Plot the histograms for each distribution. Use par(mfrow=c(2,2)) in your code chunk to combine the four histogram in a single plot. Add titles to the histograms indicating the distribution. Set the x-axis label to show the calculated skewness and kurtosis, i.e. skewness = ####, kurtosis = ####

## Part c

Repeat Part b, but with QQ-norm plots.

## Part d

Repeat Part b, but with box-whisker plots.

Hints for SAS. If you create the samples in IML, use

Normal = j(1, 1000, .);  
call randgen(Normal, "NORMAL", 0, `);

You can generate samples in the data step using

do i = 1 to 1000;  
 Normal = rand('NORMAL',0,1);  
 output;  
end;

RAND doesn’t provide a Laplace option, but you can create samples from this distribution by

rand('EXPONENTIAL')-rand('EXPONENTIAL');

To group multiple plots, use

ods graphics / width=8cm height=8cm;  
ods layout gridded columns=2;  
ods region;  
 ... first plot  
  
ods region;  
 ... second plot  
  
ods layout end;

You might need to include

ods graphics off;  
  
ods graphics on;  
ODS GRAPHICS / reset=All;

to return the SAS graphics output to normal.

## Exercise 3.

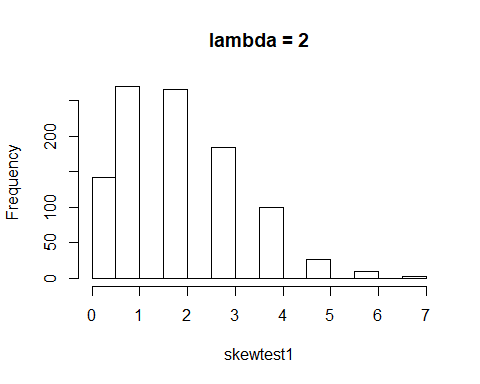
Iterate over a sequence of , from 2 to 64, doubling each time, and draw 1000 samples from the Poisson distribution for each .

Calculate the skewness for each of samples, and plot histograms, QQ-norm and box-whisker plots for each. You can use par(mfrow=c(1,3)) to display all three graphs for one in one line. Add lambda=## to the title of the histogram, and skewness=## to the title of the box-whisker plot.

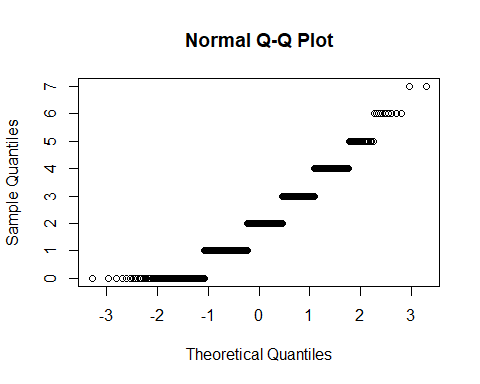
#This section will iterate a sequence from 2 to 64, which will result in a sequence of 6 integers: 2,4,8,16,32,64  
  
# This will prevent the use of scientific notation in the output  
options(scipen=999)  
  
#The sequence is being generated and assigned to "mu"  
mu.sequence <- c(1,2,3,4,5,6)  
mu <- 2^mu.sequence  
  
mu

## [1] 2 4 8 16 32 64

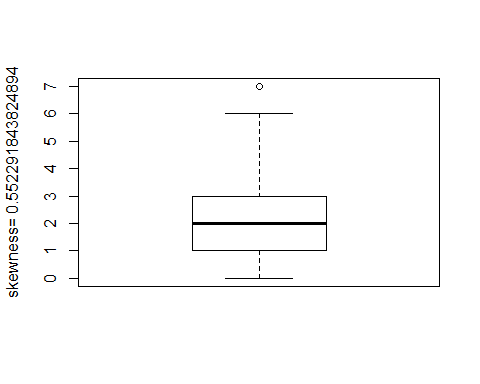
#This section will calculate the skewness, plot histogram, QQ-norm, and box-whisker plots for the 1st sample  
  
#This loads the library "moments" so the skewness function can be used.  
library(moments)  
  
#The number of sample values for use with the Poisson distribution is being assigned to "N";The sample values along with the sequence "mu" will be input into the rpois() function  
N <- 1000  
  
set.seed(123456)  
skewtest1 <- rpois(N,mu[1])  
skew1 <- skewness(skewtest1)  
  
#These are the 3 plots being generated using the previously generated data.  
hist(skewtest1, main = "lambda = 2")



qqnorm(skewtest1)

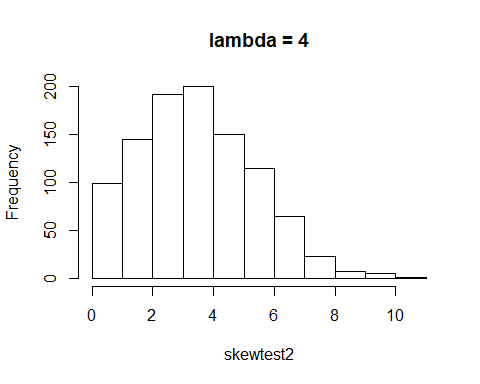


boxplot(skewtest1,ylab = paste("skewness=",skew1))

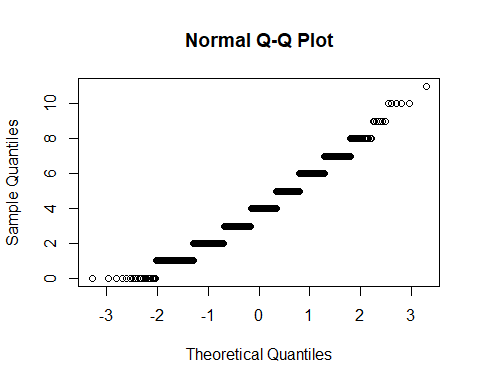


par(mfrow=c(1,3))

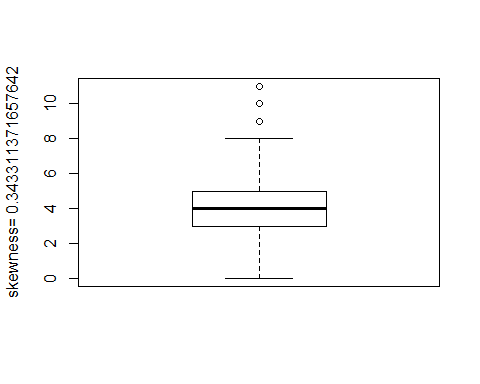
#This section will calculate the skewness, plot histogram, QQ-norm, and box-whisker plots for the 2nd sample  
  
#This loads the library "moments" so the skewness function can be used.  
library(moments)  
  
#The number of sample values for use with the Poisson distribution is being assigned to "N";The sample values along with the sequence "mu" will be input into the rpois() function  
N <- 1000  
  
set.seed(123456)  
skewtest2 <- rpois(N,mu[2])  
skew2 <- skewness(skewtest2)  
  
#These are the 3 plots being generated using the previously generated data.  
hist(skewtest2, main = "lambda = 4")



qqnorm(skewtest2)

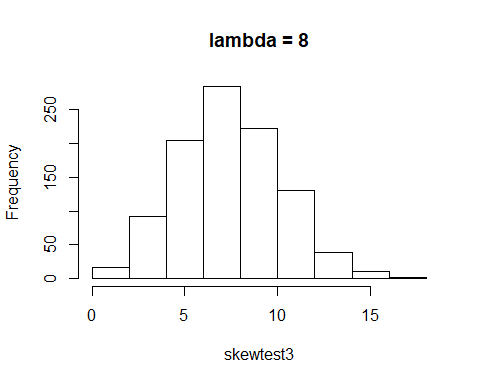


boxplot(skewtest2,ylab = paste("skewness=",skew2))

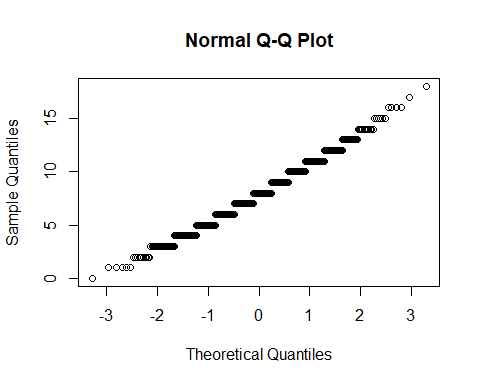


par(mfrow=c(1,3))

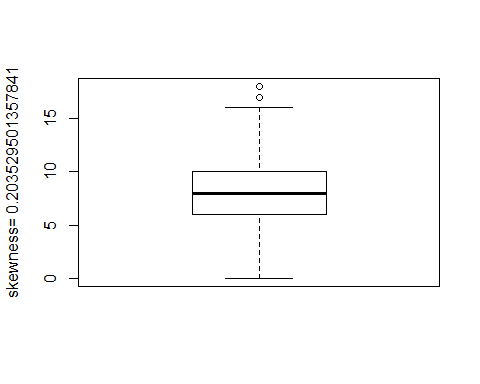
#This section will calculate the skewness, plot histogram, QQ-norm, and box-whisker plots for the 3rd sample  
  
#This loads the library "moments" so the skewness function can be used.  
library(moments)  
  
#The number of sample values for use with the Poisson distribution is being assigned to "N";The sample values along with the sequence "mu" will be input into the rpois() function  
N <- 1000  
  
set.seed(123456)  
skewtest3 <- rpois(N,mu[3])  
skew3 <- skewness(skewtest3)  
  
#These are the 3 plots being generated using the previously generated data.  
hist(skewtest3, main = "lambda = 8")



qqnorm(skewtest3)

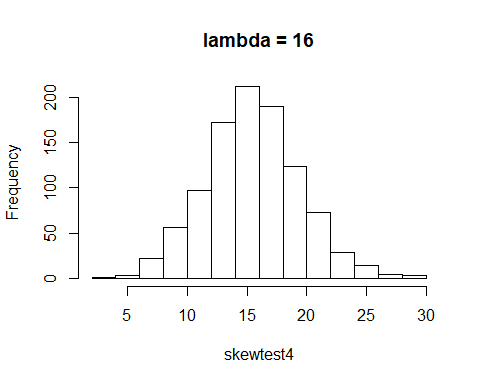


boxplot(skewtest3,ylab = paste("skewness=",skew3))

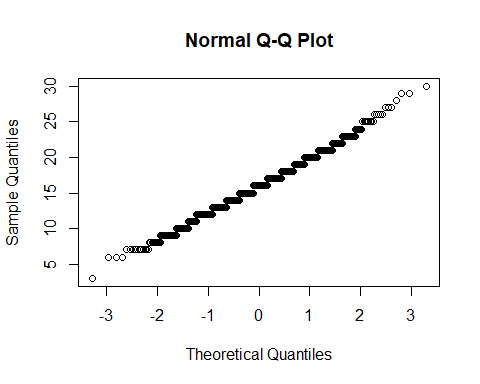


par(mfrow=c(1,3))

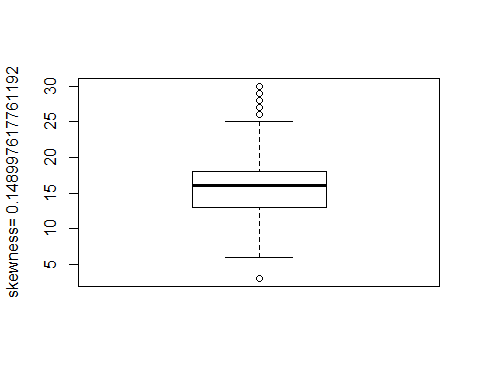
#This section will calculate the skewness, plot histogram, QQ-norm, and box-whisker plots for the 4th sample  
  
#This loads the library "moments" so the skewness function can be used.  
library(moments)  
  
#The number of sample values for use with the Poisson distribution is being assigned to "N";The sample values along with the sequence "mu" will be input into the rpois() function  
N <- 1000  
  
set.seed(123456)  
skewtest4 <- rpois(N,mu[4])  
skew4 <- skewness(skewtest4)  
  
#These are the 3 plots being generated using the previously generated data.  
hist(skewtest4, main = "lambda = 16")



qqnorm(skewtest4)

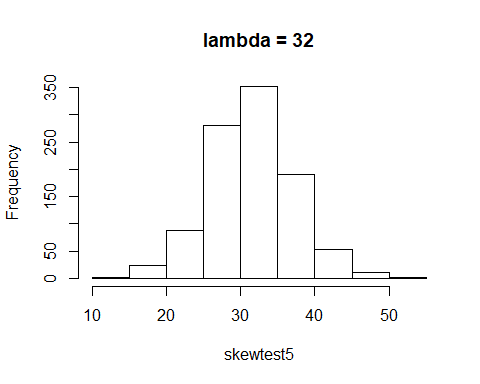


boxplot(skewtest4,ylab = paste("skewness=",skew4))

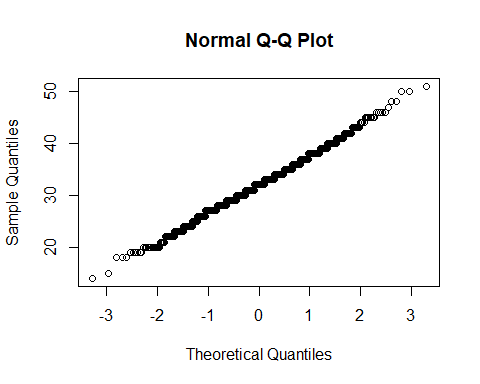


par(mfrow=c(1,3))

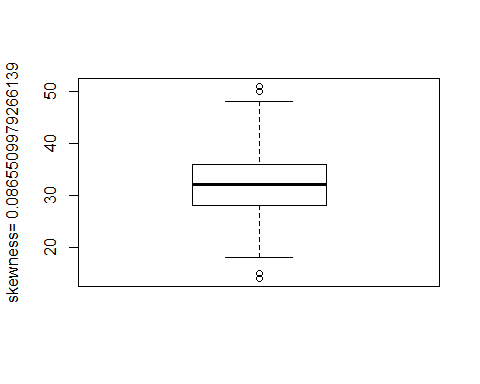
#This section will calculate the skewness, plot histogram, QQ-norm, and box-whisker plots for the 5th sample  
  
#This loads the library "moments" so the skewness function can be used.  
library(moments)  
  
#The number of sample values for use with the Poisson distribution is being assigned to "N";The sample values along with the sequence "mu" will be input into the rpois() function  
N <- 1000  
  
set.seed(123456)  
skewtest5 <- rpois(N,mu[5])  
skew5 <- skewness(skewtest5)  
  
#These are the 3 plots being generated using previously generated data.  
hist(skewtest5, main = "lambda = 32")



qqnorm(skewtest5)

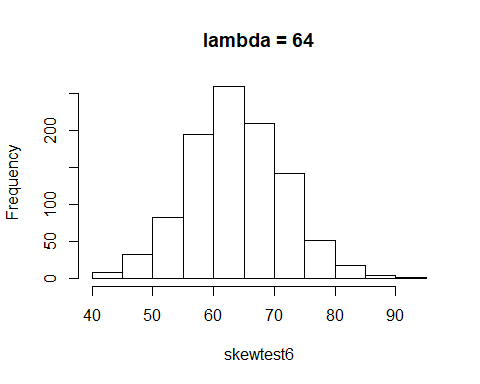


boxplot(skewtest5,ylab = paste("skewness=",skew5))

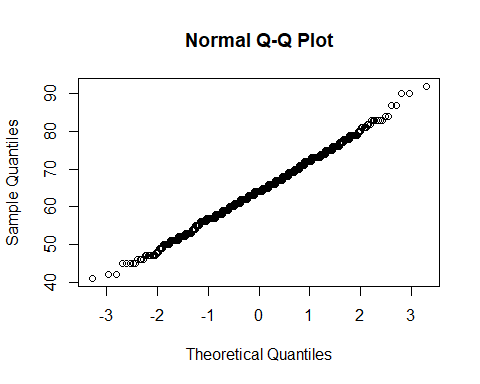


par(mfrow=c(1,3))

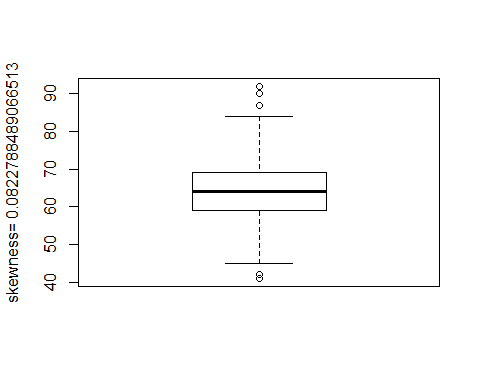
#This section will calculate the skewness, plot histogram, QQ-norm, and box-whisker plots for the 6th sample  
  
#This loads the library "moments" so the skewness function can be used.  
library(moments)  
  
#The number of sample values for use with the Poisson distribution is being assigned to "N";The sample values along with the sequence "mu" will be input into the rpois() function  
N <- 1000  
  
set.seed(123456)  
skewtest6 <- rpois(N,mu[6])  
skew6 <- skewness(skewtest6)  
  
#These are the 3 plots being generated using previously generated data.  
hist(skewtest6, main = "lambda = 64")



qqnorm(skewtest6)



boxplot(skewtest6,ylab = paste("skewness=",skew6))



par(mfrow=c(1,3))

## Part b.

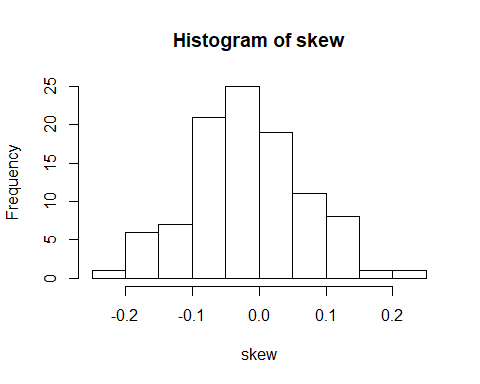
Remember that lambda represents the mean of a discrete (counting) variable. At what size mean is Poisson data no longer skewed, relative to normally distributed data? You might run this 2 or 3 times, with different seeds; this number varies, in my experience.

If you do this in SAS, create a data table with data columns each representing a different . You can see combined histogram, box-whisker and QQ-norm, for all columns, by calling

proc univariate data=Distributions plot;  
run;

At what is skewness of the Poisson distribution small enough to be considered normal?

skew <- 1:100  
for(i in 1:100) {  
 norm.sample <- rnorm(1000, mean=0, sd=1)  
 skew[i] <- skewness(norm.sample)  
}  
hist(skew)



summary(skew)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -0.23914 -0.06420 -0.02493 -0.01712 0.03951 0.20650

skew1 <- skewness(skewtest1)  
skew2 <- skewness(skewtest2)  
skew3 <- skewness(skewtest3)  
skew4 <- skewness(skewtest4)  
skew5 <- skewness(skewtest5)  
skew6 <- skewness(skewtest6)  
  
skew1

## [1] 0.5522918

skew2

## [1] 0.3433114

skew3

## [1] 0.2035295

skew4

## [1] 0.1489976

skew5

## [1] 0.086551

skew6

## [1] 0.08227885

#The output of "skew5" is 0.086551, which is less than 0.1. In this case, that would make it the mu that would be considered normal.

# Exercise 4

## Part a

Write a function that accepts

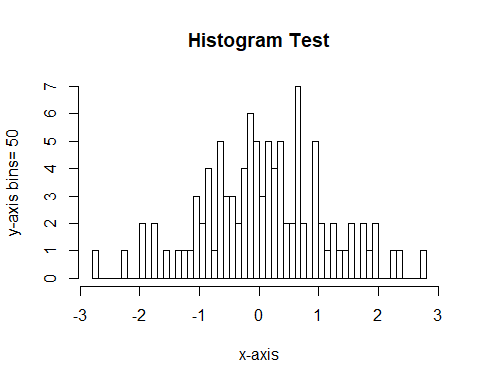
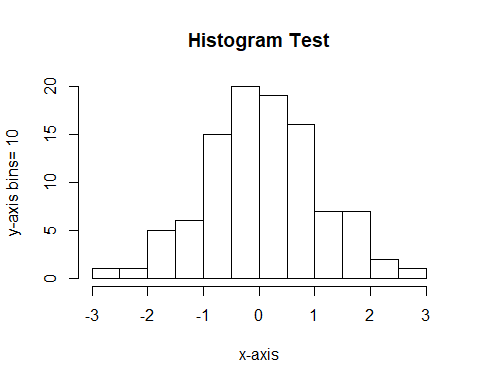
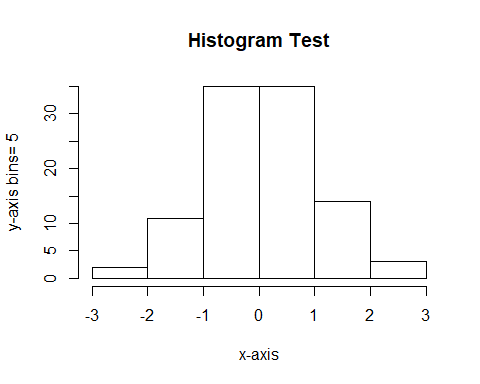
1. a vector sample of data
2. a vector of integers
3. a main title
4. an x axis label.

This function should

1. iterate over each element in the vector of integers
2. produce a histogram for sample setting the number of bins in the histogram to
3. label main and x-axis with the specified parameters.
4. label the y-axis to read Frequency, bins = and the number of bins.

Hint: You can simplify this function by using the parameter ... - see ?plot or ?hist

# testv <- c(1,2,3,4,5,6,7,8,9,10)  
  
#Generating test sequence  
testv <- rnorm(100)  
  
#Vector of test bins  
testbins <- c(5,10,50)  
  
#Creating function and running test case  
func <- function(sampv,bins,mainlabel,xlabel,ylabel) {  
 v <- c()  
 len <- length(sampv)  
 len1 <- length(bins)  
   
 #iterate the number of entries from vector  
 for(i in 1:len)  
 v[i] <- sampv[i]  
   
 for (k in 1:len1){  
 hists <- hist(v,breaks = bins[k],main = mainlabel,xlab = xlabel,ylab = paste(ylabel, bins[k]))  
 }  
  
 return(hists)  
}  
func(testv,testbins,"Histogram Test","x-axis","y-axis bins=")



## $breaks  
## [1] -2.8 -2.7 -2.6 -2.5 -2.4 -2.3 -2.2 -2.1 -2.0 -1.9 -1.8 -1.7 -1.6 -1.5  
## [15] -1.4 -1.3 -1.2 -1.1 -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2 -0.1  
## [29] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3  
## [43] 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7  
## [57] 2.8  
##   
## $counts  
## [1] 1 0 0 0 0 1 0 0 2 0 2 0 1 0 1 1 1 3 2 4 1 5 3 3 2 4 6 5 3 5 4 5 2 2 7  
## [36] 2 0 5 2 1 2 1 1 2 0 2 1 2 0 0 1 1 0 0 0 1  
##   
## $density  
## [1] 0.1 0.0 0.0 0.0 0.0 0.1 0.0 0.0 0.2 0.0 0.2 0.0 0.1 0.0 0.1 0.1 0.1  
## [18] 0.3 0.2 0.4 0.1 0.5 0.3 0.3 0.2 0.4 0.6 0.5 0.3 0.5 0.4 0.5 0.2 0.2  
## [35] 0.7 0.2 0.0 0.5 0.2 0.1 0.2 0.1 0.1 0.2 0.0 0.2 0.1 0.2 0.0 0.0 0.1  
## [52] 0.1 0.0 0.0 0.0 0.1  
##   
## $mids  
## [1] -2.75 -2.65 -2.55 -2.45 -2.35 -2.25 -2.15 -2.05 -1.95 -1.85 -1.75  
## [12] -1.65 -1.55 -1.45 -1.35 -1.25 -1.15 -1.05 -0.95 -0.85 -0.75 -0.65  
## [23] -0.55 -0.45 -0.35 -0.25 -0.15 -0.05 0.05 0.15 0.25 0.35 0.45  
## [34] 0.55 0.65 0.75 0.85 0.95 1.05 1.15 1.25 1.35 1.45 1.55  
## [45] 1.65 1.75 1.85 1.95 2.05 2.15 2.25 2.35 2.45 2.55 2.65  
## [56] 2.75  
##   
## $xname  
## [1] "v"  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"

## Part b

Test your function with the hidalgo data set (see below), using bin numbers 12, 36, and 60. You should be able to call your function with something like

plot.histograms(hidalgo.dat[,1],c(12,36,60), main="1872 Hidalgo issue",xlab= "Thickness (mm)")

to plot three different histograms of the hidalgo data set.

If you do this in SAS, write a macro that accepts a table name, a column name, a list of integers, a main axis label and an x axis label. This macro should scan over each element in the list of integers and produce a histogram for each integer value, setting the bin count to the element in the input list, and labeling main and x-axis with the specified parameters. You should label the y-axis to read Frequency, bins = and the number of bins.

Test your macro with the hidalgo data set (see below), using bin numbers 12, 36, and 60. You should be able to call your macro with something like

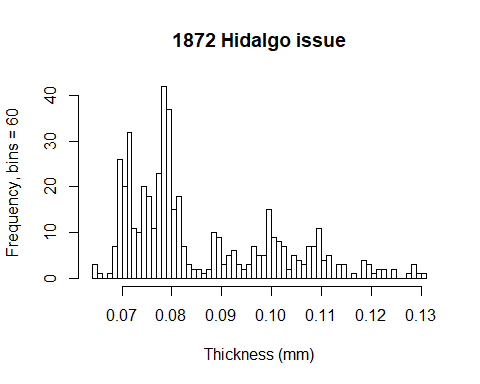
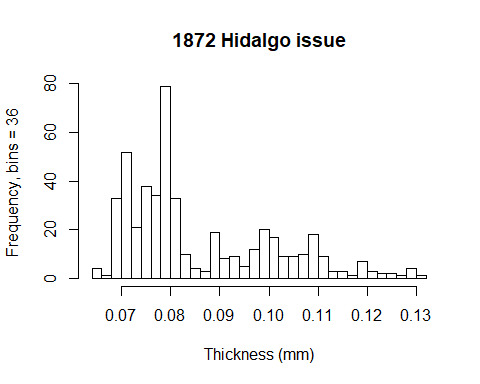
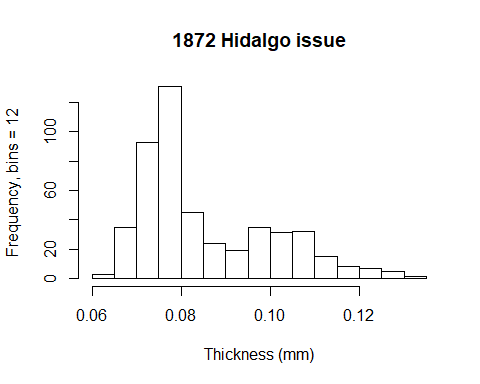
%plot\_histograms(hidalgo, y, 12 36 60, main="1872 Hidalgo issue", xlabel="Thickness (mm)");

to plot three different histograms of the hidalgo data set.

Hint: Assume 12 36 60 resolve to a single macro parameter and use %scan. Your macro definition can look something like

%macro plot\_histograms(table\_name, column\_name, number\_of\_bins, main="Main", xlabel="X Label")

#Path information for reading file from local machine  
PathTohidalgo = "C:/Users/drewm/Documents/GitHub/code-stat700/hidalgo.dat"  
  
#Assigning data from file to data frame  
hidalgo <- read.delim(PathTohidalgo,header=TRUE,skip= 0,sep = "",as.is=TRUE)  
  
#Creating function and running requested values  
hidalgo <- as.numeric(hidalgo$X.060)  
bins <- c(12,36,60)  
  
func <- function(sampv,bins,mainlabel,xlabel,ylabel) {  
 v <- c()  
 len <- length(sampv)  
 len1 <- length(bins)  
   
 #Iterate the number of entries from vector  
 for(i in 1:len)  
 v[i] <- sampv[i]  
   
 for (k in 1:length(bins)){  
 hists <- hist(v,breaks = bins[k],main = mainlabel,xlab = xlabel,ylab = paste(ylabel, bins[k]))  
 }  
  
 return(hists)  
}  
func(hidalgo,bins,"1872 Hidalgo issue","Thickness (mm)","Frequency, bins =")



## $breaks  
## [1] 0.064 0.065 0.066 0.067 0.068 0.069 0.070 0.071 0.072 0.073 0.074  
## [12] 0.075 0.076 0.077 0.078 0.079 0.080 0.081 0.082 0.083 0.084 0.085  
## [23] 0.086 0.087 0.088 0.089 0.090 0.091 0.092 0.093 0.094 0.095 0.096  
## [34] 0.097 0.098 0.099 0.100 0.101 0.102 0.103 0.104 0.105 0.106 0.107  
## [45] 0.108 0.109 0.110 0.111 0.112 0.113 0.114 0.115 0.116 0.117 0.118  
## [56] 0.119 0.120 0.121 0.122 0.123 0.124 0.125 0.126 0.127 0.128 0.129  
## [67] 0.130 0.131  
##   
## $counts  
## [1] 3 1 0 1 7 26 20 32 11 10 20 18 11 23 42 37 15 18 7 3 2 2 1  
## [24] 2 10 9 3 5 6 3 2 3 7 5 5 15 9 8 7 2 5 4 3 7 7 11  
## [47] 4 5 0 3 3 0 1 0 4 3 1 2 2 0 2 0 0 1 3 1 1  
##   
## $density  
## [1] 6.198347 2.066116 0.000000 2.066116 14.462810 53.719008 41.322314  
## [8] 66.115702 22.727273 20.661157 41.322314 37.190083 22.727273 47.520661  
## [15] 86.776860 76.446281 30.991736 37.190083 14.462810 6.198347 4.132231  
## [22] 4.132231 2.066116 4.132231 20.661157 18.595041 6.198347 10.330579  
## [29] 12.396694 6.198347 4.132231 6.198347 14.462810 10.330579 10.330579  
## [36] 30.991736 18.595041 16.528926 14.462810 4.132231 10.330579 8.264463  
## [43] 6.198347 14.462810 14.462810 22.727273 8.264463 10.330579 0.000000  
## [50] 6.198347 6.198347 0.000000 2.066116 0.000000 8.264463 6.198347  
## [57] 2.066116 4.132231 4.132231 0.000000 4.132231 0.000000 0.000000  
## [64] 2.066116 6.198347 2.066116 2.066116  
##   
## $mids  
## [1] 0.0645 0.0655 0.0665 0.0675 0.0685 0.0695 0.0705 0.0715 0.0725 0.0735  
## [11] 0.0745 0.0755 0.0765 0.0775 0.0785 0.0795 0.0805 0.0815 0.0825 0.0835  
## [21] 0.0845 0.0855 0.0865 0.0875 0.0885 0.0895 0.0905 0.0915 0.0925 0.0935  
## [31] 0.0945 0.0955 0.0965 0.0975 0.0985 0.0995 0.1005 0.1015 0.1025 0.1035  
## [41] 0.1045 0.1055 0.1065 0.1075 0.1085 0.1095 0.1105 0.1115 0.1125 0.1135  
## [51] 0.1145 0.1155 0.1165 0.1175 0.1185 0.1195 0.1205 0.1215 0.1225 0.1235  
## [61] 0.1245 0.1255 0.1265 0.1275 0.1285 0.1295 0.1305  
##   
## $xname  
## [1] "v"  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"

## Data

The hidalgo data set is in the file hidalgo.dat These data consist of paper thickness measurements of stamps from the 1872 Hidalgo issue of Mexico. This data set is commonly used to illustrate methods of determining the number of components in a mixture (in this case, different batches of paper). See <https://www.jstor.org/stable/2290118>,  
<https://books.google.com/books?id=1CuznRORa3EC&lpg=PA95&pg=PA94#v=onepage&q&f=false> and <https://books.google.com/books?id=c2_fAox0DQoC&pg=PA180&lpg=PA180&f=false> .

Some analysis suggest there are three different mixtures of paper used to produce the 1872 Hidalgo issue; other analysis suggest 3. Why do you think there might be disagreement about the number of mixtures?

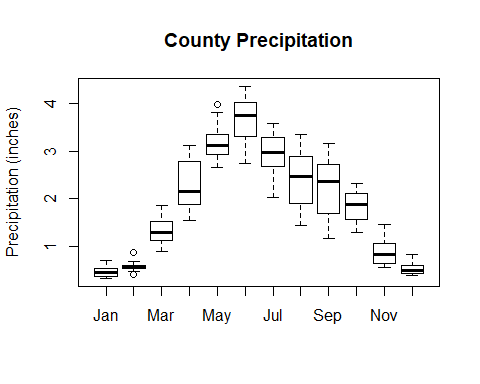
# Exercise 5.

Use countyPrecipitation and countyTemperature data. These are county-level averages of precipitation and temperature, by month, for the state of South Dakota. Plot a summary of precipitation and temperature by month, using box-whisker plots, one plot for each data set.

You might need to reshape the data. In the plot, the data are to be ordered chronologically by month, and the x-axis tick labels should show the abbreviations of the month (“Jan”, “Feb”, etc). You can create an index for month to get the order correct, but you will need to set the axis tick labels.

Label the y-axis “Temperature (degrees F)” or “Precipitation (inches)”

#Path information for reading file from local machine  
PathToPrecip = "C:/Users/drewm/Documents/GitHub/code-stat700/countyPrecipitation.tab"  
  
#Path information for reading file from local machine  
PathToTemp = "C:/Users/drewm/Documents/GitHub/code-stat700/countyTemperature.tab"  
  
#Assigning data from file to data frame  
cPrecip <- read.delim(PathToPrecip,header=TRUE,skip= 0,sep = "\t",as.is=TRUE)  
  
#Assigning data from file to data frame  
cTemp <- read.delim(PathToTemp,header=TRUE,skip= 0,sep = "\t",as.is=TRUE)  
  
#This will remove the County names from the data frame to make plotting the other data easier.  
cPrecip <- cPrecip[,-1]  
cTemp <- cTemp[,-1]  
  
  
  
# #This section will present a box-whisker plot of the County Precipatation data that was recently imported.  
  
boxplot(cPrecip,main = paste("County Precipitation"),ylab = "Precipitation (inches)")



# #This section will present a box-whisker plot of the JoyOfCooking data (Servings Per Recipe) that was recently imported.  
  
boxplot(cTemp,main = paste("County Temperature"),ylab = "Temperature (degrees F)")

