Jacob Dineen

**Unexcuted**

**#**

**#HW 3 IST 687**

**#**

**#HW 3 IST 687**

**#Step 1**

**rm(list = ls(all = TRUE))**

**urltoread <- "http://www2.census.gov/programs-surveys/popest/tables/2010-2011/state/totals/nst-est2011-01.csv" #URL to Read Vector**

**readStates <- read.csv(url(urltoread)) #Creating Vector for Census CSV DF**

**data.frame (readStates) #Call DF of readStates**

**str (readStates) #Structure of readStates**

**#Step2**

**readStates <- readStates [-1:-8,] #Remove first 8 Rows**

**summary (readStates [,6:10]) #Summarize Columns 6:10 (all null values)**

**readStates <- readStates [,-6:-10] #Remove columns 6:10**

**tail (readStates, 8) #Call bottom portion of DF for Nulls**

**readStates <- readStates [-52:-58,] #Remove Last 7 Rows (PuertoRico and Census notes)**

**readStates$stateName <- readStates [,1] #Rename Col 1 to stateName**

**readStates$base2010 <- readStates [,2] #Rename Col 2 to Base2010**

**readStates$base2011 <- readStates [,3] #Rename Col 3 to Base2011**

**readStates$Jul2010 <- readStates [,4] #Rename Col 3 to Jul2010**

**readStates$Jul2011 <- readStates [,5] #Rename Col 4 to Jul2011**

**readStates <- readStates [,-1:-5] #Remove columns 1:5 (legacy, bad names)**

**readStates$stateName <- gsub ("\\.", "", readStates$stateName) #Remove dots from stateName**

**#Removing commas from character string columns with population data**

**readStates$base2010 <-gsub(",", "", readStates$base2010)**

**readStates$base2011 <-gsub(",", "", readStates$base2011)**

**readStates$Jul2010 <-gsub(",", "", readStates$Jul2010)**

**readStates$Jul2011 <-gsub(",", "", readStates$Jul2011)**

**#Removing spaces and converting to numeric values**

**readStates$base2010 <- as.numeric (gsub(" ", "", readStates$base2010))**

**readStates$base2011 <- as.numeric (gsub(" ", "", readStates$base2011))**

**readStates$Jul2010 <- as.numeric (gsub(" ", "", readStates$Jul2010))**

**readStates$Jul2011 <- as.numeric (gsub(" ", "", readStates$Jul2011))**

**rownames (readStates) <- NULL #Normalize Row names**

**head (readStates, 5) #RowName Check**

**#Step3**

**dfStates <- data.frame(readStates) #readStates vector stored as data.frame "dfStates"**

**dfStates #call dfStates data.frame**

**mean(dfStates$Jul2011)**

**#Step4**

**which.max(dfStates$Jul2011) #Call maxpop index of July 2011 Column**

**dfStates [5,5] #Call dataframe for maxpop Jul2011 index**

**dfStates [5,1] #Call dataframe for stateName Jul2011 index**

**dfStates [order (dfStates$Jul2011),] #Sort by Max Jul2011 Population in ASC order**

**#Step5**

**#Write function that will return the percentage of the elements within the vector that is less than the same**

**v <- dfStates$Jul2011 #Store Vector for col "Jul2011"**

**mean <- mean(dfStates$Jul2011) #Store Vector for mean Jul2011 Pop**

**####Version 1**

**function1 <- function(v) {**

**length(which(v<mean))/ length(v) #Length of which states less than mean/length of states**

**paste(round(100\* length(which(v<mean))/ length(v), 2), "%", sep = "") #Convert to percentage format**

**meanDfStates <- paste(round(100\* length(which(v<mean))/ length(v), 2), "%", sep = "")**

**return(meanDfStates)**

**}**

**function1 (dfStates$Jul2011) #Call Function for col "Jul2011"**

**###OR Version 2**

**function2 <- function(v) {**

**sum(v<mean, na.rm = TRUE)/ length(v) #Sum of Trues less than mean/ length of column**

**paste(round(100\* sum(v<mean, na.rm = TRUE)/ length(dfStates$Jul2011), 2), "%", sep = "") #Convert to percentage format**

**}**

**function2 (dfStates$Jul2011) #Call Function for col "Jul2011"**

**Executed:**

> #

> #HW 3 IST 687

>

>

> #Step 1

> rm(list = ls(all = TRUE))

> urltoread <- "http://www2.census.gov/programs-surveys/popest/tables/2010-2011/state/totals/nst-est2011-01.csv" #URL to Read Vector

> readStates <- read.csv(url(urltoread)) #Creating Vector for Census CSV DF

> data.frame (readStates) #Call DF of readStates

table.with.row.headers.in.column.A.and.column.headers.in.rows.3.through.4...leading.dots.indicate.sub.parts.

1 Table 1. Annual Estimates of the Population for the United States, Regions, States, and Puerto Rico: April 1, 2010 to July 1, 2011

2 Geographic Area

3

4 United States

5 Northeast

6 Midwest

7 South

8 West

9 .Alabama

10 .Alaska

11 .Arizona

12 .Arkansas

13 .California

14 .Colorado

15 .Connecticut

16 .Delaware

17 .District of Columbia

18 .Florida

19 .Georgia

20 .Hawaii

21 .Idaho

22 .Illinois

23 .Indiana

24 .Iowa

25 .Kansas

26 .Kentucky

27 .Louisiana

28 .Maine

29 .Maryland

30 .Massachusetts

31 .Michigan

32 .Minnesota

33 .Mississippi

34 .Missouri

35 .Montana

36 .Nebraska

37 .Nevada

38 .New Hampshire

39 .New Jersey

40 .New Mexico

41 .New York

42 .North Carolina

43 .North Dakota

44 .Ohio

45 .Oklahoma

46 .Oregon

47 .Pennsylvania

48 .Rhode Island

49 .South Carolina

50 .South Dakota

51 .Tennessee

52 .Texas

53 .Utah

54 .Vermont

55 .Virginia

56 .Washington

57 .West Virginia

58 .Wisconsin

59 .Wyoming

60

61 Puerto Rico

62 Note: The April 1, 2010 Population Estimates base reflects changes to the Census 2010 population from geographic program revisions. It does not reflect changes from the Count Question Resolution program. See Geographic Terms and Definitions at http://www.census.gov/popest/about/geo/terms.html for a list of the states that are included in each region.

63 Suggested Citation:

64 Table 1. Annual Estimates of the Population for the United States, Regions, States, and Puerto Rico: April 1, 2010 to July 1, 2011 (NST-EST2011-01)

65 Source: U.S. Census Bureau, Population Division

66 Release Date: December 2011

X X.1 X.2 X.3 X.4 X.5 X.6 X.7 X.8

1 NA NA NA NA NA

2 April 1, 2010 Population Estimates (as of July 1) NA NA NA NA NA

3 Census Estimates Base 2010 2011 NA NA NA NA NA

4 308,745,538 308,745,538 309,330,219 311,591,917 NA NA NA NA NA

5 55,317,240 55,317,244 55,366,108 55,521,598 NA NA NA NA NA

6 66,927,001 66,926,987 66,976,458 67,158,835 NA NA NA NA NA

7 114,555,744 114,555,757 114,857,529 116,046,736 NA NA NA NA NA

8 71,945,553 71,945,550 72,130,124 72,864,748 NA NA NA NA NA

9 4,779,736 4,779,735 4,785,401 4,802,740 NA NA NA NA NA

10 710,231 710,231 714,146 722,718 NA NA NA NA NA

11 6,392,017 6,392,013 6,413,158 6,482,505 NA NA NA NA NA

12 2,915,918 2,915,921 2,921,588 2,937,979 NA NA NA NA NA

13 37,253,956 37,253,956 37,338,198 37,691,912 NA NA NA NA NA

14 5,029,196 5,029,196 5,047,692 5,116,796 NA NA NA NA NA

15 3,574,097 3,574,097 3,575,498 3,580,709 NA NA NA NA NA

16 897,934 897,934 899,792 907,135 NA NA NA NA NA

17 601,723 601,723 604,912 617,996 NA NA NA NA NA

18 18,801,310 18,801,311 18,838,613 19,057,542 NA NA NA NA NA

19 9,687,653 9,687,660 9,712,157 9,815,210 NA NA NA NA NA

20 1,360,301 1,360,301 1,363,359 1,374,810 NA NA NA NA NA

21 1,567,582 1,567,582 1,571,102 1,584,985 NA NA NA NA NA

22 12,830,632 12,830,632 12,841,980 12,869,257 NA NA NA NA NA

23 6,483,802 6,483,800 6,490,622 6,516,922 NA NA NA NA NA

24 3,046,355 3,046,350 3,050,202 3,062,309 NA NA NA NA NA

25 2,853,118 2,853,118 2,859,143 2,871,238 NA NA NA NA NA

26 4,339,367 4,339,362 4,347,223 4,369,356 NA NA NA NA NA

27 4,533,372 4,533,372 4,545,343 4,574,836 NA NA NA NA NA

28 1,328,361 1,328,361 1,327,379 1,328,188 NA NA NA NA NA

29 5,773,552 5,773,552 5,785,681 5,828,289 NA NA NA NA NA

30 6,547,629 6,547,629 6,555,466 6,587,536 NA NA NA NA NA

31 9,883,640 9,883,635 9,877,143 9,876,187 NA NA NA NA NA

32 5,303,925 5,303,925 5,310,658 5,344,861 NA NA NA NA NA

33 2,967,297 2,967,297 2,970,072 2,978,512 NA NA NA NA NA

34 5,988,927 5,988,927 5,995,715 6,010,688 NA NA NA NA NA

35 989,415 989,415 990,958 998,199 NA NA NA NA NA

36 1,826,341 1,826,341 1,830,141 1,842,641 NA NA NA NA NA

37 2,700,551 2,700,551 2,704,283 2,723,322 NA NA NA NA NA

38 1,316,470 1,316,472 1,316,807 1,318,194 NA NA NA NA NA

39 8,791,894 8,791,894 8,799,593 8,821,155 NA NA NA NA NA

40 2,059,179 2,059,180 2,065,913 2,082,224 NA NA NA NA NA

41 19,378,102 19,378,104 19,395,206 19,465,197 NA NA NA NA NA

42 9,535,483 9,535,475 9,560,234 9,656,401 NA NA NA NA NA

43 672,591 672,591 674,629 683,932 NA NA NA NA NA

44 11,536,504 11,536,502 11,537,968 11,544,951 NA NA NA NA NA

45 3,751,351 3,751,354 3,760,184 3,791,508 NA NA NA NA NA

46 3,831,074 3,831,074 3,838,332 3,871,859 NA NA NA NA NA

47 12,702,379 12,702,379 12,717,722 12,742,886 NA NA NA NA NA

48 1,052,567 1,052,567 1,052,528 1,051,302 NA NA NA NA NA

49 4,625,364 4,625,364 4,637,106 4,679,230 NA NA NA NA NA

50 814,180 814,180 816,598 824,082 NA NA NA NA NA

51 6,346,105 6,346,110 6,357,436 6,403,353 NA NA NA NA NA

52 25,145,561 25,145,561 25,253,466 25,674,681 NA NA NA NA NA

53 2,763,885 2,763,885 2,775,479 2,817,222 NA NA NA NA NA

54 625,741 625,741 625,909 626,431 NA NA NA NA NA

55 8,001,024 8,001,030 8,023,953 8,096,604 NA NA NA NA NA

56 6,724,540 6,724,540 6,742,950 6,830,038 NA NA NA NA NA

57 1,852,994 1,852,996 1,854,368 1,855,364 NA NA NA NA NA

58 5,686,986 5,686,986 5,691,659 5,711,767 NA NA NA NA NA

59 563,626 563,626 564,554 568,158 NA NA NA NA NA

60 NA NA NA NA NA

61 3,725,789 3,725,789 3,721,978 3,706,690 NA NA NA NA NA

62 NA NA NA NA NA

63 NA NA NA NA NA

64 NA NA NA NA NA

65 NA NA NA NA NA

66 NA NA NA NA NA

> str (readStates) #Structure of readStates

'data.frame': 66 obs. of 10 variables:

$ table.with.row.headers.in.column.A.and.column.headers.in.rows.3.through.4...leading.dots.indicate.sub.parts.: Factor w/ 65 levels "",".Alabama",..: 62 53 1 64 55 54 60 65 2 3 ...

$ X : Factor w/ 60 levels "","1,052,567",..: 1 59 60 27 38 47 10 49 32 50 ...

$ X.1 : Factor w/ 59 levels "","1,052,567",..: 1 1 59 27 38 47 10 49 32 50 ...

$ X.2 : Factor w/ 60 levels "","1,052,528",..: 1 60 21 28 39 48 10 51 33 50 ...

$ X.3 : Factor w/ 59 levels "","1,051,302",..: 1 1 21 28 38 48 10 50 33 51 ...

$ X.4 : logi NA NA NA NA NA NA ...

$ X.5 : logi NA NA NA NA NA NA ...

$ X.6 : logi NA NA NA NA NA NA ...

$ X.7 : logi NA NA NA NA NA NA ...

$ X.8 : logi NA NA NA NA NA NA ...

>

> #Step2

> readStates <- readStates [-1:-8,] #Remove first 8 Rows

> summary (readStates [,6:10]) #Summarize Columns 6:10 (all null values)

X.4 X.5 X.6 X.7 X.8

Mode:logical Mode:logical Mode:logical Mode:logical Mode:logical

NA's:58 NA's:58 NA's:58 NA's:58 NA's:58

> readStates <- readStates [,-6:-10] #Remove columns 6:10

> tail (readStates, 8) #Call bottom portion of DF for Nulls

table.with.row.headers.in.column.A.and.column.headers.in.rows.3.through.4...leading.dots.indicate.sub.parts.

59 .Wyoming

60

61 Puerto Rico

62 Note: The April 1, 2010 Population Estimates base reflects changes to the Census 2010 population from geographic program revisions. It does not reflect changes from the Count Question Resolution program. See Geographic Terms and Definitions at http://www.census.gov/popest/about/geo/terms.html for a list of the states that are included in each region.

63 Suggested Citation:

64 Table 1. Annual Estimates of the Population for the United States, Regions, States, and Puerto Rico: April 1, 2010 to July 1, 2011 (NST-EST2011-01)

65 Source: U.S. Census Bureau, Population Division

66 Release Date: December 2011

X X.1 X.2 X.3

59 563,626 563,626 564,554 568,158

60

61 3,725,789 3,725,789 3,721,978 3,706,690

62

63

64

65

66

> readStates <- readStates [-52:-58,] #Remove Last 7 Rows (PuertoRico and Census notes)

> readStates$stateName <- readStates [,1] #Rename Col 1 to stateName

> readStates$base2010 <- readStates [,2] #Rename Col 2 to Base2010

> readStates$base2011 <- readStates [,3] #Rename Col 3 to Base2011

> readStates$Jul2010 <- readStates [,4] #Rename Col 3 to Jul2010

> readStates$Jul2011 <- readStates [,5] #Rename Col 4 to Jul2011

> readStates <- readStates [,-1:-5] #Remove columns 1:5 (legacy, bad names)

> readStates$stateName <- gsub ("\\.", "", readStates$stateName) #Remove dots from stateName

>

> #Removing commas from character string columns with population data

> readStates$base2010 <-gsub(",", "", readStates$base2010)

> readStates$base2011 <-gsub(",", "", readStates$base2011)

> readStates$Jul2010 <-gsub(",", "", readStates$Jul2010)

> readStates$Jul2011 <-gsub(",", "", readStates$Jul2011)

>

> #Removing spaces and converting to numeric values

> readStates$base2010 <- as.numeric (gsub(" ", "", readStates$base2010))

> readStates$base2011 <- as.numeric (gsub(" ", "", readStates$base2011))

> readStates$Jul2010 <- as.numeric (gsub(" ", "", readStates$Jul2010))

> readStates$Jul2011 <- as.numeric (gsub(" ", "", readStates$Jul2011))

> rownames (readStates) <- NULL #Normalize Row names

> head (readStates, 5) #RowName Check

stateName base2010 base2011 Jul2010 Jul2011

1 Alabama 4779736 4779735 4785401 4802740

2 Alaska 710231 710231 714146 722718

3 Arizona 6392017 6392013 6413158 6482505

4 Arkansas 2915918 2915921 2921588 2937979

5 California 37253956 37253956 37338198 37691912

>

> #Step3

> dfStates <- data.frame(readStates) #readStates vector stored as data.frame "dfStates"

> dfStates #call dfStates data.frame

stateName base2010 base2011 Jul2010 Jul2011

1 Alabama 4779736 4779735 4785401 4802740

2 Alaska 710231 710231 714146 722718

3 Arizona 6392017 6392013 6413158 6482505

4 Arkansas 2915918 2915921 2921588 2937979

5 California 37253956 37253956 37338198 37691912

6 Colorado 5029196 5029196 5047692 5116796

7 Connecticut 3574097 3574097 3575498 3580709

8 Delaware 897934 897934 899792 907135

9 District of Columbia 601723 601723 604912 617996

10 Florida 18801310 18801311 18838613 19057542

11 Georgia 9687653 9687660 9712157 9815210

12 Hawaii 1360301 1360301 1363359 1374810

13 Idaho 1567582 1567582 1571102 1584985

14 Illinois 12830632 12830632 12841980 12869257

15 Indiana 6483802 6483800 6490622 6516922

16 Iowa 3046355 3046350 3050202 3062309

17 Kansas 2853118 2853118 2859143 2871238

18 Kentucky 4339367 4339362 4347223 4369356

19 Louisiana 4533372 4533372 4545343 4574836

20 Maine 1328361 1328361 1327379 1328188

21 Maryland 5773552 5773552 5785681 5828289

22 Massachusetts 6547629 6547629 6555466 6587536

23 Michigan 9883640 9883635 9877143 9876187

24 Minnesota 5303925 5303925 5310658 5344861

25 Mississippi 2967297 2967297 2970072 2978512

26 Missouri 5988927 5988927 5995715 6010688

27 Montana 989415 989415 990958 998199

28 Nebraska 1826341 1826341 1830141 1842641

29 Nevada 2700551 2700551 2704283 2723322

30 New Hampshire 1316470 1316472 1316807 1318194

31 New Jersey 8791894 8791894 8799593 8821155

32 New Mexico 2059179 2059180 2065913 2082224

33 New York 19378102 19378104 19395206 19465197

34 North Carolina 9535483 9535475 9560234 9656401

35 North Dakota 672591 672591 674629 683932

36 Ohio 11536504 11536502 11537968 11544951

37 Oklahoma 3751351 3751354 3760184 3791508

38 Oregon 3831074 3831074 3838332 3871859

39 Pennsylvania 12702379 12702379 12717722 12742886

40 Rhode Island 1052567 1052567 1052528 1051302

41 South Carolina 4625364 4625364 4637106 4679230

42 South Dakota 814180 814180 816598 824082

43 Tennessee 6346105 6346110 6357436 6403353

44 Texas 25145561 25145561 25253466 25674681

45 Utah 2763885 2763885 2775479 2817222

46 Vermont 625741 625741 625909 626431

47 Virginia 8001024 8001030 8023953 8096604

48 Washington 6724540 6724540 6742950 6830038

49 West Virginia 1852994 1852996 1854368 1855364

50 Wisconsin 5686986 5686986 5691659 5711767

51 Wyoming 563626 563626 564554 568158

> mean(dfStates$Jul2011)

[1] 6109645

>

>

> #Step4

> which.max(dfStates$Jul2011) #Call maxpop index of July 2011 Column

[1] 5

> dfStates [5,5] #Call dataframe for maxpop Jul2011 index

[1] 37691912

> dfStates [5,1] #Call dataframe for stateName Jul2011 index

[1] "California"

> dfStates [order (dfStates$Jul2011),] #Sort by Max Jul2011 Population in ASC order

stateName base2010 base2011 Jul2010 Jul2011

51 Wyoming 563626 563626 564554 568158

9 District of Columbia 601723 601723 604912 617996

46 Vermont 625741 625741 625909 626431

35 North Dakota 672591 672591 674629 683932

2 Alaska 710231 710231 714146 722718

42 South Dakota 814180 814180 816598 824082

8 Delaware 897934 897934 899792 907135

27 Montana 989415 989415 990958 998199

40 Rhode Island 1052567 1052567 1052528 1051302

30 New Hampshire 1316470 1316472 1316807 1318194

20 Maine 1328361 1328361 1327379 1328188

12 Hawaii 1360301 1360301 1363359 1374810

13 Idaho 1567582 1567582 1571102 1584985

28 Nebraska 1826341 1826341 1830141 1842641

49 West Virginia 1852994 1852996 1854368 1855364

32 New Mexico 2059179 2059180 2065913 2082224

29 Nevada 2700551 2700551 2704283 2723322

45 Utah 2763885 2763885 2775479 2817222

17 Kansas 2853118 2853118 2859143 2871238

4 Arkansas 2915918 2915921 2921588 2937979

25 Mississippi 2967297 2967297 2970072 2978512

16 Iowa 3046355 3046350 3050202 3062309

7 Connecticut 3574097 3574097 3575498 3580709

37 Oklahoma 3751351 3751354 3760184 3791508

38 Oregon 3831074 3831074 3838332 3871859

18 Kentucky 4339367 4339362 4347223 4369356

19 Louisiana 4533372 4533372 4545343 4574836

41 South Carolina 4625364 4625364 4637106 4679230

1 Alabama 4779736 4779735 4785401 4802740

6 Colorado 5029196 5029196 5047692 5116796

24 Minnesota 5303925 5303925 5310658 5344861

50 Wisconsin 5686986 5686986 5691659 5711767

21 Maryland 5773552 5773552 5785681 5828289

26 Missouri 5988927 5988927 5995715 6010688

43 Tennessee 6346105 6346110 6357436 6403353

3 Arizona 6392017 6392013 6413158 6482505

15 Indiana 6483802 6483800 6490622 6516922

22 Massachusetts 6547629 6547629 6555466 6587536

48 Washington 6724540 6724540 6742950 6830038

47 Virginia 8001024 8001030 8023953 8096604

31 New Jersey 8791894 8791894 8799593 8821155

34 North Carolina 9535483 9535475 9560234 9656401

11 Georgia 9687653 9687660 9712157 9815210

23 Michigan 9883640 9883635 9877143 9876187

36 Ohio 11536504 11536502 11537968 11544951

39 Pennsylvania 12702379 12702379 12717722 12742886

14 Illinois 12830632 12830632 12841980 12869257

10 Florida 18801310 18801311 18838613 19057542

33 New York 19378102 19378104 19395206 19465197

44 Texas 25145561 25145561 25253466 25674681

5 California 37253956 37253956 37338198 37691912

>

> #Step5

> #Write function that will return the percentage of the elements within the vector that is less than the same

>

> v <- dfStates$Jul2011 #Store Vector for col "Jul2011"

> mean <- mean(dfStates$Jul2011) #Store Vector for mean Jul2011 Pop

>

> ####Version 1

> function1 <- function(v) {

+ length(which(v<mean))/ length(v) #Length of which states less than mean/length of states

+ paste(round(100\* length(which(v<mean))/ length(v), 2), "%", sep = "") #Convert to percentage format

+ meanDfStates <- paste(round(100\* length(which(v<mean))/ length(v), 2), "%", sep = "")

+ return(meanDfStates)

+ }

>

> function1 (dfStates$Jul2011) #Call Function for col "Jul2011"

[1] "66.67%"

>

>

> ###OR Version 2

>

> function2 <- function(v) {

+ sum(v<mean, na.rm = TRUE)/ length(v) #Sum of Trues less than mean/ length of column

+ paste(round(100\* sum(v<mean, na.rm = TRUE)/ length(dfStates$Jul2011), 2), "%", sep = "") #Convert to percentage format

+

+ }

>

> function2 (dfStates$Jul2011) #Call Function for col "Jul2011"

[1] "66.67%"