Web Scraping Data

Mark Baber

2021

# My Scripts

This file will be my scripts of how/if I managed to get any data webscraped from sites such as Stats Wales & ONS (maybe some more).

## 1 - Packages

First step is to get load a few libraries which will be used.

library(knitr)  
library(rvest)  
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.4 v dplyr 1.0.7  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 2.0.1 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x readr::guess\_encoding() masks rvest::guess\_encoding()  
## x dplyr::lag() masks stats::lag()

library(taRifx)

##   
## Attaching package: 'taRifx'

## The following objects are masked from 'package:dplyr':  
##   
## between, distinct, first, last

## The following object is masked from 'package:purrr':  
##   
## rep\_along

## 2 - Set URL

This section will have code which will set a URL, which will then be scraped together. First is number of gps employed in general practices.

# set url  
url <- 'https://statswales.gov.wales/Catalogue/Health-and-Social-Care/General-Medical-Services/number-of-gps-employed-in-general-practices'  
# grab content from url  
content <- read\_html(url)

After setting the url, and grabbing the content from the url, lets go ahead and try to only get the contents which is part of the css field ‘table’.

get.tables <- content %>%   
 html\_element("table") %>%   
 html\_table()

now to subset the data as it is very messy and only keep the data which is actually important to us.

# first only select what columns to keep  
slice.one <- get.tables %>%   
 subset(select = 1:7)  
# now select what rows to keep  
slice.two <- slice.one[c(18:25),]  
# only keep the gp staff data  
mydf.staff.gp <- slice.two[1:7]

Now we have a decent dataset, lets go through at make sure the row names are correctly added.

# kept row names in as can't add them later  
# row.names1 = slice.two[,1]  
# shorten the names manually.  
row.names = c("Wales", "Betsi Cadwaladr Uni",  
 "Powys Teaching", "Hywel Dda Uni",  
 "Swansea Bay Uni", "Cwm Taf Morgannwg Uni",  
 "Aneurin Bevan Uni", "Cardiff and Vale Uni")  
  
  
# set the names of the table (should be sliced but typing is easier atm.)  
col.names <- c("Local Health Board",  
 "GP Partner/Provider/SPartner","Salaried",  
 "GP Practitioner", "GP Registrar",  
 "GP Retainer", "GP Locum")  
  
# Apply the names to the table (slice.two)  
mydf.staff.gp <- setNames(mydf.staff.gp, col.names)

Within the data, there are a few columns which are doubles, example “1,200” - this will need to be converted to a numeric value, which can be done by assigning x value with x as a numeric. (This was done by looking at the value and assigning it as it is typed out instead.)

Also noticed there are 2 symbols within our tibble which isn’t required and will slow down the next steps, these symbols are “-” and “\*” - lets attempt to replace these with 0’s.

df.staff.gp <- data.frame(mydf.staff.gp)  
  
# replace the "-", "\*", "\*" with 0's  
df.staff.gp$GP.Retainer[3:5] <- c(0,0,0)  
  
df.staff.gp$GP.Retainer %>% mean()

## Warning in mean.default(.): argument is not numeric or logical: returning NA

## [1] NA

# replace doubles to numeric  
# df.staff.gp <- lapply(df.staff.gp[-1], destring)  
# this gave me issues, so is done manually below.  
  
# can be done manually.  
df.staff.gp$GP.Partner.Provider.SPartner[1] <- 1467  
df.staff.gp$GP.Practitioner[1] <- 1963  
print(df.staff.gp)

## Local.Health.Board GP.Partner.Provider.SPartner  
## 1 Wales 1467  
## 2 Betsi Cadwaladr University Health Board 307  
## 3 Powys Teaching Health Board 79  
## 4 Hywel Dda University Health Board 169  
## 5 Swansea Bay University Health Board 208  
## 6 Cwm Taf Morgannwg University Health Board 207  
## 7 Aneurin Bevan University Health Board 272  
## 8 Cardiff and Vale University Health Board 238  
## Salaried GP.Practitioner GP.Registrar GP.Retainer GP.Locum  
## 1 496 1963 382 24 828  
## 2 92 399 69 6 222  
## 3 19 98 10 0 57  
## 4 49 218 58 0 87  
## 5 54 262 59 0 111  
## 6 91 298 75 4 90  
## 7 101 373 46 3 147  
## 8 94 332 65 8 114

Firstly, it looks like the date from the website has been defaulted to the older date, and I was unable to switch it to work with this code. I will be mindful with this going forward.

Date Picker - Stats Wales

Date Picker - Stats Wales

Now that the data has been sliced and carefully selected the parts of the data to keep, I want to try and get the Total added onto the end of the table which will make analysing it a bit easier.

# check column classes  
sapply(df.staff.gp, class)

## Local.Health.Board GP.Partner.Provider.SPartner   
## "character" "character"   
## Salaried GP.Practitioner   
## "character" "character"   
## GP.Registrar GP.Retainer   
## "character" "character"   
## GP.Locum   
## "character"

# get list of the character classes  
chars <- sapply(df.staff.gp, is.character)  
# replace all chars with numeric instead  
df.staff.gp[ , chars] <- as.data.frame(apply(df.staff.gp[ , chars], 2, as.numeric))

## Warning in apply(df.staff.gp[, chars], 2, as.numeric): NAs introduced by  
## coercion

# sanity check it worked  
sapply(df.staff.gp, class)

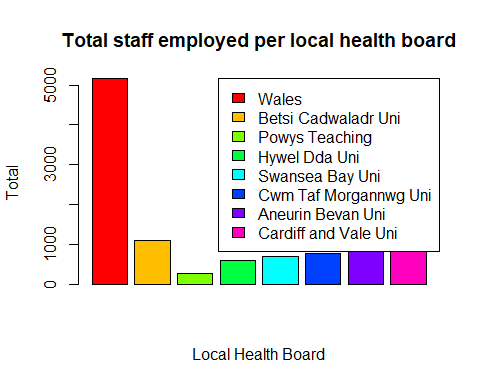
## Local.Health.Board GP.Partner.Provider.SPartner   
## "numeric" "numeric"   
## Salaried GP.Practitioner   
## "numeric" "numeric"   
## GP.Registrar GP.Retainer   
## "numeric" "numeric"   
## GP.Locum   
## "numeric"

# write.csv(df.staff.gp, "data/staff-per-gp.csv")

There was a few issues with the data, as it was a mix of characters, strings and doubles (obviously) - So above, I tried to ‘sapply’ through each column to see which were characters, save these as a list and check the characters against the numeric entries and encode them as numeric. A bit confusing, I know but it worked.

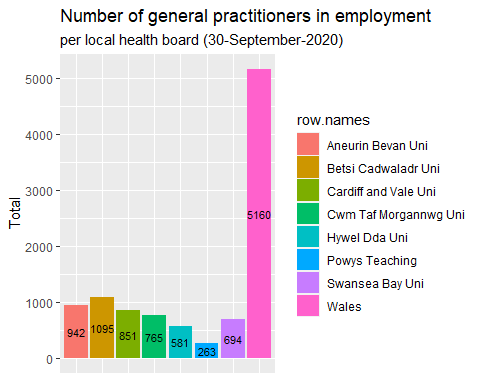
Next lets try to get the Total of staff in GPs across Wales

# sum up the rows  
# write.csv(df.staff.gp, file = "data/gp-number-check.csv")  
df.staff.gp$Total <- rowSums(df.staff.gp[,c(-1)])  
# plot the data  
barplot(df.staff.gp$Total, main = "Total staff employed per local health board",   
 xlab = "Local Health Board",   
 ylab = "Total",   
 horiz = F,   
 col = rainbow(8),   
 legend.text = row.names,   
 args.legend = list(x = "topright"))



This plot could be nicer, with the use of ggplot2. This package gives you a lot of control over creating plots which can be both a positive and a negative.

plot.one <-   
 ggplot(df.staff.gp, aes(row.names, Total, fill = row.names)) +   
 geom\_col() +  
 geom\_text(aes(label = Total, y = (Total/2)), size = 3)  
  
plot.one +   
 labs(title = "Number of general practitioners in employment",   
 subtitle = "per local health board (30-September-2020)") +  
 theme(axis.title.x = element\_blank(),  
 axis.text.x = element\_blank(),  
 axis.ticks.x = element\_blank())



From looking at the plot above it looks like Wales is the Total of all over districts, lets do a sanity check to see if this is correct (if not, look into how these local health boards are set up for better context.)

gp.districts <-   
 (942 + 1095 + 851 + 765 + 581 + 263 + 694)  
  
print(gp.districts)

[1] 5191

print(5160 - (gp.districts))

[1] -31

#kable(gp.districts, caption = "Sanity check of GP districts")  
# write.csv(x = df.staff.gp, "gp-Totals.csv", row.names = T)

It looks like there seems to be 31 entries missing from this dataset, this will have to be kept in mind going forward.

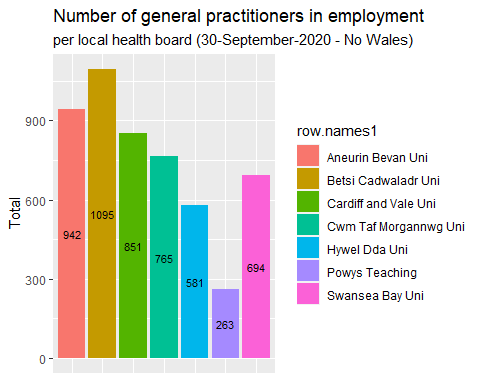
df.no.wales <- df.staff.gp[2:7]   
df.no.wales <- df.no.wales[c(-1),]  
head(df.no.wales)

## GP.Partner.Provider.SPartner Salaried GP.Practitioner GP.Registrar  
## 2 307 92 399 69  
## 3 79 19 98 10  
## 4 169 49 218 58  
## 5 208 54 262 59  
## 6 207 91 298 75  
## 7 272 101 373 46  
## GP.Retainer GP.Locum  
## 2 6 222  
## 3 0 57  
## 4 0 87  
## 5 0 111  
## 6 4 90  
## 7 3 147

# shorten the names manually.  
row.names1 = c("Betsi Cadwaladr Uni",  
 "Powys Teaching",   
 "Hywel Dda Uni",  
 "Swansea Bay Uni",   
 "Cwm Taf Morgannwg Uni",  
 "Aneurin Bevan Uni",   
 "Cardiff and Vale Uni")  
  
  
# set the names of the table (should be sliced but typing is easier atm.)  
col.names1 <- c("GP Partner/Provider/SPartner","Salaried",  
 "GP Practitioner", "GP Registrar",  
 "GP Retainer", "GP Locum")  
  
# Apply the names to the table   
df.no.wales <- setNames(df.no.wales, col.names1)  
# add Total again  
df.no.wales$Total <- rowSums(df.no.wales)

Now we have removed ‘Wales’, lets create another plot but without Wales this time, just the health boards across wales.

plot.one.no.wales <-   
 ggplot(df.no.wales, aes(row.names1, Total, fill = row.names1)) +   
 geom\_col() +  
 geom\_text(aes(label = Total, y = (Total/2)),   
 size = 3)  
  
plot.one.no.wales +   
 labs(title = "Number of general practitioners in employment",   
 subtitle = "per local health board (30-September-2020 - No Wales)") +  
 theme(axis.title.x = element\_blank(),  
 axis.text.x = element\_blank(),  
 axis.ticks.x = element\_blank())



# write.csv(df.no.wales, file = "data/number-of-general.csv")

Here we can also pull some descriotive analysis from the dataset using a library called funModeling.

library(funModeling)

## Loading required package: Hmisc

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

##   
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:dplyr':  
##   
## src, summarize

## The following objects are masked from 'package:base':  
##   
## format.pval, units

## funModeling v.1.9.4 :)  
## Examples and tutorials at livebook.datascienceheroes.com  
## / Now in Spanish: librovivodecienciadedatos.ai

profiling\_num(df.no.wales)

## variable mean std\_dev variation\_coef p\_01 p\_05  
## 1 GP Partner/Provider/SPartner 211.42857 74.02445 0.3501156 84.40 106.0  
## 2 Salaried 71.42857 30.94542 0.4332359 20.80 28.0  
## 3 GP Practitioner 282.85714 102.52874 0.3624753 105.20 134.0  
## 4 GP Registrar 54.57143 21.70144 0.3976703 12.16 20.8  
## 5 GP Retainer 3.00000 3.21455 1.0715168 0.00 0.0  
## 6 GP Locum 118.28571 53.52169 0.4524781 58.80 66.0  
## 7 Total 741.57143 269.16033 0.3629594 282.08 358.4  
## p\_25 p\_50 p\_75 p\_95 p\_99 skewness kurtosis iqr range\_98  
## 1 188.0 208 255.0 296.5 304.90 -0.5703036 2.643342 67.0 [84.4, 304.9]  
## 2 51.5 91 93.0 98.9 100.58 -0.6507881 1.961554 41.5 [20.8, 100.58]  
## 3 240.0 298 352.5 391.2 397.44 -0.6977630 2.532922 112.5 [105.2, 397.44]  
## 4 52.0 59 67.0 73.2 74.64 -1.3454481 3.666173 15.0 [12.16, 74.64]  
## 5 0.0 3 5.0 7.4 7.88 0.3902060 1.729969 5.0 [0, 7.88]  
## 6 88.5 111 130.5 199.5 217.50 1.0043769 3.130103 42.0 [58.8, 217.5]  
## 7 637.5 765 896.5 1049.1 1085.82 -0.5569860 2.611259 259.0 [282.08, 1085.82]  
## range\_80  
## 1 [133, 286]  
## 2 [37, 96.8]  
## 3 [170, 383.4]  
## 4 [31.6, 71.4]  
## 5 [0, 6.8]  
## 6 [75, 177]  
## 7 [453.8, 1003.2]

str(df.no.wales)

## 'data.frame': 7 obs. of 7 variables:  
## $ GP Partner/Provider/SPartner: num 307 79 169 208 207 272 238  
## $ Salaried : num 92 19 49 54 91 101 94  
## $ GP Practitioner : num 399 98 218 262 298 373 332  
## $ GP Registrar : num 69 10 58 59 75 46 65  
## $ GP Retainer : num 6 0 0 0 4 3 8  
## $ GP Locum : num 222 57 87 111 90 147 114  
## $ Total : num 1095 263 581 694 765 ...

df\_status(df.no.wales)

## variable q\_zeros p\_zeros q\_na p\_na q\_inf p\_inf type  
## 1 GP Partner/Provider/SPartner 0 0.00 0 0 0 0 numeric  
## 2 Salaried 0 0.00 0 0 0 0 numeric  
## 3 GP Practitioner 0 0.00 0 0 0 0 numeric  
## 4 GP Registrar 0 0.00 0 0 0 0 numeric  
## 5 GP Retainer 3 42.86 0 0 0 0 numeric  
## 6 GP Locum 0 0.00 0 0 0 0 numeric  
## 7 Total 0 0.00 0 0 0 0 numeric  
## unique  
## 1 7  
## 2 7  
## 3 7  
## 4 7  
## 5 5  
## 6 7  
## 7 7

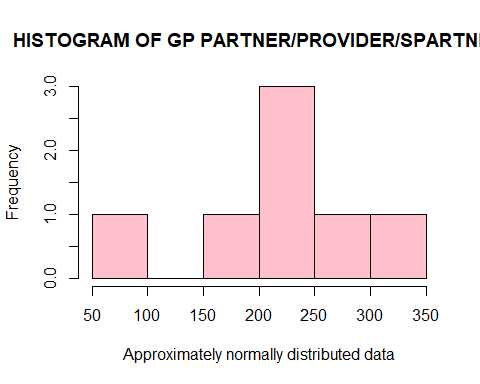
# write.csv(profiling\_num(df.no.wales), file = "data/desc-gps-employed.csv")

## 2.X - Plots for Normality.

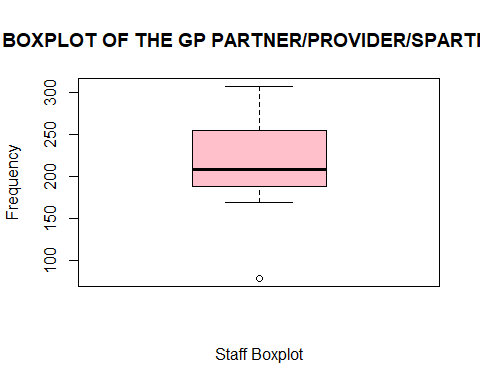
This section will be a few histograms and boxplots to look for normality and any outliers within the data.

* GP Partners

# save this plot as png  
gpPartnerHist <-   
 hist(df.no.wales$`GP Partner/Provider/SPartner`,  
 main = toupper("Histogram of GP Partner/Provider/SPartner"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")

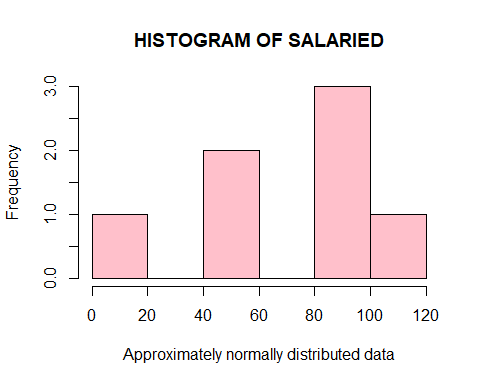


# plot a boxplot  
gpPartnerBoxplot <-   
 boxplot(df.no.wales$`GP Partner/Provider/SPartner`,  
 main = toupper("Boxplot of the GP Partner/Provider/SPartner"),  
 xlab = "Staff Boxplot",  
 ylab = "Frequency",  
 col = "pink")

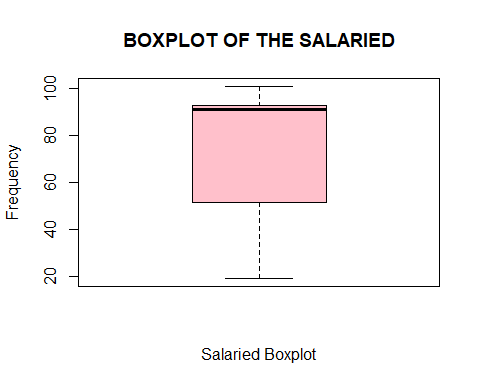


* Salaried

gpSalariedHist <-   
 hist(df.no.wales$Salaried,  
 main = toupper("Histogram of Salaried"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")

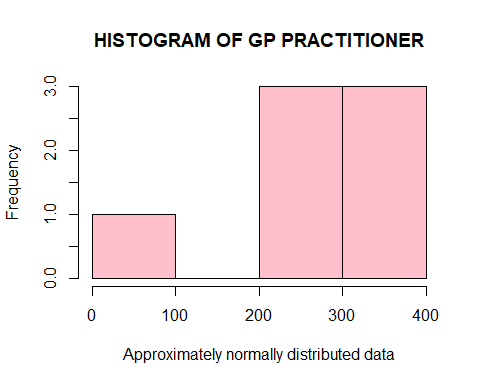


# plot a boxplot  
gpSalariedBox <-   
 boxplot(df.no.wales$Salaried,  
 main = toupper("Boxplot of the Salaried"),  
 xlab = "Salaried Boxplot",  
 ylab = "Frequency",  
 col = "pink")

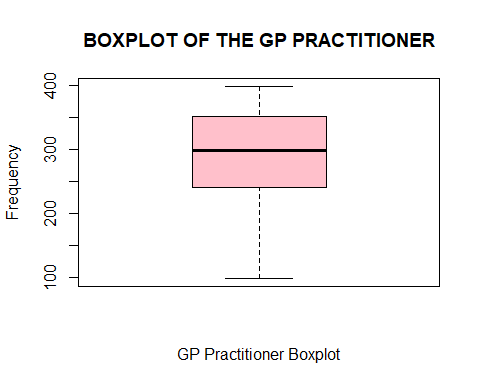


* GP Practitioner

gpPractitionerHist <-   
 hist(df.no.wales$`GP Practitioner`,  
 main = toupper("Histogram of GP Practitioner"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")

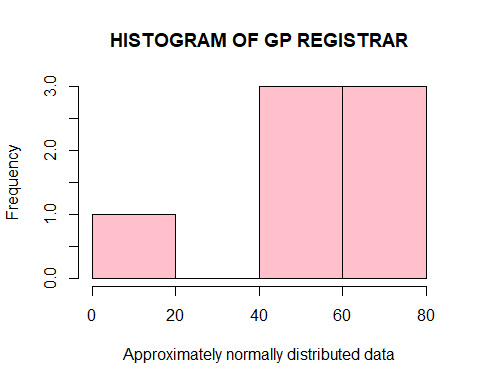


# plot a boxplot  
gpPractitionerBox <- boxplot(df.no.wales$`GP Practitioner`,  
 main = toupper("Boxplot of the GP Practitioner"),  
 xlab = "GP Practitioner Boxplot",  
 ylab = "Frequency",  
 col = "pink")

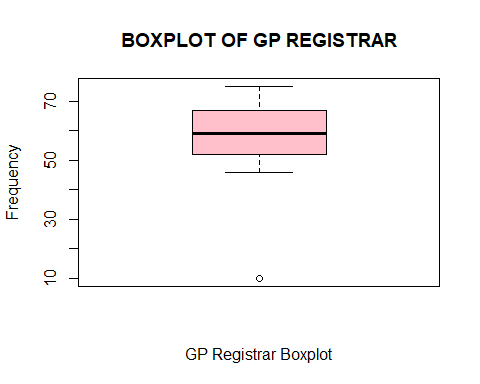


* GP Registrar

gpRegistrarHist <-   
 hist(df.no.wales$`GP Registrar`,  
 main = toupper("Histogram of GP Registrar"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")

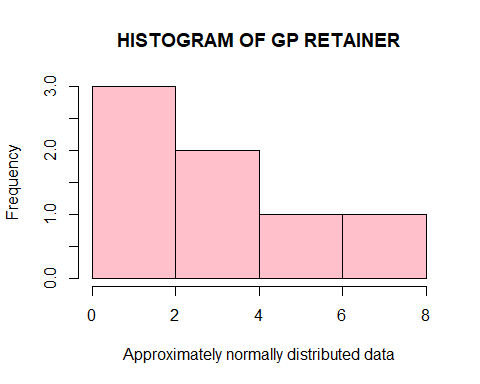


# plot a boxplot  
gpRegistrarBox <-   
 boxplot(df.no.wales$`GP Registrar`,  
 main = toupper("Boxplot of GP Registrar"),  
 xlab = "GP Registrar Boxplot",  
 ylab = "Frequency",  
 col = "pink")

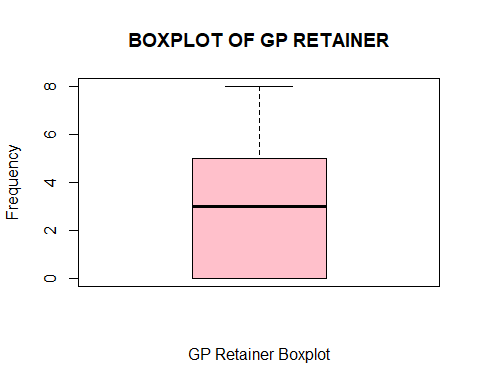


* GP Retainer

gpRetainerHist <-   
 hist(df.no.wales$`GP Retainer`,  
 main = toupper("Histogram of GP Retainer"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")

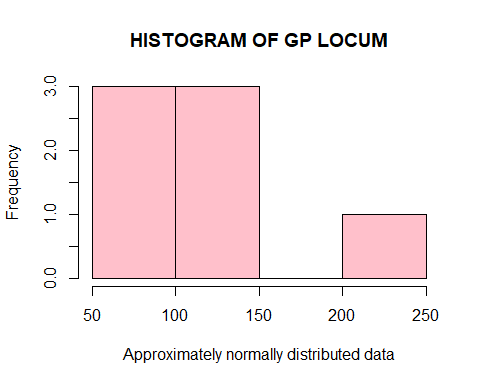


# plot a boxplot  
gpRetainerBox <-   
 boxplot(df.no.wales$`GP Retainer`,  
 main = toupper("Boxplot of GP Retainer"),  
 xlab = "GP Retainer Boxplot",  
 ylab = "Frequency",  
 col = "pink")

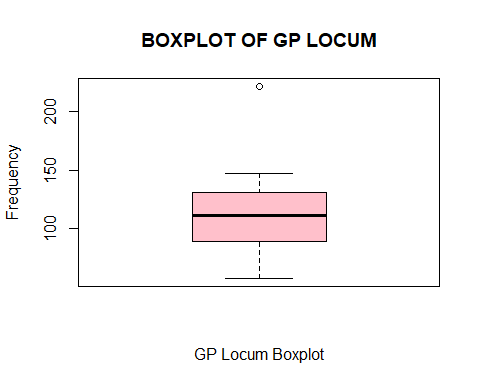


* GP Locum

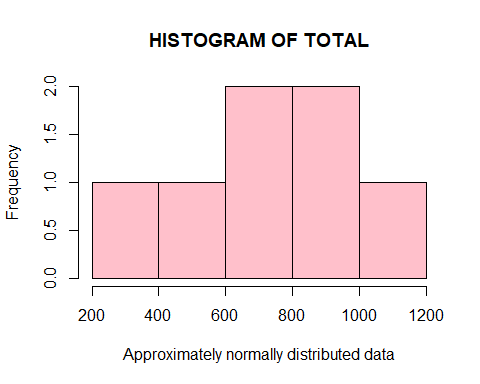
gpLocumHist <-   
 hist(df.no.wales$`GP Locum`,  
 main = toupper("Histogram of GP Locum"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")



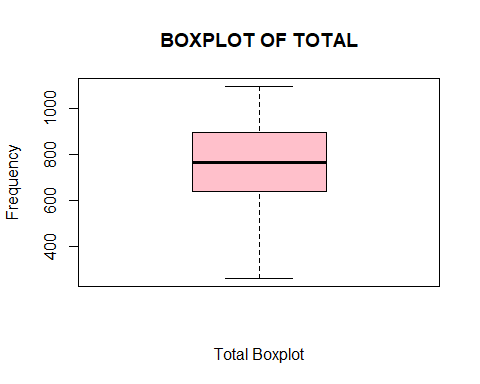
# plot a boxplot  
gpLocumBox <-   
 boxplot(df.no.wales$`GP Locum`,  
 main = toupper("Boxplot of GP Locum"),  
 xlab = "GP Locum Boxplot",  
 ylab = "Frequency",  
 col = "pink")



gpTotalHist <-   
 hist(df.no.wales$Total,  
 main = toupper("Histogram of Total"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")



# plot a boxplot  
gpTotalHist <-   
 boxplot(df.no.wales$Total,  
 main = toupper("Boxplot of Total"),  
 xlab = "Total Boxplot",  
 ylab = "Frequency",  
 col = "pink")



Now we have the staff employed per HB within Wales, it would be interesting to see how the population is within these HB areas. I will try to do the same thing as above, but with a different dataset.

# set new url  
url.pop <- "https://statswales.gov.wales/Catalogue/Population-and-Migration/Population/Estimates/Local-Health-Boards/populationestimates-by-welshhealthboard-year"  
# grab content from url  
content.pop <- read\_html(url.pop)  
# grab the tables from the page  
get.tables.pop <- content.pop %>%   
 html\_element("table") %>%   
 html\_table()

This table seems to be much bigger than the first one, so some of the script might need tweaking.

# first only select what columns to keep  
slice.pop1 <- get.tables.pop %>%   
 subset(select = 2:14)  
# now select what rows to keep  
slice.pop2 <- slice.pop1[c(25:32),]  
# drop the area code  
slice.pop3 <- slice.pop2[c(-2)]  
  
# get the years  
slice.pop.years <- get.tables.pop[15:25]  
slice.pop.years <- slice.pop.years[c(18),]  
  
# get the areas  
slice.pop.areas <- slice.pop3[1]  
  
# Get all data without areas  
df.pop <- slice.pop3[-1]  
# set the column names (years)  
df.pop <- setNames(df.pop, slice.pop.years)

This has sliced the data correctly, but now we need to convert it from char/list to numeric again.

# de-string the numbers ( ## this was an issue >.< ## )  
df.pop <- lapply(df.pop, destring)  
# check the class again  
class(df.pop)

## [1] "list"

# the class is a list, which we want as a dataframe again as numeric.  
df.pop <- data.frame(matrix(unlist(df.pop),  
 ncol = 11,   
 nrow = 8),  
 stringsAsFactors = FALSE)  
# finally converted from character/list to df.  
class(df.pop)

## [1] "data.frame"

Here we have managed to convert the data which was a lot harder this time with a lot of types (doubles, characters and lists) - this was made easier with a package discovered on 05/07/2021 called ‘taRifx’ and using ‘destring’.

Now that the data has been stripped and converted, we need to add the column names and row names again, to create a plots like above with Wales and without Wales.

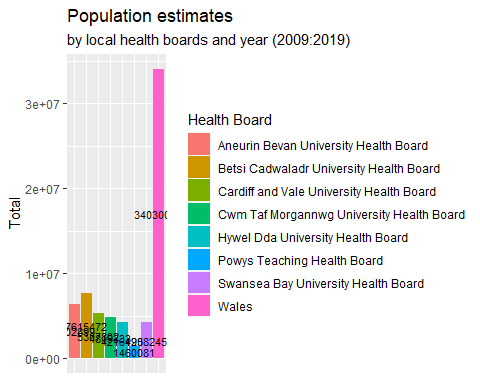
# get the years, but only keep the years and remove 'Mid-year'  
# and trim the white space, just to make sure.  
slice.pop.years <- slice.pop.years %>% str\_replace\_all('Mid-year', '') %>% trimws()  
# add the col names with setNames  
df.pop <- setNames(df.pop, slice.pop.years)  
# column bind the Health Board Areas with df.pop  
df.pop <- cbind(slice.pop.areas, df.pop)  
# sanity check  
df.pop

## X2 2009 2010 2011 2012  
## 1 Wales 3038872 3049971 3063758 3074067  
## 2 Betsi Cadwaladr University Health Board 684575 685911 688417 689924  
## 3 Powys Teaching Health Board 133090 132878 133071 133015  
## 4 Hywel Dda University Health Board 379051 380195 381867 383399  
## 5 Swansea Bay University Health Board 375138 376949 378571 379541  
## 6 Cwm Taf Morgannwg University Health Board 430682 431423 432634 434288  
## 7 Aneurin Bevan University Health Board 572518 574778 577077 578178  
## 8 Cardiff and Vale University Health Board 463818 467837 472121 475722  
## 2013 2014 2015 2016 2017 2018 2019  
## 1 3082412 3092036 3099086 3113150 3125165 3138631 3152879  
## 2 691180 693067 693360 694826 696284 698369 699559  
## 3 132786 132777 132730 132337 132515 132447 132435  
## 4 383833 383927 383129 383656 384239 385615 387284  
## 5 379975 381419 383262 386140 387570 389372 390308  
## 6 435710 437214 438884 441301 443368 445190 448639  
## 7 579346 580794 582245 584831 587743 591225 594164  
## 8 479582 482838 485476 490059 493446 496413 500490

df.pop <- rename(df.pop, "Health Board" = "X2")  
# add Total again  
df.pop$Total <- rowSums(df.pop[-1])

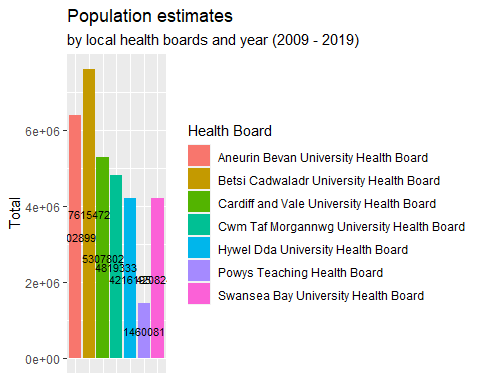
Now to try and plot the Total population per health board.

# update the col names pop  
col.names.pop <- c("Health Board", slice.pop.years)  
  
plot.two <-   
 ggplot(df.pop, aes(`Health Board`, Total, fill = `Health Board`)) +   
 geom\_col() +  
 geom\_text(aes(label = Total, y = (Total/2)), size = 3)  
  
plot.two +   
 labs(title = "Population estimates", subtitle = "by local health boards and year (2009:2019)") +  
 theme(axis.title.x = element\_blank(),  
 axis.text.x = element\_blank(),  
 axis.ticks.x = element\_blank())



Now to try it without Wales again, which is throwing off the plot.

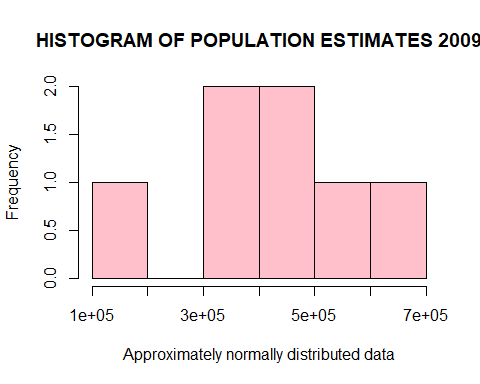
# save df.pop without wales  
df.pop.no.wales <- df.pop[-1,]  
#write.csv(df.pop.no.wales, 'pop-estimates.csv')  
  
plot.two.no.wales <-   
 ggplot(df.pop[-1,], aes(`Health Board`, Total, fill = `Health Board`)) +   
 geom\_col() +   
 geom\_text(aes(label = Total, y = (Total/2)), size = 3)  
  
plot.two.no.wales +   
 labs(title = "Population estimates", subtitle = "by local health boards and year (2009 - 2019)") +  
 theme(axis.title.x = element\_blank(),  
 axis.text.x = element\_blank(),  
 axis.ticks.x = element\_blank())



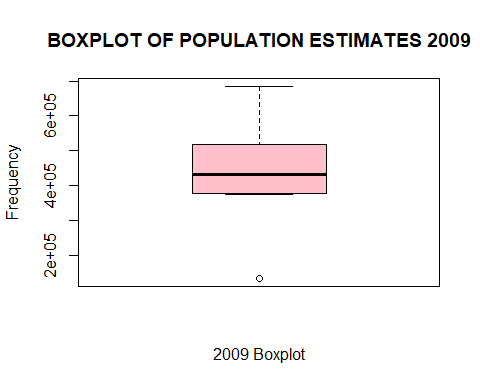
# write.csv(df.pop.no.wales, file = "data/pop-estimations.csv")

Before moving on to some descriptive statistics and hypothesis testing, lets first get a little look at the data for normality again with some plots.

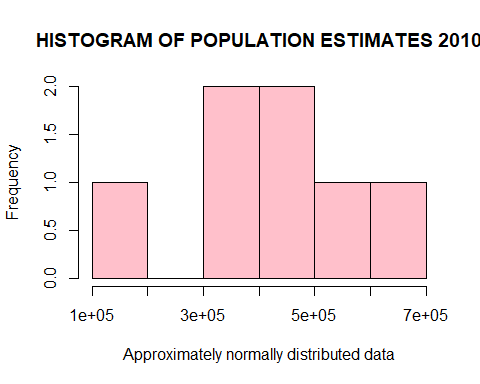
# 2009  
pop2009Hist <-   
 hist(df.pop.no.wales$`2009`,  
 main = toupper("Histogram of Population Estimates 2009"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")



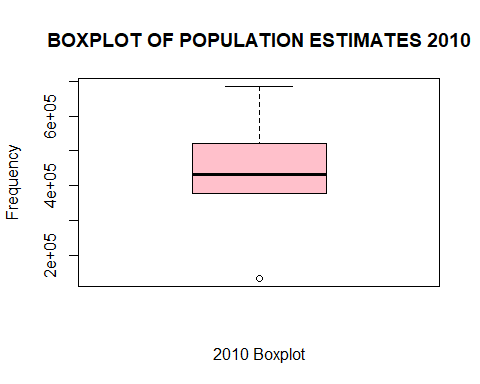
# plot a boxplot  
pop2009Box <-   
 boxplot(df.pop.no.wales$`2009`,  
 main = toupper("Boxplot of Population Estimates 2009"),  
 xlab = "2009 Boxplot",  
 ylab = "Frequency",  
 col = "pink")



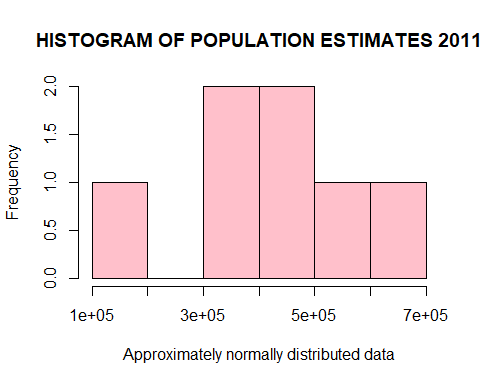
# 2010  
pop2010Hist <-   
 hist(df.pop.no.wales$`2010`,  
 main = toupper("Histogram of Population Estimates 2010"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")



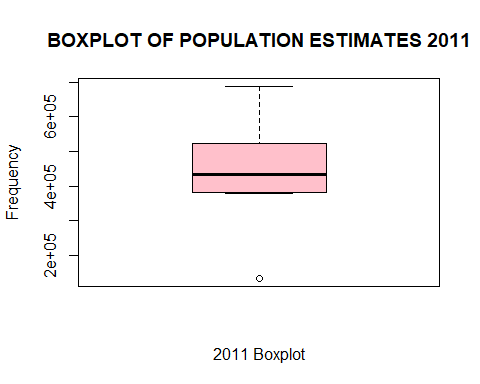
# plot a boxplot  
pop2010Box <-   
 boxplot(df.pop.no.wales$`2010`,  
 main = toupper("Boxplot of Population Estimates 2010"),  
 xlab = "2010 Boxplot",  
 ylab = "Frequency",  
 col = "pink")



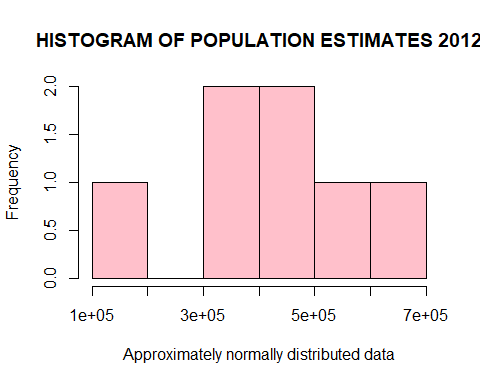
pop2011Hist <-   
 hist(df.pop.no.wales$`2011`,  
 main = toupper("Histogram of Population Estimates 2011"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")



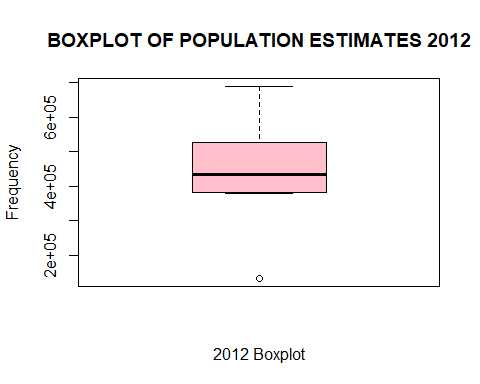
# plot a boxplot  
pop2011Box <-   
 boxplot(df.pop.no.wales$`2011`,  
 main = toupper("Boxplot of Population Estimates 2011"),  
 xlab = "2011 Boxplot",  
 ylab = "Frequency",  
 col = "pink")



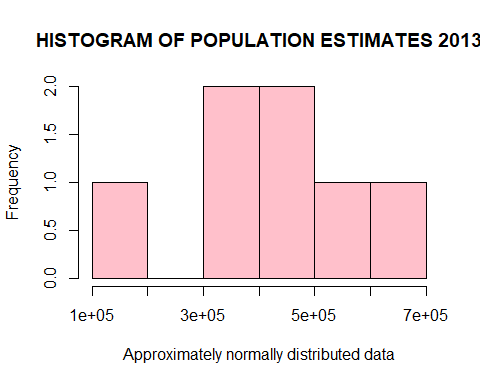
# 2012  
pop2012Hist <-   
 hist(df.pop.no.wales$`2012`,  
 main = toupper("Histogram of Population Estimates 2012"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")



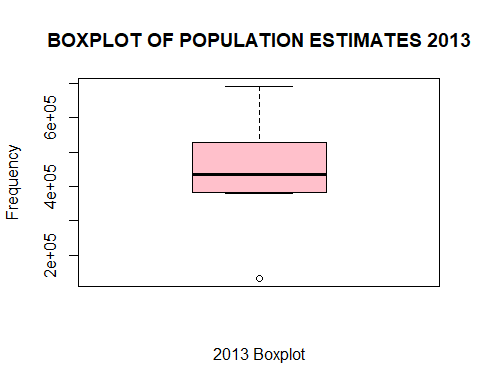
# plot a boxplot  
pop2012Box <-   
 boxplot(df.pop.no.wales$`2012`,  
 main = toupper("Boxplot of Population Estimates 2012"),  
 xlab = "2012 Boxplot",  
 ylab = "Frequency",  
 col = "pink")



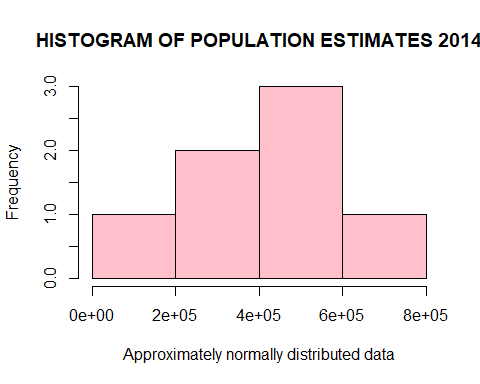
# 2013  
pop2013Hist <-   
 hist(df.pop.no.wales$`2013`,  
 main = toupper("Histogram of Population Estimates 2013"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")



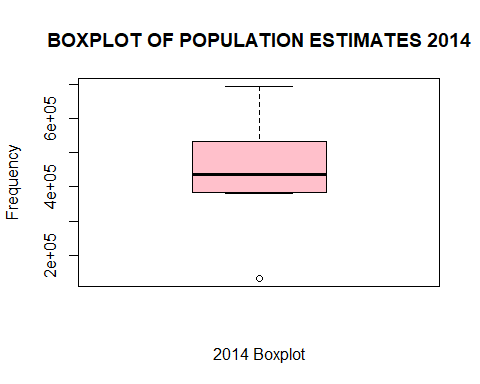
# plot a boxplot  
pop2013Box <-   
 boxplot(df.pop.no.wales$`2013`,  
 main = toupper("Boxplot of Population Estimates 2013"),  
 xlab = "2013 Boxplot",  
 ylab = "Frequency",  
 col = "pink")



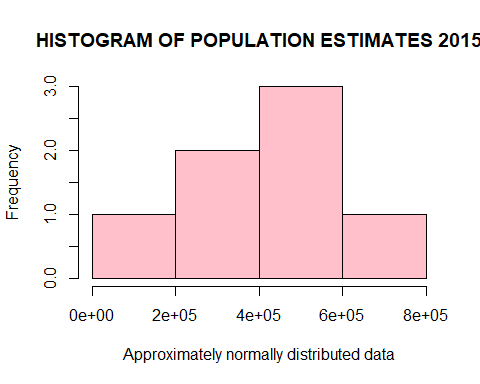
# 2014  
pop2014Hist <-   
 hist(df.pop.no.wales$`2014`,  
 main = toupper("Histogram of Population Estimates 2014"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")



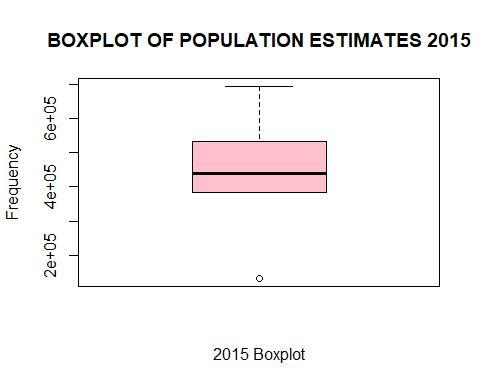
# plot a boxplot  
pop2014Box <-   
 boxplot(df.pop.no.wales$`2014`,  
 main = toupper("Boxplot of Population Estimates 2014"),  
 xlab = "2014 Boxplot",  
 ylab = "Frequency",  
 col = "pink")



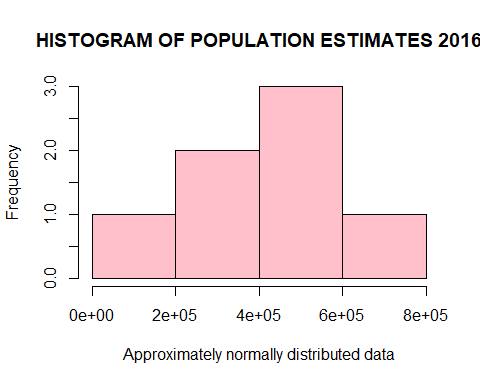
# 2015  
pop2015Hist <-   
 hist(df.pop.no.wales$`2015`,  
 main = toupper("Histogram of Population Estimates 2015"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")



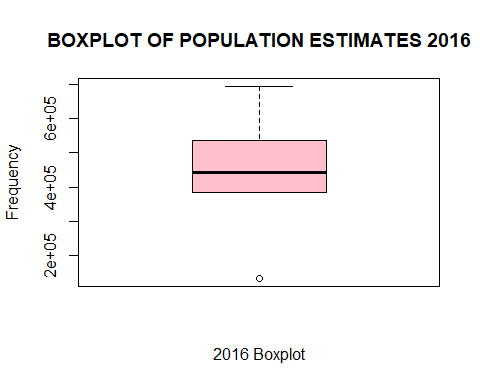
# plot a boxplot  
pop2015Box <-   
 boxplot(df.pop.no.wales$`2015`,  
 main = toupper("Boxplot of Population Estimates 2015"),  
 xlab = "2015 Boxplot",  
 ylab = "Frequency",  
 col = "pink")



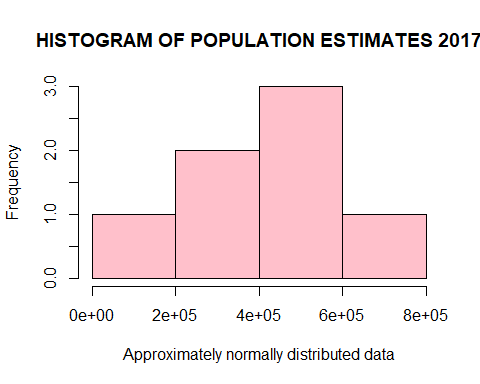
# 2016  
pop2016Hist <-   
 hist(df.pop.no.wales$`2016`,  
 main = toupper("Histogram of Population Estimates 2016"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")



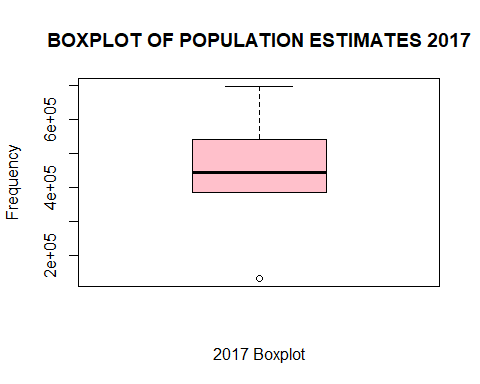
# plot a boxplot  
pop2016Box <-   
 boxplot(df.pop.no.wales$`2016`,  
 main = toupper("Boxplot of Population Estimates 2016"),  
 xlab = "2016 Boxplot",  
 ylab = "Frequency",  
 col = "pink")



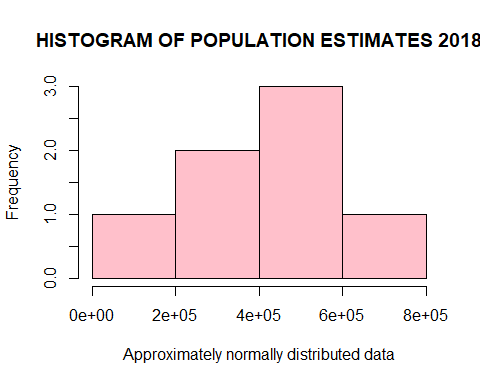
# 2017  
pop2017Hist <-   
 hist(df.pop.no.wales$`2017`,  
 main = toupper("Histogram of Population Estimates 2017"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")



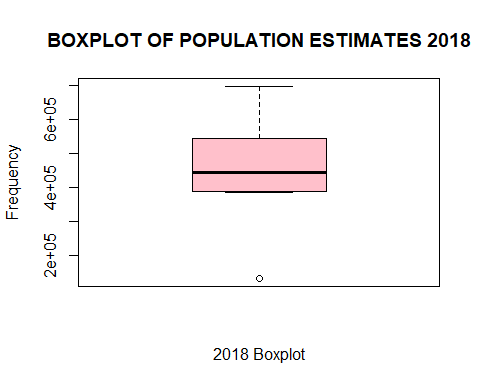
# plot a boxplot  
pop2017Box <-   
 boxplot(df.pop.no.wales$`2017`,  
 main = toupper("Boxplot of Population Estimates 2017"),  
 xlab = "2017 Boxplot",  
 ylab = "Frequency",  
 col = "pink")



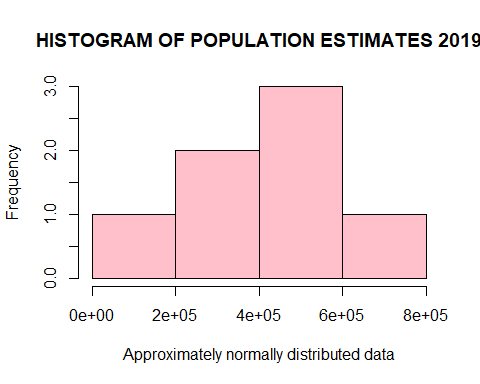
# 2018  
pop2018Hist <-   
 hist(df.pop.no.wales$`2018`,  
 main = toupper("Histogram of Population Estimates 2018"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")



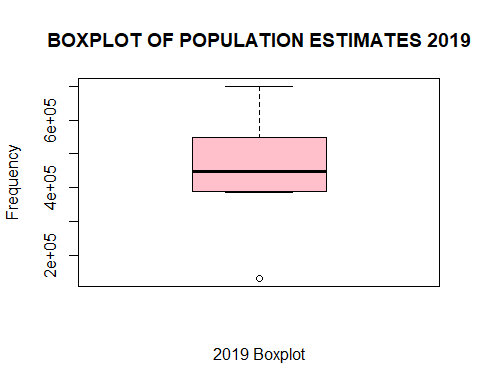
# plot a boxplot  
pop2018Box <-   
 boxplot(df.pop.no.wales$`2018`,  
 main = toupper("Boxplot of Population Estimates 2018"),  
 xlab = "2018 Boxplot",  
 ylab = "Frequency",  
 col = "pink")



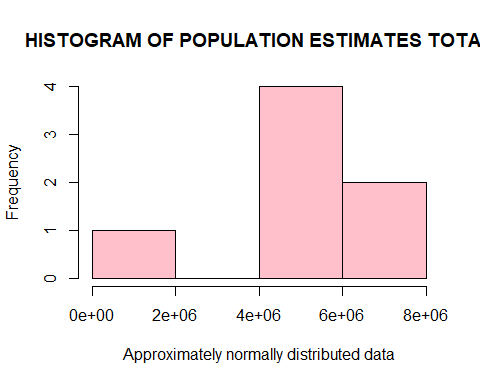
# 2019  
pop2019Hist <-   
 hist(df.pop.no.wales$`2019`,  
 main = toupper("Histogram of Population Estimates 2019"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")



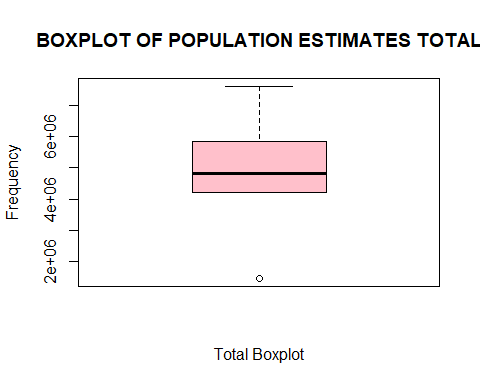
# plot a boxplot  
pop2019Box <-   
 boxplot(df.pop.no.wales$`2019`,  
 main = toupper("Boxplot of Population Estimates 2019"),  
 xlab = "2019 Boxplot",  
 ylab = "Frequency",  
 col = "pink")



# total  
popTotalHist <-   
 hist(df.pop.no.wales$Total,  
 main = toupper("Histogram of Population Estimates Total"),  
 xlab = "Approximately normally distributed data",  
 col = "pink")



# plot a boxplot  
popTotalBox <-   
 boxplot(df.pop.no.wales$`Total`,  
 main = toupper("Boxplot of Population Estimates Total"),  
 xlab = "Total Boxplot",  
 ylab = "Frequency",  
 col = "pink")



* *That was a lot of plots!!*

This data can also be ‘profiled’ to draw out some descriptive statistics from the data frame.

structure(df.pop.no.wales)

## Health Board 2009 2010 2011 2012 2013  
## 2 Betsi Cadwaladr University Health Board 684575 685911 688417 689924 691180  
## 3 Powys Teaching Health Board 133090 132878 133071 133015 132786  
## 4 Hywel Dda University Health Board 379051 380195 381867 383399 383833  
## 5 Swansea Bay University Health Board 375138 376949 378571 379541 379975  
## 6 Cwm Taf Morgannwg University Health Board 430682 431423 432634 434288 435710  
## 7 Aneurin Bevan University Health Board 572518 574778 577077 578178 579346  
## 8 Cardiff and Vale University Health Board 463818 467837 472121 475722 479582  
## 2014 2015 2016 2017 2018 2019 Total  
## 2 693067 693360 694826 696284 698369 699559 7615472  
## 3 132777 132730 132337 132515 132447 132435 1460081  
## 4 383927 383129 383656 384239 385615 387284 4216195  
## 5 381419 383262 386140 387570 389372 390308 4208245  
## 6 437214 438884 441301 443368 445190 448639 4819333  
## 7 580794 582245 584831 587743 591225 594164 6402899  
## 8 482838 485476 490059 493446 496413 500490 5307802

descPopNoWales <- str(df.pop.no.wales)

## 'data.frame': 7 obs. of 13 variables:  
## $ Health Board: chr "Betsi Cadwaladr University Health Board" "Powys Teaching Health Board" "Hywel Dda University Health Board" "Swansea Bay University Health Board" ...  
## $ 2009 : num 684575 133090 379051 375138 430682 ...  
## $ 2010 : num 685911 132878 380195 376949 431423 ...  
## $ 2011 : num 688417 133071 381867 378571 432634 ...  
## $ 2012 : num 689924 133015 383399 379541 434288 ...  
## $ 2013 : num 691180 132786 383833 379975 435710 ...  
## $ 2014 : num 693067 132777 383927 381419 437214 ...  
## $ 2015 : num 693360 132730 383129 383262 438884 ...  
## $ 2016 : num 694826 132337 383656 386140 441301 ...  
## $ 2017 : num 696284 132515 384239 387570 443368 ...  
## $ 2018 : num 698369 132447 385615 389372 445190 ...  
## $ 2019 : num 699559 132435 387284 390308 448639 ...  
## $ Total : num 7615472 1460081 4216195 4208245 4819333 ...

#write.csv(descPopNoWales, "data/tidy/desc-pop-no-wales.csv")  
profiling\_num(df.pop.no.wales)

## variable mean std\_dev variation\_coef p\_01 p\_05 p\_25  
## 1 2009 434124.6 173159.9 0.3988714 147612.9 205704.4 377094.5  
## 2 2010 435710.1 173797.3 0.3988828 147522.3 206099.3 378572.0  
## 3 2011 437679.7 174598.5 0.3989184 147801.0 206721.0 380219.0  
## 4 2012 439152.4 175098.8 0.3987199 147806.6 206972.8 381470.0  
## 5 2013 440344.6 175703.9 0.3990146 147617.3 206942.7 381904.0  
## 6 2014 441719.4 176375.5 0.3992931 147695.5 207369.6 382673.0  
## 7 2015 442726.6 176687.4 0.3990892 147753.9 207849.7 383195.5  
## 8 2016 444735.7 177480.0 0.3990686 147416.1 207732.7 384898.0  
## 9 2017 446452.1 178181.1 0.3991047 147618.4 208032.2 385904.5  
## 10 2018 448375.9 179095.7 0.3994320 147637.1 208397.4 387493.5  
## 11 2019 450411.3 179794.7 0.3991790 147725.9 208889.7 388796.0  
## 12 Total 4861432.4 1939675.6 0.3989926 1624970.8 2284530.2 4212220.0  
## p\_50 p\_75 p\_95 p\_99 skewness kurtosis iqr  
## 1 430682 518168.0 650957.9 677851.6 -0.3212795 2.711161 141073.5  
## 2 431423 521307.5 652571.1 679243.0 -0.3347102 2.712069 142735.5  
## 3 432634 524599.0 655015.0 681736.6 -0.3426468 2.712260 144380.0  
## 4 434288 526950.0 656400.2 683219.2 -0.3544562 2.717188 145480.0  
## 5 435710 529464.0 657629.8 684470.0 -0.3639029 2.714589 147560.0  
## 6 437214 531816.0 659385.1 686330.6 -0.3694750 2.713008 149143.0  
## 7 438884 533860.5 660025.5 686693.1 -0.3810017 2.711027 150665.0  
## 8 441301 537445.0 661827.5 688226.3 -0.4003386 2.714511 152547.0  
## 9 443368 540594.5 663721.7 689771.5 -0.4094833 2.708331 154690.0  
## 10 445190 543819.0 666225.8 691940.4 -0.4179294 2.705951 156325.5  
## 11 448639 547327.0 667940.5 693235.3 -0.4353946 2.705255 158531.0  
## 12 4819333 5855350.5 7251700.1 7542717.6 -0.3763845 2.711816 1643130.5  
## range\_98 range\_80  
## 1 [147612.88, 677851.58] [278318.8, 617340.8]  
## 2 [147522.26, 679243.02] [279320.6, 619231.2]  
## 3 [147801, 681736.6] [280371, 621613]  
## 4 [147806.56, 683219.24] [280930.6, 622876.4]  
## 5 [147617.34, 684469.96] [281099.4, 624079.6]  
## 6 [147695.52, 686330.62] [281962.2, 625703.2]  
## 7 [147753.94, 686693.1] [282969.4, 626691]  
## 8 [147416.14, 688226.3] [283128.4, 628829]  
## 9 [147618.44, 689771.54] [283549.4, 631159.4]  
## 10 [147637.08, 691940.36] [284347.8, 634082.6]  
## 11 [147725.94, 693235.3] [285344.4, 636322]  
## 12 [1624970.84, 7542717.62] [3108979.4, 6887928.2]

# write.csv(profiling\_num(df.pop.no.wales), file = "data/desc-population.csv")

Now lets try to do a time series ( big ask :P ) with the dataset, the data might need to be manipulated even more, but lets see.

# remove health board  
tempD <- df.pop[-1]  
# remove Wales  
tempD <- tempD[-1,]  
# remove Total  
tempD <- tempD[1:11]

# create a time series object as tempD  
tempD %>% ts(start = 2009, frequency = 1) -> tempTS  
flipD <- t(tempD)   
flipNames <- t(slice.pop.areas)   
flipNames <- flipNames[-1]  
flipD <- colnames(flipNames)  
  
# create a time vector  
time <- 1:length(flipD)  
# check the summary  
summary(flipD)

## Length Class Mode   
## 0 NULL NULL

#plot.ts(flipD)

## 3 - Exploring just Aneurin Bevan

This section will be used to try and do something with just the area of Aneurin Bevan hopefully looking at the GPs vs Population estimates in one table and maybe doing something with the time series data.

The rest of this will be continued in a different RMD file, to focus on new data sets to try and join.

df.bevan1 <- rbind(df.pop.no.wales[6,])  
df.bevan2 <- df.no.wales[6,]  
print(paste(round(df.bevan1$Total/df.bevan2$Total), "people need to be seen by 1 doctor."))

## [1] "6797 people need to be seen by 1 doctor."

# add a random entry to the dataframe  
visitsPer1K = vector()   
# visitsPer1K$LHB <- row.names1

# write.csv(df.pop.no.wales, "data/population-estimates.csv")  
# Remove everything which we no longer need to free up some memory/RAM.  
# rm(list = ls())  
# gc()

### - Other datasets to look at:

* [Population by area, ethnicity and gender](https://statswales.gov.wales/Catalogue/Equality-and-Diversity/Ethnicity/Census-2001/Population-by-Area-Ethnicity-Gender)
* [Population estimates by ethnicity and year](https://statswales.gov.wales/Catalogue/Equality-and-Diversity/Equality/Population-Estimates-by-Ethnicity-Year)
* [Population estimates by local authority and ethnicity](https://statswales.gov.wales/Catalogue/Population-and-Migration/Population/Estimates/Ethnicity/PopulationEstimates-by-Localauthority-Ethnicity)
* [Population estimates by local authority, ethnicity and age](https://statswales.gov.wales/Catalogue/Equality-and-Diversity/Ethnicity/Population-Estimates/PopulationEstimates-by-LocalAuthority-Ethnicity-Age)
* [Population estimates by local authority, gender and ethnicity](https://statswales.gov.wales/Catalogue/Equality-and-Diversity/Ethnicity/Population-Estimates/PopulationEstimates-by-LocalAuthority-Gender-Ethnicity)
* [Population estimates by year and National Park](https://statswales.gov.wales/Catalogue/Population-and-Migration/Population/Estimates/National-Park/PopulationEstimates-by-Year-NationalPark)
* [Population projection components of change by year and National park area](https://statswales.gov.wales/Catalogue/Population-and-Migration/Population/Projections/National-Park/2014-Based/populationprojectioncomponentsofchange-by-year-nationalparkarea)
* [Population projections by year and national park](https://statswales.gov.wales/Catalogue/Population-and-Migration/Population/Projections/National-Park/2013-based/populationprojections-by-year-nationalpark)
* [Population projections by year and national park](https://statswales.gov.wales/Catalogue/Population-and-Migration/Population/Projections/National-Park/2014-Based/populationprojections-by-year-nationalpark)
* [Private household population by national park and year](https://statswales.gov.wales/Catalogue/Housing/Households/Projections/National-Park/2013-based/privatehouseholdpopulation-by-nationalpark-year)
* [Private household population by national park and year](https://statswales.gov.wales/Catalogue/Housing/Households/Projections/National-Park/2014-Based/privatehouseholdpopulation-by-nationalpark-year)
* [Projected private household population by local authority and year](https://statswales.gov.wales/Catalogue/Housing/Households/Projections/Local-Authority/2011-Based/ProjectedPrivateHouseholdPopulation-by-LocalAuthority-Year)
* [Projected private household population by year](https://statswales.gov.wales/Catalogue/Housing/Households/Projections/National/2011-Based/ProjectedPrivateHouseholdPopulation-by-Year)