Intro to Data Science - HW 5

Copyright Jeffrey Stanton, Jeffrey Saltz, and Jasmina Tacheva

```
# Enter your name here:  Ryan Tervo
# Course Number:         IST 687
# Assignement Name:      Homework #5
# Due Date:              14 Nov 2022
# Submitted Date:        14 Nov 2022
```

Attribution statement: (choose only one and delete the rest)

```
# 1. I did this homework by myself, with help from the book and the professor.
```

This module: Data visualization is important because many people can make sense of data more easily when it is presented in graphic form. As a data scientist, you will have to present complex data to decision makers in a form that makes the data interpretable for them. From your experience with Excel and other tools, you know that there are a variety of **common data visualizations** (e.g., pie charts). How many of them can you name?

The most powerful tool for data visualization in R is called **ggplot**. Written by computer/data scientist **Hadley Wickham**, this **“graphics grammar”** tool builds visualizations in layers. This method provides immense flexibility, but takes a bit of practice to master.

Step 1: Make a copy of the data

A. Read the **who** dataset from this URL: https://intro-datascience.s3.us-east-2.amazonaws.com/who.csv into a new dataframe called **tb**.

Your new dataframe, tb, contains a so-called **multivariate time series**: a sequence of measurements on 23 Tuberculosis-related (TB) variables captured repeatedly over time (1980-2013). Familiarize yourself with the nature of the 23 variables by consulting the dataset’s codebook which can be found here: https://intro-datascience.s3.us-east-2.amazonaws.com/TB_data_dictionary_2021-02-06.csv.

```
#   IMPORT LIBRARIES:
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.4.0      ✔ purrr   0.3.5
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.2.1      ✔ stringr 1.4.1
## ✔ readr   2.1.3      ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
library(stringr)
```

```
#   DEFINE THE VARIABLES:
fileName <- "https://intro-datascience.s3.us-east-2.amazonaws.com/who.csv"

#   READ EXCEL FILE USING WEBSITE FILE:
tb  <- data.frame(read_csv(fileName))# show_col_types = FALSE))
```

```
## Rows: 5769 Columns: 23
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (1): iso2
## dbl (22): year, new_sp, new_sp_m04, new_sp_m514, new_sp_m014, new_sp_m1524, ...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
tb0 <- tb
```

B. How often were these measurements taken (in other words, at what frequency were the variables measured)? Put your answer in a comment.

```
#   The measurements were taken annually.
#   This can seen in the tb$year column in which data is collected on a yearly or annual basis
.
```

Step 2: Clean-up the NAs and create a subset

A. Let’s clean up the iso2 attribute in **tb**

Hint: use *is.na()* – well use *! is.na()*

```
tb1 <- tb[!is.na(tb$iso2), ]
```

B. Create a subset of **tb** containing **only the records for Canada (“CA” in the iso2 variable)**. Save it in a new dataframe called **tbCan**. Make sure this new df has **29 observations and 23 variables**.

```
#   Create tb2 per the instructions:  only CA
tbCan <- tb1[tb1$iso2 == "CA", ]

#   Verify tb2 has Correct Dimensions:
numRow <- nrow(tbCan)
numCol <- ncol(tbCan)

#   Display the Results:
printString = paste('Dataframe tb2 has ', numRow, ' rows (observations) and ', numCol, ' colum
ns (variables).', sep = '')
print(printString, quote = FALSE)
```

```
## [1] Dataframe tb2 has 29 rows (observations) and 23 columns (variables).
```

C. A simple method for dealing with small amounts of **missing data** in a numeric variable is to **substitute the mean of**

the variable in place of each missing datum.
This expression locates (and reports to the console) all the missing data elements in the variable measuring the **number of positive pulmonary smear tests for male children 0-4 years old** (there are 26 data points missing)

```
tbCan$new_sp_m04[is.na(tbCan$new_sp_m04)]
```

```
##  [1] NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA
## [26] NA
```

```
Error in eval(expr, envir, enclos): object 'tbCan' not found
Traceback:
```

D. Write a comment describing how that statement works.

```
#   Understanding how the statement works is helpful to look at the three individual parts
#   Each part has an output.
#   Part 2 and part 3 have an input.

#          tbCan$new_sp_m04[is.na(tbCan$new_sp_m04)]
#part           3             2           1

#   Part 1:  tbCan$new_sp_m04
#            input:
#            ouput:  a vector of the tbCan$new_sp_mo4 column values.

#   Part 2:  is.na('Part 1')
#            input:   A column of values from tbCan$new_sp_m04
#            output:  A column of values TRUE and FALSE.
             #   For each input element that is 'na' the corresponding output element is TRUE
             #   For each input element that is not 'na' then the corresponding output element
 is FALSE.
             #   In this case the only True elements correspond to elements which have NA, per
 the test.

#   Part 3:  tbCan$new_sp_m04['Part2']
#            input:   A vector of TRUE and FALSE values which has a length as the number of el
ements in tbCan$new_sp_m04
#            output:  A vector of values in "tbCan$new_sp_m04" of only the associated TRUE ele
ments in Part 2.

#   FINAL OUTPUT:  A vector, elements are all na, the length is equal to the number of na's in
 tbCan$new_sp_m04 column.
```

E. Write 4 more statements to check if there is missing data for the number of positive pulmonary smear tests for: **male and female** children 0-14 years old (**new_sp_m014** and **new_sp_f014**), and **male and female citizens 65 years of age and older**, respectively. What does empty output suggest about the number of missing observations?

```
#    Perform Tests
     #   Creates a vector of TRUE and FALSE for each column based on whether or not it is na.
     #   SUMs the TRUE and FALSE statements to get total TRUE for each one.  TRUE = 1 and FALS
E = 0.
```

```
test0 <- is.na(tbCan$new_sp_m04)
test0Results <- sum(test0)

test1 <- is.na(tbCan$new_sp_m014)
test1Results <- sum(test1)

test2 <- is.na(tbCan$new_sp_f014)
test2Results <- sum(test2)

test3 <- is.na(tbCan$new_sp_m65)
test3Results <- sum(test3)

test4 <- is.na(tbCan$new_sp_f65)
test4Results <- sum(test4)

#   Display Output:

printString0 <- paste('The column male    0 - 04       had ', test0Results, ' missing data.',
sep = '')
printString1 <- paste('The column male    0 - 14       had ', test1Results, ' missing data.',
sep = '')
printString2 <- paste('The column female  0 - 14       had ', test2Results, ' missing data.',
sep = '')
printString3 <- paste('The column male    65 and older had ', test3Results, ' missing data.',
sep = '')
printString4 <- paste('The column female  65 and older had ', test4Results, ' missing data.',
sep = '')

print(printString0, quote = FALSE)
```

```
## [1] The column male    0 - 04       had 26 missing data.
```

```
print(printString1, quote = FALSE)
```

```
## [1] The column male    0 - 14       had 0 missing data.
```

```
print(printString2, quote = FALSE)
```

```
## [1] The column female  0 - 14       had 0 missing data.
```

```
print(printString3, quote = FALSE)
```

```
## [1] The column male    65 and older had 0 missing data.
```

```
print(printString4, quote = FALSE)
```

```
## [1] The column female  65 and older had 0 missing data.
```

```
#  What does empty output suggest about the number of missing observations?
  #  This suggests that there are no missing fields or no na's.
```

There is an R package called **imputeTS** specifically designed to repair missing values in time series data. We will use this instead of mean substitution.
The **na_interpolation()** function in this package takes advantage of a unique characteristic of time series data: **neighboring points in time can be used to “guess” about a missing value in between**.

F. Install the **imputeTS** package (if needed) and use **na_interpolation( )** on the variable from part C. Don’t forget that you need to save the results back to the **tbCan** dataframe. Also update any attribute discussed in part E (if needed).

```
library(imputeTS)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
#   COMPLETE PART F USING 'na_interpolation()' function.
tbCan$new_sp_m04 <- na_interpolation(tbCan$new_sp_m04)
```

G. Rerun the code from C and E above to check that all missing data have been fixed.

```
#   Verify tbCan "na's" have been resolved.
test5 <- is.na(tbCan$new_sp_m04)
test5Results <- sum(test5)
#printString5 <- paste('The column had ', test5Results, ' missing data.', sep = '')
#print(printString5, quote = FALSE)

#   RERUN PART C:
tbCan$new_sp_m04[is.na(tbCan$new_sp_m04)]
```

```
## numeric(0)
```

```
#   RERUN PART E:
test0 <- is.na(tbCan$new_sp_m04)
test0Results <- sum(test0)

test1 <- is.na(tbCan$new_sp_m014)
test1Results <- sum(test1)

test2 <- is.na(tbCan$new_sp_f014)
test2Results <- sum(test2)

test3 <- is.na(tbCan$new_sp_m65)
test3Results <- sum(test3)

test4 <- is.na(tbCan$new_sp_f65)
test4Results <- sum(test4)
```

```
#   Display Output:

printString0 <- paste('The column male    0 - 04       had ', test0Results, ' missing data.', 
sep = '')
printString1 <- paste('The column male    0 - 14       had ', test1Results, ' missing data.', 
sep = '')
printString2 <- paste('The column female  0 - 14       had ', test2Results, ' missing data.', 
sep = '')
printString3 <- paste('The column male    65 and older had ', test3Results, ' missing data.', 
sep = '')
printString4 <- paste('The column female  65 and older had ', test4Results, ' missing data.', 
sep = '')

print(printString0, quote = FALSE)
```

```
## [1] The column male    0 - 04       had 0 missing data.
```

```
print(printString1, quote = FALSE)
```

```
## [1] The column male    0 - 14       had 0 missing data.
```

```
print(printString2, quote = FALSE)
```

```
## [1] The column female  0 - 14       had 0 missing data.
```

```
print(printString3, quote = FALSE)
```

```
## [1] The column male    65 and older had 0 missing data.
```

```
print(printString4, quote = FALSE)
```

```
## [1] The column female  65 and older had 0 missing data.
```

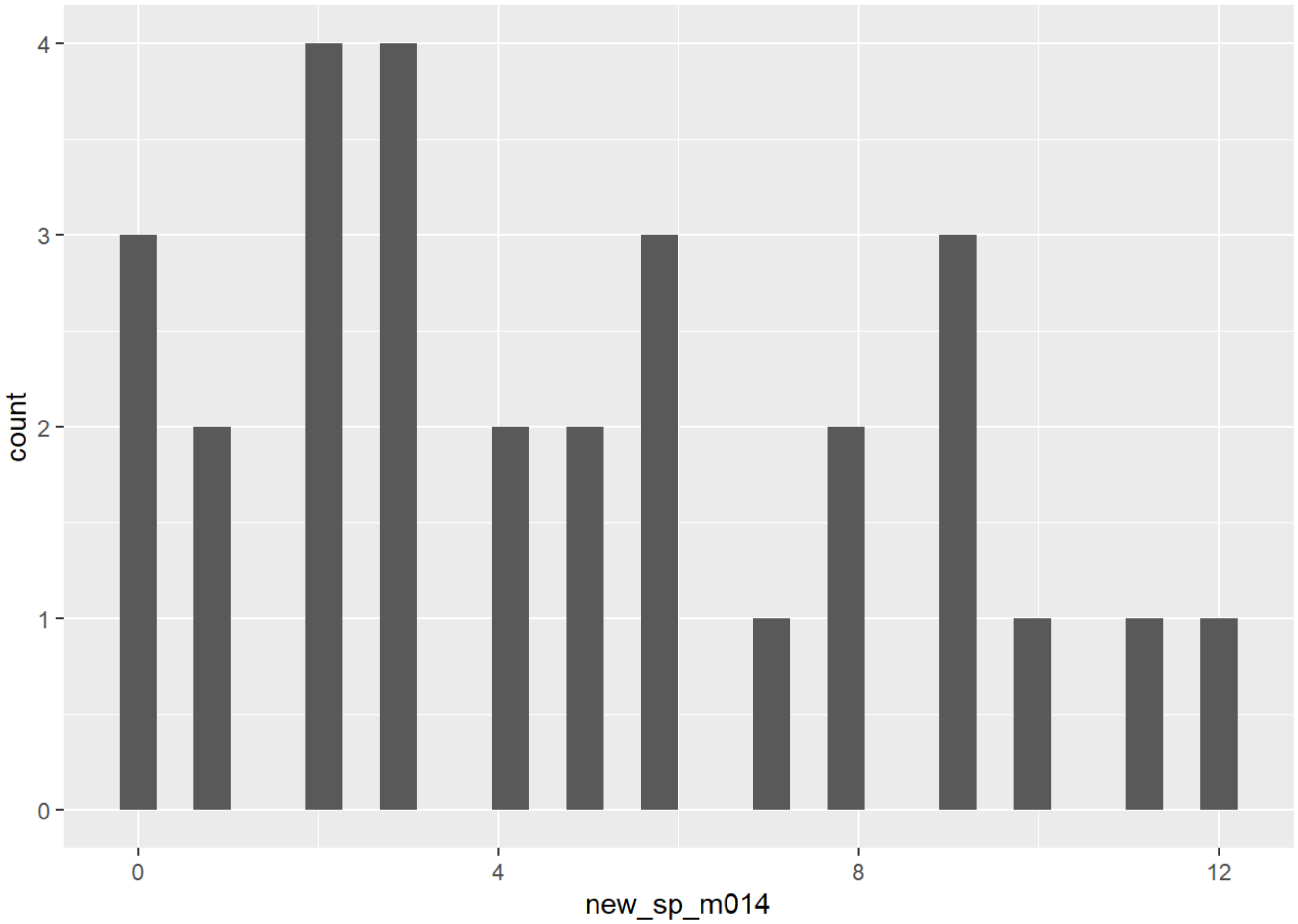
Step 3: Use ggplot to explore the distribution of each variable

Don’t forget to install and library the ggplot2 package. Then:
H. Create a histogram for **new_sp_m014**. Be sure to add a title and briefly describe what the histogram means in a comment.

```
#   INSTALL LIBRARY:
library(ggplot2)

#   CREATE PLOT USING GGPLOT:
plot1 <- ggplot(data = tbCan, aes(x = new_sp_m014)) + geom_histogram()
```

```
#   DISPLAY PLOT:
plot1
```

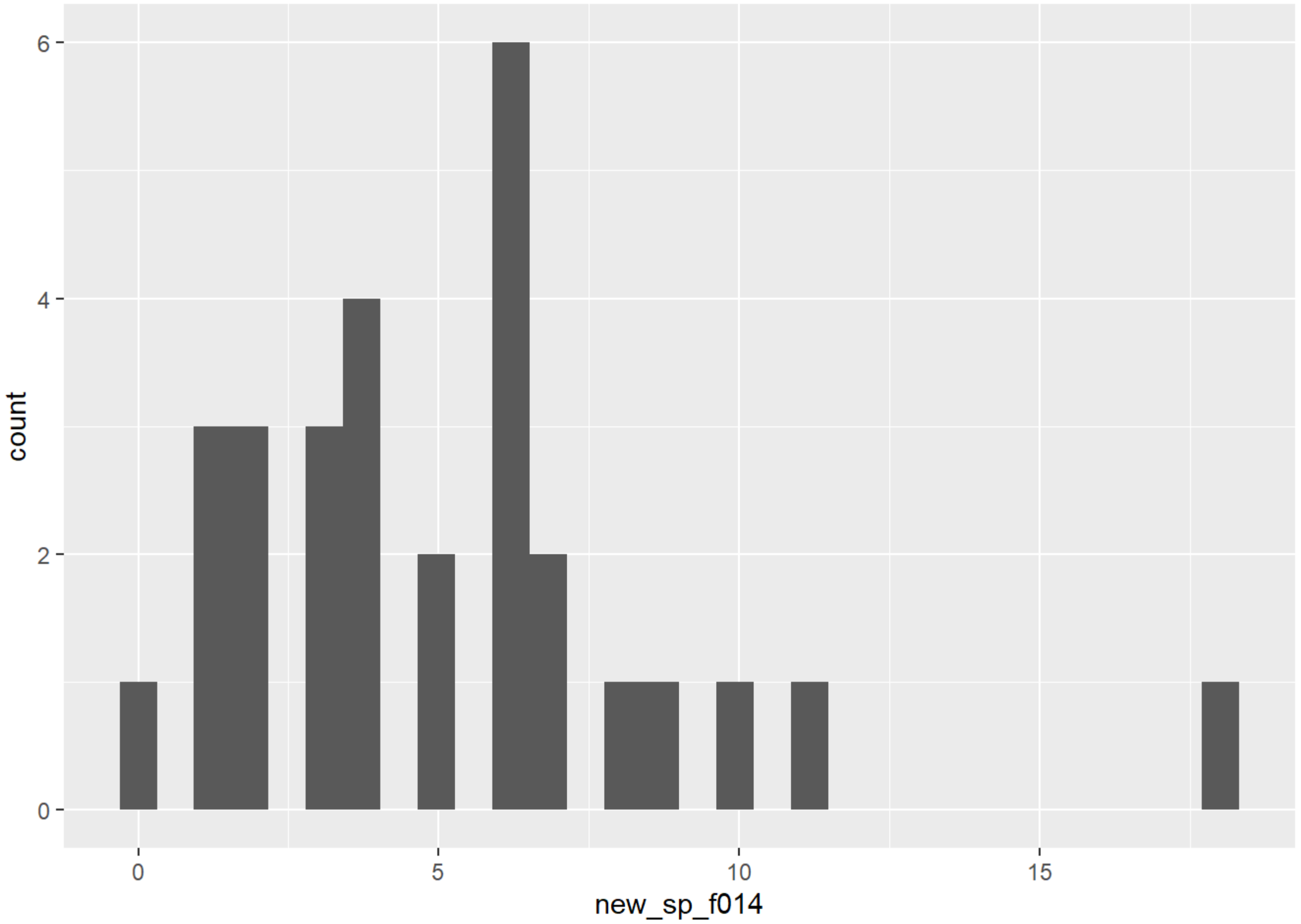
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



I. Create histograms (using ggplot) of each of the other three variables from E with ggplot( ). Which parameter do you need to adjust to make the other histograms look right?

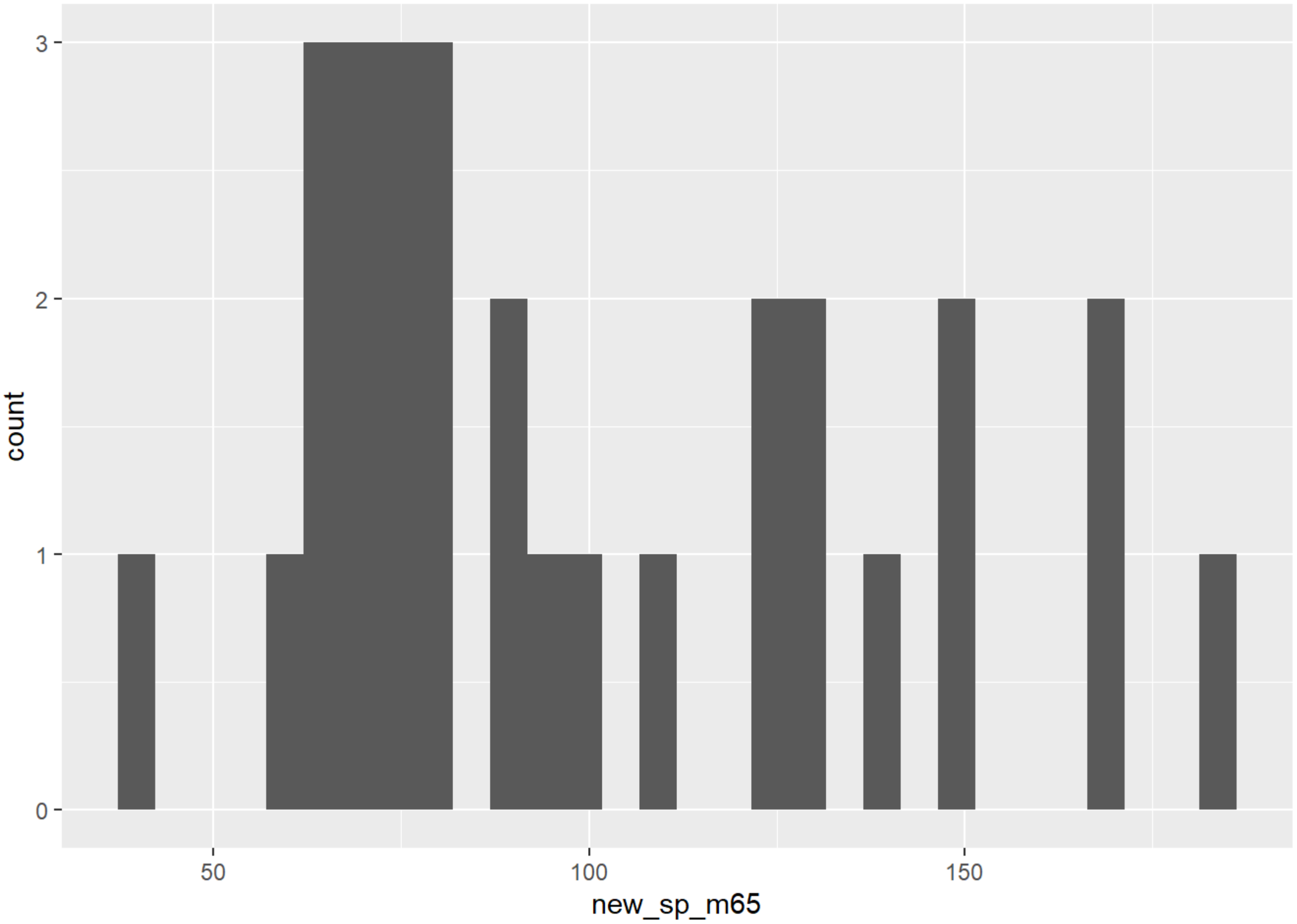
```
#   CREATE HISTOGRAMS USING GGPLOT:
plot2 <- ggplot(data = tbCan, aes(x = new_sp_f014)) + geom_histogram()
plot2
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



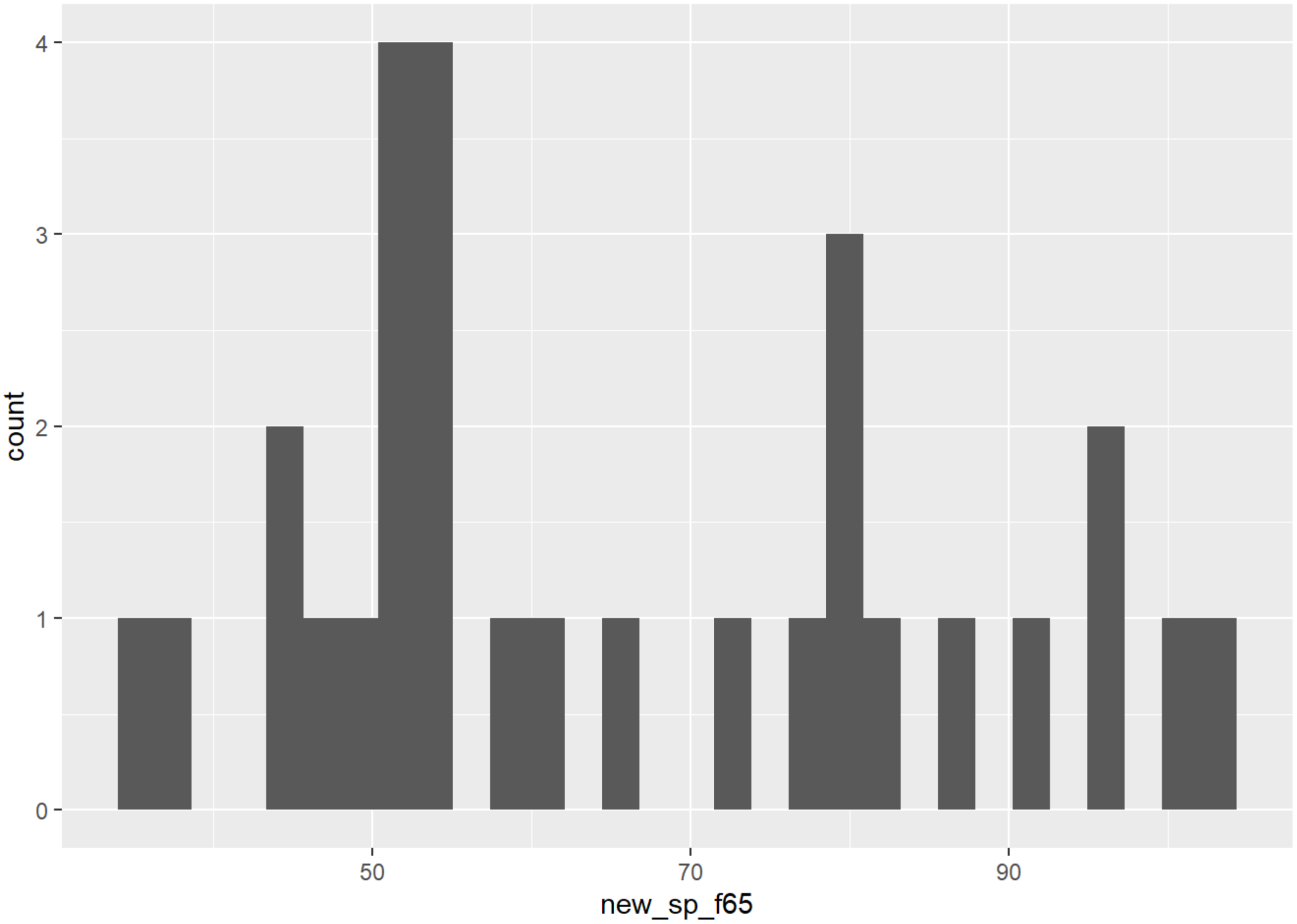
```
plot3 <- ggplot(data = tbCan, aes(x = new_sp_m65)) + geom_histogram()
plot3
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
plot4 <- ggplot(data = tbCan, aes(x = new_sp_f65)) + geom_histogram()
plot4
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

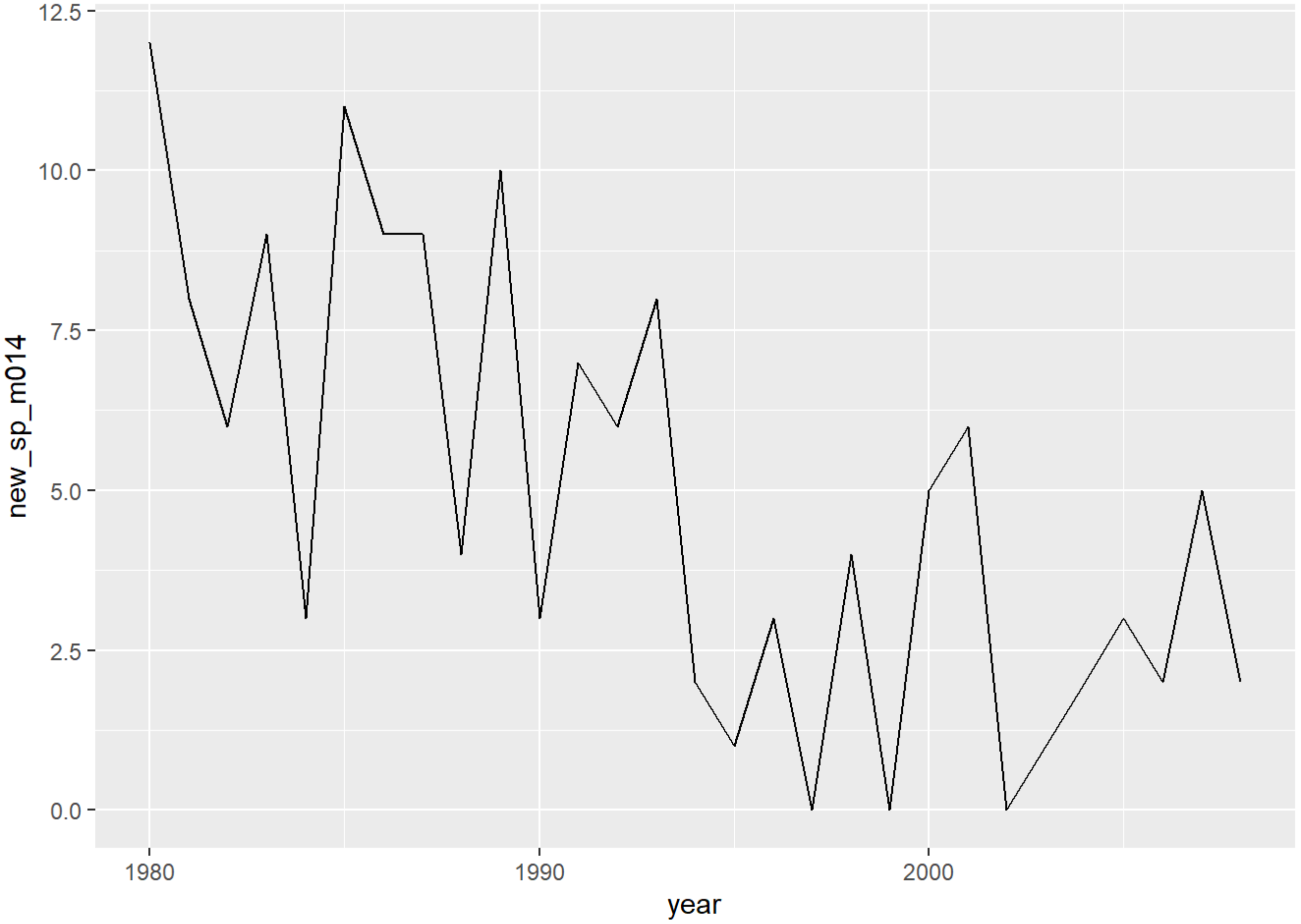


```
#   Which parameter do you need to adjust to make the other histograms look right?
    #   The parameter that needs to be adjusted is the number of bins.
```

Step 4: Explore how the data changes over time

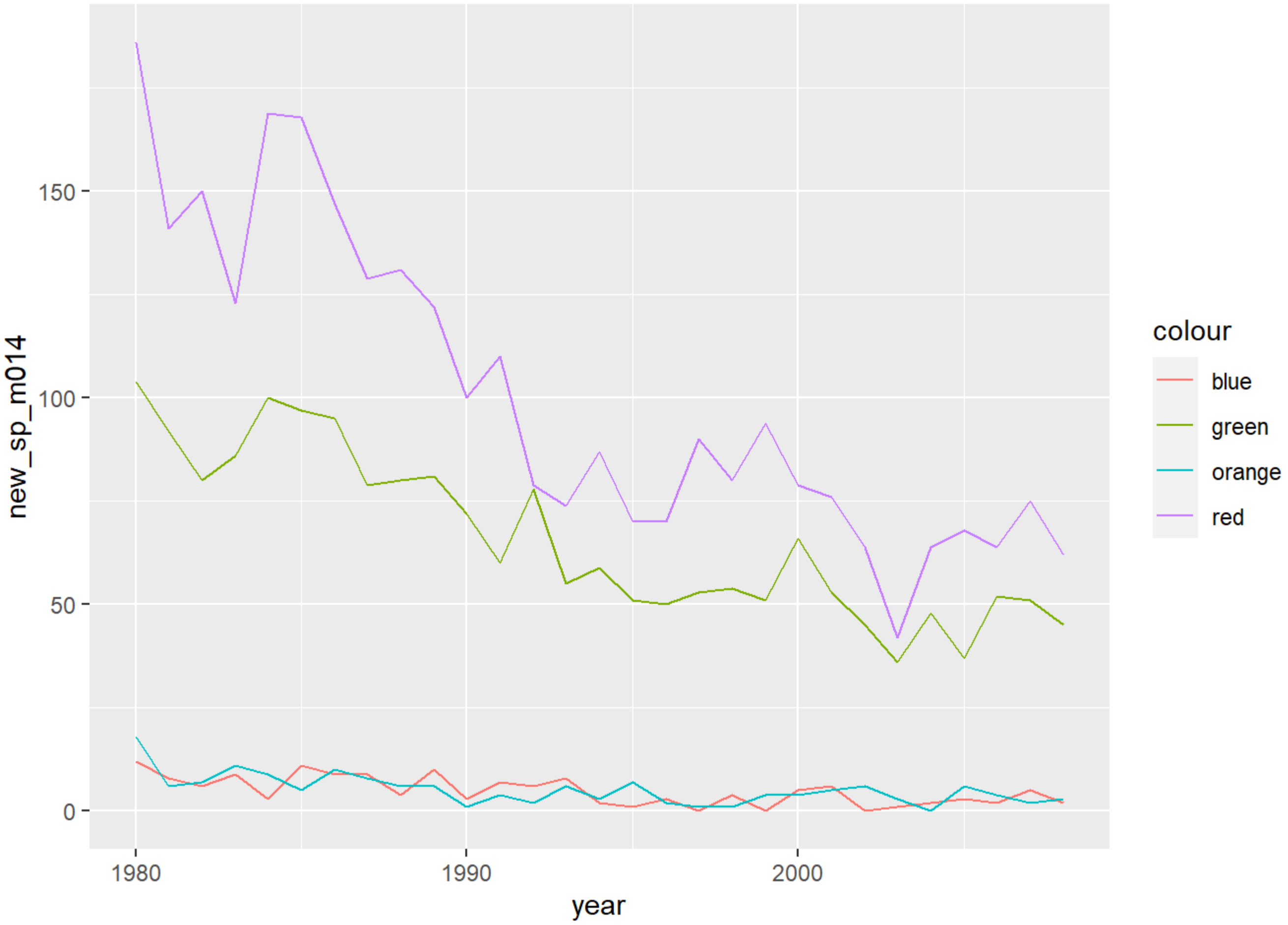
J. These data were collected in a period of several decades (1980-2013). You can thus observe changes over time with the help of a line chart. Create a **line chart**, with **year** on the X-axis and **new_sp_m014** on the Y-axis.

```
#   CREATE GGPLOT:
plot_J <- ggplot(tbCan, aes(x = year)) +  geom_line(aes(y = new_sp_m014))

#   DISPLAY PLOT:
plot_J
```



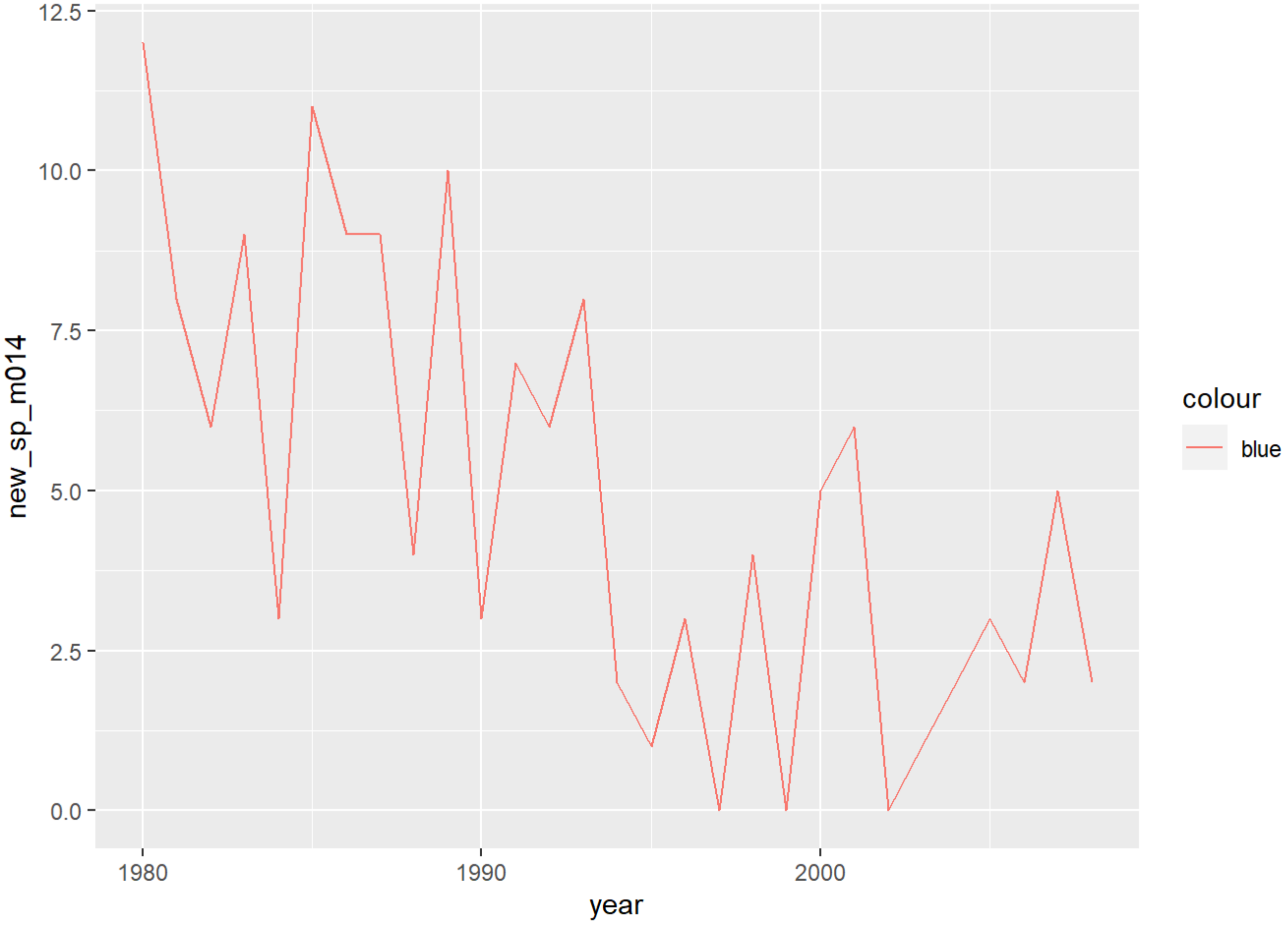
K. Next, create similar graphs for each of the other three variables. Change the **color** of the line plots (any color you want).

```
#   CREATE GGPLOT:  COMBINED
plot_K <- ggplot(tbCan, aes(x = year)) +  geom_line(aes(y = new_sp_m014, colour ="blue")) +
    geom_line(aes(y = new_sp_f014, colour ="orange")) +
    geom_line(aes(y = new_sp_m65, colour = "red")) +
    geom_line(aes(y = new_sp_f65, colour = "green"))

#   CREATE GGPLOT SEPARATE:
plot_K1 <- ggplot(tbCan, aes(x = year)) +  geom_line(aes(y = new_sp_m014, colour ="blue"))
plot_K2 <- ggplot(tbCan, aes(x = year)) +  geom_line(aes(y = new_sp_f014, colour ="blue"))
plot_K3 <- ggplot(tbCan, aes(x = year)) +  geom_line(aes(y = new_sp_m65, colour = "green"))
plot_K4 <- ggplot(tbCan, aes(x = year)) +  geom_line(aes(y = new_sp_f65, colour = "green"))

#   DISPLAY PLOTS:
plot_K
```

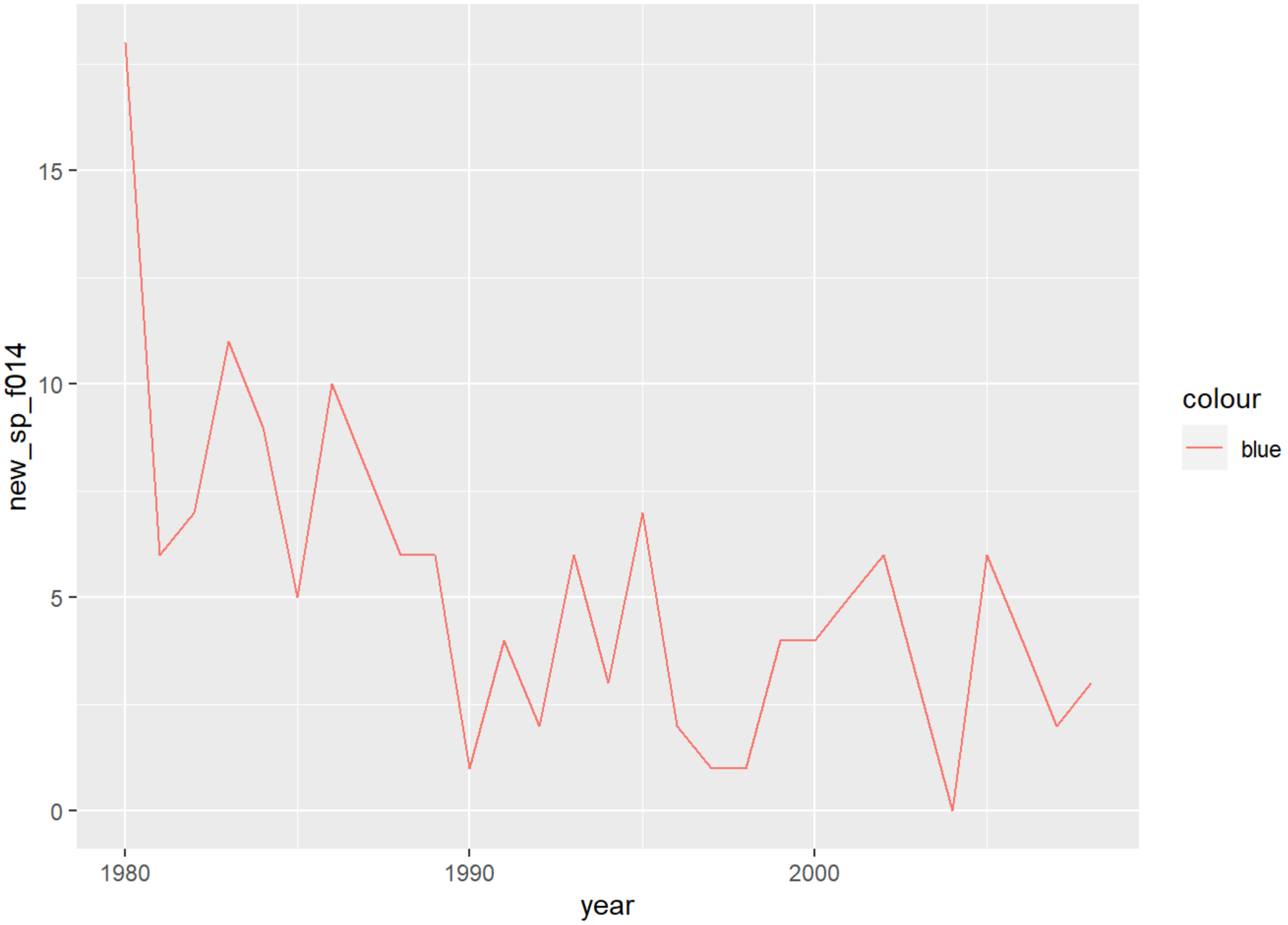


new_sp_m014
150
100
50
0
1980
1990
2000
year
colour
blue
green
orange
red

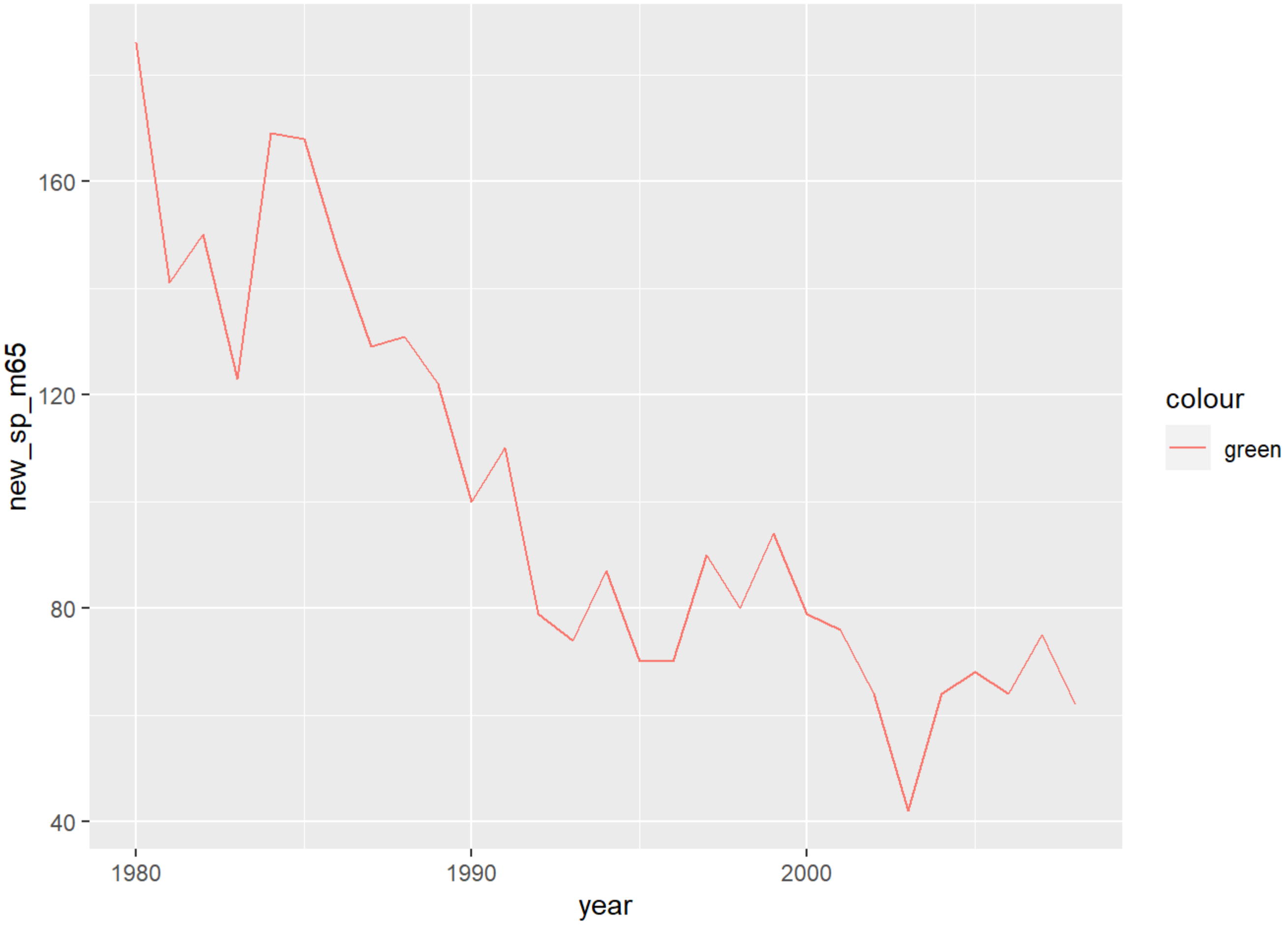
```
plot_K1
```



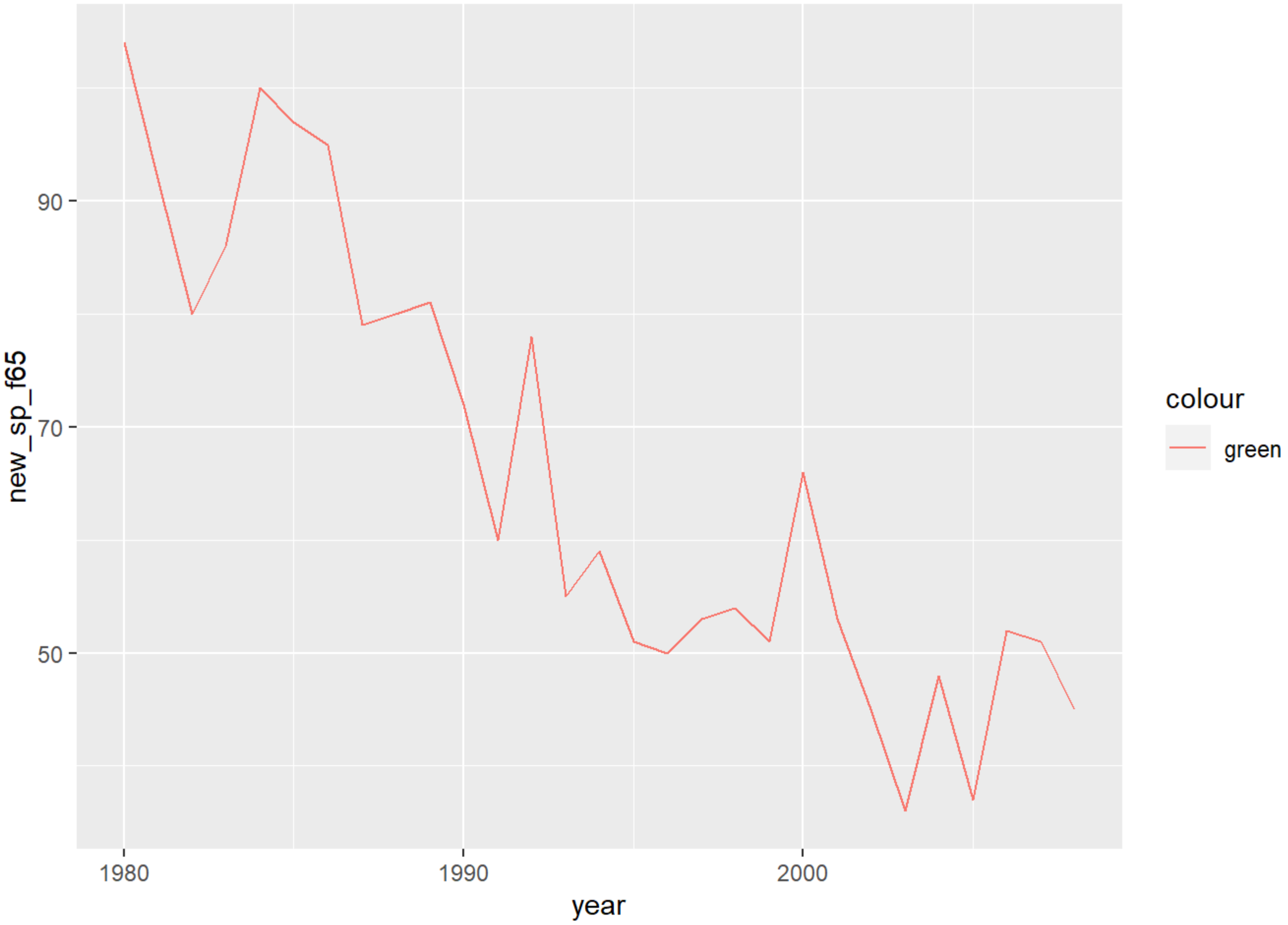
```
plot_K2
```



```
plot_K3
```



```
plot_K4
```



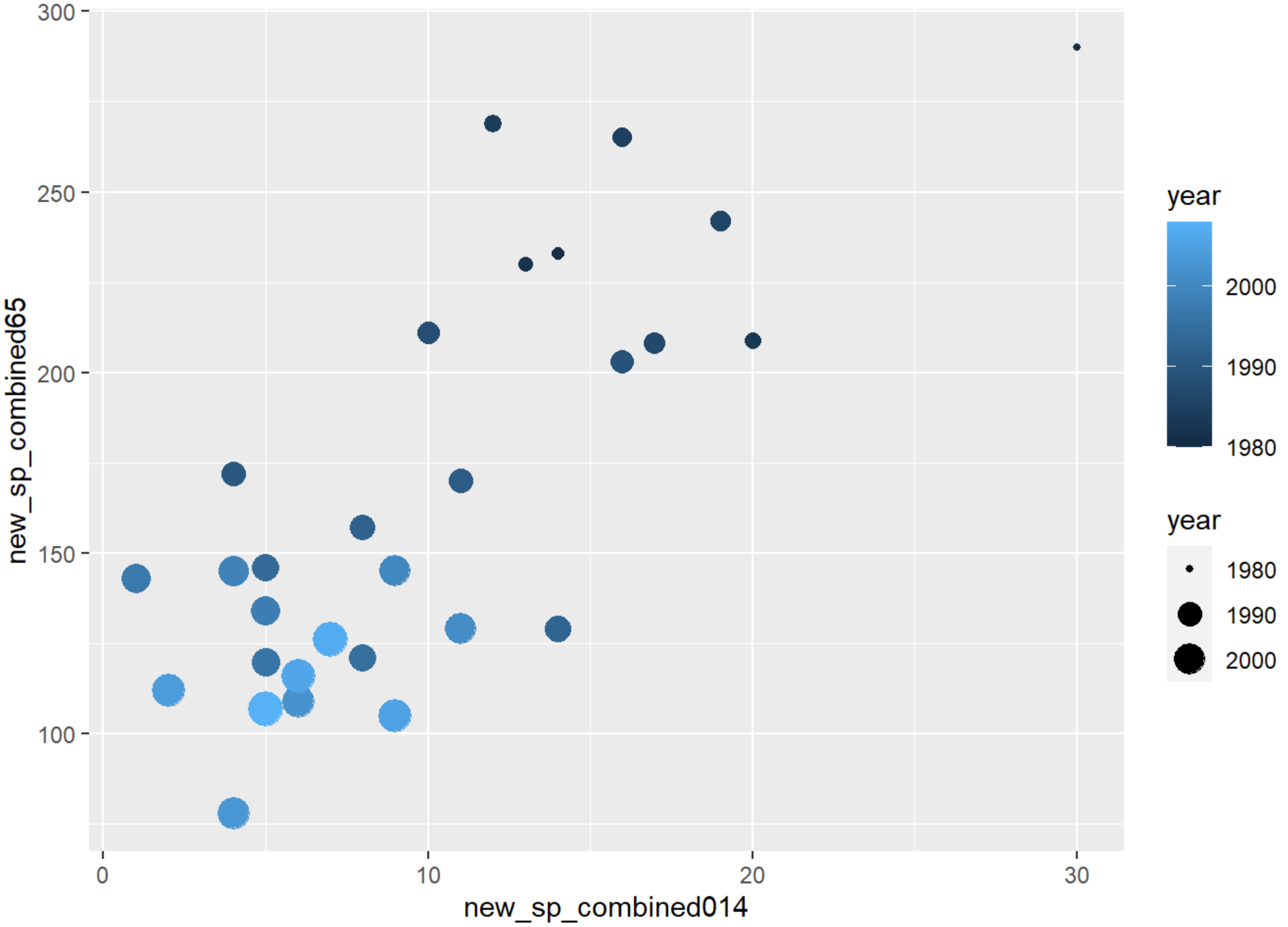
L. Using vector math, create a new variable by combining the numbers from **new_sp_m014** and **new_sp_f014**. Save the resulting vector as a new variable in the **tbCan** df called **new_sp_combined014**. This new variable represents the number of positive pulmonary smear tests for male AND female children between the ages of 0 and 14 years of age. Do the same for SP **tests among citizens 65 years of age and older** and save the resulting vector in the tbCan variable called **new_sp_combined65**.

```
#   CREATE NEW COLUMNS IN tbCan
tbCan$new_sp_combined014 <- tbCan$new_sp_m014 + tbCan$new_sp_f014
tbCan$new_sp_combined65 <- tbCan$new_sp_m65 + tbCan$new_sp_f65

head(tbCan)
```

```
##     iso2 year new_sp new_sp_m04 new_sp_m514 new_sp_m014 new_sp_m1524
## 872   CA 1980    951          1          NA          12           54
## 873   CA 1981    803          1          NA           8           49
## 874   CA 1982    812          1          NA           6           52
## 875   CA 1983    771          1          NA           9           47
## 876   CA 1984    811          1          NA           3           44
## 877   CA 1985    791          1          NA          11           42
##     new_sp_m2534 new_sp_m3544 new_sp_m4554 new_sp_m5564 new_sp_m65 new_sp_mu
## 872           75           83          100          108        186        NA
## 873           61           64           87          103        141        NA
## 874           66           69           90           91        150        NA
## 875           63           62           90           92        123        NA
## 876           75           58           68           83        169        NA
## 877           70           59           77           81        168        NA
##     new_sp_f04 new_sp_f514 new_sp_f014 new_sp_f1524 new_sp_f2534 new_sp_f3544
## 872         NA          NA          18           62           51           34
## 873         NA          NA           6           46           57           26
## 874         NA          NA           7           51           57           30
## 875         NA          NA          11           50           50           29
## 876         NA          NA           9           51           59           28
## 877         NA          NA           5           30           56           19
##     new_sp_f4554 new_sp_f5564 new_sp_f65 new_sp_fu new_sp_combined014
## 872           31           33        104        NA                 30
## 873           28           35         92        NA                 14
## 874           25           38         80        NA                 13
## 875           24           35         86        NA                 20
## 876           28           36        100        NA                 12
## 877           28           48         97        NA                 16
##     new_sp_combined65
## 872               290
## 873               233
## 874               230
## 875               209
## 876               269
## 877               265
```

M. Finally, create a **scatter plot**, showing **new_sp_combined014** on the x axis, **new_sp_combined65** on the y axis, and having the **color and size** of the point represent **year**.

```
#   CREATE GGPLOT SEPARATE:
plot_M <- ggplot(tbCan, aes(x = new_sp_combined014, y = new_sp_combined65)) +  geom_point(aes(
size = year, colour = year))

#   DISPLAY PLOTS:
plot_M
```



N. Interpret this visualization – what insight does it provide?

```
#   INTERPRETATION:
#   At all times there were more cases in the older population than in the younger population.

#   There were more cases in the 2000's than in the 1980's.
```