Manejo de datos - Proyecto Final - Rodrigo López Hernández

Capítulo 13: Análisis de información de documentos semiestructurados/Parsing information from semistructured documents

Objetivo principal del capítulo: Mostrar cómo realizar la recopilación, procesamiento y limpieza de datos de una base de datos climáticos provenientes de archivos de texto descargados desde un servidor FTP, para obtener un archivo ".csv" que sea manejable para cualquier análisis estadístico.

Problemas principales del capítulo: Los datos disponibles en archivos de texto están semiestructurados, es decir, no se pueden incorporar directamente en R debido a su formato heterogéneo y poco fácil de manipular. A diferencia de un .csv éstos archivos de texto no están diseñados/preparados para ser trabajados como "dataset", podemos encontrar varias comas consecutivas, espacios vacíos, columnas con datos junto a más datos, entre otros. Para resolverlo, el enfoque consiste en utilizar R para limpiar y convertir estos archivos en estructuras de datos ordenadas y analizables.

Sin embargo, el servidor FTP, al cual el capítulo hace referencia, fue removido y no es posible consultarlo de la forma en la que la sección 13.1 lo sugiere. Para resolver este problema, se consultó el sitio oficial de datos climáticos que ofrece el Servicio de Conservación de Recursos Naturales del Departamento de Agricultura de los Estados Unidos, particularmente los datos del estado de California. Cada archivo contiene información diaria de datos climáticos desde los años 80's aproximadamente. Las entidades encargadas de recopilar estos datos se llaman "estaciones" y están dispersas a lo largo del estado de california, el libro hace uso de la información de seis estaciones: ADIN MTN, INDEPENDENCE CAMP, SQUAW VALLEY, SPRATT CREEK, LEAVITT MEADOWS, POISON FLAT. Para el caso de SQUAW VALLEY, ésta se sustituyó por HAGAN'S MEADOW dado que la información de SQUAW VALLEY ya no existe.

En este Notebook únicamente se trabajó con la estación HAGAN'S MEADOW por practicidad de la extensión del código. Sin embargo, se puede trabajar perfectamente con cualquier estación. Para hacerlo sólo es necesario descargar el .txt correspondiente y cambiar la línea de código que se encarga de leer dicho .txt., así como los encabezados de los gráficos correspondiente, la estructura de la limpieza es la misma. A continuación se explica cómo se hizo la limpieza y tratamiento de los datos.

Descargar los archivos del sitio web

- Accedemos al sitio oficial: https://wcc.sc.egov.usda.gov/nwcc/rgrpt?report=temperature_hist&state=CA.
- Seleccionamos la estación correspondiente, en el caso del libro sugiere "HAGAN'S MEADOW".
- El sitio imprimirá en la pantalla datos con respecto al clima. Dado que no se proporciona una manera de descargarlo en ningún formato, se optó por copiar el contenido impreso y colocarlo en archivo .txt.
- El archivo .txt se nombró como "data1.txt", posterior a esto se comienza el tratamiento en R.

In [189…

```
# Creamos un vector de líneas.
# Cada línea de texto del archivo "data1.txt" es un elemento del vector Vlíneas.

Vlineas <- readLines("data1.txt")
cat("Los elementos de Vlineas es: ", length(Vlineas), "\n")

print(head(Vlineas))
```

```
Los elementos de Vlineas es:  1272
[1] "#------------------------------------------------- WARNING --------------------------------------------"
[2] "# "
[3] "# The data you have obtained from this automated Natural Resources Conservation Service "
[4] "# database are subject to revision regardless of indicated Quality Assurance level. "
[5] "# Data are released on condition that neither the NRCS nor the United States Government "
[6] "# may be held liable for any damages resulting from its use. "
```

In [190…

```
# En el archivo de texto hay elementos que no nos sirven por ejemplo, las líneas vacías o las líneas que comienzan por "#"
# donde sólo hay descripción de la información de los datos climáticos. Éstos elementos deben ser eliminados del archivo
# de texto para poder manipular correctamente los datos.

# Para hacerlo empleamos la función "grepl( - )" para buscar patrones de texto en cada elemento del vector "Vlineas".
# Vamos a conservar aquellas líneas que NO cumplan con la condición de que:
#  - "\\s*" (tenga cero o mas veces espacios en blanco) ó
#  - "#" (tenga un "#" al inicio de la línea)

VlineasL <- Vlineas[!grepl("^\\s*$|^#", Vlineas)]
cat("Los elementos de VlineasL es: ", length(VlineasL), "\n")

print(head(VlineasL))
```

```
Los elementos de VlineasL es:  1211
[1] "Water Year,Day,Oct,Oct,Oct,Nov,Nov,Nov,Dec,Dec,Dec,Jan,Jan,Jan,Feb,Feb,Feb,Mar,Mar,Mar,Apr,Apr,Apr,May,May,May,Jun,Jun,Jun,Jul,Jul,Jul,Aug,Aug,Aug,Sep,Sep,Sep"
[2] ",,Air Temperature Minimum (degF),Air Temperature Maximum (degF),Air Temperature Average (degF),Air Temperature Minimum (degF),Air Temperature Maximum (degF),Air Temperature Average (degF),Air Temperature Minimum (degF),Air Tempera
ture Maximum (degF),Air Temperature Average (degF),Air Temperature Minimum (degF),Air Temperature Maximum (degF),Air Temperature Average (degF),Air Temperature Minimum (degF),Air Temperature Maximum (degF),Air Temperature Average (deg
F),Air Temperature Minimum (degF),Air Temperature Maximum (degF),Air Temperature Average (degF),Air Temperature Minimum (degF),Air Temperature Maximum (degF),Air Temperature Average (degF),Air Temperature Minimum (degF),Air Temperature
Maximum (degF),Air Temperature Average (degF),Air Temperature Minimum (degF),Air Temperature Maximum (degF),Air Temperature Average (degF),Air Temperature Minimum (degF),Air Temperature Maximum (degF),Air Temperature Average (degF),Air
Temperature Minimum (degF),Air Temperature Maximum (degF),Air Temperature Average (degF),Air Temperature Minimum (degF),Air Temperature Maximum (degF),Air Temperature Average (degF)"
[3] "1987,01,32.0,32.0,32.0,32.0,32.0,32.0,32.0,32.0,32.0,,,,32.0,32.0,32.0,,,,,,,32.0,32.0,32.0,,,,33.6,,52.3,37.6,80.6,56.8,40.5,75.4,52.7"
[4] "1987,02,32.0,32.0,32.0,32.0,32.0,32.0,32.0,32.0,32.0,,,,,,,,,,,,,,,,,,,,,39.6,82.6,59.4,37.9,71.8,52.7"
[5] "1987,03,32.0,32.0,32.0,32.0,32.0,32.0,32.0,32.0,32.0,,,,,,,,,,32.0,32.0,32.0,32.0,32.0,32.0,,,,,,,40.1,81.1,59.5,37.0,70.2,51.8"
[6] "1987,04,32.0,32.0,32.0,32.0,32.0,32.0,32.0,32.0,32.0,,,,,,,,,,,,,32.0,32.0,32.0,,,,,,,-36.4,84.7,59.9,33.8,66.6,49.3"
```

In [191…

```
# Una vez que ya se limpiaron las líneas en blanco, vamos a preparar el archivo para ser manipulado. Vamos a extraer los años
# del archivo de texto. Empleamos nuevamente la función "grepl( - , TRUE)" para extraer las líneas completas que coincidan con el patrón:
# - \\d{4} (dígitos numéricos que tengan exactamente 4 posiciones, los años).

Danho <- grep("^\\d{4}", VlineasL, value = TRUE)
```

In [192…

```
# Separamos los años por comas.

Dcomas <- strsplit(Danho, split = ",")
print(head(Dcomas))
```

```
[[1]]
 [1] "1987" "01"   "32.0" "32.0" "32.0" "32.0" "32.0" "32.0" "32.0" "32.0"
[11] "32.0" ""     ""     ""     "32.0" "32.0" "32.0" ""     ""     ""
[21] ""     ""     ""     "32.0" "32.0" "32.0" ""     ""     ""     "33.6"
[31] ""     "52.3" "37.6" "80.6" "56.8" "40.5" "75.4" "52.7"

[[2]]
 [1] "1987" "02"   "32.0" "32.0" "32.0" "32.0" "32.0" "32.0" "32.0" "32.0"
[11] "32.0" ""     ""     ""     ""     ""     ""     ""     ""     ""
[21] ""     ""     ""     ""     ""     ""     ""     ""     ""     ""
[31] ""     ""     "39.6" "82.6" "59.4" "37.9" "71.8" "52.7"

[[3]]
 [1] "1987" "03"   "32.0" "32.0" "32.0" "32.0" "32.0" "32.0" "32.0" "32.0"
[11] "32.0" ""     ""     ""     ""     ""     ""     ""     ""     ""
[21] "32.0" "32.0" "32.0" "32.0" "32.0" "32.0" ""     ""     ""     ""
[31] ""     ""     "40.1" "81.1" "59.5" "37.0" "70.2" "51.8"

[[4]]
 [1] "1987"  "04"    "32.0"  "32.0"  "32.0"  "32.0"  "32.0"  "32.0"  "32.0"
[10] "32.0"  "32.0"  ""      ""      ""      ""      ""      ""      ""
[19] ""      ""      ""      ""      ""      "32.0"  "32.0"  "32.0"  ""
[28] ""      ""      ""      ""      ""      "-36.4" "84.7"  "59.9"  "33.8"
[37] "66.6"  "49.3"

[[5]]
 [1] "1987" "05"   "32.0" "32.0" "32.0" "32.0" "32.0" "32.0" "32.0" "32.0"
[11] "32.0" ""     ""     ""     ""     ""     ""     ""     ""     ""
[21] "32.0" "32.0" "32.0" "32.0" "32.0" "32.0" ""     ""     ""     ""
[31] ""     ""     "42.8" "83.1" "60.6" "36.0" "71.2" "50.7"

[[6]]
 [1] "1987" "06"   "32.0" "32.0" "32.0" ""     ""     ""     ""     ""
[11] ""     ""     ""     ""     ""     ""     ""     ""     ""     ""
[21] "32.0" "32.0" "32.0" ""     ""     ""     ""     ""     ""     ""
[31] ""     ""     "42.3" "82.8" "59.9" "33.6" "71.6" "49.5"
```

In [193…

```
# Con los datos separador por comas, ahora vamos a elaborar un dataframe (df) para una mejor manipulación de los datos.
# Vamos a emplear la función "do.call( - )" para unir una lista de vectores en una sola matriz Así mismo, usaremos la
# función rbind( - ) para unir por filas dichos vectores en una sola matriz
# Después emplearemos as.data.frame( - ) para convertir la matriz en una tabla. "stringsAsFactors = FALSE" evita que las variables se vuelvan
# categoricas.

df <- do.call(rbind, Dcomas)
df <- as.data.frame(df, stringsAsFactors = FALSE)
print(head(df))
```

```
Warning message in (function (..., deparse.level = 1) :
"number of columns of result is not a multiple of vector length (arg 26)"
    V1 V2   V3   V4   V5   V6   V7   V8   V9  V10  V11 V12 V13 V14  V15  V16
1 1987 01 32.0 32.0 32.0 32.0 32.0 32.0 32.0 32.0 32.0             32.0 32.0
2 1987 02 32.0 32.0 32.0 32.0 32.0 32.0 32.0 32.0 32.0
3 1987 03 32.0 32.0 32.0 32.0 32.0 32.0 32.0 32.0 32.0
4 1987 04 32.0 32.0 32.0 32.0 32.0 32.0 32.0 32.0 32.0
5 1987 05 32.0 32.0 32.0 32.0 32.0 32.0 32.0 32.0 32.0
6 1987 06 32.0 32.0 32.0
   V17 V18 V19 V20  V21  V22  V23  V24  V25  V26 V27 V28 V29  V30 V31  V32
1 32.0                             32.0 32.0 32.0             33.6     52.3
2
3                   32.0 32.0 32.0 32.0 32.0 32.0
4                                  32.0 32.0 32.0
5                   32.0 32.0 32.0 32.0 32.0 32.0
6                   32.0 32.0 32.0
    V33  V34  V35  V36  V37  V38
1  37.6 80.6 56.8 40.5 75.4 52.7
2  39.6 82.6 59.4 37.9 71.8 52.7
3  40.1 81.1 59.5 37.0 70.2 51.8
4 -36.4 84.7 59.9 33.8 66.6 49.3
5  42.8 83.1 60.6 36.0 71.2 50.7
6  42.3 82.8 59.9 33.6 71.6 49.5
```

In [194…

```
# Para dar forma al data frame, damos nombre a las columnas.

colnames(df) <- c("Año", "Día", "Oct_Min", "Oct_Max", "Oct_Prom", "Nov_Min", "Nov_Max", "Nov_Prom",
                        "Dic_Min", "Dic_Max", "Dic_Prom", "Ene_Min", "Ene_Max", "Ene_Prom", "Feb_Min",
                        "Feb_Max", "Feb_Prom", "Mar_Min", "Mar_Max", "Mar_Prom", "Abr_Min", "Abr_Max",
                        "Abr_Prom", "May_Min", "May_Max", "May_Prom", "Jun_Min", "Jun_Max", "Jun_Prom",
                        "Jul_Min", "Jul_Max", "Jul_Prom", "Ago_Min", "Ago_Max", "Ago_Prom",
                        "Sep_Min", "Sep_Max", "Sep_Prom")
print(head(df))
```

```
   Año Día Oct_Min Oct_Max Oct_Prom Nov_Min Nov_Max Nov_Prom Dic_Min Dic_Max
1 1987  01    32.0    32.0     32.0    32.0    32.0     32.0    32.0    32.0
2 1987  02    32.0    32.0     32.0    32.0    32.0     32.0    32.0    32.0
3 1987  03    32.0    32.0     32.0    32.0    32.0     32.0    32.0    32.0
4 1987  04    32.0    32.0     32.0    32.0    32.0     32.0    32.0    32.0
5 1987  05    32.0    32.0     32.0    32.0    32.0     32.0    32.0    32.0
6 1987  06    32.0    32.0     32.0
  Dic_Prom Ene_Min Ene_Max Ene_Prom Feb_Min Feb_Max Feb_Prom Mar_Min Mar_Max
1     32.0                             32.0    32.0     32.0
2     32.0
3     32.0
4     32.0
5     32.0
6
  Mar_Prom Abr_Min Abr_Max Abr_Prom May_Min May_Max May_Prom Jun_Min Jun_Max
1                                      32.0    32.0     32.0
2
3             32.0    32.0     32.0    32.0    32.0     32.0
4                                      32.0    32.0     32.0
5             32.0    32.0     32.0    32.0    32.0     32.0
6             32.0    32.0     32.0
  Jun_Prom Jul_Min Jul_Max Jul_Prom Ago_Min Ago_Max Ago_Prom Sep_Min Sep_Max
1             33.6             52.3    37.6    80.6     56.8    40.5    75.4
2                                      39.6    82.6     59.4    37.9    71.8
3                                      40.1    81.1     59.5    37.0    70.2
4                                     -36.4    84.7     59.9    33.8    66.6
5                                      42.8    83.1     60.6    36.0    71.2
6                                      42.3    82.8     59.9    33.6    71.6
  Sep_Prom
1     52.7
2     52.7
3     51.8
4     49.3
5     50.7
6     49.5
```

In [195…

```
# Vamos a convertir los valores del dataframe a valores con formato numérico, aquellas entradas vacías del dataframe se les
# asignará "NA".

for (i in 1:ncol(df)) {
  df[[i]] <- as.numeric(df[[i]])
}

print(head(df))
```

```
   Año Día Oct_Min Oct_Max Oct_Prom Nov_Min Nov_Max Nov_Prom Dic_Min Dic_Max
1 1987   1      32      32       32      32      32       32      32      32
2 1987   2      32      32       32      32      32       32      32      32
3 1987   3      32      32       32      32      32       32      32      32
4 1987   4      32      32       32      32      32       32      32      32
5 1987   5      32      32       32      32      32       32      32      32
6 1987   6      32      32       32      NA      NA       NA      NA      NA
  Dic_Prom Ene_Min Ene_Max Ene_Prom Feb_Min Feb_Max Feb_Prom Mar_Min Mar_Max
1       32      NA      NA       NA      32      32       32      NA      NA
2       32      NA      NA       NA      NA      NA       NA      NA      NA
3       32      NA      NA       NA      NA      NA       NA      NA      NA
4       32      NA      NA       NA      NA      NA       NA      NA      NA
5       32      NA      NA       NA      NA      NA       NA      NA      NA
6       NA      NA      NA       NA      NA      NA       NA      NA      NA
  Mar_Prom Abr_Min Abr_Max Abr_Prom May_Min May_Max May_Prom Jun_Min Jun_Max
1       NA      NA      NA       NA      32      32       32      NA      NA
2       NA      NA      NA       NA      NA      NA       NA      NA      NA
3       NA      32      32       32      32      32       32      NA      NA
4       NA      NA      NA       NA      32      32       32      NA      NA
5       NA      32      32       32      32      32       32      NA      NA
6       NA      32      32       32      NA      NA       NA      NA      NA
  Jun_Prom Jul_Min Jul_Max Jul_Prom Ago_Min Ago_Max Ago_Prom Sep_Min Sep_Max
1       NA    33.6      NA     52.3    37.6    80.6     56.8    40.5    75.4
2       NA      NA      NA       NA    39.6    82.6     59.4    37.9    71.8
3       NA      NA      NA       NA    40.1    81.1     59.5    37.0    70.2
4       NA      NA      NA       NA   -36.4    84.7     59.9    33.8    66.6
5       NA      NA      NA       NA    42.8    83.1     60.6    36.0    71.2
6       NA      NA      NA       NA    42.3    82.8     59.9    33.6    71.6
  Sep_Prom
1     52.7
2     52.7
3     51.8
4     49.3
5     50.7
6     49.5
```

In [196…

```
# Exportamos un CSV

write.csv(df, "base_limpia1.csv", row.names = FALSE)
```

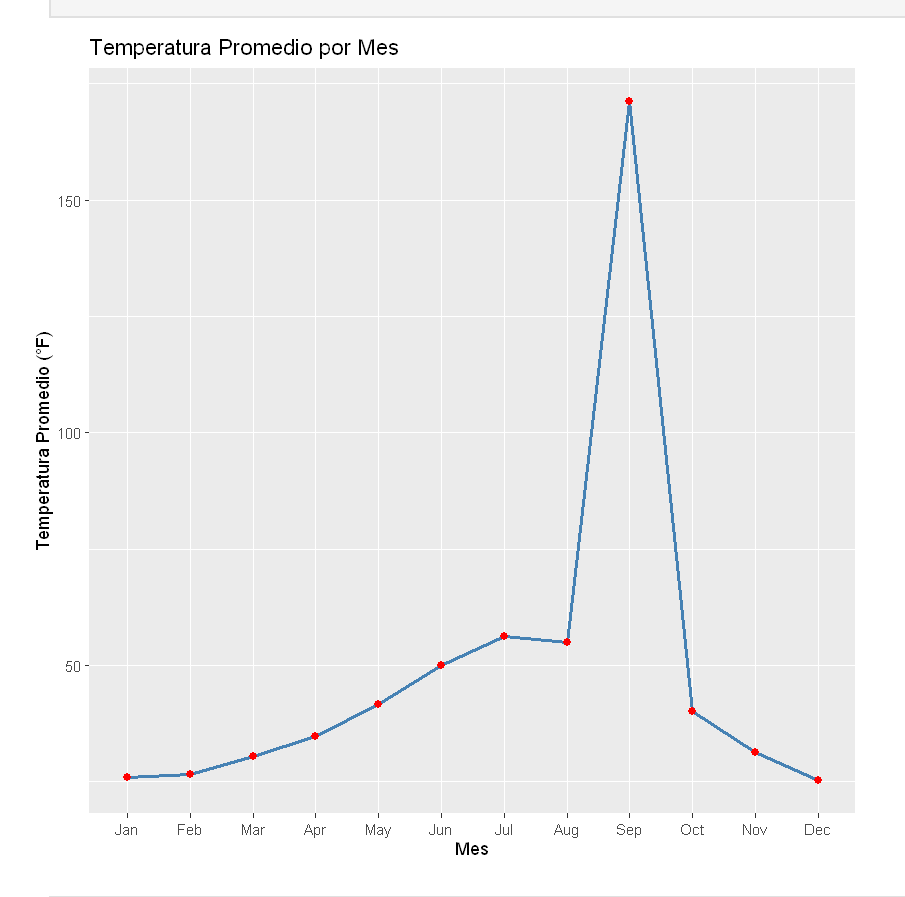
In [197…

```
# Calcular promedio mensual para todos los años.
# Empleamos la función "grep( - )" para buscar las columnas que posean valores de promedios.
# Asignamos un valor de "na.rm = TRUE" para descartar los valores con "NA".

prom_mensual <- colMeans(df[, grep("_Prom", colnames(df))], na.rm = TRUE)
print(prom_mensual)

# Reorganizamos el vector de promedios dado que nuestros valores comienzan por el mes de octubre y es conveniente hacerlo con enero.
meses_ordenados <- c("Ene_Prom", "Feb_Prom", "Mar_Prom", "Abr_Prom", "May_Prom", "Jun_Prom",
                     "Jul_Prom", "Ago_Prom", "Sep_Prom", "Oct_Prom", "Nov_Prom", "Dic_Prom")
prom_mensual_ordenado <- prom_mensual[meses_ordenados]
print(prom_mensual_ordenado)

# Creamos un dataframe para graficar
meses <- 1:12
df_grafica <- data.frame(
  Mes = factor(month.abb, levels = month.abb),
  Temperatura_Promedio = prom_mensual_ordenado
)
```

```
Oct_Prom  Nov_Prom  Dic_Prom  Ene_Prom  Feb_Prom  Mar_Prom  Abr_Prom  May_Prom
39.98968  31.15052  25.31684  25.94289  26.53100  30.40391  34.73292  41.50418
Jun_Prom  Jul_Prom  Ago_Prom  Sep_Prom
50.04781  56.18789  54.97421 171.12440
Ene_Prom  Feb_Prom  Mar_Prom  Abr_Prom  May_Prom  Jun_Prom  Jul_Prom  Ago_Prom
25.94289  26.53100  30.40391  34.73292  41.50418  50.04781  56.18789  54.97421
Sep_Prom  Oct_Prom  Nov_Prom  Dic_Prom
171.12440  39.98968  31.15052  25.31684
```

In [198…

```
# Graficamos los datos obtenidos
library(ggplot2)
ggplot(df_grafica, aes(x = Mes, y = Temperatura_Promedio)) +
  geom_line(group = 1, color = "steelblue", size = 1) +
  geom_point(color = "red", size = 2) +
  labs(title = "Temperatura Promedio por Mes",
       x = "Mes",
       y = "Temperatura Promedio (°F)")
```



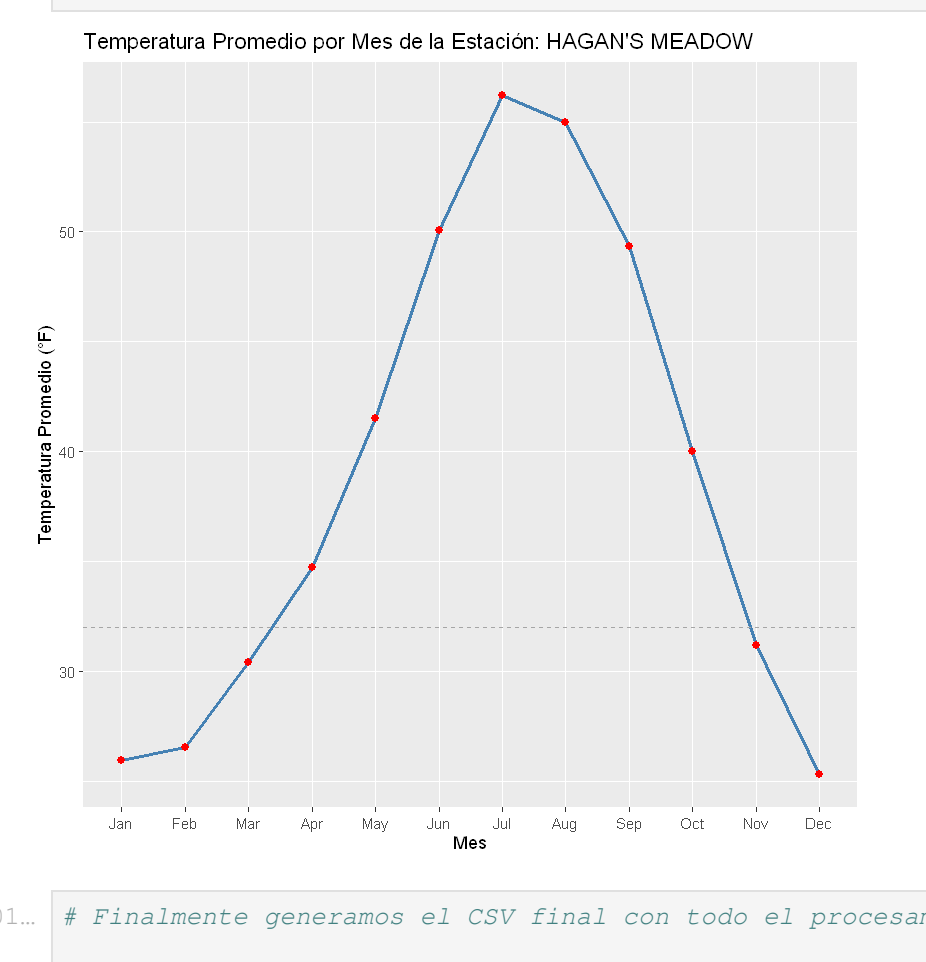
In [199…

```
# Vemos ahora un punto atípico en la gráfica, esto se debe a un llenado erroneo de las celdas en el CSV generado.
# Al parecer, para el mes de septiembre, cuando las variables de temperatura mínima y máxima toman un valor de "NA",
# se autorellena con el valor de año correspondiente, es decir, 1990, 2020, 2025, entre otros. Por ello el valor del promedio
# es demasiado alto. Para esto vamos a descartar todos estos valores atípicos.
# Vamos entones a detectar valores con exactamente 4 dígitos sin decimales en la columna Sep_Prom

filtro_4_digitos <- grepl("^\\d{4}$", df$Sep_Prom)
df$Sep_Prom[filtro_4_digitos] <- NA
print(df$Sep_Prom)
```

```
   [1] 52.7 52.7 51.8 49.3 50.7 49.5 49.1 51.3 51.1 48.9 48.9 49.3 48.2 48.2
  [15] 46.6 45.0 46.9 48.7 50.4 52.7 57.2 55.9 52.2 50.7 52.9   NA 34.2 50.2
  [29] 49.1 49.3   NA 57.0 59.0 59.5 60.6 59.9 59.2 59.2 56.8 52.0 50.9 44.8
  [43] 44.1 43.7 48.0 48.7 47.3 43.3 42.6 46.2 37.9 41.2 43.2 44.6 45.1 50.0
  [57] 47.7 44.4 48.9 48.9 50.5   NA 47.8 48.6 49.6 51.1 50.7 53.4 42.8 47.3
  [71] 48.2 49.8 44.4 45.7 48.0 50.7 36.1 48.0 37.4 35.1 40.5 43.5 47.1 49.3
  [85] 49.6 53.4 51.6 49.6 53.8 46.8 42.6 40.3   NA 51.3 53.4 52.0 54.7 54.3
  [99] 54.0 52.3 53.2 55.4 56.1 54.5 53.8 56.3 55.9 48.6 43.9 45.3 44.4 45.9
 [113] 48.7 53.1 48.9 32.2 39.7 43.0 45.0 42.3 47.3 52.3 49.1   NA 53.4 56.3
 [127] 54.0 48.9 53.4 49.3 48.0 48.7 44.6 40.3 43.9 44.4 46.9 49.3 50.5 50.2
 [141] 52.9 54.1 54.7 53.2   NA 50.7 51.1 51.4 52.0 52.2 46.2 46.6 50.2 50.4
 [155]   NA 47.1 49.3 43.9 43.7 46.0 46.9 46.6 48.9 52.7 54.7 51.8 46.0 46.4
 [169] 46.2 47.3 49.5 46.2 45.1 47.3 48.9 53.4 54.5 53.4 49.8 42.6 45.7 49.3
 [183] 51.8 54.9 55.4   NA   NA 52.5 52.9 53.4 51.6 50.5 50.0 54.3 55.2 54.7
 [197] 52.5 49.1 44.1 45.1   NA 37.4 37.9 40.3 44.1 52.9 49.8 37.9   NA 47.7
 [211] 46.2 46.4 47.8 48.4 47.5 48.6   NA 49.1 45.9 46.4 50.5 51.6 53.1 54.5
 [225] 55.2 52.0 42.4 38.1 34.2 38.5 43.5 51.1 48.6 47.8 47.5 51.8 52.3 58.3
 [239] 52.5 51.6 45.7 46.6 50.5 50.7 45.3 38.5 40.3   NA 56.3 58.5 55.2 51.6
 [253] 48.9 49.5 48.7 48.4 47.8 48.2 51.3 51.4 53.1 54.0 55.6 53.8 46.9 49.8
 [267] 51.6 52.9 50.0 48.9 46.6 45.3 43.3 41.2 39.4 36.5 36.7 40.1   NA 55.0
 [281] 51.8 53.2 53.1 45.1 45.9 47.8 49.6 52.3 51.4 52.5 48.9 37.6 40.8 38.7
 [295] 35.6 38.5 40.5 42.8 47.5 50.4 50.5 47.7 45.5 45.7 46.2 44.4   NA 49.6
 [309] 49.3   NA 55.6 43.5 46.6 49.3 45.9 47.8 50.5 53.2 54.1 54.5 47.5 48.7
 [323] 51.1 49.8 40.8 43.3 47.5 44.1 39.4 42.4 45.1 50.2 48.9 49.5 44.2 47.7
 [337] 44.6 45.7 50.5 51.1   NA 58.6 58.3 54.7 55.0 54.7 52.9 53.1 51.8 45.5
 [351] 41.5 45.9 50.0 53.6 53.2 51.4 53.1 51.4 47.8 43.7 42.8 41.9 41.9 41.9
 [365] 40.5 39.4 36.0 35.4 38.7 41.9 42.3   NA 40.5 41.5 45.9 49.8 51.1 52.2
 [379] 53.4 53.6 49.3 49.1 53.4 53.6 50.5 49.6 49.5 50.2 52.0 46.9 48.6 48.4
 [393] 50.5 48.7 49.1 49.5 51.8 50.4 49.1 46.4 49.3 48.7   NA 38.8 37.8 40.8
 [407] 39.0 37.6 41.5 46.8 47.8 49.6 48.6 50.7 54.0 57.6 57.9 55.0 55.8 55.2
 [421] 57.4 57.9 59.4 54.7 42.4 41.4 43.5 45.1 45.9 46.9 47.1 48.2 50.5   NA
 [435] 55.2 56.1 55.4 54.3 52.3 47.8 55.6 54.1 54.7 55.6 50.9 47.5 46.4 49.3
 [449] 49.6 49.6 51.1 52.9 52.5 51.8 51.6 52.7 50.7 55.8 49.1 48.7 48.6 45.7
 [463] 45.1 50.4   NA 56.7 57.4 57.9 55.8 53.6 44.2 40.5 41.9 48.0 49.8 50.7
 [477] 52.0 52.9 55.8 54.7 44.4 45.3 46.0 48.0 51.1 53.1 53.6 54.0 53.1 52.0
 [491] 48.4 48.9 42.4 38.8 36.7   NA 54.0 55.8 55.6 51.6 53.1 52.7 53.1 47.7
 [505] 44.1 43.5 54.9 53.8 51.1 51.8 53.2 52.2 44.1 46.6 48.4 49.6 53.1 54.5
 [519] 54.3 56.1 54.3 52.3 54.0 54.7 52.7 52.9   NA 64.9 53.2 44.1 48.9 51.6
 [533] 54.0 54.7 53.4 52.5 54.0 54.5 55.9 51.6 46.6 49.8 50.7 51.1 44.2 31.1
 [547] 31.5 39.2 44.6 46.6 49.1 49.1 47.5 46.8 46.4 45.1 41.9   NA 52.7 52.0
 [561] 49.3 47.3 46.4 49.1 50.9 47.5 43.5 38.1 33.8 38.5 42.8 44.2 45.3 41.7
 [575] 40.1 43.7 47.1 48.4 48.0 47.3 47.7 40.1 43.7 41.7 42.1 48.2 46.8 48.6
 [589]   NA 54.1 55.0 57.0 57.9 57.4 55.0 51.4 50.9 46.4 48.9 52.0 54.0 53.8
 [603] 50.0 33.1 34.7 42.8 47.7 47.5 38.3 45.0 36.5 40.8 43.2 43.5 45.7 46.6
 [617] 48.0 49.3 52.2   NA 57.2 57.7 61.9 54.9 49.3 55.4 53.6 54.1 55.8 53.6
 [631] 52.7 53.4 51.3 39.9 43.5 42.6 40.8 44.6 41.4 42.4 49.6 37.0 37.4 37.8
 [645] 40.5 42.1 44.4 42.3 31.5 44.8   NA 41.9 49.3 51.3 53.2 55.2 55.8 53.8
 [659] 55.9 52.0 46.4 48.9 50.4 52.0 53.2 52.9 50.4 50.0 49.6 46.8 41.9 40.3
 [673] 39.9 45.7 51.3 53.6 52.7 50.4 50.7 48.6 50.7   NA 56.5 54.0 55.4 54.7
 [687] 53.4 54.0 48.0 51.1 52.9 55.2 57.0 52.2 50.2 45.0 49.6 52.2 53.2 55.9
 [701] 57.0 53.2 49.1 49.1 52.7 53.6 55.9 54.3 54.1 51.1 42.4 34.7   NA 50.7
 [715] 56.1 54.9 52.0 51.8 53.4 53.8 44.4 37.8 43.9 48.0 49.5 47.8 47.8 46.4
 [729] 48.0 47.5 50.7 48.0 42.6 43.0 41.7 43.0 48.0 52.5 52.2 54.3 55.2 56.1
 [743] 54.9   NA 51.6 53.6 53.1 53.2 51.8 52.0 53.2 53.6 55.6 48.4 46.4 47.1
 [757] 48.6 48.9 50.9 47.5 44.6 47.8 50.7 51.3 51.1 53.4 53.1 49.6 46.0 46.4
 [771] 51.4 52.0 51.6 54.0   NA 48.9 51.3 51.6 53.1 53.8 51.8 54.0 58.5 59.4
 [785] 57.6 52.7 52.5 54.1 53.2 52.2 51.1 49.8 51.3 51.4 52.5 53.1 52.5 52.2
 [799] 51.8 47.1 49.8 49.5 49.1 50.0 51.6   NA 56.8 59.4 59.5 58.6 58.3 57.0
 [813] 54.3 54.0 57.2 54.9 54.0 54.3 53.4 53.4 56.1 58.5 47.3 40.5 47.8 49.3
 [827] 39.9 37.2 43.2 47.7 33.3 32.2 34.9 44.1 47.1 47.8   NA 55.0 55.4 55.0
 [841] 53.4 52.2 53.2 55.6 49.6 46.8 51.3 53.4 53.6 53.8 53.1 56.7 59.2 57.6
 [855] 53.1 54.3 55.8 46.8 49.6 53.6 55.9 48.6 39.6 33.4 34.9 39.2 43.2   NA
 [869] 52.2 52.2 50.2 42.8 42.4 45.5 49.5 52.9 55.9 56.5 58.3 59.0 57.9 50.9
 [883] 46.2 41.4 42.8 46.2 49.5 53.1 52.7 53.4 51.8 55.0 55.4 57.0 53.8 52.5
 [897] 55.0 52.7   NA 55.2 50.2 48.6 47.5 45.0 47.7 51.4 51.8 53.2 55.4 53.6
 [911] 51.4 41.0 40.8 45.7 47.1 49.5 53.1 54.1 54.1 47.8 35.1 40.5 46.8 51.1
 [925] 53.8 51.8 49.5 50.9 49.1   NA 57.0 59.0 59.4 59.5 58.3 54.9 52.9 46.2
 [939] 47.5 55.2 57.7 50.9 49.6 47.5 45.7 46.8 45.0 46.0 44.4 40.1 30.0 30.0
 [953] 32.4 37.0 40.3 43.9 41.7 43.2 42.8 43.5   NA 50.7 54.3 54.7 54.7 53.4
 [967] 54.0 57.0 53.6 51.1 51.1 50.4 50.0 49.1 51.3 45.5 45.7 49.8 47.1 43.0
 [981] 48.2 48.6 48.9 46.8 47.7 49.5 49.8 52.2 54.1 51.6 49.8   NA 55.8 58.8
 [995] 59.2 55.8 55.2 55.8 51.8 42.8 41.4 38.3 43.9 50.2 51.1 52.0 54.0 40.1
[1009] 38.3 42.4 36.1 39.9 42.4 46.2 44.2 48.4 51.4 52.0 50.9 38.1 28.4 29.1
[1023]   NA 57.9 57.4 60.6 62.6 61.7 59.2 58.5 51.1 49.3 46.6 49.8 51.6 52.0
[1037] 55.4 54.0 53.2 54.1 52.0 47.5 50.4 51.6 54.3 51.1 53.6 53.1 54.0 50.0
[1051] 47.5 51.6 52.7   NA 50.0 46.6 48.6 52.5 55.2 58.5 58.8 59.4 62.2 55.0
[1065] 54.0 53.4 50.7 54.1 53.2 49.3 51.8 53.4 48.6 47.5 49.5 50.0 49.5 52.0
[1079] 51.3 51.4 52.5 41.9 39.4 41.2   NA 61.9 62.8 64.2 63.3 62.8 60.1 61.5
[1093] 60.4 59.7 57.0 56.1 51.8 51.3 47.3 45.7 44.8 43.7 43.7 41.9 41.9 37.4
[1107] 40.5 44.6 48.0 50.2 54.1 52.2 54.9 53.8 46.8   NA 47.5 48.9 43.9 44.2
[1121] 49.1 50.2 52.7 51.4 54.0 53.1 49.8 49.8 50.0 53.8 50.4 50.7 50.4 47.3
[1135] 47.5 46.6 38.8 39.4 41.9 43.9 47.5 49.8 45.5 43.7 43.0 37.0   NA 59.0
[1149] 57.9 55.2 56.7 59.4 59.7 59.4 55.9 54.1 57.4 50.2 43.2 49.1 49.8 48.2
[1163] 39.0 40.8 40.8 42.4 45.3 49.1 48.9 50.0 52.5 52.5 50.2 52.0 53.1 52.2
[1177] 51.8   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
[1191]   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA
[1205]   NA   NA   NA   NA   NA
```

In [200…

```
# Ejecutamos nuevamente las antepenultimas lineas de codigo para ver el gráfico final.
prom_mensual <- colMeans(df[, grep("_Prom", colnames(df))], na.rm = TRUE)
meses_ordenados <- c("Ene_Prom", "Feb_Prom", "Mar_Prom", "Abr_Prom", "May_Prom", "Jun_Prom",
                     "Jul_Prom", "Ago_Prom", "Sep_Prom", "Oct_Prom", "Nov_Prom", "Dic_Prom")
prom_mensual_ordenado <- prom_mensual[meses_ordenados]
meses <- 1:12
df_grafica <- data.frame(
  Mes = factor(month.abb, levels = month.abb),
  Temperatura_Promedio = prom_mensual_ordenado
)
library(ggplot2)
ggplot(df_grafica, aes(x = Mes, y = Temperatura_Promedio)) +
  geom_line(group = 1, color = "steelblue", size = 1) +
  geom_point(color = "red", size = 2) +
  geom_hline(yintercept = 32, linetype = "dashed", color = "darkgray") +
  labs(title = "Temperatura Promedio por Mes de la Estación: HAGAN'S MEADOW",
       x = "Mes",
       y = "Temperatura Promedio (°F)")
```



In [201…

```
# Finalmente generamos el CSV final con todo el procesamiento de datos

write.csv(df, "base_limpia_final1.csv", row.names = FALSE)
```

Resultados

Podemos observar que derivado de la limpieza de datos se obtuvo un archivo ".csv" manejable para realizar cualquier análisis estadístico. Ahora los datos están ordenados de una forma en la que cualquier análista de datos puede emplearlos y realizar las estadísticas requeridas.

Conclusión

En conclusión es importante tener buenos conocimientos sobre manipulaciones de datos ya que no siempre las bases de datos se encuentran en un estado óptimo para su uso. Así mismo, es importante saber localizar alternativas a las fuentes de datos, ya que como se muestra en este capítulo muchas veces los servidores dejan de estar disponibles.

In [ ]: