HT#3

María José Ramírez Cifuentes/Biancka Raxon

2025-02-12

Gráfica 1:

Librerías

```
library(ggplot2)
library(tidyr)
library(dplyr)
library(reshape2)
```

Data Frame

```
#Data frame
datos <- data.frame(
  Tamaño = c(10, 100, 500, 1000, 1500, 2000, 2500, 3000),
  GnomeSort = c(27, 614, 7548, 82, 24222, 35475, 43499, 341),
  InsertionSort = c(24, 314, 3405, 325, 12558, 13210, 34288, 176),
  RadixSort = c(27, 80, 247, 517, 471, 520, 682, 697),
  QuickSort = c(8, 237, 9, 957, 7, 5, 5, 6),
  MergeSort = c(8, 322, 9, 770, 15, 5, 5, 6),
  GnomeSort_Ordered = c(28, 25, 52, 82, 132, 121, 128, 137),
  InsertionSort_Ordered = c(38, 46, 89, 325, 465, 261, 353, 232),
  RadixSort_Ordered = c(56, 193, 456, 269, 652, 632, 753, 704),
  QuickSort_Ordered = c(7, 5, 7, 9, 13, 7, 4, 5),
  MergeSort_Ordered = c(11, 10, 10, 18, 41, 9, 7, 13))
datos
```

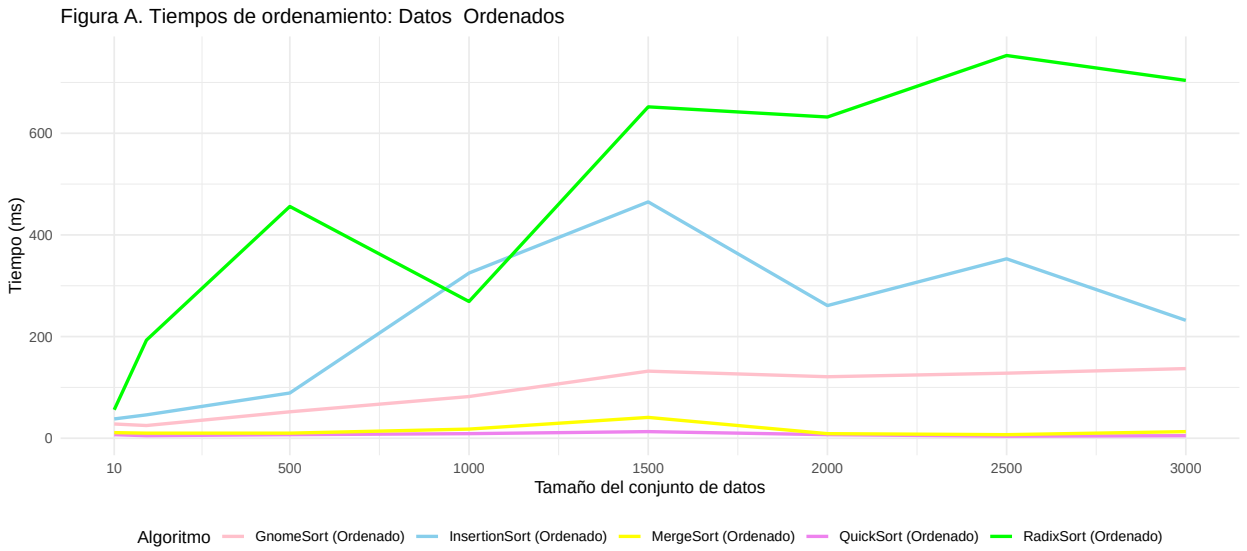
```
##   Tamaño GnomeSort InsertionSort RadixSort QuickSort MergeSort
## 1     10        27            24        27         8         8
## 2    100       614           314        80       237       322
## 3    500      7548          3405       247         9         9
## 4   1000        82           325       517       957       770
## 5   1500     24222         12558       471         7        15
## 6   2000     35475         13210       520         5         5
## 7   2500     43499         34288       682         5         5
## 8   3000       341           176       697         6         6
##   GnomeSort_Ordered InsertionSort_Ordered RadixSort_Ordered QuickSort_Ordered
## 1                28                    38                56                 7
## 2                25                    46               193                 5
## 3                52                    89               456                 7
```

```
## 4                82                 325                269                 9
## 5               132                 465                652                13
## 6               121                 261                632                 7
## 7               128                 353                753                 4
## 8               137                 232                704                 5
##   MergeSort_Ordered
## 1                11
## 2                10
## 3                10
## 4                18
## 5                41
## 6                 9
## 7                 7
## 8                13
```

Gráfica comparación tiempo datos ordenados

```
library(ggplot2)

ggplot(datos, aes(x = Tamaño)) +
  geom_line(aes(y = GnomeSort_Ordered, color = "GnomeSort (Ordenado)"), size = 1) +
  geom_line(aes(y = InsertionSort_Ordered, color = "InsertionSort (Ordenado)"), size = 1) +
  geom_line(aes(y = RadixSort_Ordered, color = "RadixSort (Ordenado)"), size = 1) +
  geom_line(aes(y = QuickSort_Ordered, color = "QuickSort (Ordenado)"), size = 1) +
  geom_line(aes(y = MergeSort_Ordered, color = "MergeSort (Ordenado)"), size = 1) +
  labs(
    title = "Figura A. Tiempos de ordenamiento: Datos  Ordenados",
    x = "Tamaño del conjunto de datos",
    y = "Tiempo (ms)",
    color = "Algoritmo"
  ) +
  theme_minimal() +
  scale_color_manual(
    values = c(
      "GnomeSort (Ordenado)" = "pink",
      "InsertionSort (Ordenado)" = "skyblue",
      "RadixSort (Ordenado)" = "green",
      "QuickSort (Ordenado)" = "violet",
      "MergeSort (Ordenado)" = "yellow"
    )
  ) +
  scale_x_continuous(breaks = c(10, 500, 1000, 1500, 2000, 2500, 3000)) +
  theme(legend.position = "bottom")
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

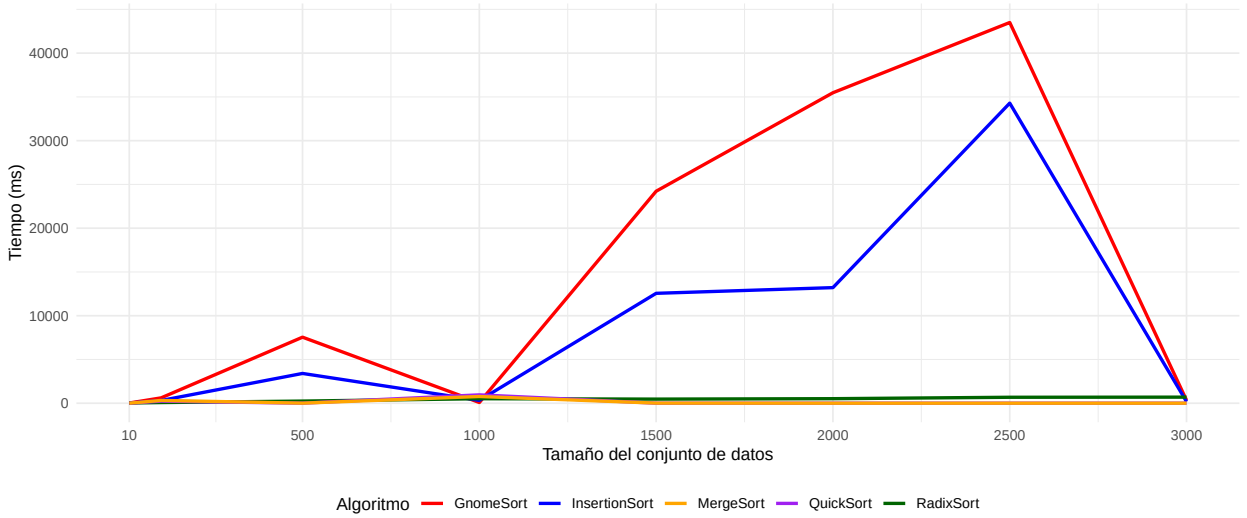


En la gráfica de tiempos de ejecución para datos ordenados, se observa que Radix Sort es el algoritmo que toma más tiempo en comparación con los demás, mientras que QuickSort es el más eficiente, seguido de MergeSort.

Gráfica comparación tiempo datos NO ordenados

```
ggplot(datos, aes(x = Tamaño)) +
  geom_line(aes(y = GnomeSort, color = "GnomeSort"), size = 1) +
  geom_line(aes(y = InsertionSort, color = "InsertionSort"), size = 1) +
  geom_line(aes(y = RadixSort, color = "RadixSort"), size = 1) +
  geom_line(aes(y = QuickSort, color = "QuickSort"), size = 1) +
  geom_line(aes(y = MergeSort, color = "MergeSort"), size = 1) +
  labs(
    title = "Figura A. Tiempos de ordenamiento: Datos Ordenados",
    x = "Tamaño del conjunto de datos",
    y = "Tiempo (ms)",
    color = "Algoritmo"
  ) +
  theme_minimal() +
  scale_color_manual(
    values = c(
      "GnomeSort" = "red",
      "InsertionSort" = "blue",
      "RadixSort" = "darkgreen",
      "QuickSort" = "purple",
      "MergeSort" = "orange"
    )
  ) +
  scale_x_continuous(breaks = c(10, 500, 1000, 1500, 2000, 2500, 3000)) +
  theme(legend.position = "bottom")
```

Figura A. Tiempos de ordenamiento: Datos Ordenados



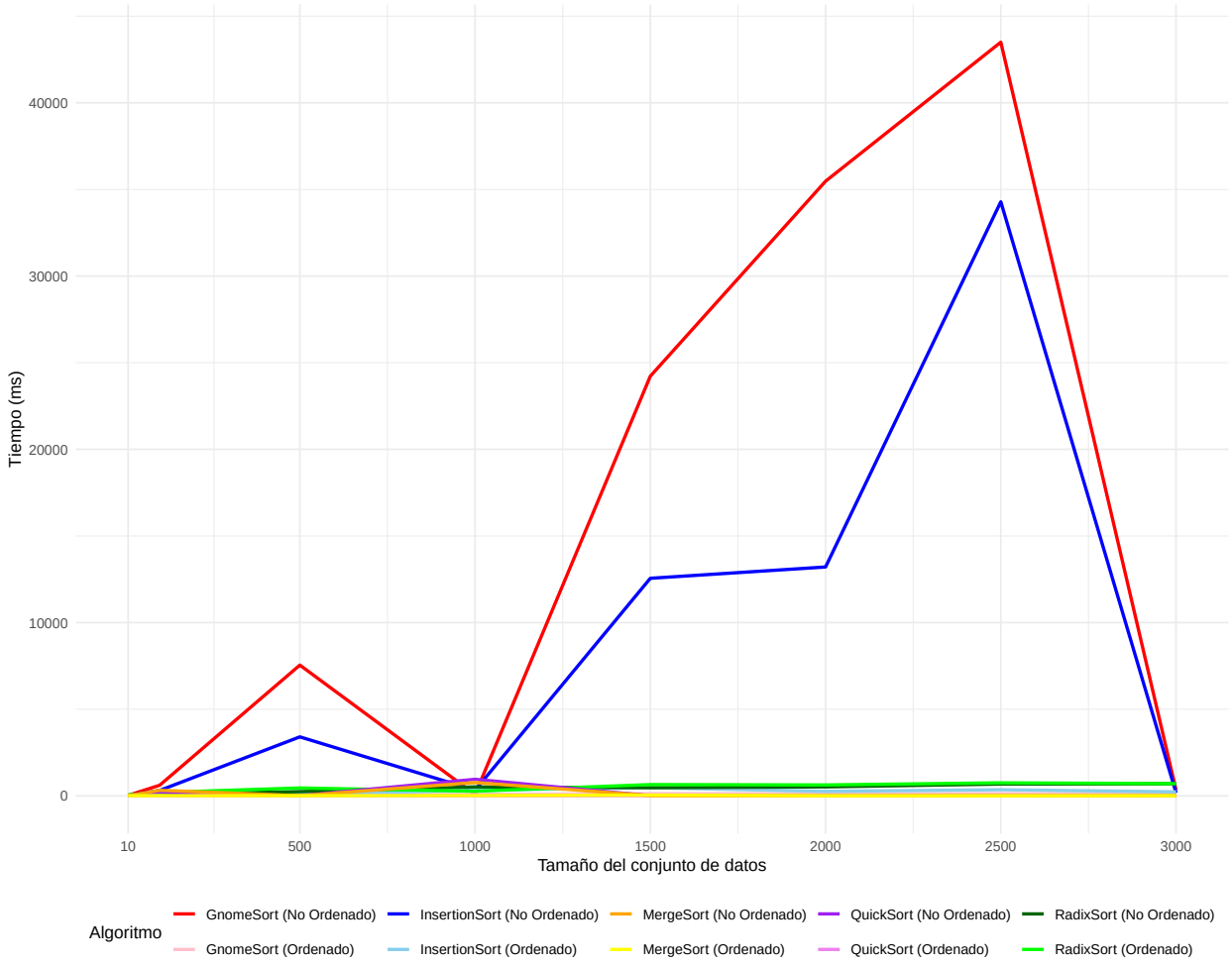
En la gráfica de tiempos de ejecución para datos NO ordenados, se observa que GnomeSort e InsertionSort son los algoritmos que toman más tiempo en comparación con los demás, mientras que QuickSort, MergeSort y RadixSort son los más eficientes. Esto tiene sentido ya que por lo general tanto GnomeSort como InsertionSort son los que más tiempo se toman en promedio

Gráfica comparación tiempo datos ordenados vs No ordenados

```
ggplot(datos, aes(x = Tamaño)) +
  geom_line(aes(y = GnomeSort, color = "GnomeSort (No Ordenado)"), size = 1) +
   geom_line(aes(y = GnomeSort_Ordered, color = "GnomeSort (Ordenado)"), size = 1) +
  geom_line(aes(y = InsertionSort, color = "InsertionSort (No Ordenado)"), size = 1) +
  geom_line(aes(y = InsertionSort_Ordered, color = "InsertionSort (Ordenado)"), size = 1) +
  geom_line(aes(y = RadixSort, color = "RadixSort (No Ordenado)"), size = 1) +
  geom_line(aes(y = RadixSort_Ordered, color = "RadixSort (Ordenado)"), size = 1) +
  geom_line(aes(y = QuickSort, color = "QuickSort (No Ordenado)"), size = 1) +
  geom_line(aes(y = QuickSort_Ordered, color = "QuickSort (Ordenado)"), size = 1) +
  geom_line(aes(y = MergeSort, color = "MergeSort (No Ordenado)"), size = 1) +
  geom_line(aes(y = MergeSort_Ordered, color = "MergeSort (Ordenado)"), size = 1) +
  labs(
    title = "Figura A. Comparación de tiempos de ordenamiento: Datos no ordenados vs Datos ya ordenados
    x = "Tamaño del conjunto de datos",
    y = "Tiempo (ms)",
    color = "Algoritmo"
  ) +
  theme_minimal() +
  scale_color_manual(
    values = c(
      "GnomeSort (No Ordenado)" = "red",
      "GnomeSort (Ordenado)" = "pink",
      "InsertionSort (No Ordenado)" = "blue",
      "InsertionSort (Ordenado)" = "skyblue",
      "RadixSort (No Ordenado)" = "darkgreen",
      "RadixSort (Ordenado)" = "green",
      "QuickSort (No Ordenado)" = "purple",
```

```
    "QuickSort (Ordenado)" = "violet",
    "MergeSort (No Ordenado)" = "orange",
    "MergeSort (Ordenado)" = "yellow"
  )
) +

scale_x_continuous(breaks = c(10, 500, 1000, 1500, 2000, 2500, 3000)) +
theme(legend.position = "bottom")
```

Figura A. Comparación de tiempos de ordenamiento: Datos no ordenados vs Datos ya ordenados

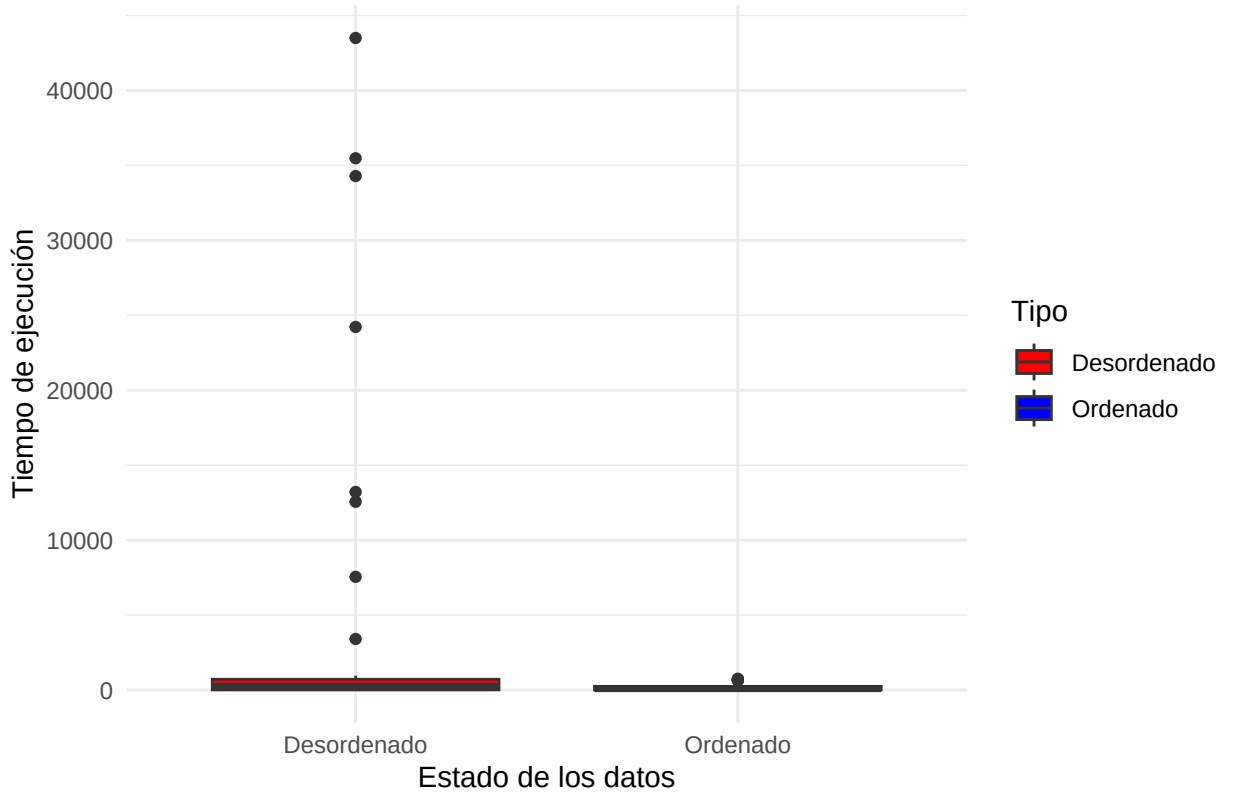


Boxplot (Comparación datos ordenados vs no ordenados)

```
datos_long <- pivot_longer(datos, cols = -Tamaño, names_to = "Algoritmo", values_to = "Tiempo")


datos_long$Tipo <- ifelse(grepl("_Ordered", datos_long$Algoritmo), "Ordenado", "Desordenado")


ggplot(datos_long, aes(x = Tipo, y = Tiempo, fill = Tipo)) +
  geom_boxplot() +
  theme_minimal() +
  labs(title = "Figura B. Boxplot Comparación de tiempo de ejecución: Datos desordenados vs ordenados",
```

```
      x = "Estado de los datos",
      y = "Tiempo de ejecución") +
  scale_fill_manual(values = c("Desordenado" = "red", "Ordenado" = "blue"))
```

Figura B. Boxplot Comparación de tiempo de ejecución: Datos desordena



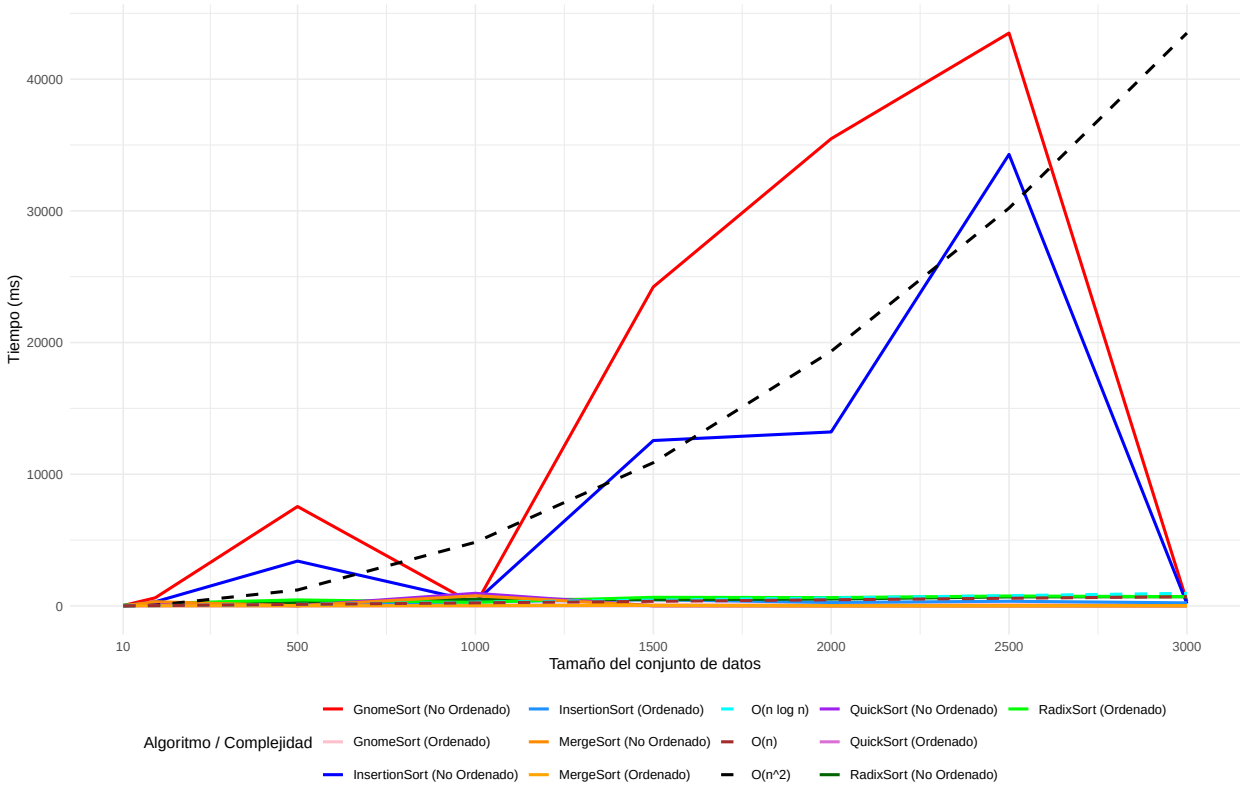
En la gráfica "Comparación de tiempos de ordenamiento: Datos no ordenados vs Datos ya ordenados", se observa que la diferencia en tiempos de ejecución entre ambos conjuntos de datos no es significativa en la mayoría de los algoritmos. Sin embargo, Gnome Sort e Insertion Sort presentan una notable variación, ya que son los algoritmos con mayor tiempo de ejecución.

Este comportamiento se debe a que estos algoritmos generan valores atípicos al alejarse considerablemente de la media central. En la Figura B, se evidencia que en los datos desordenados existen múltiples valores atípicos que se distancian del promedio, mientras que en los datos ordenados solo se observa un único valor atípico.

```
datos$O_n2 <- (datos$Tamaño^2) / max(datos$Tamaño^2) * max(datos$GnomeSort)
datos$O_nlogn <- (datos$Tamaño * log(datos$Tamaño)) / max(datos$Tamaño * log(datos$Tamaño)) * max(datos$
datos$O_n <- datos$Tamaño / max(datos$Tamaño) * max(datos$RadixSort)

ggplot(datos, aes(x = Tamaño)) +
  # Líneas de tiempos experimentales
  geom_line(aes(y = GnomeSort, color = "GnomeSort (No Ordenado)"), size = 1) +
  geom_line(aes(y = GnomeSort_Ordered, color = "GnomeSort (Ordenado)"), size = 1) +
  geom_line(aes(y = InsertionSort, color = "InsertionSort (No Ordenado)"), size = 1) +
  geom_line(aes(y = InsertionSort_Ordered, color = "InsertionSort (Ordenado)"), size = 1) +
  geom_line(aes(y = RadixSort, color = "RadixSort (No Ordenado)"), size = 1) +
  geom_line(aes(y = RadixSort_Ordered, color = "RadixSort (Ordenado)"), size = 1) +
  geom_line(aes(y = QuickSort, color = "QuickSort (No Ordenado)"), size = 1) +
```

```
  geom_line(aes(y = QuickSort_Ordered, color = "QuickSort (Ordenado)"), size = 1) +
  geom_line(aes(y = MergeSort, color = "MergeSort (No Ordenado)"), size = 1) +
  geom_line(aes(y = MergeSort_Ordered, color = "MergeSort (Ordenado)"), size = 1) +

  # Líneas de rendimiento teórico
  geom_line(aes(y = O_n2, color = "O(n^2)"), linetype = "dashed", size = 1) +
  geom_line(aes(y = O_nlogn, color = "O(n log n)"), linetype = "dashed", size = 1) +
  geom_line(aes(y = O_n, color = "O(n)"), linetype = "dashed", size = 1) +

  labs(
    title = "Figura A. Comparación de tiempos de ordenamiento: Datos no ordenados vs Datos ya ordenados
    x = "Tamaño del conjunto de datos",
    y = "Tiempo (ms)",
    color = "Algoritmo / Complejidad"
  ) +
  theme_minimal() +
  scale_color_manual(
    values = c(
      "GnomeSort (No Ordenado)" = "red",
      "GnomeSort (Ordenado)" = "pink",
      "InsertionSort (No Ordenado)" = "blue",
      "InsertionSort (Ordenado)" = "dodgerblue",
      "RadixSort (No Ordenado)" = "darkgreen",
      "RadixSort (Ordenado)" = "green",
      "QuickSort (No Ordenado)" = "purple",
      "QuickSort (Ordenado)" = "orchid",
      "MergeSort (No Ordenado)" = "darkorange",
      "MergeSort (Ordenado)" = "orange",
      "O(n^2)" = "black",
      "O(n log n)" = "cyan",
      "O(n)" = "brown"
    )
  ) +
  scale_x_continuous(breaks = c(10, 500, 1000, 1500, 2000, 2500, 3000)) +
  theme(legend.position = "bottom")
```

Figura A. Comparación de tiempos de ordenamiento: Datos no ordenados vs Datos ya ordenados vs Complejidad Teórica



Se añadieron líneas de referencia con la complejidad Big-O para comparar el rendimiento teórico y empírico. GnomeSort e InsertionSort presentan datos atípicos, lo que explica sus tiempos elevados frente a los demás algoritmos.

La complejidad de cada algoritmo se evalúa en tres escenarios:

- *Mejor caso: cuando los datos ya están ordenados o casi ordenados.*
- *Caso promedio: comportamiento esperado con datos aleatorios.*
- *Peor caso: cuando los datos están en el peor orden posible para el algoritmo. Complejidad de los algoritmos:*
- *Gnome Sort: Mejor: $O(n)$, Promedio y Peor: $O(n^2)$*
- *Insertion Sort: Mejor: $O(n)$, Promedio y Peor: $O(n^2)$*
- *Radix Sort: Siempre $O(nk)$*
- *QuickSort: Mejor y Promedio: $O(n \log n)$, Peor: $O(n^2)$*
- *MergeSort: Siempre $O(n \log n)$*