Taller 0 - Introducción

Adaptado del taller 2022 de Federico Molina

Series Cronológicas 2024

Marzo 2024

En este taller, se introduce el lenguaje de programación de R y el entorno integrado de desarrollo, RStudio. Además, se presenta el formato de archivos *.Rmd*, que permite generar archivos autocontenidos de texto y código.

1. Directorios y proyectos

Los archivos de datos se cargan en forma **condicional** a la ruta en la que se encuentran. Por lo tanto, es necesario conocer el directorio en el que se está trabajando. Para ello, se usa la función `getwd()`. Para cambiarlo, puede utilizarse `setwd()` o bien utilizar el menú de RStudio: `Session/Set working directory/Choose directory...`.

```
# Obtenemos el directorio en el que estamos trabajando
getwd()
```

```
## [1] "/Users/anavignolo/Desktop/Trabajo - IESTA/Talleres Series/Talleres_2024/Taller0_introduccion"
```

Para facilitar el manejo de directorios y facilitar la reproducibilidad de los *scripts*, es recomendable trabajar con proyectos. Al abrir un proyecto de R, se fija el directorio en la carpeta en la que se encuentre el archivo *.Rproj*, independientemente del sistema operativo que se tenga (Linux, Windows, etc.).

Para crear un nuevo proyecto desde el menú de opciones, se debe seleccionar la opción `New Project` dentro de `File`. Existen tres opciones:

- **New Directory:** Permite crear un proyecto en una nueva carpeta.
- **Existing Directory:** Crea el proyecto en una carpeta ya existente.
- **Version Control:** Crea un proyecto asociado a Git y Github.

Es recomendable que el nombre del proyecto no tenga **espacios** ni **caracteres especiales**.

2. Librerías

Muchas de las funciones con las que se trabajará no se encuentran en el R base, sino que pertenecen a paquetes que deben ser instalados antes de ser utilizados por primera vez mediante la función `install.packages()`. Posteriormente, debe cargárselos cada vez que se los quiera usar con la función `library()`. También es posible utilizar únicamente una función sin cargar todo el paquete que la contiene mediante comandos de la forma `library::function(...)`.

```r
# Cargamos las librerías
# Si tenemos que instalar paquetes, usamos install.packages("")

library(tidyverse)
library(here) # Para fijar el directorio
library(data.table)

# Para trabajar con series de tiempo
library(fable)
library(zoo)
library(tsibble)
library(tsibbledata)
# tsibbledata::gafa_stock
# tsibbledata::aus_livestock
```

3. Archivos R Markdown

Los *scripts* más básicos de R tienen extensión *.R*. En ellos, se escribe directamente el código y sus comentarios. Por su parte, los archivos R Markdown, de formato *.Rmd*, permiten integrar secciones de código y texto para luego compilarlo en diferentes formatos (pdf, html, docx, etc.). Esto permite generar un archivo autocontenido con todo el análisis realizado, lo cual contribuye a su reproducibilidad.

Los archivos R Markdown se componen de diferentes partes:

- **YAML:** Se encuentra al inicio del archivo y permite especificar distintos aspectos del archivo tales como el título, el autor, la fecha y el tipo de archivo de compilación, entre muchos otros.
- **Chunks** Se trata de la sección especial en la que se encuentra el código. En cada *chunk*, pueden definirse distintos aspectos tales como si mostrar o no el código en el archivo compilado o si ejecutar o no el código.
- **Texto:** Se ubica fuera de los *chunks* y funciona de manera muy similar a LaTex: las fórmulas se escriben de la misma manera, se pueden crear referencias cruzadas y agregar archivos *.bib* con la bibliografía, etc.

4. Lectura de datos

En R, pueden importarse datos en una gran variedad de formatos como por ejemplo *.dta*, *.xlsx*, *.xls* y *.csv*. Por defecto, los datos rectangulares se cargan con la clase *data.frame*, aunque esto puede cambiarse dependiendo del tipo de información con la que se esté trabajando. Por ejemplo, los datos para mapas se guardan con una clase y formato diferente.

Dependiendo del formato de los datos que se deseen importar, se deben utilizar distintas librerías:

```r
# Algunas funciones para cargar datos
utils::read.table()
data.table::fread()
readr::read_csv()
haven::read_dta()
```

Para facilitar el manejo de archivos dentro del directorio en el que se esté trabajando, es útil usar la función `here()`, la cual permite especificar su ubicación a la hora cargar los datos. Esta función utiliza la misma sintaxis sin importar el sistema operativo, lo que contribuye a la reproducibilidad del código.

5. Manipulación de datos

Para manipular datos, pueden usarse únicamente las funciones de R base, o bien utilizarse librerías especializadas como `dplyr` como parte de `tidyverse`.

Dentro de `tidyverse`, existe la librería `magrittr`, la cual permite utilizar el operador "pipe", que permite aplicar sucesivamente funciones a un conjunto de datos:

```
# En R base
sum(length(unique(x)))

# Utilizando pipes
x %>%
  unique %>%
  length() %>%
  sum()
```

6. Series de tiempo con R

Para trabajar con series temporales en R base, se suele trabajar con objetos de clase `ts`. Sin embargo, existen otras librerías en R que permiten trabajar más fácilmente con múltiples series simultáneamente (`xts`, `zoo`, `fable`, etc.):

```
# Posibles funciones para definir series temporales
xts::xts()
zoo::zoo()
tsibble::tsibble()
```

7. Ejemplo - Estadísticas de tráfico áereo

7.1. Lectura de datos

```
# Cargamos un csv
dt <- data.table::fread(here("Taller0_introduccion","Air_Traffic_Passenger_Statistics.csv"))

# Corregimos los nombres de las variables (quitamos espacios, mayúsculas, caraceres especiales, etc.)
names(dt)
```

```
##  [1] "Activity Period"             "Operating Airline"
##  [3] "Operating Airline IATA Code" "Published Airline"
##  [5] "Published Airline IATA Code" "GEO Summary"
##  [7] "GEO Region"                  "Activity Type Code"
##  [9] "Price Category Code"         "Terminal"
## [11] "Boarding Area"               "Passenger Count"
```

```
setnames(dt, names(dt), janitor::make_clean_names(names(dt)))
names(dt)
```

```
##  [1] "activity_period"             "operating_airline"
##  [3] "operating_airline_iata_code" "published_airline"
##  [5] "published_airline_iata_code" "geo_summary"
##  [7] "geo_region"                  "activity_type_code"
##  [9] "price_category_code"         "terminal"
## [11] "boarding_area"               "passenger_count"
```

```
# Trabajamos con datos de clase data.frame
class(dt)
```

```
## [1] "data.table" "data.frame"
```

7.2. Manipulación de datos

```
# Renombramos distintas variables y convertimos la variable "date" en formato fecha
dt <- dt %>%
  rename(date = activity_period,
         airline = operating_airline,
         passenger = passenger_count,
         region = geo_region) %>%
  mutate(date = as.Date(paste0(date, 01), format = "%Y%m%d"))

head(dt)
```

```
##          date      airline operating_airline_iata_code published_airline
##        <Date>       <char>                       <char>            <char>
## 1: 2005-07-01 ATA Airlines                           TZ      ATA Airlines
## 2: 2005-07-01 ATA Airlines                           TZ      ATA Airlines
## 3: 2005-07-01 ATA Airlines                           TZ      ATA Airlines
## 4: 2005-07-01   Air Canada                           AC        Air Canada
## 5: 2005-07-01   Air Canada                           AC        Air Canada
## 6: 2005-07-01    Air China                           CA         Air China
##    published_airline_iata_code   geo_summary region activity_type_code
##                          <char>        <char> <char>             <char>
## 1:                           TZ      Domestic     US           Deplaned
## 2:                           TZ      Domestic     US           Enplaned
## 3:                           TZ      Domestic     US     Thru / Transit
## 4:                           AC International Canada           Deplaned
## 5:                           AC International Canada           Enplaned
## 6:                           CA International   Asia           Deplaned
##    price_category_code      terminal boarding_area passenger
##                 <char>        <char>        <char>     <int>
## 1:            Low Fare    Terminal 1             B     27271
## 2:            Low Fare    Terminal 1             B     29131
## 3:            Low Fare    Terminal 1             B      5415
## 4:               Other    Terminal 1             B     35156
## 5:               Other    Terminal 1             B     34090
## 6:               Other International             G      6263
```

```r
tail(dt)
```

```
##          date              airline operating_airline_iata_code
##        <Date>               <char>                       <char>
## 1: 2022-06-01 Vietnam Airlines JSC                           VN
## 2: 2022-06-01 Vietnam Airlines JSC                           VN
## 3: 2022-06-01      Virgin Atlantic                           VS
## 4: 2022-06-01      Virgin Atlantic                           VS
## 5: 2022-06-01     WestJet Airlines                           WS
## 6: 2022-06-01     WestJet Airlines                           WS
##       published_airline published_airline_iata_code   geo_summary region
##                  <char>                      <char>        <char> <char>
## 1: Vietnam Airlines JSC                          VN International   Asia
## 2: Vietnam Airlines JSC                          VN International   Asia
## 3:      Virgin Atlantic                          VS International Europe
## 4:      Virgin Atlantic                          VS International Europe
## 5:     WestJet Airlines                          WS International Canada
## 6:     WestJet Airlines                          WS International Canada
##    activity_type_code price_category_code      terminal boarding_area passenger
##                <char>              <char>        <char>        <char>     <int>
## 1:           Deplaned               Other International             A      4253
## 2:           Enplaned               Other International             A      3660
## 3:           Deplaned               Other International             A     12405
## 4:           Enplaned               Other International             A     14900
## 5:           Deplaned               Other International             A     10016
## 6:           Enplaned               Other International             A     11973
```

7.3. Gráficas

```r
# Obtenemos los totales de pasajeros por fecha, aerolínea y región y graficamos
dt %>%
  group_by(date, airline, region) %>%
  summarise(passenger_sum = sum(passenger)) %>%
  ggplot(aes(x = date, y = passenger_sum, color = airline)) +
  geom_line(show.legend = FALSE) + # Demasiadas aerolíneas para mantener leyendas
  facet_wrap(~region, scales = "free_y") + # Dejamos libre la escala de las ordenadas
  labs(x = "Fecha",
       y = "Cantidad de pasajeros")
```

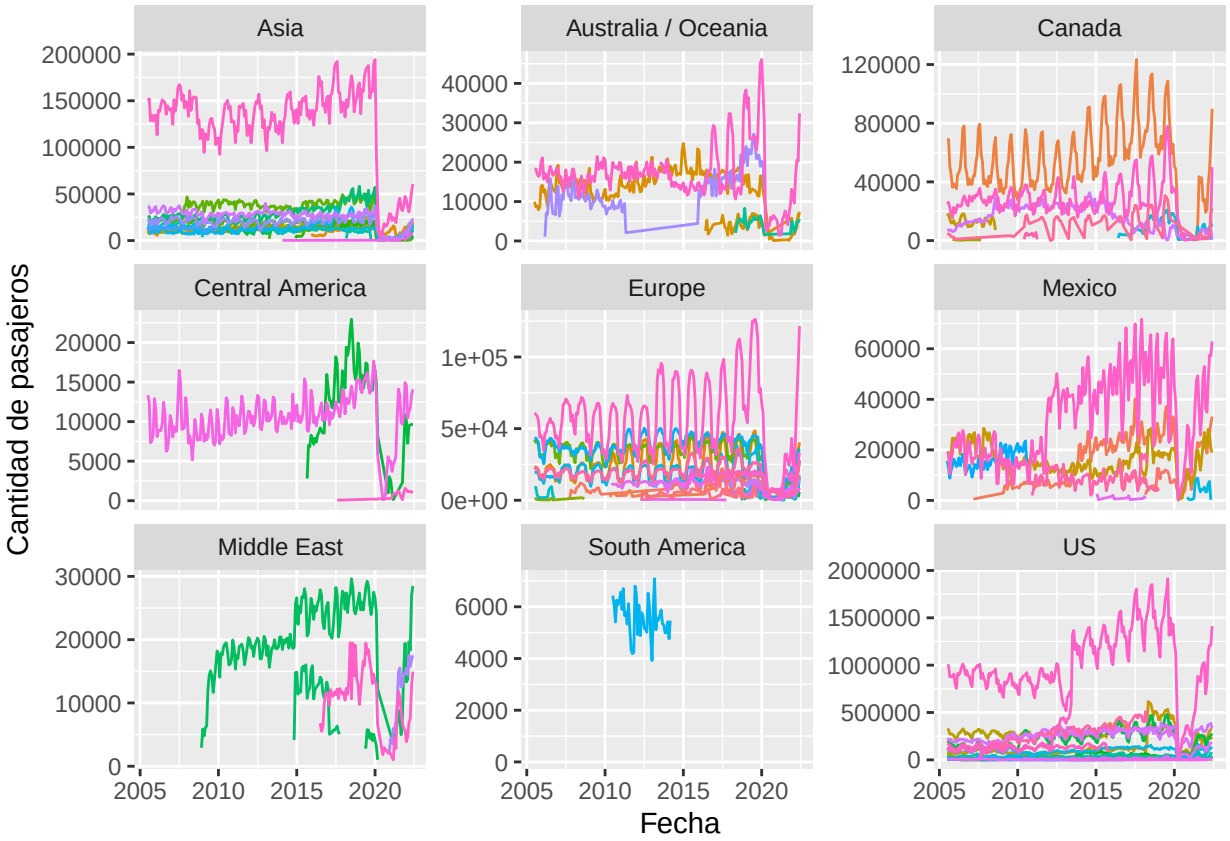

Figura 1: Evolución en la cantidad de pasajeros por aerolínea y región entre julio de 2005 y junio de 2022.

```
# Obtenemos los totales de vuelos por mes y año y graficamos
dt %>%
  group_by(date) %>%
  summarise(n = n()) %>%
  ggplot(aes(x = month(date), y = n, color = factor(year(date)))) +
  geom_line() +
  scale_x_continuous(n.breaks = 12) +
  labs(x = "Mes",
       color = "Año") +
  theme(panel.grid.minor = element_blank())
```

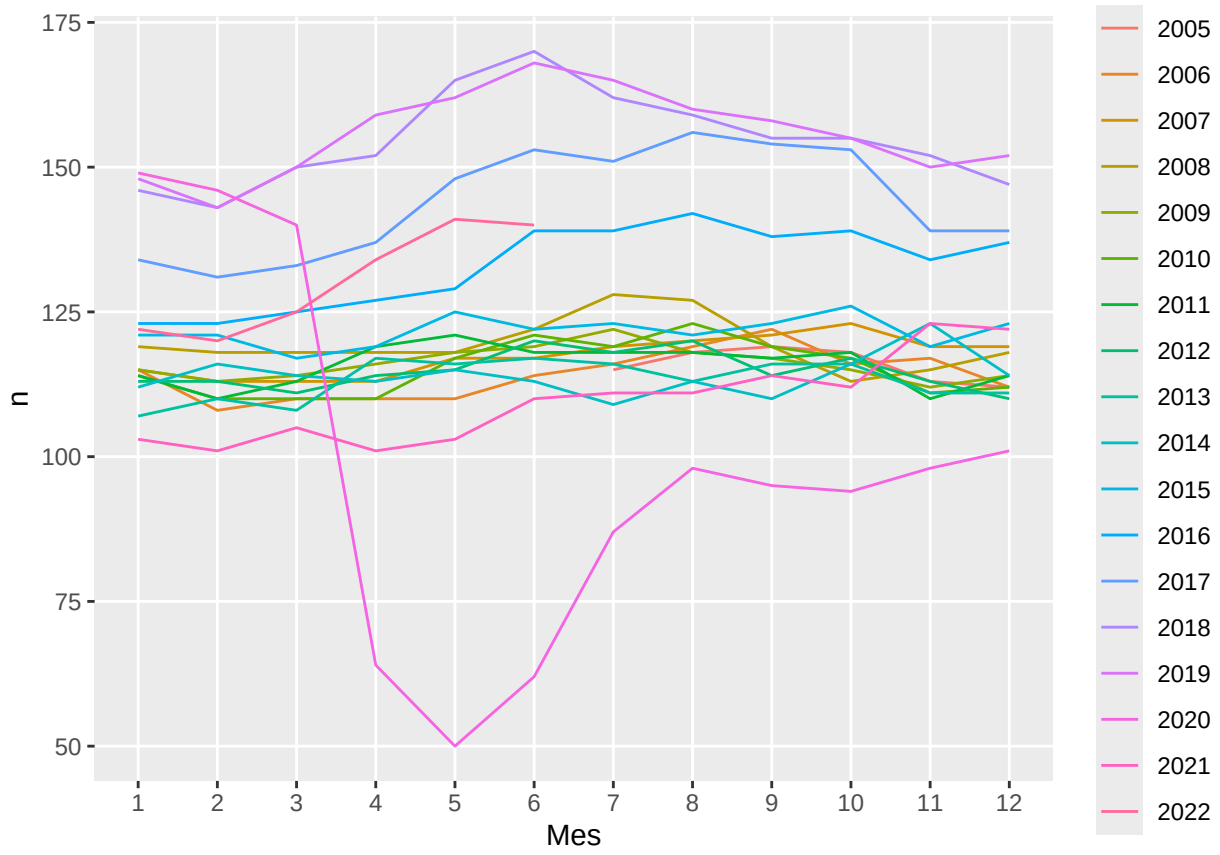

Figura 2: Cantidad mensual de vuelos para los años 2005 a 2022.

7.4. Una serie - Formato `ts`

```
# Obtenemos la cantidad mensual de vuelos
library(ggfortify) # Para que funcione la función autoplot()
dt %>%
  group_by(date) %>%
  summarise(n = n()) %>%
  select(n) %>%
  ts(start = c(2005, 7), frequency = 12) %>% # Tranformamos los datos a clase ts
  autoplot() + # Análogo a ggplot para series de tiempo
  labs(x = "Año",
       y = "Vuelos")
```

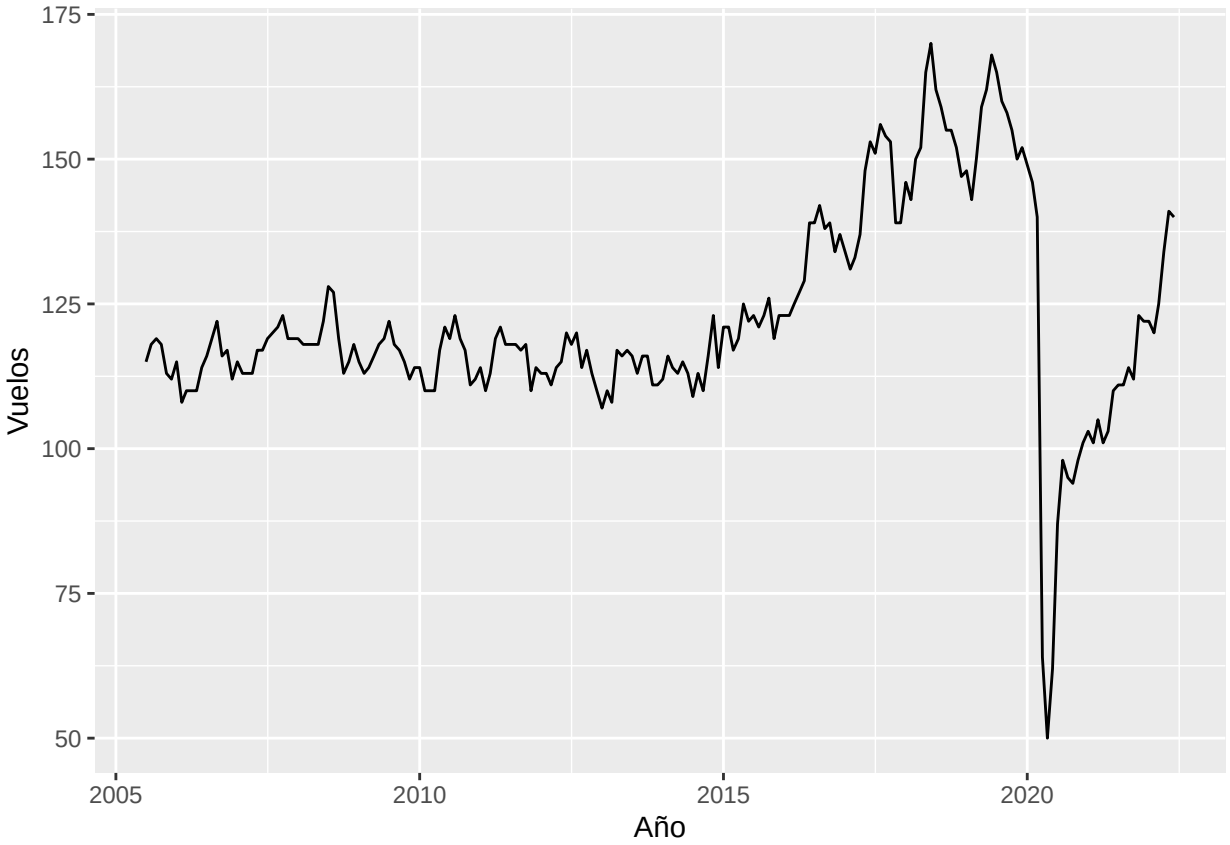

Figura 3: Evolución mensual en la cantidad de vuelos entre 2005 y 2022.

El parámetro `frequency` define la cantidad de observaciones por unidad de tiempo. Si se considera la frecuencia por año, se tiene:

- **Diaria:** 365
- **Semanal:** 52 (o 365.25/7 = 52.18 por el año bisiesto)
- **Mensual:** 12
- **Trimestral:** 4
- **Anual:** 1

7.5. Múltiples series - Formato `tsibble`

```
dt_tsibble <- dt %>%
  group_by(date, airline, region) %>%
  summarise(passenger_sum = sum(passenger)) %>%
  as_tsibble(key = c("airline", "region"), index = date) # Con key se especifica cuál es la serie y con
```

```
# Para graficar las series en un tsibble necesitamos la librería fable
dt_tsibble %>%
  autoplot(show.legend = FALSE) +
  facet_wrap(~region, scales = "free_y") +
```

```
labs(x = "Fecha",
       y = "Cantidad de pasajeros")
```

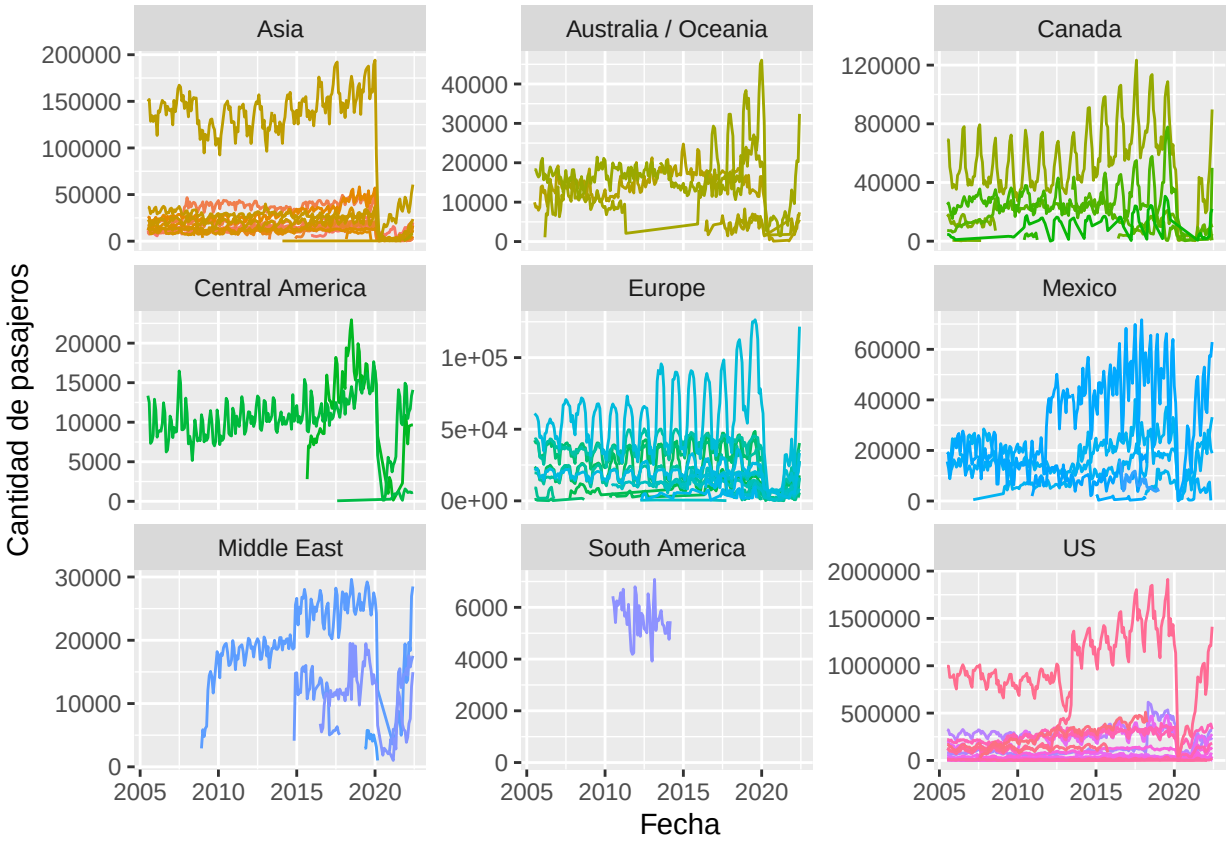

Figura 4: Evolución en la cantidad de pasajeros por aerolínea y región entre julio de 2005 y junio de 2022.