Ordenar dataframes

Jorge Loría

Datos Ejemplo

Los datos que vamos a usar se obtienen de la librería `datasets`, pero para ejemplificar vamos a usar el archivo `Datos_dataframes.RData`.

Datos Ejemplo

Los datos que vamos a usar se obtienen de la librería `datasets`, pero para ejemplificar vamos a usar el archivo `Datos_dataframes.RData`. Para cargarlo solo deben pedirle a la computadora que lo abra con `RStudio`.

Datos Ejemplo

Los datos que vamos a usar se obtienen de la librería `datasets`, pero para ejemplificar vamos a usar el archivo `Datos_dataframes.RData`. Para cargarlo solo deben pedirle a la computadora que lo abra con `RStudio`. En estos dataframes se ve como los mismos datos se pueden representar de diversas formas.

Datos Ejemplo

Los datos que vamos a usar se obtienen de la librería `datasets`, pero para ejemplificar vamos a usar el archivo `Datos_dataframes.RData`. Para cargarlo solo deben pedirle a la computadora que lo abra con `RStudio`. En estos dataframes se ve como los mismos datos se pueden representar de diversas formas.

Los datos que vamos a usar indican, para cada año y cada acción cuantas veces crecen en el año y el crecimiento porcentual que experimenta en ese año.

Datos Ejemplo

Los datos que vamos a usar se obtienen de la librería `datasets`, pero para ejemplificar vamos a usar el archivo `Datos_dataframes.RData`. Para cargarlo solo deben pedirle a la computadora que lo abra con `RStudio`. En estos dataframes se ve como los mismos datos se pueden representar de diversas formas.

Los datos que vamos a usar indican, para cada año y cada acción cuantas veces crecen en el año y el crecimiento porcentual que experimenta en ese año.

Para cargar los datos, se usa el comando `load`, indicando la dirección del archivo:

```
load(file = 'Datos_dataframes.RData')
```

datos1

```
datos1
```

```
## # A tibble: 32 x 4
##   Accion  Anno Veces Porcentual
##   <chr>  <dbl> <int>      <dbl>
## 1 CAC     1991    61    -0.0174
## 2 CAC     1992   127     0.0520
## 3 CAC     1993   128     0.217 
## 4 CAC     1994   117    -0.132 
## # ... with 28 more rows
```

datos2

```
datos2
```

```
## # A tibble: 64 x 4
##   Accion  Anno tipo           Valor
##   <chr>  <dbl> <chr>          <dbl>
## 1 CAC     1991 Veces         61
## 2 CAC     1991 Porcentual    -0.0174
## 3 CAC     1992 Veces        127
## 4 CAC     1992 Porcentual     0.0520
## # ... with 60 more rows
```

datos3

```
datos3
```

```
## # A tibble: 32 x 3
##    Accion  Anno Veces_Porcentual
##    <chr>  <dbl> <chr>
## 1 CAC     1991 61/-0.0174300541516245
## 2 CAC     1992 127/0.0519907118989636
## 3 CAC     1993 128/0.217237876276533
## 4 CAC     1994 117/-0.131590567870706
## # ... with 28 more rows
```

datos4a

Veces

```
datos4a
```

```
## # A tibble: 4 x 9
##   Accion `1991` `1992` `1993` `1994` `1995` `1996` `199
## * <chr>   <int>  <int>  <int>  <int>  <int>  <int>  <int
## 1 CAC        61    127    128    117    119    135    13
## 2 DAX        60    125    141    131    131    147    14
## 3 FTSE       63    123    125    121    140    143    13
## 4 SMI        65    147    152    130    138    145    14
```

datos4b

Porcentual

```
datos4b
```

```
## # A tibble: 4 x 9
##   Accion   `1991`  `1992` `1993`  `1994`  `1995` `1996`
## * <chr>     <dbl>   <dbl>  <dbl>   <dbl>   <dbl>  <dbl>
## 1 CAC    -0.0174   0.0520  0.217 -0.132  -0.0422  0.225
## 2 DAX    -0.0316  -0.0189  0.466 -0.0608  0.0806  0.247
## 3 FTSE   -0.00966  0.134   0.198 -0.0964  0.186   0.117
## 4 SMI    -0.00477  0.253   0.422 -0.0955  0.226   0.182
```

datos5

```
datos5
```

```
## # A tibble: 32 x 5
##   Accion Siglo Anno  Veces Porcentual
##   <chr>  <chr> <chr> <int>      <dbl>
## 1 CAC    19    91       61    -0.0174
## 2 CAC    19    92      127     0.0520
## 3 CAC    19    93      128     0.217 
## 4 CAC    19    94      117    -0.132 
## # ... with 28 more rows
```

Datos Organizados (*tidy*)

Datos Organizados (*tidy*)

Todas las familias felices se parecen, cada familia infeliz es infeliz a su manera - Leo Tolstoy (parafraseado)

Datos Organizados (*tidy*)

Todas las familias felices se parecen, cada familia infeliz es infeliz a su manera - Leo Tolstoy (parafraseado)

Tidy datasets are all alike, but every messy dataset is messy in its own way - Hadley Wickham

Principios de los datos organizados

Principios de los datos organizados

Los datos organizados siguen “3” principios básicos:

- A cada variable le corresponde una columna

Principios de los datos organizados

Los datos organizados siguen “3” principios básicos:

- A cada variable le corresponde una columna
- A cada observación le corresponde una fila

Principios de los datos organizados

Los datos organizados siguen “3” principios básicos:

- A cada variable le corresponde una columna
- A cada observación le corresponde una fila
- A cada unidad de observación le corresponde un data.frame

Principios de los datos organizados

Los datos organizados siguen “3” principios básicos:

- A cada variable le corresponde una columna
- A cada observación le corresponde una fila
- A cada unidad de observación le corresponde un data.frame

Principios de los datos organizados

Los datos organizados siguen “3” principios básicos:

- A cada variable le corresponde una columna
- A cada observación le corresponde una fila
- A cada unidad de observación le corresponde un data.frame

Corolario de las primeras dos:

- A cada valor le corresponde una única celda

A pesar de que estas definiciones son bastante intuitivas y todos los datos que hemos manejado en el curso han sido *tidy*, muchas veces se presentan un poco desestructurads en su forma cruda.

Desorganizaciones *más* usuales

- Los nombres de las columnas son valores, no nombres de variables

Desorganizaciones *más* usuales

- Los nombres de las columnas son valores, no nombres de variables
- Varios valores están guardados en una misma columna

Desorganizaciones *más* usuales

- Los nombres de las columnas son valores, no nombres de variables
- Varios valores están guardados en una misma columna
- Hay variables que se guardan tanto en filas como en columnas

Desorganizaciones *más* usuales

- Los nombres de las columnas son valores, no nombres de variables
- Varios valores están guardados en una misma columna
- Hay variables que se guardan tanto en filas como en columnas
- Varias unidades de observación están en la misma tabla

Desorganizaciones *más* usuales

- Los nombres de las columnas son valores, no nombres de variables
- Varios valores están guardados en una misma columna
- Hay variables que se guardan tanto en filas como en columnas
- Varias unidades de observación están en la misma tabla
- Una unidad de observación está guardada en varias tablas

Funciones para organizar:

Hay 6 funciones base para organizar un dataframe, vienen en el paquete `tidyr`.

Funciones para organizar:

Hay 6 funciones base para organizar un dataframe, vienen en el paquete `tidyr`. Cargue este paquete usando la función `library`.

Funciones para organizar:

Hay 6 funciones base para organizar un dataframe, vienen en el paquete `tidyr`. Cargue este paquete usando la función `library`. Estas funciones son:

función	Operación
`spread`	pasa filas actuales a columnas
`gather`	pasa columnas a filas
`separate`	separa filas
`unite`	une filas
`complete`	completa los datos
`fill`	llena los datos

Las primeras 4 son para modificar la *organización* de los dataframes, y las otras dos son para lidiar con datos faltantes.

spread

spread

Figure 1: Otro tipo de spread

spread

spread

Esta función recibe un dataframe, y el nombre de dos de sus columnas. Una que va a ser la llave (`key`), que son los valores que van a tomar las columnas del nuevo dataframe, y el valor (`value`) que van a tomar estas columnas nuevas.

```
datos2 %>%
  spread(key = tipo,value = Valor)
```

```
## # A tibble: 32 x 4
##    Accion  Anno Porcentual Veces
##    <chr>  <dbl>      <dbl> <dbl>
## 1 CAC     1991    -0.0174    61
## 2 CAC     1992     0.0520   127
## 3 CAC     1993     0.217    128
## 4 CAC     1994    -0.132    117
## # ... with 28 more rows
```

gather

gather

Esta función es en cierto sentido *inversa* a `spread`, porque pasa de varias columnas a dos. Pero debe recibir **tres** parámetros: los nombres de las columnas **nuevas** que se van a crear (una que indica la llave (`key`) y la otra que indica el valor (`value`)) y los nombres actuales que toman esas columnas usando la misma sintaxis que se usa para `select` de `dplyr`:

gather

Esta función es en cierto sentido *inversa* a `spread`, porque pasa de varias columnas a dos. Pero debe recibir **tres** parámetros: los nombres de las columnas **nuevas** que se van a crear (una que indica la llave (`key`) y la otra que indica el valor (`value`)) y los nombres actuales que toman esas columnas usando la misma sintaxis que se usa para `select` de `dplyr`:

```
datos4a %>%
  gather(key = Anno,value = Veces,`1991`:`1998`)
```

```
## # A tibble: 32 x 3
##    Accion Anno  Veces
##    <chr>  <chr> <int>
## 1 CAC    1991     61
## 2 DAX    1991     60
## 3 FTSE   1991     63
## 4 SMI    1991     65
## # ... with 28 more rows
```

gather

gather

```
datos4b %>%
  gather(key = Anno,value = Porcentaje,
         `1991`,`1992`:`1998`)
```

```
## # A tibble: 32 x 3
##   Accion Anno  Porcentaje
##   <chr>  <chr>      <dbl>
## 1 CAC    1991     -0.0174
## 2 DAX    1991     -0.0316
## 3 FTSE   1991     -0.00966
## 4 SMI    1991     -0.00477
## # ... with 28 more rows
```

gather

```
datos4b %>%
  gather(key = Anno,value = Porcentaje,
         `1991`,`1992`:`1998`)
```

```
## # A tibble: 32 x 3
##   Accion Anno  Porcentaje
##   <chr>  <chr>      <dbl>
## 1 CAC    1991    -0.0174
## 2 DAX    1991    -0.0316
## 3 FTSE   1991    -0.00966
## 4 SMI    1991    -0.00477
## # ... with 28 more rows
```

¿Qué función se puede usar para *unir* las dos tablas anteriores para formar datos1?

gather

```
datos4b %>%
  gather(key = Anno,value = Porcentaje,
         `1991`,`1992`:`1998`)
```

```
## # A tibble: 32 x 3
##   Accion Anno  Porcentaje
##   <chr>  <chr>      <dbl>
## 1 CAC    1991    -0.0174
## 2 DAX    1991    -0.0316
## 3 FTSE   1991    -0.00966
## 4 SMI    1991    -0.00477
## # ... with 28 more rows
```

¿Qué función se puede usar para *unir* las dos tablas anteriores para formar datos1? Hágalo.

separate

separate

Figure 2: Pero no para reciclar

`separate`

Esta función recibe el nombre de la columna a separar (`col`), un *vector* (`into`) de strings en que se va a separar la columna, y un parámetro (`sep`) para indicar con qué criterio se va a separar.

separate

Esta función recibe el nombre de la columna a separar (`col`), un *vector* (`into`) de strings en que se va a separar la columna, y un parámetro (`sep`) para indicar con qué criterio se va a separar.

```
datos3 %>%
  separate(col = Veces_Porcentual,convert = TRUE,
           into = c('Veces','Porcentual'),sep = '/')
```

```
## # A tibble: 32 x 4
##    Accion  Anno Veces Porcentual
##    <chr>  <dbl> <int>      <dbl>
## 1 CAC     1991    61    -0.0174
## 2 CAC     1992   127     0.0520
## 3 CAC     1993   128     0.217
## 4 CAC     1994   117    -0.132
## # ... with 28 more rows
```

¿Qué pasa si se llama sin `convert`?

separate

Esta función recibe el nombre de la columna a separar (`col`), un *vector* (`into`) de strings en que se va a separar la columna, y un parámetro (`sep`) para indicar con qué criterio se va a separar.

```
datos3 %>%
  separate(col = Veces_Porcentual,convert = TRUE,
           into = c('Veces','Porcentual'),sep = '/')
```

```
## # A tibble: 32 x 4
##    Accion  Anno Veces Porcentual
##    <chr>  <dbl> <int>      <dbl>
## 1 CAC     1991    61    -0.0174
## 2 CAC     1992   127     0.0520
## 3 CAC     1993   128     0.217
## 4 CAC     1994   117    -0.132
## # ... with 28 more rows
```

¿Qué pasa si se llama sin `convert`? ¿Si lo llama con `sep = '-'`?

unite

unite

Esta función recibe tres argumentos: `col` una columna (la nueva), las columnas que se van a unir (como en `select`), y `sep` que indica con qué character se van a separar los valores.

```
datos5 %>%
  unite(col = Anno_nuevo,Siglo,Anno,sep = '')
```

```
## # A tibble: 32 x 4
##    Accion Anno_nuevo Veces Porcentual
##    <chr>  <chr>       <int>      <dbl>
## 1 CAC    1991           61    -0.0174
## 2 CAC    1992          127     0.0520
## 3 CAC    1993          128     0.217
## 4 CAC    1994          117    -0.132
## # ... with 28 more rows
```

unite

Esta función recibe tres argumentos: `col` una columna (la nueva), las columnas que se van a unir (como en `select`), y `sep` que indica con qué character se van a separar los valores.

```
datos5 %>%
  unite(col = Anno_nuevo,Siglo,Anno,sep = '')
```

```
## # A tibble: 32 x 4
##    Accion Anno_nuevo Veces Porcentual
##    <chr>  <chr>       <int>      <dbl>
## 1 CAC    1991           61    -0.0174
## 2 CAC    1992          127     0.0520
## 3 CAC    1993          128     0.217
## 4 CAC    1994          117    -0.132
## # ... with 28 more rows
```

Hay un parámetro opcional que se llama `remove`, ¿como cambia al usarlo con `= TRUE`?

unite

Esta función recibe tres argumentos: `col` una columna (la nueva), las columnas que se van a unir (como en `select`), y `sep` que indica con qué character se van a separar los valores.

```
datos5 %>%
  unite(col = Anno_nuevo,Siglo,Anno,sep = '')
```

```
## # A tibble: 32 x 4
##    Accion Anno_nuevo Veces Porcentual
##    <chr>  <chr>       <int>      <dbl>
## 1 CAC    1991           61    -0.0174
## 2 CAC    1992          127     0.0520
## 3 CAC    1993          128     0.217
## 4 CAC    1994          117    -0.132
## # ... with 28 more rows
```

Hay un parámetro opcional que se llama `remove`, ¿como cambia al usarlo con `= TRUE`? Cambie el orden de `Siglo` y `Anno`, ¿hay diferencia?

Datos faltantes

Hay dos formas en que se pueden tener datos faltantes:

Datos faltantes

Hay dos formas en que se pueden tener datos faltantes:

- Explícitos: por lo general a través de un `NA`

Datos faltantes

Hay dos formas en que se pueden tener datos faltantes:

- Explícitos: por lo general a través de un `NA`
- Implícitos: no están presentes

Datos faltantes

Hay dos formas en que se pueden tener datos faltantes:

- Explícitos: por lo general a través de un `NA`
- Implícitos: no están presentes

Datos faltantes

Hay dos formas en que se pueden tener datos faltantes:

- Explícitos: por lo general a través de un `NA`
- Implícitos: no están presentes

En general, se puede identificar qué tipo son, sabiendo que: *explícitos son evidencia de ausencia, e implícitos son ausencia de evidencia*

Datos faltantes

Hay dos formas en que se pueden tener datos faltantes:

- Explícitos: por lo general a través de un `NA`
- Implícitos: no están presentes

En general, se puede identificar qué tipo son, sabiendo que: *explícitos son evidencia de ausencia, e implícitos son ausencia de evidencia*

¿Adónde nos topamos valores explícitos faltantes? ¿Y los implícitos?

Datos nuevos

Datos nuevos

Para esta parte, vamos a usar los `datosSalario` nuevos, que vienen en `Datos_Salarios.RData`:

```
load(file = 'Datos_Salarios.RData')
datosSalario %>% arrange(Nombre)
```

```
## # A tibble: 161 x 5
##   Genero Nombre           Anno   Mes  Salario
##   <chr>  <chr>           <int> <int>    <dbl>
## 1 male   Castillo, Omar   2018     1 1511391.
## 2 male   Castillo, Omar   2018     3 1639434.
## 3 male   Castillo, Omar   2018     4 1726227.
## 4 male   Castillo, Omar   2018     5 1597325.
## # ... with 157 more rows
```

complete

complete

Figure 3: Pero no como el cereal

complete

complete

Obtiene todas las combinaciones que se puedan de las columnas que se le indiquen:

```
completos <- datosSalario %>% complete(Nombre,Anno,Mes)
completos
```

```
## # A tibble: 180 x 5
##   Nombre           Anno   Mes Genero  Salario
##   <chr>           <int> <int> <chr>     <dbl>
## 1 Castillo, Omar  2018     1 male   1511391.
## 2 Castillo, Omar  2018     2 <NA>         NA
## 3 Castillo, Omar  2018     3 male   1639434.
## 4 Castillo, Omar  2018     4 male   1726227.
## # ... with 176 more rows
```

Con lo que se hacen explícitos los valores que se tenían antes como implícitos.

fill

`fill`

Llena los valores faltantes (explícitos!) de las columnas que se indiquen con el valor anterior o siguiente (con el parámetro `.direction`).

fill

Llena los valores faltantes (explícitos!) de las columnas que se indiquen con el valor anterior o siguiente (con el parámetro `.direction`).

```
completos %>% fill(Genero,.direction = c('up'))
```

```
## # A tibble: 180 x 5
##   Nombre           Anno   Mes Genero  Salario
##   <chr>           <int> <int> <chr>     <dbl>
## 1 Castillo, Omar  2018     1 male    1511391.
## 2 Castillo, Omar  2018     2 male         NA
## 3 Castillo, Omar  2018     3 male    1639434.
## 4 Castillo, Omar  2018     4 male    1726227.
## # ... with 176 more rows
```

fill

Llena los valores faltantes (explícitos!) de las columnas que se indiquen con el valor anterior o siguiente (con el parámetro `.direction`).

```
completos %>% fill(Genero,.direction = c('up'))
```

```
## # A tibble: 180 x 5
##   Nombre           Anno   Mes Genero  Salario
##   <chr>           <int> <int> <chr>     <dbl>
## 1 Castillo, Omar   2018     1 male    1511391.
## 2 Castillo, Omar   2018     2 male         NA
## 3 Castillo, Omar   2018     3 male    1639434.
## 4 Castillo, Omar   2018     4 male    1726227.
## # ... with 176 more rows
```

Usando `fill`, `mutate` y `group_by` llene el salario de cada observación faltante con el promedio de los meses anterior y siguiente.

Solución

```
completos %>%
  mutate(Salario_Ant = Salario,
         Salario_Sig = Salario) %>%
  group_by(Nombre) %>%
  arrange(Anno,Mes) %>%
  fill(Salario_Sig, .direction = 'up') %>%
  fill(Salario_Ant, .direction = 'down') %>%
  mutate(Salario_Sig = ifelse(is.na(Salario_Sig),0,
                              Salario_Sig),
         Salario_Ant = ifelse(is.na(Salario_Ant),0,
                              Salario_Ant)) %>%
  rowwise() %>%
  mutate(Salario = (Salario_Ant + Salario_Sig)/2) %>%
  ungroup()
```

Solución

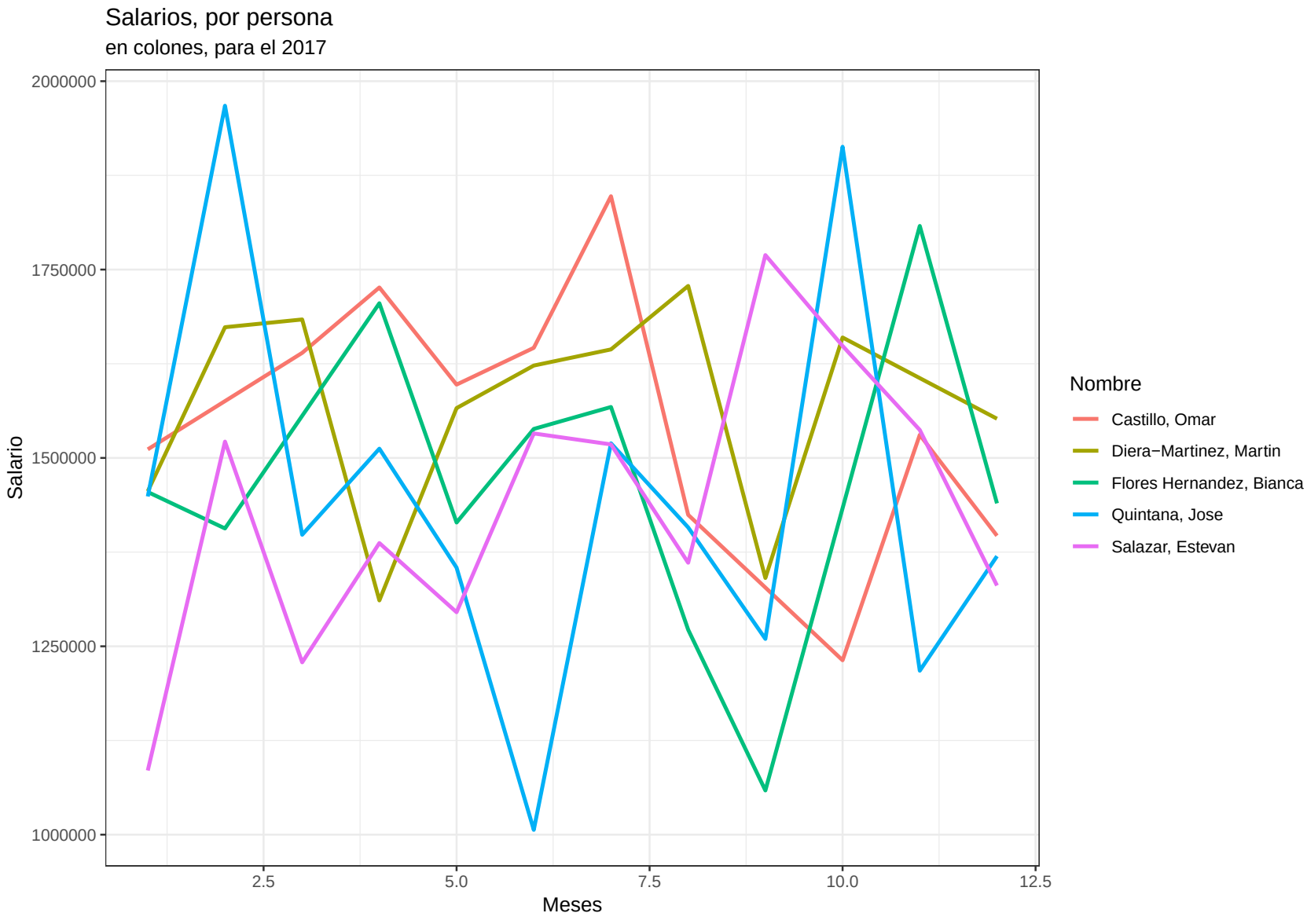
```
## # A tibble: 180 x 7
##    Nombre            Anno   Mes Genero  Salario Salario_An
##    <chr>            <int> <int> <chr>     <dbl>       <dbl
##  1 Castillo, Omar  2018     1 male   1511391.     151139
##  2 Castillo, Omar  2018     2 <NA>   1575413.     151139
##  3 Castillo, Omar  2018     3 male   1639434.     163943
##  4 Castillo, Omar  2018     4 male   1726227.     172622
##  5 Castillo, Omar  2018     5 male   1597325.     159732
##  6 Castillo, Omar  2018     6 male   1646110.     164611
##  7 Castillo, Omar  2018     7 male   1847345.     184734
##  8 Castillo, Omar  2018     8 male   1424470.     142447
##  9 Castillo, Omar  2018     9 <NA>   1327964.     142447
## 10 Castillo, Omar  2018    10 male   1231458.     123145
## 11 Castillo, Omar  2018    11 male   1530651.     153065
## 12 Castillo, Omar  2018    12 male   1396773.     139677
## # ... with 168 more rows
```

Próximamente:

`ggplot2` :)

Próximamente:

`ggplot2` :)



Ejercicio

A partir de la tabla `datos1`, obtenga las tablas `datos2`, `datos3`, `datos4a`, `datos4b` y `datos5`.