

# Graficar

Jorge Loría

Oct 11, 2017

## Base

Cargue los datos que se le entregaron, con el comando correspondiente.

```
load('Datos_Credito.RData')
```

Para esta clase vamos a ocupar dos librerías:

- ▶ dplyr

## Base

Cargue los datos que se le entregaron, con el comando correspondiente.

```
load('Datos_Credito.RData')
```

Para esta clase vamos a ocupar dos librerías:

- ▶ dplyr
- ▶ ggplot2

## Estructura de estos datos

Vamos a usar el dataframe que se indicó anteriormente, el cual tiene las siguientes columnas:

```
str(creditcardmarketing_bbm)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame': 18000 obs. of 17 variables
## $ Num_cliente      : int 1 2 3 4 5 6 7 8 9 10 ...
## $ Acepta_oferta    : chr "No" "No" "No" "No" ...
## $ Premio           : chr "Air Miles" "Air Miles" "Air Miles" "Air Miles" ...
## $ Tipo_mensajeria : chr "Letter" "Letter" "Postcard" "Letter" ...
## $ Nivel_ingresos   : chr "High" "Medium" "High" "Medium" ...
## $ Num_cuentas      : int 1 1 2 2 1 1 1 1 1 2 ...
## $ Proteccion_deficit: chr "No" "No" "No" "No" ...
## $ Rating            : chr "High" "Medium" "Medium" "High" ...
## $ Num_tarjetas_credito: int 2 2 2 1 2 3 2 4 2 3 ...
## $ Num_casas         : int 1 2 1 1 1 1 1 1 1 2 ...
## $ Tamano hogar     : int 4 5 2 4 6 4 3 4 4 4 ...
```

## Descripción de las variables

```
names(creditcardmarketing_bbm)
```

```
## [1] "Num_cliente"           "Acepta_oferta"          "Premio"  
## [4] "Tipo_mensajeria"       "Nivel_ingresos"        "Num_cuentas"  
## [7] "Proteccion_deficit"    "Rating"                 "Num_tarjetas_credito"  
## [10] "Num_casas"              "Tamano hogar"          "Duenno_hogar"  
## [13] "Balance_promedio"       "Bal_1"                  "Bal_2"  
## [16] "Bal_3"                  "Bal_4"
```

## Primer gráfico

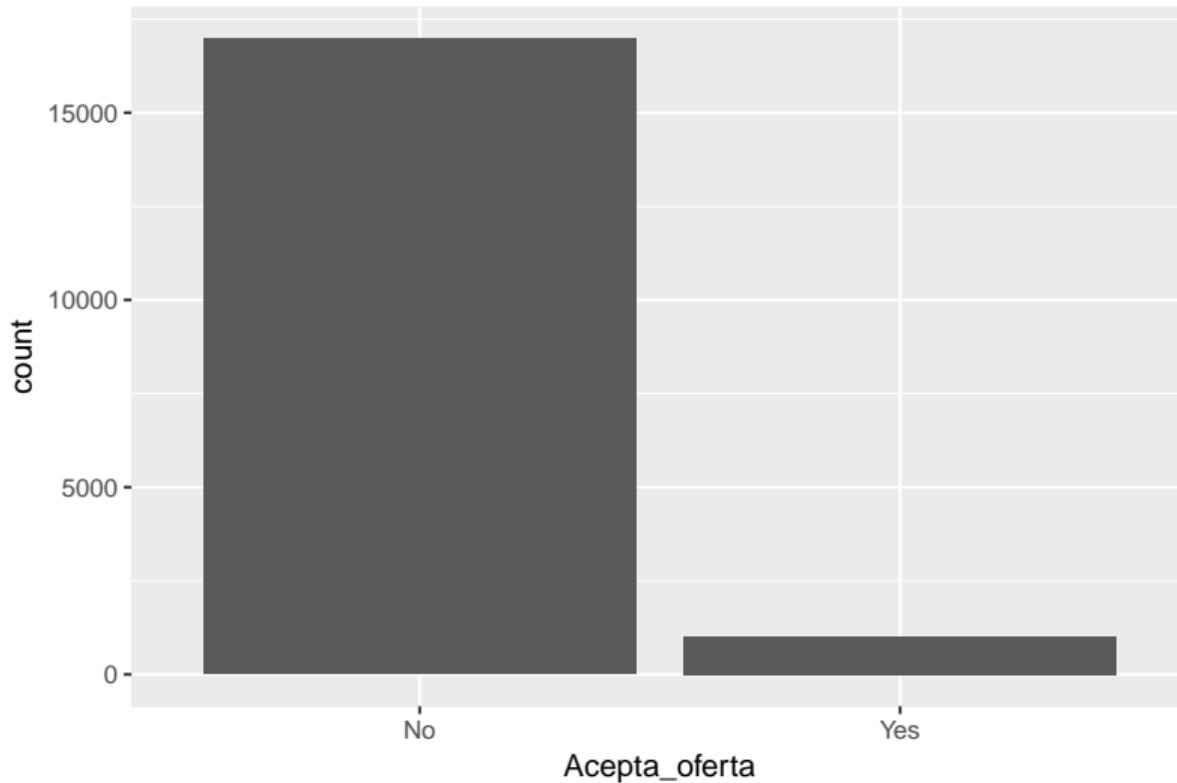
Para describir una variable categórica se puede usar un gráfico de barras, estos le dan una altura correspondiente a la cantidad de observaciones que se observen en cada categoría

## Primer gráfico

Para describir una variable categórica se puede usar un gráfico de barras, estos le dan una altura correspondiente a la cantidad de observaciones que se observen en cada categoría

```
ggplot(data = creditcardmarketing_bbm) +  
  geom_bar(mapping = aes(x = Acepta_oferta))
```

## Primer gráfico

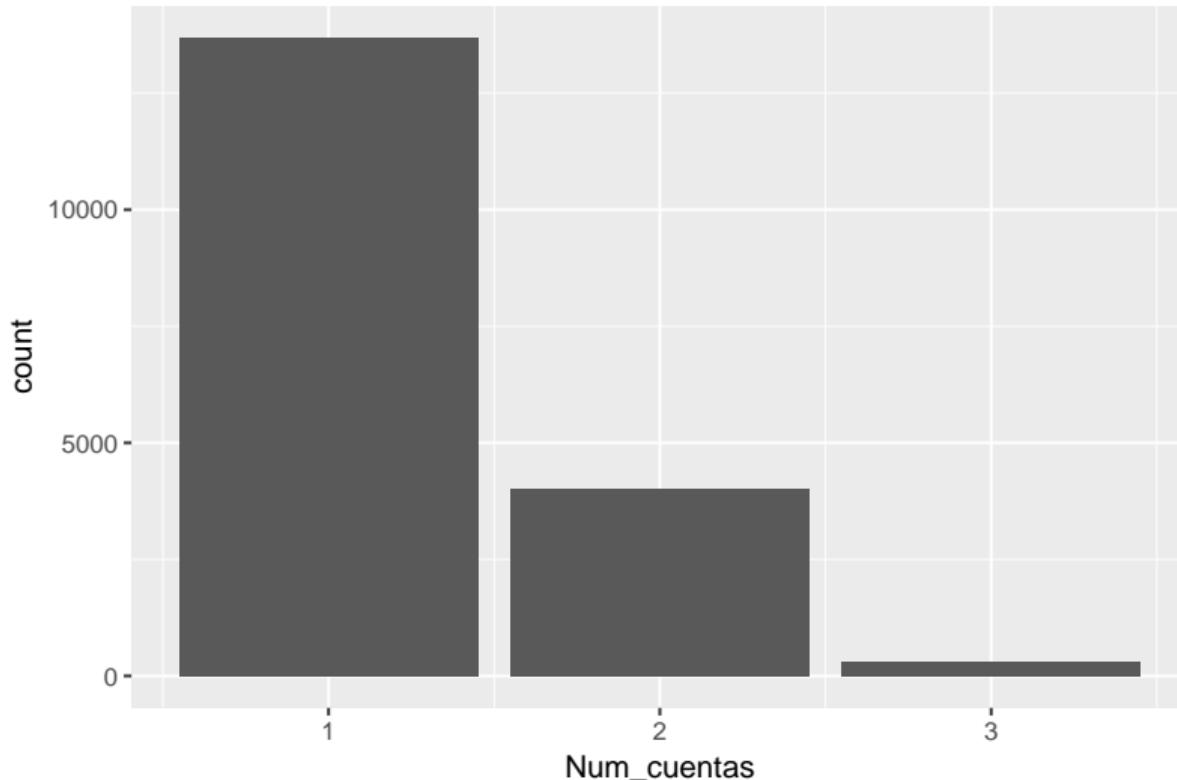


## Segundo gráfico

Repita el gráfico anterior, pero usando la variable Num\_cuentas

## Segundo gráfico

Repita el gráfico anterior, pero usando la variable Num\_cuentas



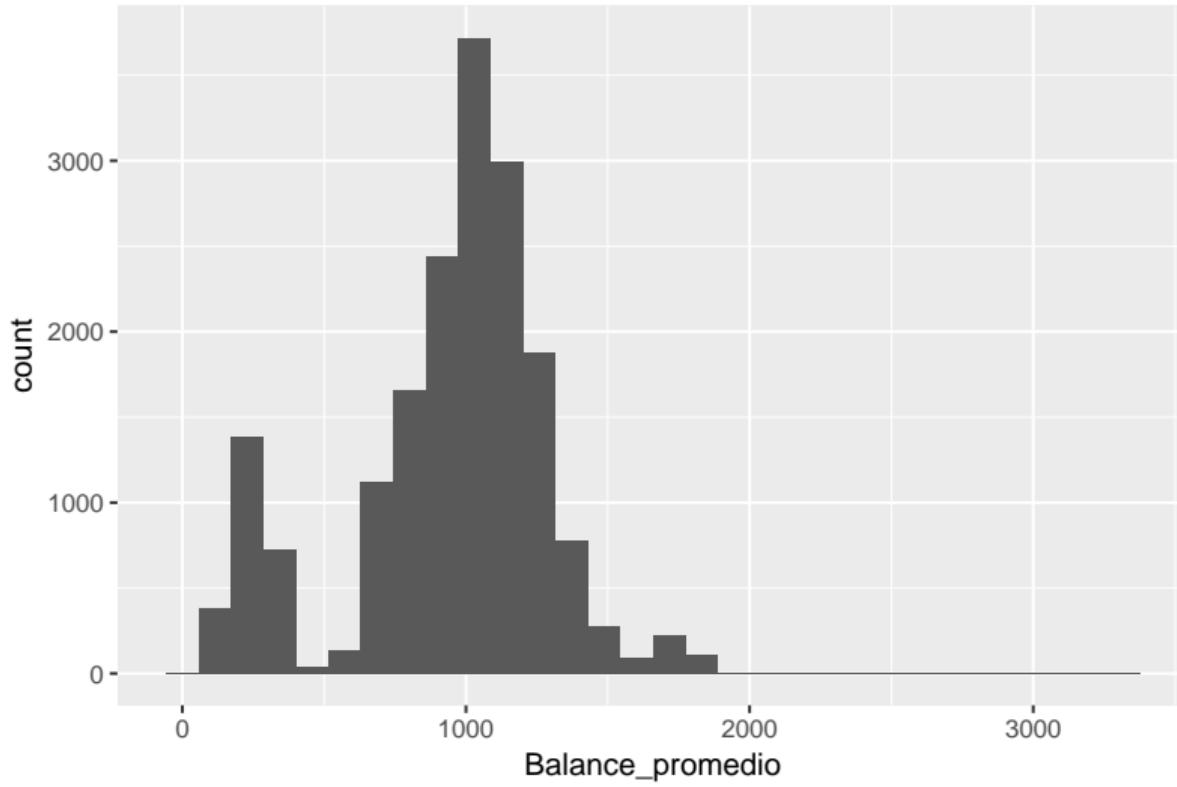
## Una variable continua

Para describir una variable continua se usan los histogramas, los cuales hacen “cajitas” de cierto ancho, y dependiendo de cuantas observaciones se encuentren en cada cajita se le asigna esa altura.

```
ggplot(data = creditcardmarketing_bbm) +  
  geom_histogram(mapping = aes(x=Balance_promedio),na.rm=T)
```

## Una variable continua

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



## Una variable continua, 2

Repita el gráfico anterior, pero usando Bal\_1.

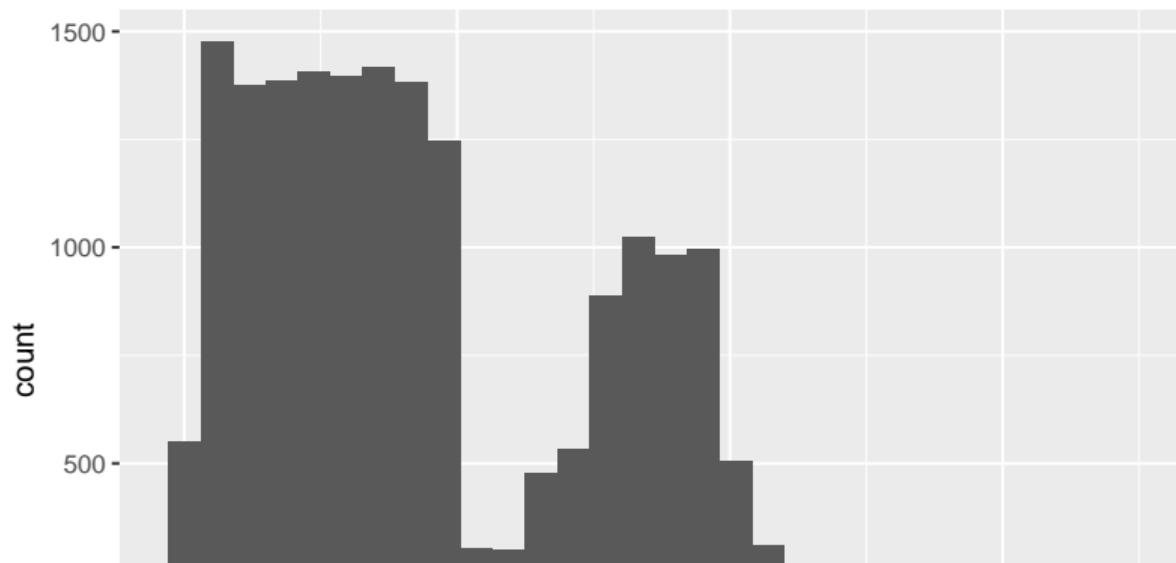
## Una variable continua, 2

Repita el gráfico anterior, pero usando Bal\_1. ¿Qué pasa si no pone el parámetro na.rm = T?

## Una variable continua, 2

Repita el gráfico anterior, pero usando Bal\_1. ¿Qué pasa si no pone el parámetro na.rm = T?

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.  
## Warning: Removed 24 rows containing non-finite values (stat_bin).
```



## Interacción de variables

Con dos variables continuas:

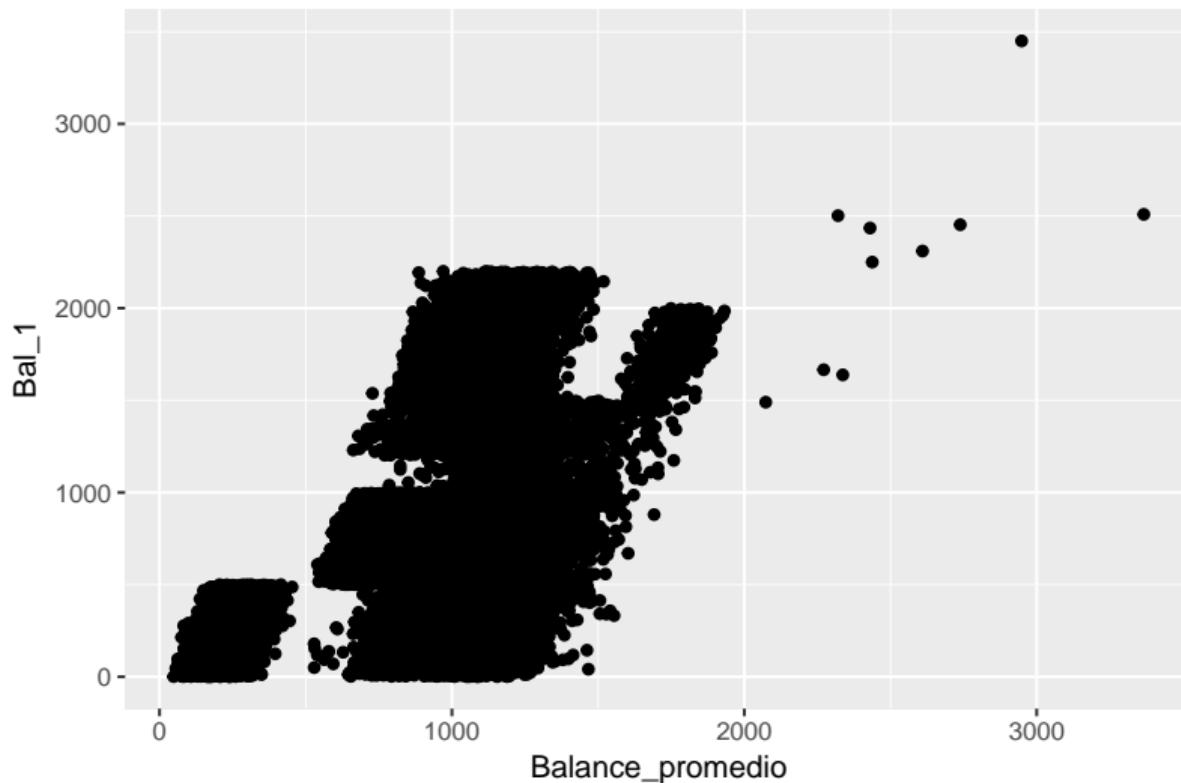
## Interacción de variables

Con dos variables continuas:

```
ggplot(data = creditcardmarketing_bbm,  
       mapping = aes(x = Balance_promedio,y = Bal_1)) +  
       geom_point(na.rm =TRUE)
```

# Interacción de variables

Con dos variables continuas:

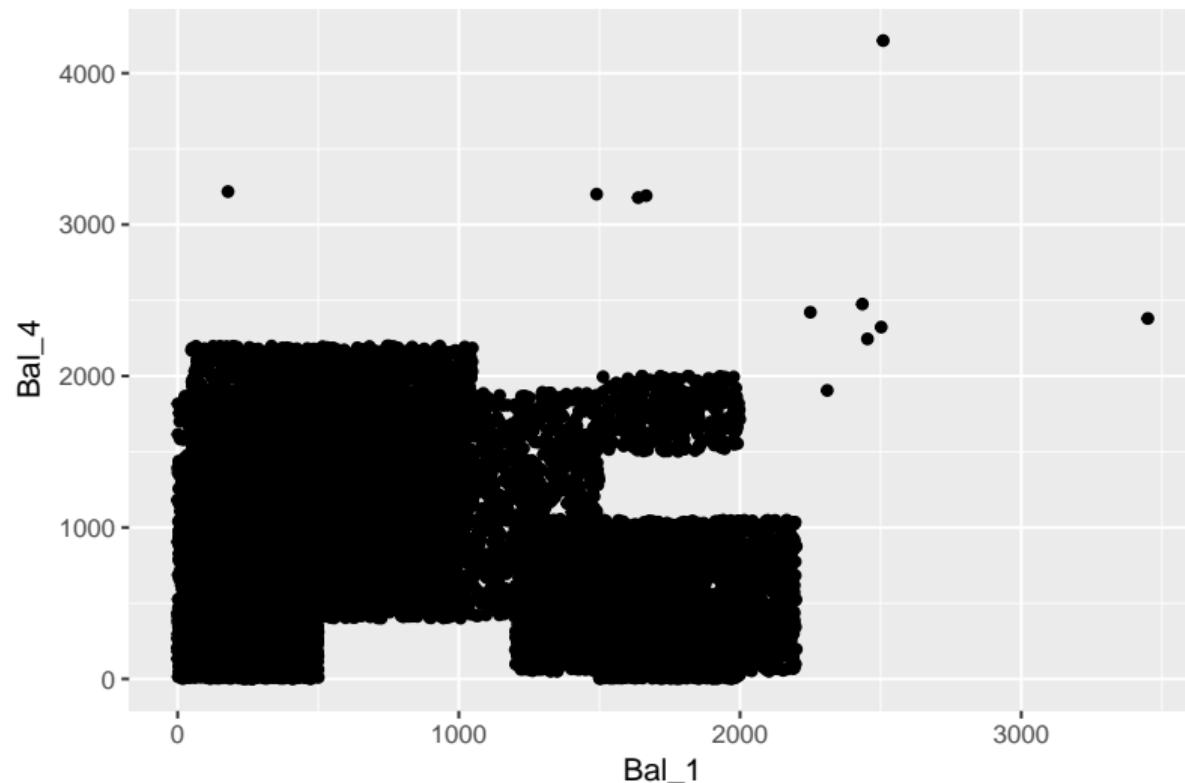


## Interacción de variables

Repita el gráfico anterior, pero usando Bal\_1 y Bal\_4

## Interacción de variables

Repita el gráfico anterior, pero usando Bal\_1 y Bal\_4



## Preguntas

¿Qué función de dplyr usaría para identificar al sujeto con Bal\_1 mayor a 3000?

## Preguntas

¿Qué función de dplyr usaría para identificar al sujeto con Bal\_1 mayor a 3000?

¿Cuántos tienen Bal\_4 mayor a 3000?

## Preguntas

¿Qué función de dplyr usaría para identificar al sujeto con Bal\_1 mayor a 3000?

¿Cuántos tienen Bal\_4 mayor a 3000?

¿Cuántos tienen alguno de los dos anteriores?

## Preguntas

¿Qué función de dplyr usaría para identificar al sujeto con Bal\_1 mayor a 3000?

¿Cuántos tienen Bal\_4 mayor a 3000?

¿Cuántos tienen alguno de los dos anteriores?

```
filter(Bal_1 > 3000)
```

## Preguntas

¿Qué función de dplyr usaría para identificar al sujeto con Bal\_1 mayor a 3000?

¿Cuántos tienen Bal\_4 mayor a 3000?

¿Cuántos tienen alguno de los dos anteriores?

`filter(Bal_1 > 3000)`

`filter(Bal_4 > 3000)`

## Preguntas

¿Qué función de dplyr usaría para identificar al sujeto con Bal\_1 mayor a 3000?

¿Cuántos tienen Bal\_4 mayor a 3000?

¿Cuántos tienen alguno de los dos anteriores?

`filter(Bal_1 > 3000)`

`filter(Bal_4 > 3000)`

`filter(Bal_1 > 3000 | Bal_4 > 3000)`

## Preguntas

¿Qué función de dplyr usaría para identificar al sujeto con Bal\_1 mayor a 3000?

¿Cuántos tienen Bal\_4 mayor a 3000?

¿Cuántos tienen alguno de los dos anteriores?

`filter(Bal_1 > 3000)`

`filter(Bal_4 > 3000)`

`filter(Bal_1 > 3000 | Bal_4 > 3000)`

Agregar colores

## Agregar colores



Figure 1: A pintar!

## Agregar colores

Si se quieren agregar colores, se agrega en el aes() el parámetro color = <VARIABLE\_A\_USAR> (o colour):

## Agregar colores

Si se quieren agregar colores, se agrega en el aes() el parámetro color = <VARIABLE\_A\_USAR> (o colour):

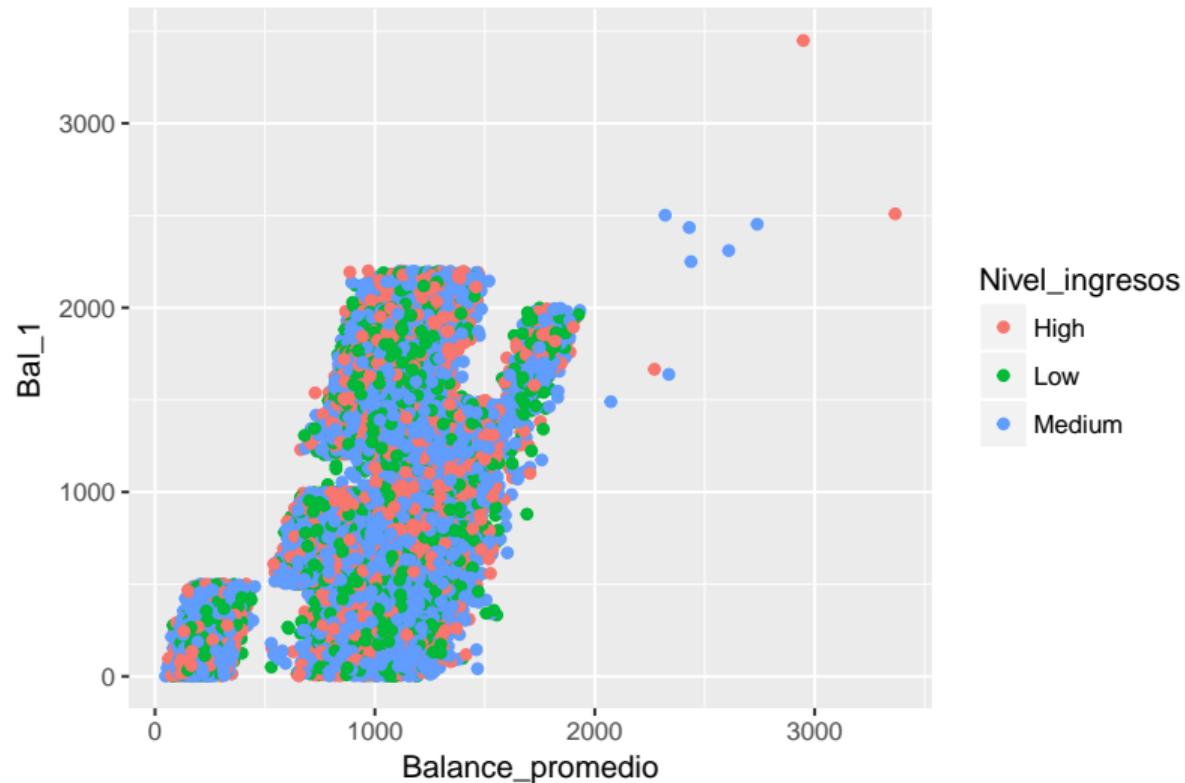
```
ggplot(data = creditcardmarketing_bbm) +  
  geom_point(mapping = aes(x = Balance_promedio,  
                           y = Bal_1,  
                           color = Nivel_ingresos),  
             na.rm = TRUE)
```

Agregar colores

Y queda coloreado...

## Agregar colores

Y queda coloreado...

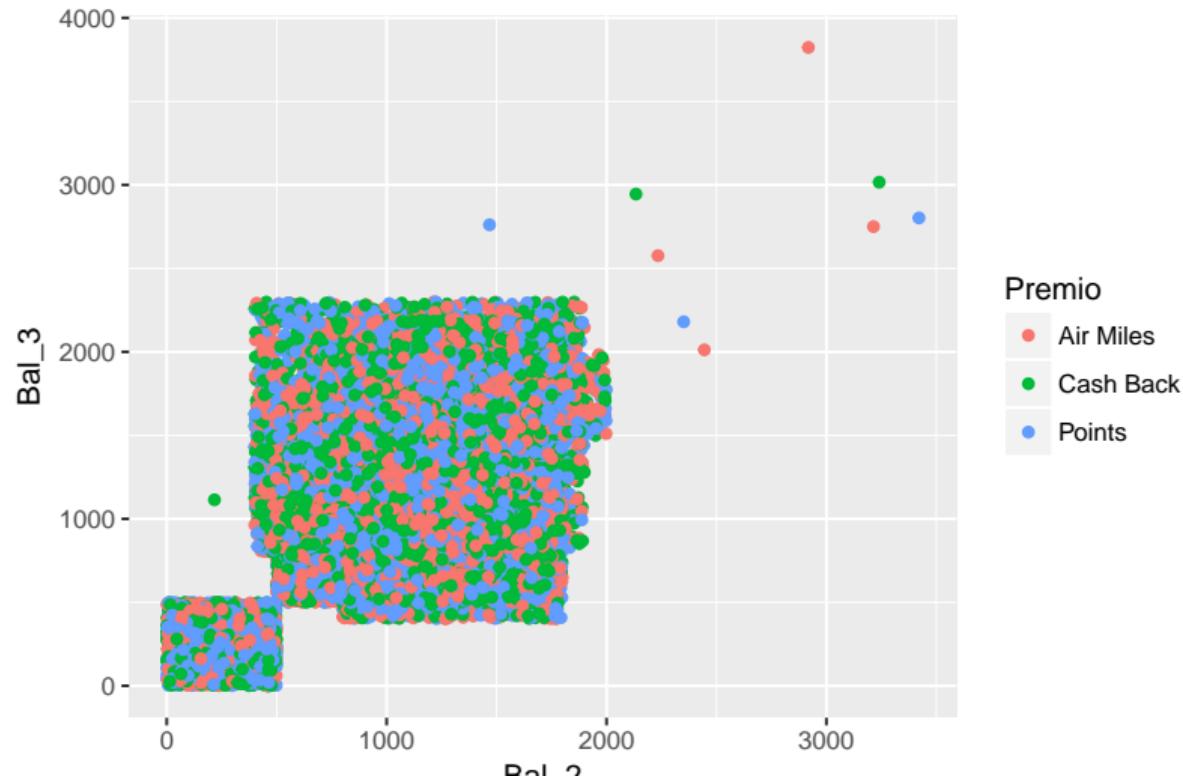


## Ejercicio colores

Repita el gráfico anterior, pero usando Bal\_2, en el eje X, Bal\_3 en el eje Y, y coloreando con la columna Premio.

## Ejercicio colores

Repita el gráfico anterior, pero usando Bal\_2, en el eje X, Bal\_3 en el eje Y, y coloreando con la columna Premio.



# Transparencia



# RENDICIÓN DE CUENTAS

Figure 2: Pero no política

## Transparencia

Muchas veces quedan imágenes como la anterior en la que quedan manchas de colores en la imagen y no se entiende bien

## Transparencia

Muchas veces quedan imágenes como la anterior en la que quedan manchas de colores en la imagen y no se entiende bien, para esto se usa el parámetro alpha, para indicar el nivel de transparencia que se quiere en la imagen a usar:

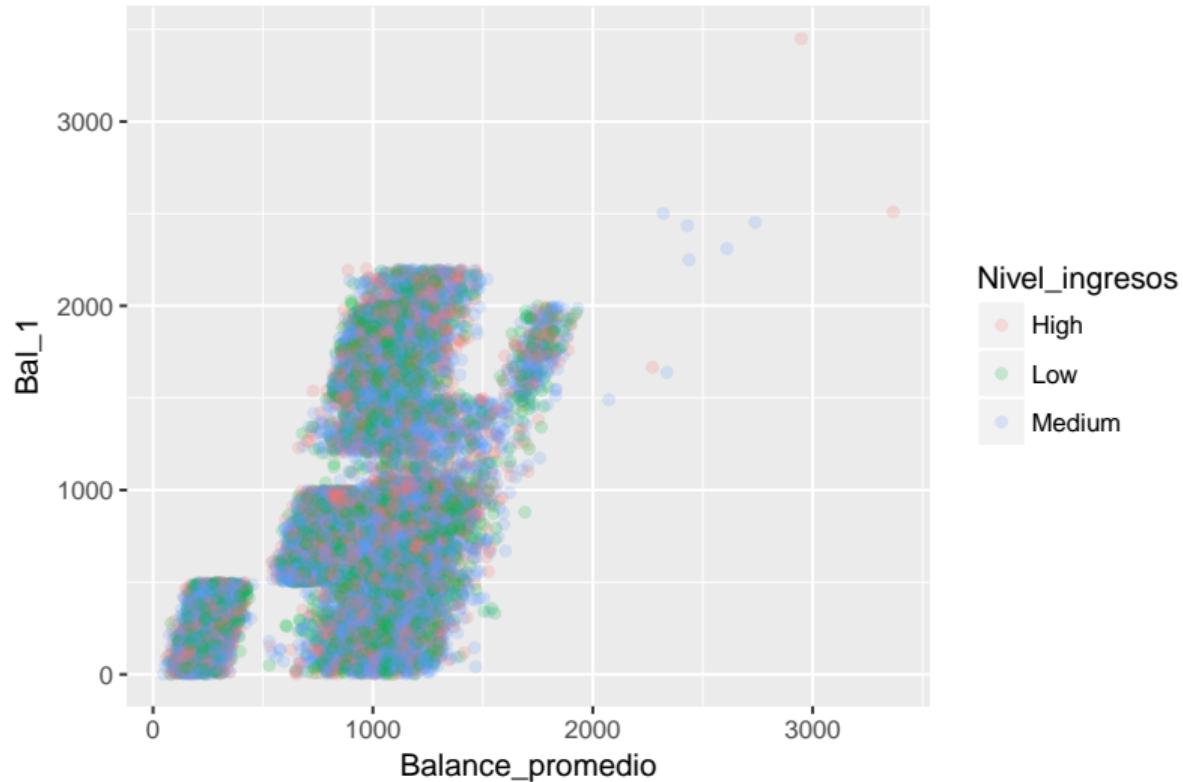
## Transparencia

Muchas veces quedan imágenes como la anterior en la que quedan manchas de colores en la imagen y no se entiende bien, para esto se usa el parámetro alpha, para indicar el nivel de transparencia que se quiere en la imagen a usar:

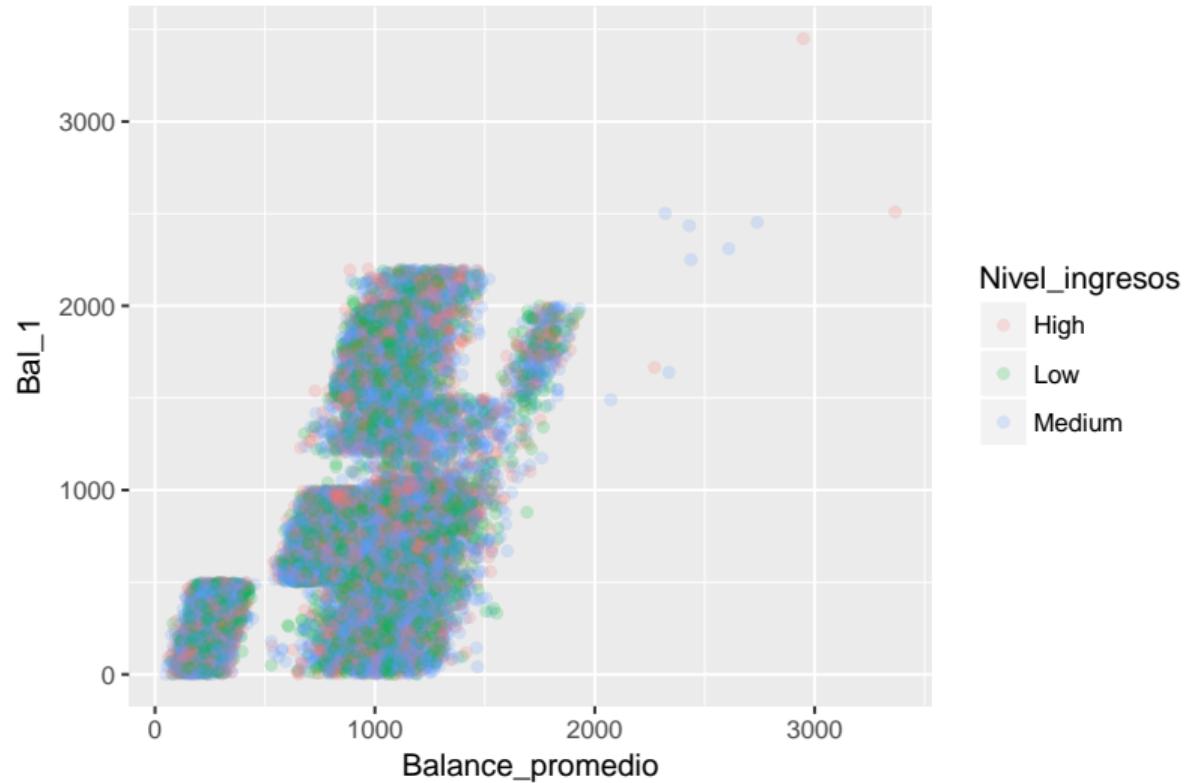
```
ggplot(data = creditcardmarketing_bbm) +  
  geom_point(mapping = aes(x = Balance_promedio,  
                           y = Bal_1, color = Nivel_ingresos),  
             alpha = 1/5,  
             na.rm = TRUE)
```

Note que alpha es un parámetro de la función geom\_point, no de la función aes

# Transparencia



# Transparencia



Con esto se interpreta mejor en *cuales* lugares están concentrados los balances.

Tamaño

Tamaño

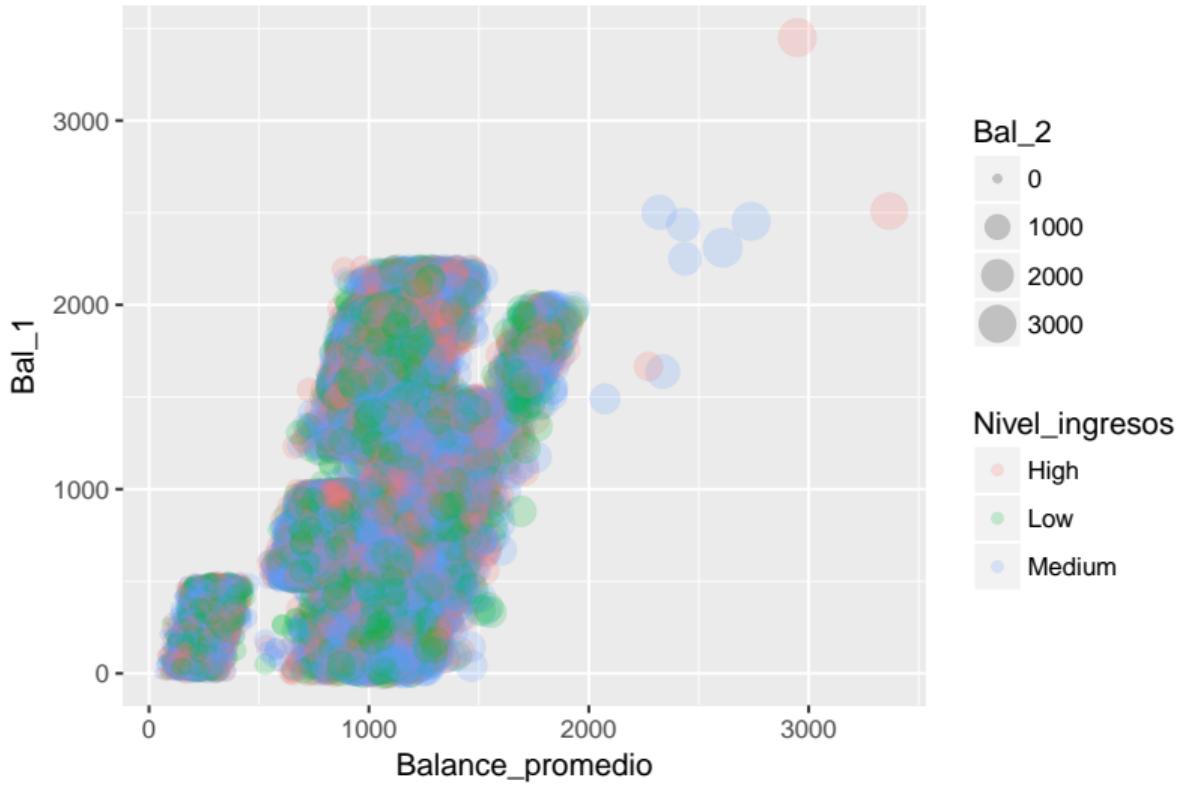
Importa

## Tamaño

Existen bastantes más parámetros que se pueden incluir, entre estos está `size`, que puede definirse como una constante o bien, que dependa de otra variable:

```
ggplot(data = creditcardmarketing_bbm) +  
  geom_point(mapping = aes(x = Balance_promedio,  
                           y = Bal_1,color = Nivel_ingresos,  
                           size = Bal_2,),  
             alpha = 1/5,  
             na.rm = TRUE)
```

# Tamaño

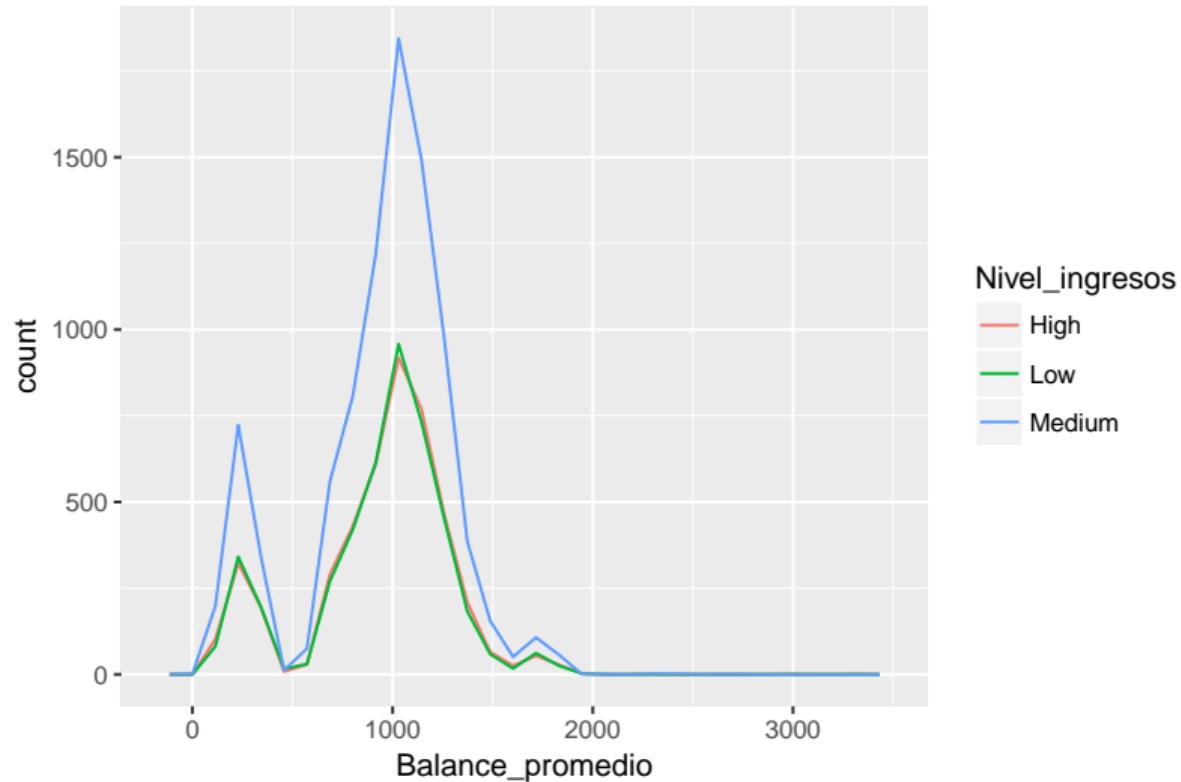


## Variables continuas vs discretas

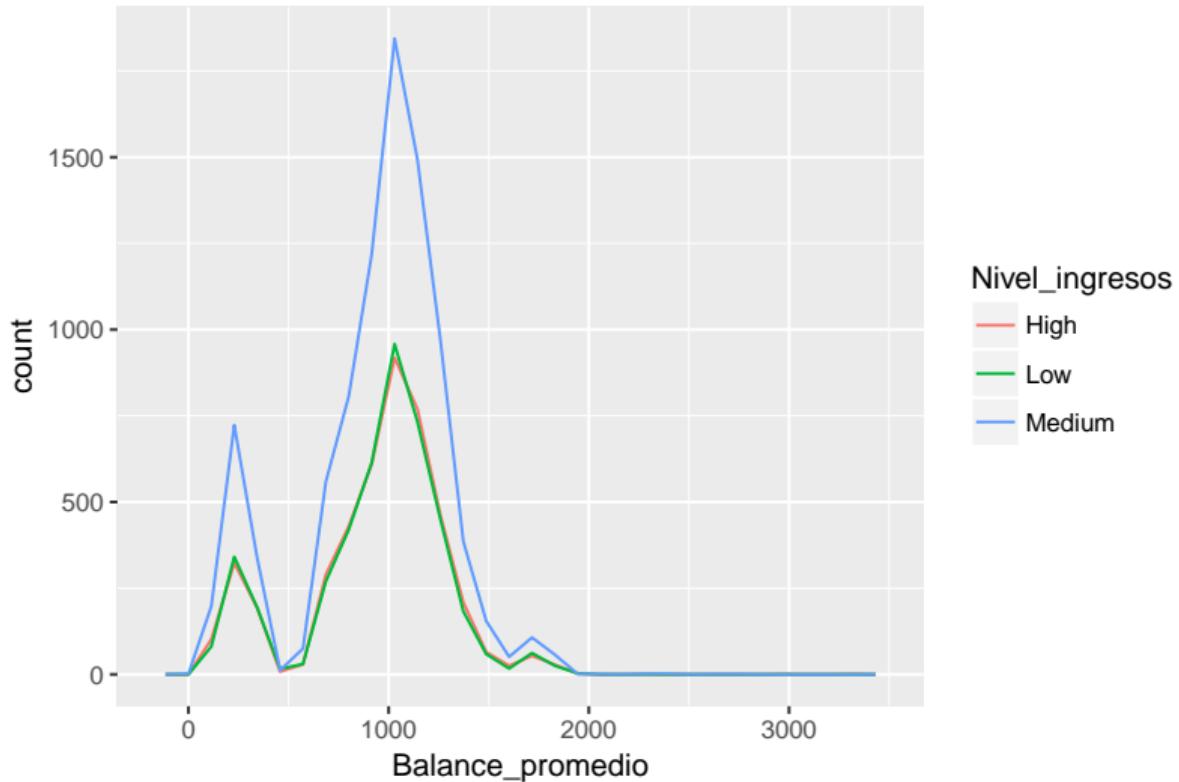
Para graficar variables continuas vs las variables discretas se puede usar la función `geom_freqpoly`, que es muy similar a la función del `histogram` en el sentido de que muestra cual es la distribución de las variables. Solo que es más claro para agrupar en distintos grupos que en el histograma.

```
ggplot(data = creditcardmarketing_bbm) +  
  geom_freqpoly(mapping = aes(x = Balance_promedio,  
                               colour = Nivel_ingresos),  
                na.rm = TRUE, bins = 30)
```

## Variables continuas vs discretas



## Variables continuas vs discretas



Parece que las distribuciones son similares entre los niveles de ingreso.

Verlo como porcentaje:

Verlo como porcentaje:

Para graficarlo como porcentaje se incluye el parámetro `y=..density..`

## Verlo como porcentaje:

Para graficarlo como porcentaje se incluye el parámetro `y=..density..`

```
ggplot(data = creditcardmarketing_bbm) +  
  geom_freqpoly(mapping = aes(x = Balance_promedio,  
                               colour = Nivel_ingresos,  
                               y = ..density..),  
                na.rm = TRUE,  
                bins = 30)
```

El parámetro `bins`, está predefinido como 30, y da un aviso de que vale eso. Por lo que lo ponemos explícito de que vale 30. En cursos avanzados se ven métodos para elegir este parámetro apropiadamente.

## Verlo como porcentaje:

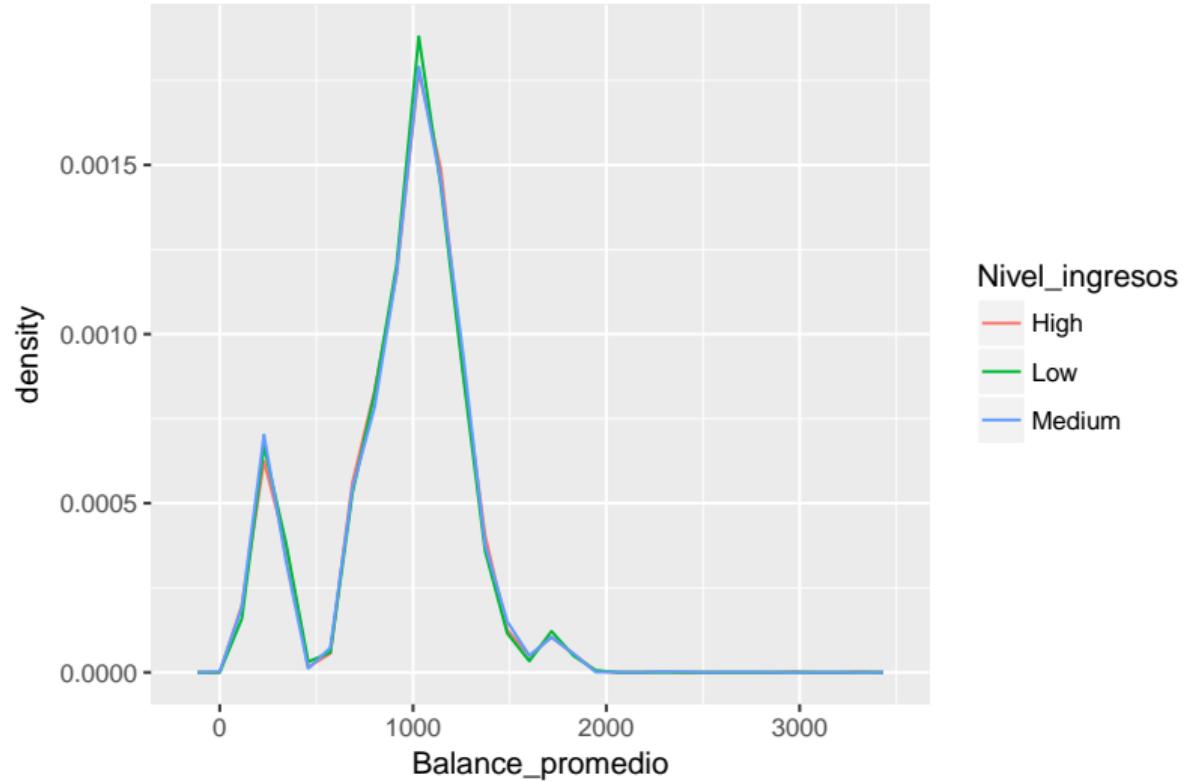
Para graficarlo como porcentaje se incluye el parámetro `y=..density..`

```
ggplot(data = creditcardmarketing_bbm) +  
  geom_freqpoly(mapping = aes(x = Balance_promedio,  
                               colour = Nivel_ingresos,  
                               y = ..density..),  
                na.rm = TRUE,  
                bins = 30)
```

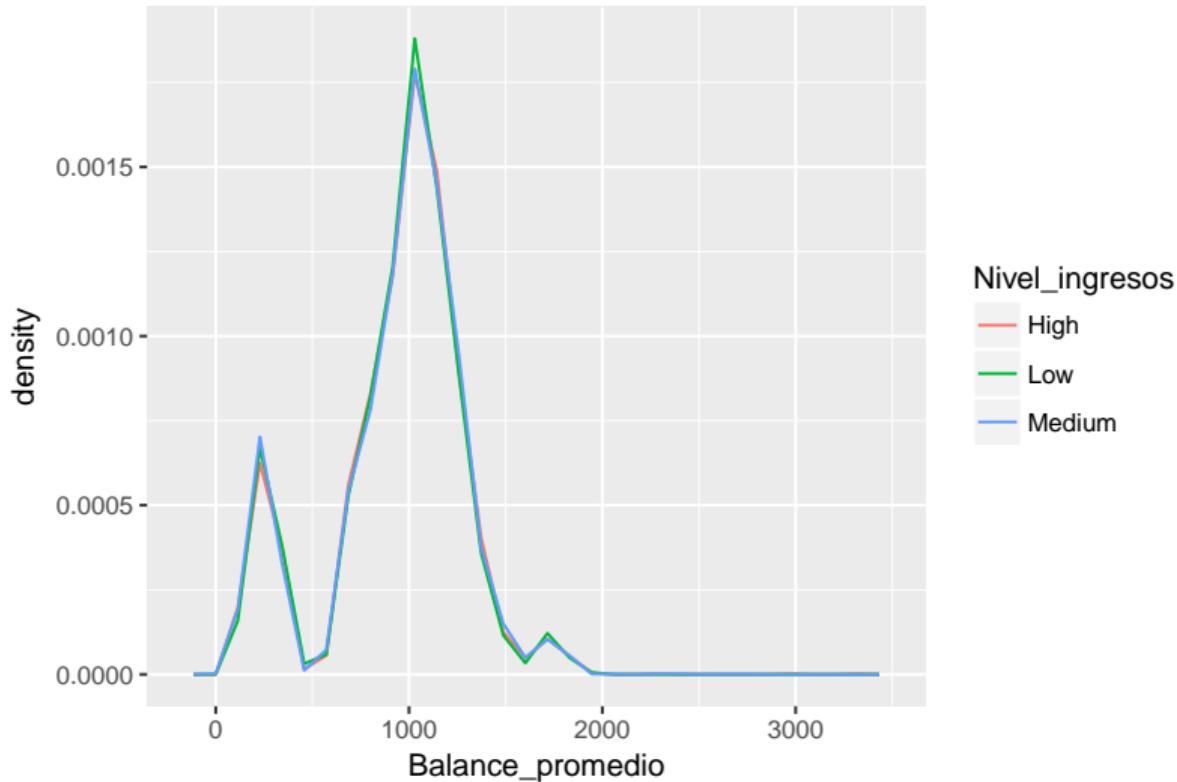
El parámetro `bins`, está predefinido como 30, y da un aviso de que vale eso. Por lo que lo ponemos explícito de que vale 30. En cursos avanzados se ven métodos para elegir este parámetro apropiadamente. Si intenta no incluirlo, se indica que se pone como 30.

Verlo como porcentaje

## Verlo como porcentaje



## Verlo como porcentaje



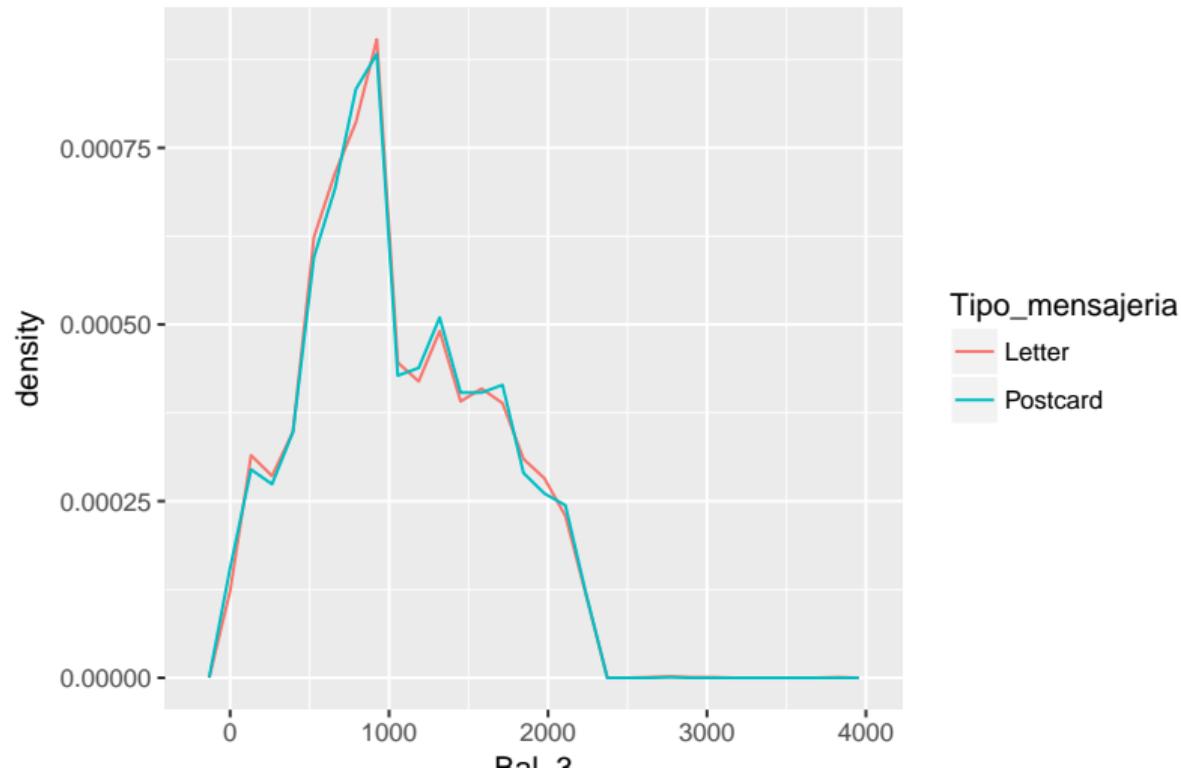
Esto nos indica que el balance promedio no depende del nivel de ingresos.

## Ejercicio

Repita el gráfico anterior, usando el Bal\_3 agrupando con el tipo de mensajería.

## Ejercicio

Repita el gráfico anterior, usando el Bal\_3 agrupando con el tipo de mensajería. ¿Hay mucha diferencia?



## Otro ejercicio

En el gráfico anterior, ¿qué pasa si pone el parámetro `bins = 20`? ¿Igual a 40? ¿Igual a 2?

## Dos variables categóricas

## Dos variables categóricas

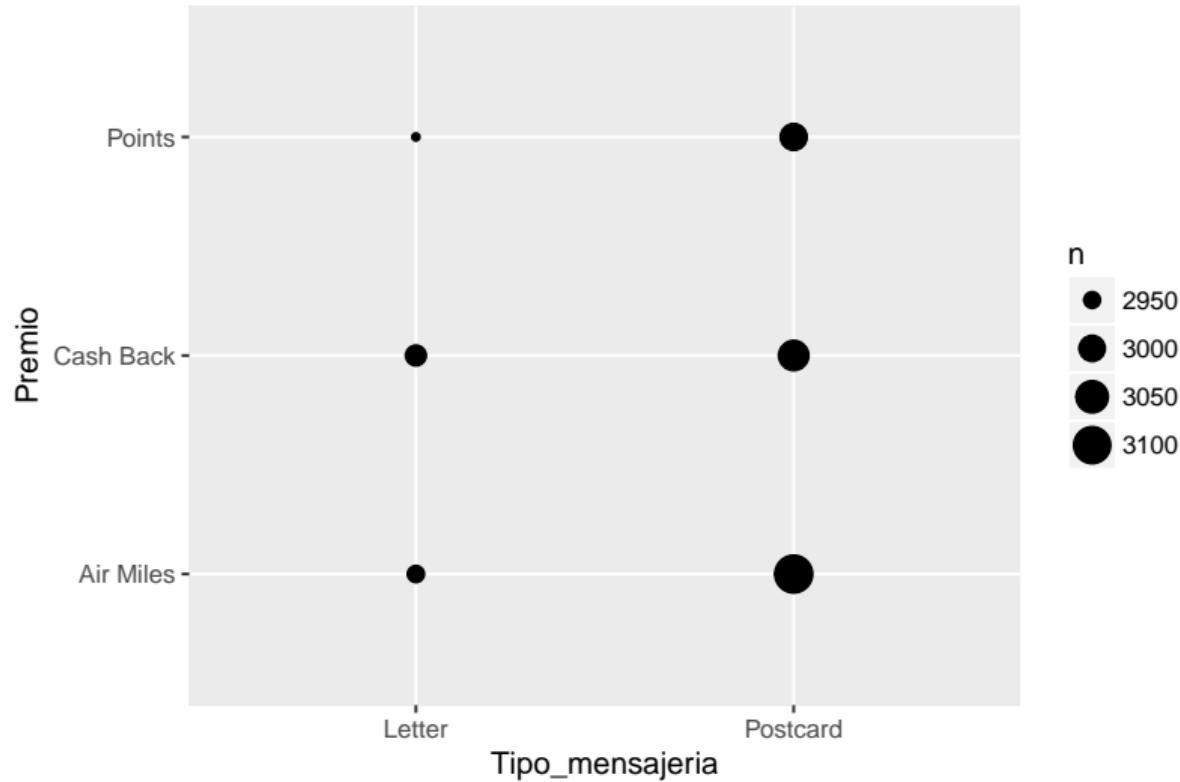
Otro tipo de relación que se puede explorar es entre dos variables categóricas, usando la función `geom_count`, la cual agrupa entre ambas categorías y cuenta cuantos hay en cada combinación:

## Dos variables categóricas

Otro tipo de relación que se puede explorar es entre dos variables categóricas, usando la función `geom_count`, la cual agrupa entre ambas categorías y cuenta cuantos hay en cada combinación:

```
ggplot(creditcardmarketing_bbm) +  
  geom_count(aes(Tipo_mensajeria,Premio))
```

## Dos variables categóricas

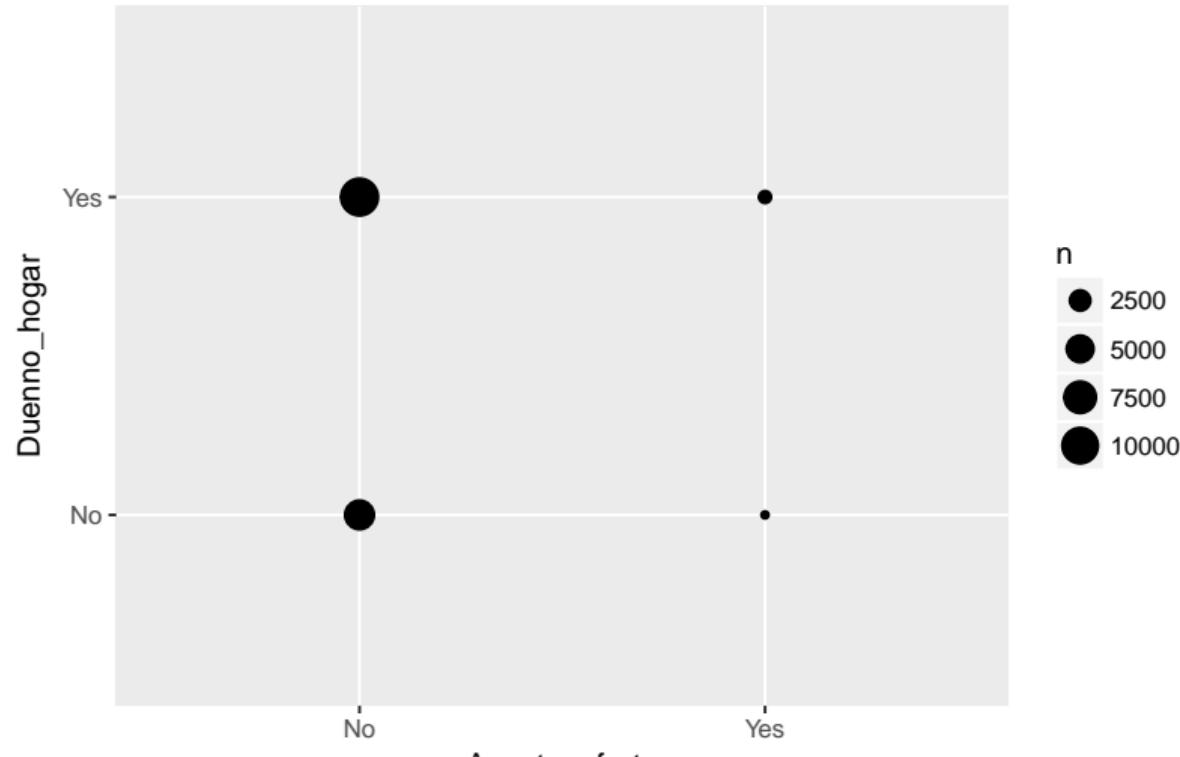


## Ejercicio count

Repita el ejercicio anterior, pero revisando la relación entre Acepta\_oferta y Duenno\_hogar.

## Ejercicio count

Repita el ejercicio anterior, pero revisando la relación entre Acepta\_oferta y Duenno\_hogar.



## Sintaxis general de un ggplot

Todos los gráficos que se hacen en ggplot, siguen la siguiente sintaxis:

## Sintaxis general de un ggplot

Todos los gráficos que se hacen en ggplot, siguen la siguiente sintaxis:

```
ggplot(data = <Mis_Datos>) +  
<GEOM_FUNCION>(mapping = aes(<EXPLICACION_VARIABLES_A_USAR>))
```

## Sintaxis general de un ggplot

Todos los gráficos que se hacen en ggplot, siguen la siguiente sintaxis:

```
ggplot(data = <Mis_Datos>) +  
<GEOM_FUNCION>(mapping = aes(<EXPLICACION_VARIABLES_A_USAR>))
```

Se le pueden agregar más *capas*, como vamos a ver proximamente...

## Lineas

Para mostrar progreso a través del tiempo de alguna variable o algo por el estilo, se pueden usar líneas, con la función `geom_line`:

```
library(tidyr)

creditos_Mes <- creditcardmarketing_bbm %>%
  gather(key = Mes,value = Balance,num_range('Bal_',1:4)) %>%
  separate(col = Mes,into = c('Bal','Mes'),sep = '_',convert = TRUE) %>%
  select(-Bal) %>%
  filter(Num_cliente < 10)

ggplot(data = creditos_Mes,
        aes(x = Mes,y = Balance,
            color = factor(Num_cliente))) +
  geom_line()
```

## Lineas

Para mostrar progreso a través del tiempo de alguna variable o algo por el estilo, se pueden usar líneas, con la función `geom_line`:

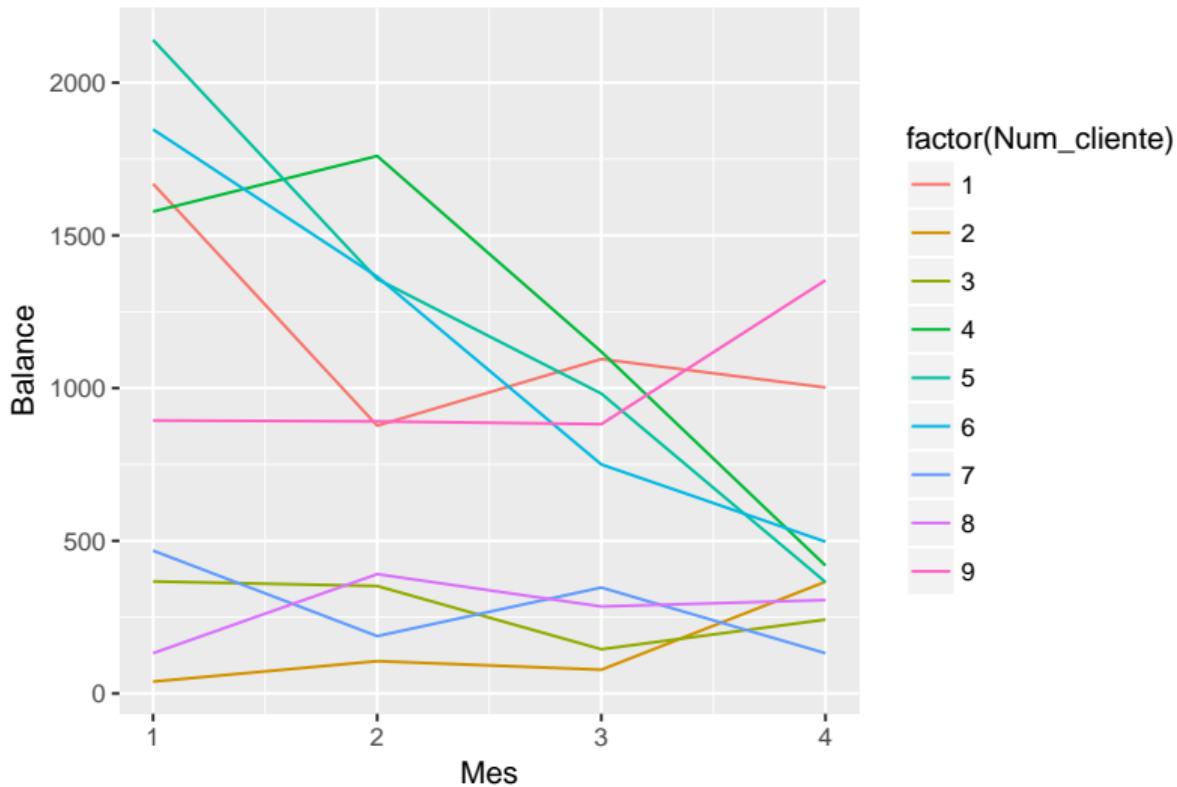
```
library(tidyr)

creditos_Mes <- creditcardmarketing_bbm %>%
  gather(key = Mes,value = Balance,num_range('Bal_',1:4)) %>%
  separate(col = Mes,into = c('Bal','Mes'),sep = '_',convert = TRUE) %>%
  select(-Bal) %>%
  filter(Num_cliente < 10)

ggplot(data = creditos_Mes,
        aes(x = Mes,y = Balance,
            color = factor(Num_cliente))) +
  geom_line()
```

¿Qué hace el código anterior?

# Lines



## Ejercicio lineas

## Ejercicio lineas

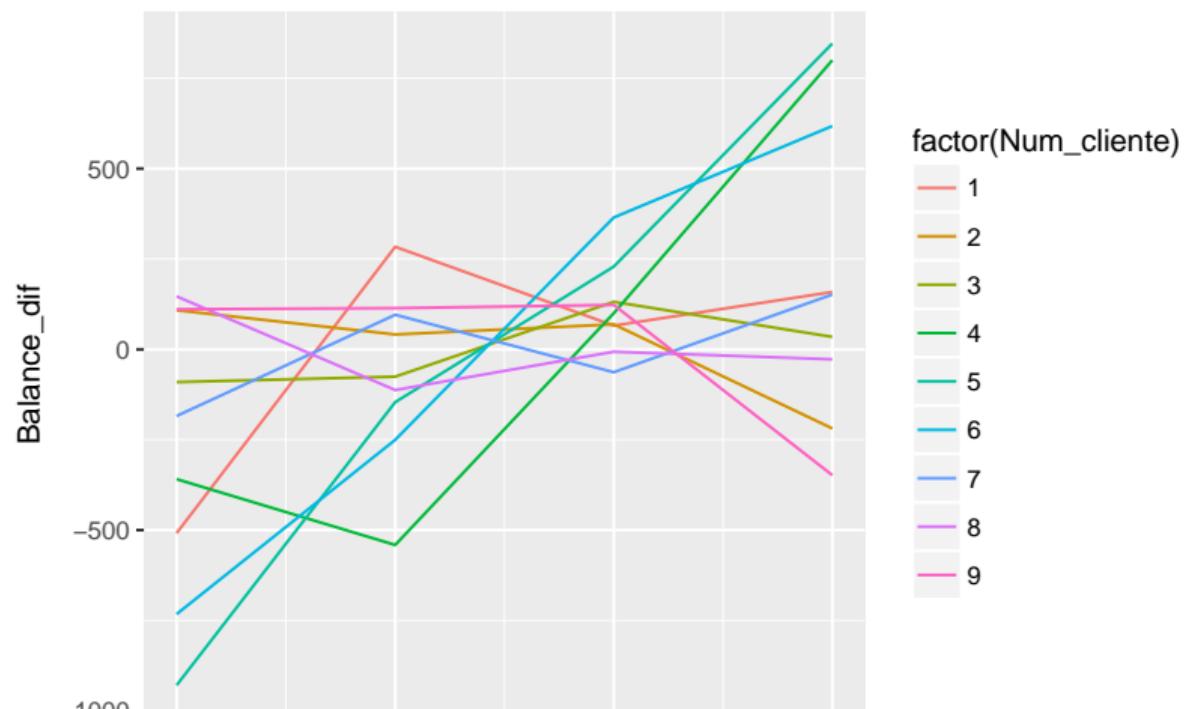
¿Qué pasa si se llama sin poner el factor alrededor de Num\_cliente?

## Ejercicio lineas

¿Qué pasa si se llama sin poner el factor alrededor de Num\_cliente? Haga una nueva columna, que indique la diferencia entre Balance\_promedio y Balance. Grafique esta diferencia respecto a estos meses.

## Ejercicio lineas

¿Qué pasa si se llama sin poner el factor alrededor de Num\_cliente? Haga una nueva columna, que indique la diferencia entre Balance\_promedio y Balance. Grafique esta diferencia respecto a estos meses.



# Capas

# Capas



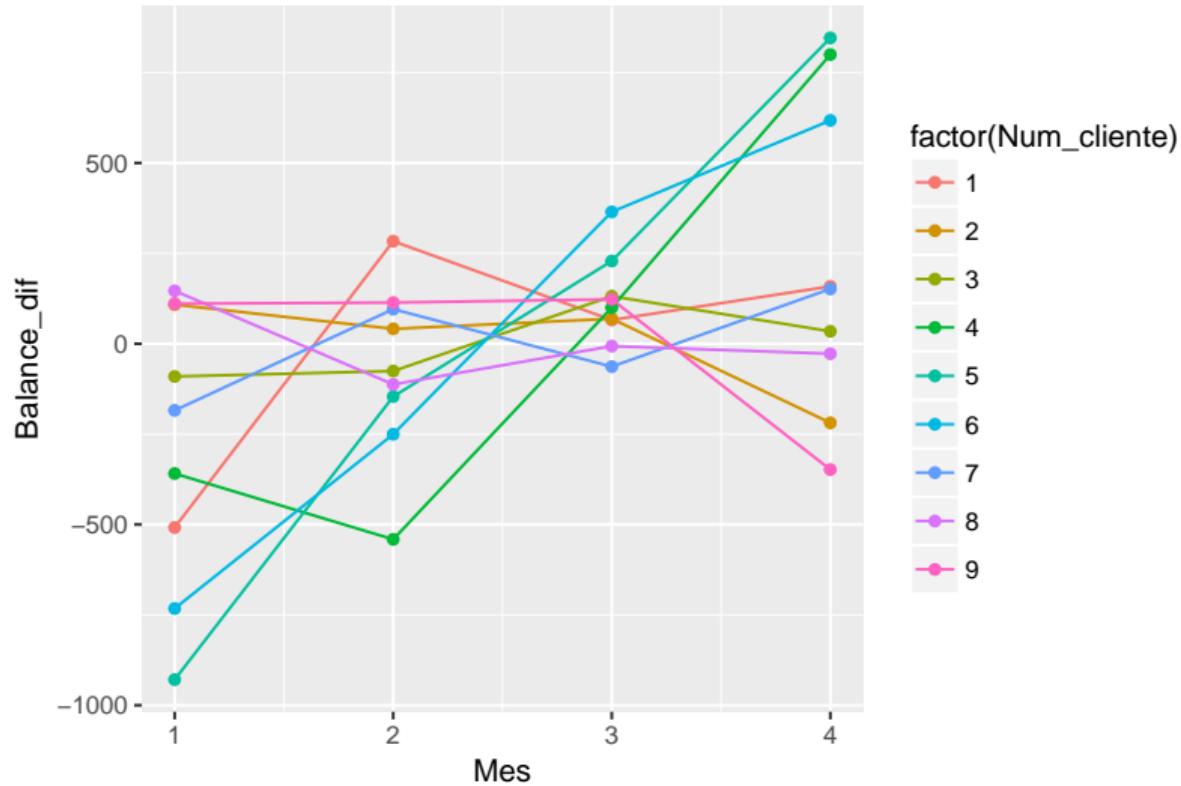
Figure 3: Pero no para la lluvia

## Capas:

Hasta el momento nos hemos limitado a sumar una capa, pero podemos agregarle más capas, para recalcar ciertos valores:

```
creditos_Mes %>%
  mutate(Balance_dif = Balance_promedio - Balance) %>%
  ggplot(mapping = aes(x = Mes, y = Balance_dif,
                        color = factor(Num_cliente))) +
  geom_line() +
  geom_point()
```

# Capas:

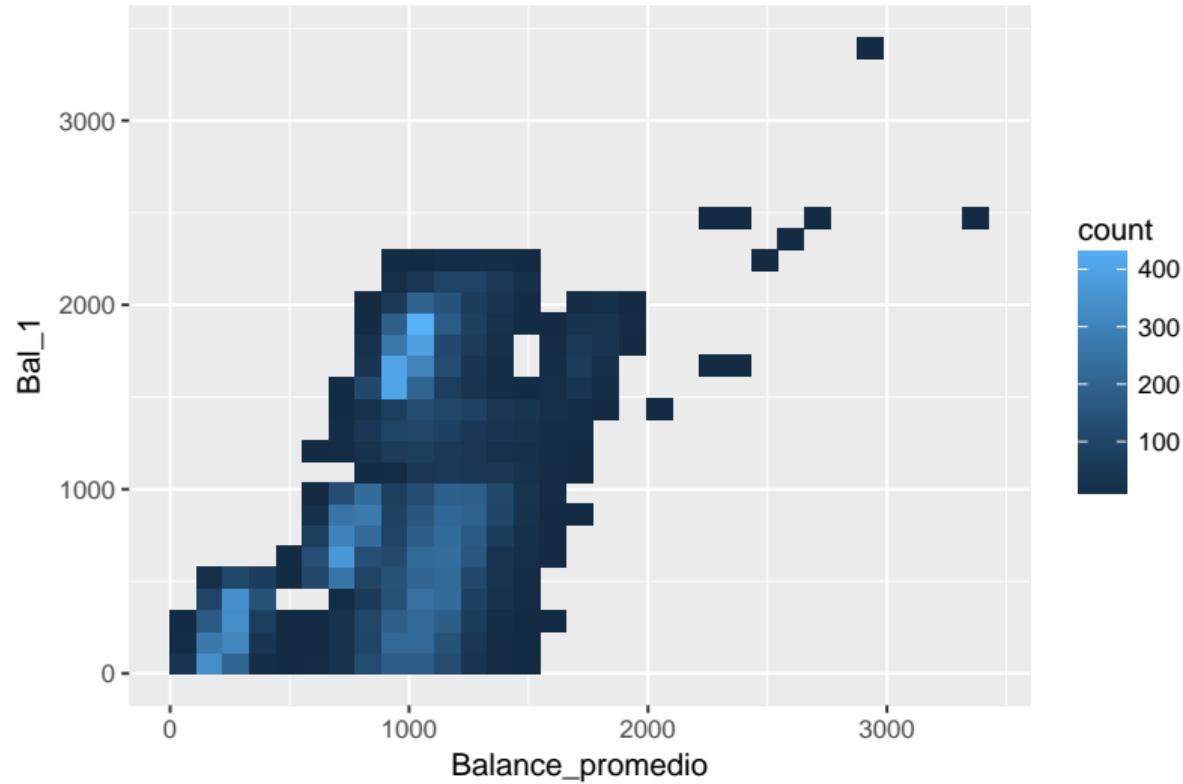


## Otro tipo de gráfico:

Para comprender mejor la relación entre dos variables continuas, se puede usar la función geom\_bin2d:

```
creditcardmarketing_bbm %>%
  ggplot(aes(x = Balance_promedio, y = Bal_1)) +
  geom_bin2d(na.rm = TRUE)
```

## Otro tipo de gráfico:



## Ejercicio geom\_bin2d

Esta función recibe un parámetro que se llama `binwidth`, que es un vector que indica cual es el grosor de los bins que va a hacer (similar al histograma), para contar.

## Ejercicio geom\_bin2d

Esta función recibe un parámetro que se llama `binwidth`, que es un vector que indica cual es el grosor de los `bins` que va a hacer (similar al histograma), para contar. Repita 3 veces el gráfico anterior, e incluya este parámetro usando los siguientes pares:

- ▶ `c(20,20)`
- ▶ `c(100,20)`
- ▶ `c(30,50)`

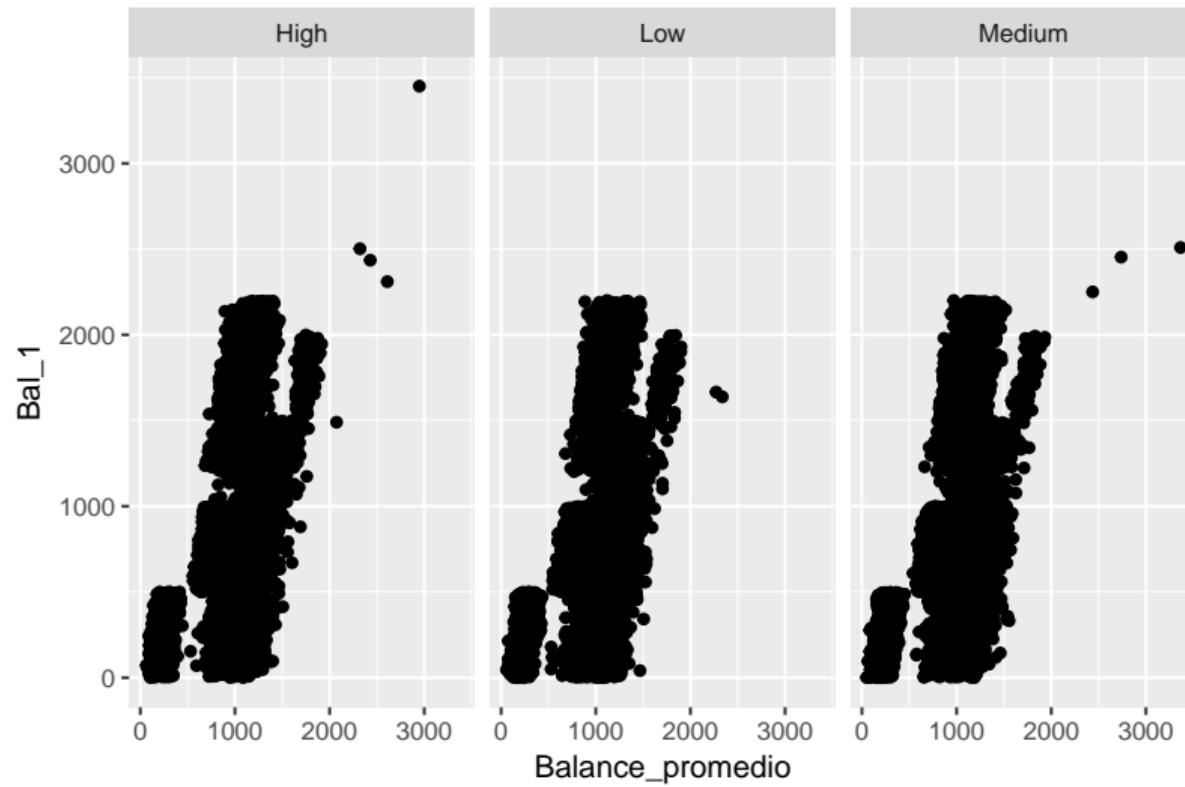
Más variables...

## Más variables...

Para que el análisis descriptivo sea más claro, se pueden incluir más variables a la hora de hacer gráficos. Para esto se puede usar la capa facet\_grid

```
creditcardmarketing_bbm %>%  
  ggplot(aes(x = Balance_promedio,y = Bal_1)) +  
  geom_point(na.rm = TRUE) +  
  facet_grid(.~Rating)
```

## Más variables...



## Ejercicio más variables

Repita el gráfico anterior, pero usando `facet_grid(Tipo_mensajeria~.)`.

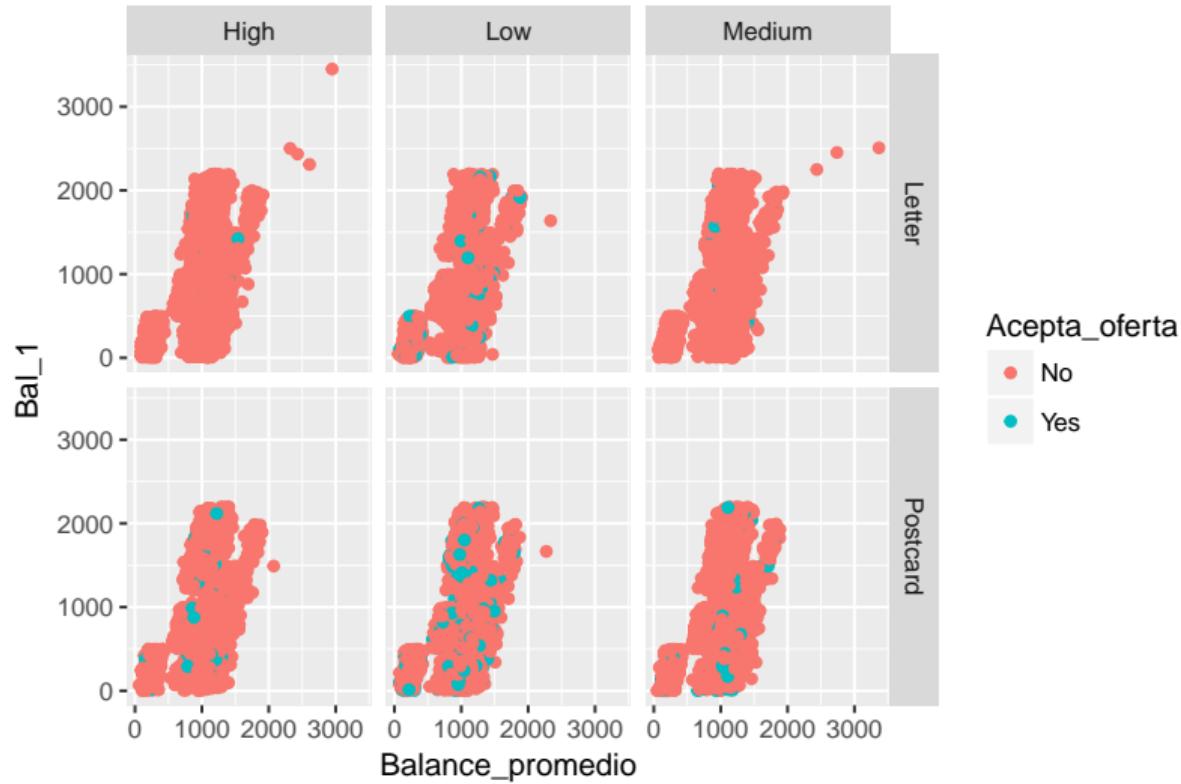
## Ejercicio más variables

Repita el gráfico anterior, pero usando `facet_grid(Tipo_mensajeria~.)`. Repita el gráfico anterior pero usando `facet_grid(Tipo_mensaeria~Rating)`

## Ejercicio más variables

Repita el gráfico anterior, pero usando `facet_grid(Tipo_mensajeria~.)`. Repita el gráfico anterior pero usando `facet_grid(Tipo_mensaeria~Rating)`, y si además colorea con `Acepta_oferta?`

# Ejercicio más variables



## Objetos

Todos los ggplots son objetos, por lo que no solo se obtiene un gráfico, sino que es fácil de conservar:

```
plot_1 <- ggplot(data = creditcardmarketing_bbm) +  
  geom_histogram(mapping = aes(Balance_promedio),  
                 bins = 30,na.rm = TRUE)
```

## Objetos

Todos los ggplots son objetos, por lo que no solo se obtiene un gráfico, sino que es fácil de conservar:

```
plot_1 <- ggplot(data = creditcardmarketing_bbm) +  
  geom_histogram(mapping = aes(Balance_promedio),  
                 bins = 30,na.rm = TRUE)
```

Si llama plot\_1 desde la consola se despliega en la pestaña de Plots.

## Objetos

Todos los ggplots son objetos, por lo que no solo se obtiene un gráfico, sino que es fácil de conservar:

```
plot_1 <- ggplot(data = creditcardmarketing_bbm) +  
  geom_histogram(mapping = aes(Balance_promedio),  
                 bins = 30,na.rm = TRUE)
```

Si llama plot\_1 desde la consola se despliega en la pestaña de Plots.

## Graficar agregando capas

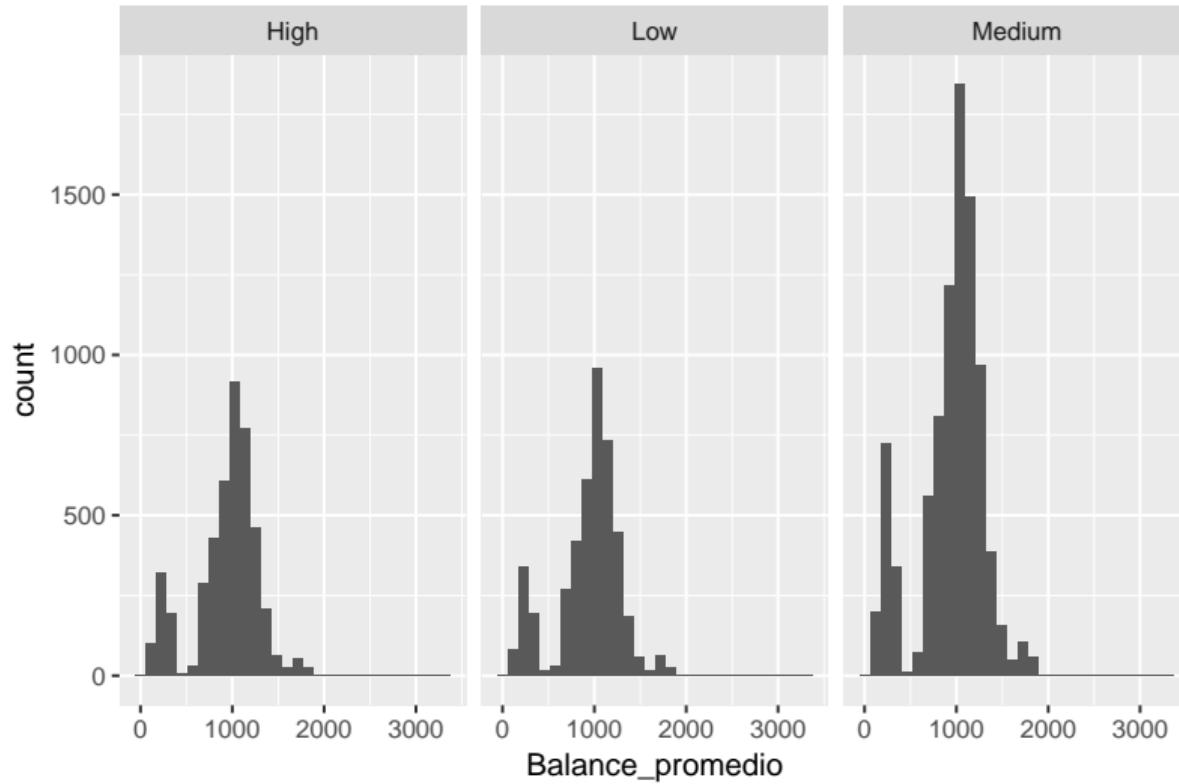
## Graficar agregando capas

Si quiere realizar modificaciones, usa la misma sintaxis de siempre, agregando capas:

```
plot_1 +  
  facet_grid(.~Nivel_ingresos)
```

Esto se presta para ir desarrollando progresivamente los gráficos, y efectivamente ir explorando las relaciones que se puede presentar

## Graficar agregando capas

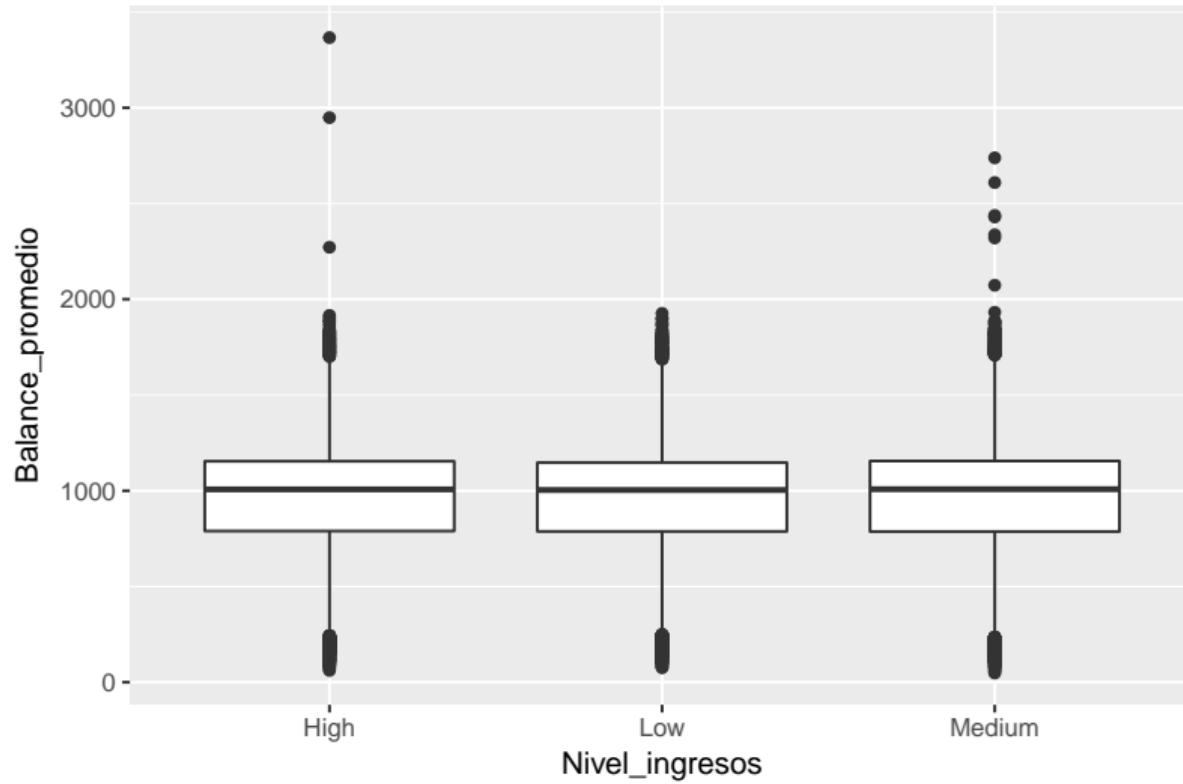


## Box-plots

Una herramienta muy útil para graficar, son los boxplots, que ayudan a entender como están distribuidas las variables y cuales son “raras”

```
ggplot(data = creditcardmarketing_bbm) +  
  geom_boxplot(mapping = aes(y = Balance_promedio,x = Nivel_ingresos),  
               na.rm = TRUE)
```

## Box-plots

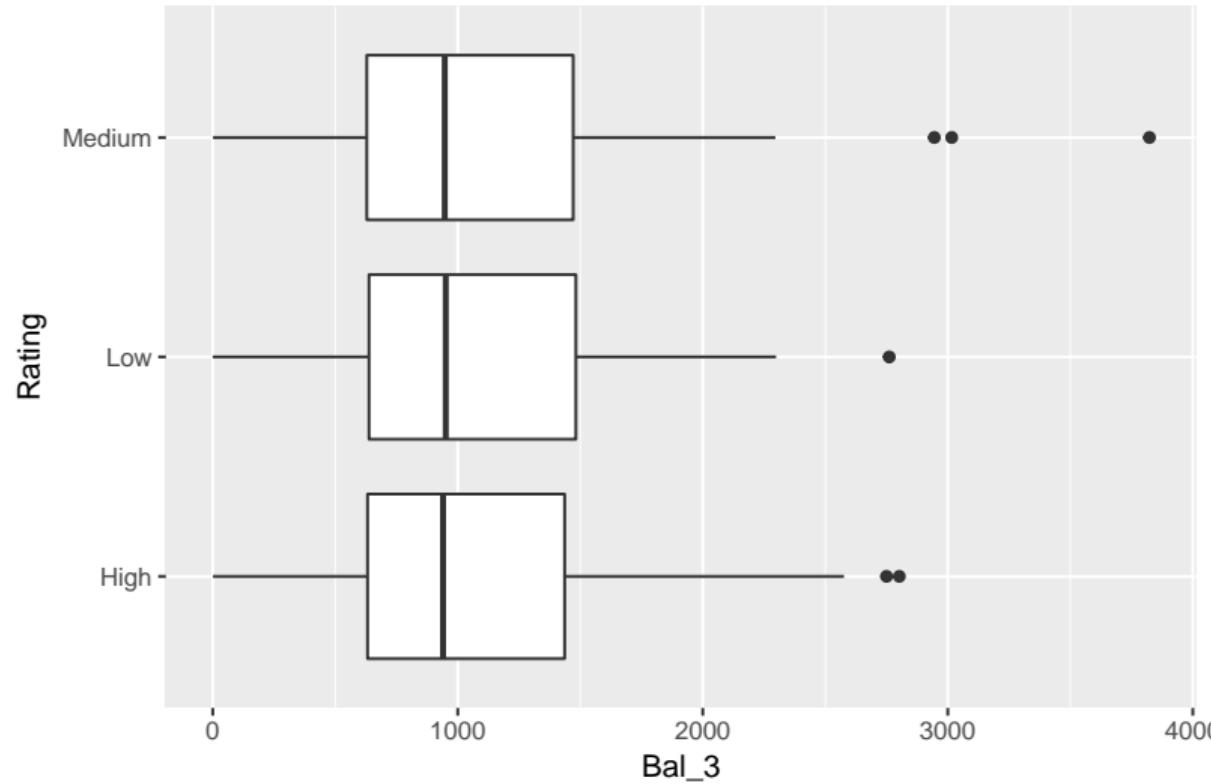


## Girar coordenadas

Algunas veces, se tienen nombres muy largos para desplegar en el eje X, por lo que se pueden girar las coordenadas, para que se entiendan mejor, y se reciba la misma información:

```
ggplot(data = creditcardmarketing_bbm) +  
  geom_boxplot(mapping = aes(y = Bal_3,x = Rating),  
               na.rm = TRUE) +  
  coord_flip()
```

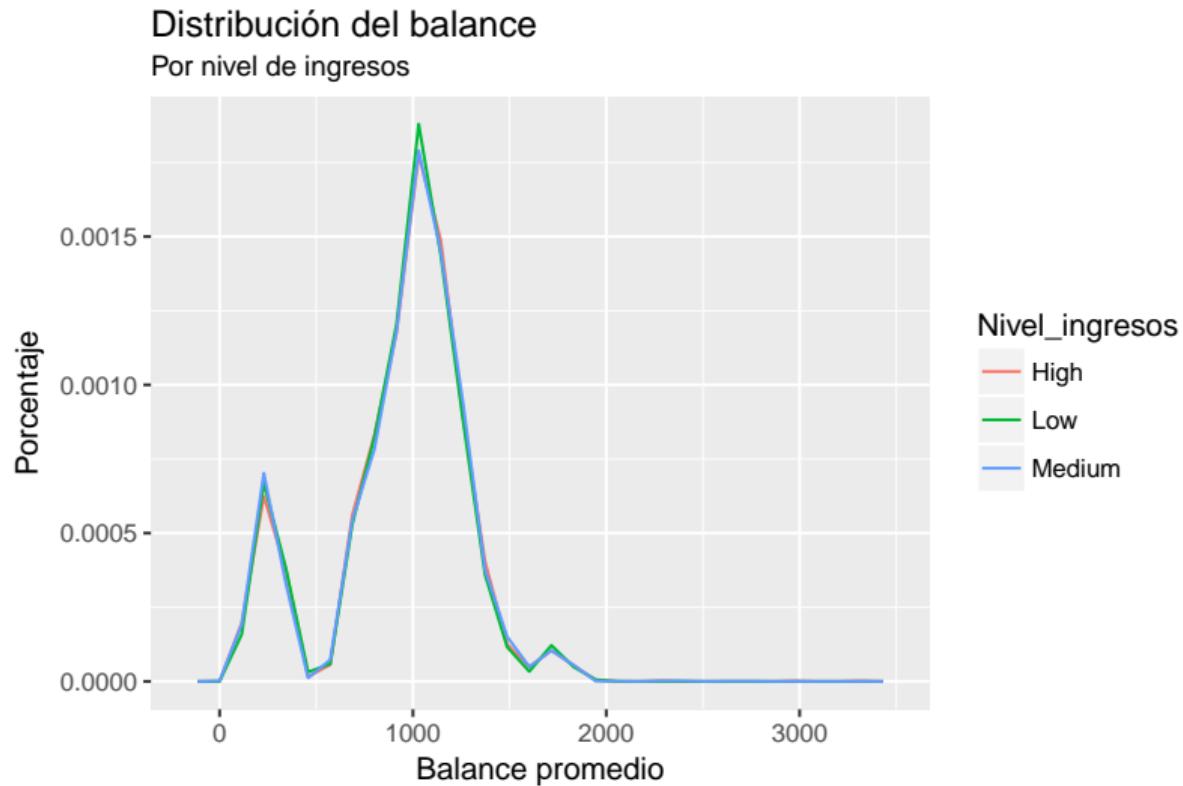
## Girar coordenadas



## Titulos y ejes

```
plot_titulo <- ggplot(data = creditcardmarketing_bbm) +  
  geom_freqpoly(mapping = aes(x = Balance_promedio,  
                               colour = Nivel_ingresos,  
                               y = ..density..),  
                 na.rm = TRUE,bins = 30) +  
  ggtitle('Distribución del balance',  
          subtitle = 'Por nivel de ingresos') +  
  xlab('Balance promedio') +  
  ylab('Porcentaje')
```

# Titulos y ejes



## Etiquetas

Bueno, pero también sería bueno poder modificar el nombre del color, y para agregar una pequeña leyenda (caption) se puede usar el parámetro `caption`:

## Etiquetas

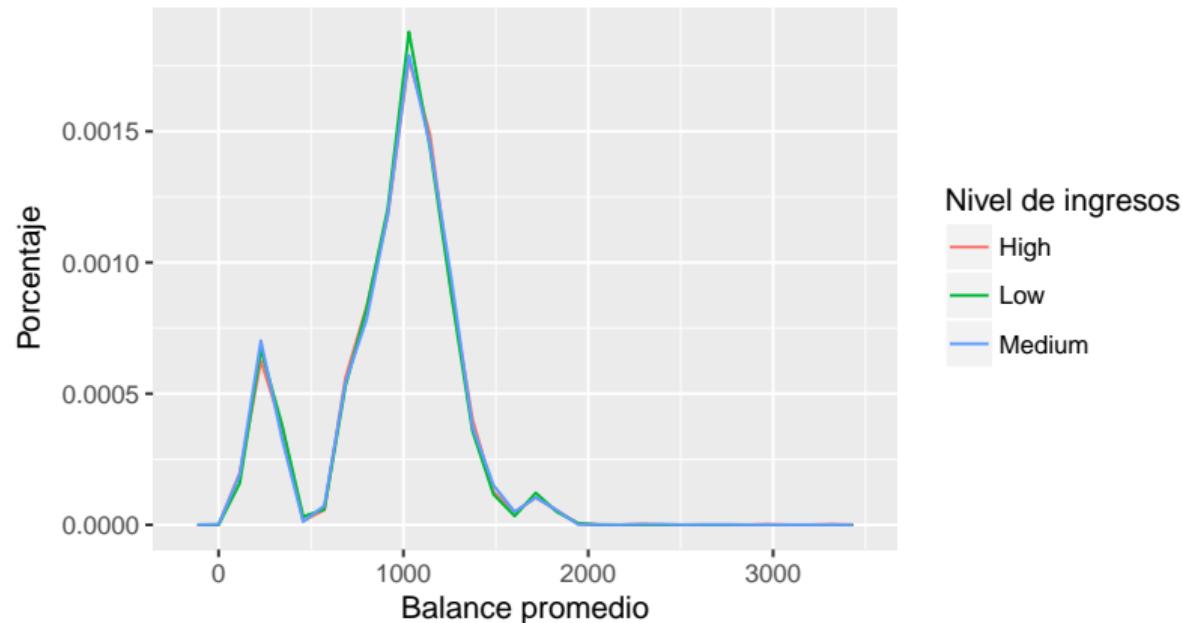
Bueno, pero también sería bueno poder modificar el nombre del color, y para agregar una pequeña leyenda (caption) se puede usar el parámetro `caption`:

```
plot_titulo +  
  labs(color = 'Nivel de ingresos',  
        caption = 'Fuente: data.world.com,\n Creacion propia')
```

# Etiquetas

## Distribución del balance

Por nivel de ingresos



Fuente: [data.world.com](http://data.world.com),  
Creacion propia

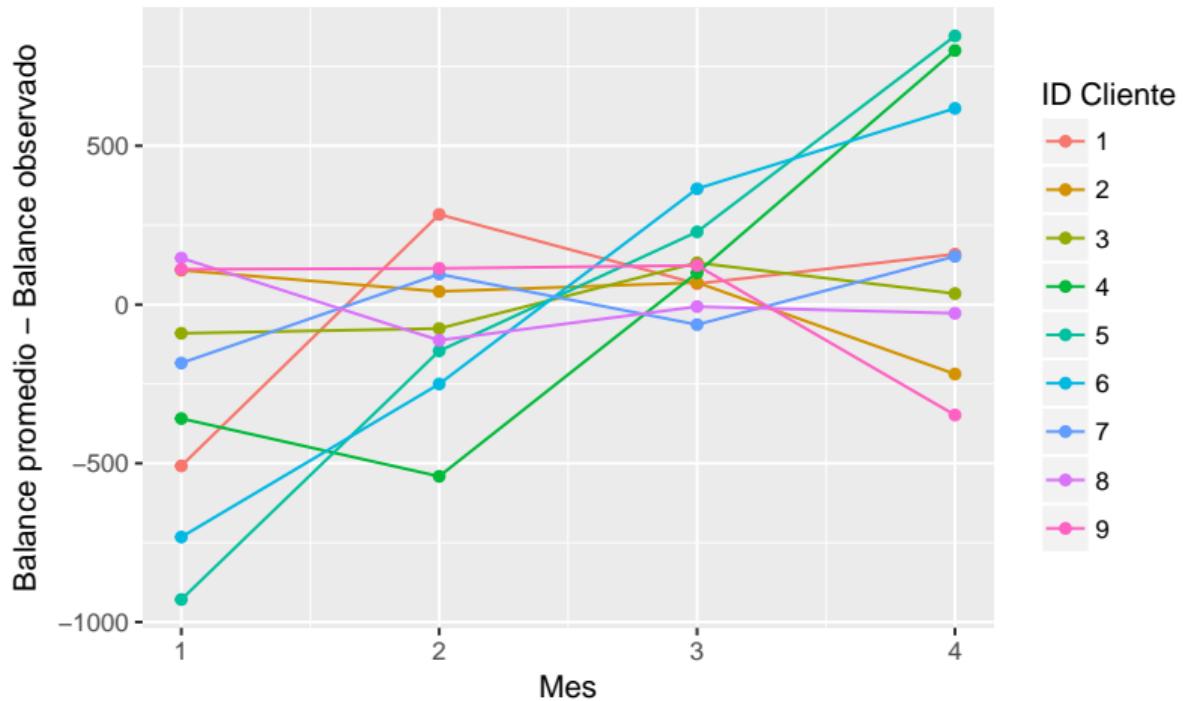
## labs

Si se quiere, se pueden incluir todos los parámetros en la capa de labs:

```
creditos_Mes %>%
  mutate(Balance_dif = Balance_promedio - Balance) %>%
  ggplot(mapping = aes(x = Mes, y = Balance_dif,
                        color = factor(Num_cliente))) +
  geom_line() +
  geom_point() +
  labs(title = 'Historia del balance',
       subtitle = 'Por persona',
       x = 'Mes',
       y = 'Balance promedio - Balance observado',
       color = 'ID Cliente')
```

## Historia del balance

Por persona



## theme

Hay varios temas que se pueden aplicar a los gráficos, para cambiar varios parámetros predefinidos. Este se agrega como una capa más. Están:

- ▶ `theme_bw()`

Y hay paquetes para incluir aún más temas...

## theme

Hay varios temas que se pueden aplicar a los gráficos, para cambiar varios parámetros predefinidos. Este se agrega como una capa más. Están:

- ▶ `theme_bw()`
- ▶ `theme_classic()`

Y hay paquetes para incluir aún más temas...

## theme

Hay varios temas que se pueden aplicar a los gráficos, para cambiar varios parámetros predefinidos. Este se agrega como una capa más. Están:

- ▶ `theme_bw()`
- ▶ `theme_classic()`
- ▶ `theme_dark()`

Y hay paquetes para incluir aún más temas...

## theme

Hay varios temas que se pueden aplicar a los gráficos, para cambiar varios parámetros predefinidos. Este se agrega como una capa más. Están:

- ▶ `theme_bw()`
- ▶ `theme_classic()`
- ▶ `theme_dark()`
- ▶ `theme_get()`

Y hay paquetes para incluir aún más temas...

## theme

Hay varios temas que se pueden aplicar a los gráficos, para cambiar varios parámetros predefinidos. Este se agrega como una capa más. Están:

- ▶ `theme_bw()`
- ▶ `theme_classic()`
- ▶ `theme_dark()`
- ▶ `theme_get()`
- ▶ `theme_gray()`

Y hay paquetes para incluir aún más temas...

## theme

Hay varios temas que se pueden aplicar a los gráficos, para cambiar varios parámetros predefinidos. Este se agrega como una capa más. Están:

- ▶ `theme_bw()`
- ▶ `theme_classic()`
- ▶ `theme_dark()`
- ▶ `theme_get()`
- ▶ `theme_gray()`
- ▶ `theme_light()`

Y hay paquetes para incluir aún más temas...

## theme

Hay varios temas que se pueden aplicar a los gráficos, para cambiar varios parámetros predefinidos. Este se agrega como una capa más. Están:

- ▶ `theme_bw()`
- ▶ `theme_classic()`
- ▶ `theme_dark()`
- ▶ `theme_get()`
- ▶ `theme_gray()`
- ▶ `theme_light()`
- ▶ `theme_linedraw()`

Y hay paquetes para incluir aún más temas...

## theme

Hay varios temas que se pueden aplicar a los gráficos, para cambiar varios parámetros predefinidos. Este se agrega como una capa más. Están:

- ▶ `theme_bw()`
- ▶ `theme_classic()`
- ▶ `theme_dark()`
- ▶ `theme_get()`
- ▶ `theme_gray()`
- ▶ `theme_light()`
- ▶ `theme_linedraw()`
- ▶ `theme_minimal()`

Y hay paquetes para incluir aún más temas...

# Ejercicios

## Ejercicios

En los gráficos que usan `geom_line`, explore usar los parámetros `linetype` y `size`.

## Ejercicios

En los gráficos que usan `geom_line`, explore usar los parámetros `linetype` y `size`.

En los gráficos de `geom_bar`, explore usando el parámetro `fill`.

## Ejercicios

En los gráficos que usan `geom_line`, explore usar los parámetros `linetype` y `size`.

En los gráficos de `geom_bar`, explore usando el parámetro `fill`.

Juegue con los temas y los gráficos que ha hecho. Use al menos una vez cada tema.

## Ejercicios

En los gráficos que usan `geom_line`, explore usar los parámetros `linetype` y `size`.

En los gráficos de `geom_bar`, explore usando el parámetro `fill`.

Juegue con los temas y los gráficos que ha hecho. Use al menos una vez cada tema.

Lea sobre `ggsave`

Próximamente:

optim