Taller de Visualización en R: ggplot 2 y otros

Andrés Blanco y Santiago Costas

Instalando y Cargando las librerías

El primer paso siempre que comencemos una sesión de R es cargar las librerías que utilizaremos. En caso que no las tengan instaladas, primero hay que hacerlo.

```
## Correr solo si no tienen instaladas las librerías
install.packages(c('tidyverse', 'sciplot', 'patchwork', 'viridis', 'RColorBrewer'))
## en caso de que quieran correr solo alguno de ellos, el codigo es igual
## solo con el paquete de interes por ejemplo, install.packages('tidyverse')
```

```
library(tidyverse); library(sciplot); library(patchwork); library(viridis); library(RColorBrewer)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
## Loading required package: viridisLite
```

Cargando la base de datos

En este caso usamos una función del paquete tidyverse `read_csv2` la cual le agrega otros atributos a la tabla. Utilizando el comando `head` podemos ver las primaras filas y las columnas (variables) con su tipo de objeto (que es muy importante saber las diferencias pero que hoy no lo analizaremos en este taller). La funciones de R base más usadas son `read.csv` o `read.csv2`

```
data <- read_csv2("producción soja.csv")
```

```
## i Using "','" as decimal and "'.'" as grouping mark. Use `read_delim()` for more control.
```

```
## Rows: 112 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ";"
## chr (1): Area
## dbl (4): Year, sup, Yield, Production
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

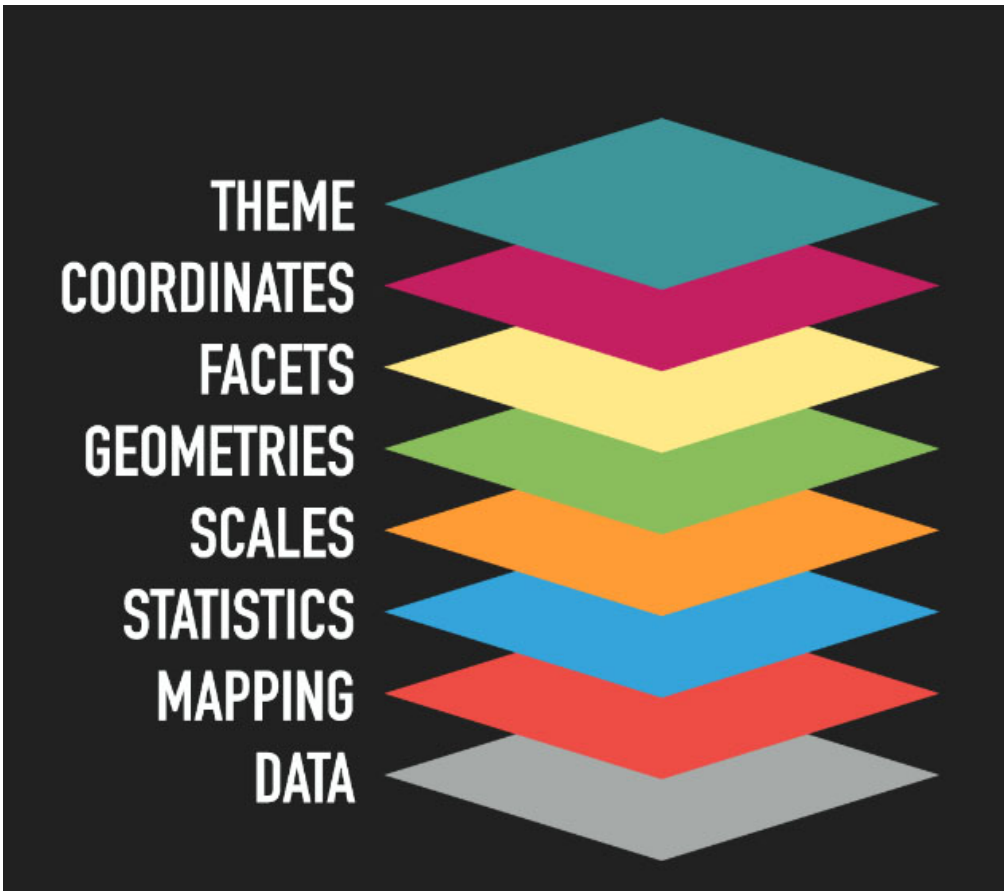
Las unidades están en metros cuadrados, si las queremos en hectáreas:

```
data$Yield <- data$Yield/10000

head(data)
```

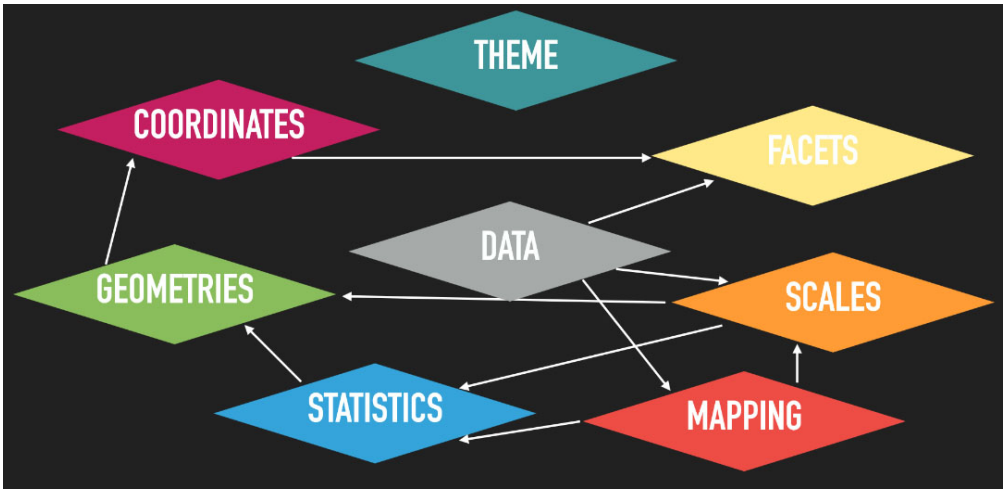
```
## # A tibble: 6 x 5
##   Area       Year   sup Yield Production
##   <chr>     <dbl> <dbl> <dbl>      <dbl>
## 1 Argentina  1961   980 0.976        957
## 2 Argentina  1962  9649 1.16       11220
## 3 Argentina  1963 19302 0.980      18920
## 4 Argentina  1964 12220 1.15       14000
## 5 Argentina  1965 16422 1.04       17000
## 6 Argentina  1966 15689 1.16       18200
```

1. Primeros Gráficos

Con los datos cargados hacemos los primeros gráficos con estética estándar. Es importante entender que ggplot2 funciona siguiendo la teoría de *grammar of graphics*, cuyo objetivo es **generalizar** la forma de crearlos. Los gráficos se componen de **'capas'** que se van **solapando**.



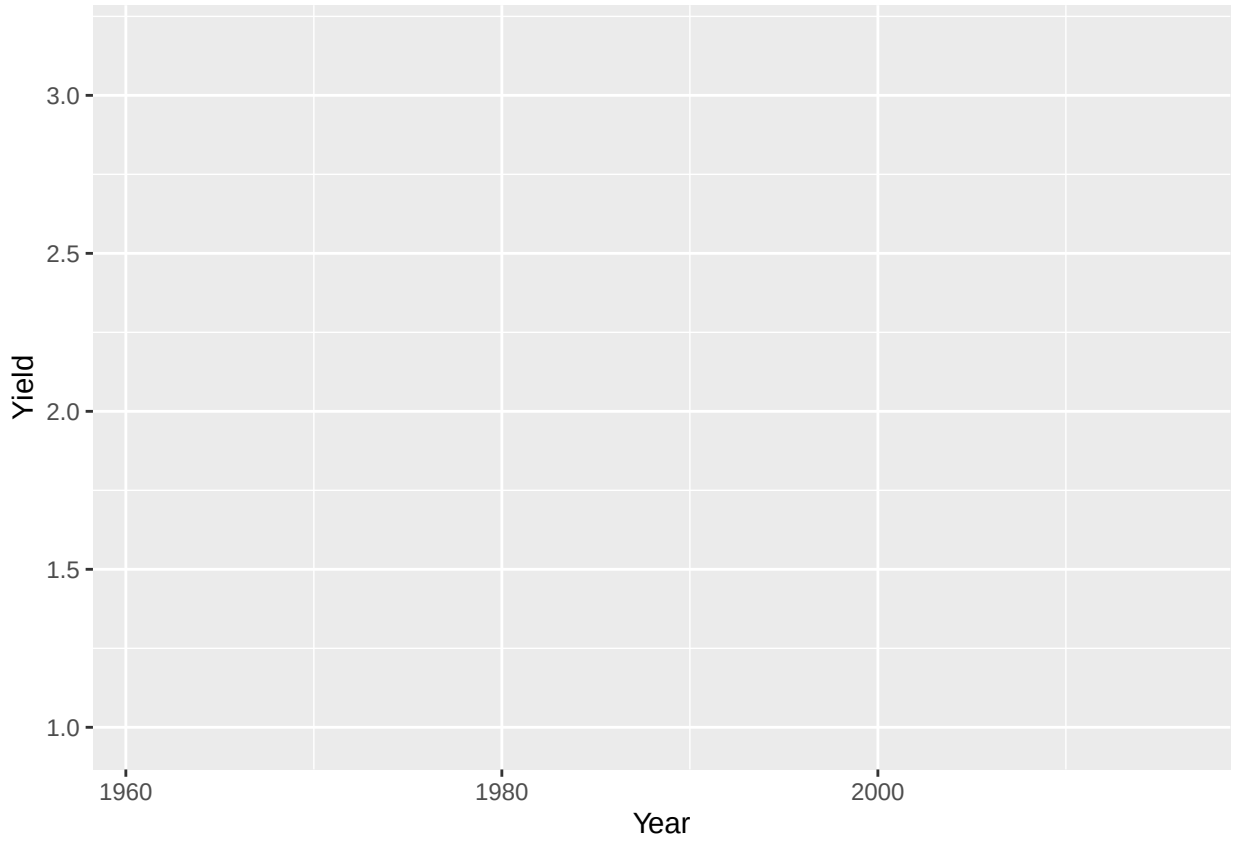
Sacada del taller de Thomas Lin Pedersen ggplot2 workshop - Pedersen



1.1. Datos y Mapping

En este punto le indicamos qué datos vamos a usar y el mapping, que conecta tus datos con su significado. El software no sabe por defecto la naturaleza de tus datos, por lo que es imprescindible indicarle cuál es la x cuál es la y o demás variables/factores que pueden estar involucrados. Es decir conecta los datos y las variables con las propiedades gráficas. Si solo especificamos esto, el resultado es la nada misma.

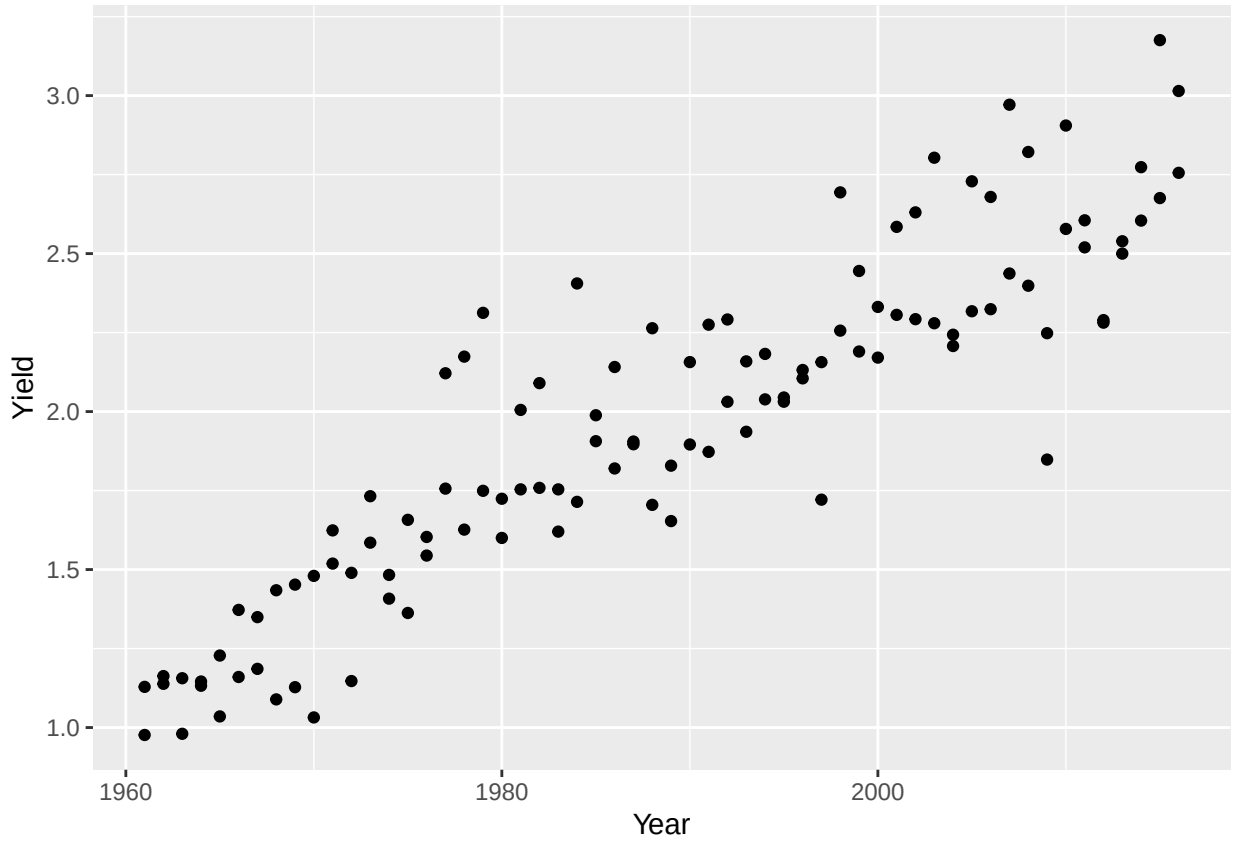
```
G1 <- ggplot(data = data,
       mapping = aes(x= Year, y = Yield))  ; G1
```



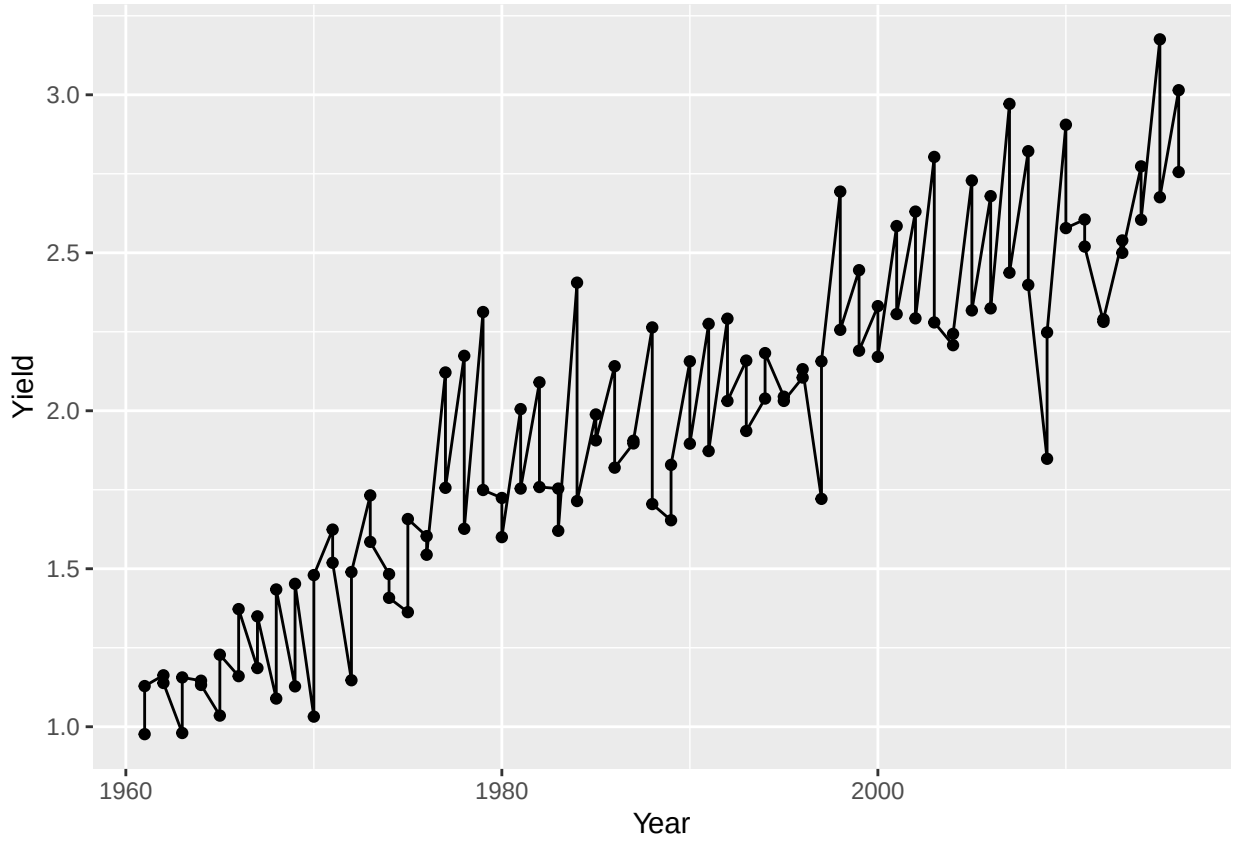
1.2. Geom

Los `geom` toman los valores escalados que vienen del mapping y los interpretan de una forma particular. Los mismos valores pueden pueden tener una gran lista diferente de posibles representaciones geométricas (puntos, líneas, barras, cajas) y los `geom` son como especificas estas formas. Acá es donde se destaca más el funcionamiento de 'capas' ya que estas son las verdaderas capas gráficas junto con otras que veremos mas adelante.

```
G1 + geom_point()
```

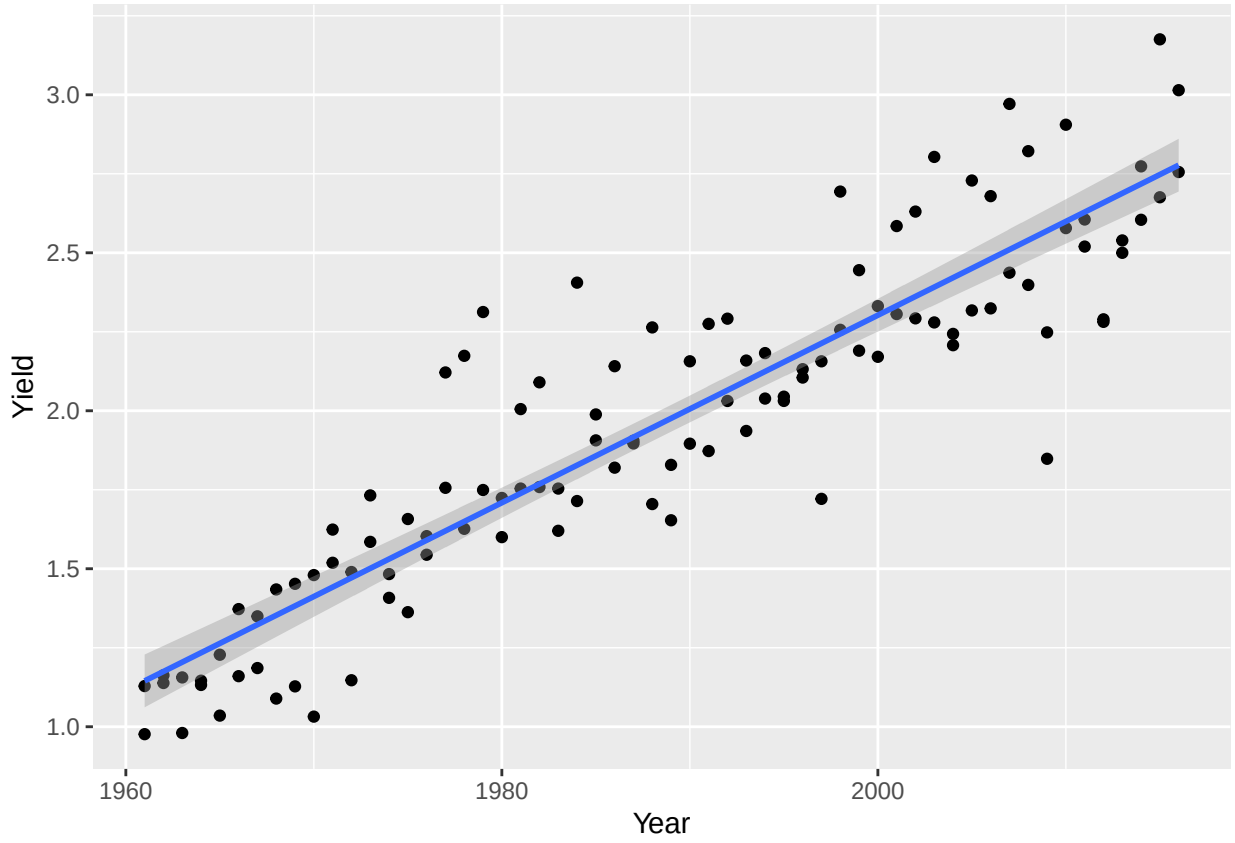



```
G1 + geom_point() + geom_line()
```



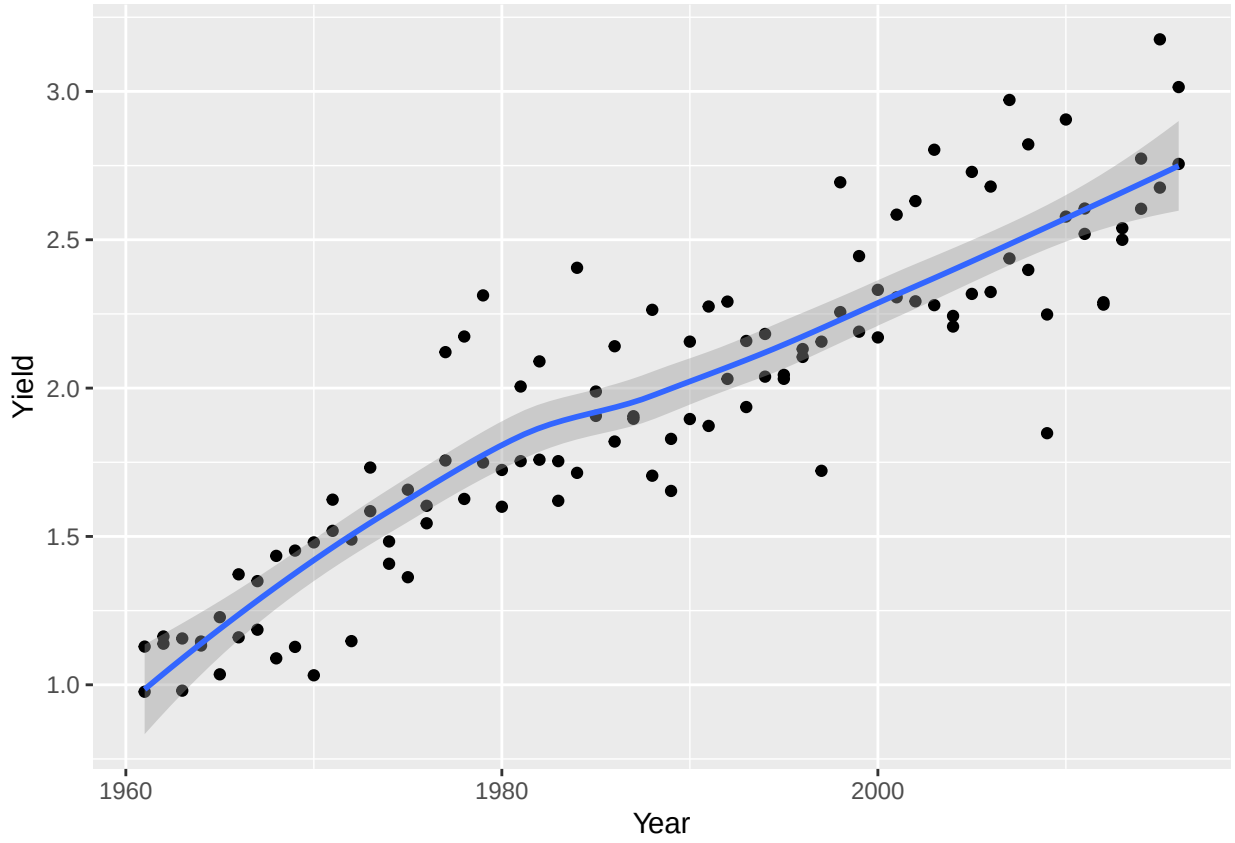

```
G1 + geom_point() + geom_smooth(method = 'lm')
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
G1 + geom_point() + geom_smooth()
```

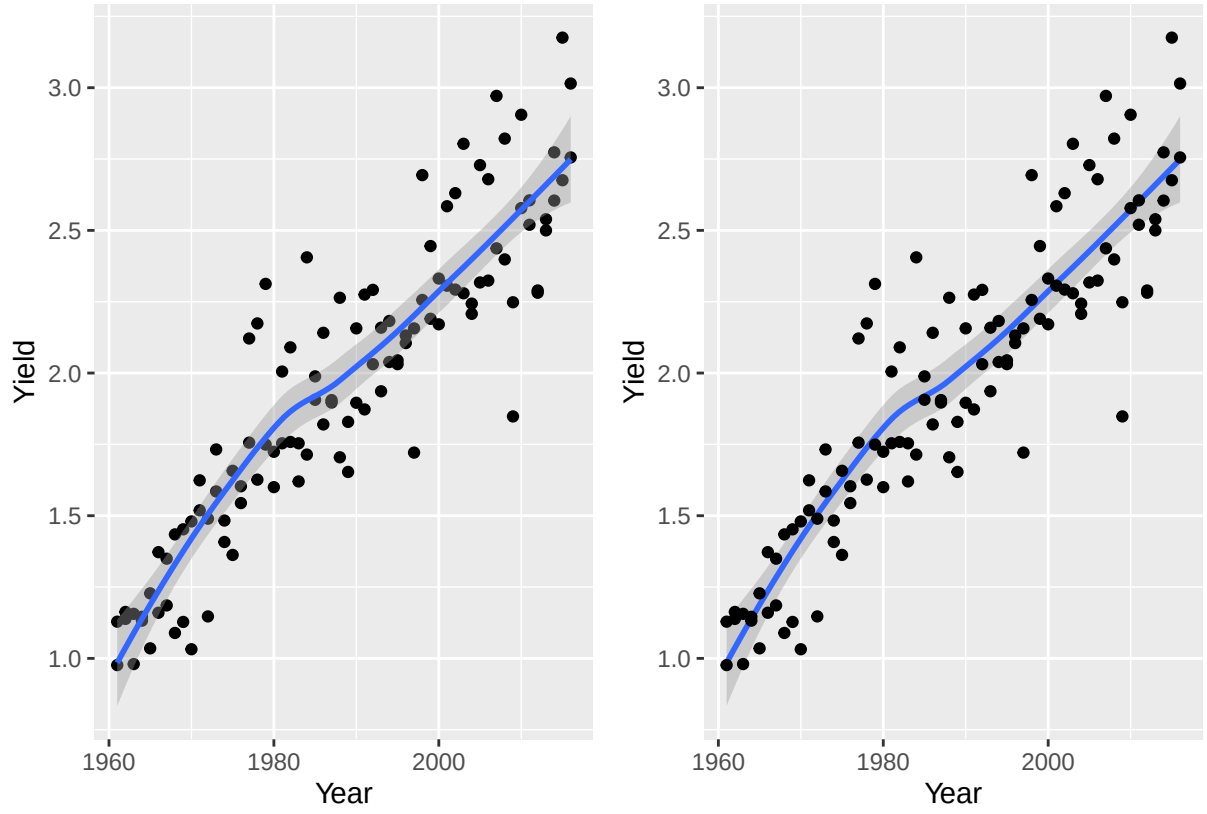
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



El orden importa

```
G1 + geom_point() + geom_smooth() + G1 + geom_smooth() + geom_point()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

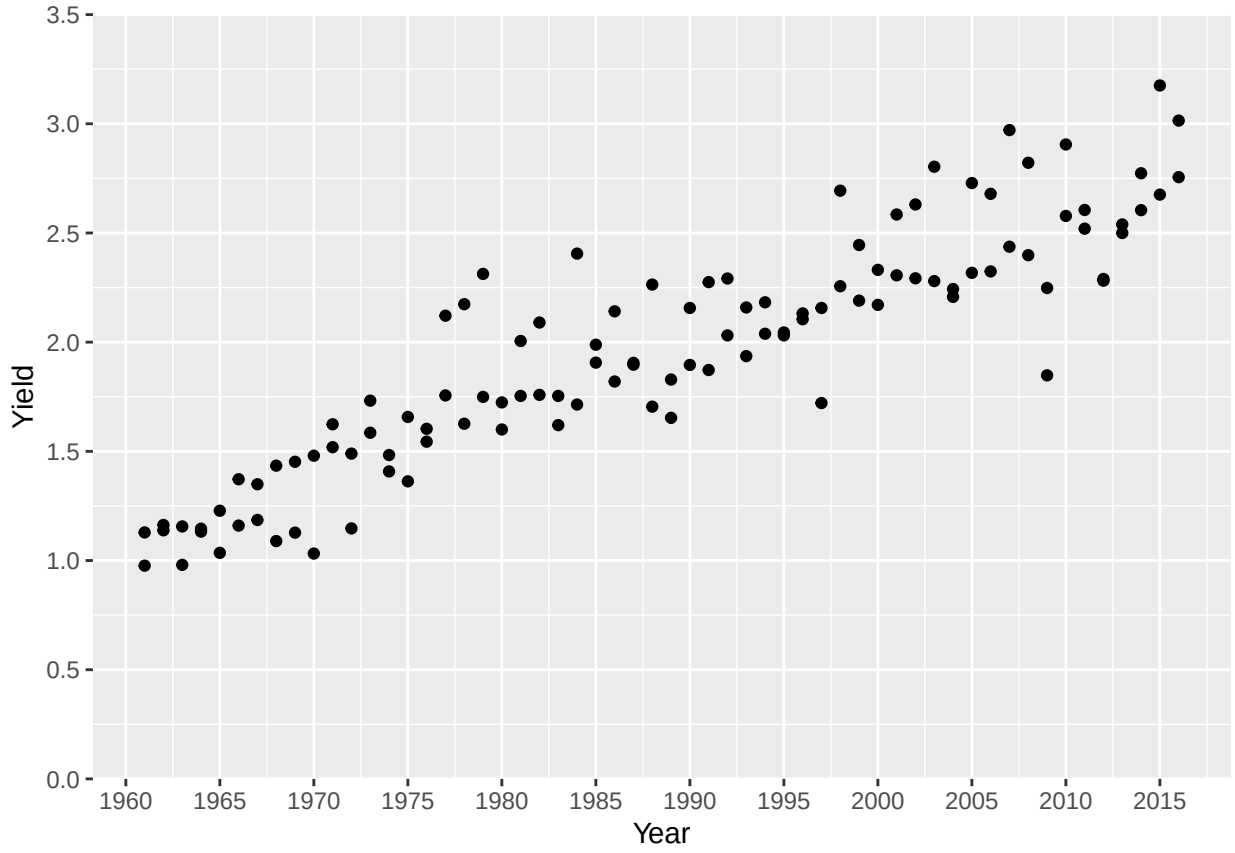


1.3. Scales

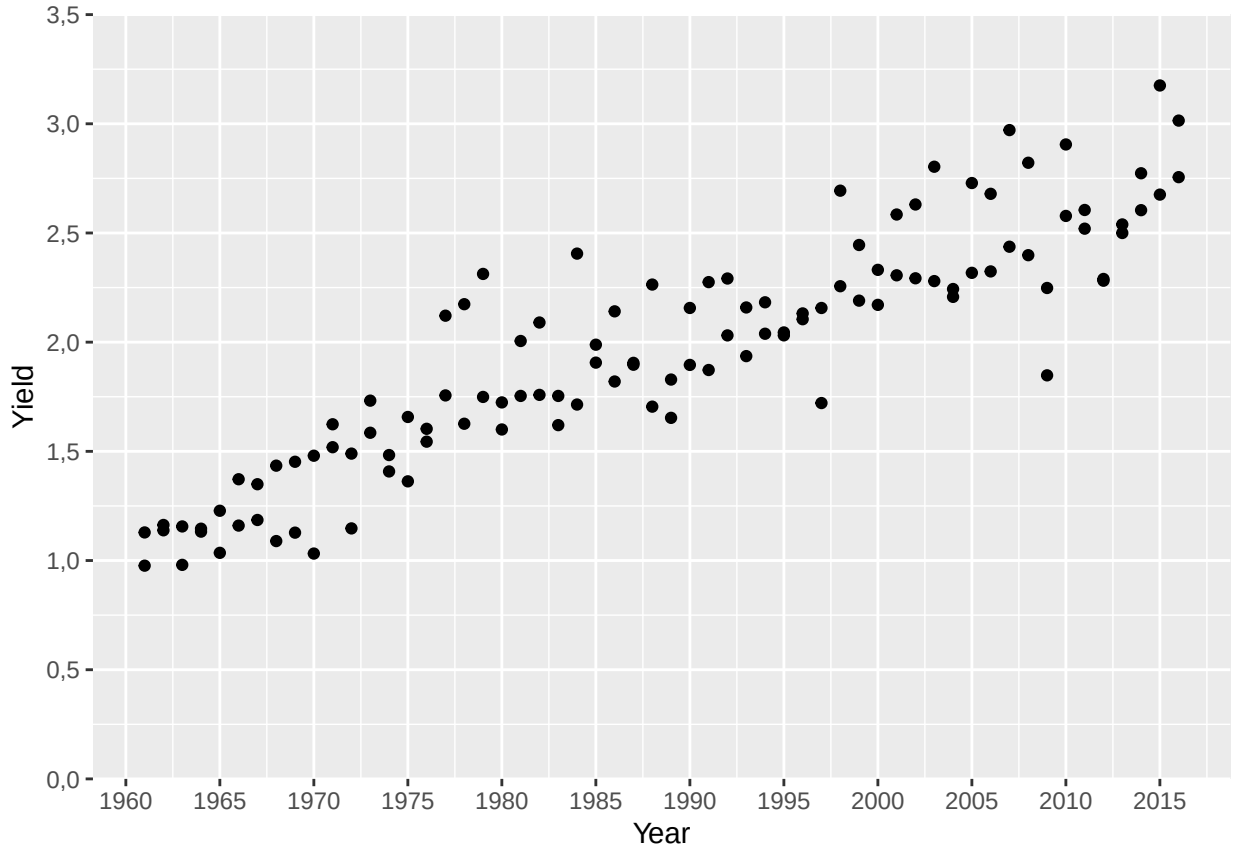
Transcribe los rangos de las variables a los rangos de la propiedades gráficas. Por ejemplo, si queremos clasificar por un factor y que esto sea representado por el color, la *escala* le dice que el valor "argentina" por ejemplo va a ser "azul" y mundo será "rojo". También aplica en variables continuas , por ejemplo en un scatterplot conecta los números con la posición. En los pasos anteriores se definieron escalas por defecto al usar mapping y geom. Ahora vamos a especificarla, esto modificará la escala gráfica a mostrar en los ejes.

Escalas continuas

```
G1 <- G1 + geom_point() ## Simplemente guardamos el objeto para lo que viene

G1 +
  scale_y_continuous(expand = c(0,0),
                     limits = c(0, 3.5),
                     breaks = seq(0, 3.5, by = .5))+
  scale_x_continuous(breaks = seq(1960,2016, by=5))
```




```
G1 <- G1 +
  scale_y_continuous(expand = c(0,0),
                     limits = c(0, 3.5),
                     breaks = seq(0, 3.5, by = .5),
                     labels=function(x) format(x, decimal.mark =",", scientific = FALSE))+
  scale_x_continuous(breaks = seq(1960,2016, by=5)) ; G1
```

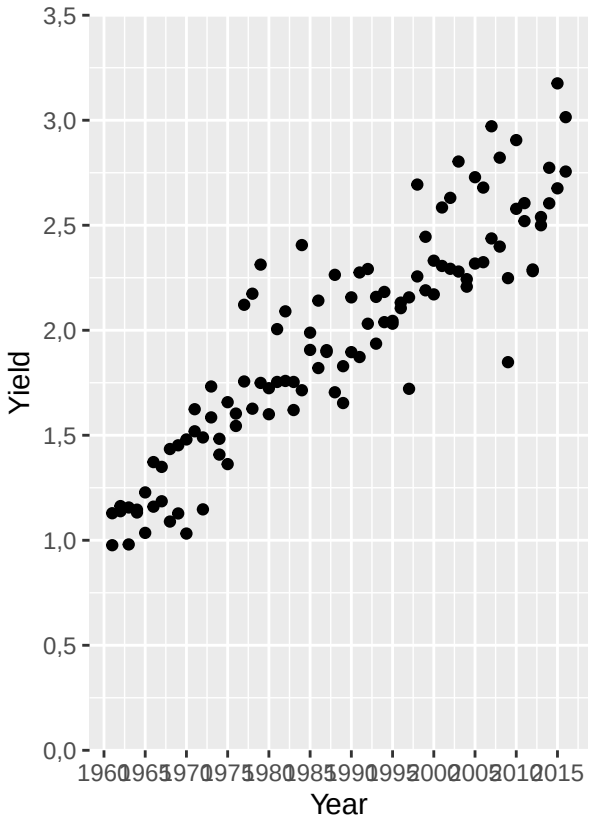


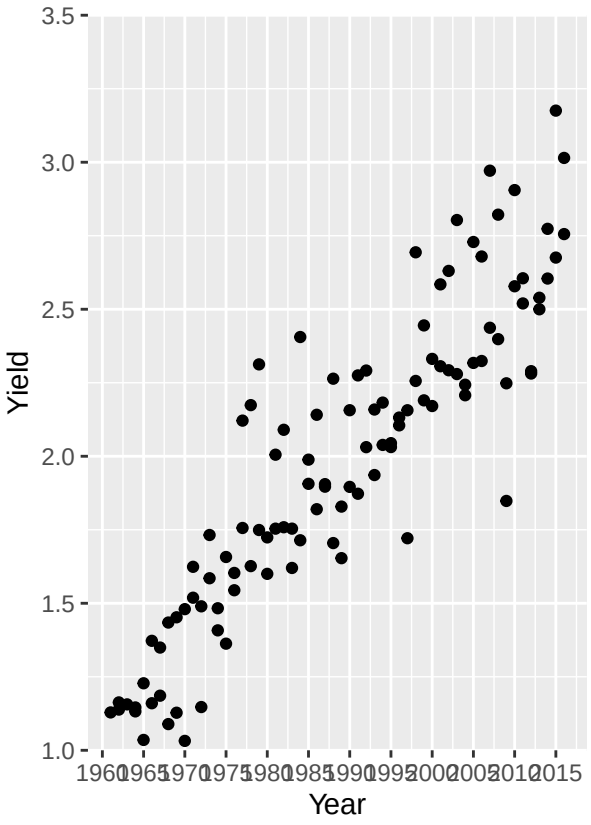

```
G1+ G1 +
  scale_y_continuous(expand = c(0,0),
                     limits = c(1, 3.5),
                     breaks = seq(0, 3.5, by = .5))+
  scale_x_continuous(breaks = seq(1960,2016, by=5))
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.
```

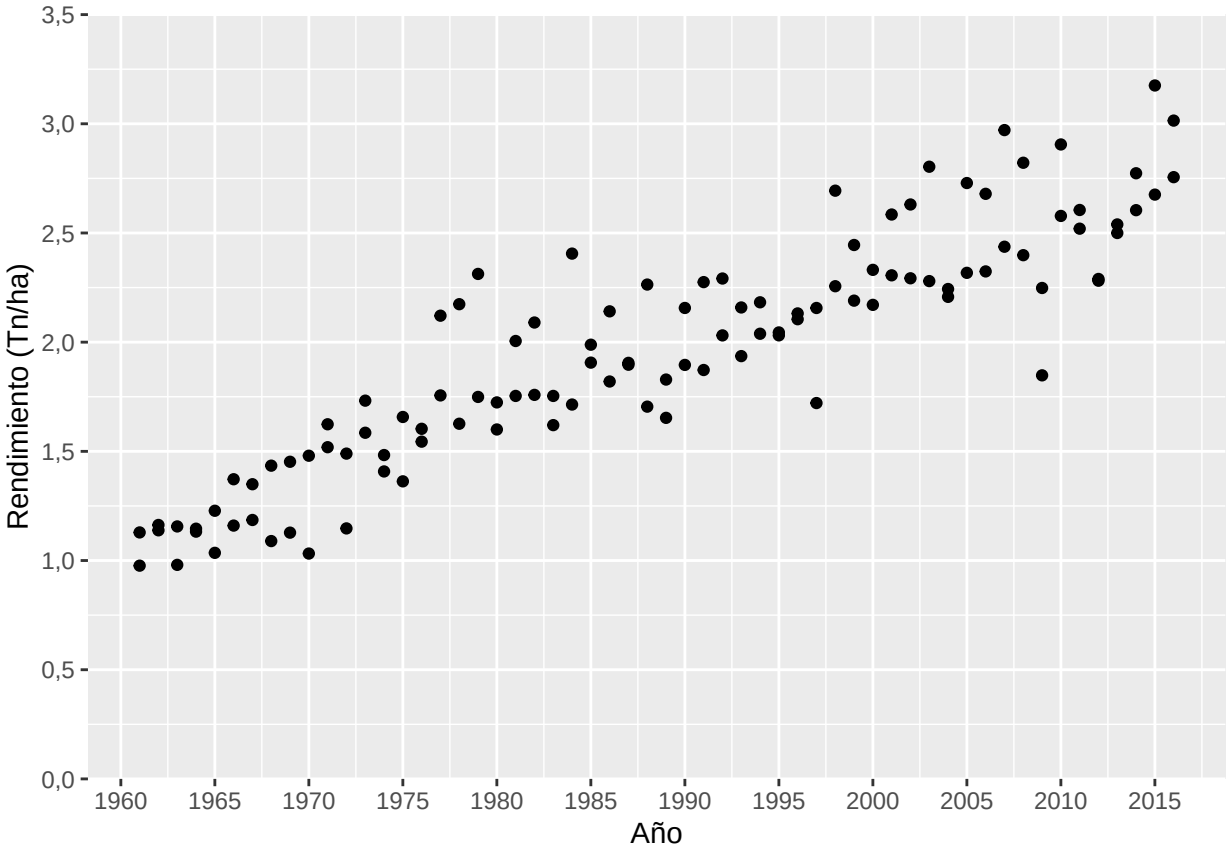
```
## Scale for 'x' is already present. Adding another scale for 'x', which will
## replace the existing scale.
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```





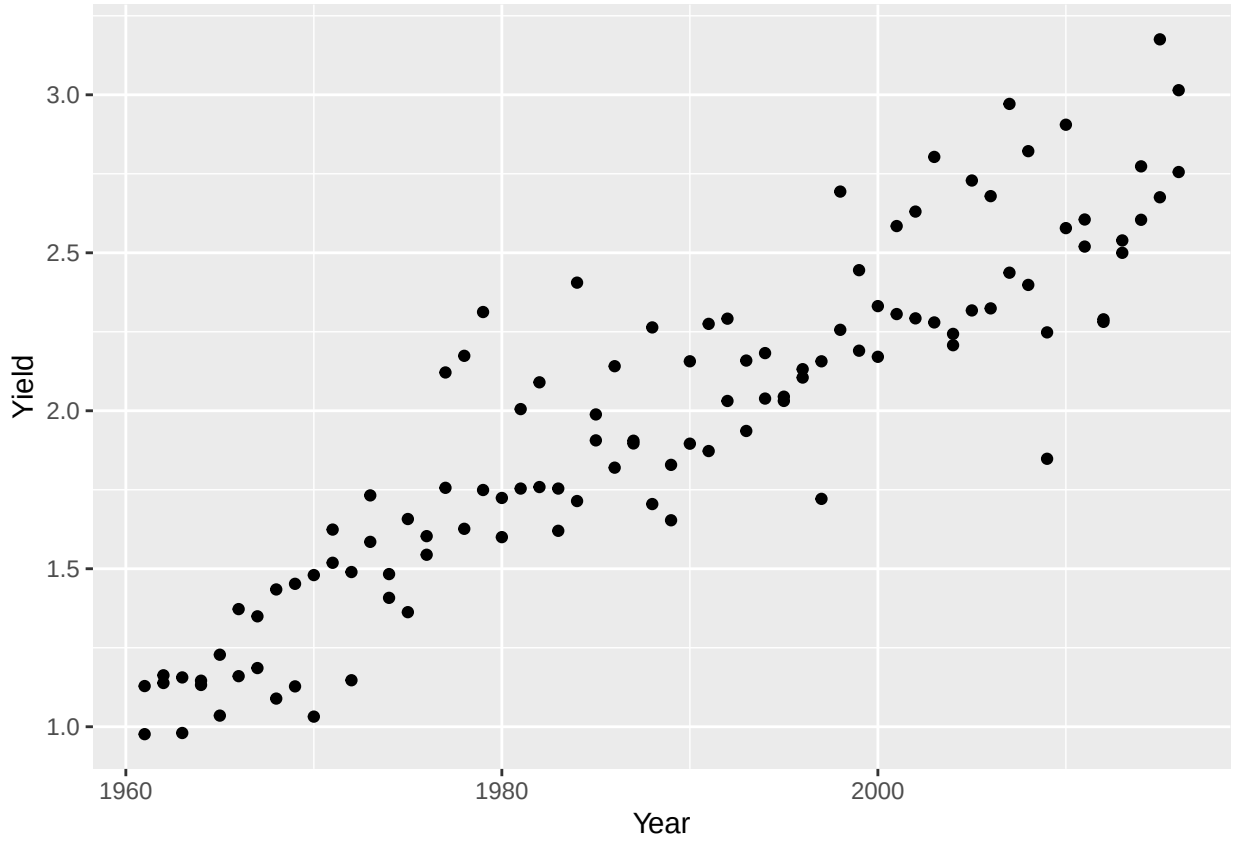

```
# labs establece los títulos de ejes (y leyenda)
G1 <- G1+
  labs(x= "Año",
       y = "Rendimiento (Tn/ha)") ; G1
```



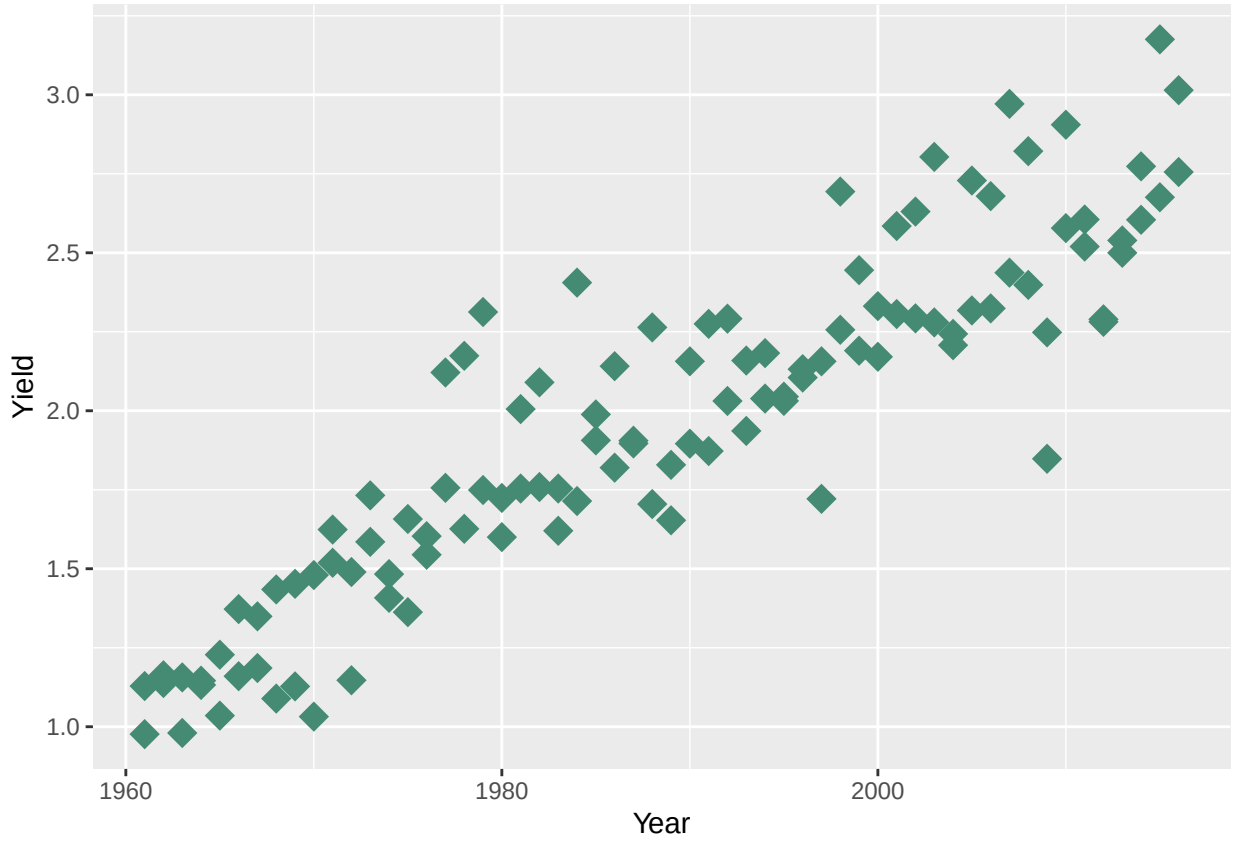
Modificando las escalas de color, forma y tamaño los geoms

Las variables de clasificación siguen generando colores, formas y tamaños por defecto.

```
ggplot(data = data,
       aes(x= Year, y = Yield)) +
  geom_point()
```




```
ggplot(data = data,
       aes(x= Year, y = Yield)) +
  geom_point(col="aquamarine4", shape=18, size= 5)
```



2. Gráficos con 2 variables explicativas

Ahora vamos a empezar a jugar con las diferentes posibilidades utilizando una variable de clasificación y modificando propiedades gráficas que antes dejamos por defecto.

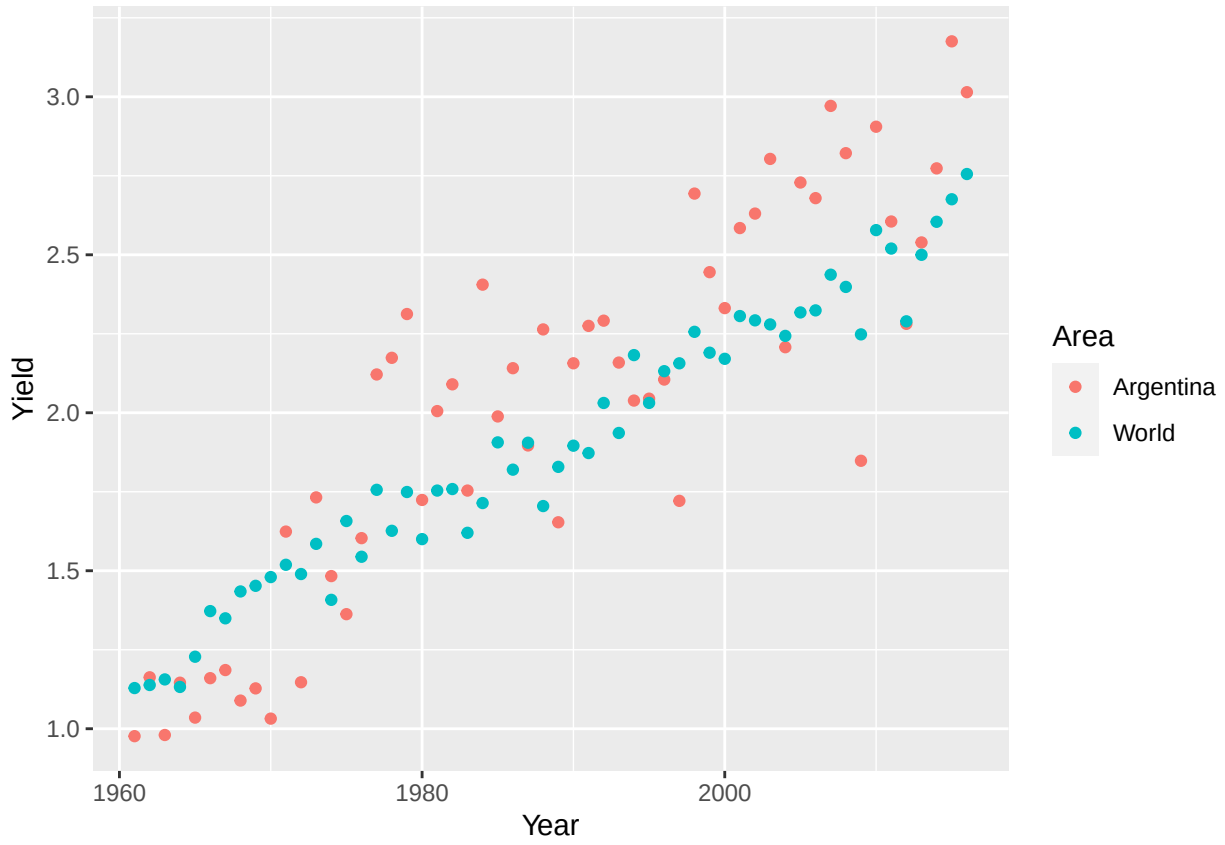
2.1. Clasificando por factores

En esta sección incluiremos una variable más en el *mapping* la cual va funcionar como factor de clasificación. Todo lo que incluyamos como clasificación en el *mapping* va a generar una leyenda.

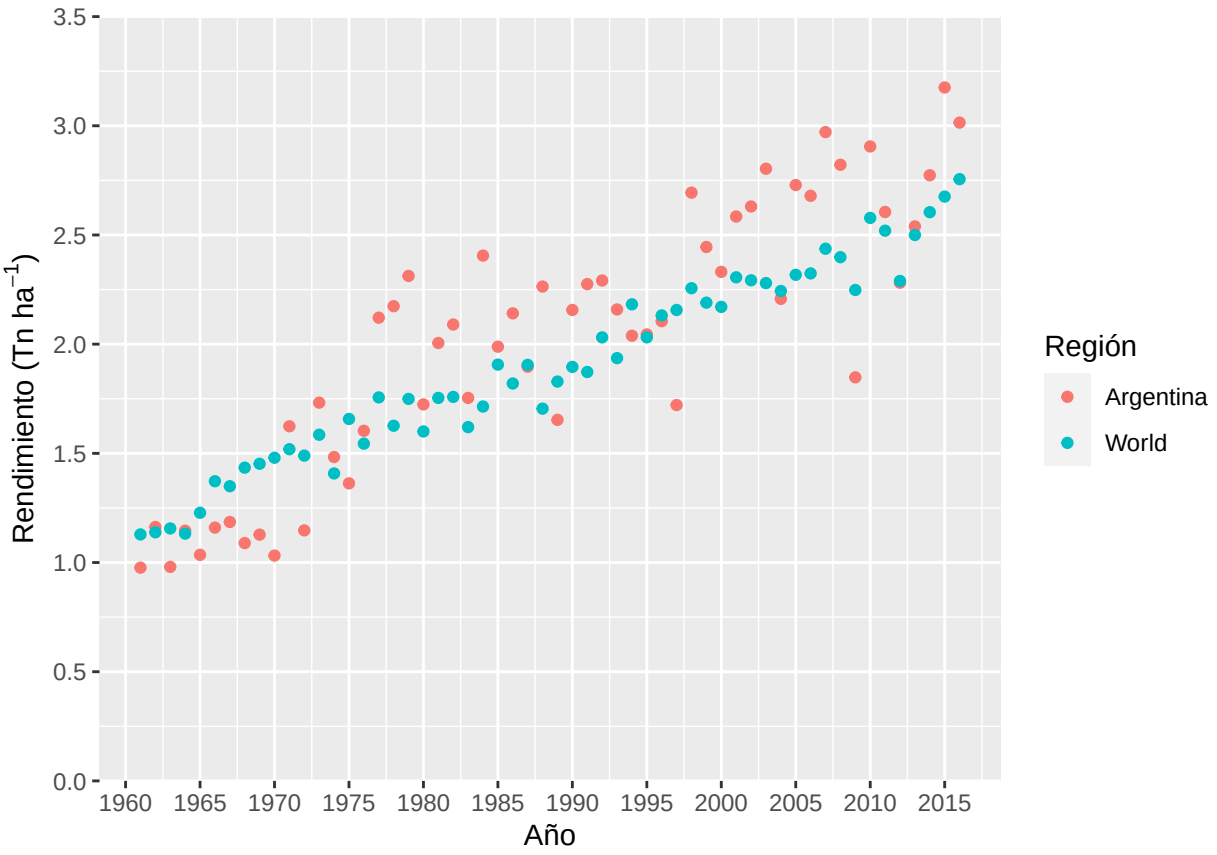
Color

Especificaremos que el *color* va a estar determinado de acuerdo con el el *área*, aunque todavía la escala de estos colores será por defecto.

```
G2 <- ggplot(data = data,
      mapping = aes(x= Year, y = Yield,
            color = Area)) +
  geom_point() ; G2
```



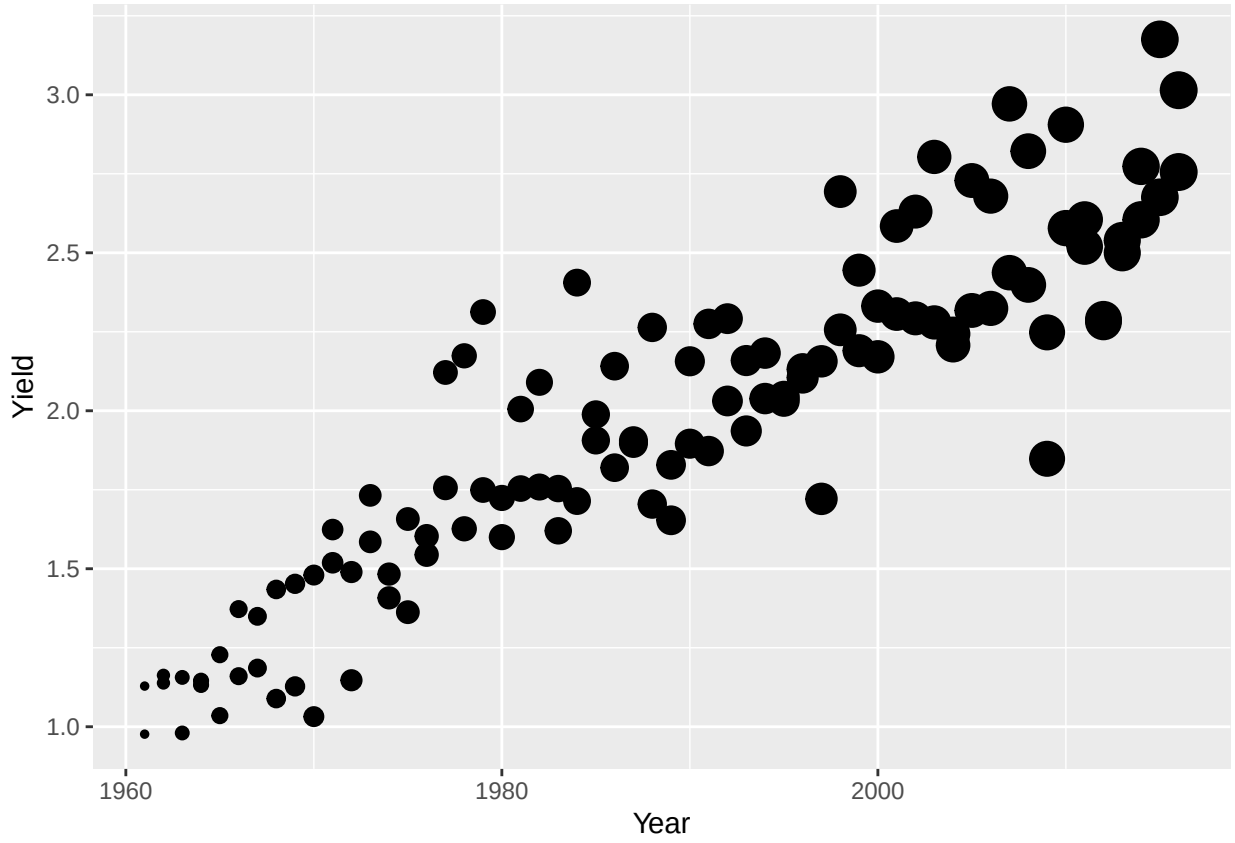

```
G2 <- G2+
  scale_y_continuous(expand = c(0,0), limits = c(0, 3.5), breaks = seq(0, 5, by = .5))+
  scale_x_continuous(breaks = seq(1960,2016, by=5)) +
  labs(x= "Año",
       y = expression(paste("Rendimiento (Tn ",ha^-1, ")")),
       col = "Región") ; G2
```



Tamaño

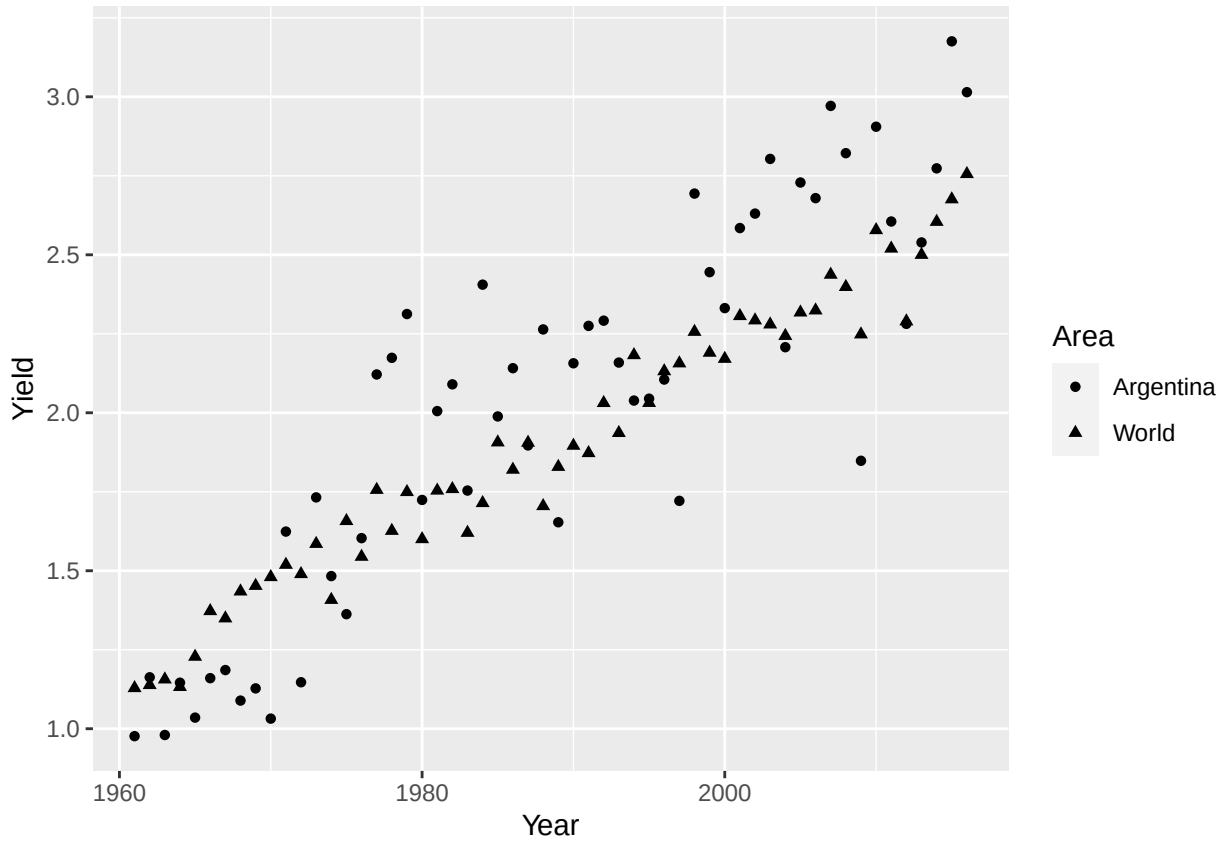
Ahora haremos lo mismo, pero en vez del color, modificaremos el tamaño de acuerdo con el año. Como nos interesa la tendencia y no el valor exacto, eliminamos la leyenda.

```
ggplot(data = data,
       mapping = aes(x= Year, y = Yield,
            size = Year)) +
  geom_point(show.legend = F) ## se puede eliminar la leyenda utilizando este codigo
```



Forma

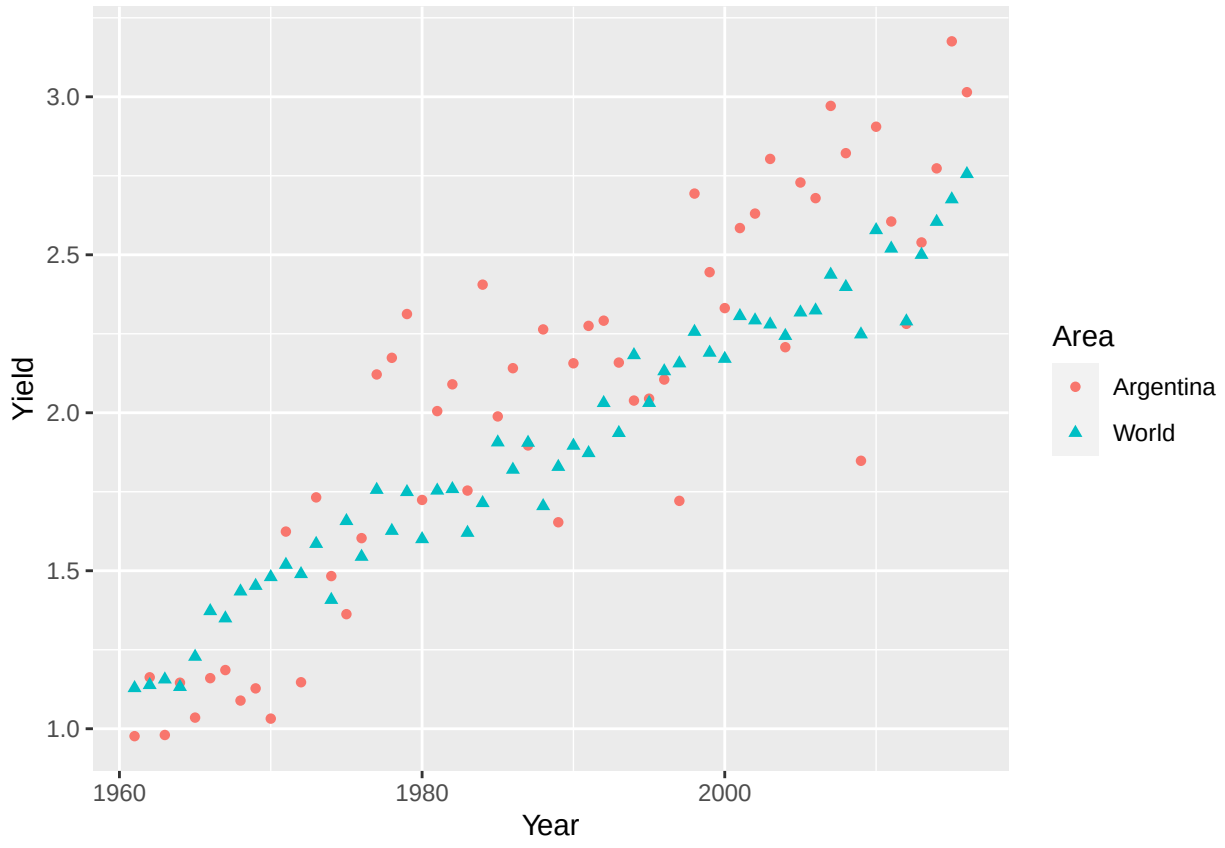
```
ggplot(data = data,
       mapping = aes(x= Year, y = Yield,
            shape = Area)) +
  geom_point()
```



Combinando

Se pueden combinar más de un parámetro de clasificación. Si el factor es el mismo, la leyenda se unificará.

```
ggplot(data = data,
       mapping = aes(x= Year, y = Yield,
            shape = Area, col = Area)) +
  geom_point()
```



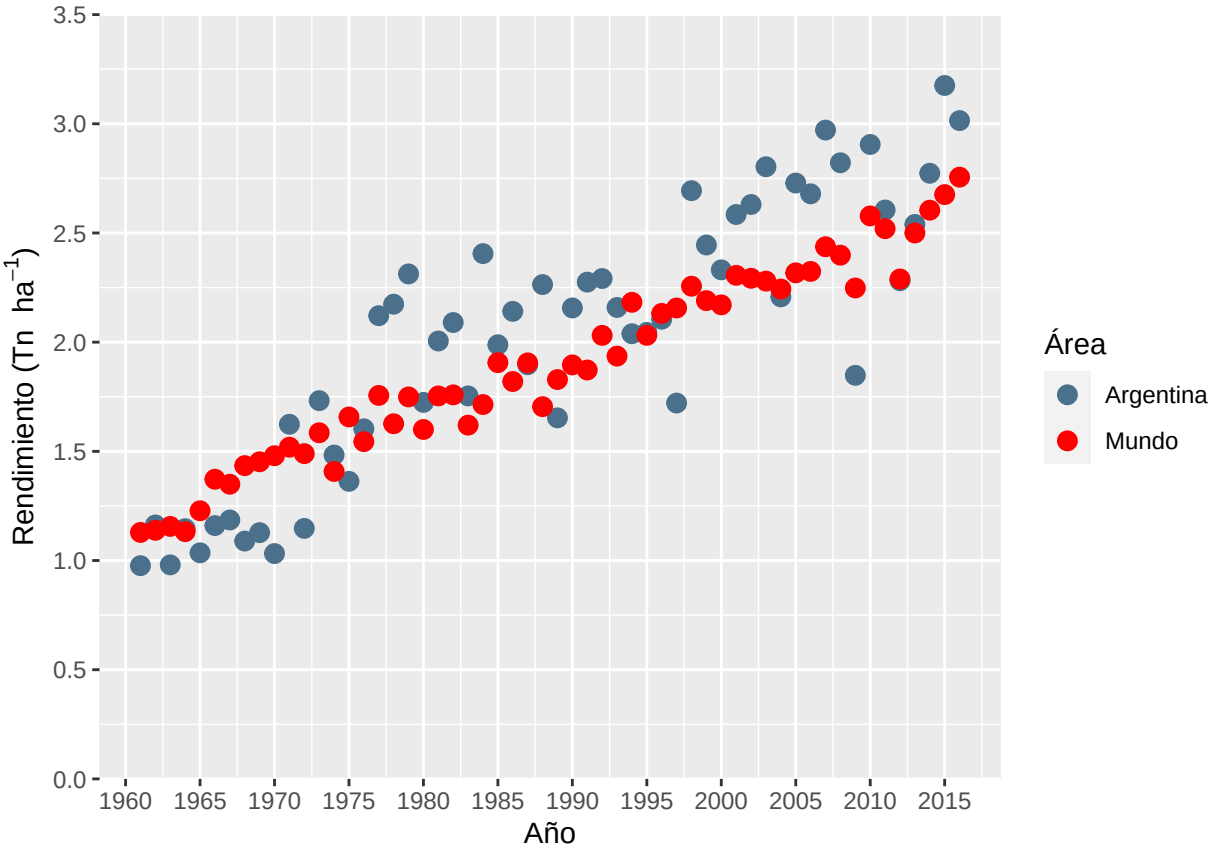
2.2. Modificando las escalas correspondientes a las variables de clasificación

Ahora vamos a modificar las escalas continuas (como hicimos antes), pero también escalas discretas. Recordemos que si usamos variables dentro del aes, estas serán de clasificación y generarán su propia leyenda (pueden ser variables continuas como año).

Ajuste manual

Usando `scale_color_manual` vamos a modificar manualmente la escala correspondiente a la variable de clasificación que determina el color. Este es un ejemplo para color, pero también podría ser shape, fill, etc (con `scale_shape_manual` , `scale_fill_manual` , etc). Este ajuste manual siempre estará asociado a que en el *mapping* haya una variable que determine el color, por lo que deberemos asignar tantos colores como niveles de la variable participen.

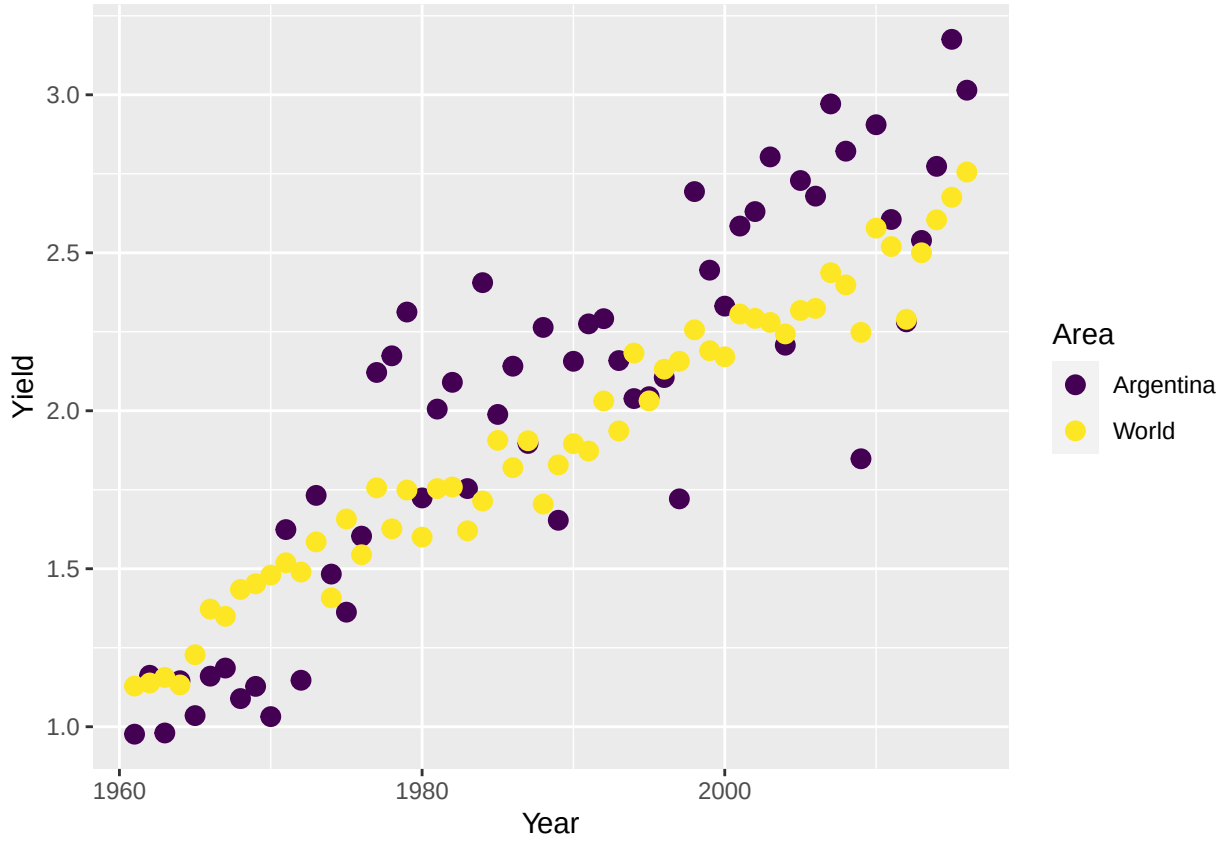
```
G3 <- ggplot(data = data,
       aes(x= Year, y = Yield, col= Area)) +
  geom_point(size=3)+
  scale_color_manual(values = c("skyblue4","red"),
                     labels=c("Argentina", "Mundo")) + # etiquetas de la leyenda
  scale_y_continuous(expand = c(0,0), limits = c(0, 3.5), breaks = seq(0, 5, by = .5))+
  scale_x_continuous(breaks = seq(1960,2016, by=5)) +
  labs(x = "Año",
       y = expression(paste("Rendimiento (Tn","  ", ha^-1 , ")")),
       col = "Área"); G3
```



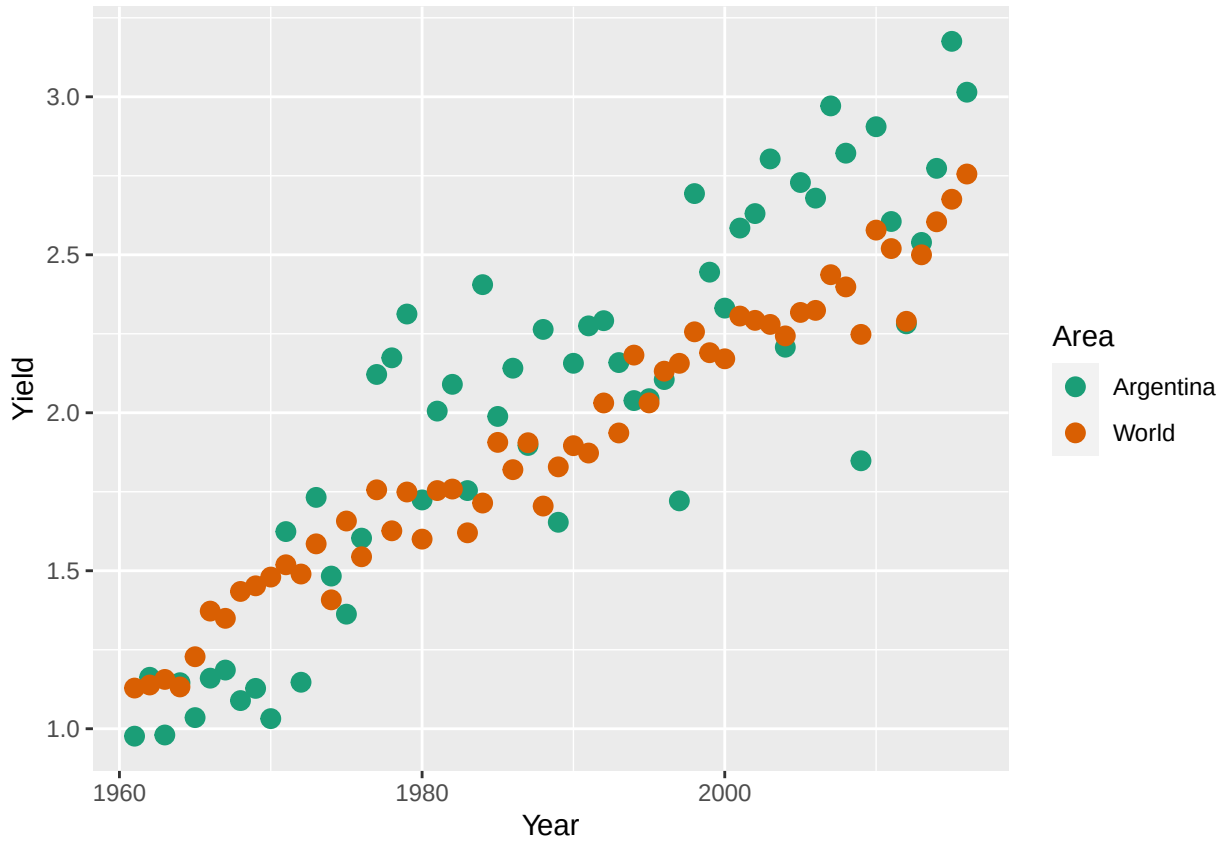
Utilizando paletas de color de forma automática

También se puede modificar los colores con paletas predeterminadas. Existen paletas desarrolladas con diferentes objetivos, por ejemplo que puedan ser vistas por daltónicxs. En este caso usamos **Viridis** que es para daltónicxs y **RcolorBrewer**

```
## Viridis
ggplot(data = data,
       aes(x= Year, y = Yield, col= Area)) +
  geom_point(size=3) +
  scale_color_viridis_d()
```




```
## Rcolorbrewer
ggplot(data = data,
       aes(x= Year, y = Yield, col= Area)) +
  geom_point(size=3) + scale_color_brewer(type = 'qual', palette = 2)
```

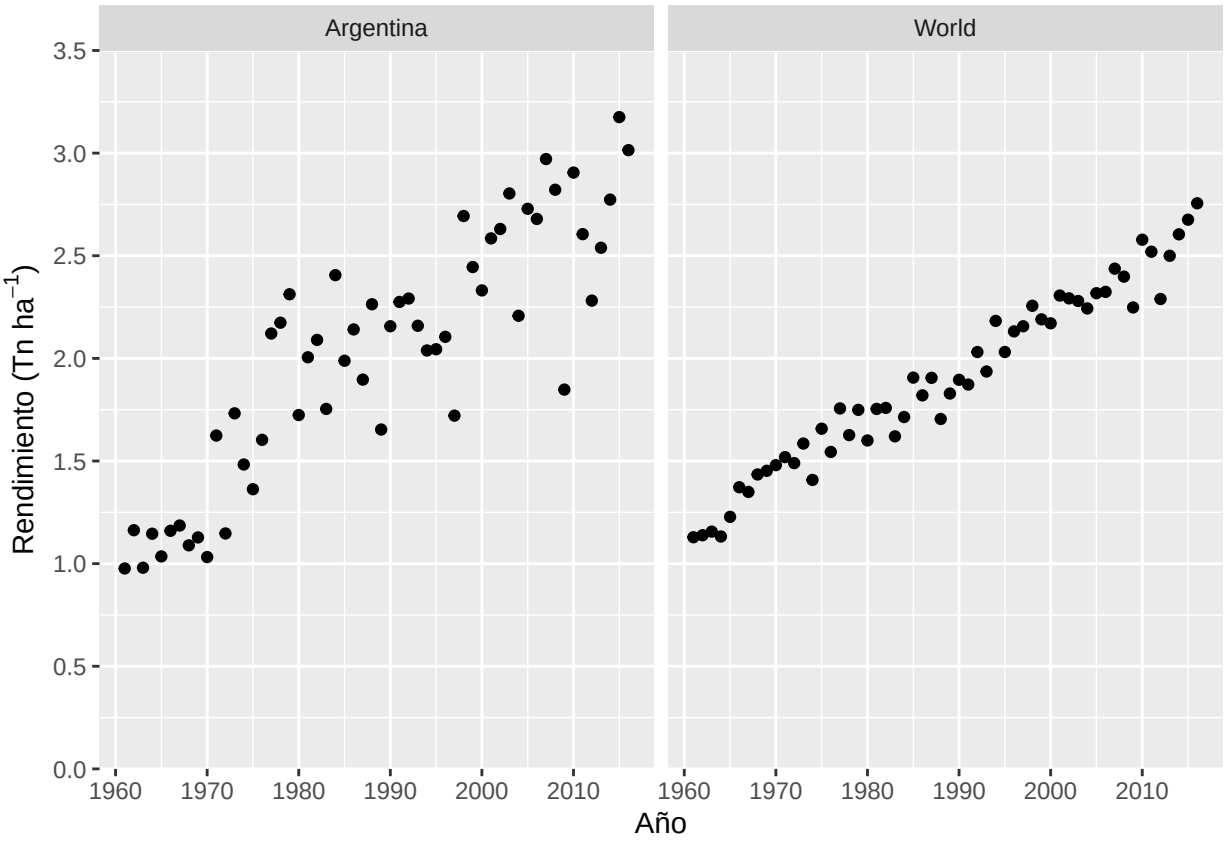


Para más información de estas paletas visitar https://colorbrewer2.org y https://ggplot2.tidyverse.org/reference/scale_viridis.html

2.3. Facets

Esta es una capa opcional que funciona haciendo un subset de los datos de acuerdo a una variable de clasificación. Con esto realizará varios gráficos uno por cada nivel de la variable, dividiendo la ventana gráfica y compartiendo la misma escala entre ellos.

```
G4 <- ggplot(data = data,
       aes(x= Year, y = Yield)) +
  geom_point() +
  facet_grid(.~Area)+
  scale_y_continuous(expand = c(0,0), limits = c(0, 3.5), breaks = seq(0, 5, by = .5))+
  scale_x_continuous(breaks = seq(1960,2016, by=10)) +
  labs(x= "Año",
       y = expression(paste("Rendimiento (Tn ",ha^-1, ")"))) ; G4
```



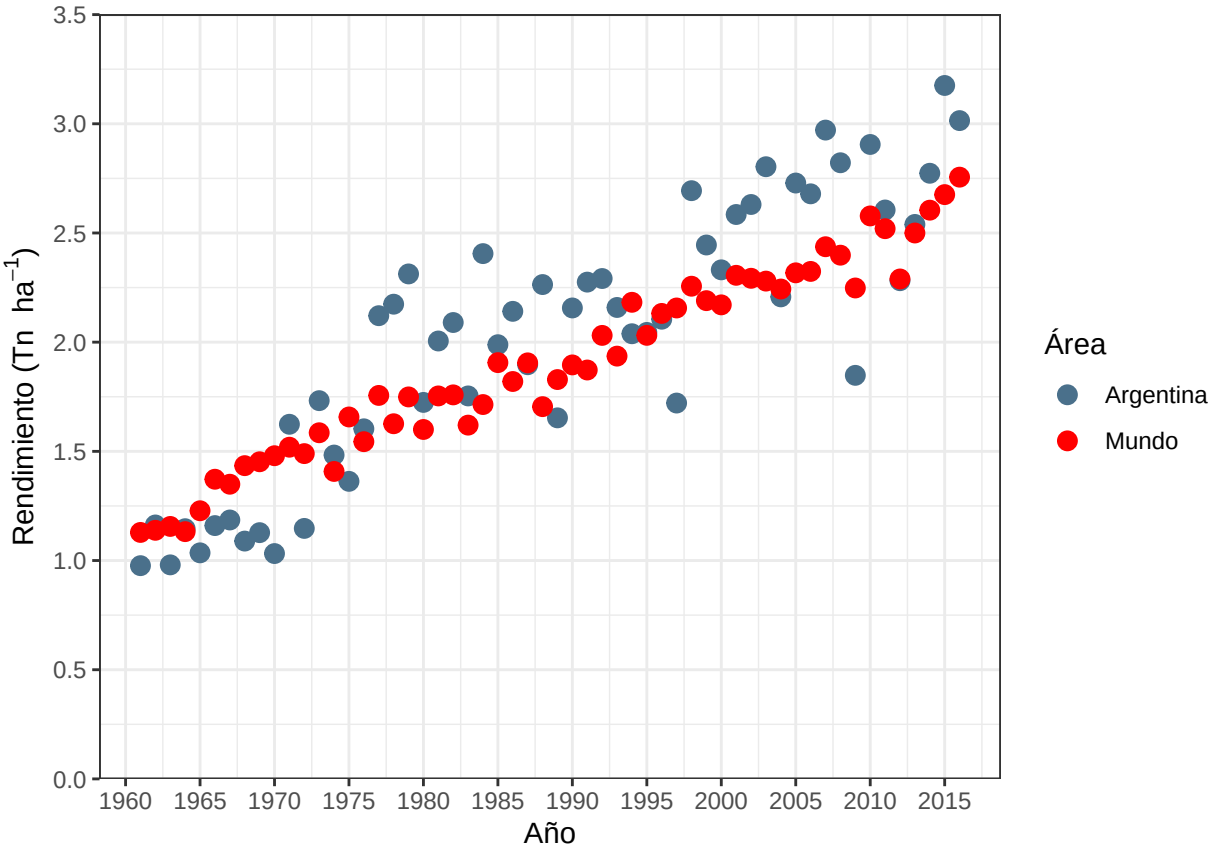
2.4. Tema o Theme

Define la estética general del gráfico, son todas las partes del gráfico que no tienen que ver con los datos. El color del background, los ejes, lineas, etc.

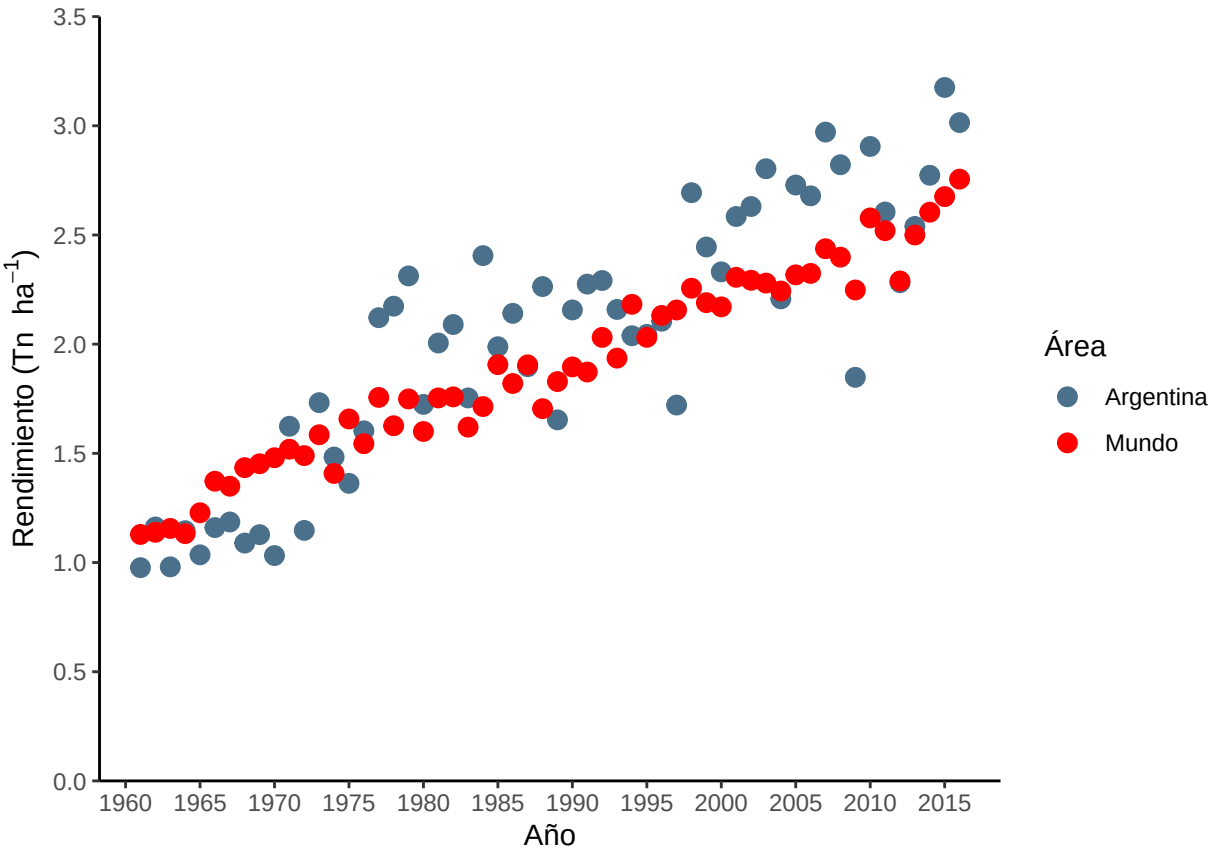
Temas predeterminados

Existen algunos **themes** predeterminados que determinan la estética del gráfico. Un ejemplo es el que viene por defecto que venimos utilizando `theme_grey()`, pero existen otros como:`theme_bw()` `theme_light()` `theme_dark` `theme_void` , etc.

```
G3 + theme_bw()
```



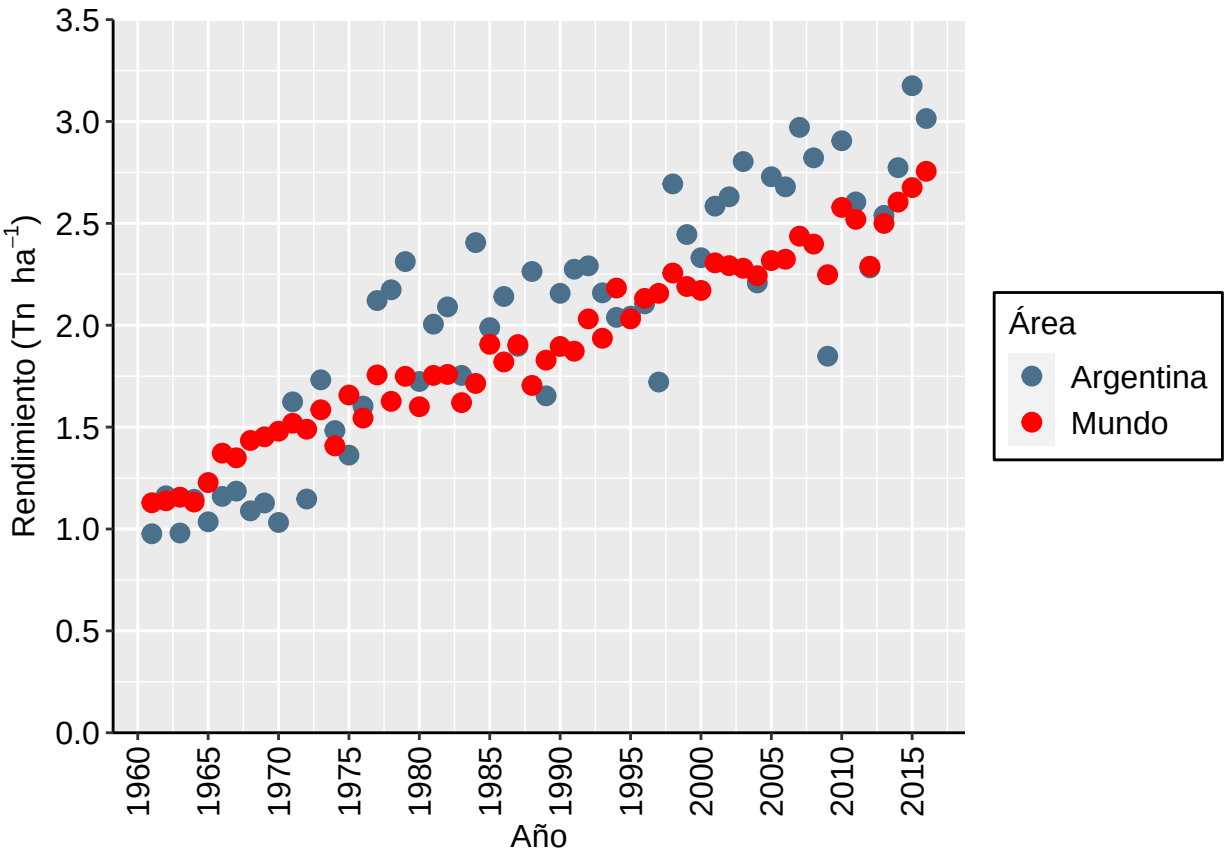

```
G3 + theme_classic()
```



Edición manual del tema

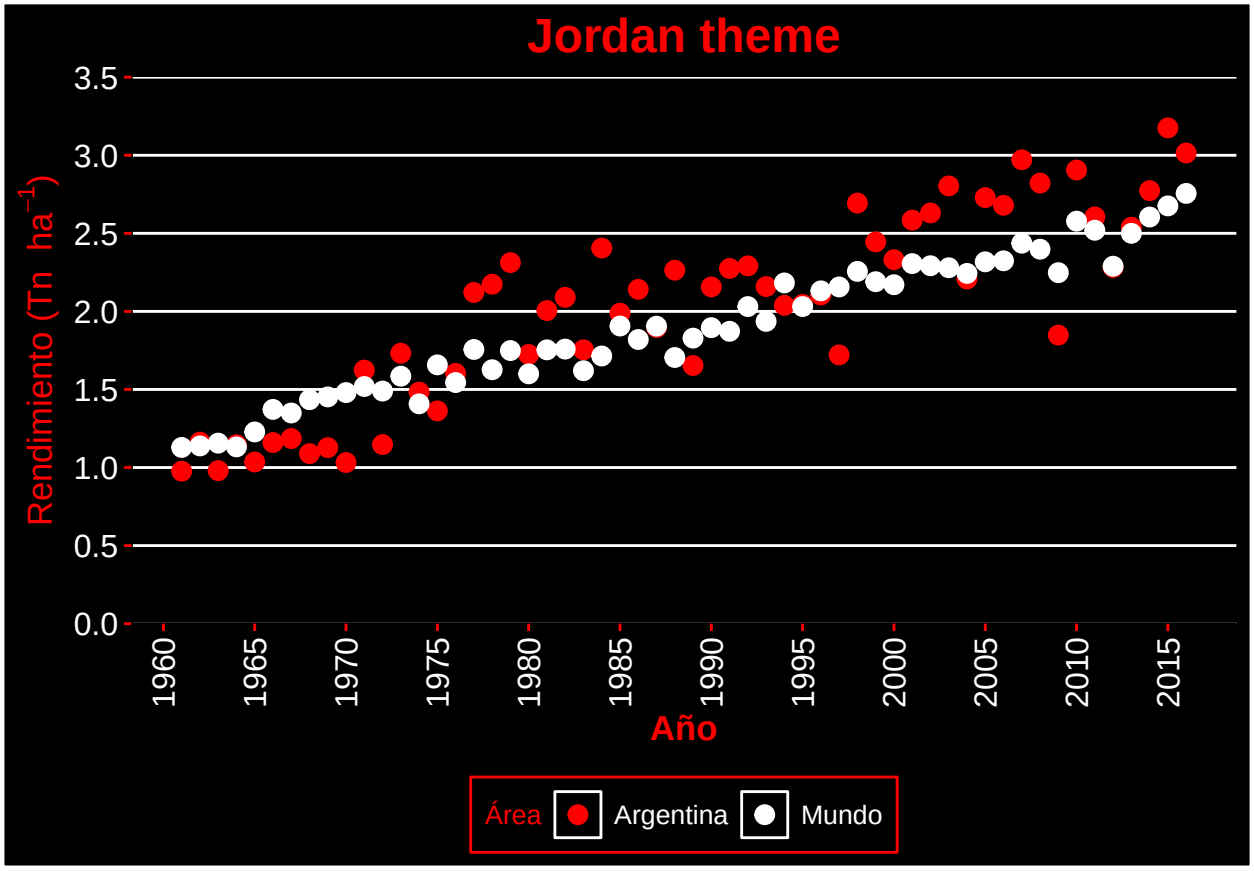
Acá es importante saber que se puede modificar cada ítem gráfico que queramos, desde la posición de la leyenda con sus títulos, hasta el color de las lineas del fondo. Lo que se les ocurra. Un consejo, una vez que obtengan una estética estándar propia que les cierre, guárdenlo en una sesión de R, luego simplemente pegan el objeto cada vez que hagan un gráfico, y se olvidan de la estética. Para nosotros es uno de los mejores usos de ggplot2.

```
G3 <- G3 +
  theme(text = element_text(size=12, colour = "black"),
        axis.text = element_text(size=12 , colour = "black"),
        axis.text.x = element_text(angle = 90, vjust = 0.5),
        legend.text = element_text(size=12, colour = "black"),
        legend.background=element_rect(colour="black"),
        axis.line = element_line(colour = "black")) ; G3
```




```
## un ejemplo freaky para quienes nos gusta el basquet

MICHAEL <-   theme(
    legend.position = "bottom", legend.title = element_text(colour = "red", size = 10),
    legend.background = element_rect(fill = "black", colour = 'red'),
    legend.key = element_rect(fill = "black", colour = "white"),
    legend.text = element_text(colour = "white", size = 10),
    plot.background = element_rect(fill = "black", colour = "white"),
    panel.background = element_rect(fill = "black"),
  #  panel.background = element_rect(fill = "white"),
    axis.text = element_text(colour = "white"),
    plot.title = element_text(colour = "red", face = "bold", size = 18, vjust = 1,
                              hjust = 0.5),
    axis.title = element_text(colour = "red", face = "bold", size = 13),
    panel.grid.major.y = element_line(colour = "white"),
    panel.grid.minor.y = element_blank(),
    panel.grid.major.x = element_blank(),
    panel.grid.minor.x = element_blank(),
    strip.text = element_text(colour = "white"),
    strip.background = element_rect(fill = "white"),
    axis.ticks = element_line(colour = "red")
  )


G3 + scale_color_manual(values = c('red','white'), labels = c('Argentina', 'Mundo')) + ggtitle('Jordan
  MICHAEL
```

```
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
```



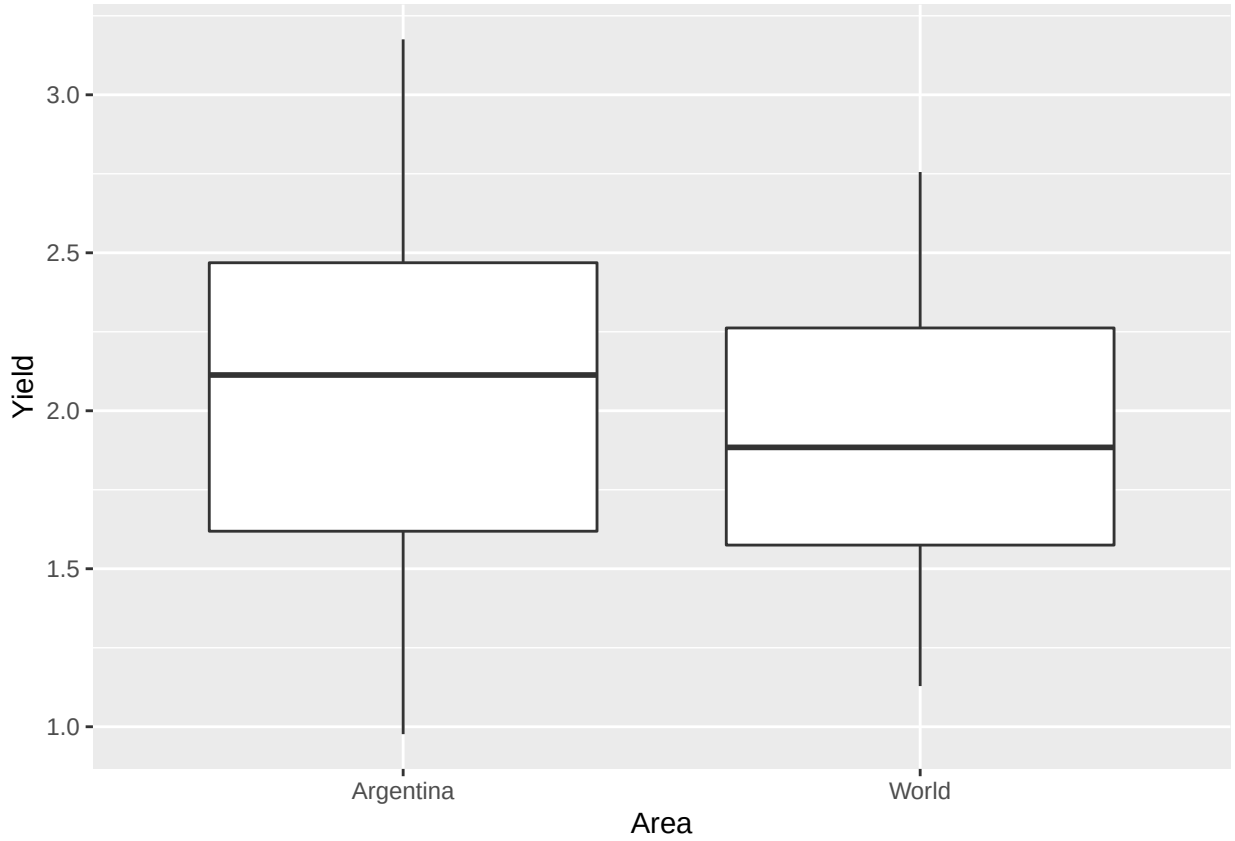
3. Otros gráficos con los mismos datos

Vamos a explorar la idea de la generalización de la forma de hacer gráficos. Con exactamente los mismos datos, las representaciones pueden ser diferentes. Esto lo hacemos con los `geom_`

Boxplot

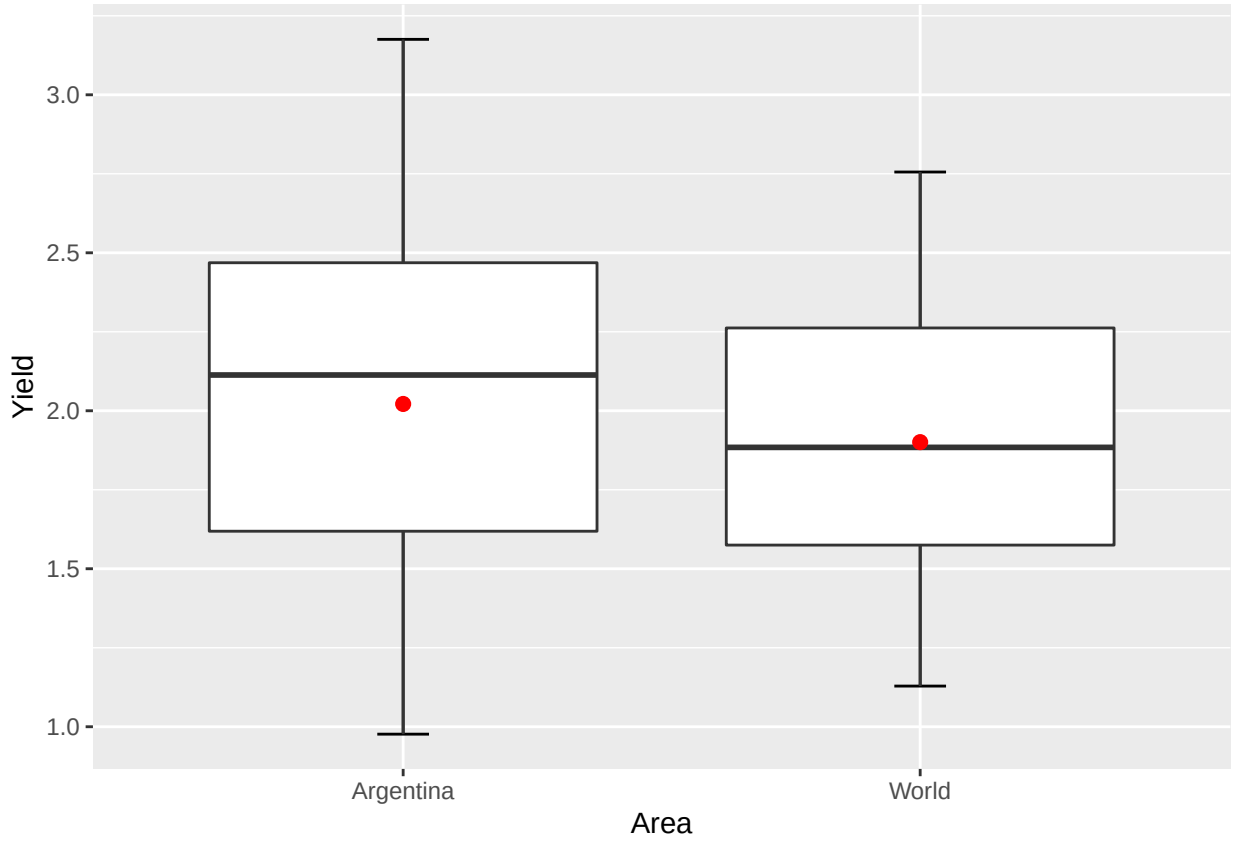
Los `geom` como el boxplot incluyen una capa de *statistics* que está implícita. Muchas veces los datos crudos no representan lo que queremos graficar. En estos casos, la capa de *stats* transforma los datos de cierta forma que nos interese (usando estadística). En este caso calcula los cuartiles para dibujar las cajas.

```
ggplot(data = data,
       aes(x= Area, y = Yield)) +
  geom_boxplot()
```



Podemos agregar el extremo y la media (el orden importa)

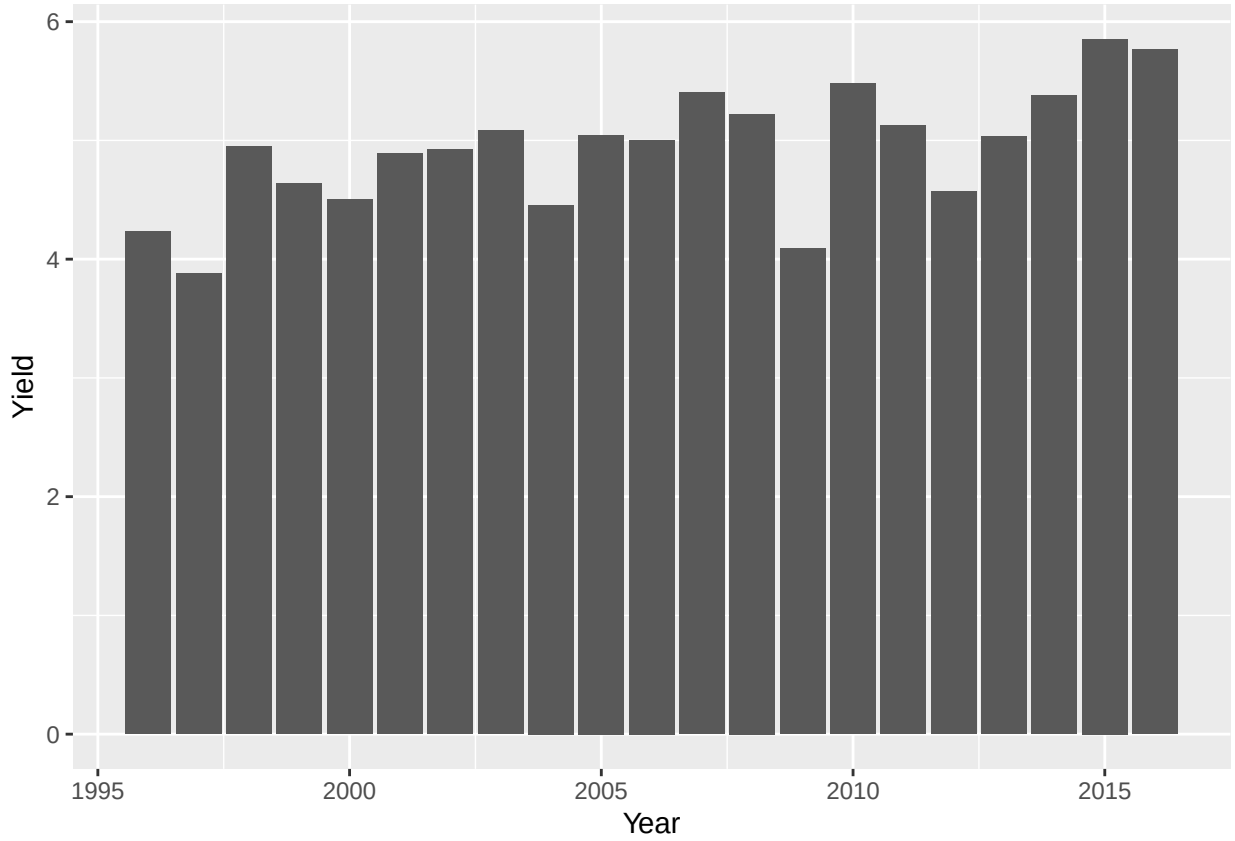
```
ggplot(data = data,
       aes(x= Area, y = Yield)) +
  stat_boxplot(geom ='errorbar', width=0.1)+
  geom_boxplot()+
  stat_summary(fun=mean, geom="point", shape=16, size=2.5, colour= "red")
```



Columnas

Otro ejemplo de representación, esta vez con `geom_col`

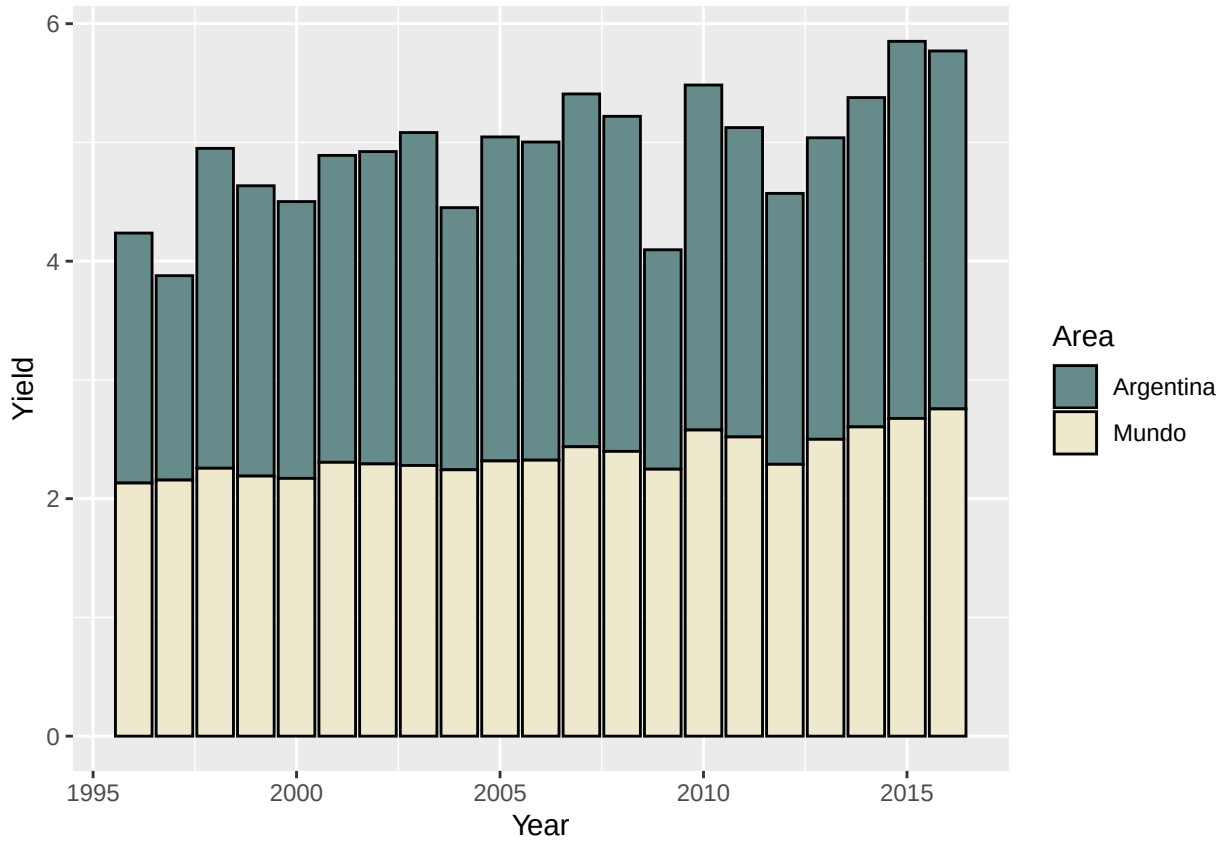
```
data_96 <- data %>% filter(data$Year>1995) ## filtramos los años mayores a 1995

ggplot(data = data_96,
       aes(x= Year, y = Yield)) +
  geom_col()
```



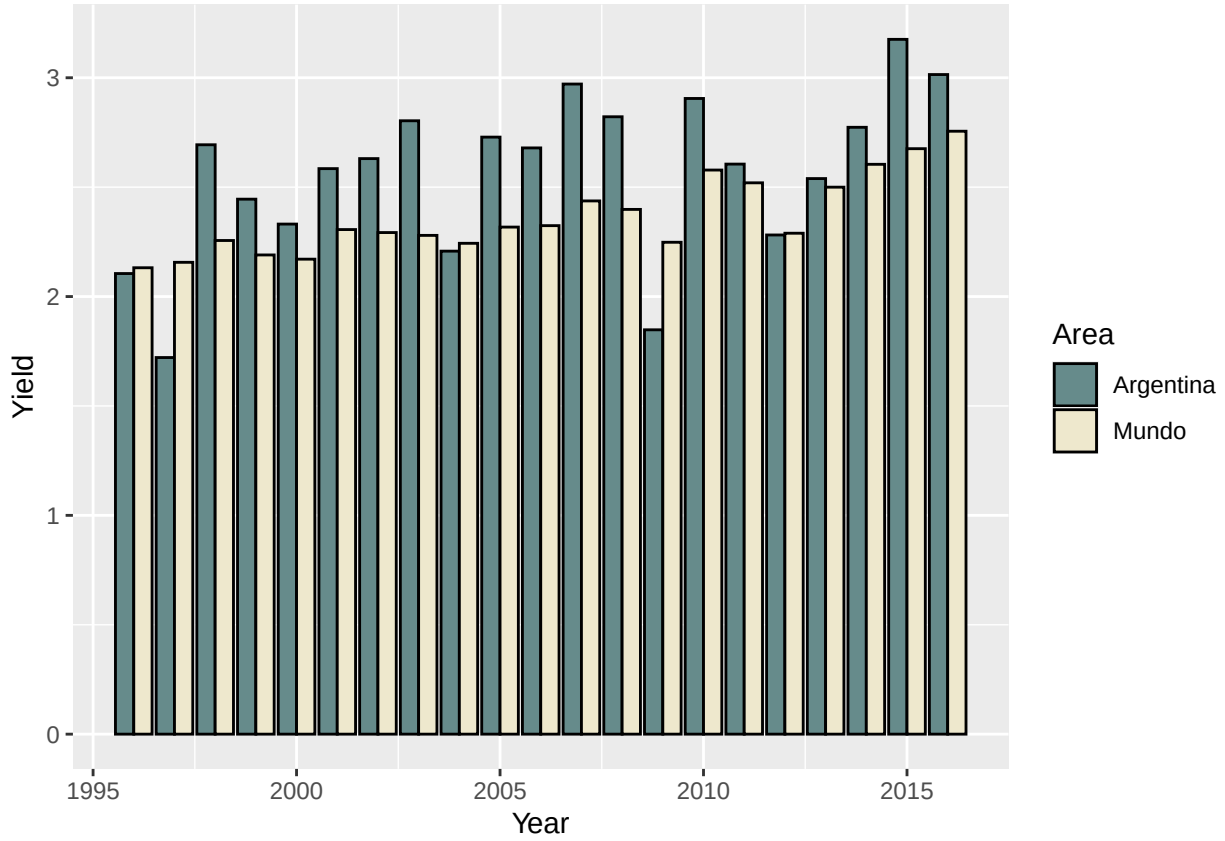
Clasificando por variables

Cuando clasificamos por color, por defecto las barras se muestran apiladas, si las queremos al lado debemos cambiar el `position`

```
G5 <- ggplot(data = data_96,
       aes(x= Year, y = Yield, fill= Area)) +
  geom_col(col= "black")+
  scale_fill_manual(values = c("paleturquoise4","cornsilk2"),
                    labels=c("Argentina", "Mundo")); G5
```

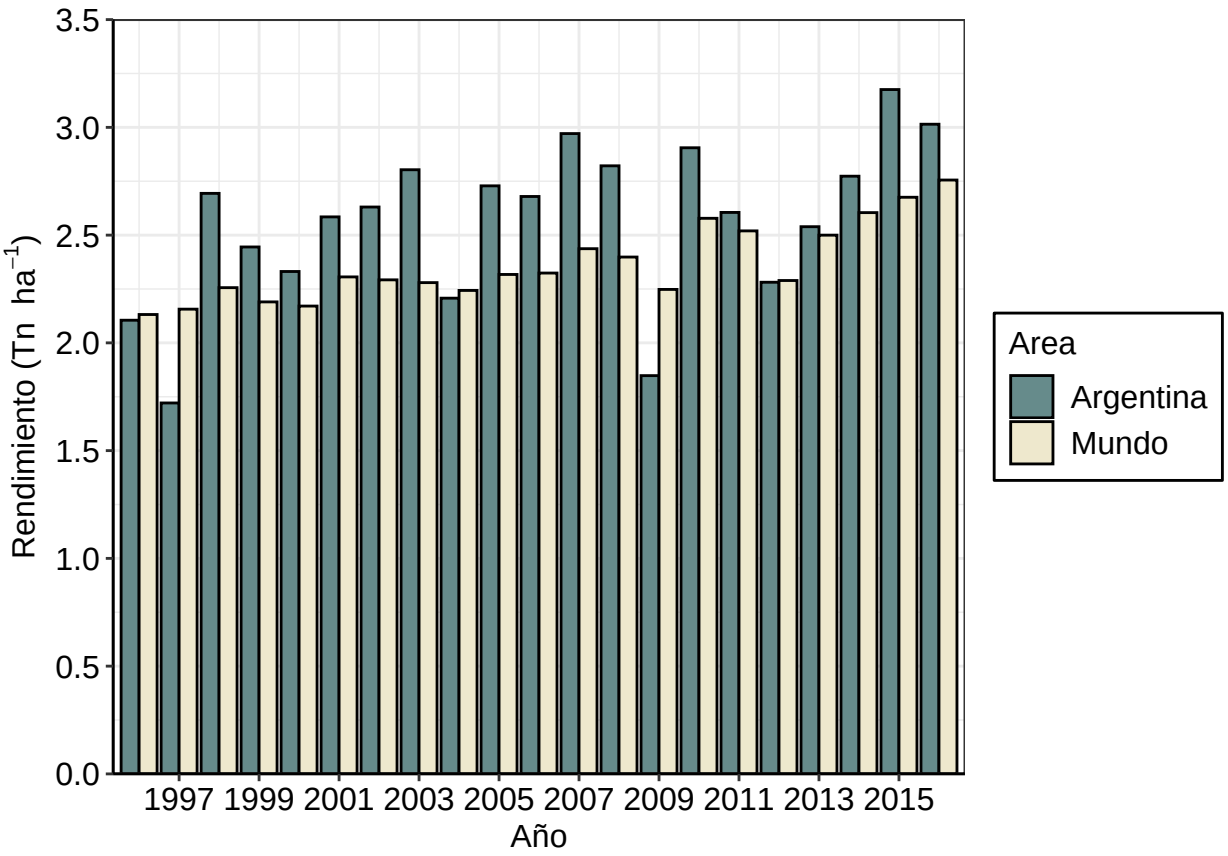



```
G5 <- ggplot(data = data_96,
       aes(x= Year, y = Yield, fill= Area)) +
  geom_col(col= "black", position = 'dodge')+
  scale_fill_manual(values = c("paleturquoise4","cornsilk2"),
                    labels=c("Argentina", "Mundo")); G5
```



Incorporando cuestiones de escala y estéticas vistas anteriormente, mejoramos nuestro gráfico:

```
G5 <- G5 +
  scale_y_continuous(expand = c(0,0), limits = c(0, 3.5), breaks = seq(0, 5, by = .5))+
  scale_x_continuous(expand = c(0,0.2), breaks = seq(1995,2016, by=2)) +
  labs(x= "Año",
       y = expression(paste("Rendimiento (Tn","  ", ha^-1, ")")))+
  theme_bw()+
  theme(text = element_text(size=12, colour = "black"),
        axis.text = element_text(size=12 , colour = "black"),
        legend.text = element_text(size=12, colour = "black"),
        legend.background=element_rect(colour="black"),
        axis.line = element_line(colour = "black")) ; G5
```



4. Un poquito más de ggplot2 y un poquito más allá dentro de tidyverse

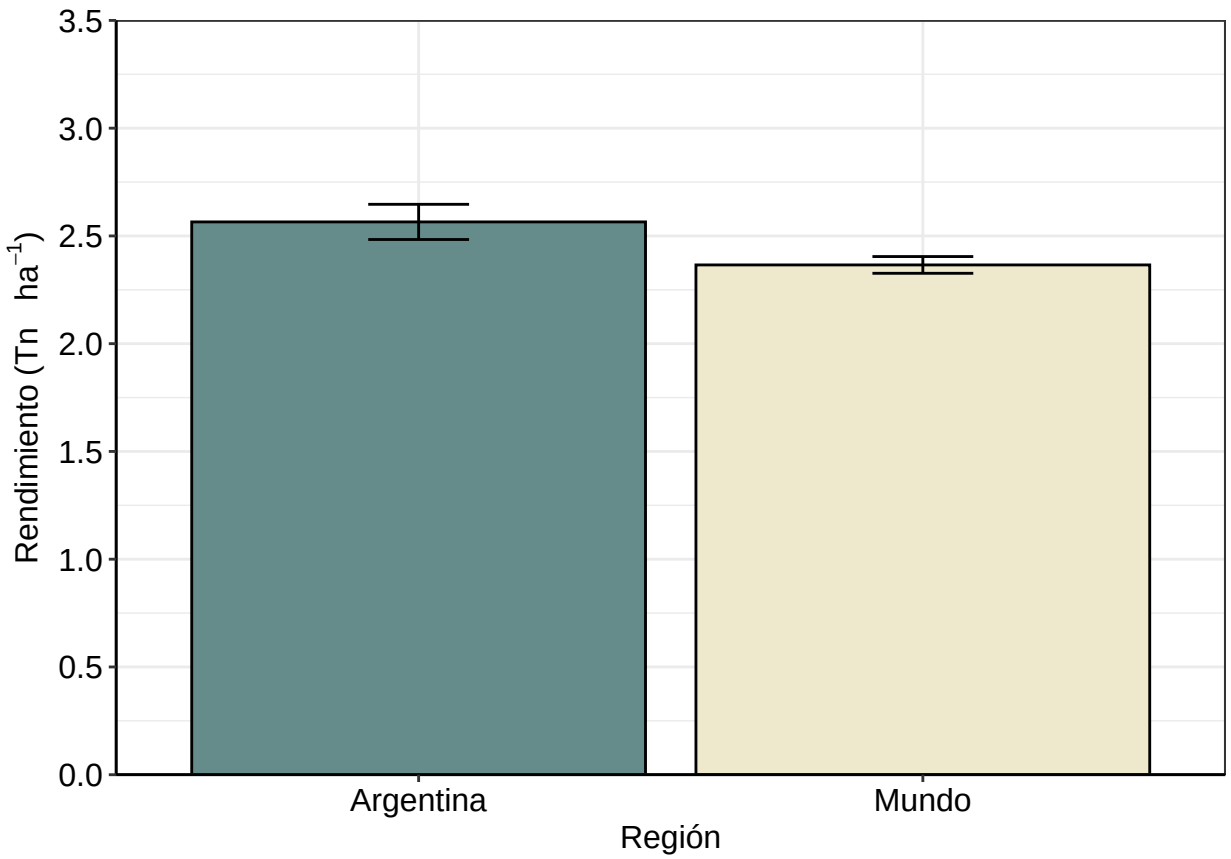
Ahora vamos a hacer un poquito de manejo de tables con dplyr (tidyverse) y aplicar un poco lo que hemos visto de ggplot2

4.1. Transformando los datos para graficar medias y desvíos

Esta función probablemente se pueda hacer dentro de una capa con los stats que hemos visto, pero la forma que encontramos es utilizando **dplyr** manejando la tabla original. Al usar pipes %>%y no crear un nuevo objeto, nos permite graficarlo directamente. Tener en cuenta que la función `se` pertenece al paquete *sciplot*.

```
G6 <- data_96 %>%  select(Area, Yield) %>% # seleccionamos la variables de interés
group_by(Area) %>%  # marcamos las variables que agrupan
  summarise_all(funs(mean, se, sd)) %>% # calculamos los parámetros
  ggplot(aes(Area, mean))+ # y las graficamos directamente
  geom_col(fill=c("paleturquoise4","cornsilk2"), col="black")+
  geom_errorbar(aes(ymin=mean-se, ymax=mean+se), width=.2)+
  scale_y_continuous(expand = c(0,0), limits = c(0, 3.5), breaks = seq(0, 5, by = .5))+
  scale_x_discrete(labels=c("Argentina", "Mundo"))+
  labs(x= "Región",
       y = expression(paste("Rendimiento (Tn   ",ha^-1, ")")))+
  theme_bw()+
  theme(text = element_text(size=12, colour = "black"),
        axis.text = element_text(size=12 , colour = "black"),
```

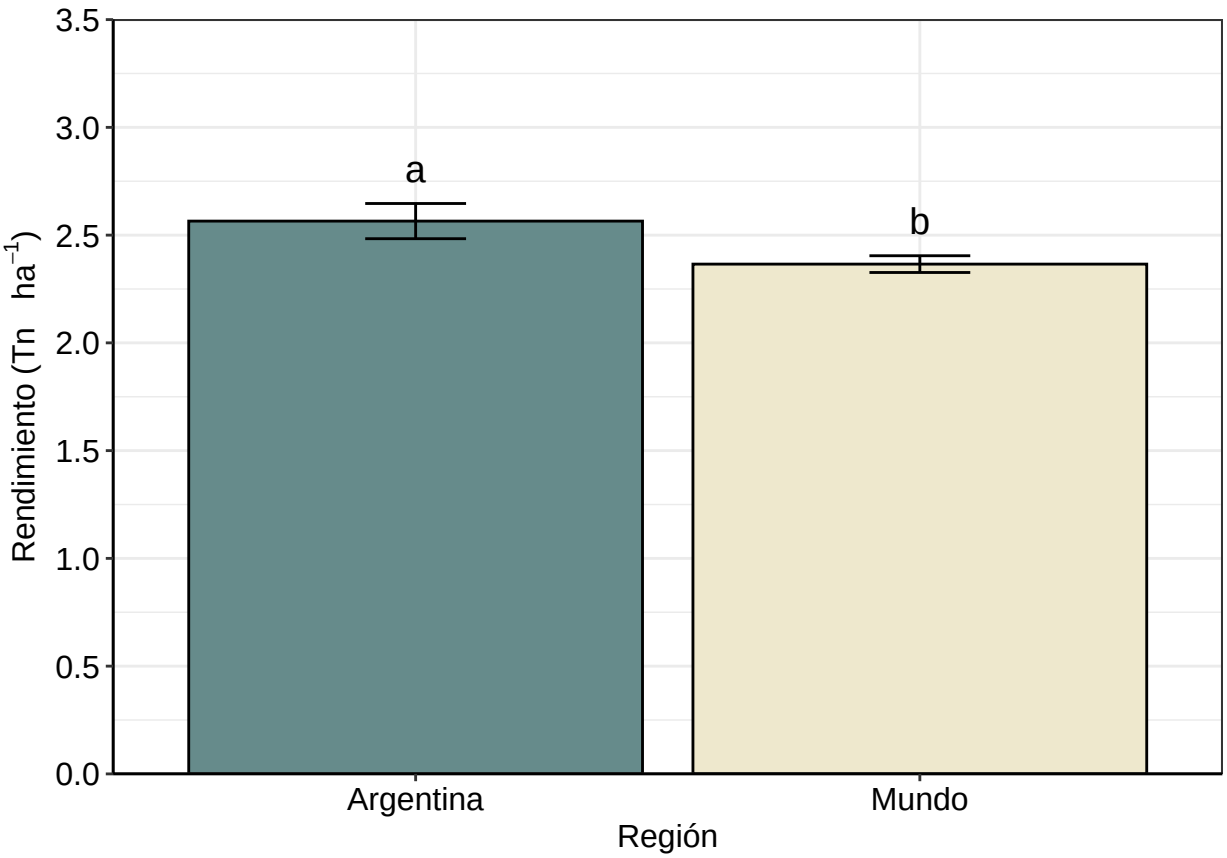
```
        legend.text = element_text(size=12, colour = "black"),
        legend.background=element_rect(colour="black"),
        axis.line = element_line(colour = "black")) ; G6
```



4.2. Agregando texto

Acá no utilizamos el mapping global que definimos antes, sino que usamos "otros" datos. Lo haremos con `geom_text`

```
G6 + geom_text(mapping = aes(y=mean+se, label= c("a", "b")),
               vjust=-0.8, size=5)
```



4.3 Combinando representaciones (geoms)

Una buena forma de visualizar algunas variables es superponiendo diferentes tipos de gráficos. Para ello vamos a integrar algunas cosas que venimos trabajando.

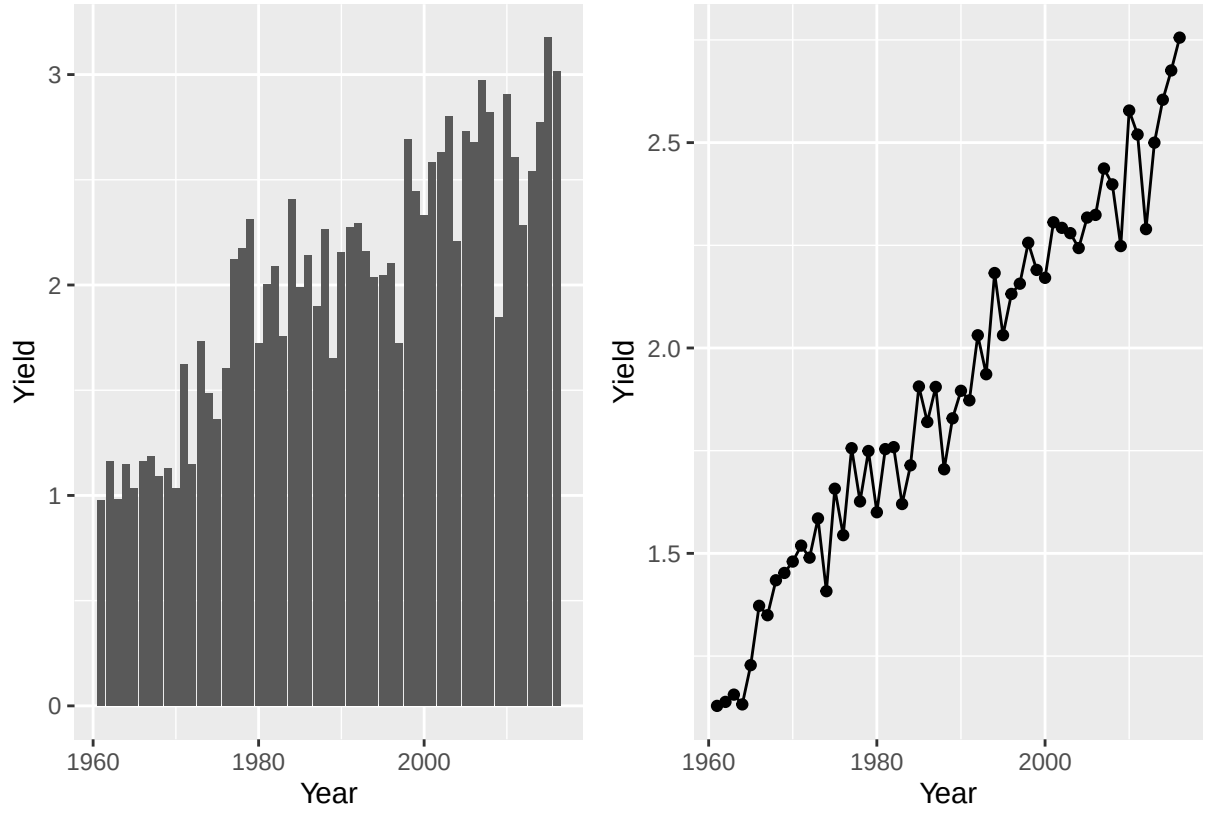
Supongamos que queremos comparar el rendimiento en Argentina y el mundo por años en los últimos 20 años. Para esto ya habíamos generado el gráfico ¿G5?, veamos otra opción.

Generamos una base de datos para cada región usando *dplyr*

```
data_argentina <- data %>% filter(data$Area == "Argentina")
data_mundo <- data %>% filter(data$Area != "Argentina")
```

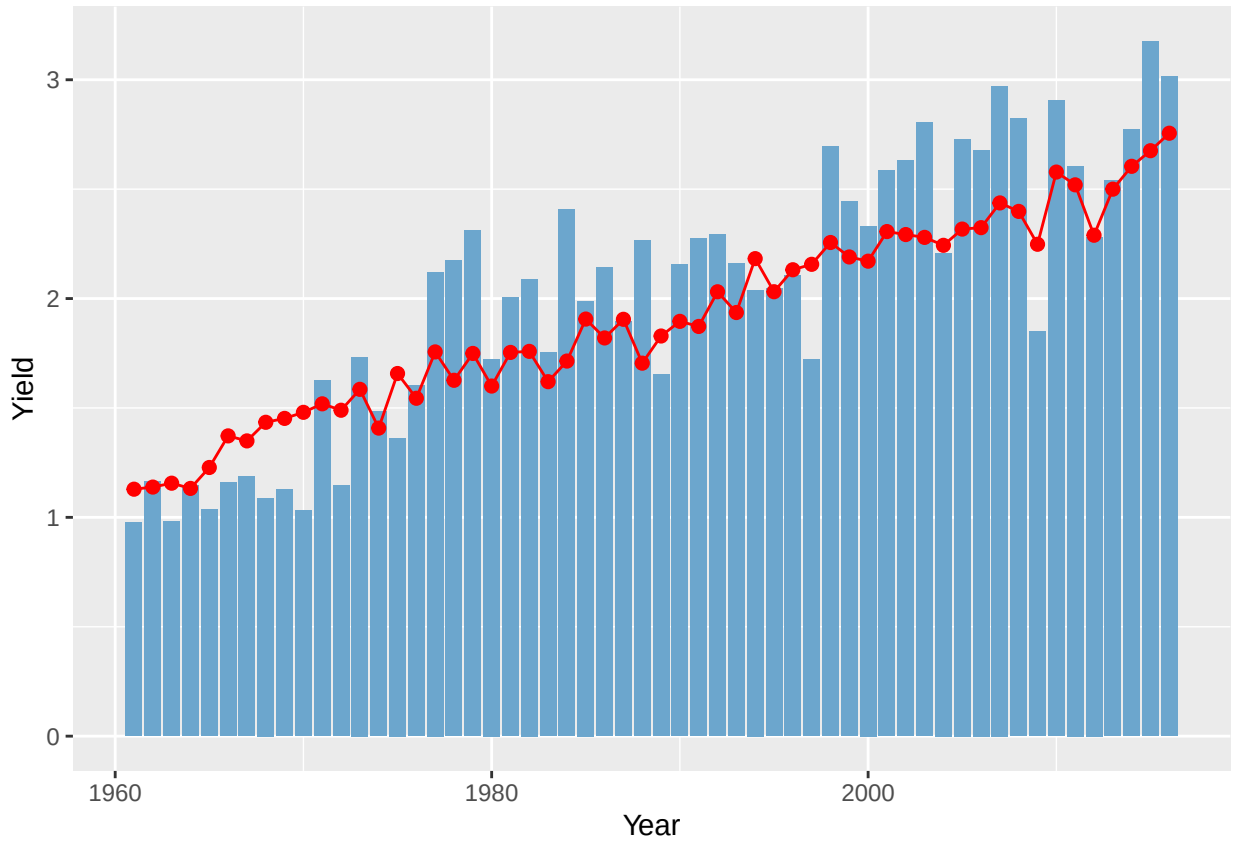
Luego vamos a superponer un gráfico base de columnas de Argentina y uno de puntos y de lineas del mundo

```
G7 <- ggplot(data_argentina, aes(Year, Yield)) +
  geom_col()
G8 <- ggplot(data_mundo, aes(Year, Yield)) +
  geom_line() +
  geom_point()
G7 + G8
```



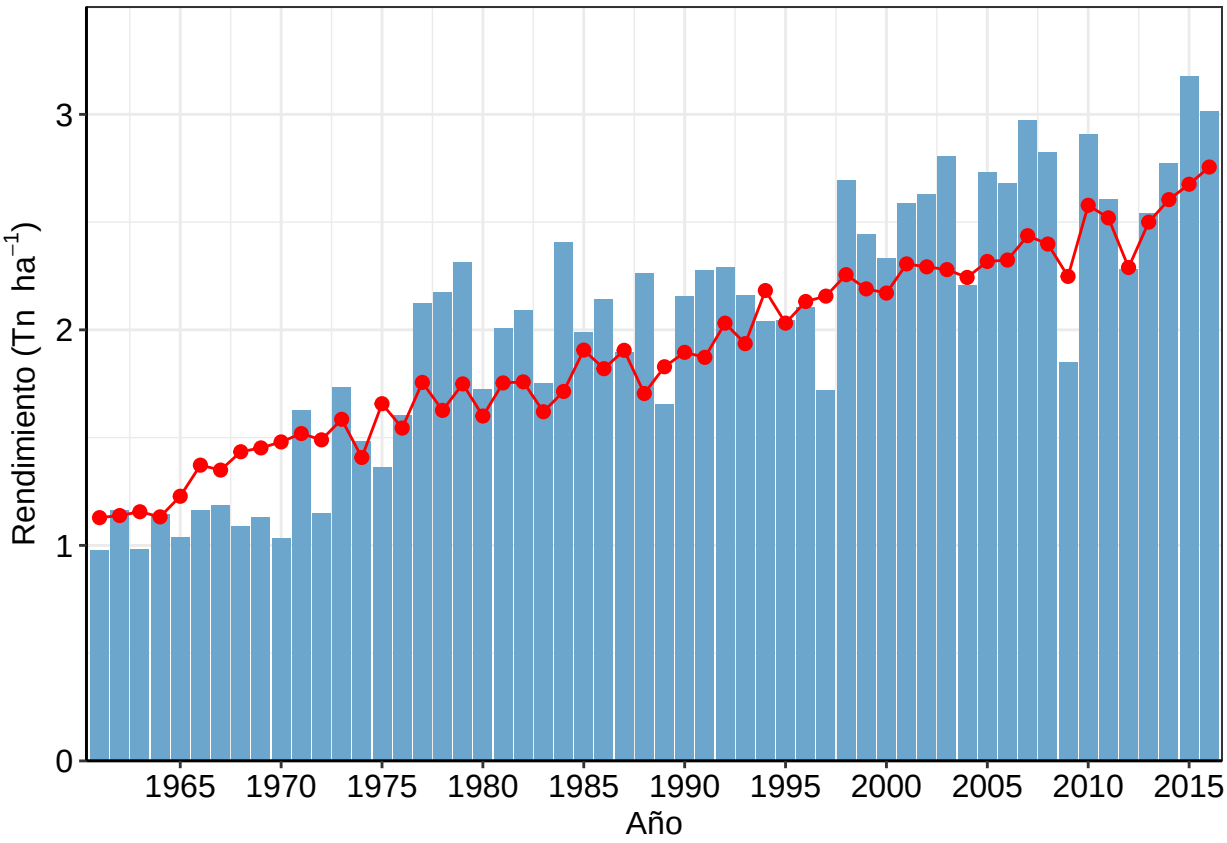
¿Cómo los combinamos? Cambiando los **datos** y el **mapping** dentro de los geoms, es decir no utilizando el global para uno de ellos. Veamos

```
G7 <- ggplot(data = data_argentina, mapping = aes(Year, Yield)) + ## funcion global
  geom_col(fill="skyblue3") + ## para el geom_col utilizamos los datos globales
  geom_line(data = data_mundo, aes(Year, Yield), col= "red")+ ## cambiamos los datos
  geom_point(data = data_mundo, aes(Year, Yield), col= "red", size=2) ; G7 ## igual que antes
```



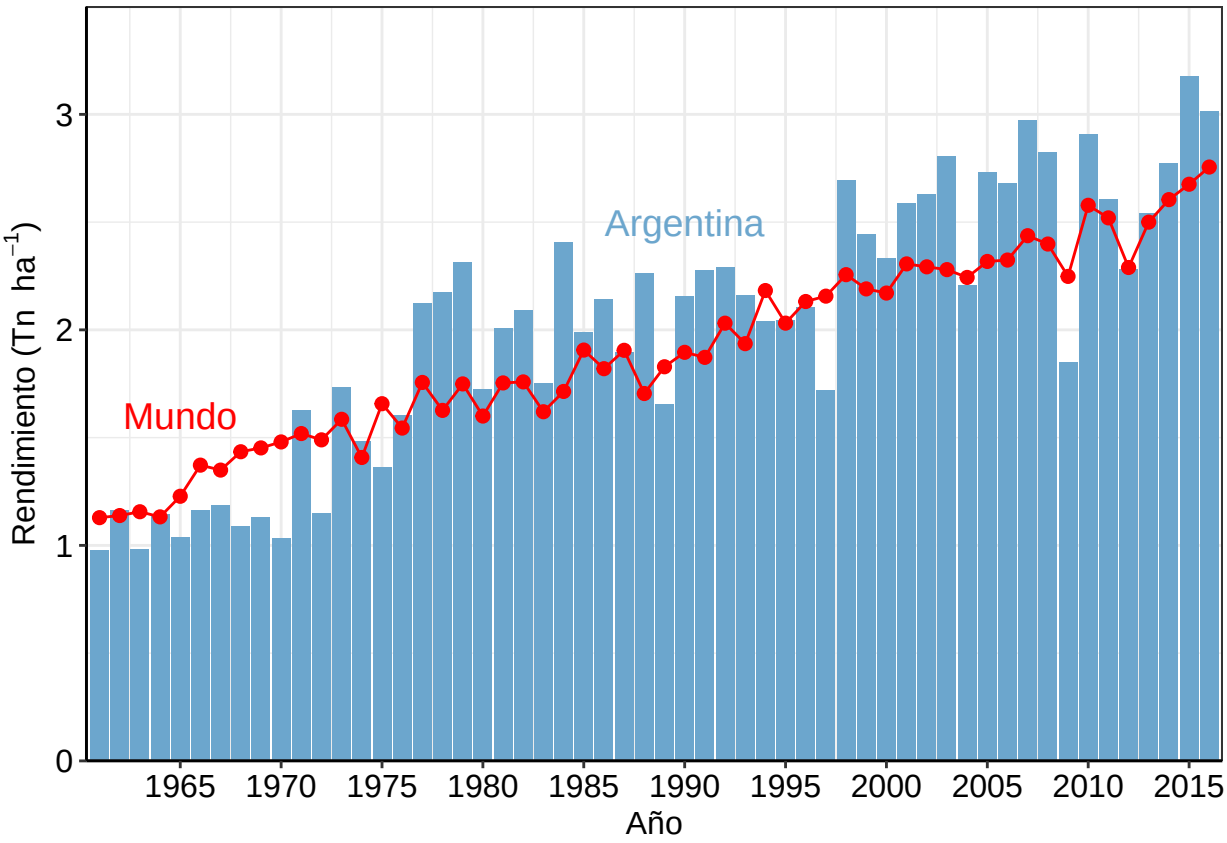
Copiando y pegando lo previo podemos mejorar la estética de nuestro gráfico

```
G7 <- G7 +
  scale_y_continuous(expand = c(0,0), limits = c(0, 3.5, breaks = seq(0, 5, by = .5))) +
  scale_x_continuous(expand = c(0,0.2), breaks = seq(1960,2016, by=5)) +
  labs(x= "Año",
       y = expression(paste("Rendimiento (Tn  ",ha^-1, ")")))+
  theme_bw()+
  theme(text = element_text(size=12, colour = "black"),
        axis.text = element_text(size=12 , colour = "black"),
        legend.text = element_text(size=12, colour = "black"),
        legend.background=element_rect(colour="black"),
        axis.line = element_line(colour = "black")) ; G7
```



Falta lograr identificar en el gráfico las dos bases de datos. Agregamos texto aclarando

```
G7 <- G7 +
  annotate("text", x=1990, y= 2.5, label = "Argentina", col="skyblue3", size=5)+
  annotate("text", x=1965, y= 1.6, label = "Mundo", col="red", size=5) ; G7
```

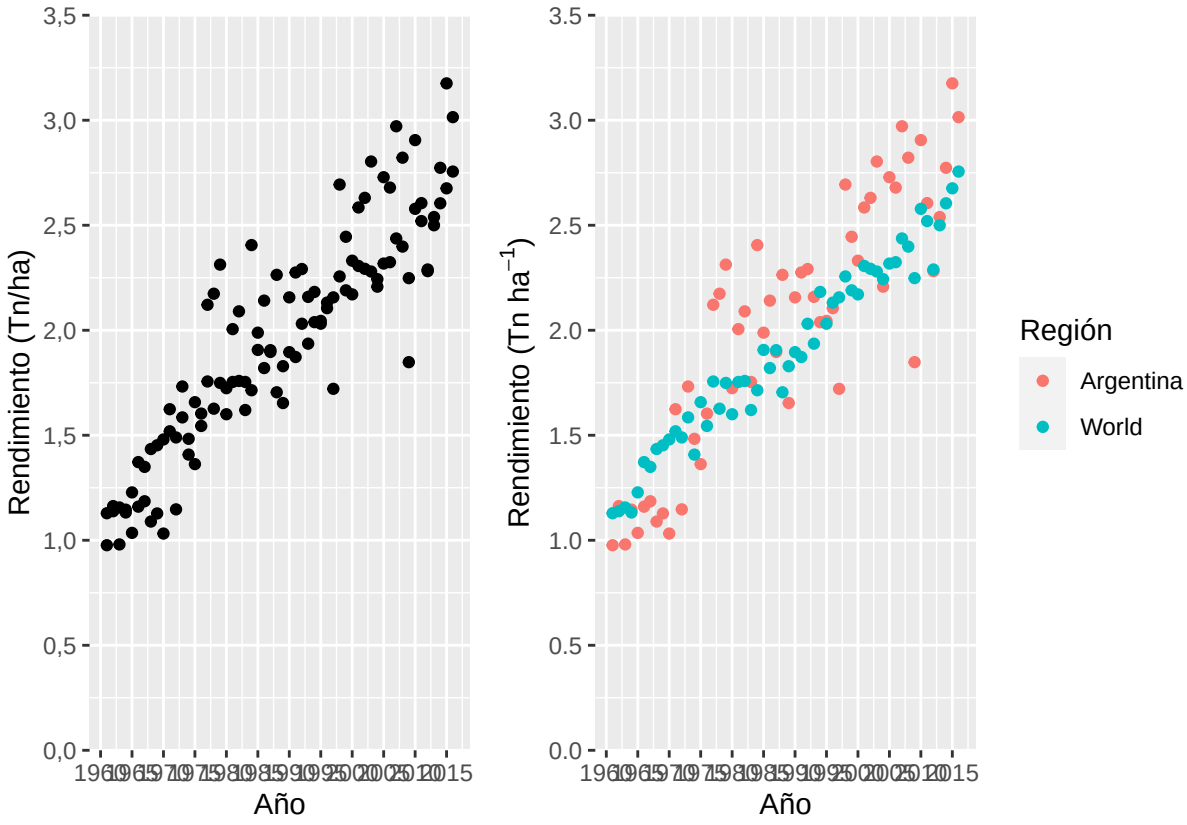


5. Dividiendo la pantalla gráfica para mostrar más de un gráfico

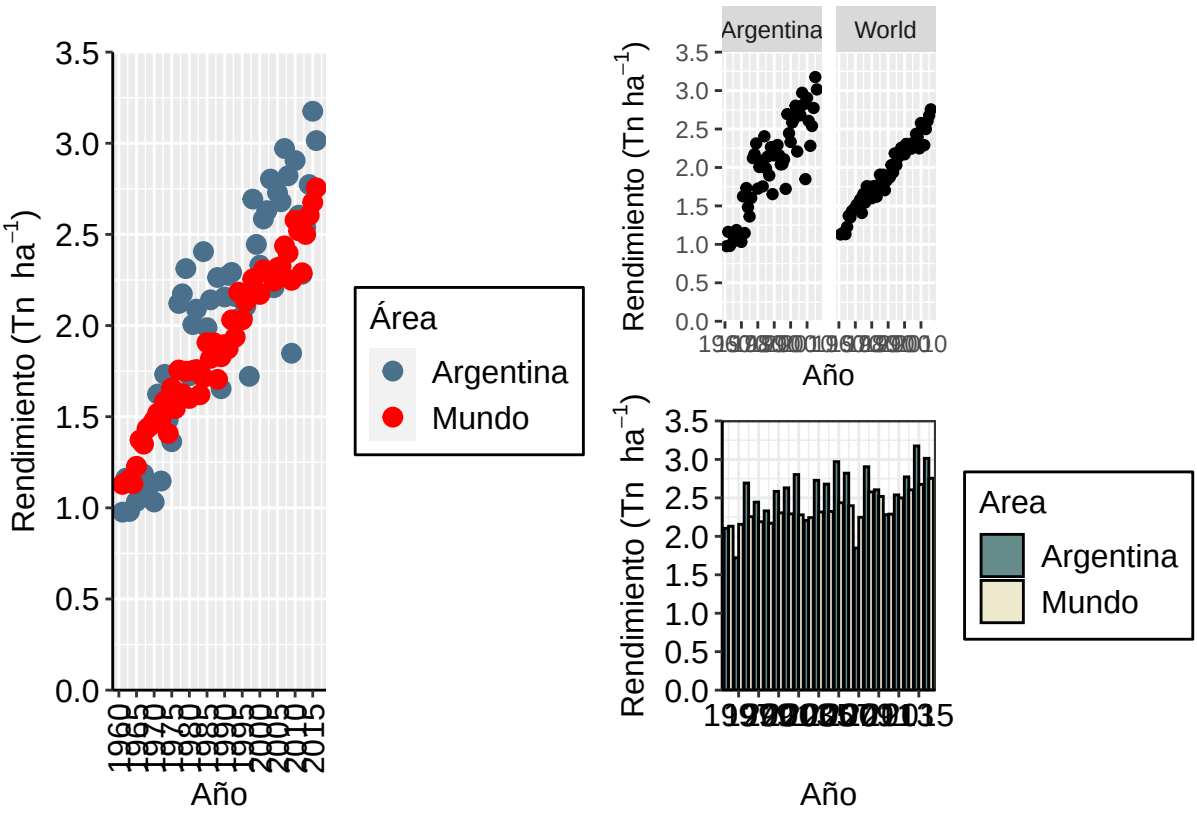
Podemos combinar diferentes gráficos en uno gracias al paquete **patchwork** que ya hemos utilizado pero ahora exploraremos su potencial y sus funciones.

5.1. Algunos ejemplos

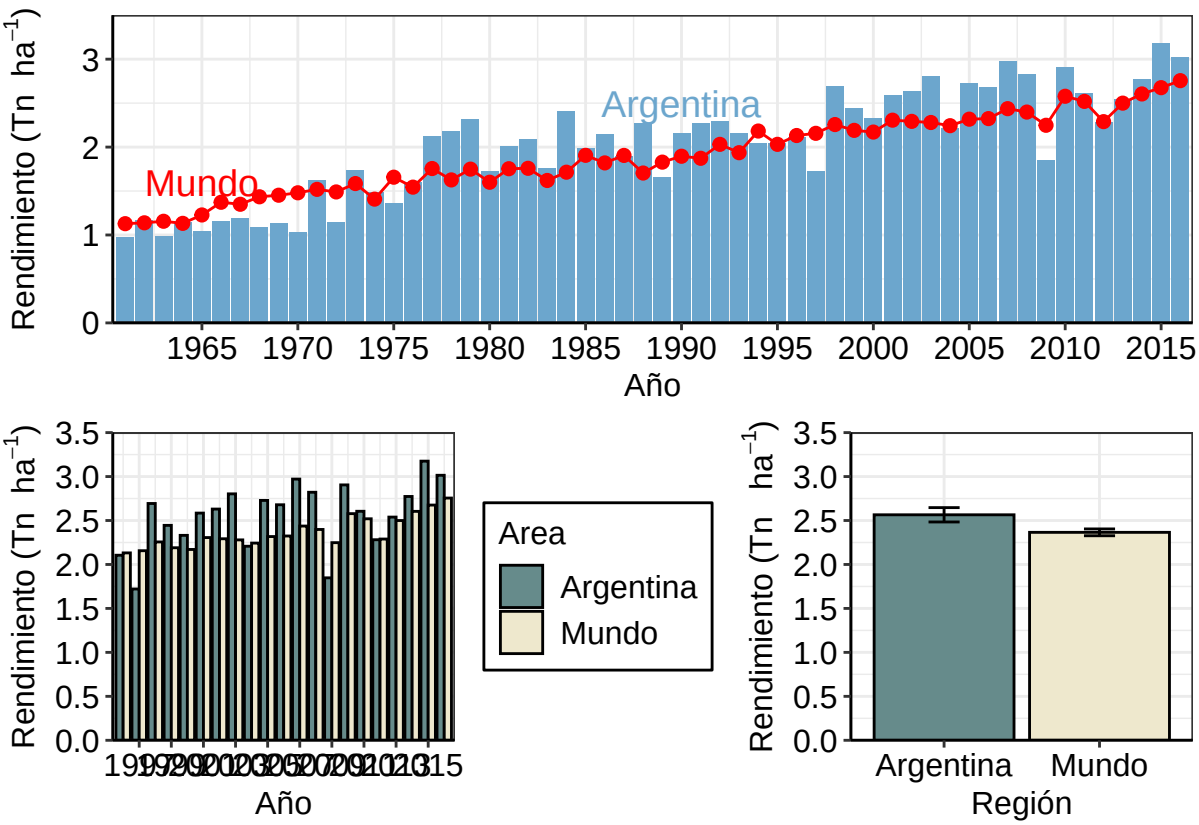
```
G1 + G2 ## en dos columnas
```




```
G3 + G4 / G5 # grafico G3 en 1 columna y 2 filas, G4 y G5 en 1 fila y 1 columna
```



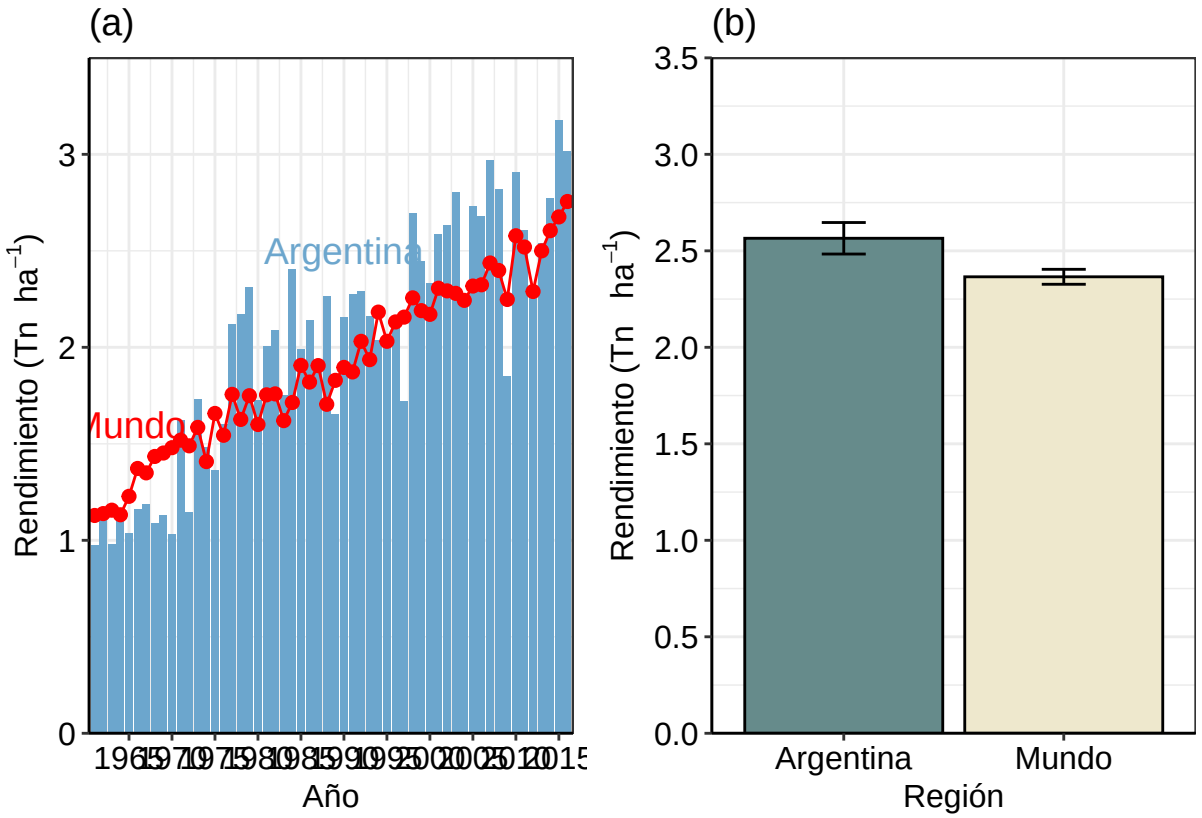

```
G7 / (G5 + G6) # G7 en 1 fila y 2 columnas, G5 y G6 en 1 fila y 1 columna
```



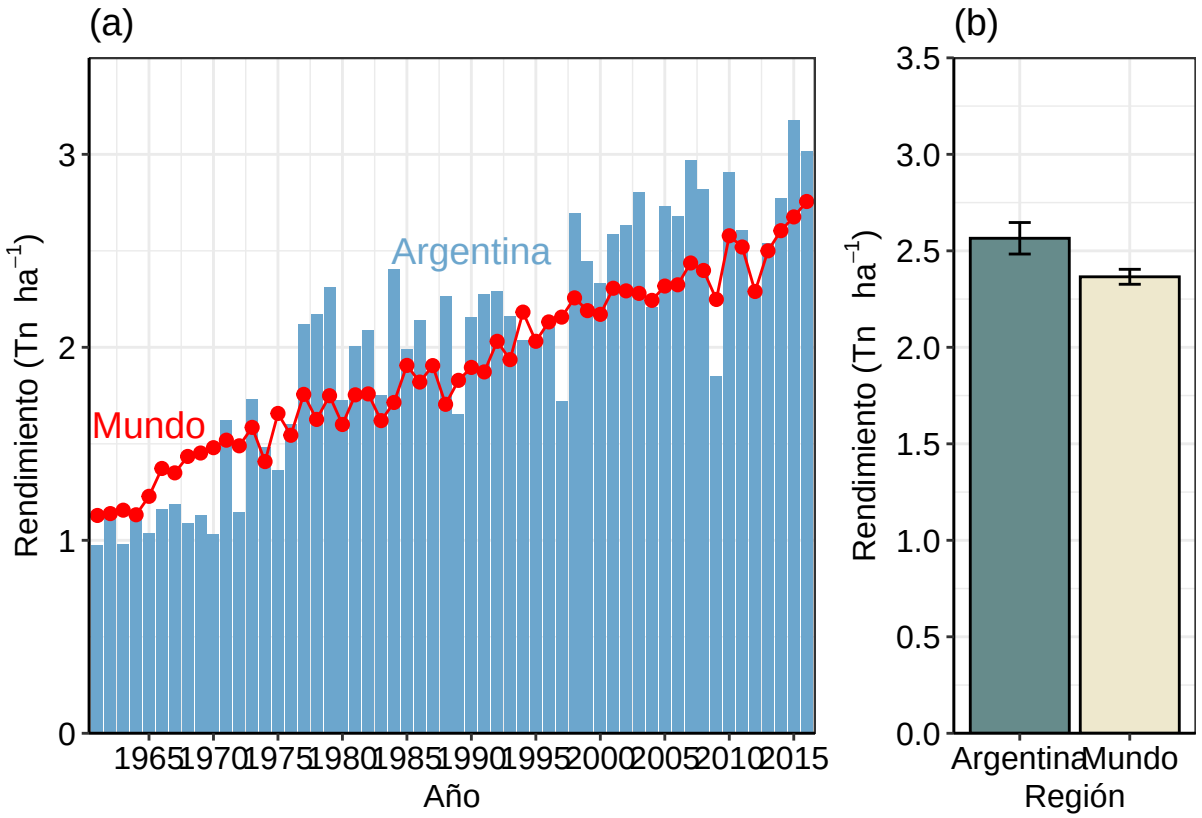
5.2. Dividiendo la pantalla gráfica y agregando referencia a los gráficos

Esto lo haremos utiliizando la función `wrap_plots` también del paquete **patchwork**. La ventana gráfica funciona como una combinación de filas y columnas que marcan la posición. En la representación de un solo gráfico hay 1 sola columna y 1 una sola fila que ocupa toda la ventana. En este caso vamos a usar dos gráficos distribuidos en dos columnas y una sola fila (uno al lado del otro). Si quisiéramos uno arriba de otro sería en 2 filas y una sola columna.

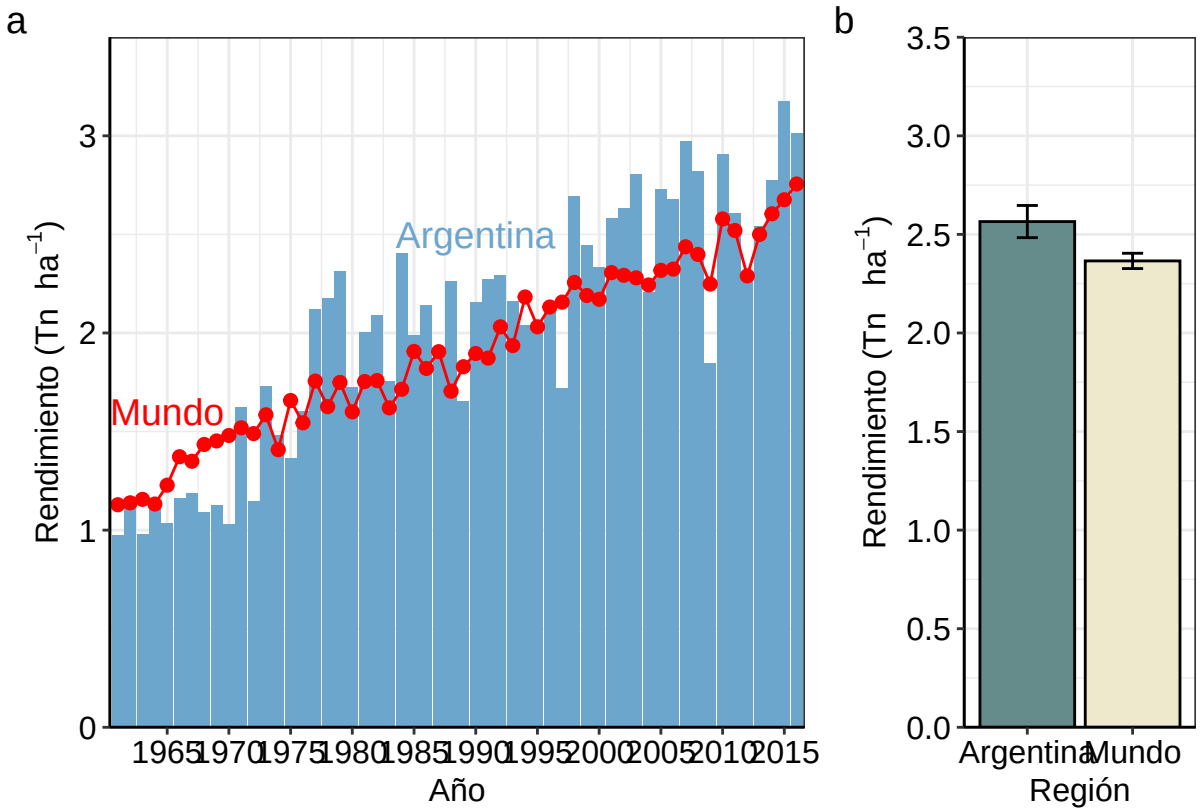
```
wrap_plots(G7 + ggtitle("(a)") +
             G6 + ggtitle("(b)")) #agregamos etiquetas manualmente
```



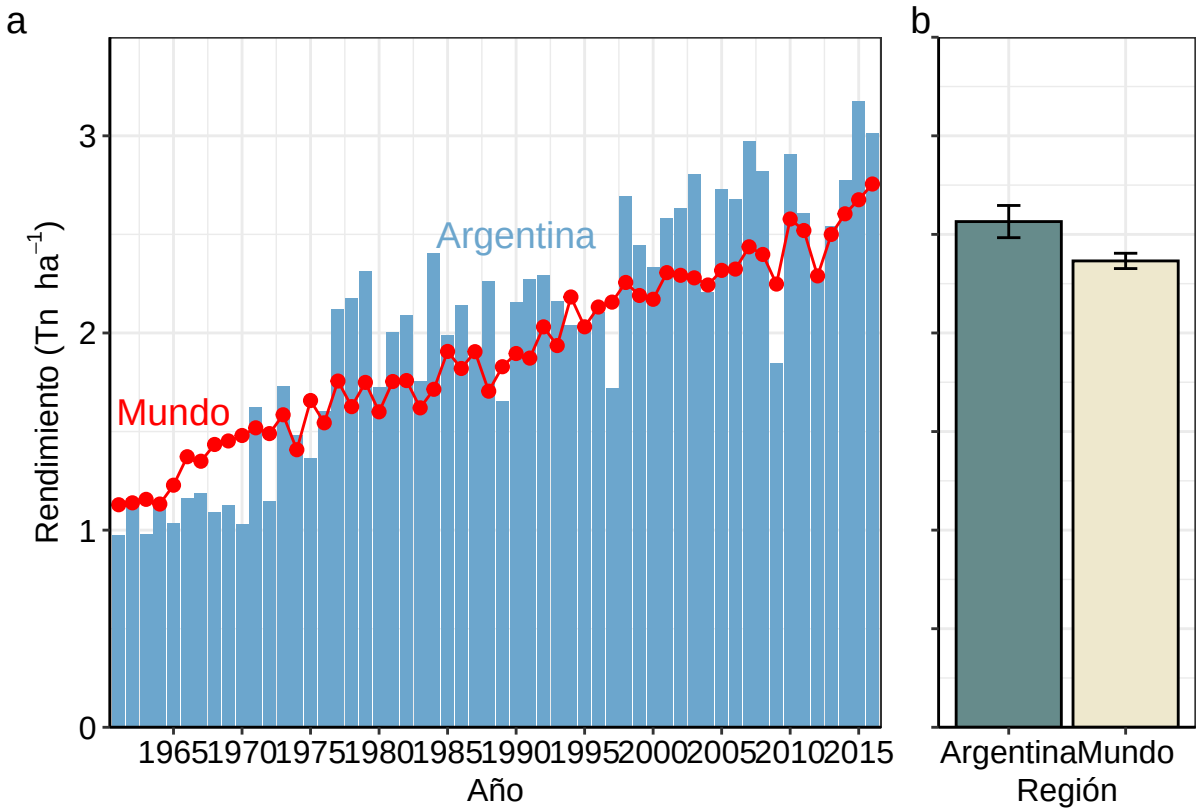

```
wrap_plots(G7 + ggtitle("(a)") ,
           G6 +  ggtitle("(b)"),
           ncol = 2,
           widths = c(3,1)) #relativizamos los anchos
```




```
wrap_plots(G7,
           G6,
           ncol = 2,
           widths = c(3,1)) &
  plot_annotation(tag_levels = 'a')  # etiquetas automáticas
```




```
G8 <- wrap_plots(G7,
           G6 +
             theme(axis.title.y = element_blank(), # eliminamos un eje repetido
                   axis.text.y = element_blank()),
           ncol = 2,
           widths = c(3,1)) &
  plot_annotation(tag_levels = 'a') ; G8
```



7. Exportando los gráficos

Los gráficos pueden ser exportados en varios formatos. Exploraremos algunos de ellos. Con cualquiera de las funciones que vamos a ver el archivo se escribirá en el misma carpeta que el directorio de trabajo (working directory), a no ser que le agreguemos ruta al archivo.

7.1. Utilizando *ggplot2*

Para exportar en diferentes formatos es siempre la misma función `ggsave`, lo que va a determinar el tipo de archivo es la extensión *.pdf*, *.png*, *.jpg*, etc. (la función no soporta *svg*). También se puede controlar el tamaño y los dpi, entre otras cosas. La función guarda el último gráfico corrido, por defecto.

```
ggsave("Gráfico 8.png") # exporta tal cual se ve en la ventana Plots
ggsave("Gráfico 8.pdf", width = 10, height = 8, dpi= 600) # esta es el que usaremos en inkscape
```

7.2. Utilizando R base

Aquí la función determina el formato (además del nombre), también se puede controlar tamaño (y la unidad de medida del mismo), pixeles, etc. Las funciones llevan el nombre de la extensión por ejemplo: `svg()` , `pdf()`, `png()`, etc. Aquí la diferencia es que es necesario poner el código del gráfico para crearlo (en este caso como ya lo tenemos guardado en un objeto, simplemente se llama al objeto **G8**) y cerrarlo con dev.off()

```
svg("Gráfico 8.svg", width = 12, height = 10)
G8
dev.off()
```

```
## pdf
```

```
##   2
```

8. Links recomendados para seguir explorando

Este taller pretende ser una mera aproximación al uso de R como motor gráfico mediante el paquete *ggplot2*. Esperamos despierte el interés en sus participantes para que a partir de aquí exploren el universo de posibilidades que se les ofrece. Para guiar esta búsqueda les recomendamos los siguientes enlaces que poseen ejemplos e información interesante:

Cheetsheets de Tidyverse

Son resúmenes con casi todas las funciones de cada paquete y una breve explicación de como funciona.

Referencia de ggplot2

Explicaciones básicas y ejemplos de todas las capas posibles con ejemplos y combinaciones. Es la página que se usa de referencia cuando no te acordás algo. Recomendamos explorarla es muy interesante.

Ejemplos de gráficos

La página esta organizada en secciones según objetivos a graficar (distribuciones, correlaciones, mapas, etc), dentro de ellas los diferentes posibles gráficos. Los gráficos estan hechos en ggplot2 pero también con R base. ¡Es buenísima!.

Workshop ggplot2 parte 1

Workshop ggplot2 parte 2

Taller de ggplot2 dado por Thomas Lin Pedersen uno de los desarolladores de Rstudio. Muy recomendable, explora cada función de ggplot2. En total es un poco más de 4 horas, no tiene desperdicio.

Patchwork

Describe los diferentes usos de Patchwork. Es la referencia para usarlo.

R Graphics for Cookbook

Guía práctica que provee diferentes "recetas" para ayudar a generar gráficos de calidad rápidamente. Utiliza ggplot2 como base.