Módulo 4. Gráficos avanzados con ggplot

Curso Herramientas para Data Science

Prof. Jose Jacobo Zubcoff Vallejo, PhD

Universidad de Alicante

Módulo 4. Gráficos avanzados con ggplot

Una de las fortalezas de R es la facilidad para incorporar paquetes. Algunos de ellos están especializados en algunos tipos de análisis y gráficos. Así, uno de los más conocidos y usados para gráficos avanzados es *ggplot2*.

Para instalar el paquete ggplot2 (ya vimos en el módulo 1 como instalar y cargar paquetes), podemos usar los comandos de la siguiente manera:

```
#install.packages("ggplot2") #1- Instalar el paquete ggplot2
library(ggplot2)             #2- Cargar el paquete en memoria
```

Diagramas con ggplot2

Cuando usamos ggplot2 para hacer gráficos, en realidad lo que hacemos es crear un contenedor, al que le tendremos que incorporar la información de:

- los datos que vamos a usar
- los ejes que se van a representar
- y una capa por cada gráfico que se quiera dibujar en esos ejes

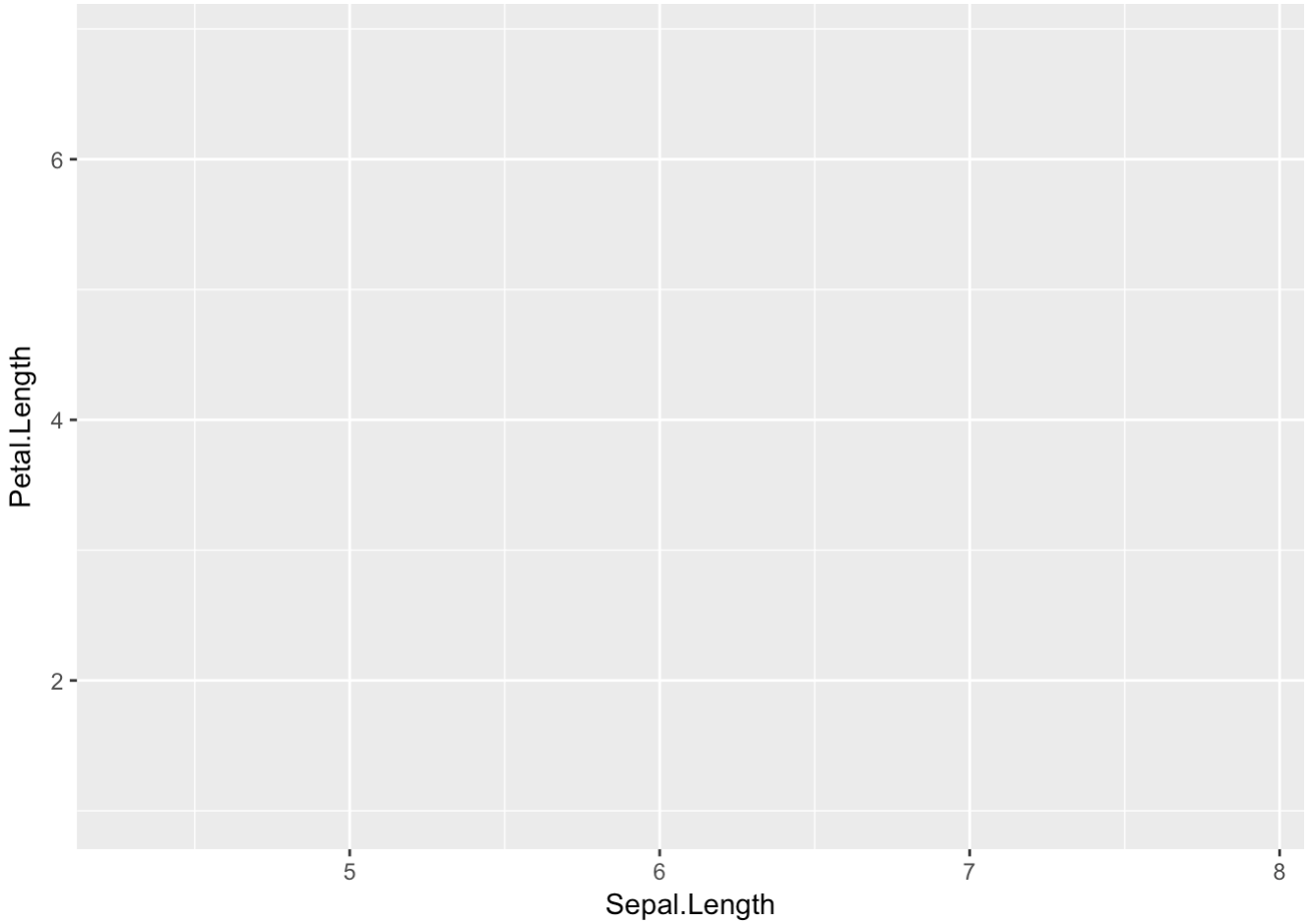
Así, para cada diagrama que se dibuje con *ggplot2* debemos definir al menos:

- *data=* es obligatoriamente un conjunto de datos o “data.frame”
- *aes()* es la configuración de los ejes
- y la capa de gráficos propiamente dicha

Por ejemplo, para el conjunto de datos denominado “iris” (que contiene datos sobre 3 especies de esta planta): iris setosa, iris versicolor e iris virginica. En este caso, representaremos la totalidad de los datos buscando un patrón entre “Sepal.Length” y “Petal.Length”.

Empezaremos creando el contenedor:

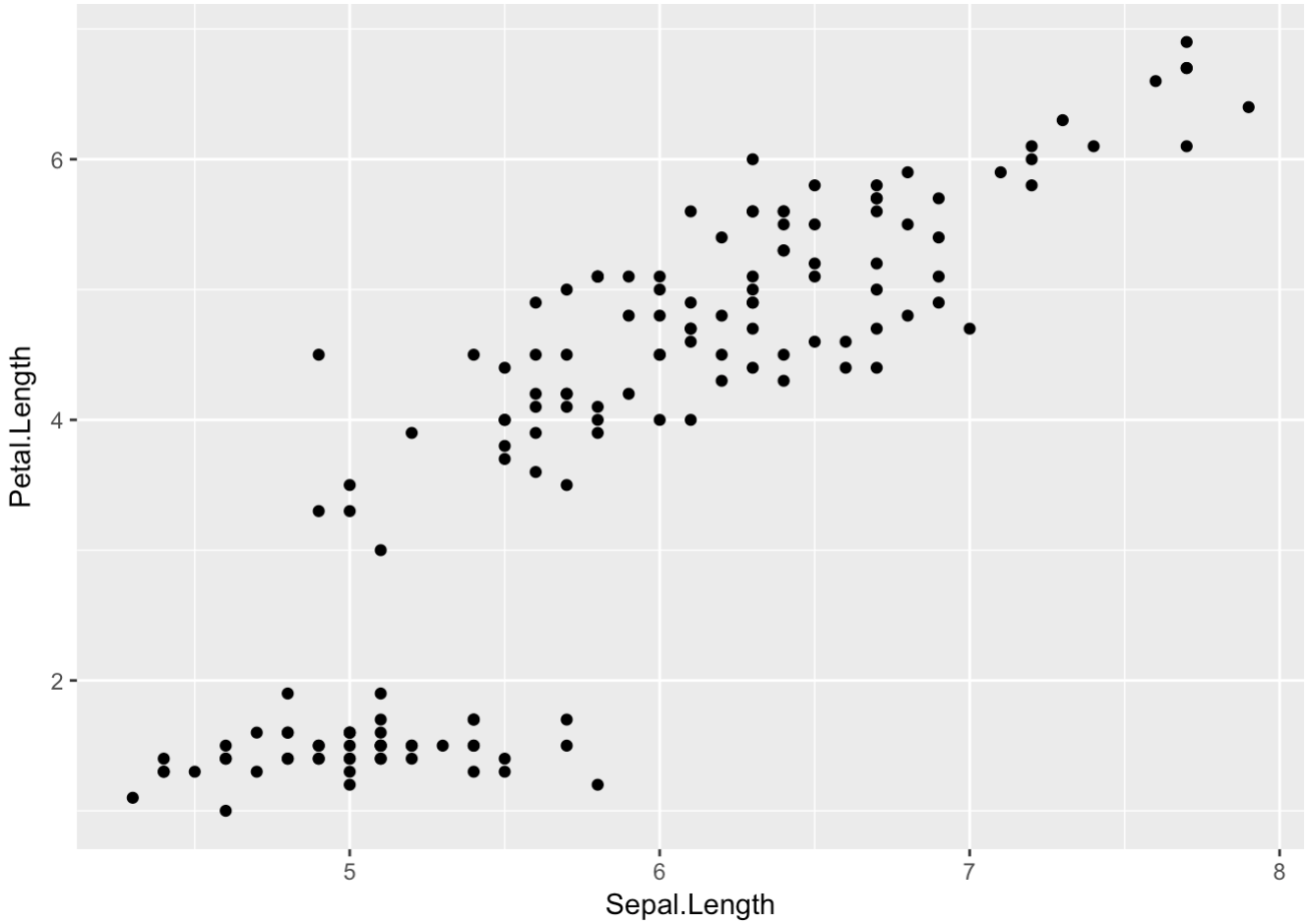
```
ggplot(data=iris, aes(Sepal.Length, Petal.Length))
```



Como se puede observar, hemos creado el “contendor” del gráfico, con sus ejes, y etiqueta los ejes. Cabe destacar que el “contenedor” del gráfico averigua el rango que necesita cada eje para representar los datos. Pero falta lo principal, ¿dónde están los puntos?¿o las líneas?¿o el área?. La respuesta es que aún no la hemos dibujado. Está todo preparado pero falta pedir que tipo de gráfico queremos hacer.

Vamos a probar con un nube de puntos o “scatterplot”:

```
ggplot(data=iris, aes(Sepal.Length, Petal.Length)) + geom_point()
```



Ahora prbaremos a usar líneas (aunque no es la mejor opción en este caso):

```
ggplot(data=iris, aes(Sepal.Length, Petal.Length)) + geom_line()
```

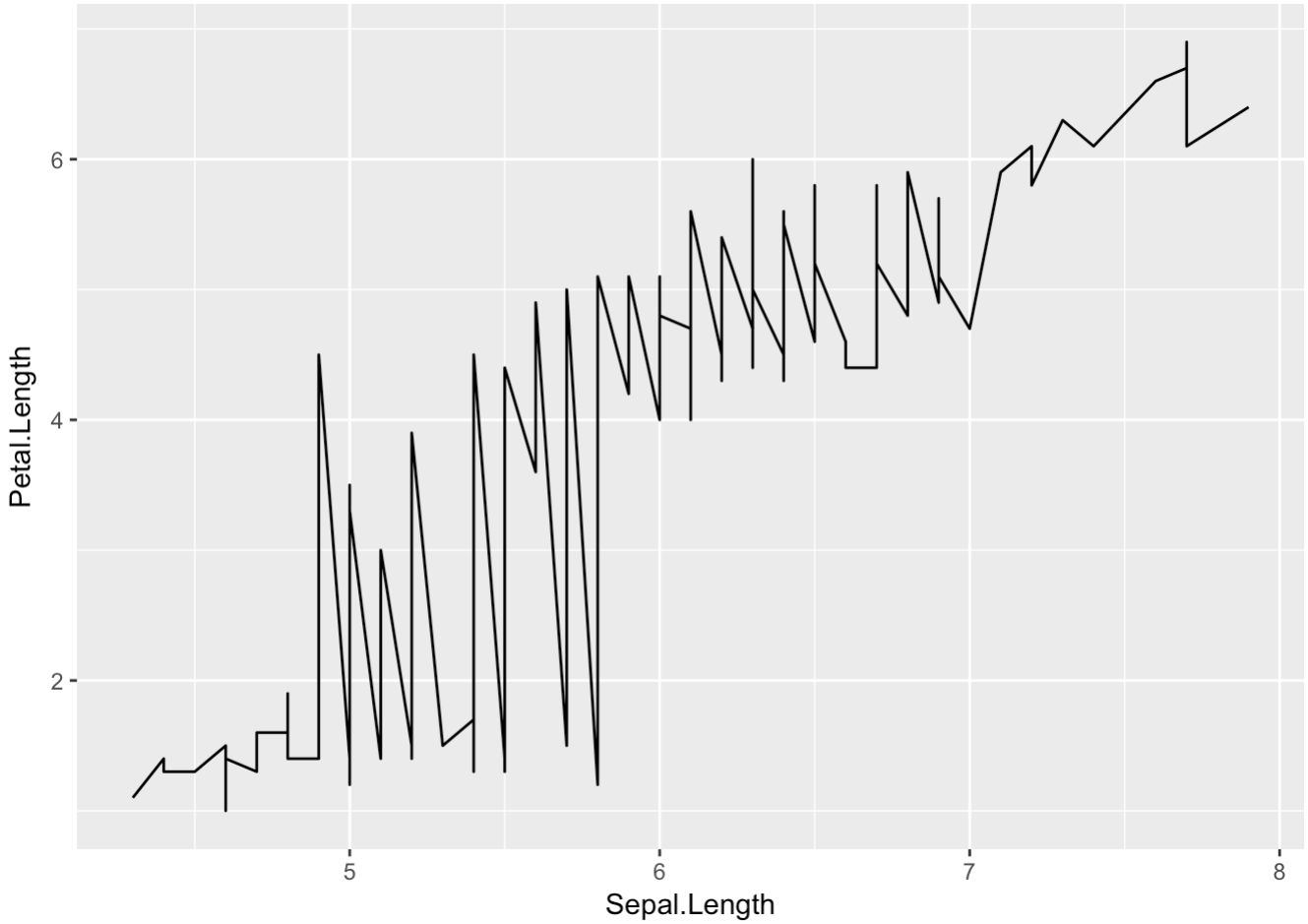
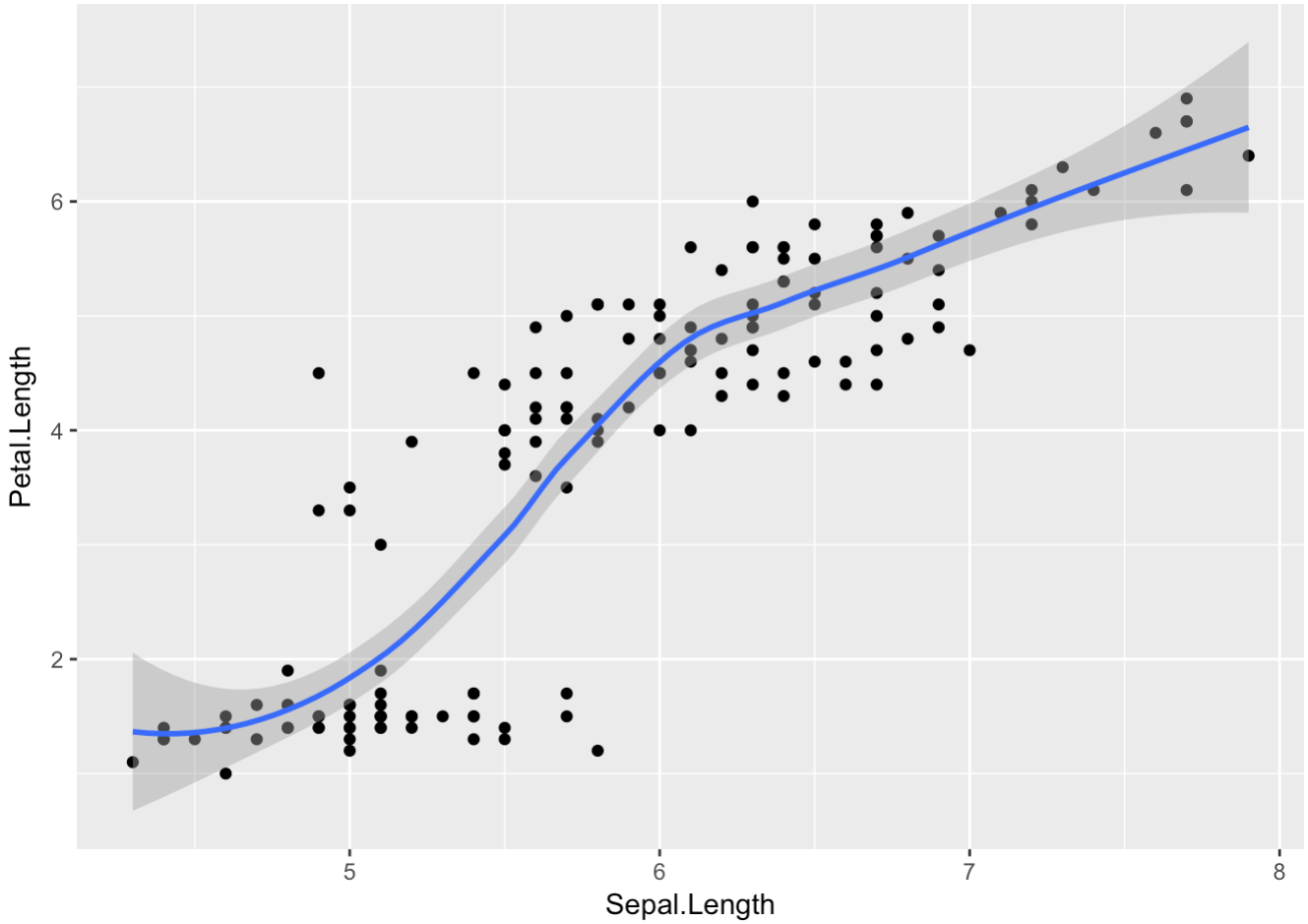


Diagrama de puntos avanzado

Con la librería *ggplot2* podemos mejorar fácilmente la representación gráfica. Al principio puede parecer mas tedioso tener que abrir un contenedor de gráficos y luego añadir el diagrama que se desee, pero esto a su vez es útil cuando se quieren usar algunas representaciones comunes. Por ejemplo, si queremos agregar un sombreado basado en cálculos estadísticos de funciones de suavizado, podemos hacerlo simplemente agregando “stat_smooth()”.

```
ggplot(data=iris, aes(Sepal.Length, Petal.Length)) + geom_point() + stat_smooth()
```

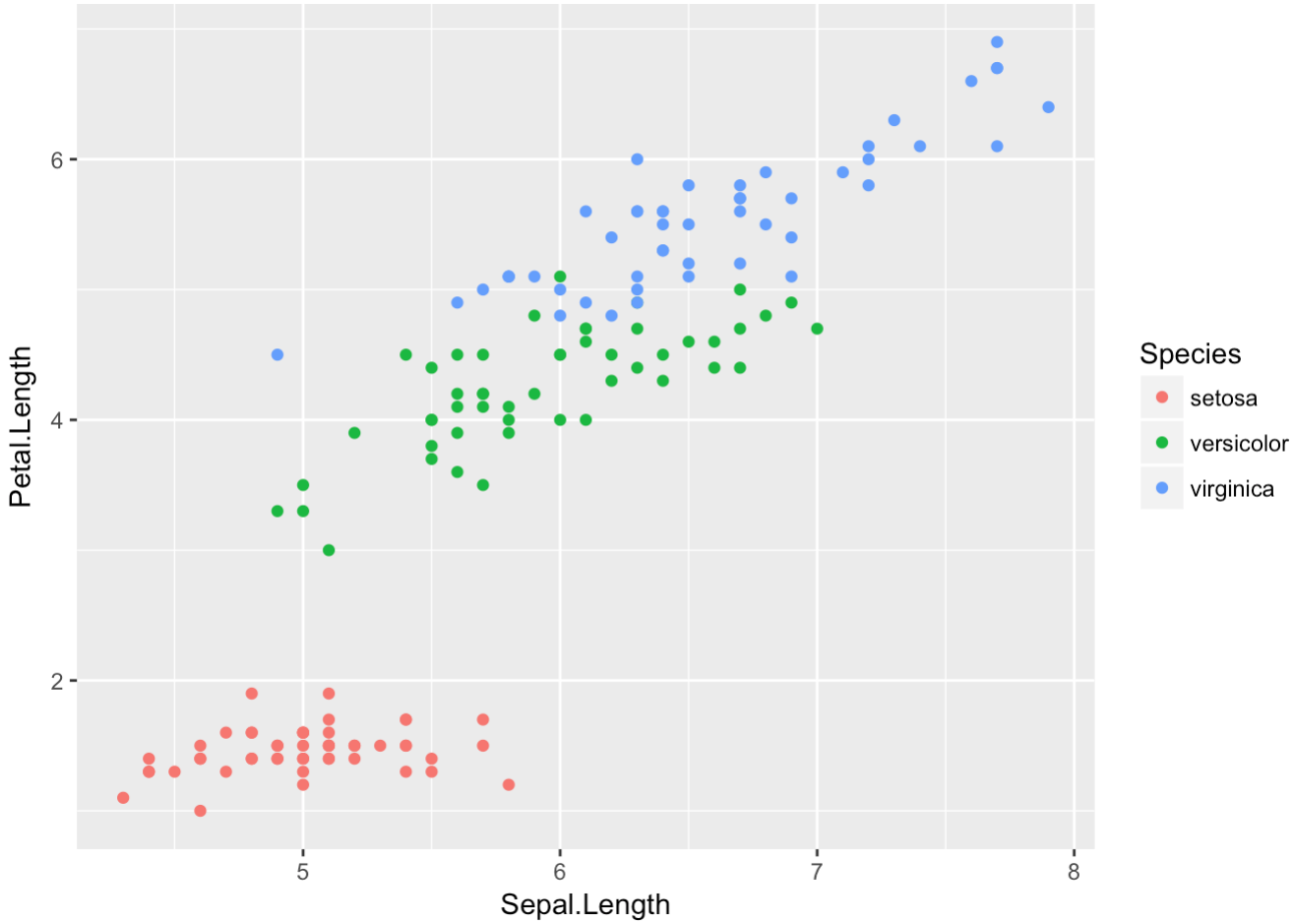
```
## `geom_smooth()` using method = 'loess'
```



Agrupar elementos por colores

Podemos pintar los puntos con distintos colores, según la especie por ejemplo.

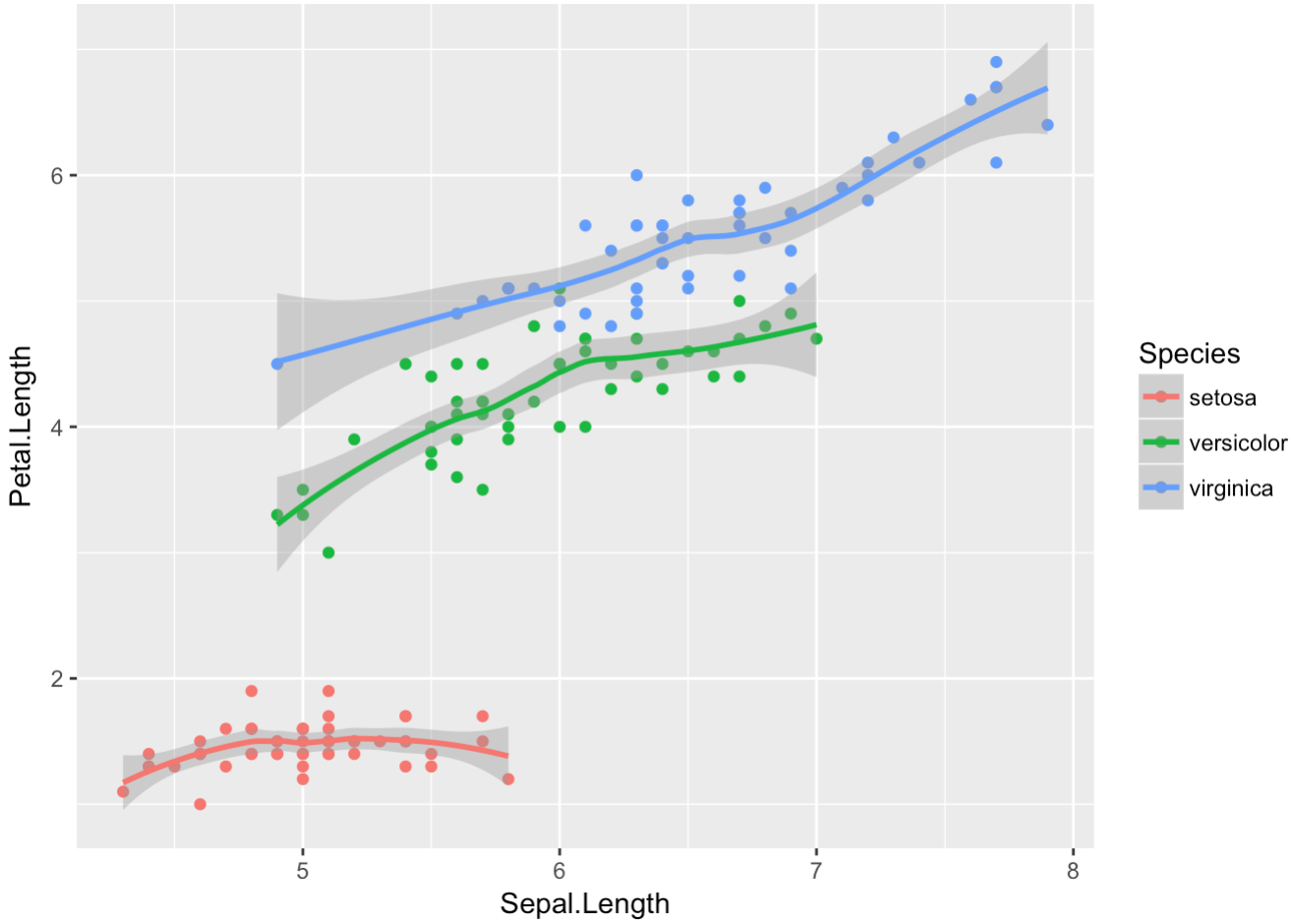
```
ggplot(data=iris, aes(Sepal.Length, Petal.Length,color=Species)) + geom_point()
```



Ahora le agregaremos la líneas de suavizado, con sus bandas de confianza, también por especies:

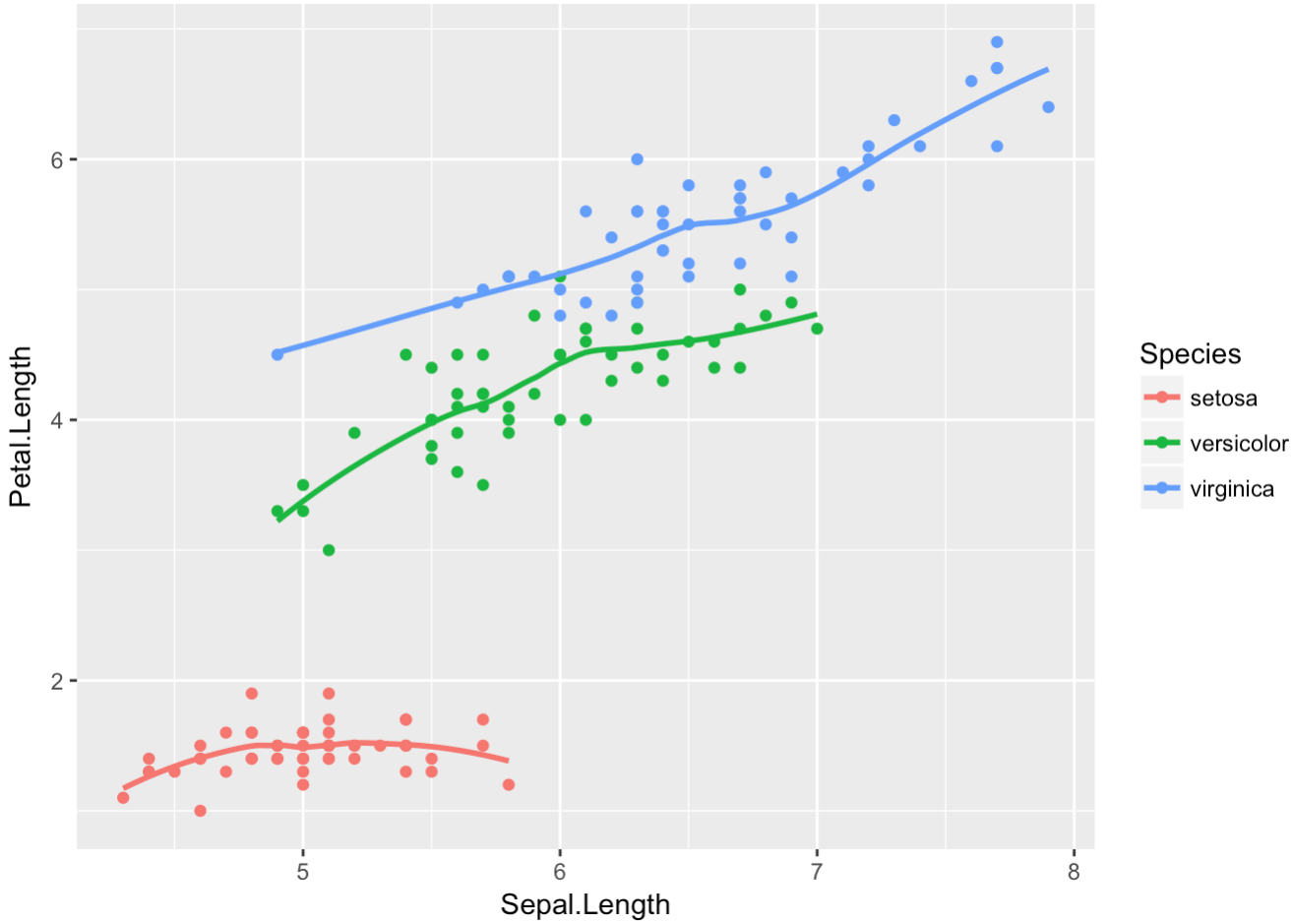
```
# Creamos el contenedor de ggplot
ggplot(data=iris, aes(Sepal.Length, Petal.Length,color=Species)) +
  geom_point() +      # los puntos
  stat_smooth()       # líneas y bandas de suavizado (smooth)
```

```
## `geom_smooth()` using method = 'loess'
```




```
# Ahora sin banda de confianza
# Creamos el contenedor de ggplot
ggplot(data=iris, aes(Sepal.Length, Petal.Length,color=Species)) +
  geom_point() +      # los puntos
  stat_smooth(se=FALSE)       # líneas y bandas de suavizado (smooth)
```

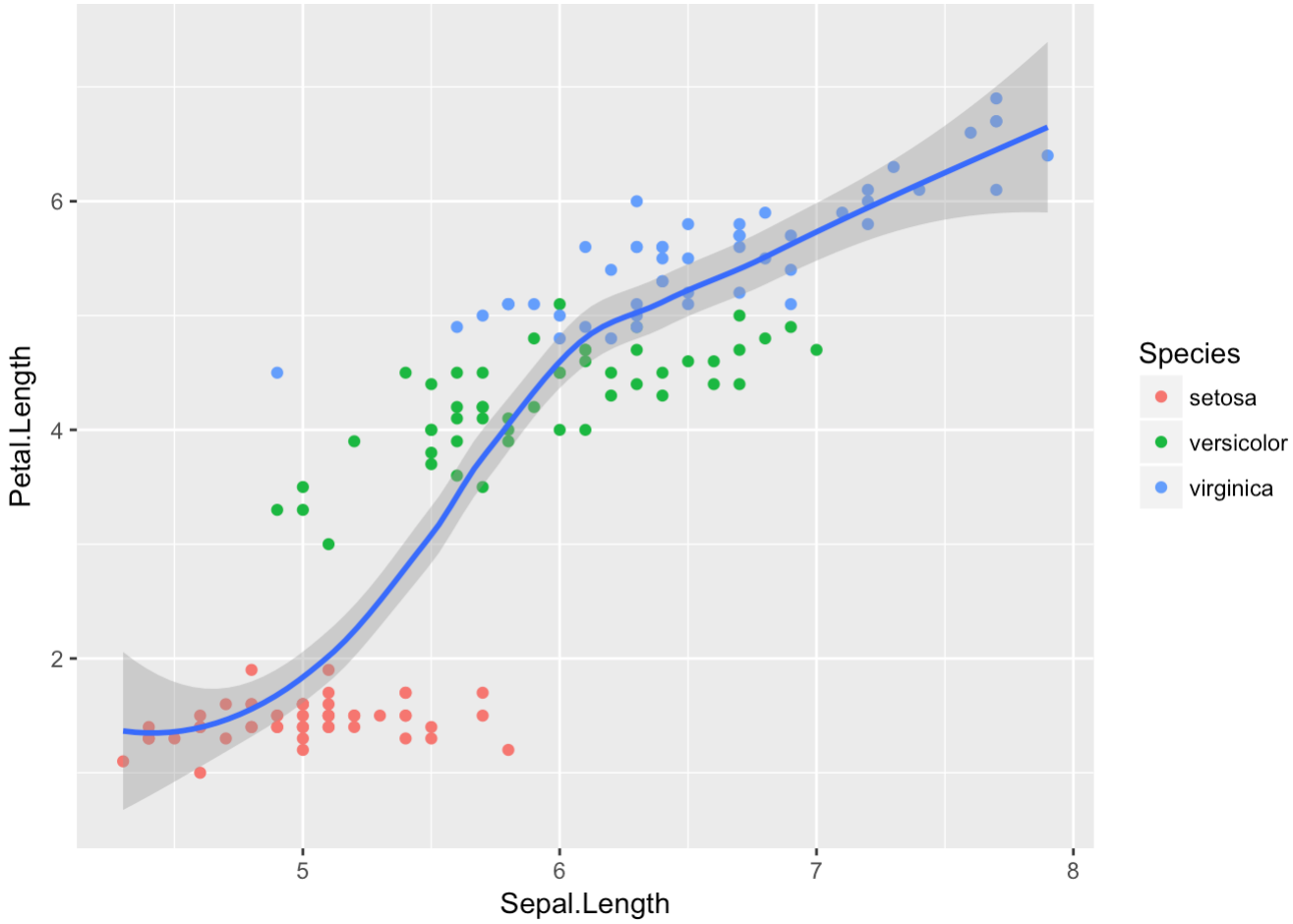
```
## `geom_smooth()` using method = 'loess'
```



Si queremos que los puntos tengan la información de especie (coloreados por especie por ejemplo) pero queremos mostrar la tendencia global (sin agrupar por especie), entonces sería así:

```
# Creamos el contenedor de ggplot
ggplot(data=iris) +
  geom_point( aes(x=Sepal.Length, y=Petal.Length,color=Species)) +      # SOLO los puntos coloreados por especie
  stat_smooth(aes(x=Sepal.Length, y=Petal.Length))       # líneas y bandas de suavizado (smooth) sin colorear por especie
```

```
## `geom_smooth()` using method = 'loess'
```



Agrupar elementos por tamaños

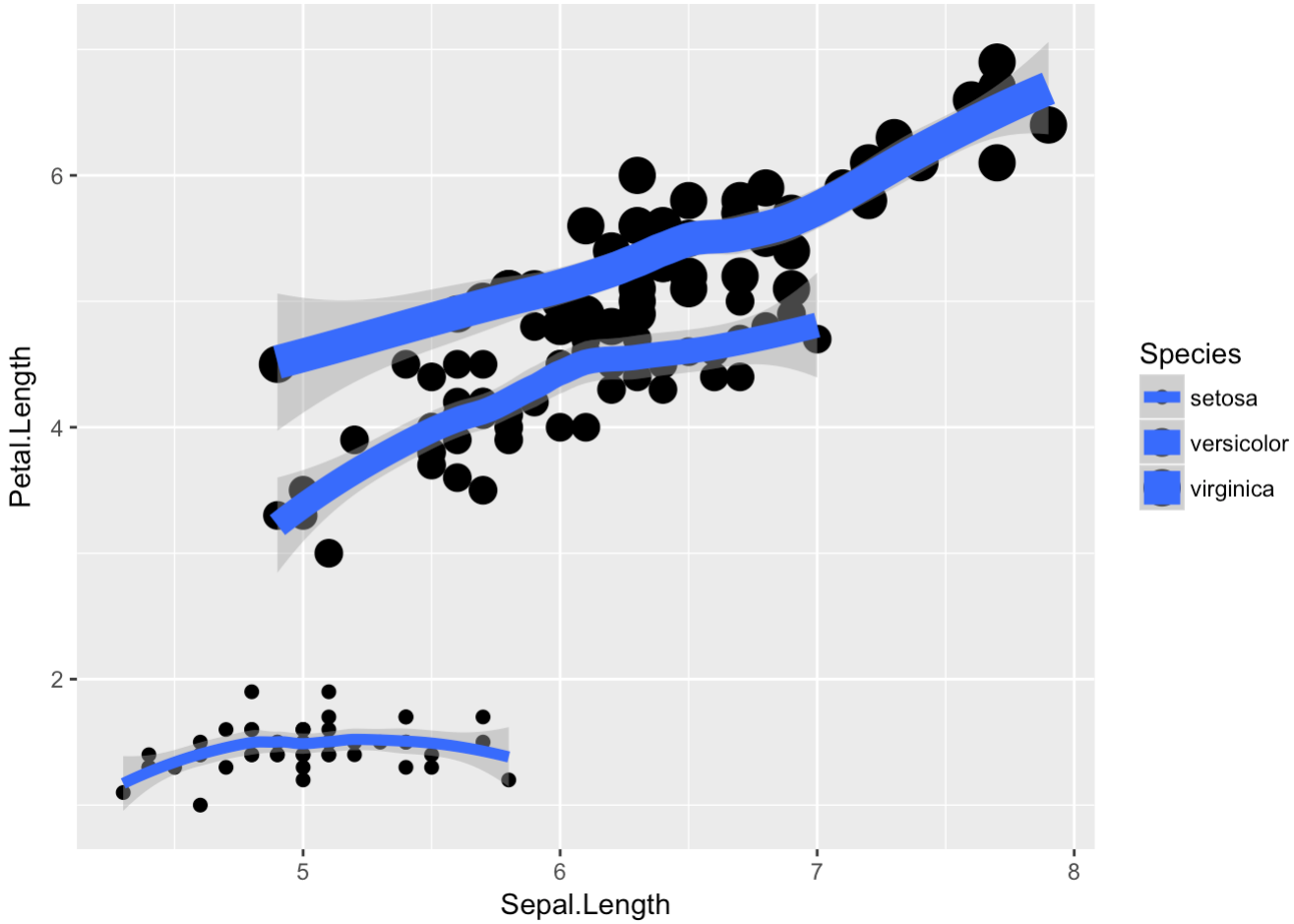
No es recomendable usar tamaño para representar variables discretas. En general es frecuente usar el tamaño para representar variables continuas.

Para poner un (mal) ejemplo de usar el tamaño para representar variables discretas (en este caso la especie), veamos:

```
# Creamos el contenedor de ggplot
ggplot(data=iris, aes(Sepal.Length, Petal.Length,size=Species)) +
  geom_point() +      # los puntos
  stat_smooth()       # líneas y bandas de suavizado (smooth)
```

```
## Warning: Using size for a discrete variable is not advised.
```

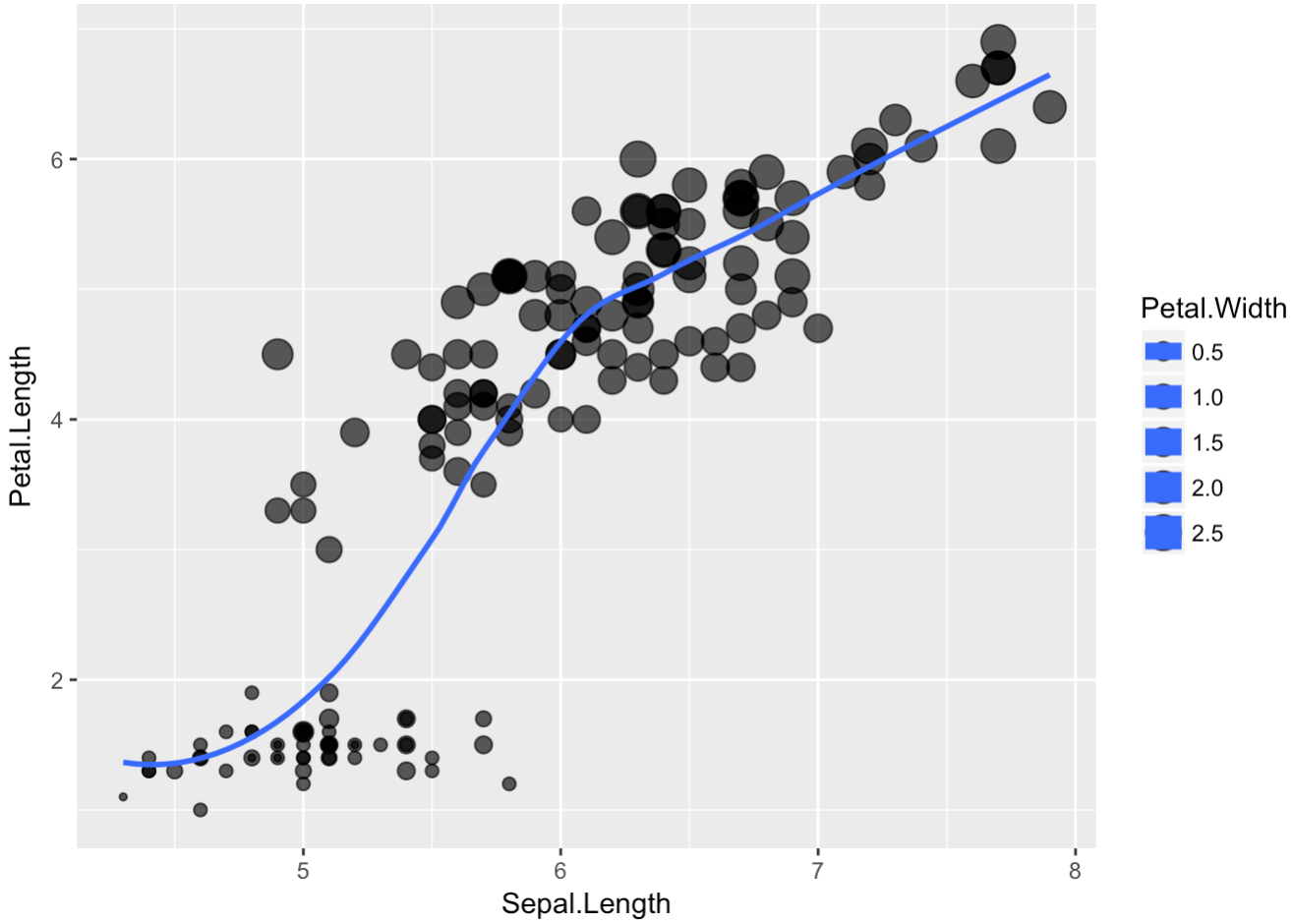
```
## `geom_smooth()` using method = 'loess'
```



No queda nada bien ni es nada claro. Por tanto, no es recomendable para representar variables discretas, pero sí para continuas. Por ejemplo:

```
# Creamos el contenedor de ggplot (usaremos el Petal.Width para el tamaño)
ggplot(data=iris, aes(Sepal.Length, Petal.Length,size=Petal.Width )) +
  geom_point(alpha = I(0.7)) +      # los puntos ahora con transparencia (alpha)
  stat_smooth(se=F)       # líneas y bandas de suavizado (smooth)
```

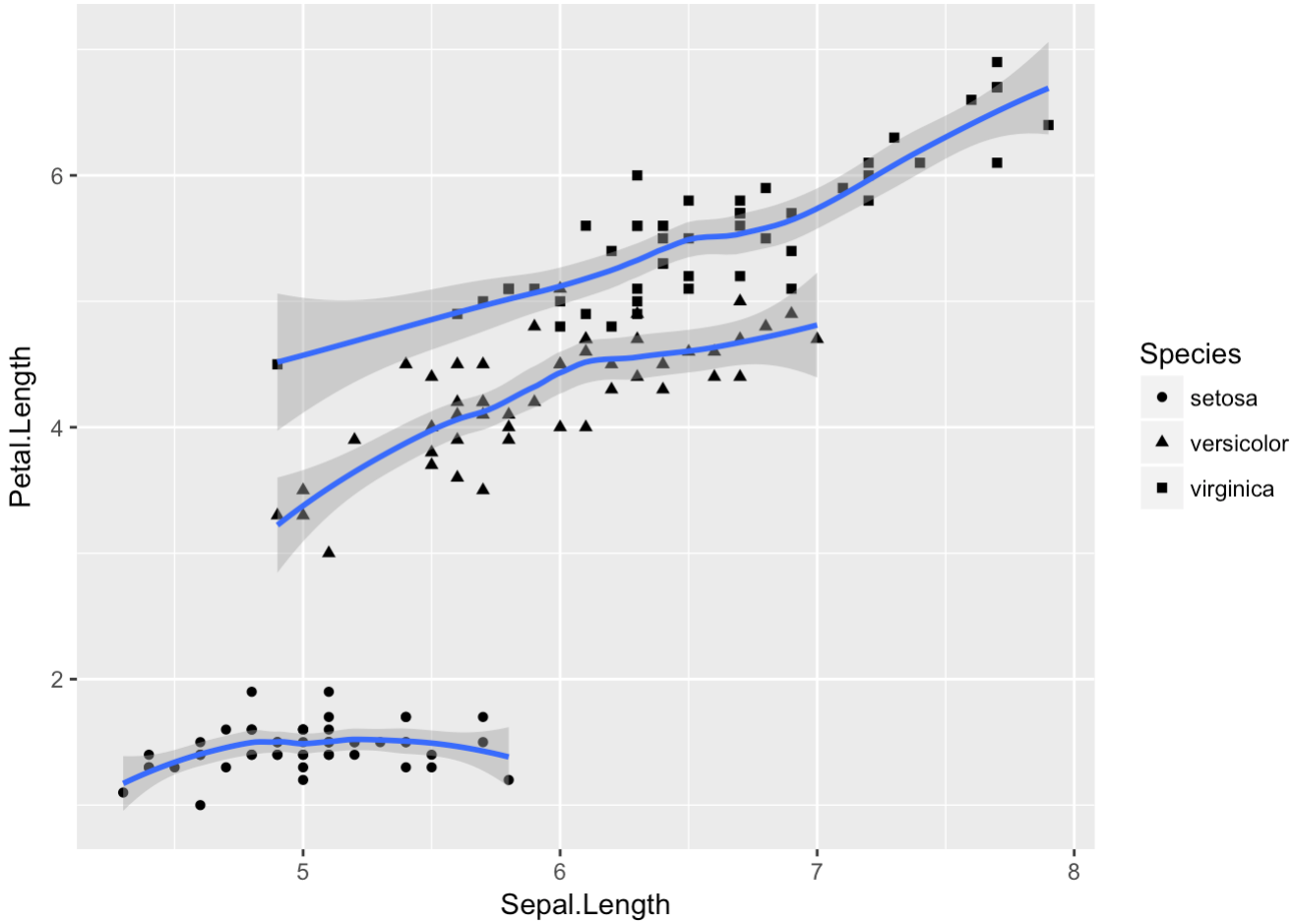
```
## `geom_smooth()` using method = 'loess'
```



Agrupar elementos por formas

```
# Creamos el contenedor de ggplot
ggplot(data=iris, aes(Sepal.Length, Petal.Length,shape=Species)) +
  geom_point() +      # los puntos
  stat_smooth()       # líneas y bandas de suavizado (smooth)
```

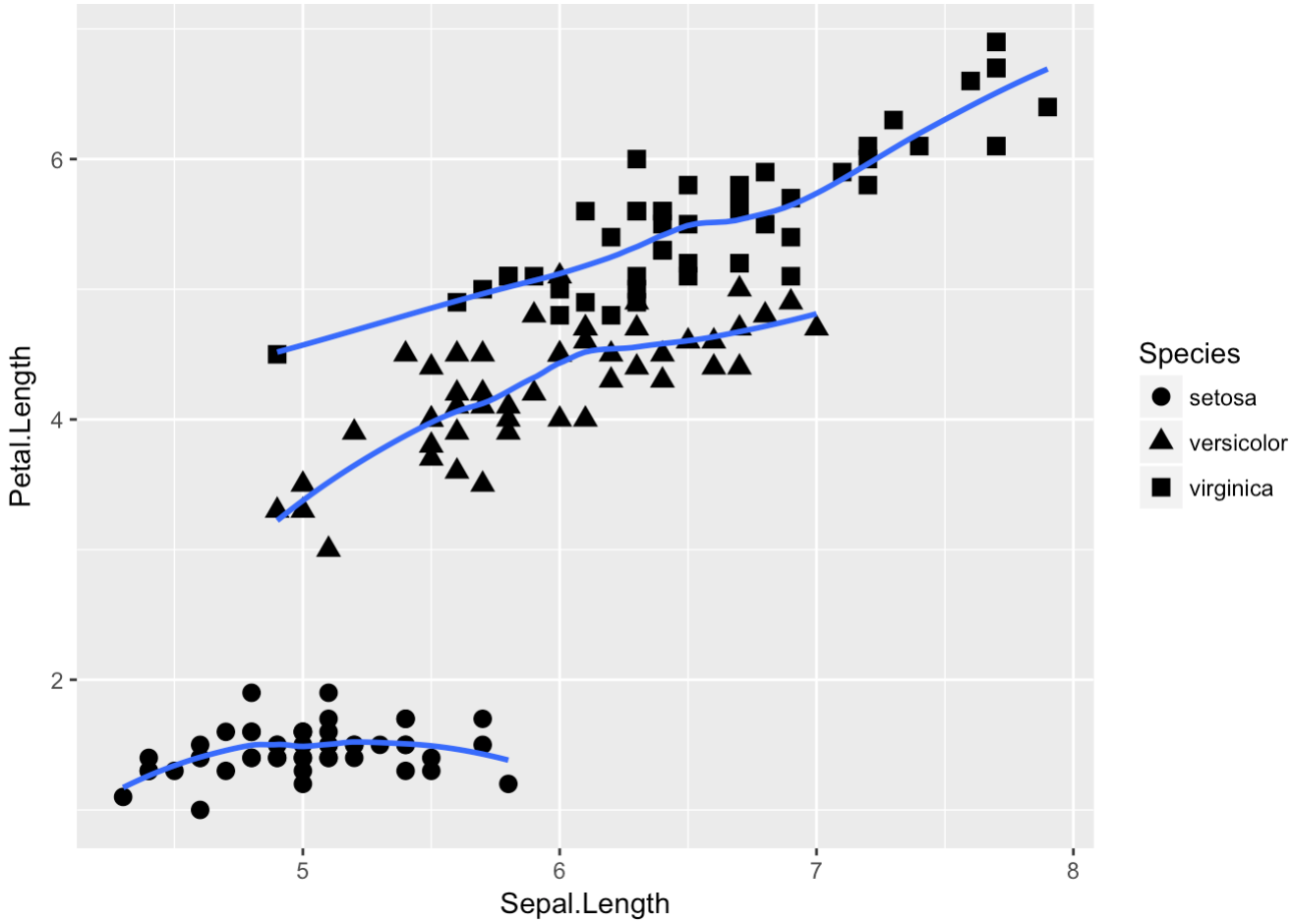
```
## `geom_smooth()` using method = 'loess'
```



Si queremos cambiar el tamaño, podemos usar “size=” dentro de “geom_point()” para que sea mas visible. Por ejemplo:

```
# Creamos el contenedor de ggplot
ggplot(data=iris, aes(Sepal.Length, Petal.Length,shape=Species)) +
  geom_point(size=3) +      # los puntos
  stat_smooth(se=F)       # líneas y bandas de suavizado (smooth)
```

```
## `geom_smooth()` using method = 'loess'
```



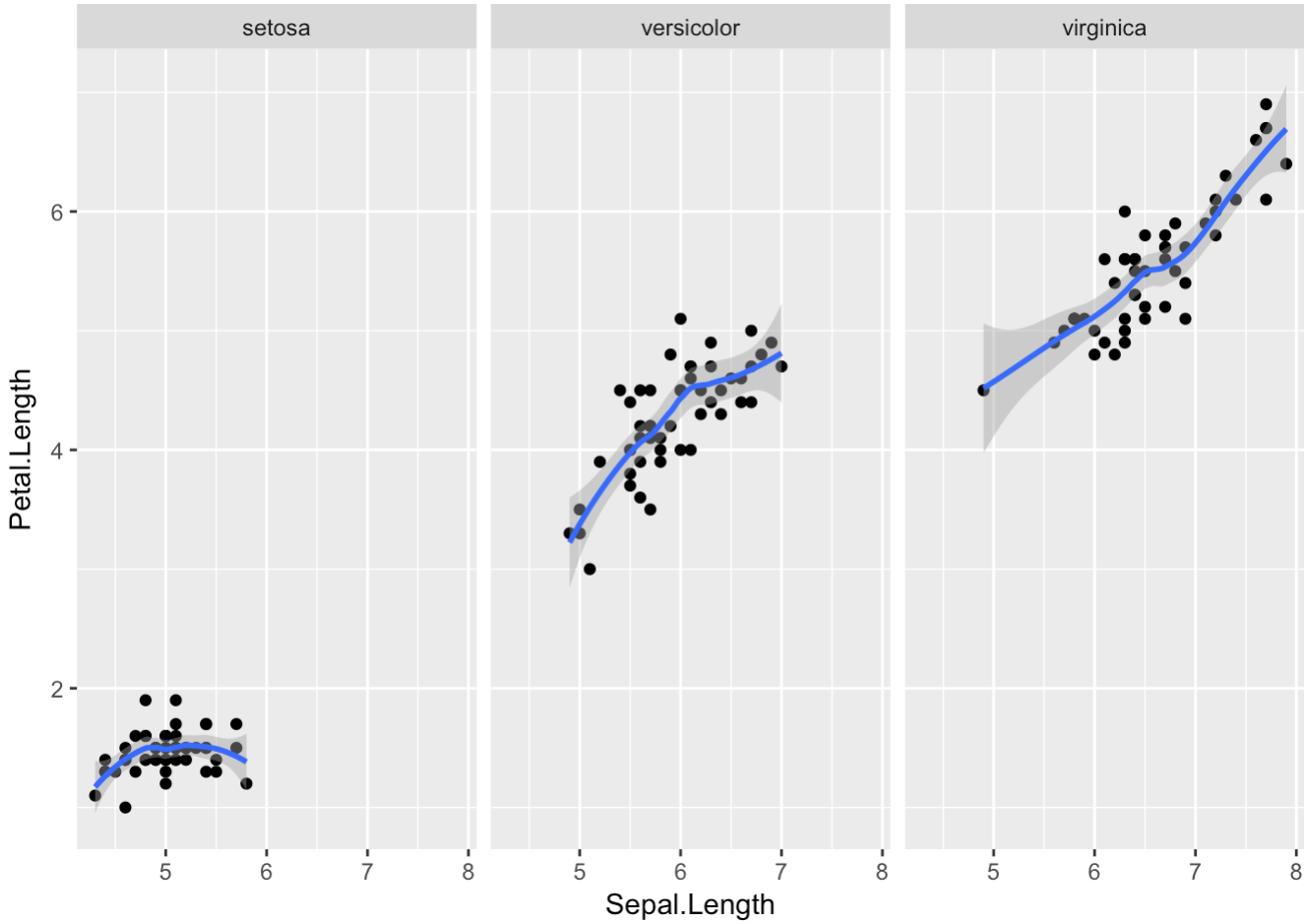
Agrupar elementos:

Facets: como dibujar varios gráficos en un mismo contenedor

Se puede dibujar cada especie en un gráfico distinto. Cuando hay mucha información en un sólo gráfico (muchos puntos, etc.) resulta útil para comparar entre niveles de un factor, separar por ejemplo, entre especies.

```
# Creamos el contenedor de ggplot
ggplot(data=iris, aes(Sepal.Length, Petal.Length)) +
  geom_point() +          # los puntos
  stat_smooth() +         # líneas y bandas de suavizado (smooth)
  facet_wrap(~ Species)  # las especies van en gráficos distintos
```

```
## `geom_smooth()` using method = 'loess'
```

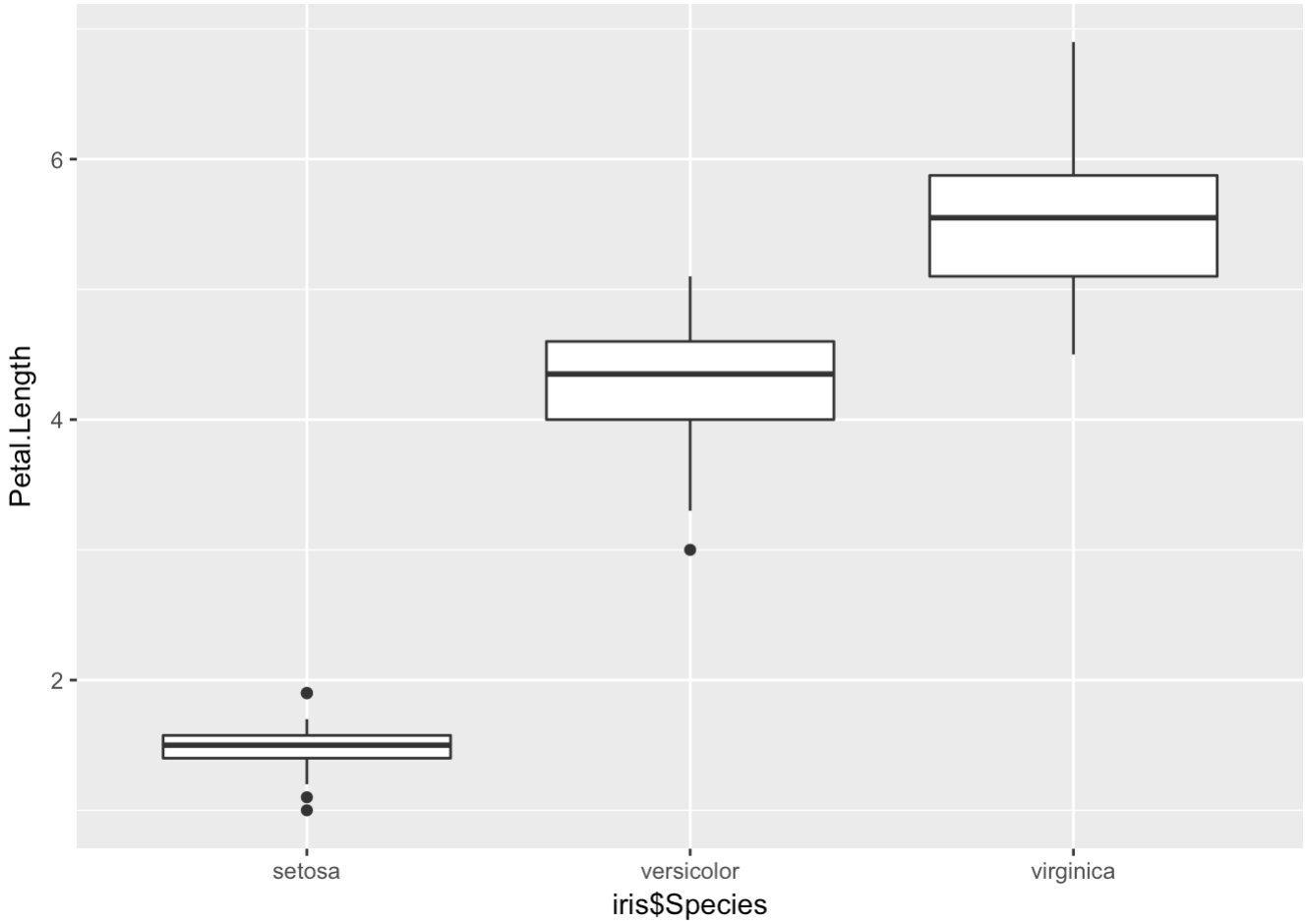


Boxplot o diagramas de cajas

La librería *ggplot2* permite hacer diagramas de cajas o boxplots.

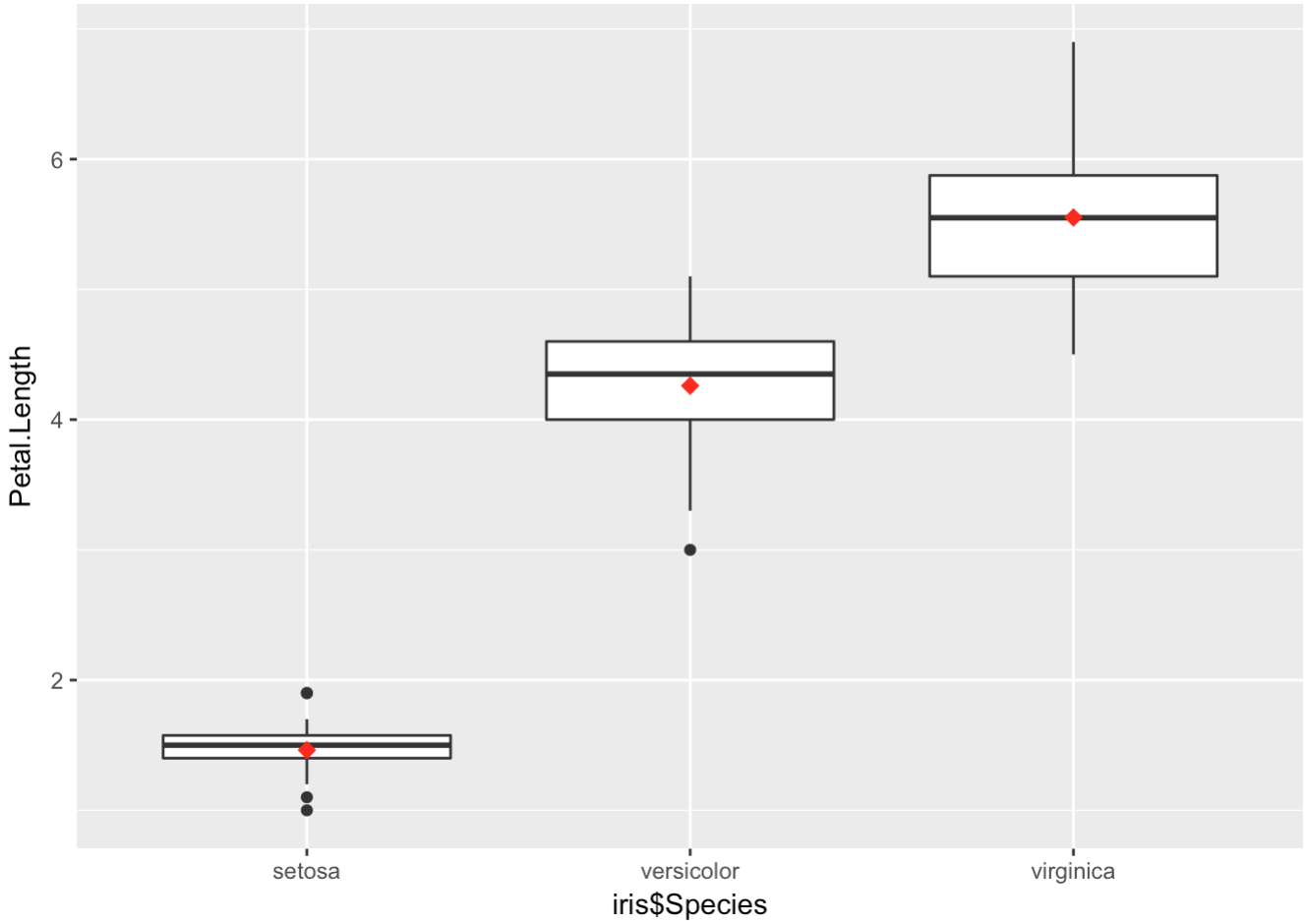
Empezaremos por uno básico, que luego se puede personalizar.

```
#Creamos el contenedor
ggplot(data=iris, aes(iris$Species, Petal.Length)) +
  geom_boxplot()    # dibujamos el diagrama de cajas
```



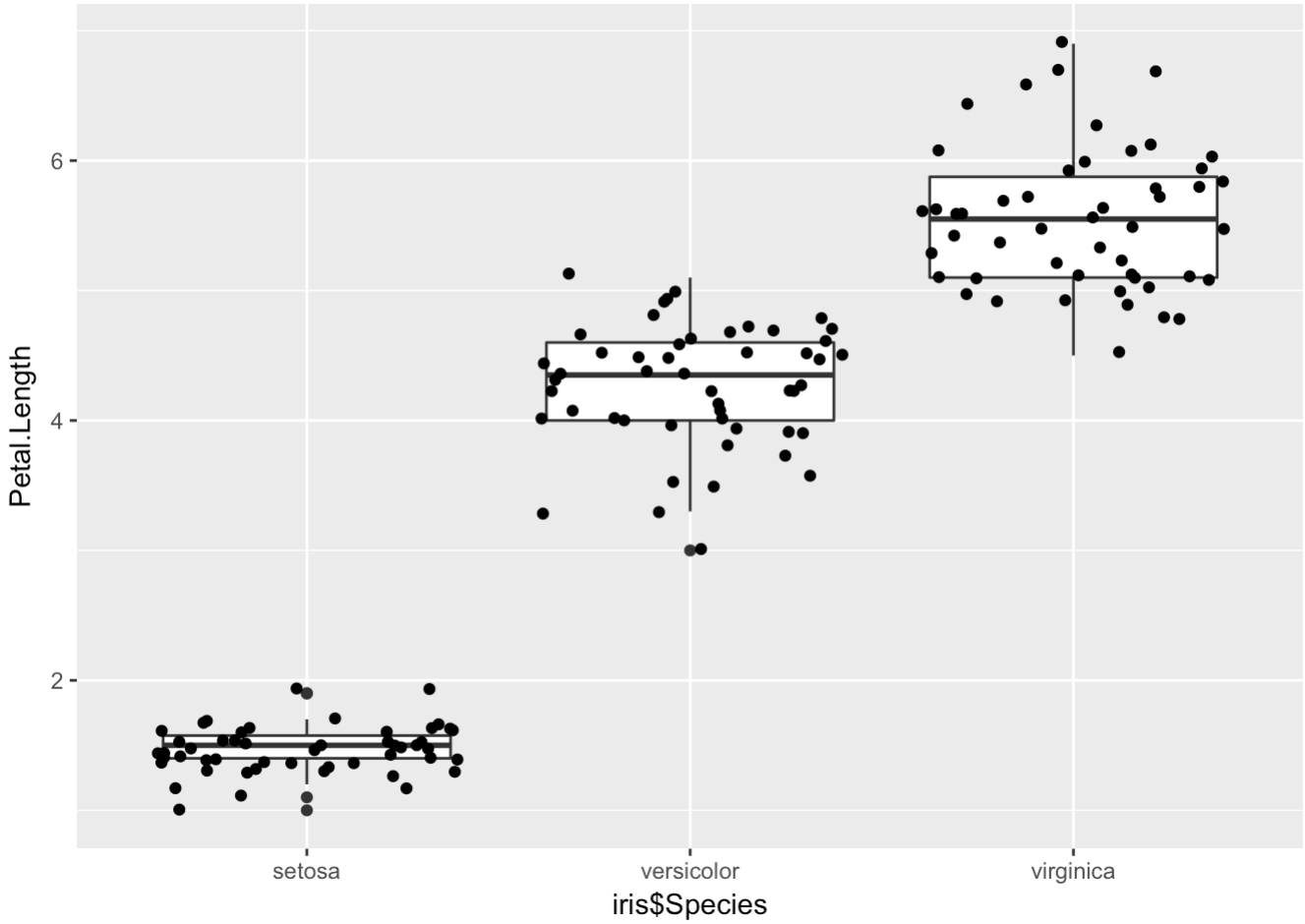
Ahora vamos a añadirle las medias calculadas por grupo.

```
#Creamos el contenedor
ggplot(data=iris, aes(iris$Species, Petal.Length)) +
  geom_boxplot()  +  # dibujamos el diagrama de cajas
  stat_summary(fun.y=mean, geom="point", shape=18,
                size=3, color="red")
```



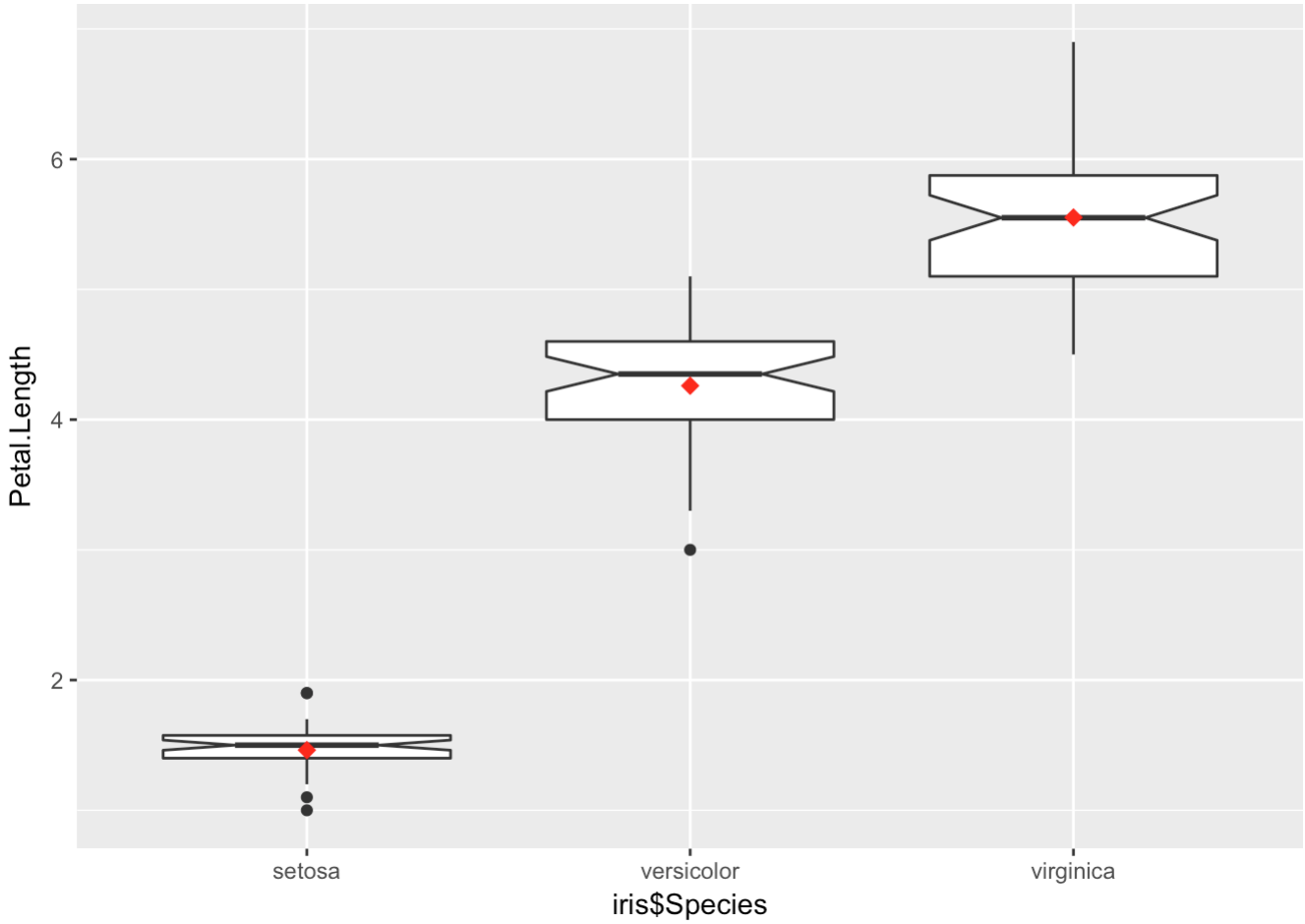
Podemos añadirle los puntos observados, con un poco de desplazamiento (jitter). Así:

```
#Creamos el contenedor
ggplot(data=iris, aes(iris$Species, Petal.Length)) +
  geom_boxplot()  + # dibujamos el diagrama de cajas
  geom_jitter()     #pinta los puntos con pequeño desplazamiento
```



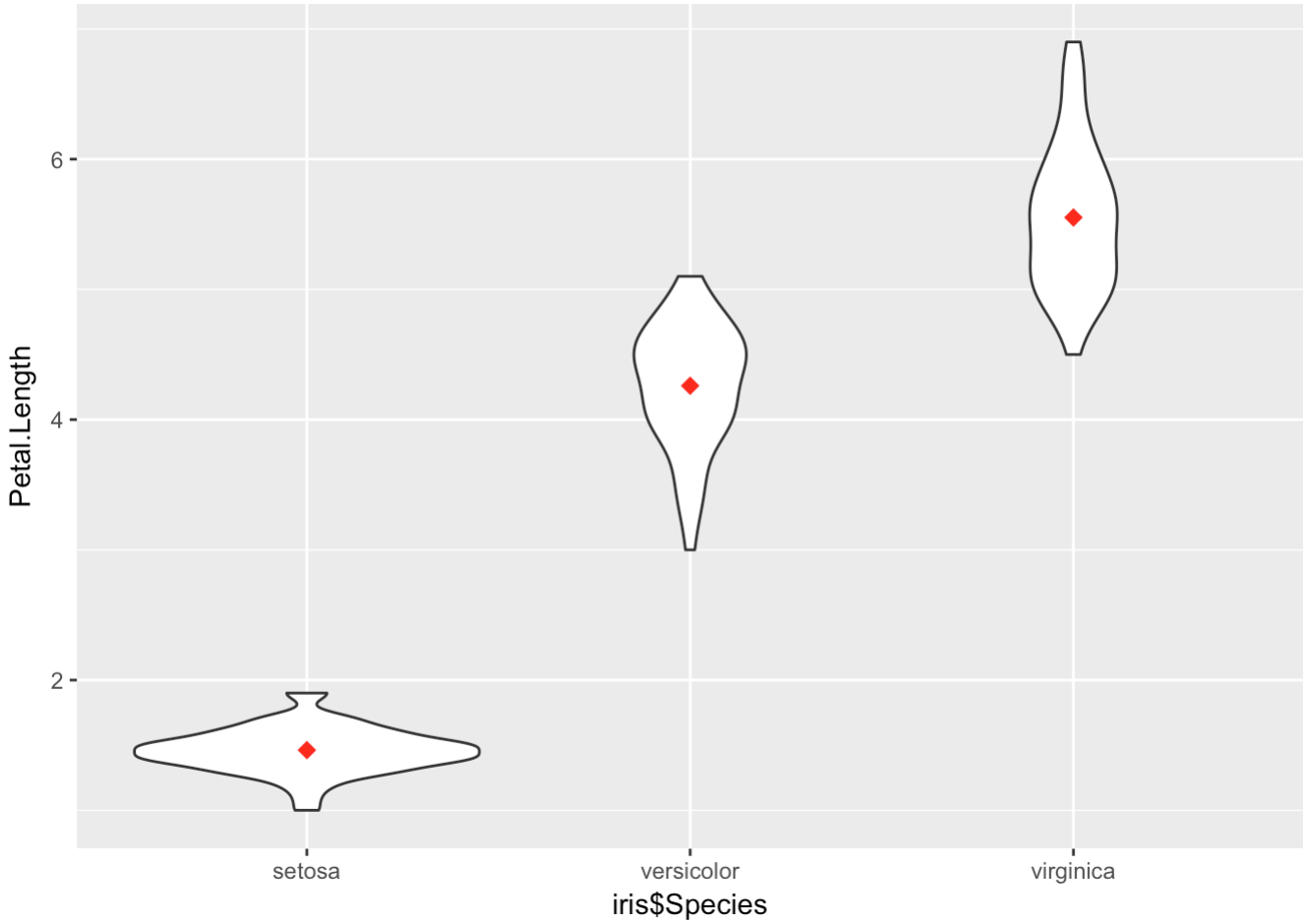
Para ver información acerca de la variabilidad entorno a la media podemos usar el siguiente diagrama:

```
#Creamos el contenedor
ggplot(data=iris, aes(iris$Species, Petal.Length)) +
  geom_boxplot(notch = TRUE)  +  # dibujamos el diagrama de cajas
  stat_summary(fun.y=mean, geom="point", shape=18,
                size=3, color="red")  #pintamos la media con un punto ROJO
```



O un diagrama de tipo violín:

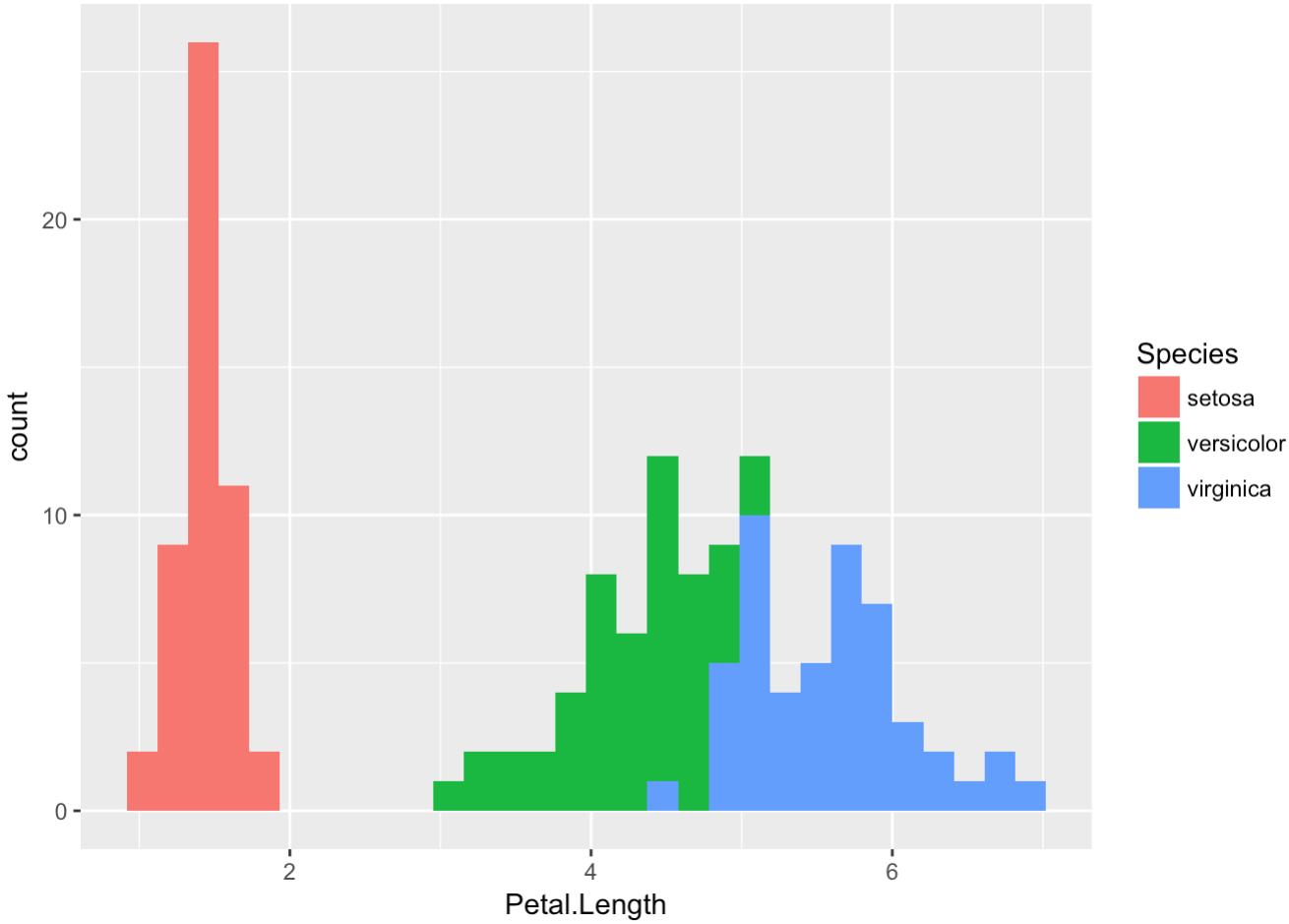
```
#Creamos el contenedor
ggplot(data=iris, aes(iris$Species, Petal.Length)) +
  geom_violin()  +  # dibujamos el diagrama de tipo VIOLIN
  stat_summary(fun.y=mean, geom="point", shape=18,
                size=3, color="red")   #pintamos la media con un punto ROJO
```



Histogramas

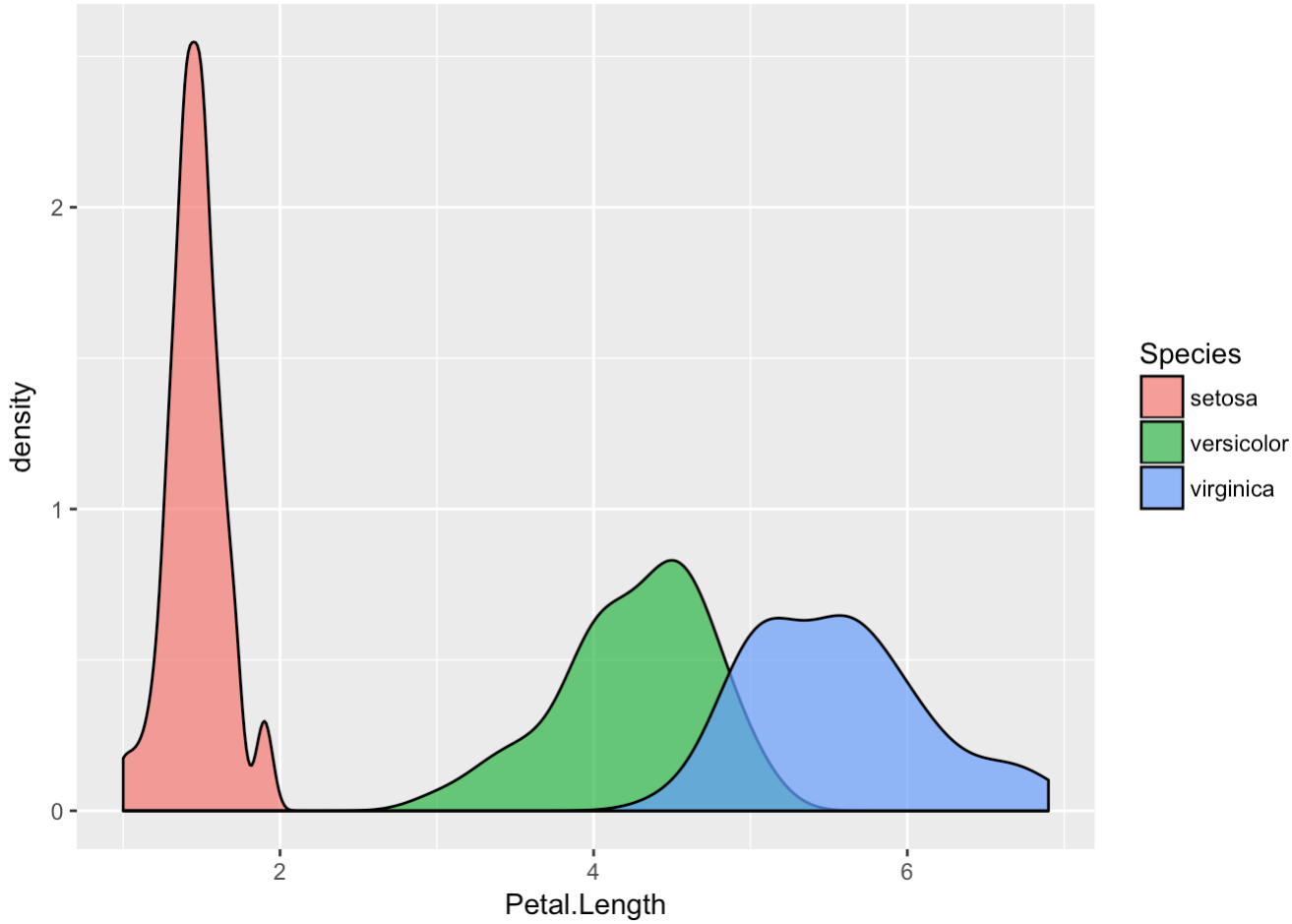
```
#Creamos el contenedor, preparamos a pintar por grupos de especies
ggplot(data=iris, aes(Petal.Length,fill=Species)) +
  geom_histogram()   # dibujamos el histograma
```

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



Diagramas de densidad

```
#Creamos el contenedor
ggplot(data=iris, aes(Petal.Length,fill=Species)) +
  geom_density(alpha=0.7)   # dibujamos el diagrama de densidad
```



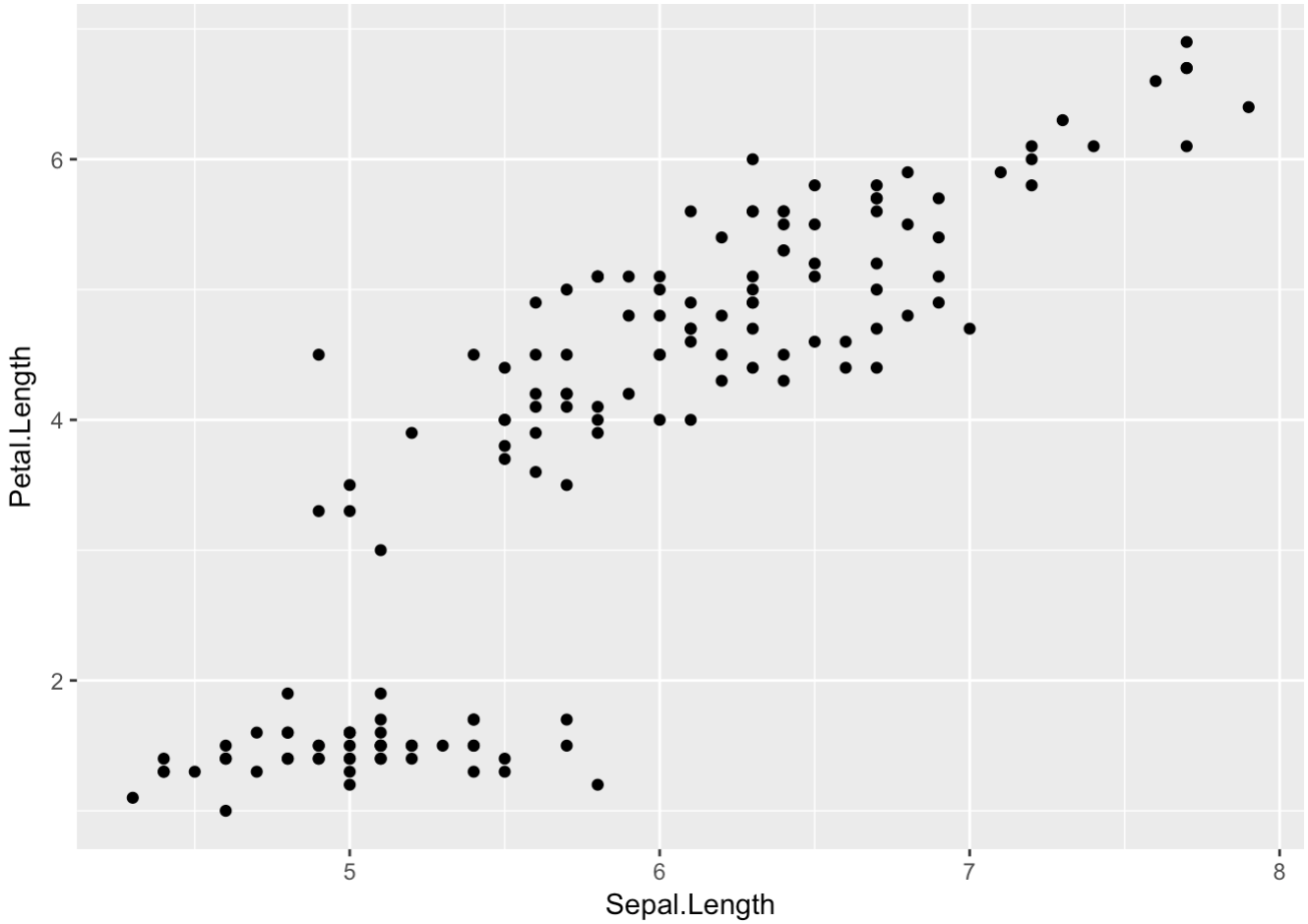
qplot: un ggplot simplificado

Para algunos diagramas básicos *ggplot2* tiene funciones que crean el contenedor a partir de un único comando. Por ejemplo, podemos usar *qplot()* para varios diagramas básicos.

Diagrama de puntos con qplot

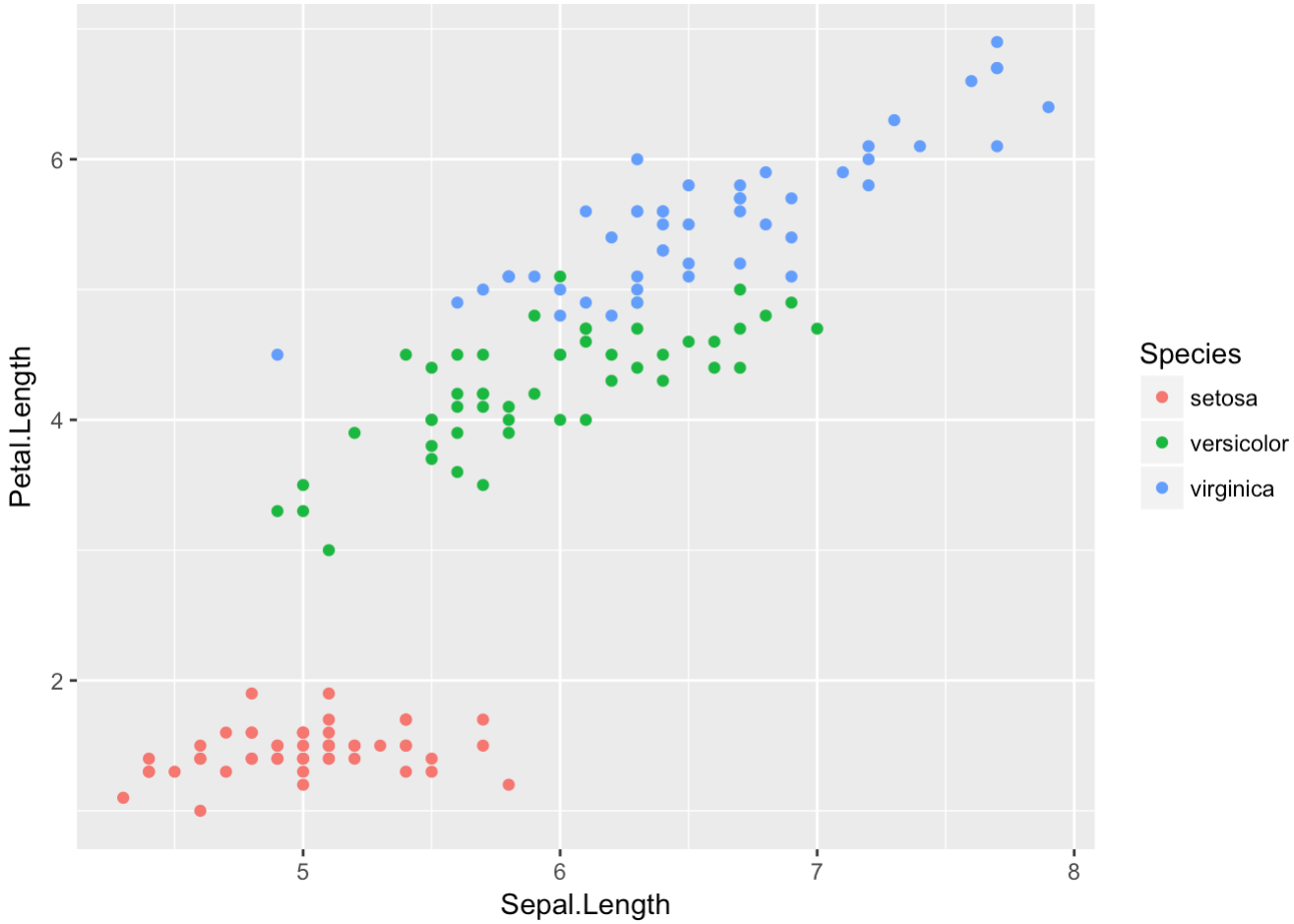
Un diagrama básico es el denominado “nube de puntos” o scatterplot. El paquete *ggplot2* tiene varias funciones que permiten representar unos datos, una de ellas es la función “qplot()”. Recomendamos mirar la ayuda de la propia función para mayores detalles.

```
qplot(Sepal.Length, Petal.Length, data = iris)
```



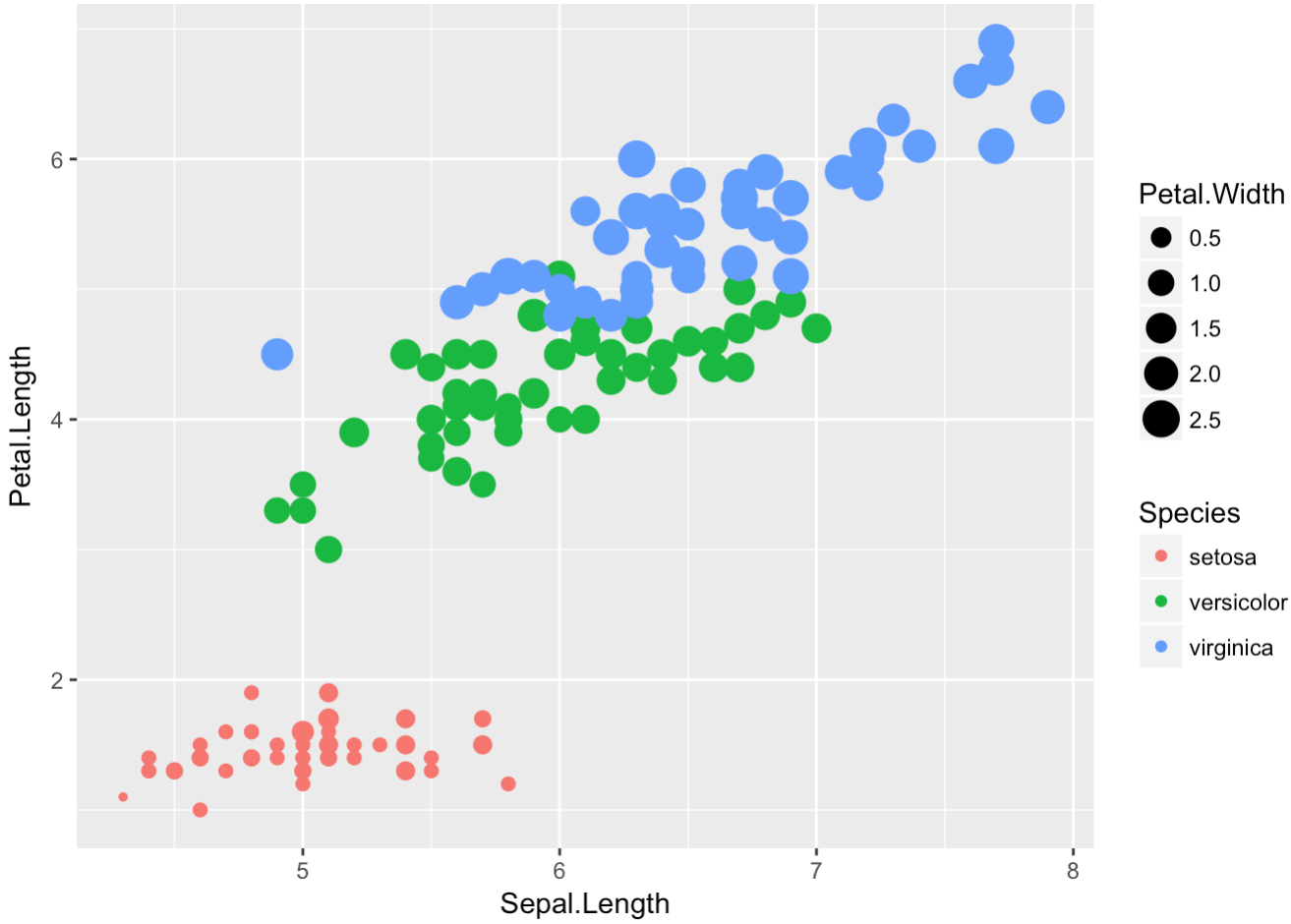
Podemos colorear estos puntos según la especie, por ejemplo:

```
#Colorear los puntos por especies
qplot(Sepal.Length, Petal.Length, data = iris, color = Species)
```

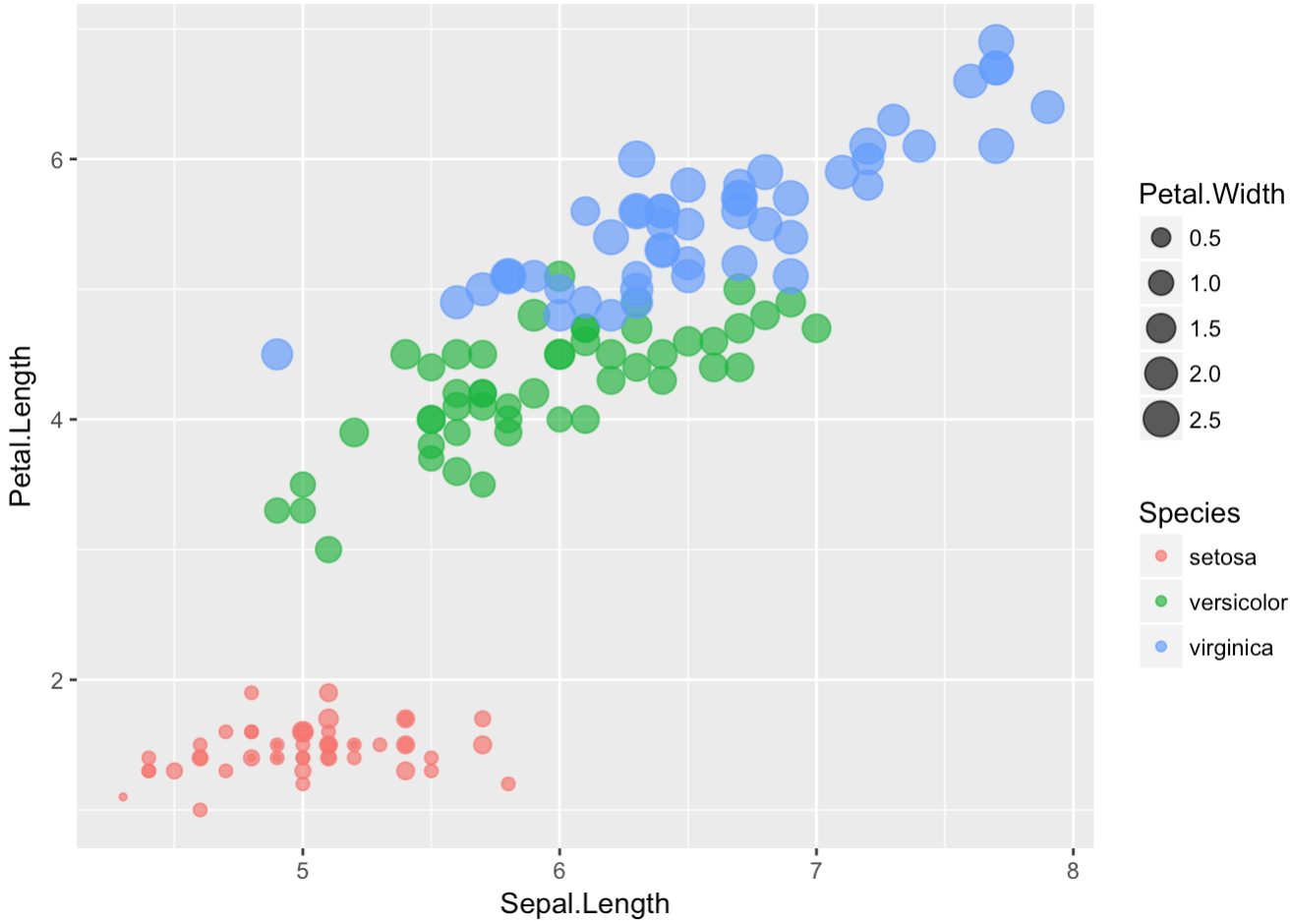


A veces es necesario mostrar otra variable en el mismo gráfico para encontrar una relación o patrón entre variables. En este caso, podríamos analizar conjuntamente ancho del pétalo “Petal.Width” usando, por ejemplo:

```
# Añadimos una variable representando el tamaño de los puntos
qplot(Sepal.Length, Petal.Length, data = iris, color = Species, size = Petal.Width)
```

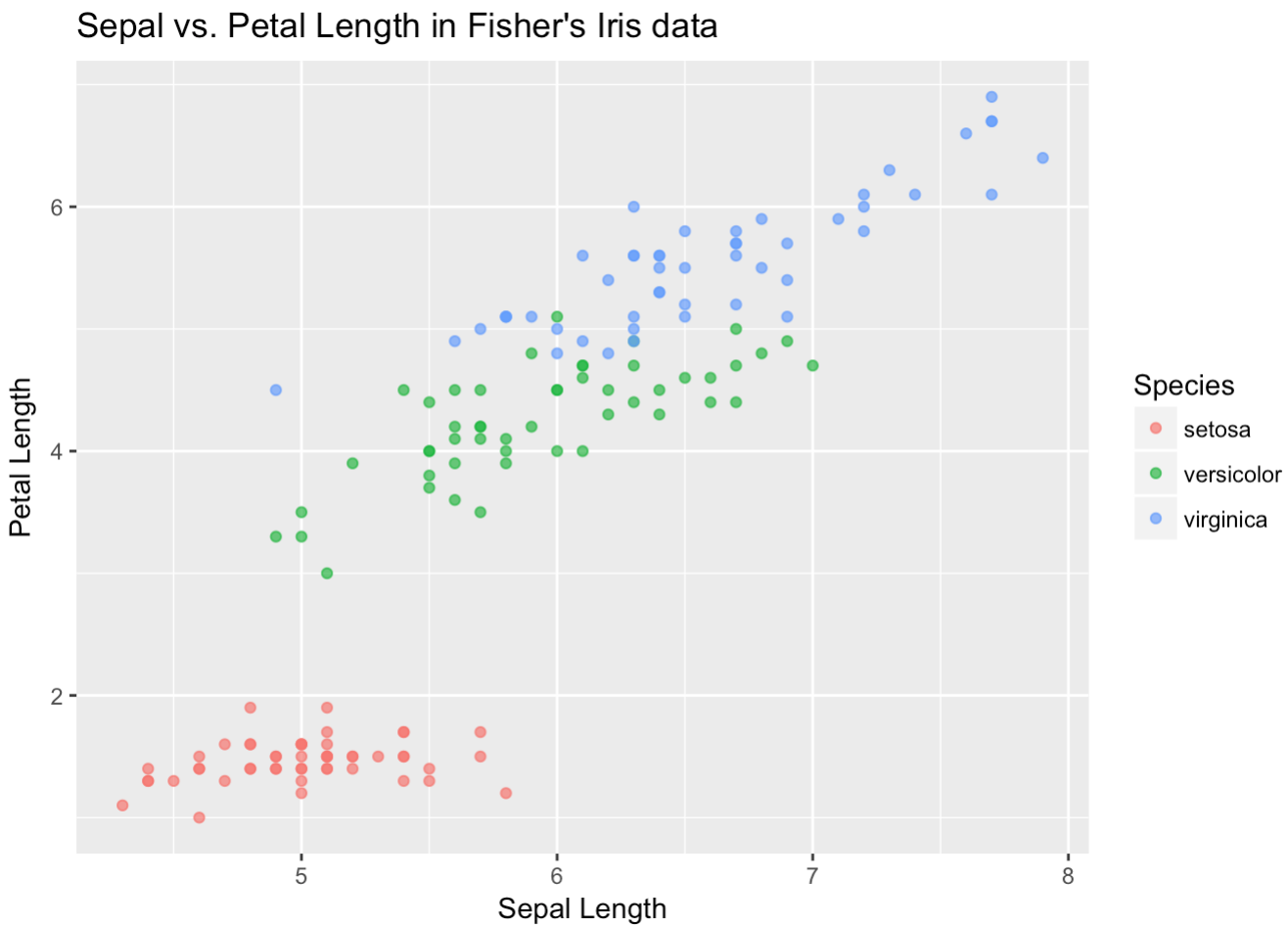



```
#Añadimos transparencia con "alpha=" para observar solapes
qplot(Sepal.Length, Petal.Length, data = iris, color = Species, size = Petal.Width, a
lpha = I(0.7))
```



Para finalizar, corregimos las etiquetas de los ejes y pondremos un título al gráfico.

```
qplot(Sepal.Length, Petal.Length, data = iris, color = Species,
    xlab = "Sepal Length", ylab = "Petal Length", alpha = I(0.7),
    main = "Sepal vs. Petal Length in Fisher's Iris data")
```



Librería Plotly: Gráficos interactivos y más llamativos

Como siempre ocurre con R, aparecen cada vez más librerías o paquetes que mejoran algunas características de los anteriores, o bien, proponen nuevas. En este caso vamos a presentar algunos gráficos usando el paquete *plotly*.

Todos los gráficos hechos con el paquete *plotly* tienen la capacidad de ser interactivos, siempre que la aplicación que muestre el gráfico lo permita. Así, si se copia el gráfico y se pega en un documento de Word, o un PDF, perderá la capacidad de ser interactivo. Si se usa en una página Web, o en el propio R o en RStudio, seguirá tenieno la interactividad.

Cuando hacemos referencia a diagramas interactivos, queremos decir que pasando el ratón por alguna parte del gráfico, inmediatamente aparece información relacionada con el mismo (valor del punto señalada, o de la barra o boxplot, grupo, etc.).

Veamos varios ejemplos, recordad que la interactividad dependerá del tipo de documento donde se haya insertado el gráfico, en un documento HTML es totalmente interactivo.

```
#install.packages("plotly") #esto lo hacemos en nuestro entorno RStudio una sola vez
library(plotly)   #Cargamos el paquete en memoria
```

```
## Warning: package 'plotly' was built under R version 3.4.1
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

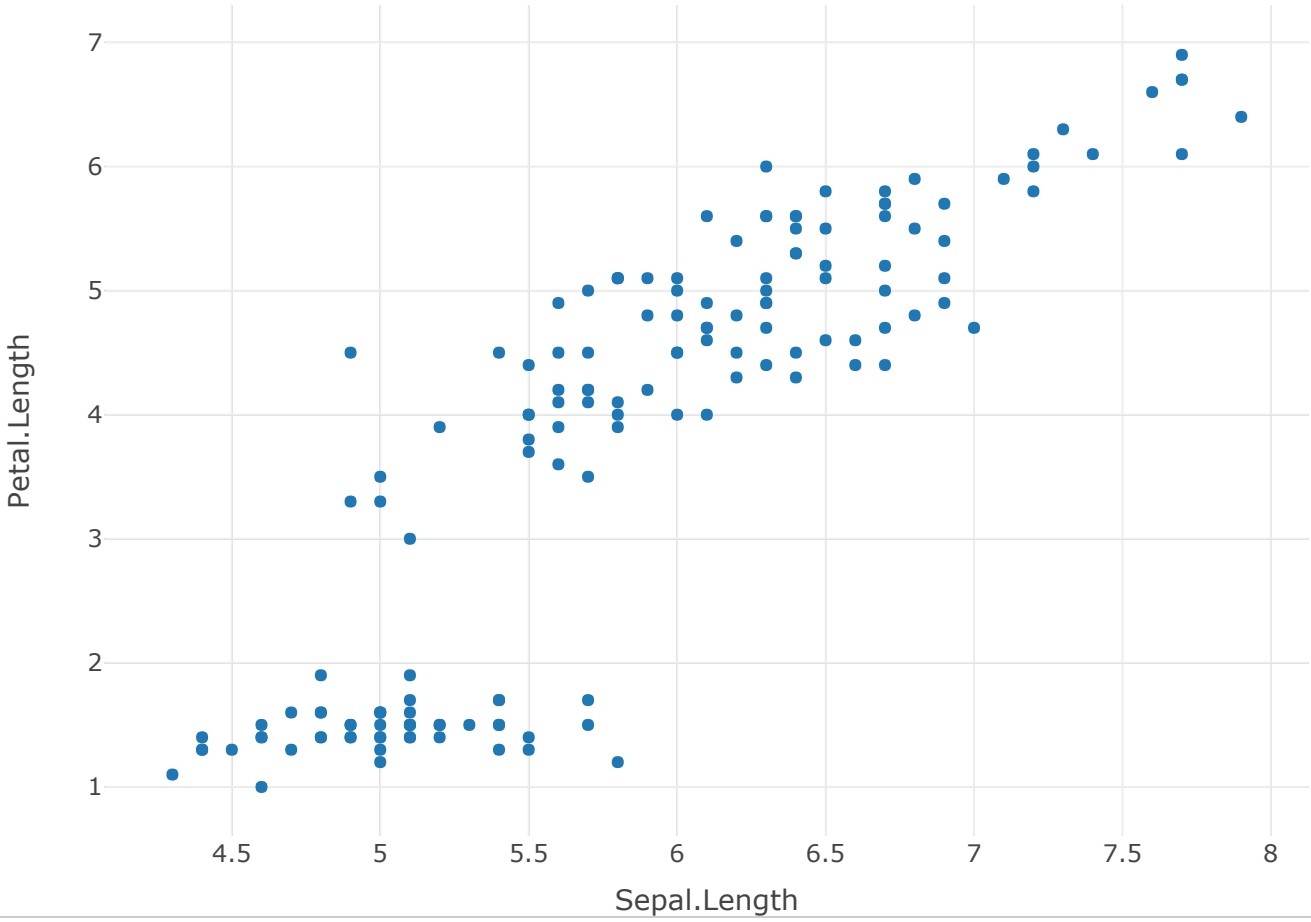
```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
#Creamos el gráfico
p <- plot_ly(data = iris, x = ~Sepal.Length, y = ~Petal.Length)
p
```

```
## No trace type specified:
##   Based on info supplied, a 'scatter' trace seems appropriate.
##   Read more about this trace type -> https://plot.ly/r/reference/#scatter
```

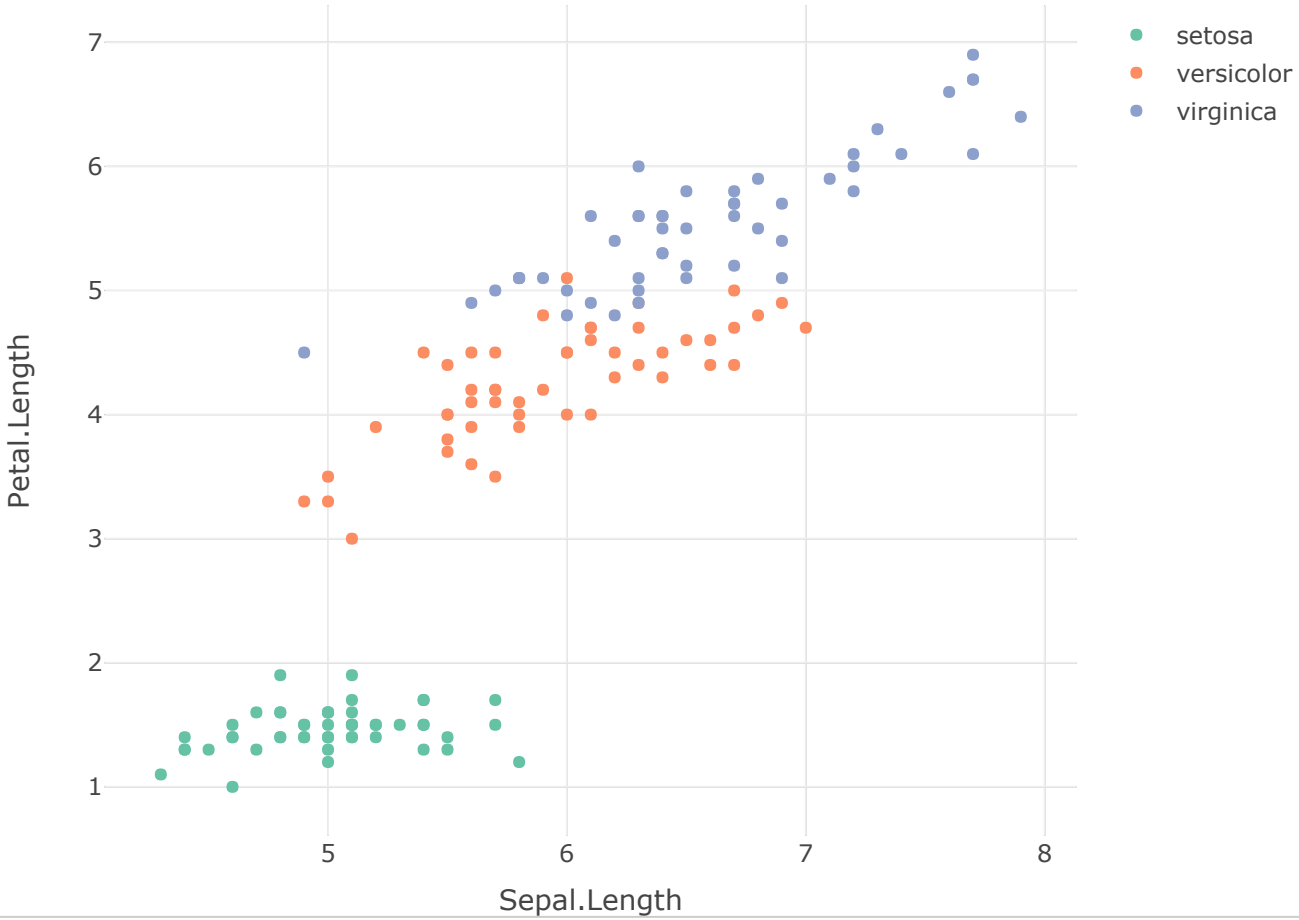
```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```




```
plot_ly(data = iris, x = ~Sepal.Length, y = ~Petal.Length, color = ~Species)
```

```
## No trace type specified:
##   Based on info supplied, a 'scatter' trace seems appropriate.
##   Read more about this trace type -> https://plot.ly/r/reference/#scatter
```

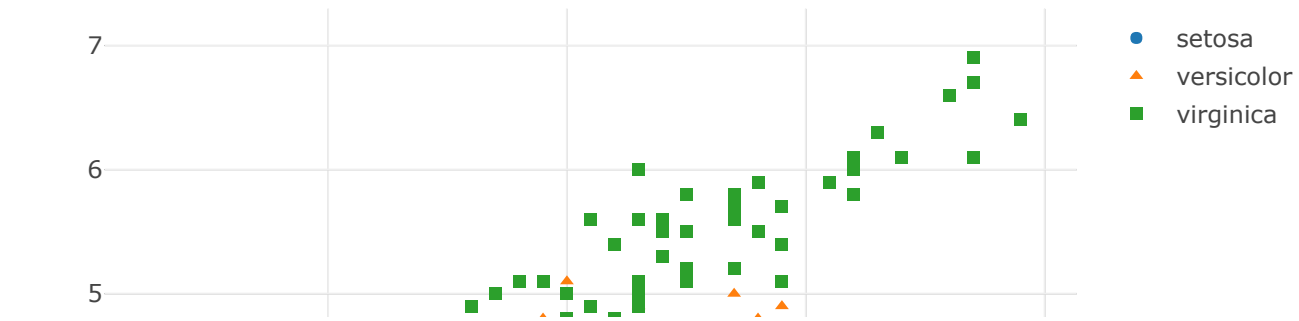
```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```

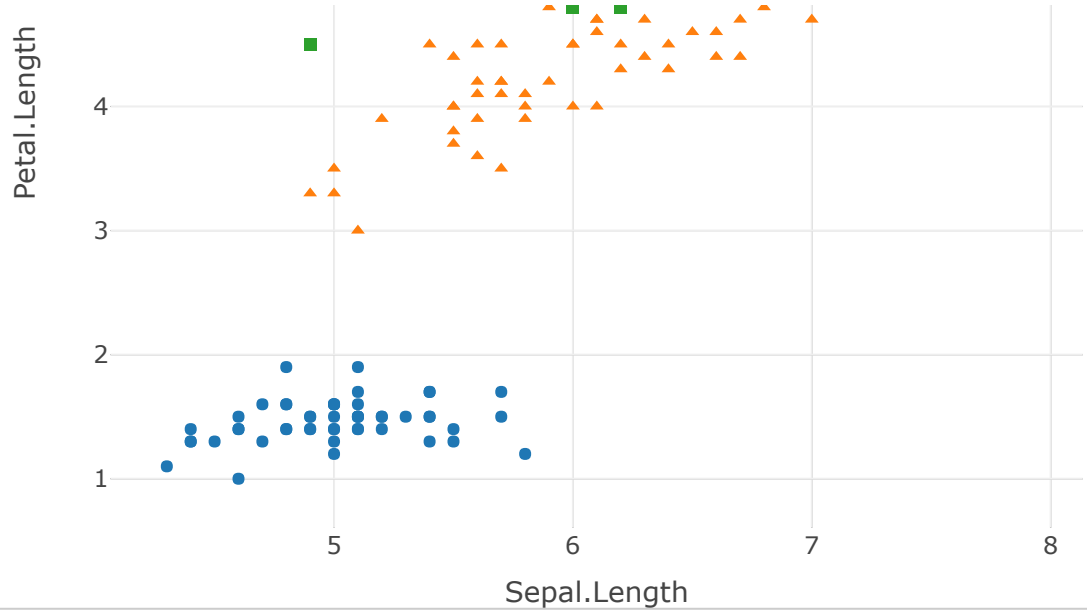



```
plot_ly(data = iris, x = ~Sepal.Length, y = ~Petal.Length, symbol = ~Species)
```

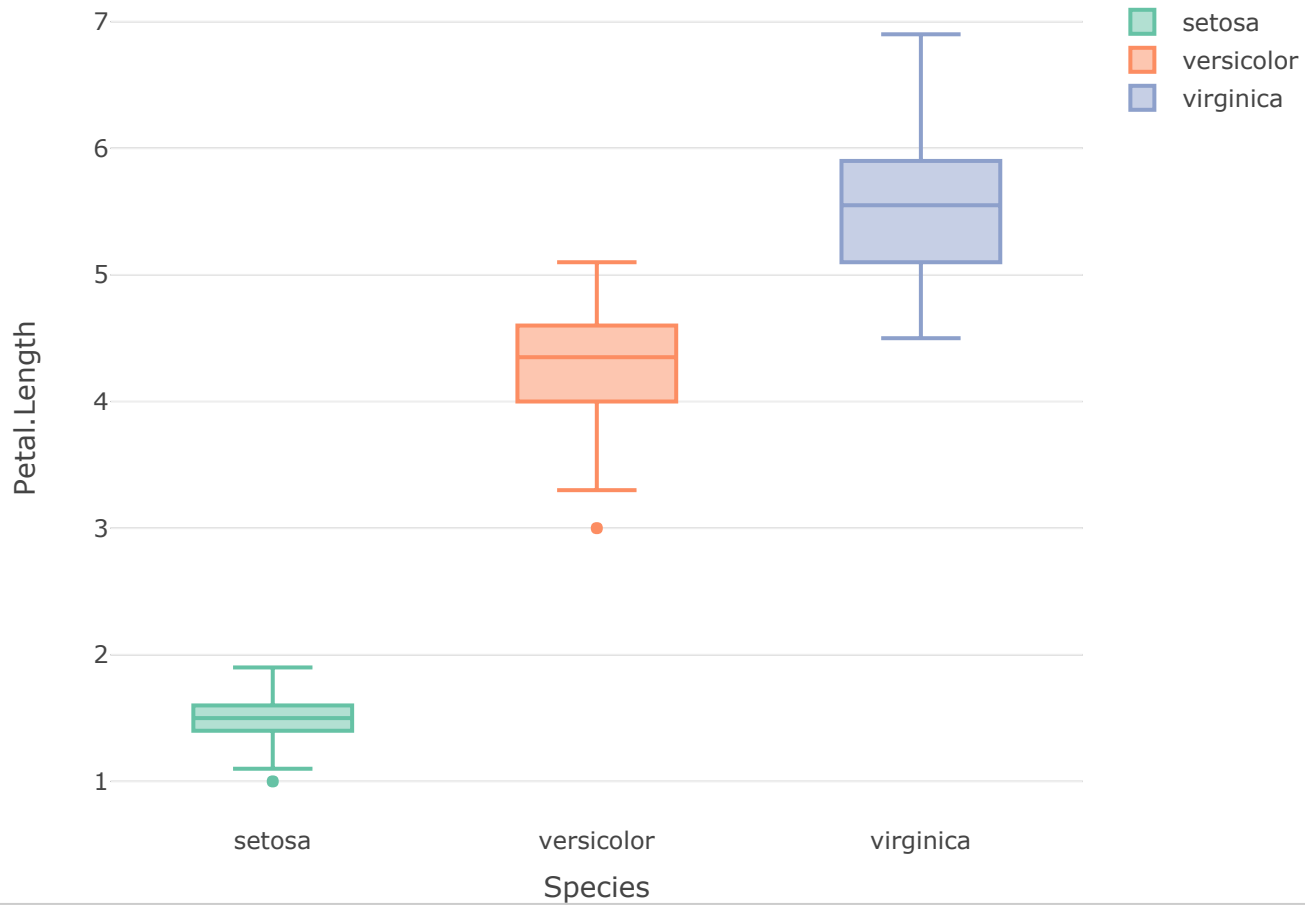
```
## No trace type specified:
##   Based on info supplied, a 'scatter' trace seems appropriate.
##   Read more about this trace type -> https://plot.ly/r/reference/#scatter
```

```
## No scatter mode specifed:
##   Setting the mode to markers
##   Read more about this attribute -> https://plot.ly/r/reference/#scatter-mode
```

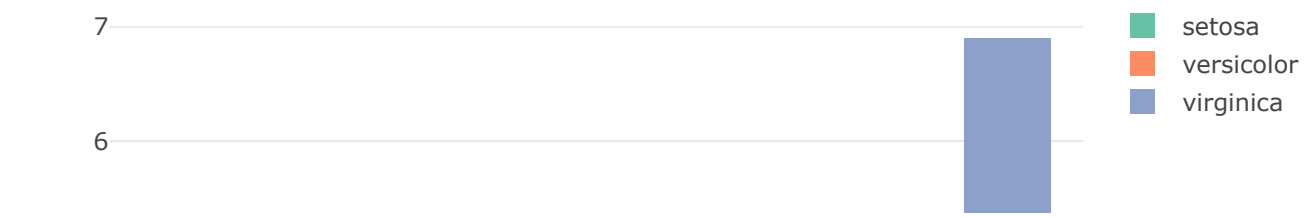


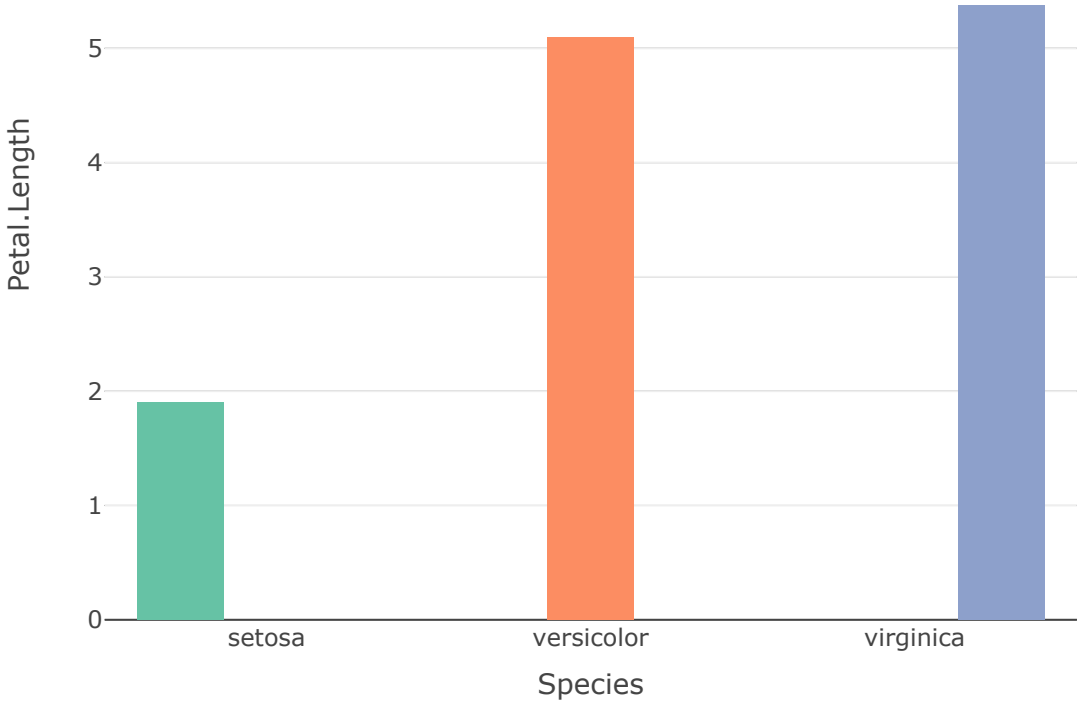



```
plot_ly(data = iris, x = ~Species, y = ~Petal.Length, color = ~Species,  type =
"box")
```




```
plot_ly(data = iris, x = ~Species, y = ~Petal.Length, color = ~Species,  type = "bar"
 )
```





Nota final acerca de la posibilidad de gráficos en R

Como en todos los casos en que presentamos funciones o parámetros de funciones, recomendamos buscar en la ayuda de R o de los paquetes usados para más detalles acerca de otras características que pueden ser de utilidad para personalizar gráficos.