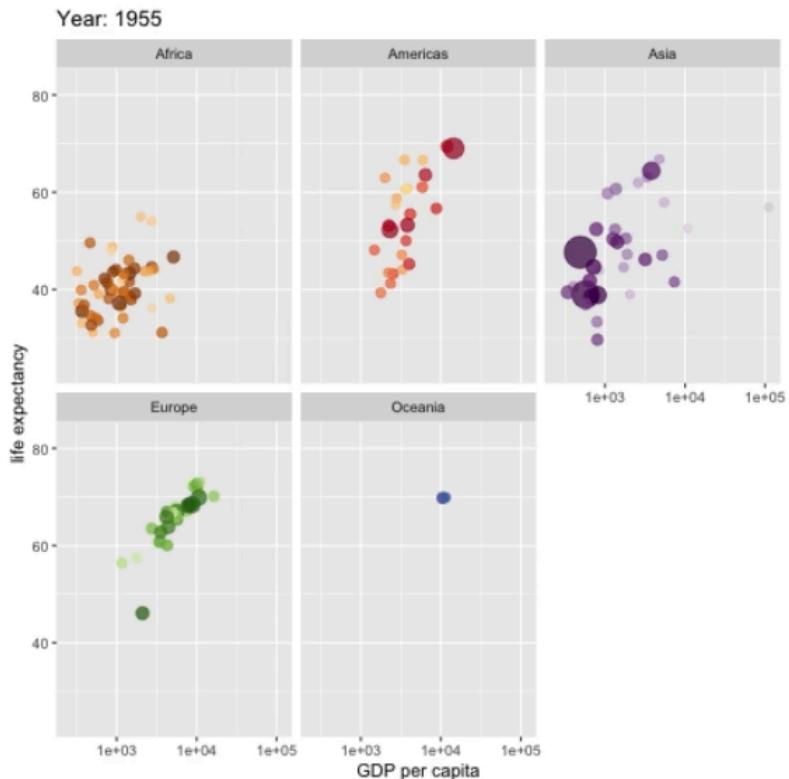




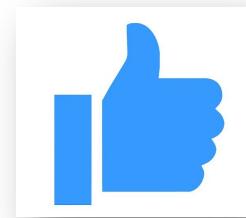
Visualización de datos en R!



Visualización de datos en R!



Recuerden levantar la mano si tienen preguntas o el pulgar si todo esta claro!

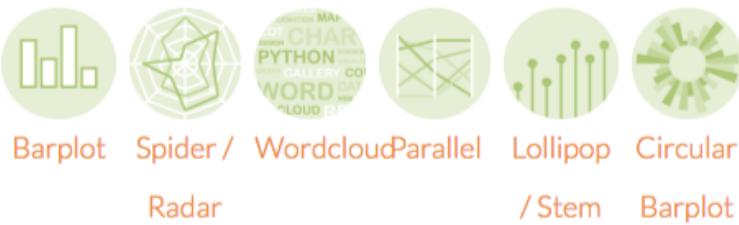




Scatter Heatmap Correlogram Bubble Connected Density

Scatter 2D

Rankings



Barplot Spider / Wordcloud Parallel / Stem

Lollipop Circular Barplot

Part of a whole



Treemap Dendrogram Venn Diagram

Stacked Bar Pie Doughnut Circular

Bar Chart

Evolution



Stacked Line Area Streamgraph Parallel

Area

Maps



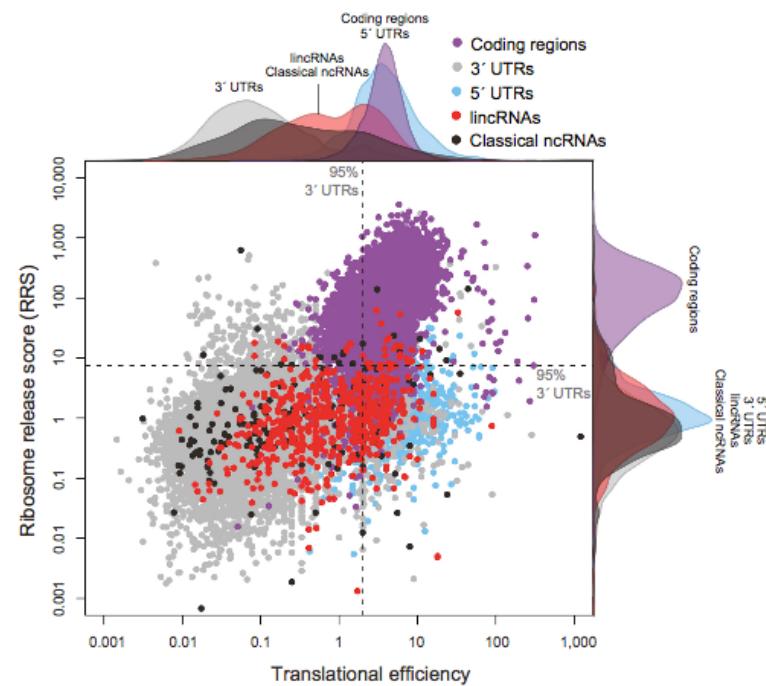
Bubble Choropleth Leaflet Another Map

¿Qué haremos hoy??

GRÁFICOS EN R!

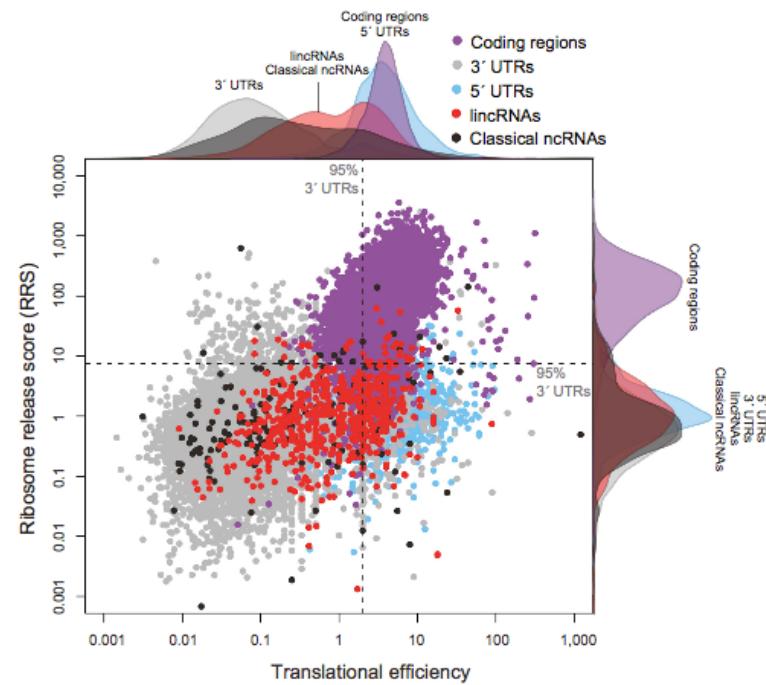
Gráficos en R

- Inspiración
- Gráficos base en R
 - Gráficos de puntos
 - Manipulación de gráficos
 - Otros tipos básicos de gráficos
 - Guardar gráficos



Gráficos en R

- **Inspiración**
- Gráficos base en R
 - Gráfica de puntos
 - Manipulación de graficas
 - Otros tipos básicos de graficas
 - Guardar graficas



¡Inspiración! Gráficos en R... Wow

<https://www.r-graph-gallery.com/> : Inspiración y código

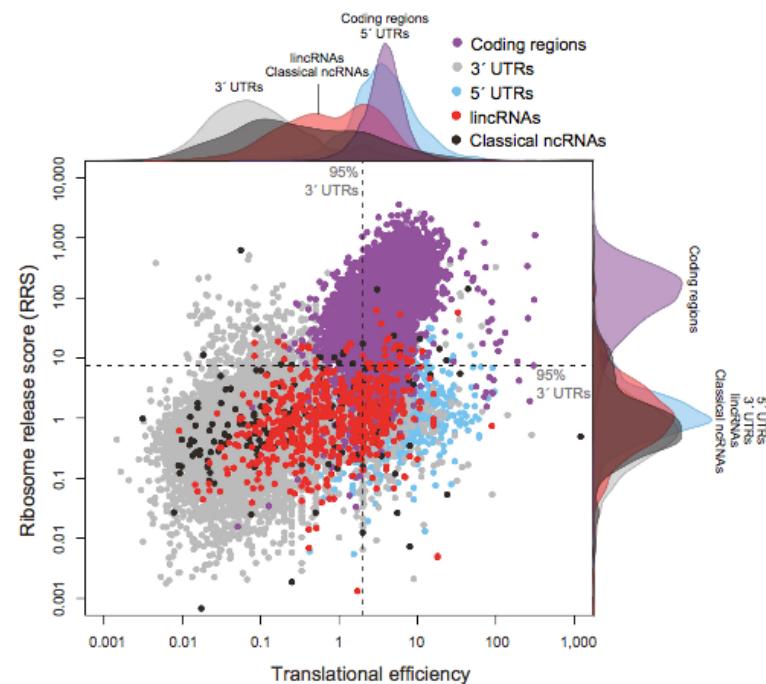


THE R GRAPH
GALLERY

<http://www.r-graph-gallery.com/all-graphs/>

Gráficos en R

- **Inspiración**
- **Gráficos base en R**
 - **Gráfica de puntos**
 - Manipulación de graficas
 - Otros tipos básicos de graficas
 - Guardar graficas



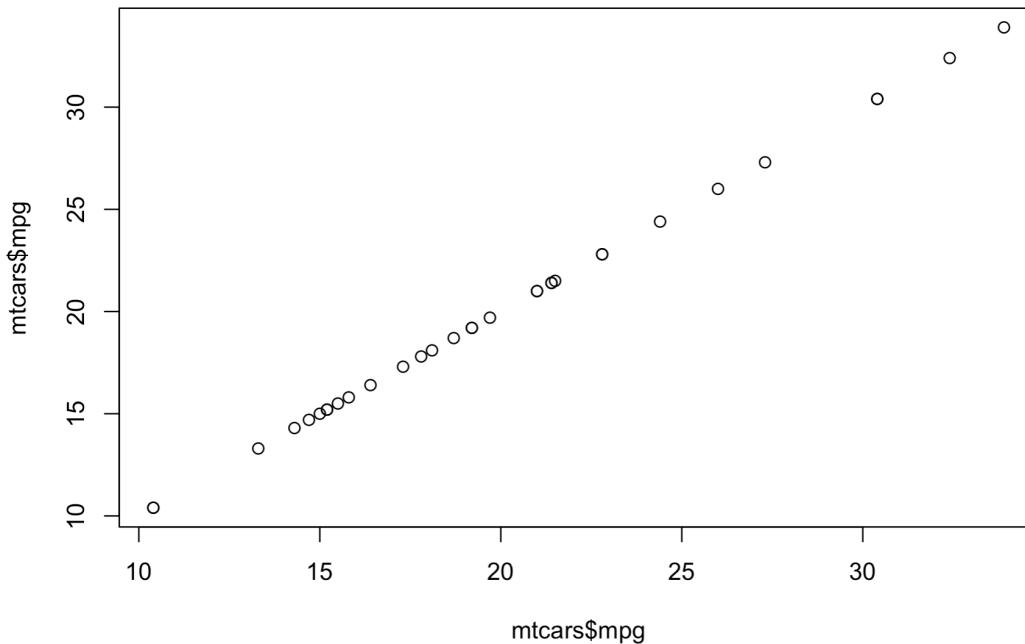
Gráficos en R: resumen

- Hay varias formas de hacer gráficos en R
 - **Gráficos base**
 - los cambios en el diseño son bastante fáciles, altamente modificables.
 - **ggplot2**
 - Bueno para graficos “multipanel”, vistas alternativas rápidas de datos, los cambios en el diseño básico pueden ser difíciles.

Gráficos base: `plot()`

- `plot` es la función genérica para graficar objetos R
- puntos, líneas, etc.

?`plot`



Tres formas de usar `plot()` para hacer una grafica de puntos

```
> plot(x = primates$Bodywt, y = primates$Brainwt)

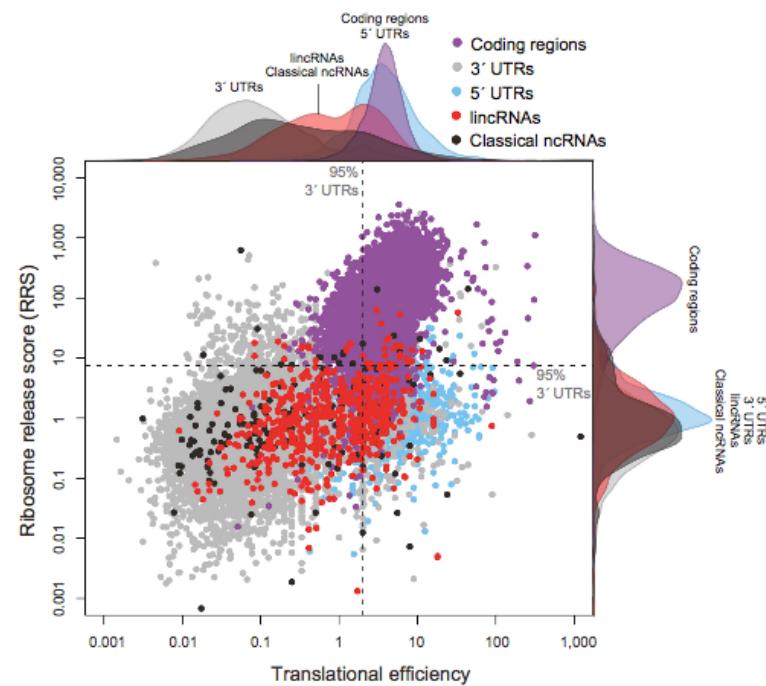
> plot(primates$Brainwt ~ primates$Bodywt)
# ~ traducido como Brainwt *como una función de* Bodywt

> plot(Brainwt ~ Bodywt, data = primates)
# indicar a R de dónde proceden los datos utilizando
data =
```



Gráficos en R

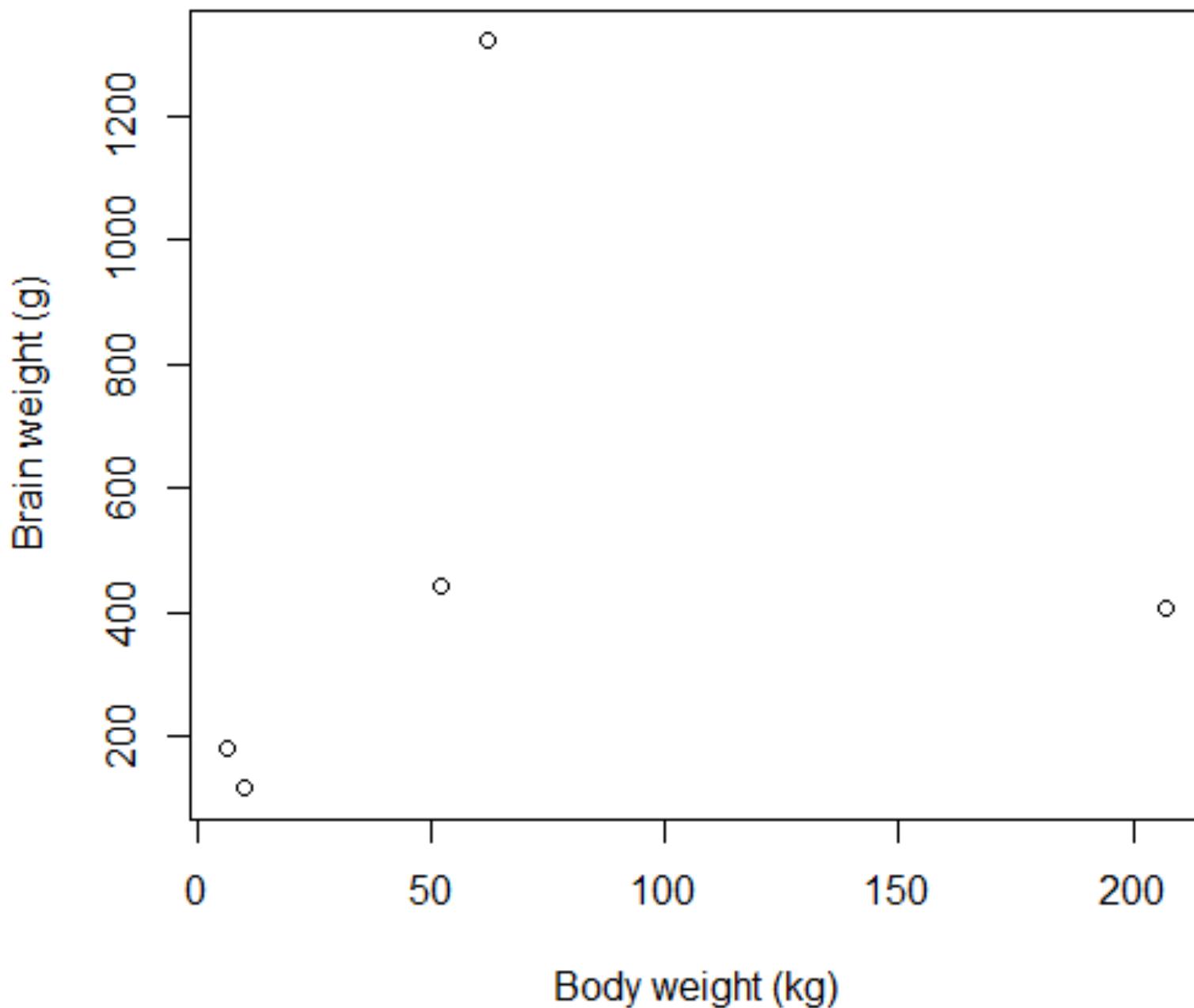
- **Inspiración**
- **Gráficos base en R**
 - Gráfica de puntos
 - **Manipulación de graficas**
 - Otros tipos básicos de graficas
 - Guardar graficas



Nombres de los ejes

- De forma predeterminada, R utiliza el nombre de las variables como nombres de eje.
- Use `xlab` y `ylab` para cambiar los nombres.

```
plot(x = primates$Bodywt, y = primates$Brainwt,  
      xlab = "Body weight (kg)",  
      ylab = "Brain weight (g)"  
    )
```

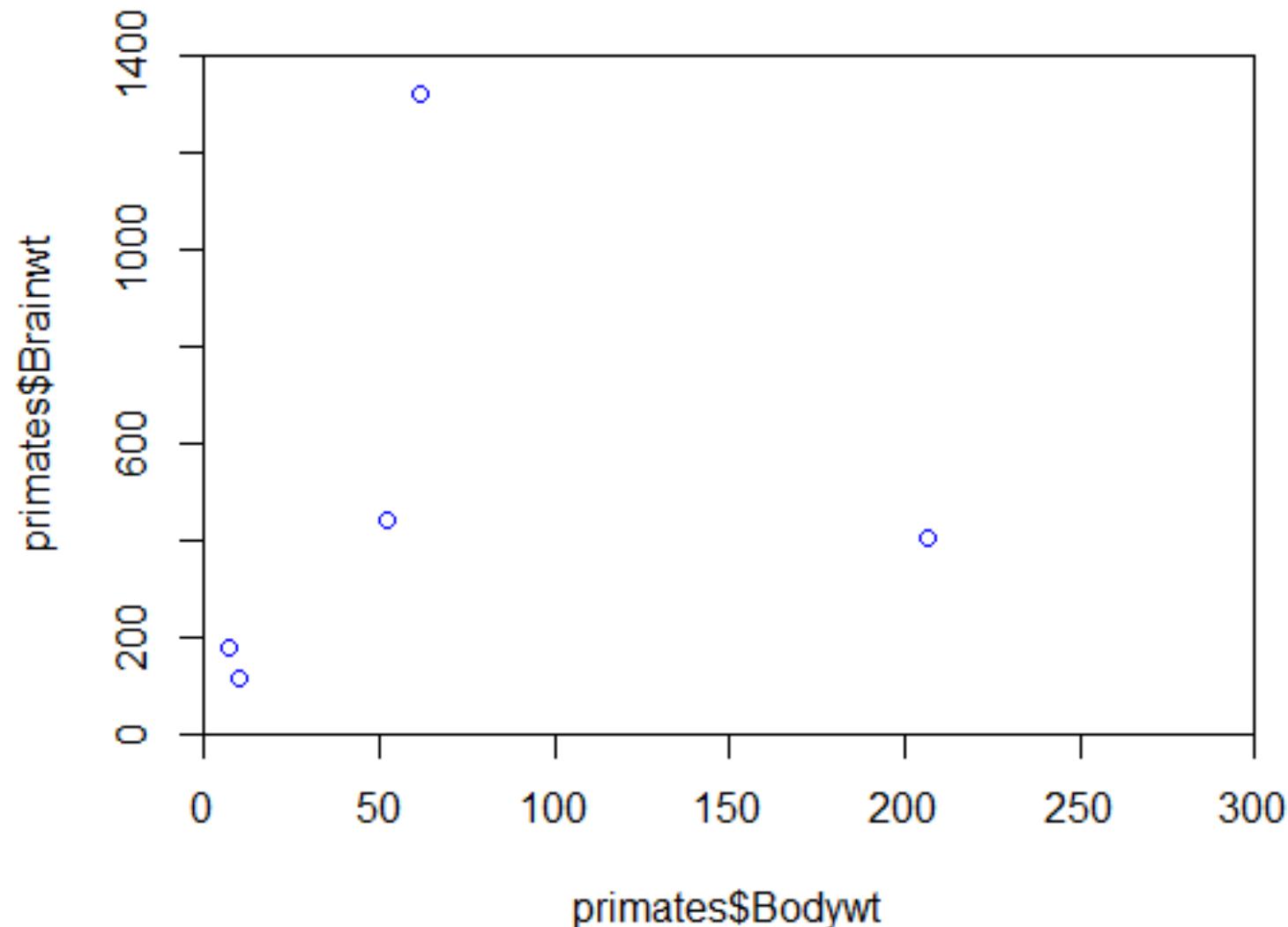


colores en R

- Se pueden especificar colores de puntos, líneas, texto, etc.
- Los colores se pueden aplicar a varias partes del grafico con los argumentos:
 - `col` (color predeterminado)
 - `col.axis` (color de los ejes)
 - `col.lab` (nombres de los ejes x, y)
 - `col.main` (Titulo del grafico)
- Los colores se pueden especificar con el nombre o número de color
`col="blue"`

```
plot(x = primates$Bodywt, y = primates$Brainwt,  
      col="blue")
```

```
plot(x = primates$Bodywt, y = primates$Brainwt, col="blue")
```



pch: Controla las características de los puntos

- El valor predeterminado es un círculo abierto (`pch=1`) de la tamaño 1 (`cex=1`)
 - `pch` abreviatura de `plotting character`
 - `cex` abreviatura de `character expansion`

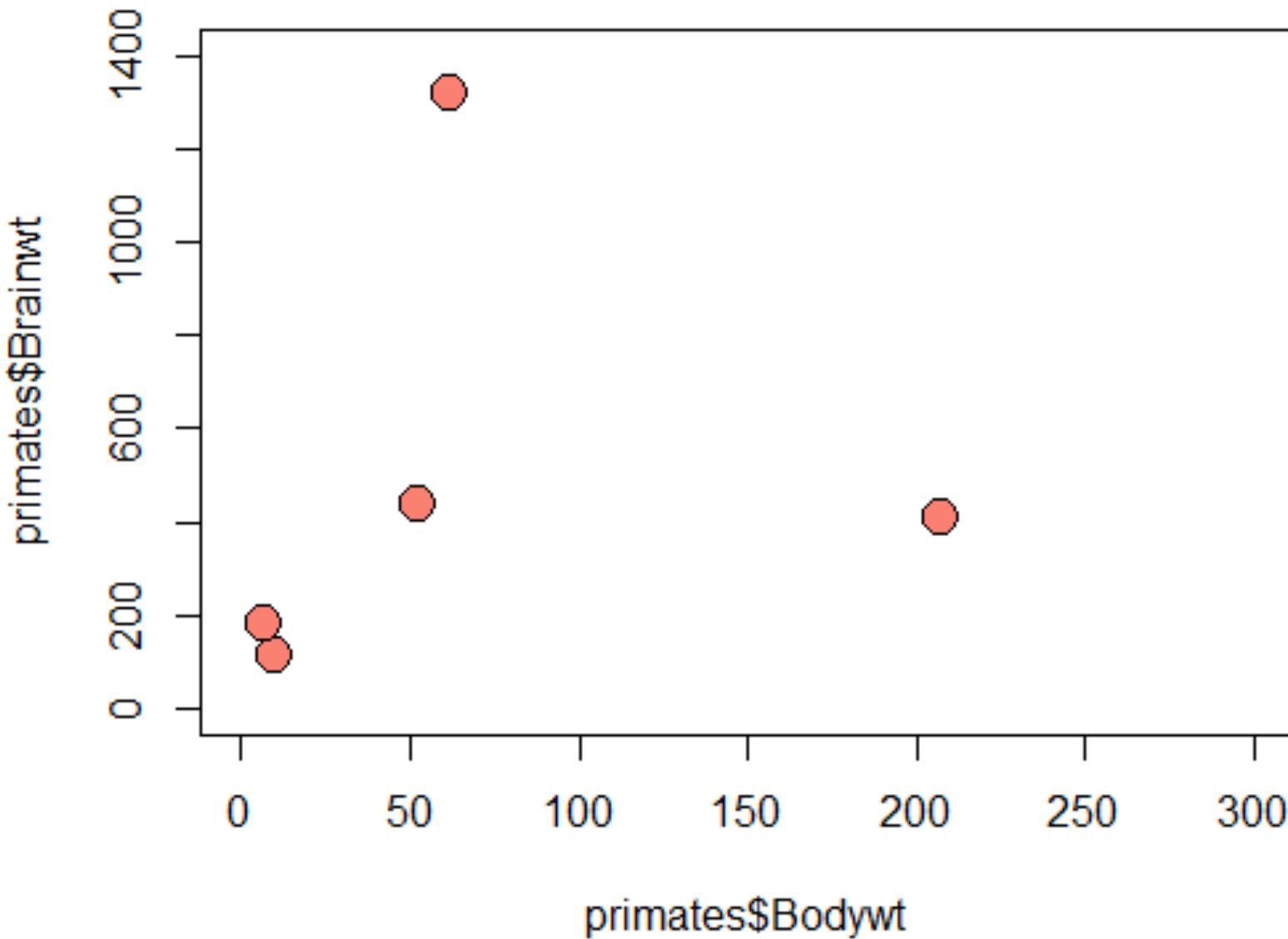
`pch` controls the type of symbol, either an integer between 1 and 25, or any single char within ""

1 ○	2 △	3 +	4 ×	5 ◊	6 ▽	7 ☒	8 *
9 ◆	10 ⊕	11 ☀	12田	13⊗	14□	15■	
16●	17▲	18◆	19●	20●	21○	22□	23◊
24△	25▽	* *	. .	X X	a a	? ?	

Opciones útiles para pch

- Use `pch=21` para llenar círculos:
 - Especifique el color del círculo con `col`
 - Especifique el color de relleno con `bg`

```
plot(x = primates$Bodywt, y = primates$Brainwt,  
      pch=21, col="black", bg="salmon"  
)
```

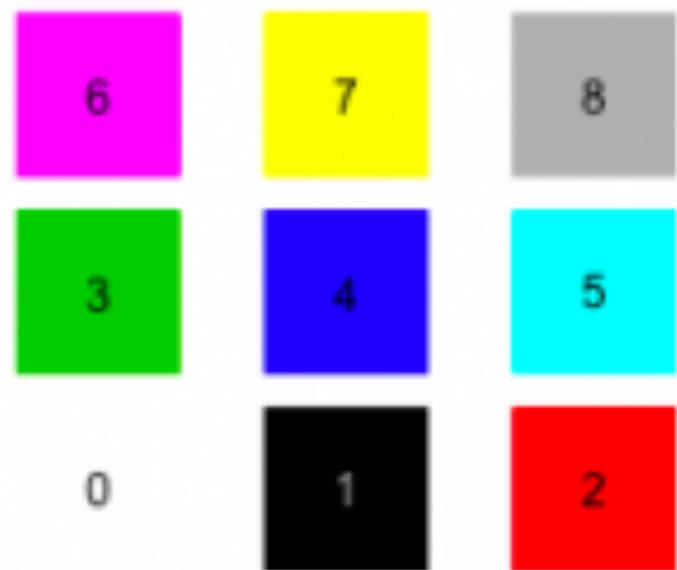


colores en R

- Varias maneras de llamar los colores
- Números: `col=1, col=2`
 - Práctico, pero limitado
- Quiero colores extraños!



#Paleta predeterminada



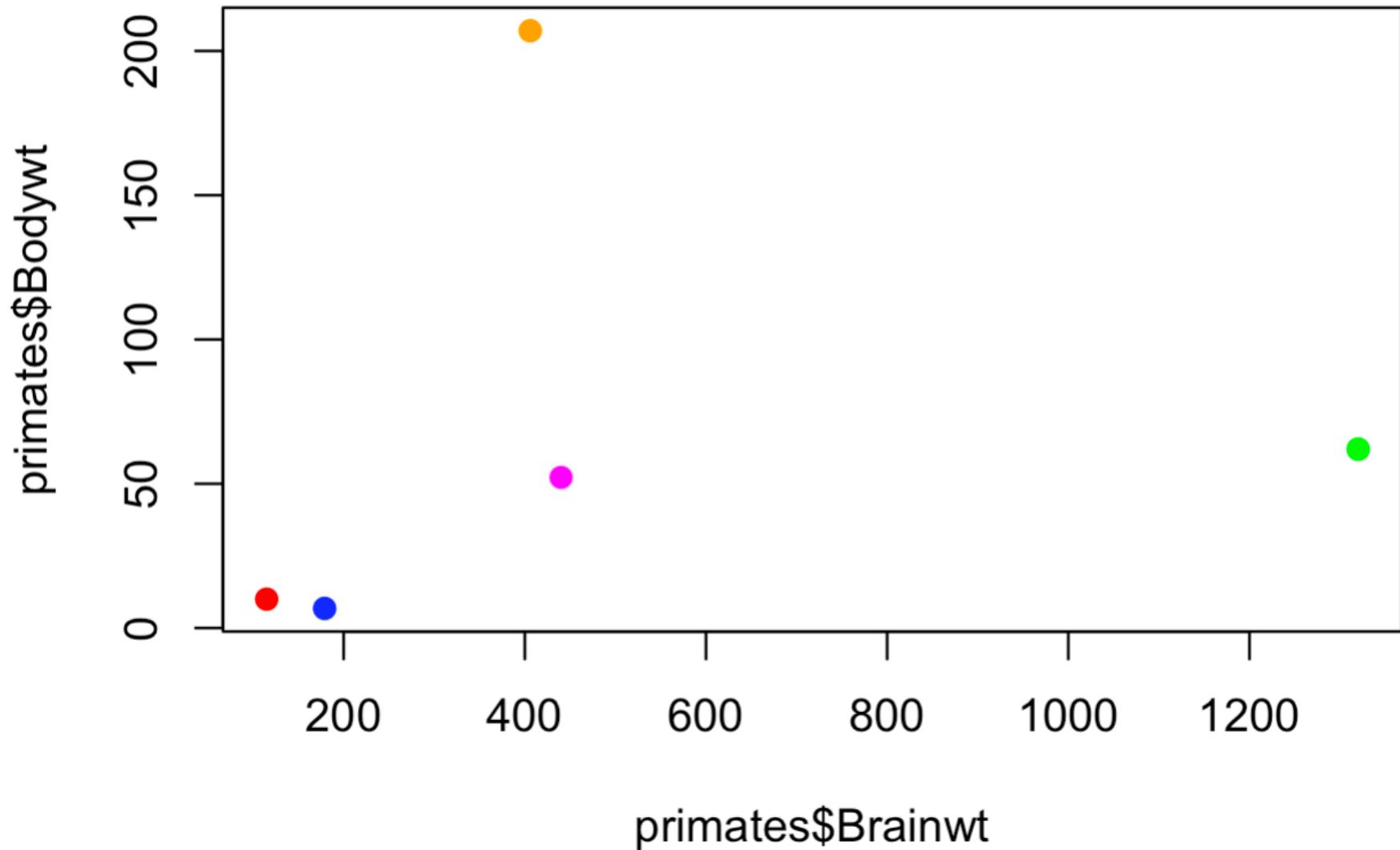
Otros Colores!

- Hay 657 colores con nombre en R

```
colors()
```

```
point.colors <- c("red", "orange", "green",  
"blue", "magenta")
```

```
plot(primates$Brainwt, primates$Bodywt,  
      pch=19, col=point.colors)
```



Hay otras maneras de llamar **Colores**

51	chartreuse4	#458B00
52	chocolate	#D2691E
53	chocolate1	#FF7F24

- También es posible llamar a una función por su número de índice. Por ejemplo, si necesita el número de índice de color 143, use `colors()[143]`
- Todos los colores se pueden definir por su código hexadecimal: #69b3a2. Para encontrar el color de sus sueños, visita este link: <https://htmlcolorcodes.com/>

R - carta de colores

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125
126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225
226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250

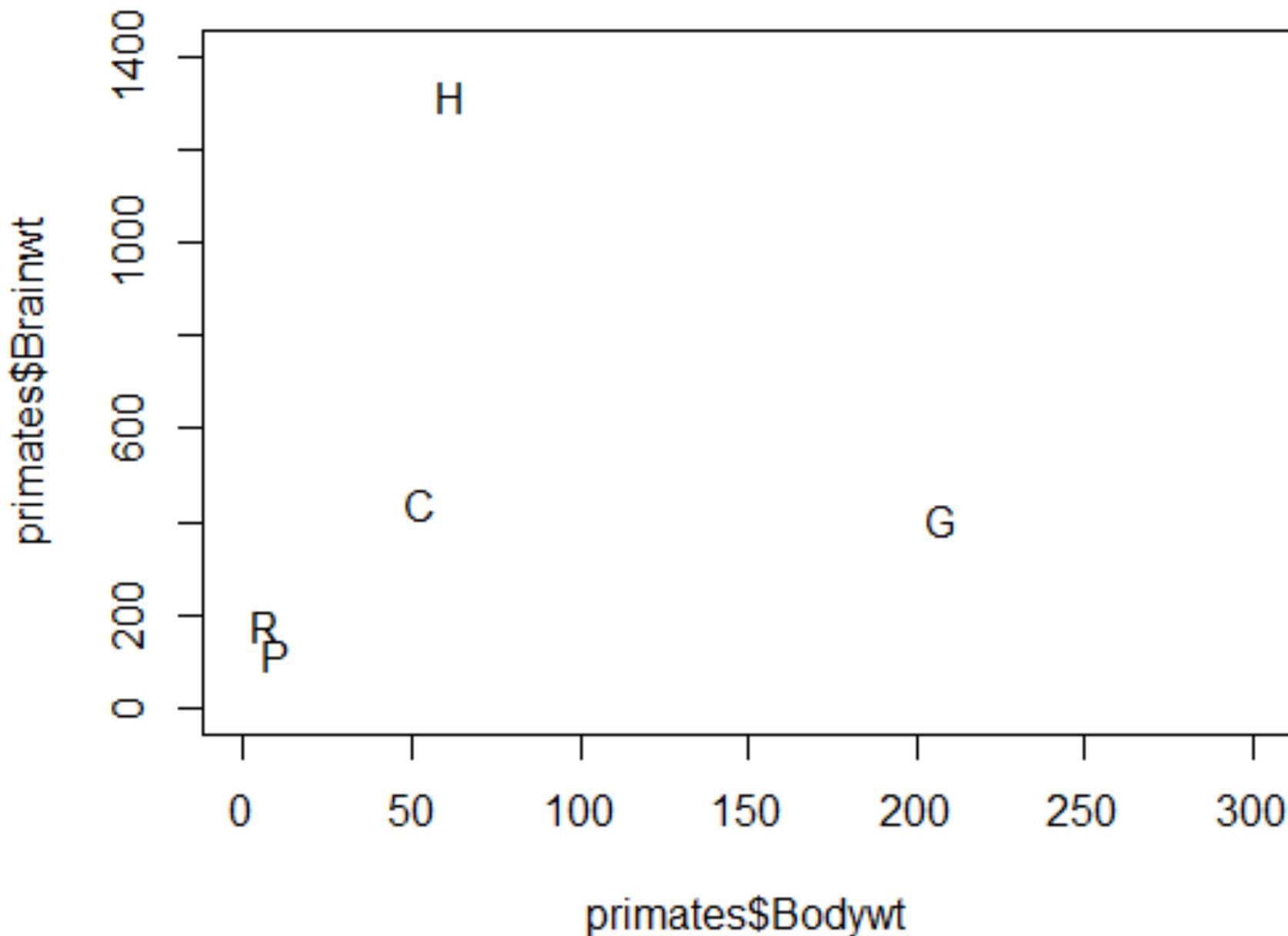
1	white	#FFFFFF	255	255	255
2	aliceblue	#F0F8FF	240	248	255
3	antiquewhite	#FAEBD7	250	235	215
4	antiquewhite1	#FFEFD8	255	239	219
5	antiquewhite2	#EEDFCC	238	223	204
6	antiquewhite3	#CDC0B0	205	192	176
7	antiquewhite4	#8B8A7B	139	131	120

51	chartreuse4	#458B00	69	139	0
52	chocolate	#D2691E	210	105	30
53	chocolate1	#FF7F24	255	127	36
54	chocolate2	#EE7621	238	118	33
55	chocolate3	#CD661D	205	102	29
56	chocolate4	#8B4513	139	69	19
57	coral	#FF7FFF	255	127	80

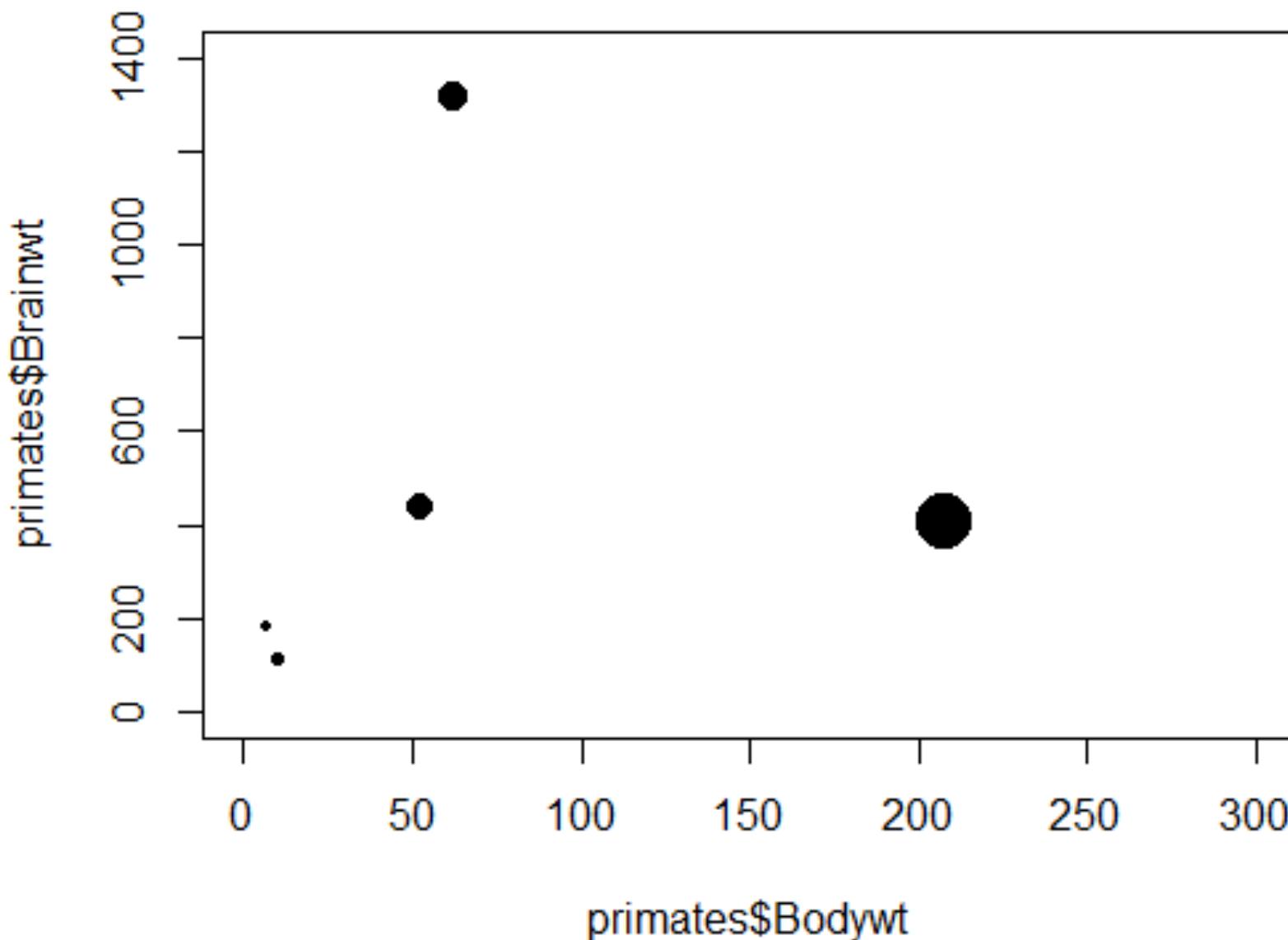
Opciones vectoriales para graficas

- Muchas opciones de graficos toman vectores, cada elemento se aplica a una iteración (e.g., un punto)
- Los vectores se **reciclan** si se suministran muy pocos números
- Diferentes caracteres de punto: **pch=1:5**
- Diferentes letras de punto: **pch=c("a","t","c","g")**
- Diferentes colores: **col=1:5**
- Diferentes tamaños: **cex=1:5**

Primera letra del nombre de cada primate



Tamaño del círculo proporcional al peso corporal



Adición de leyendas

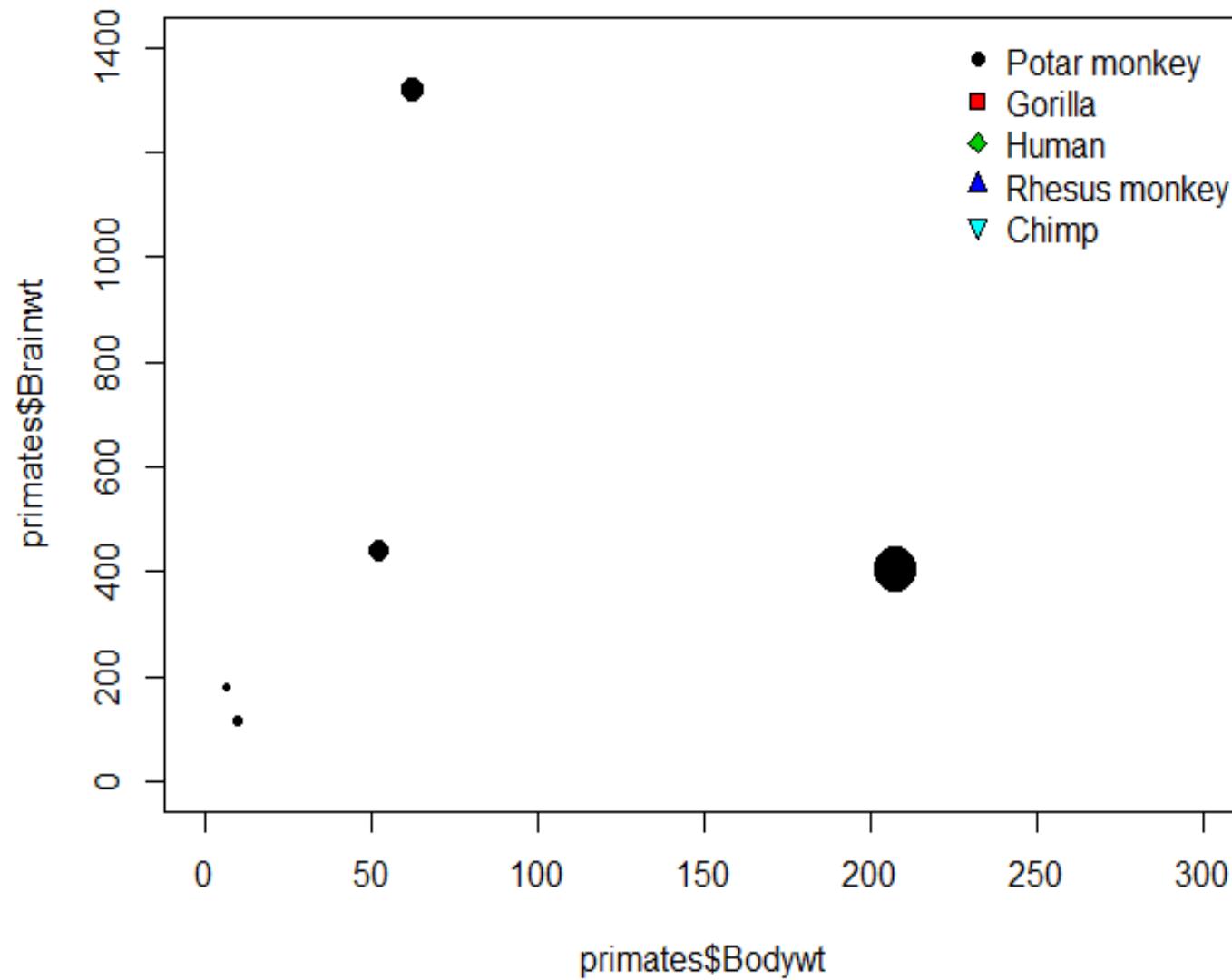
Buscar ayuda en la función de leyenda (`?legend`) para ver los argumentos!

legend(x="topright",
 legend=primates[,1],
 pt.bg=1:5,
 pch=21:25,
 bty="n")

también "bottomleft", etc., o
coordenadas (x=100, y=100)

Las leyendas no utilizan ninguna información en la gráfica

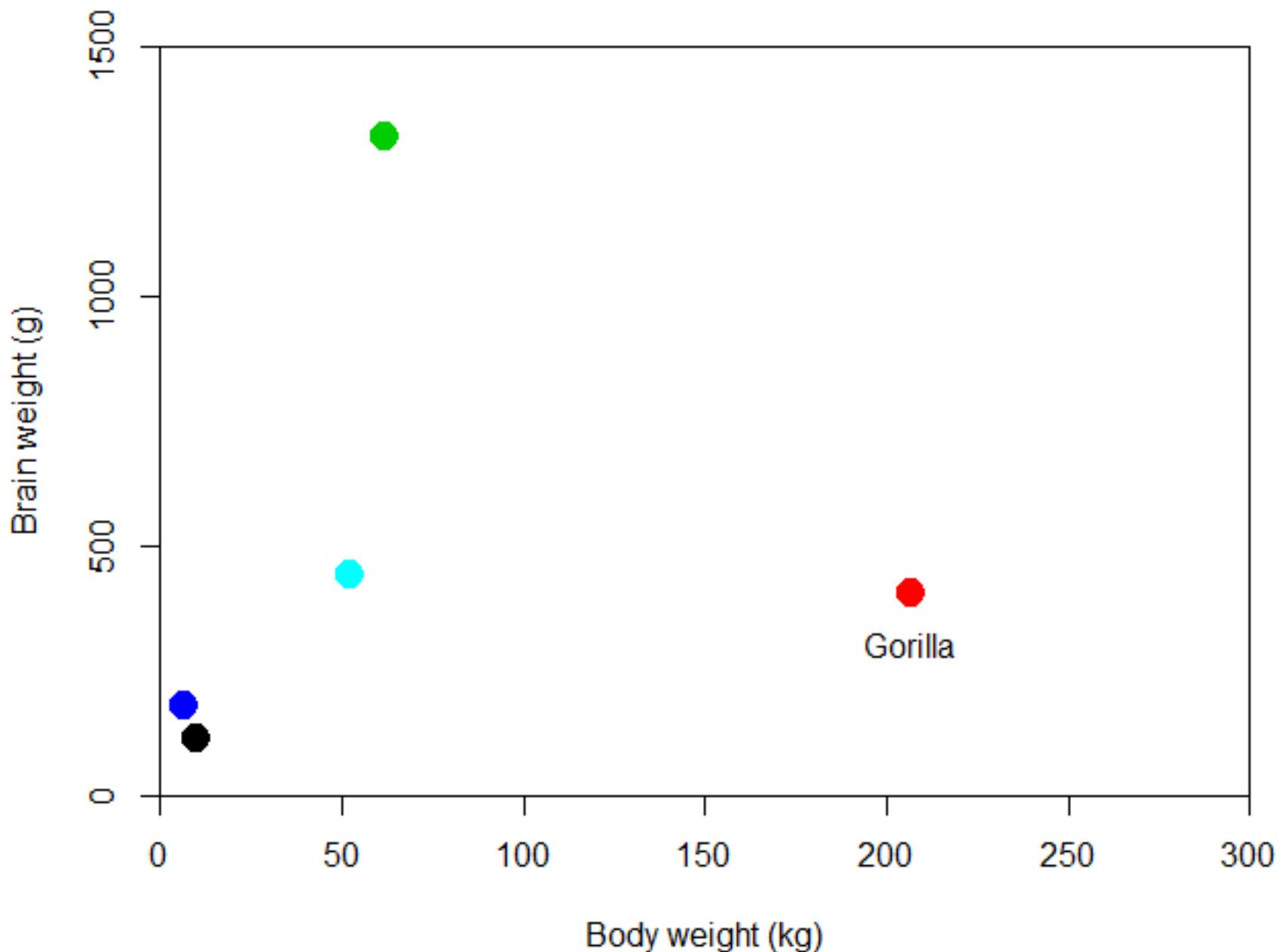
Si desea que la leyenda corresponda con el grafico, debe especificar símbolos, tamaños y colores idénticos para el grafico y la leyenda



Etiquetar los puntos con `text()`

- Añadir texto a los graficos utilizando `text`
- Los argumentos son x, y, y las etiquetas

```
text(x=207, y=306, label="Gorilla")
```



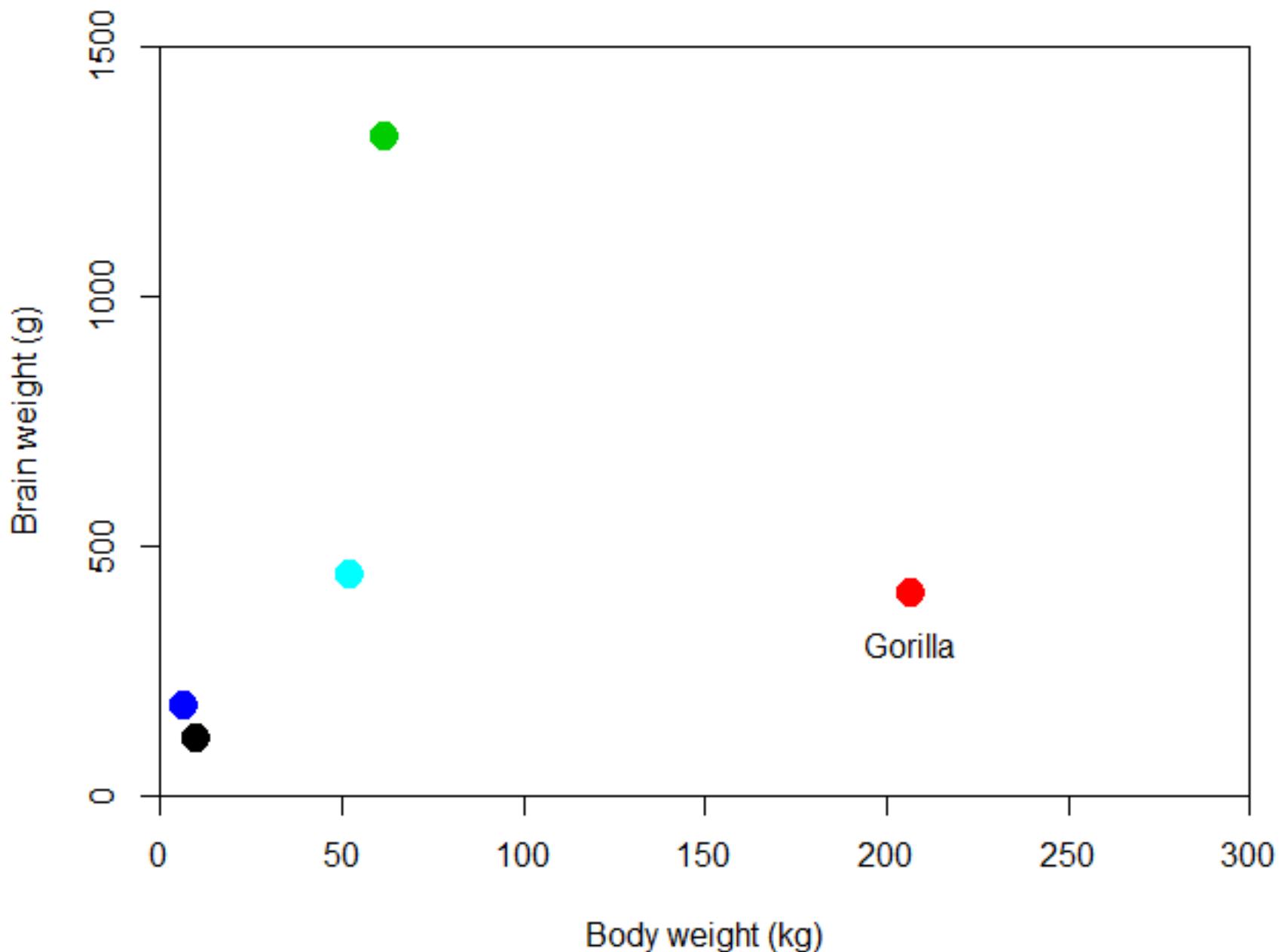
Adición de texto: usando `locator()`

- Función interactiva: clic en la gráfica y devuelve las coordenadas x e y

```
> locator(1)  Omita el 1 para varios clics, pulse <esc>para salir  
$x  
[1] 207.6493  
$y  
[1] 305.7384
```

- Añadir texto en esas coordenadas

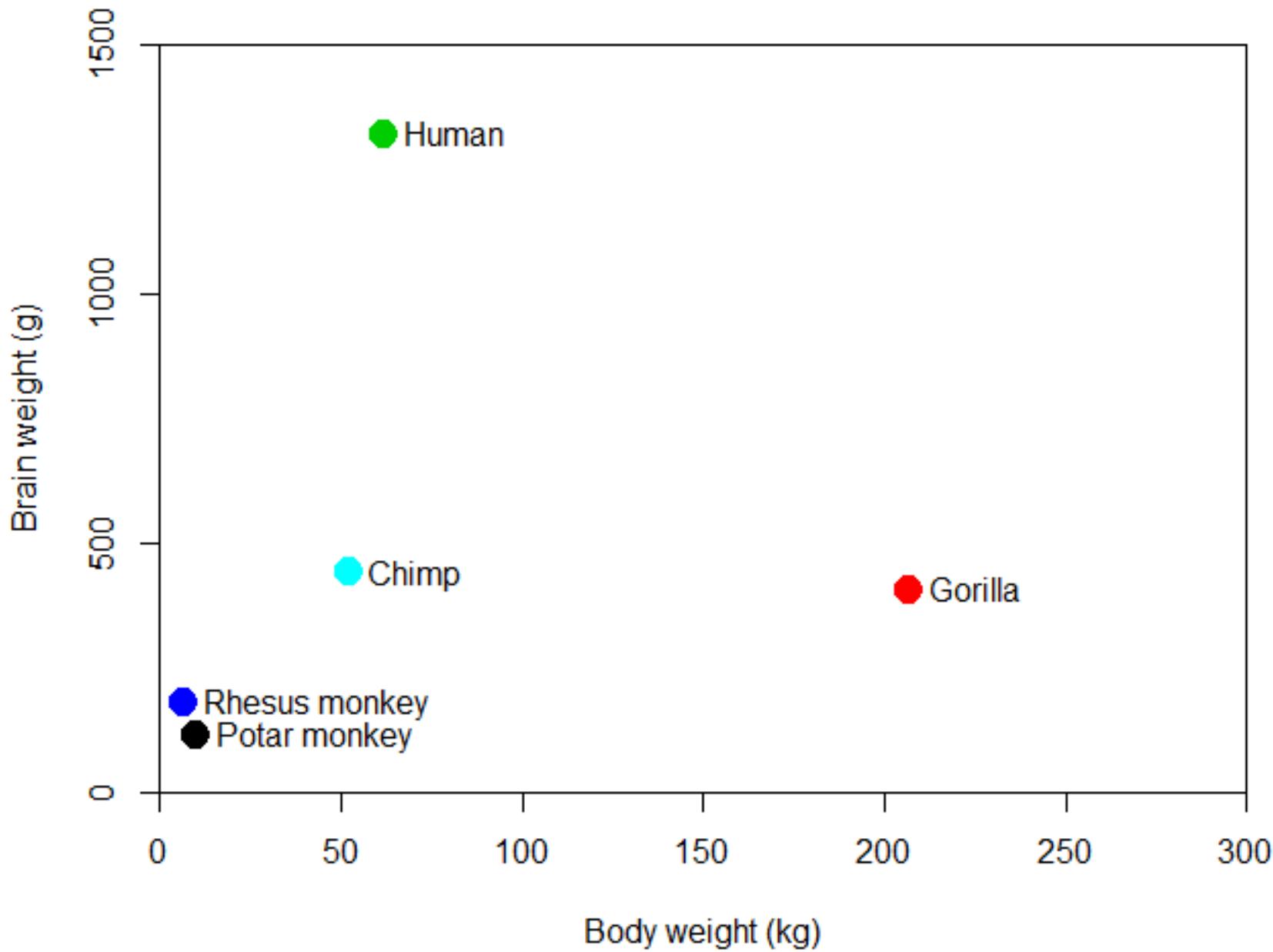
```
text(x=207, y=306, label="Gorilla")
```



Etiquetar los puntos con `text()`

- También puede utilizar ubicaciones de puntos
- Después de crear el grafico, use `text()`
 - `pos=1` Abajo
 - `pos=2` a la izquierda
 - `pos=3` encima
 - `pos=4` A la derecha
 -

```
text(x = primates$Bodywt, y = primates$Brainwt,  
      labels = primates[,1], pos=4)
```



Adición de puntos y líneas

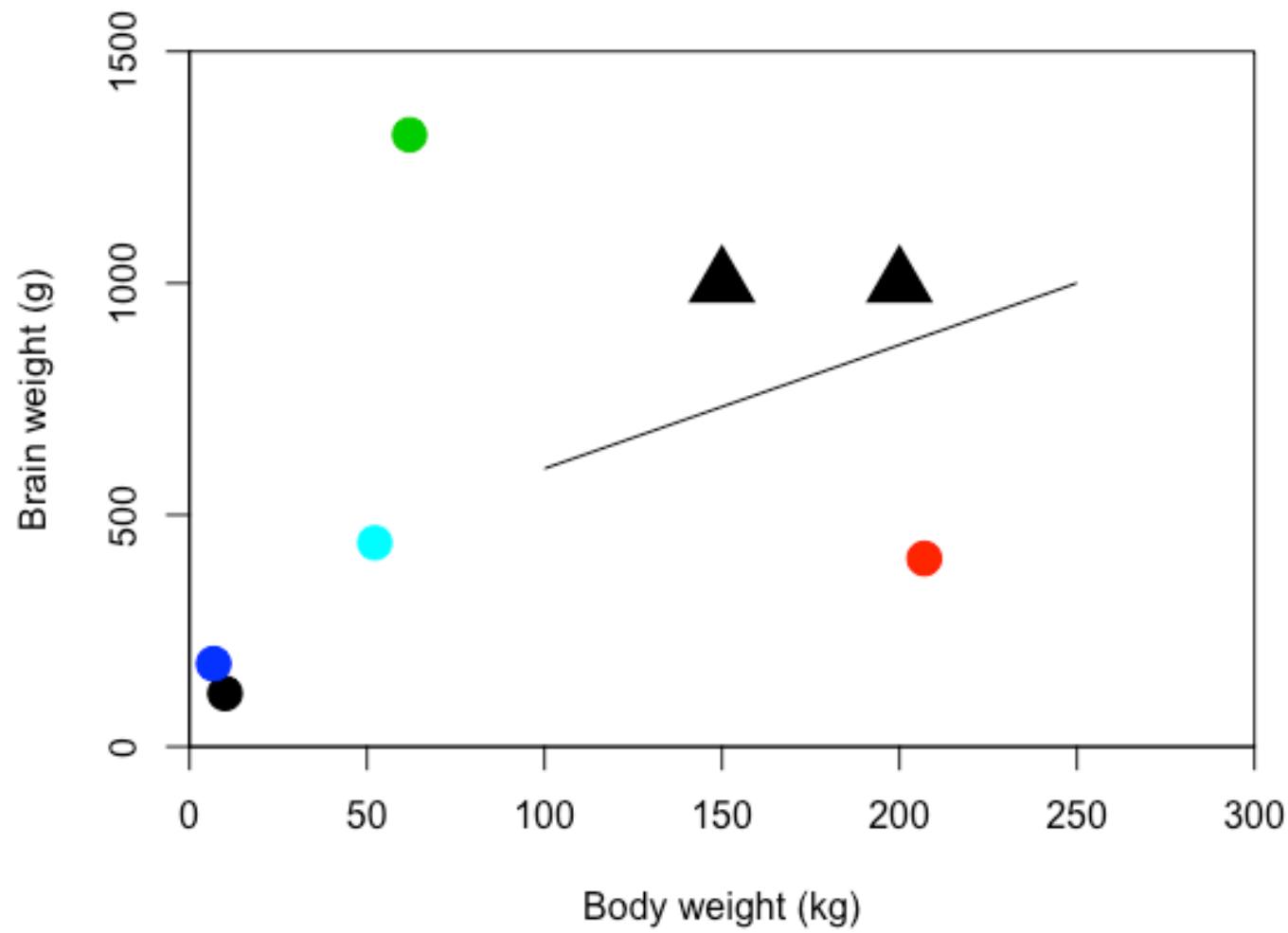
- Puede añadir una serie de puntos o líneas usando `points()` y `lines()`

#añadir una línea

```
lines(x=c(100,250), y=c(600,1000))
```

#añadir algunos puntos

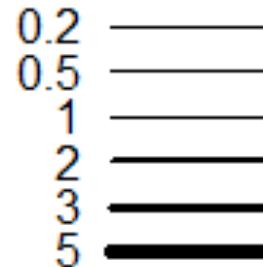
```
points(x=c(150,200), y=c(1000,1000), cex=3,  
pch=17)
```



lines()

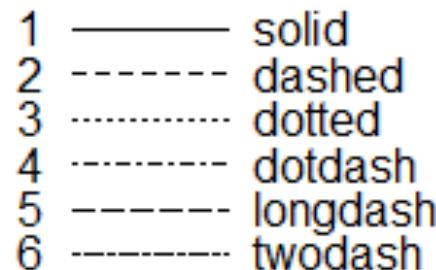
lwd = line widths

lwd values

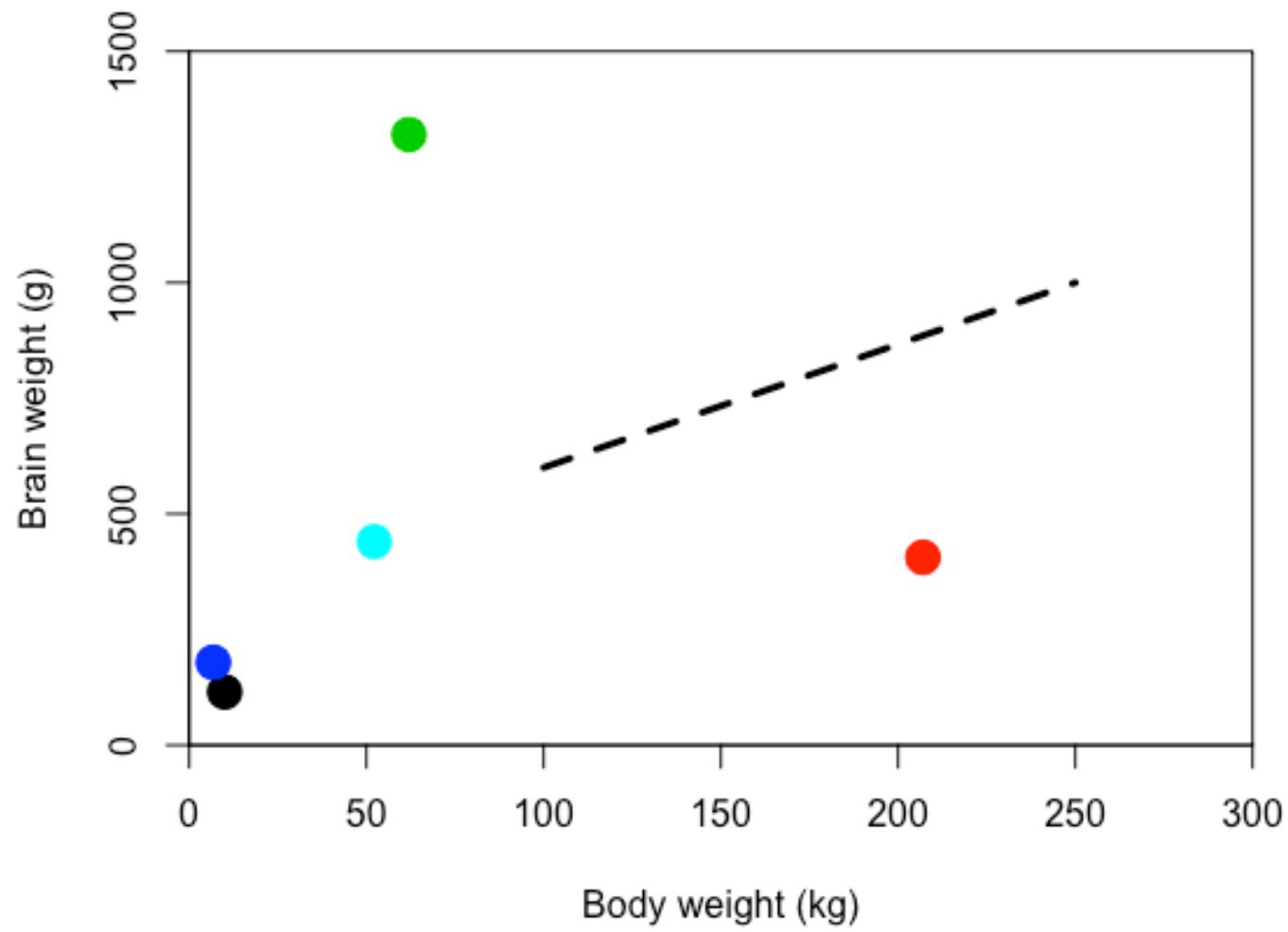


lty = line types

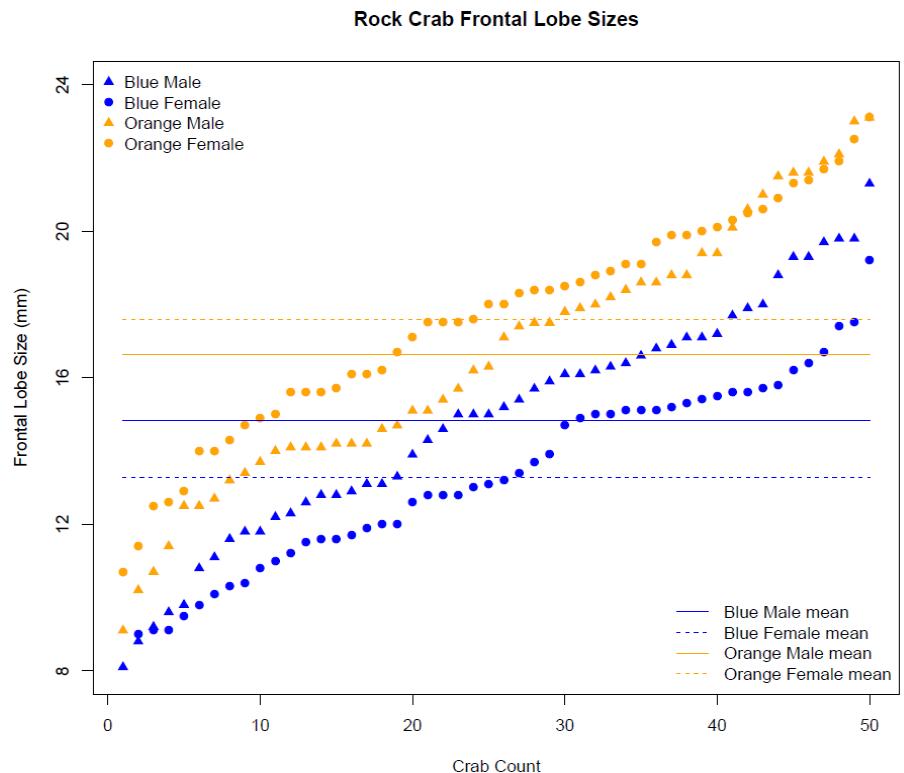
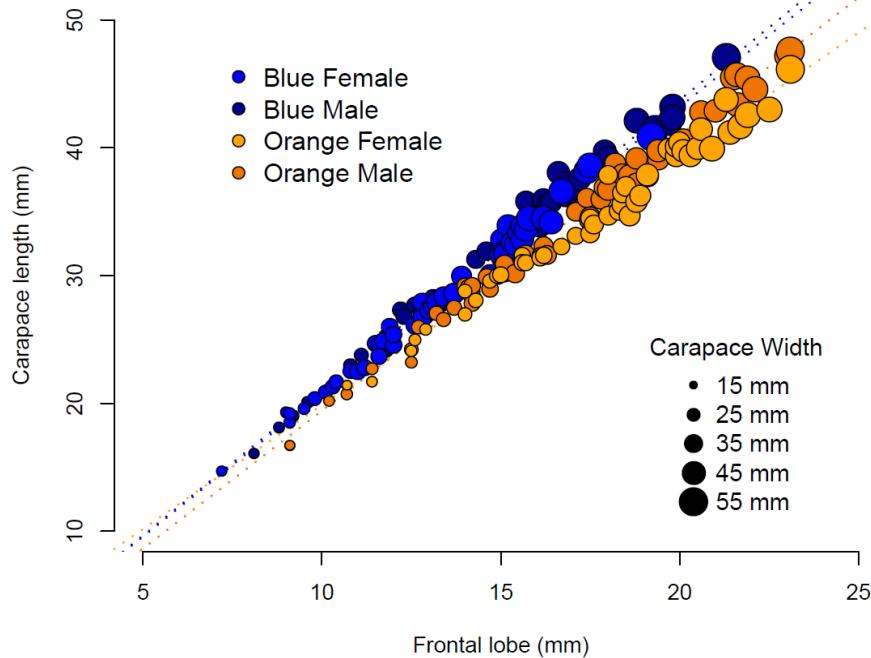
lty values



```
lines(x=c(100,250), y=c(600,1000), lwd=3, lty=2)
```



Líneas y puntos: plot(), lines(), points()



Puedes hacer todo tipo de cosas divertidas!

Lista completa de parámetros: `par()`

- sólo un puñado de comandos se enumeran con `?plot`
- Muchos comandos adicionales se enumeran con `?par.`
- Usar `par()` por sí mismo aplica comandos a múltiples gráficos!!!
- Para ver la configuración actual de `par`:

```
> par()  
$xlog  
[1] FALSE  
...  
...
```

Puedes cambiar cosas como:

- Cambiar el eje x (`xlim`)
- Cambiar el eje y: (`ylim`)
- Añadir un contorno al grafico (`bty`)
- Rotar caracteres (`crt`)
- Cambiar la fuente (`font`)
- Cambiar el color de fondo (`fg`)
- Cambiar los márgenes (`mar`)

GOOGLEALOOOO!



par() Graphical Parameters

Visual cheat sheet for some plot parameters in R. See [?par](#) for more information.

Symbol Styles

pch Point Types	lty Line Types
○ 1	□ 14
△ 2	■ 15
+	● 16
×	▲ 17
◇ 5	◆ 18
▽ 6	● 19
▣ 7	● 20
* 8	○ 21
◊ 9	□ 22
⊕ 10	◆ 23
⊗ 11	▲ 24
田 12	▼ 25
⊗ 13	you can also use any character

Figures Arrangement

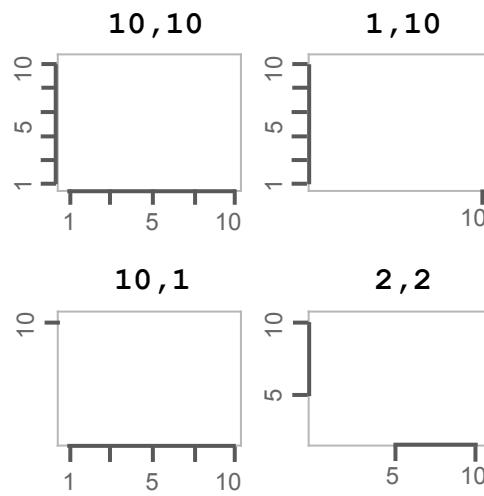
mfrow | Multiple Figures by Row
2,3



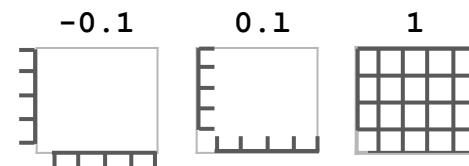
Also available [mfcol](#) for multiple figures by column

Axes

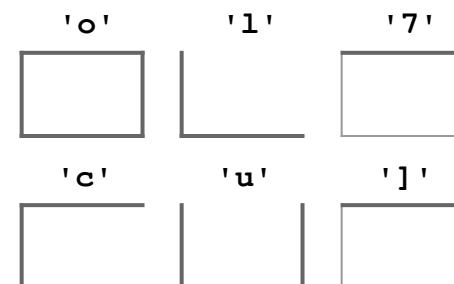
lab | Tick Placement



tck | Tick Length



bty | Box Type



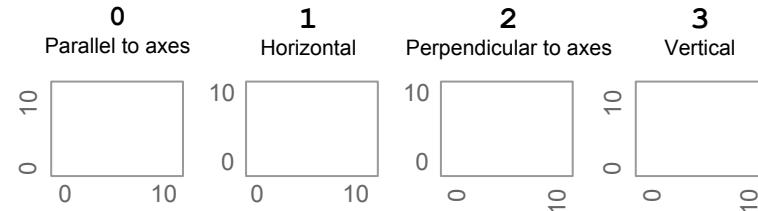
Text and Labels

family, font | Typeface and Font Style

family: mono	family: serif	family: sans
font: 1	font: 1	font: 1
family: mono	family: serif	family: sans
font: 2	font: 2	font: 2
family: mono	family: serif	family: sans
font: 3	font: 3	font: 3
family: mono	family: serif	family: sans
font: 4	font: 4	font: 4

Also available: [font.main](#) (main title), [font.lab](#) (axis labels), [font.sub](#) (subtitle)

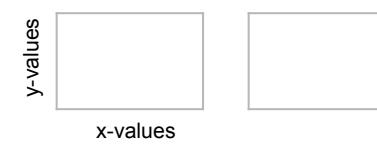
las | Label Orientation



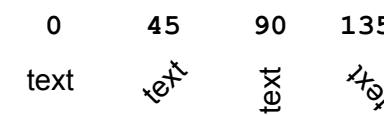
ann | Plot Annotation

TRUE FALSE

Some Title



srt | String Rotation



lheight | Line Height

1

The quick brown fox jumps
over the lazy dog and runs
away with all the food

1.5

The quick brown fox jumps
over the lazy dog and runs
away with all the food

Based on Flowing
Data's cheat sheet

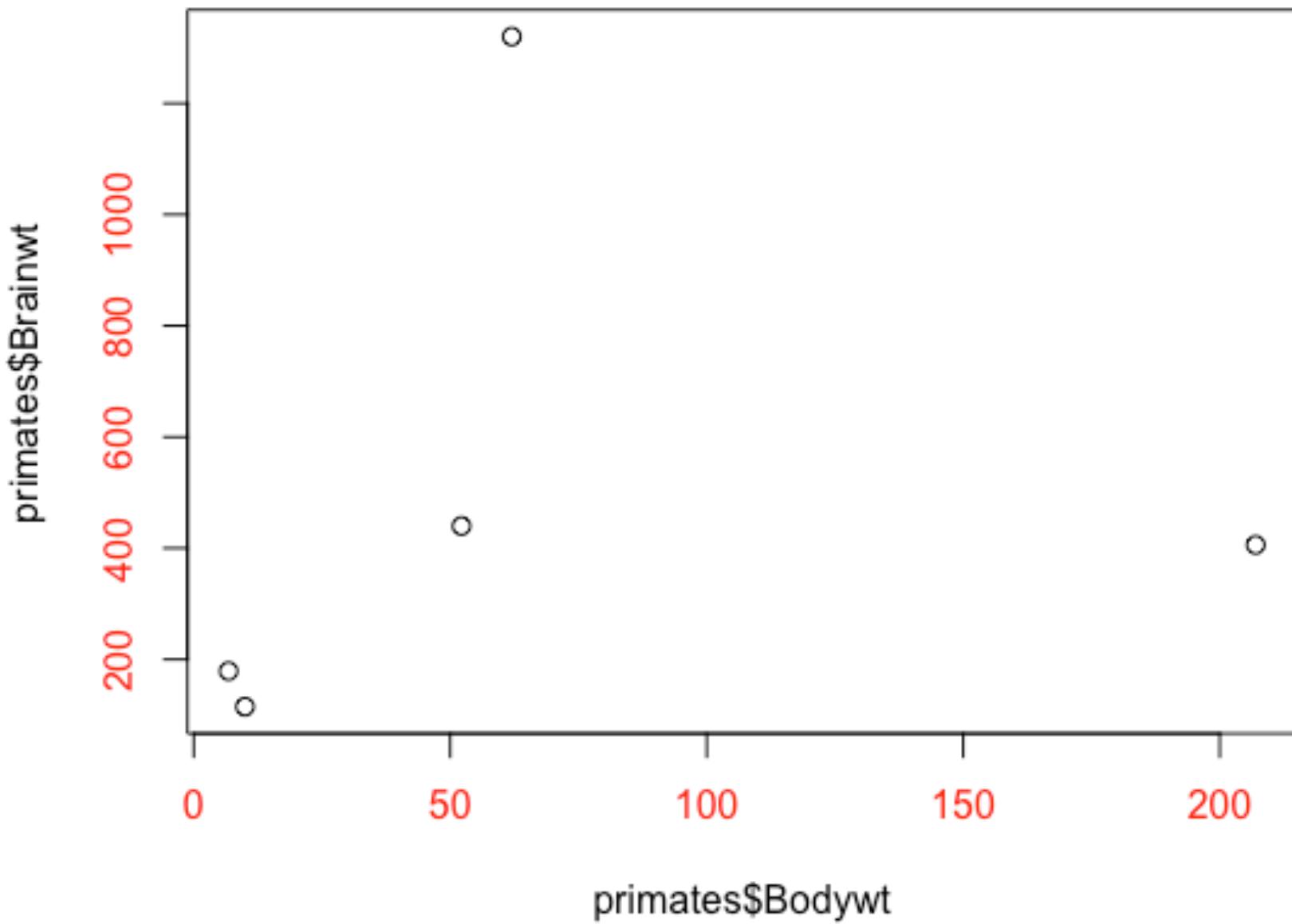
`par()` para los cambios globales

- Guardar los parámetros de `par`

```
old.par <- par()
```

- Cambiar a un nuevo valor
- `par(col.axis="red")`

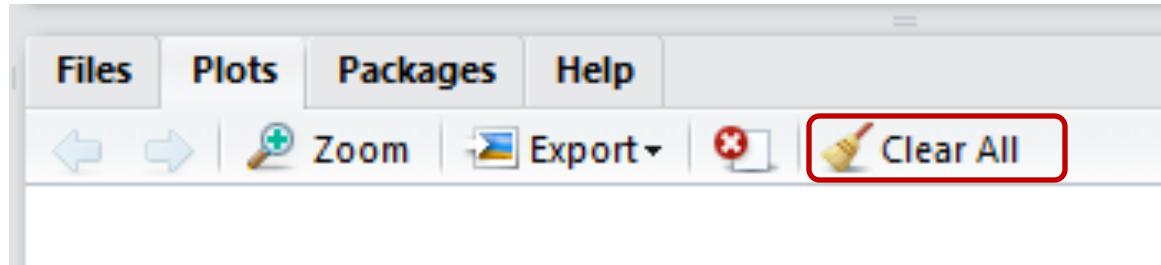
```
plot(x=primates$Bodywt, y=primates$Brainwt)
```



Para volver a valores predeterminados

Restore defaults:

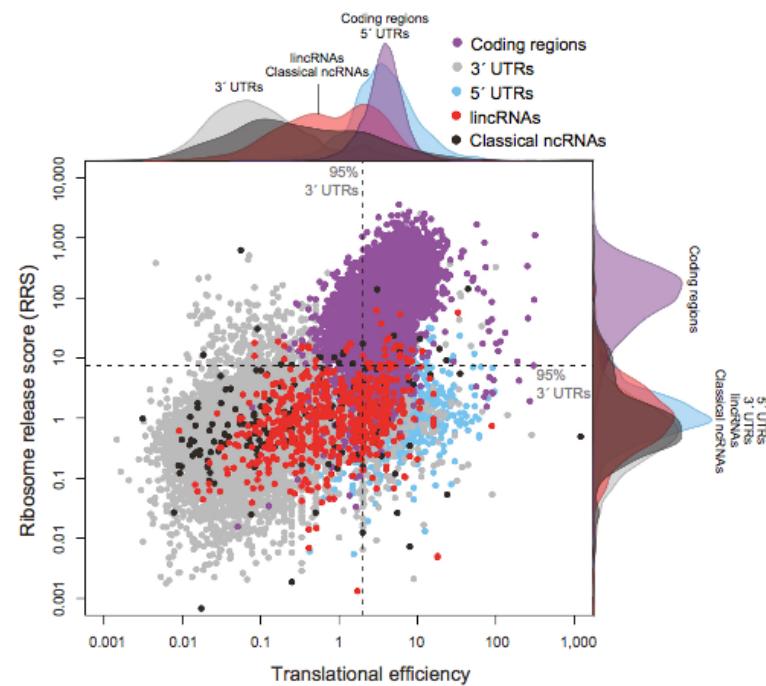
```
par(old.par)
```



Borra todas las figuras y restablece los valores
predeterminados - par()

Gráficos en R

- **Inspiración**
- **Gráficos base en R**
 - Gráfica de puntos
 - Manipulación de graficas
 - Otros tipos básicos de graficas**
 - Guardar graficas



Más tipos de graficos usando plot()

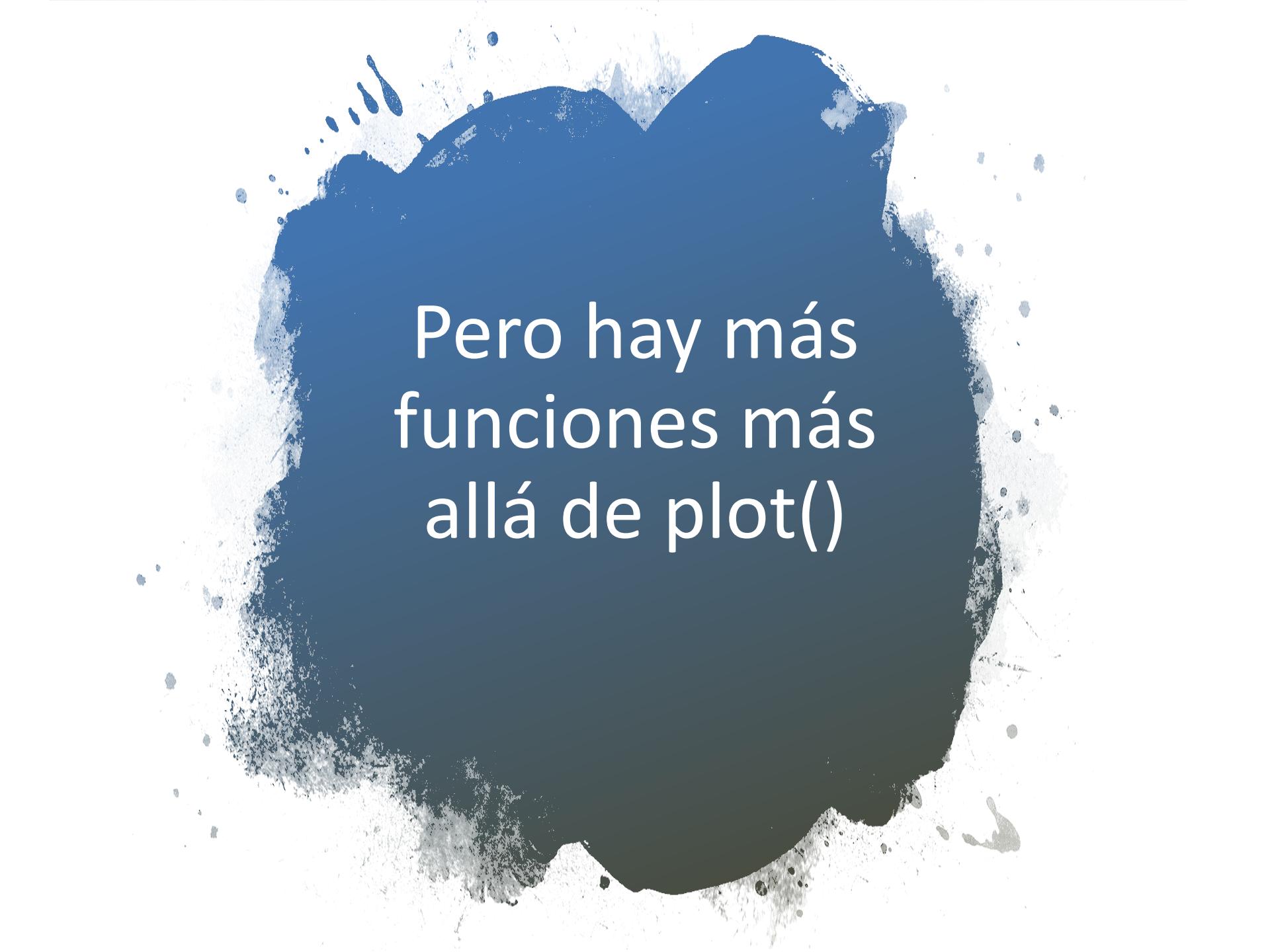
- Con `plot()`, `type` especifica el tipo de grafica:

- "p" Puntos
- "l" Líneas
- "b" tanto líneas como puntos
- "c" líneas sin puntos
- "o" sobreplotado
- "h" líneas verticales similares a histogramas (o de alta densidad)
- "s" escalera
- "n" para no trazar

Prueba esto:

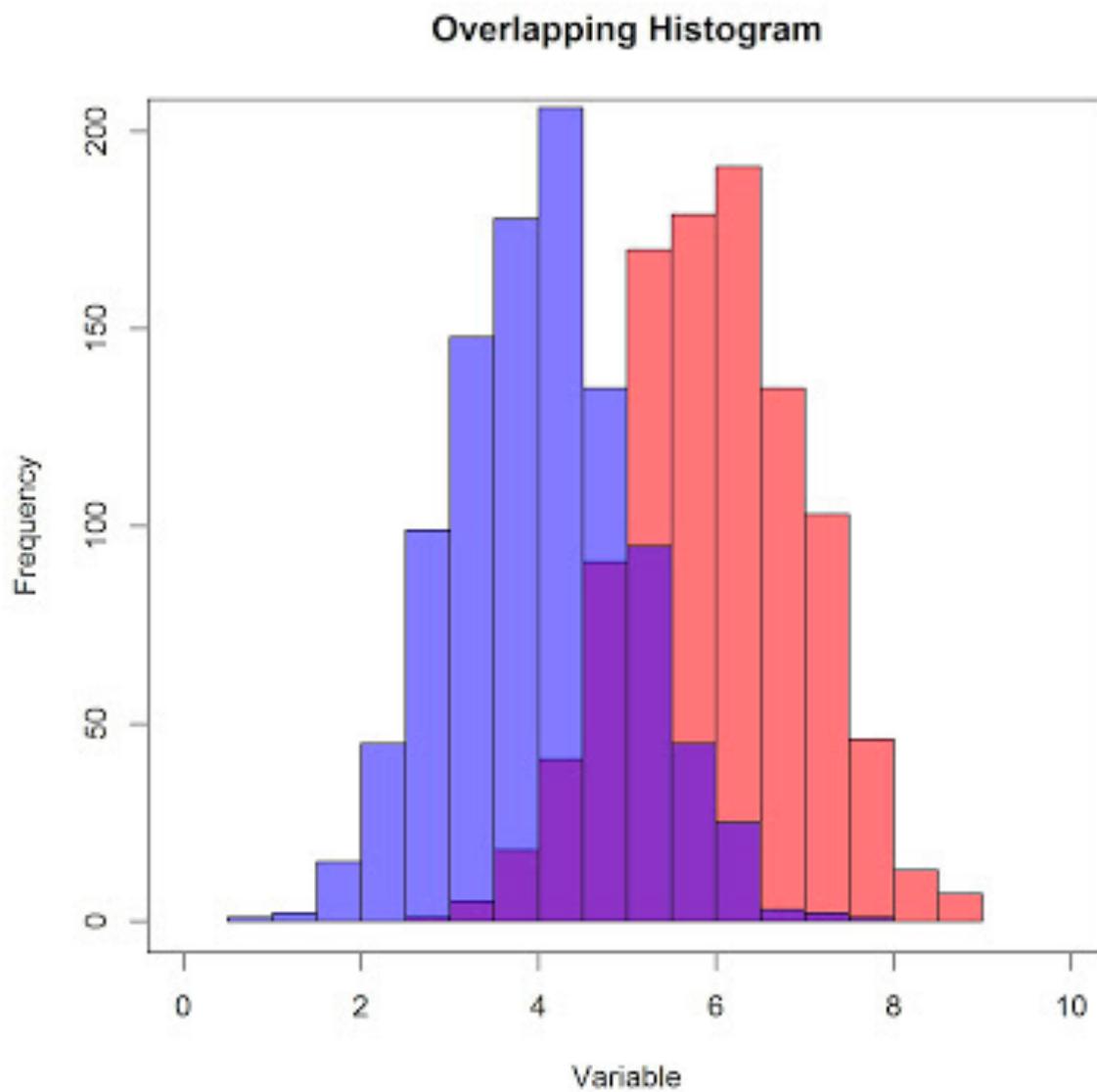
```
plot(x = primates$Bodywt, y = primates$Brainwt, type="p")
```

... Ahora cambia el tipo, probando cada una de las opciones!



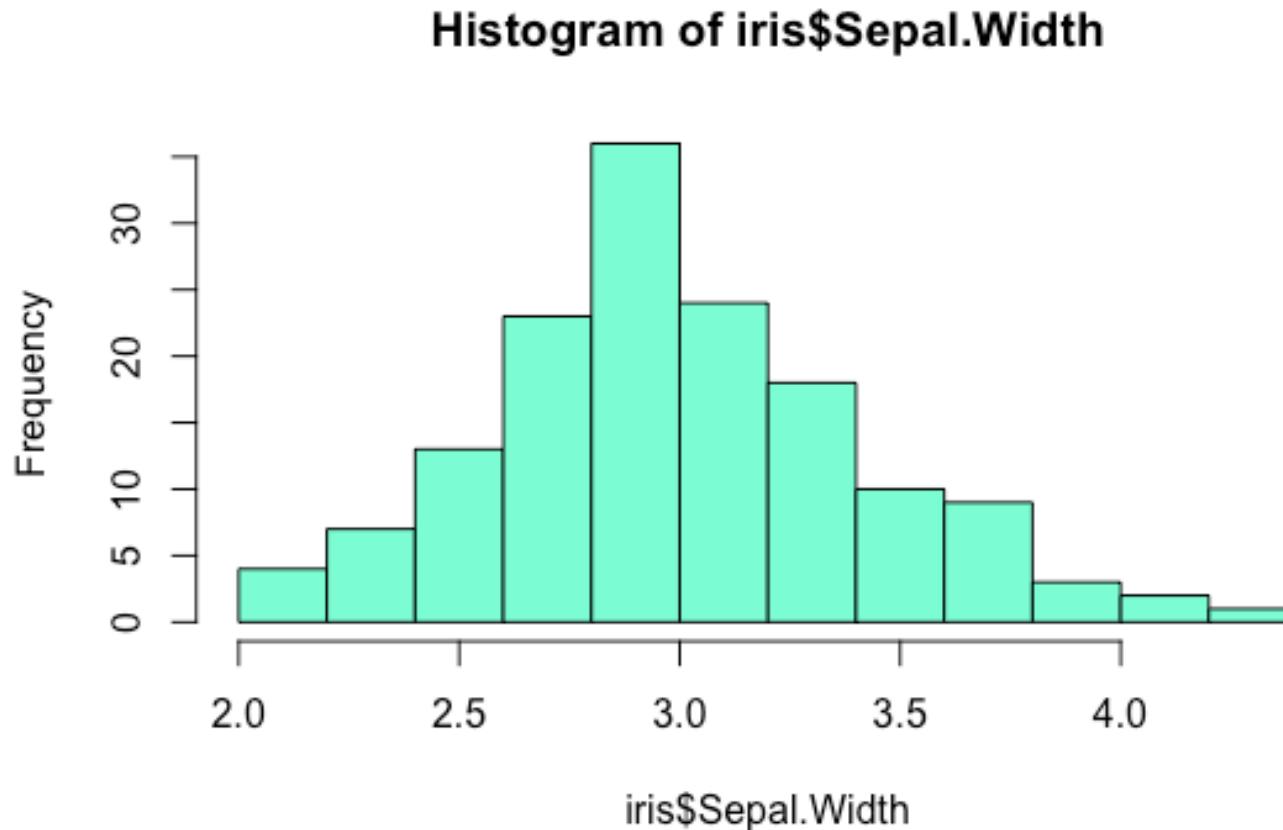
Pero hay más
funciones más
allá de plot()

Histogramas: `hist()`

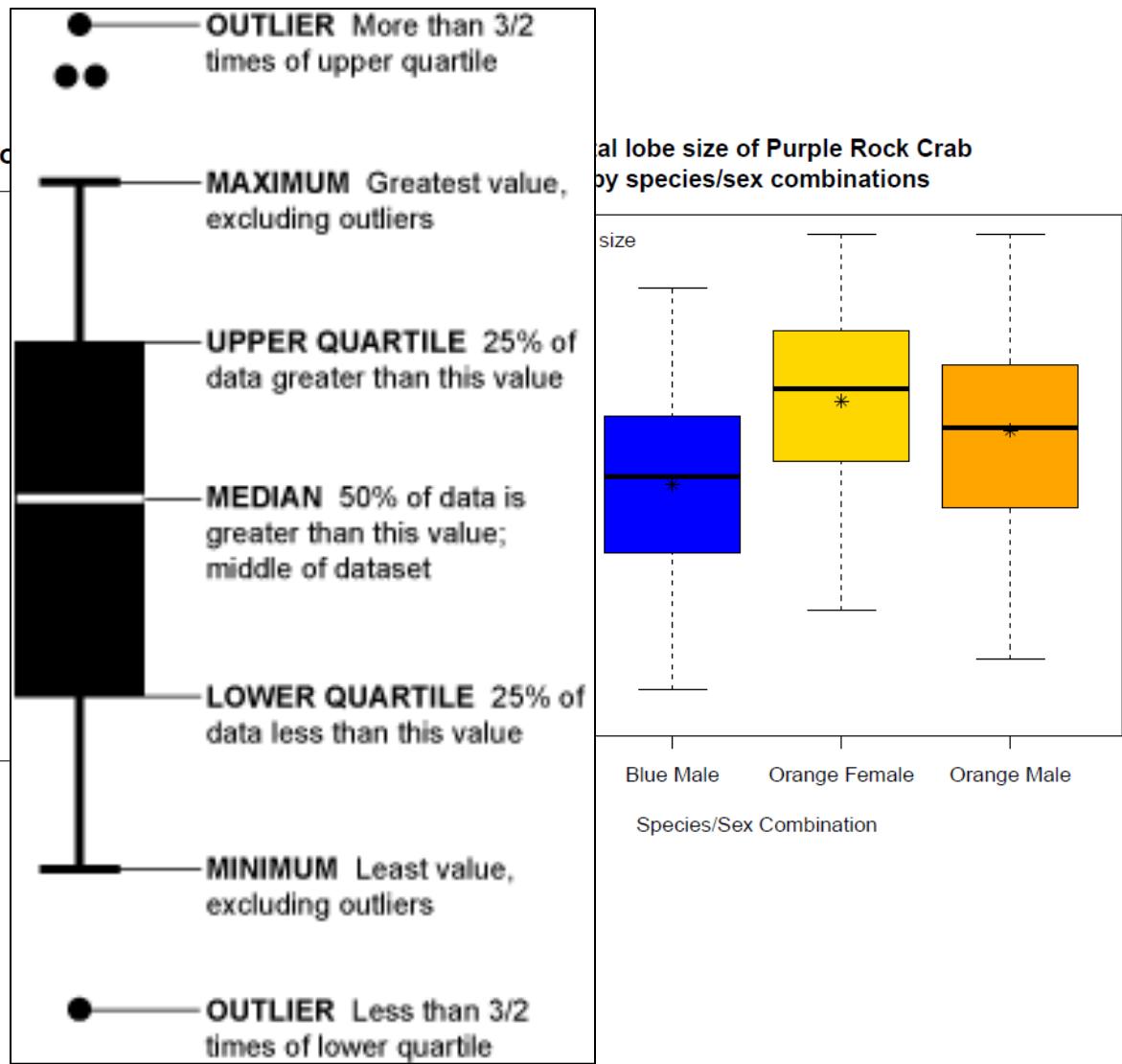
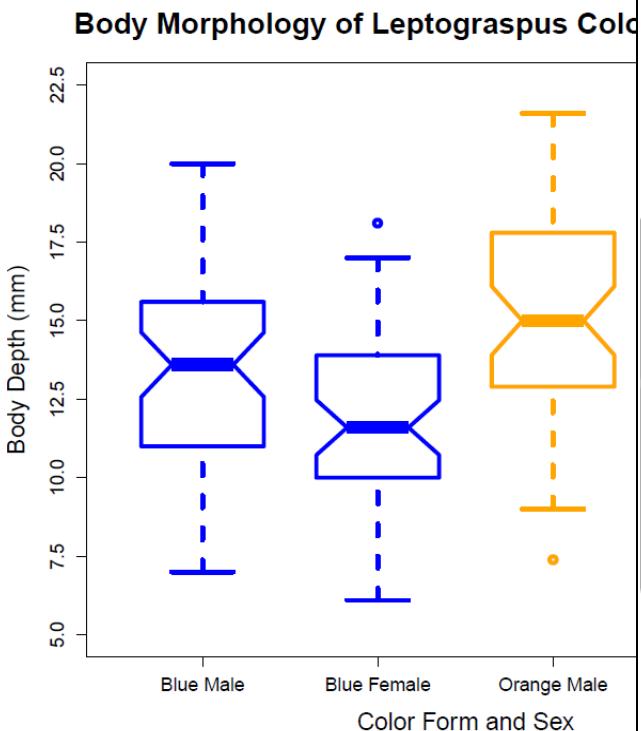


Histogramas: `hist()`

```
hist(iris$Sepal.Width, col="aquamarine")
```



Box plots: `boxplot()`



Boxplots

Los datos deben ser una tabla de datos con números y factores

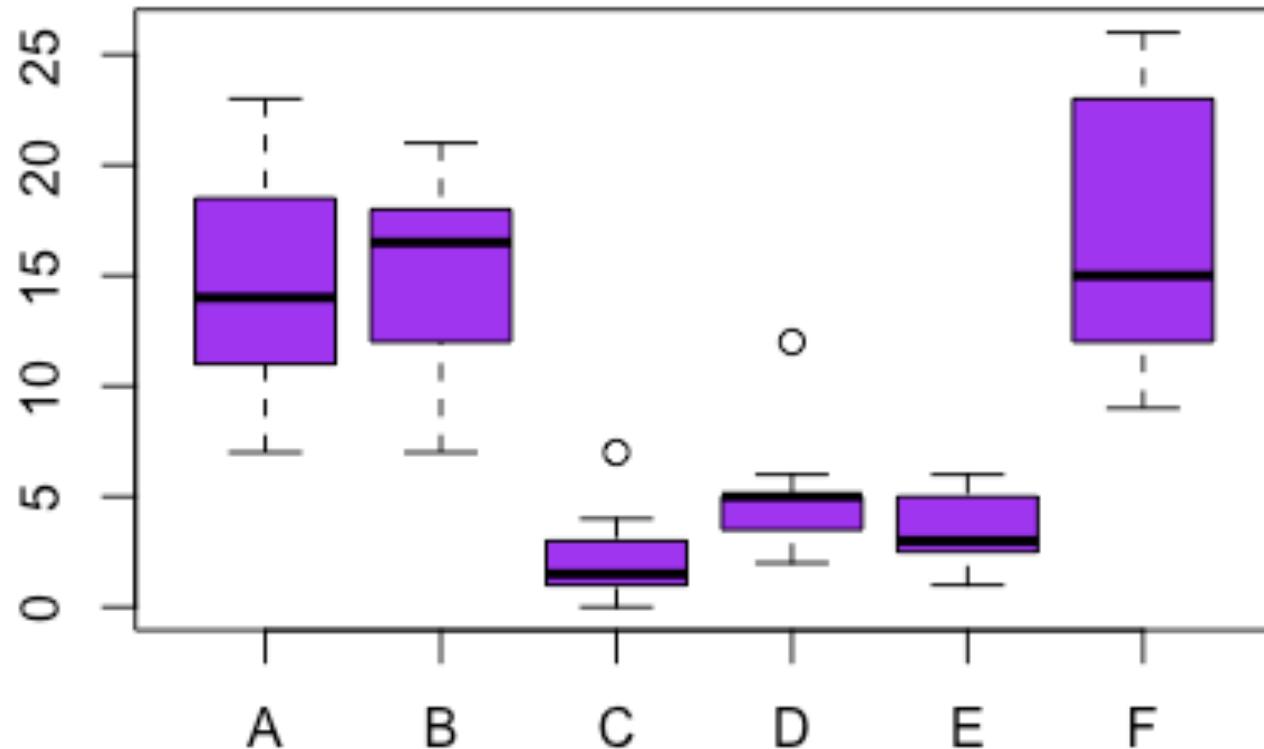
```
> str(InsectSprays)
```

'data.frame': 72 obs. of 2 variables:

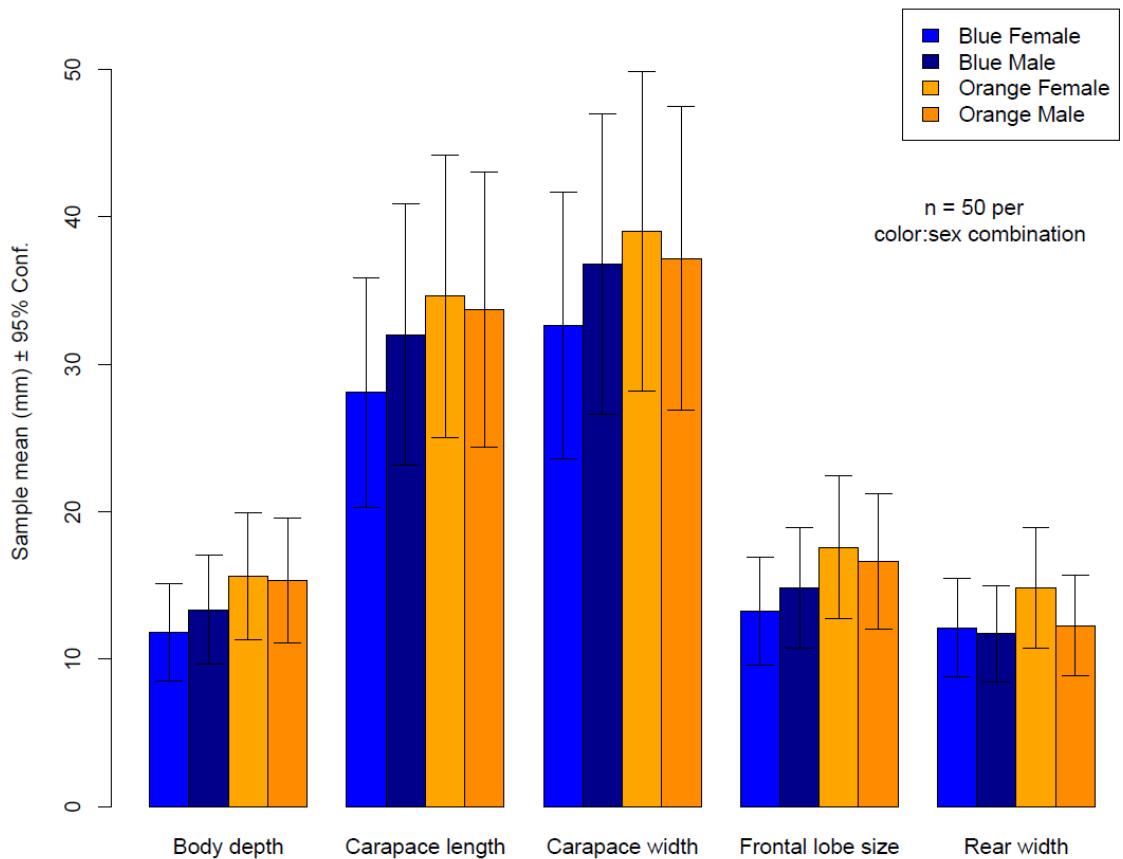
\$ count: num 10 7 20 14 14 12 10 23 17 20 ...

\$ spray: Factor w/ 6 levels "A","B","C","D",...: 1 1 1 1 1 1 1 1 1 1

```
boxplot(count ~ spray, data = InsectSprays, col = "purple")
```



Bar plots: barplot()



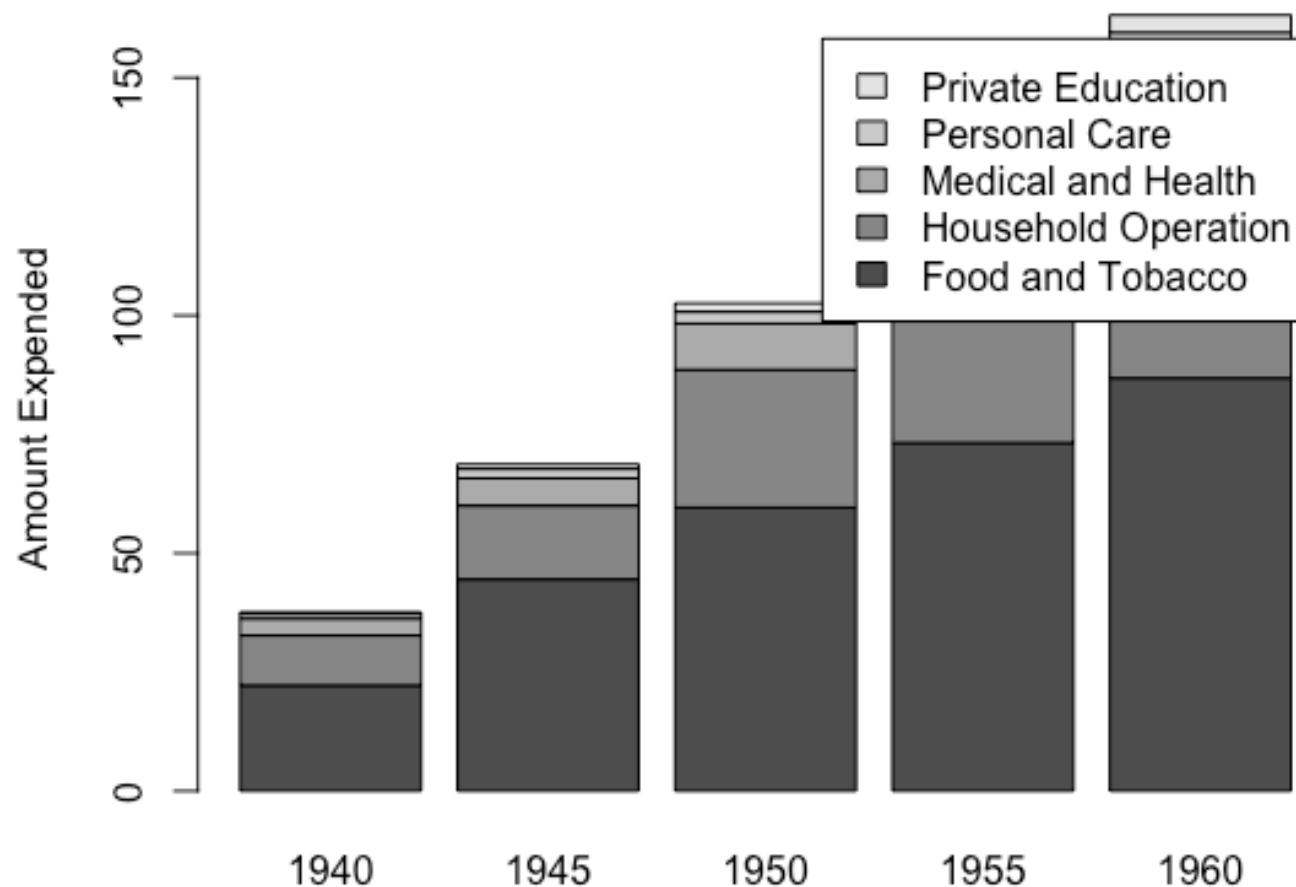
barplot()

- Los datos deben ser un vector o una tabla
- Los nombres de fila y los nombres de columna se utilizan de forma predeterminada

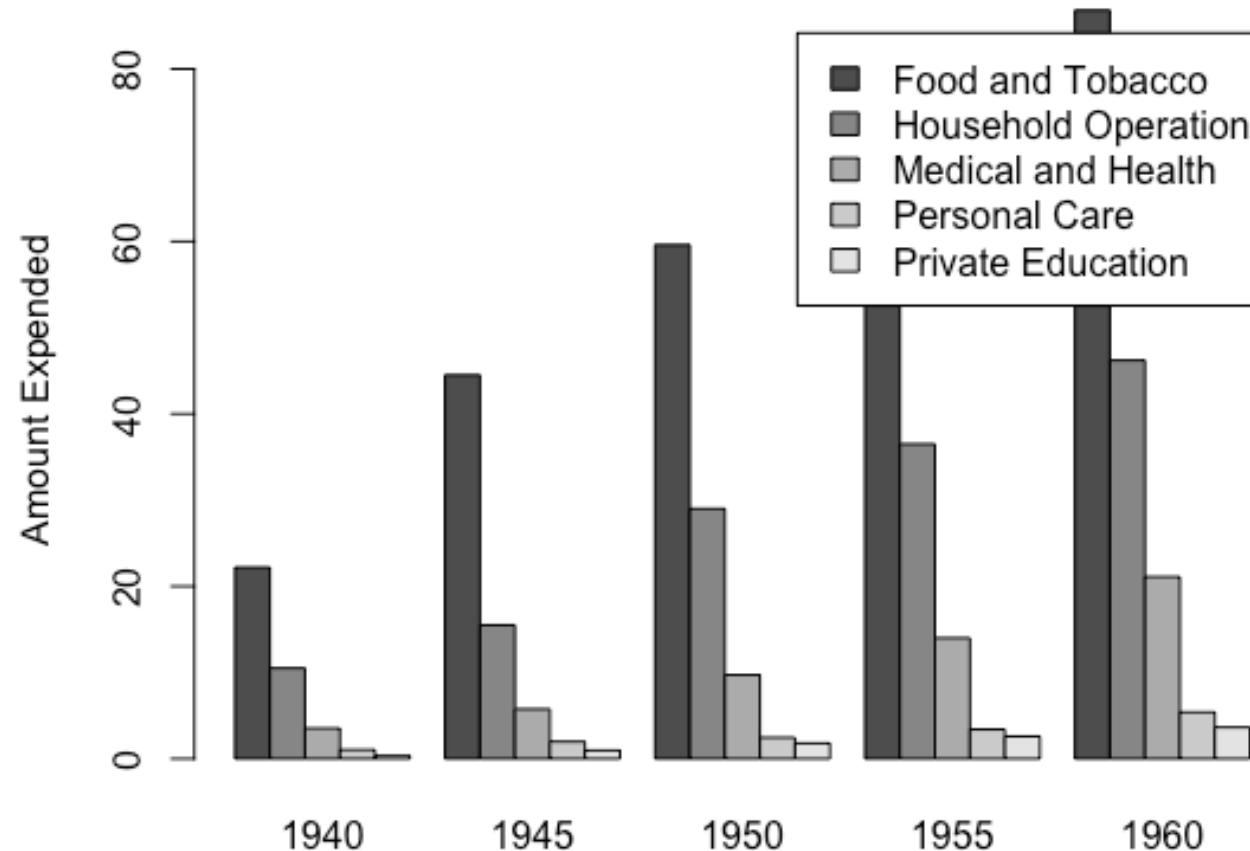
> [USPersonalExpenditure](#)

	1940	1945	1950	1955	1960
Food and Tobacco	22.200	44.500	59.60	73.2	86.80
Household Operation	10.500	15.500	29.00	36.5	46.20
Medical and Health	3.530	5.760	9.71	14.0	21.10
Personal Care	1.040	1.980	2.45	3.4	5.40
Private Education	0.341	0.974	1.80	2.6	3.64

```
> barplot(USPersonalExpenditure,  
legend=TRUE, ylab="Amount Expended")
```

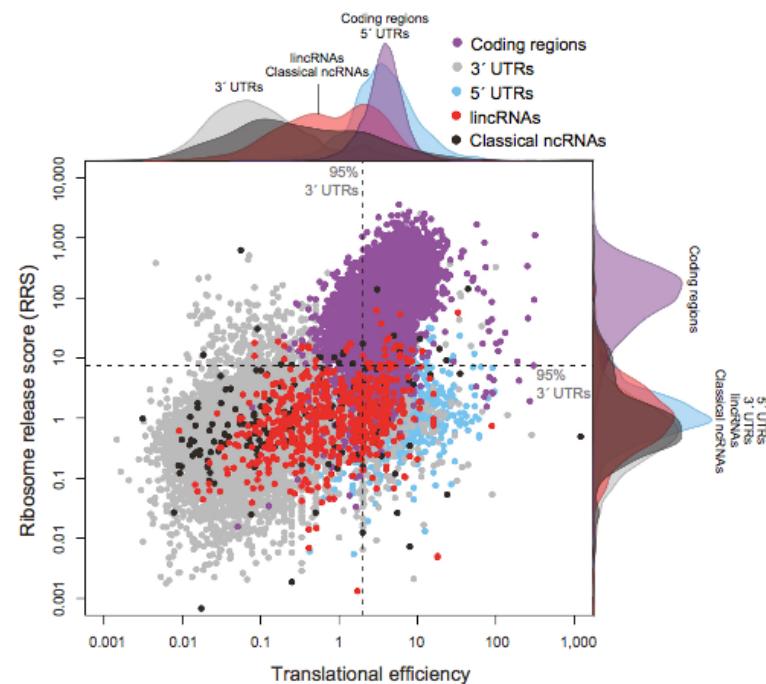


```
> barplot(USPersonalExpenditure,  
legend=TRUE, ylab="Amount Expended",  
beside=TRUE)
```



Gráficos en R

- **Inspiración**
- **Gráficos base en R**
 - Gráfica de puntos
 - Manipulación de graficas
 - Otros tipos básicos de graficas
 - Guardar graficas**



Dos formas de guardar graficas:

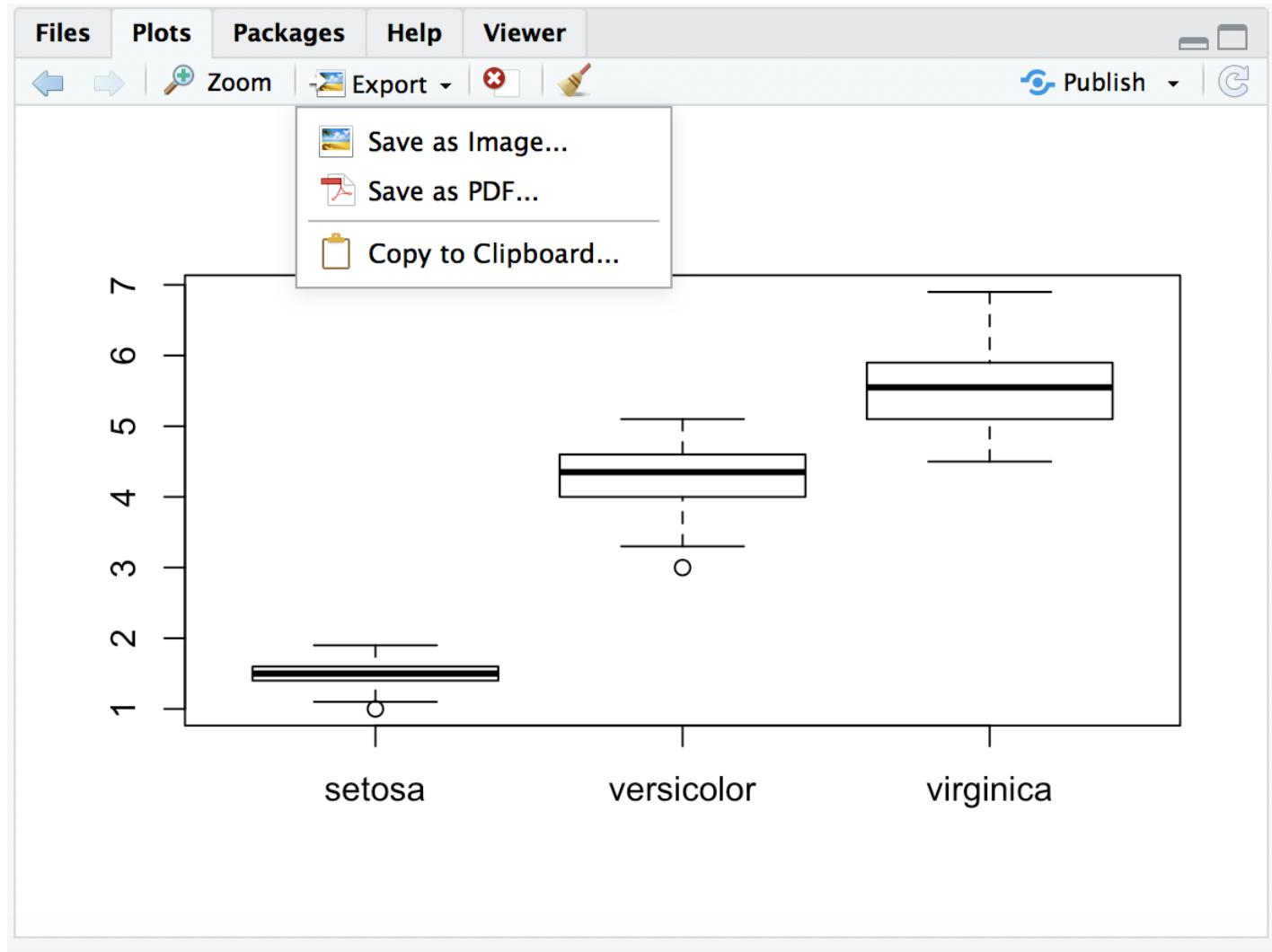
(1) Clic

(2) Código

Regla: Las graficas exploratorias en Rstudio están bien, pero guardar las graficas finales con Código es mejor

Guardar las graficas en R studio

Plots panel → Export → Save as Image or Save as PDF





Guardar las graficas en R studio

- Su resolución dependerá de la configuración de la pantalla
- Graficas no reproducibles.
- No seas perezoso como este conejito.

Dos formas de guardar graficas:

(1) clic

(2) Código

Regla: Las graficas exploratorias en Rstudio están bien, pero guardar las graficas finales con Código es mejor

Código

Guardar graficas con código, garantiza que las gráficas sean reproducibles

- Tres pasos:
 1. Abra la conexión al nuevo archivo
 2. Hacer el grafico
 3. Cierre la conexión



Tres pasos:

```
pdf(file= "myplot.pdf") 1. Abrir la conexión  
plot(Sepal.Width~Sepal.Length, data=iris)  
dev.off() 3. Cerrar la conexión
```

2. Hacer el grafico

Funciones para guardar las graficas:

- `pdf()` puede acercar, múltiples páginas
- `png()` tiene un pequeño tamaño de archivo sin pérdida de calidad
- `tiff()` Archivos grandes
- `jpeg()` tamaños de archivo pequeños, pérdida de calidad (No lo usen!)
- `bmp()` tamaños de archivo colosales, sin compresión (No lo usen!)



Lo que hemos aprendido hasta ahora

- **Inspiración**
- **Gráficos base en R**

- Gráfica de puntos
- Manipulación de graficas
- Otros tipos básicos de graficas
- Guardar graficas

