

MUSHROOM4ALL

INFORME DE PROYECTO

Autores:

Abraham Carrera Groba

Miguel Magaña Suanzes

Mushrooms4All

Abraham Carrera, Miguel Magaña

Introducción

Mushrooms4all es una empresa encargada de fomentar la venta y recolección de setas. Uno de los problemas del sector es la existencia de setas no aptas para el consumo. Por suerte, disponen de una gran base de datos con información sobre distintas características de las setas y su aptitud (o no) para el consumo humano.

Sin embargo, les gustaría ser capaces de predecir a partir de la base de conocimiento si una seta desconocida es o no venenosa simplemente identificando sus características. Para ello solicitan la creación de una aplicación que mediante la creación de un modelo de los datos disponibles lleve a cabo dicha predicción.

Entendimiento y preparación de los datos

Descripción de los datos

En primer lugar, tras recibir de manos de Mushrooms4All el dataset, lo importamos así como el conjunto de librerías que vamos a utilizar.

Hide

```
suppressMessages(library(caret))
suppressMessages(library(ggplot2))
suppressMessages(library(dplyr))
suppressMessages(library(gridExtra))
suppressMessages(library(gmodels))
suppressMessages(library(ggparallel))
suppressMessages(library(rpart.plot))
suppressMessages(library(sqldf))
suppressMessages(library(tidyverse))
suppressMessages(library(vcd))
suppressMessages(library(GoodmanKruskal))
suppressMessages(library(dismo))
suppressMessages(library(MLmetrics))
suppressMessages(library(ROCR))
suppressMessages(library(e1071))
suppressMessages(library(randomForest))
setas<-read.csv("../inputs/mushrooms.csv", header = TRUE)
```

A continuación se analiza la apariencia de nuestros datos:

- Consisten en 8124 observaciones de 23 variables
- Todas las variables son categoricas con letras como valores
- Las variables son las siguientes:
 - cap-shape: forma del gorro.
 - cap-surface: superficie del gorro.
 - cap-color: color del gorro.
 - bruises: si tiene manchas.
 - odor: olor.

- gill-attachment: unión de las agallas.
- gill-spacing: espacio entre las agallas.
- gill-size: tamaño de las agallas.
- gill-color: color de las agallas.
- stalk-shape: forma del tallo.
- stalk-root: raíz del tallo.
- stalk-surface-above-ring: tipo de superficie sobre el anillo.
- stalk-surface-below-ring: tipo de superficie por debajo del anillo.
- stalk-color-above-ring: color sobre el anillo.
- stalk-color-below-ring: color bajo el anillo.
- veil-type: tipo de velo.
- veil-color: color del velo.
- ring-number: número de anillos.
- ring-type: tipo de anillo.
- spore-print-color: color de las esporas.
- population: tamaño de la población.
- habitat: lugar en el que crecen.

Hide

```
head(setas)
```

cla... <fctr>	cap.shape <fctr>	cap.surface <fctr>	cap.color <fctr>	bruises <fctr>	o... <fctr>	gill.attachment <fctr><fctr>	gill.spacing <fctr>	gil <fctr>
1 p	s	f	g	t	p	f	d	n
2 e	x	g	y	t	a	d	c	b
3 e	k	s	n	t	l	f	w	b
4 p	b	y	w	t	y	f	c	n
5 e	b	s	g	f	n	n	c	b
6 e	k	s	y	f	a	f	w	n

6 rows | 1-10 of 23 columns

Hide

```
dim(setas)
```

```
[1] 8124 23
```

Hide

```
glimpse(setas)
```

Observations: 8,124

Variables: 23

```
$ class <fct> p, e, e, p, e, e, e, e, p, e, e, e, e, p, e, e, e, p, p, p,
e, p, e, e, e, p, e, e, e, e, e, p, e, e, e, e, e, ...
$ cap.shape <fct> s, x, k, b, b, k, f, b, s, b, x, c, b, x, s, s, f, x, b, s,
c, x, c, b, b, f, f, k, b, f, b, x, k, f, b, x, s, ...
$ cap.surface <fct> f, g, s, y, s, s, f, y, y, s, s, g, g, g, f, g, f, f, y, g,
s, y, y, y, s, g, y, y, s, s, s, y, y, y, g, s, f, ...
$ cap.color <fct> g, y, n, w, g, y, w, w, y, y, r, y, c, w, e, r, w, n, w, y,
p, g, y, w, w, r, e, u, r, y, p, w, r, e, n, c, g, ...
$ bruises <fct> t, t, t, t, f, f, t, t, f, t, t, t, t, t, t, t, t, t, f, t,
f, t, t, t, f, t, t, t, f, t, t, t, t, t, f, f, ...
$ odor <fct> p, a, l, y, n, a, a, l, p, a, l, p, a, c, m, n, y, p, y, m,
a, y, l, a, l, c, a, l, n, a, l, c, f, l, l, l, n, ...
$ gill.attachment <fct> f, d, f, f, n, f, a, f, f, d, n, d, f, f, a, d, a, f, n, d,
n, d, d, a, f, d, a, f, f, f, f, f, f, f, f, a, ...
$ gill.spacing <fct> d, c, w, c, c, w, c, c, d, c, d, w, c, d, w, w, w, c, c, c,
w, c, w, w, c, d, c, w, c, c, c, d, w, c, c, d, c, ...
$ gill.size <fct> n, b, b, n, b, n, b, b, n, b, n, b, b, n, b, n, b, b, n, n,
n, n, b, b, b, n, b, b, n, n, b, b, b, b, b, n, b, ...
$ gill.color <fct> k, k, n, e, o, n, g, r, b, g, b, n, k, r, n, k, k, p, n, h,
p, h, p, w, o, b, n, w, k, h, g, u, w, e, n, u, k, ...
$ stalk.shape <fct> e, e, t, t, t, t, e, t, t, e, e, e, e, e, e, e, e, t, e,
t, e, e, e, t, e, e, e, t, t, e, e, e, e, e, t, e, ...
$ stalk.root <fct> e, c, c, e, b, z, c, c, u, c, u, c, b, r, r, z, e, c, b, e,
c, e, c, z, b, e, e, c, e, e, r, u, z, r, b, u, e, ...
$ stalk.surface.above.ring <fct> s, y, s, s, s, s, s, s, s, y, s, y, y, y, f, s, f, s, s, s,
s, y, y, s, s, y, f, y, s, k, k, s, s, f, s, s, f, ...
$ stalk.surface.below.ring <fct> s, y, k, k, s, s, f, s, s, y, s, y, s, y, f, y, f, f, s, s,
s, y, s, f, s, y, s, s, s, s, k, y, y, f, s, s, f, ...
$ stalk.color.above.ring <fct> c, e, b, w, w, b, w, y, w, w, w, w, n, w, p, o, g, b, g, w,
w, w, b, w, g, c, n, b, e, g, c, g, o, w, n, y, w, ...
$ stalk.color.below.ring <fct> c, e, b, g, e, b, w, y, w, o, w, w, w, c, w, o, g, w, g, w,
w, g, w, w, g, w, w, w, w, g, w, g, w, w, n, y, g, ...
$ veil.type <fct> p, p, p, p, p, p, p, u, p, p, u, p, p, p, p, p, p, p, p, p,
u, p, p, p, u, p, p, p, p, p, p, p, p, p, p, p, ...
$ veil.color <fct> w, w, w, w, w, w, w, y, w, y, w, o, o, w, o, n, w, w, w,
w, o, w, n, w, w, n, o, y, w, w, w, w, w, o, w, n, ...
$ ring.number <fct> t, o, o, b, b, o, o, t, t, o, o, o, o, o, t, o, o, o, b, b,
o, o, b, o, o, t, o, o, o, o, t, o, o, b, t, b, ...
$ ring.type <fct> p, p, p, n, z, p, p, p, p, p, s, f, f, p, e, l, e, e, s, p,
s, l, p, p, n, f, p, p, p, p, n, p, f, p, l, p, p, ...
$ spore.print.color <fct> k, u, n, k, w, k, w, y, b, k, b, w, n, b, k, n, n, k, h, u,
n, n, u, r, k, n, k, n, k, w, n, n, r, n, n, y, k, ...
$ population <fct> n, a, v, s, a, n, n, y, v, s, n, s, a, v, a, v, s, c, s, s,
c, s, s, v, a, n, s, v, a, v, y, s, v, s, a, n, a, ...
$ habitat <fct> u, p, g, u, g, p, m, w, w, g, g, g, d, g, g, p, g, g, g, w,
m, d, d, d, l, w, g, m, u, l, p, l, u, p, m, u, u, ...
```

Numero de niveles de cada variable:

Hide

```
number_class <- function(x){
  x <- length(levels(x))
}
x <- setas %>% map_dbl(function(.x) number_class(.x)) %>% as_tibble() %>%
  rownames_to_column() %>% arrange(desc(value))
colnames(x) <- c("Variable name", "Number of levels")
print(x)
```

Variable name	Number of levels
<chr>	<dbl>
gill.color	12
cap.color	10
odor	9
stalk.color.above.ring	9
stalk.color.below.ring	9
spore.print.color	9
ring.type	8
stalk.root	7
habitat	7
cap.shape	6
1-10 of 23 rows	
Previous 1 2 3 Next	

Verificación de la calidad de los datos

Se observa que los valores de las variables se definen utilizando una sola letra, lo que los convierte en ilegibles para el observador humano. Por ello se llevará a cabo un procedimiento de formateo de los datos (ver apartado siguiente).

Se comprobó la existencia de missing values, no encontrando ninguno.

Hide

```
map_dbl(setas, function(.x) {sum(is.na(.x))})
```

	class	cap.shape	cap.surface	ca
p.color	bruises			
0	0	0	0	
l.size	odor	gill.attachment	gill.spacing	gil
0	gill.color	0	0	
0	0			
w.ring	stalk.shape	stalk.root	stalk.surface.above.ring	stalk.surface.below.ring
0	stalk.color.above.ring	0	0	
0	0			
stalk.color.below.ring		veil.type	veil.color	ring.
number	ring.type			
0	0	0	0	
0	0			
spore.print.color		population	habitat	
0	0	0	0	

Formateo de los datos*

*Aunque según el CRISP-DM esta tarea correspondería a la fase de preparación de los datos y aunque aún no se ha finalizado el reporte de la fase de entendimiento, hemos decidido incluirla en este apartado para lograr gráficos con nombres de valor de variables significativos.

Comenzamos renombrando los nombres de las variables para adaptarlos al formato Snake Case y dotar algunos de ellos de más significado como el caso de la antigua “class” nueva “edibility”.

[Hide](#)

```
colnames(setas) <- c("edibility", "cap_shape", "cap_surface",
                    "cap_color", "bruises", "odor",
                    "gill_attachement", "gill_spacing", "gill_size",
                    "gill_color", "stalk_shape", "stalk_root",
                    "stalk_surface_above_ring", "stalk_surface_below_ring", "stalk_color_
above_ring",
                    "stalk_color_below_ring", "veil_type", "veil_color",
                    "ring_number", "ring_type", "spore_print_color",
                    "population", "habitat")
```

Renombramos los también valores de cada una de las variables a sus significados extraídos de la explicación del dataset.

[Hide](#)

```

#setas <- setas %>% map_df(function(.x) as.factor(.x))
levels(setas$edibility) <- c("edible", "poisonous")
levels(setas$cap_shape) <- c("bell", "conical", "flat", "knobbed", "sunken", "convex")
levels(setas$cap_color) <- c("buff", "cinnamon", "red", "gray", "brown", "pink",
                             "green", "purple", "white", "yellow")
levels(setas$cap_surface) <- c("fibrous", "grooves", "scaly", "smooth")
levels(setas$bruises) <- c("no", "yes")
levels(setas$odor) <- c("almond", "creosote", "foul", "anise", "musty", "none", "pungent", "s
picy", "fishy")
levels(setas$gill_attachment) <- c("attached", "descending", "free", "notched")
levels(setas$gill_spacing) <- c("close", "crowded", "distant")
levels(setas$gill_size) <- c("broad", "narrow")
levels(setas$gill_color) <- c("buff", "red", "gray", "chocolate", "black", "brown", "orange",
                             "pink", "green", "purple", "white", "yellow")
levels(setas$stalk_shape) <- c("enlarging", "tapering")
levels(setas$stalk_root) <- c("bulbous", "club", "cup", "equal", "rhizomorphs", "rooted", "mis
sing")
levels(setas$stalk_surface_above_ring) <- c("fibrous", "silky", "smooth", "scaly")
levels(setas$stalk_surface_below_ring) <- c("fibrous", "silky", "smooth", "scaly")
levels(setas$stalk_color_above_ring) <- c("buff", "cinnamon", "red", "gray", "brown", "pink",
                             "green", "purple", "white", "yellow")
levels(setas$stalk_color_below_ring) <- c("buff", "cinnamon", "red", "gray", "brown", "pink",
                             "green", "purple", "white", "yellow")
levels(setas$veil_type) <- c("partial", "universal")
levels(setas$veil_color) <- c("brown", "orange", "white", "yellow")
# En el dataset pone B o y T y en la descripción None one and two.
levels(setas$ring_number) <- c("b", "none", "one", "two")
levels(setas$ring_type) <- c("cobwebby", "evanescent", "flaring", "large", "none", "pendant",
"sheathing", "zone")
levels(setas$spore_print_color) <- c("buff", "chocolate", "black", "brown", "orange",
                             "green", "purple", "white", "yellow")
levels(setas$population) <- c("abundant", "clustered", "numerous", "scattered", "several", "s
olitary")
levels(setas$habitat) <- c("wood", "grasses", "leaves", "meadows", "paths", "urban", "waste")
glimpse(setas)

```

Observations: 8,124

Variables: 23

```
$ edibility          <fct> poisonous, edible, edible, poisonous, edible, edible, edibl
e, edible, poisonous, edible, edible, edible, edible...
$ cap_shape          <fct> sunken, convex, knobbed, bell, bell, knobbed, flat, bell, su
nken, bell, convex, conical, bell, convex, sunken, ...
$ cap_surface        <fct> fibrous, grooves, scaly, smooth, scaly, scaly, fibrous, smoo
th, smooth, scaly, scaly, grooves, grooves, grooves...
$ cap_color          <fct> gray, yellow, brown, white, gray, yellow, white, white, yell
ow, yellow, green, yellow, cinnamon, white, red, gr...
$ bruises            <fct> yes, yes, yes, yes, no, no, yes, yes, no, yes, yes, yes, ye
s, yes, yes, yes, yes, yes, no, yes, no, yes, yes, y...
$ odor               <fct> pungent, almond, anise, fishy, none, almond, almond, anise,
pungent, almond, anise, pungent, almond, creosote, ...
$ gill_attachment    <fct> free, descending, free, free, notched, free, attached, free,
free, descending, notched, descending, free, free,...
$ gill_spacing       <fct> crowded, close, distant, close, close, distant, close, clos
e, crowded, close, crowded, distant, close, crowded,...
$ gill_size          <fct> narrow, broad, broad, narrow, broad, narrow, broad, broad, n
arrow, broad, narrow, broad, broad, narrow, broad, ...
$ gill_color         <fct> black, black, brown, red, orange, brown, gray, green, buff,
gray, buff, brown, black, green, brown, black, blac...
$ stalk_shape        <fct> enlarging, enlarging, tapering, tapering, tapering, taperin
g, enlarging, tapering, tapering, enlarging, enlargi...
$ stalk_root         <fct> equal, cup, cup, equal, club, missing, cup, cup, rooted, cu
p, rooted, cup, club, rhizomorphs, rhizomorphs, miss...
$ stalk_surface_above_ring <fct> smooth, scaly, smooth, smooth, smooth, smooth, smoot
h, smooth, scaly, smooth, scaly, scaly, scaly, fibr...
$ stalk_surface_below_ring <fct> smooth, scaly, silky, silky, smooth, smooth, fibrous, smoot
h, smooth, scaly, smooth, scaly, smooth, scaly, fibr...
$ stalk_color_above_ring <fct> cinnamon, red, buff, purple, purple, buff, purple, white, pu
rple, purple, purple, purple, brown, purple, green,...
$ stalk_color_below_ring <fct> cinnamon, red, buff, gray, red, buff, purple, white, purple,
pink, purple, purple, purple, cinnamon, purple, pi...
$ veil_type          <fct> partial, partial, partial, partial, partial, partial, partia
l, universal, partial, partial, universal, partial,...
$ veil_color         <fct> white, white, white, white, white, white, white, white, yell
ow, white, yellow, white, orange, orange, white, or...
$ ring_number        <fct> two, one, one, b, b, one, one, two, two, one, one, one, one,
one, two, one, one, one, b, b, one, one, b, one, o...
$ ring_type          <fct> pendant, pendant, pendant, none, zone, pendant, pendant, pen
dant, pendant, pendant, sheathing, flaring, flaring...
$ spore_print_color  <fct> black, purple, brown, black, white, black, white, yellow, bu
ff, black, buff, white, brown, buff, black, brown, ...
$ population         <fct> numerous, abundant, several, scattered, abundant, numerous,
numerous, solitary, several, scattered, numerous, s...
$ habitat            <fct> urban, paths, grasses, urban, grasses, paths, meadows, wast
e, waste, grasses, grasses, grasses, wood, grasses, ...
```

Exploración de los datos

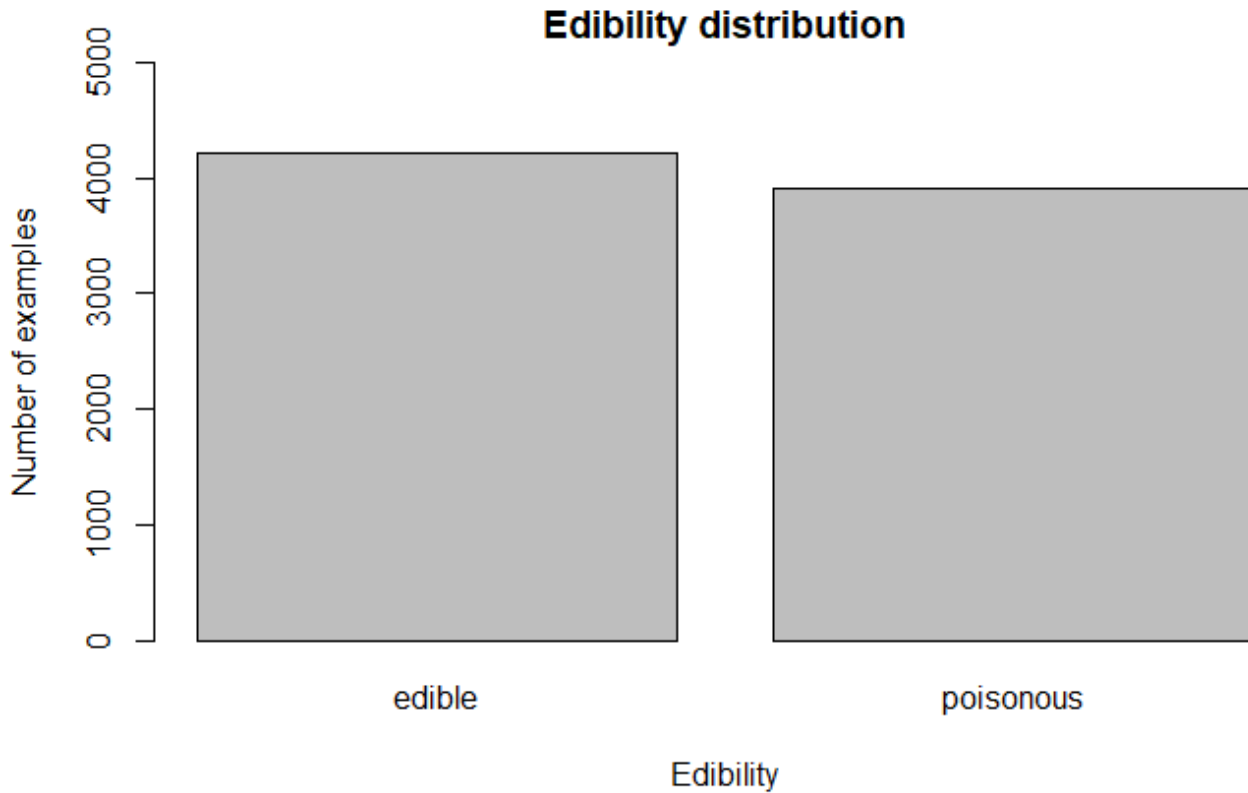
Analizamos la distribución de cada una de las variables que conforman el dataset con una serie de gráficos de barras. Los hechos relevantes que se extraen son los siguientes:

- Existe un número ligeramente mayor de setas comestibles que venenosas en el conjunto de datos.
- En la forma del gorro destacan 'flat' y 'convex'.

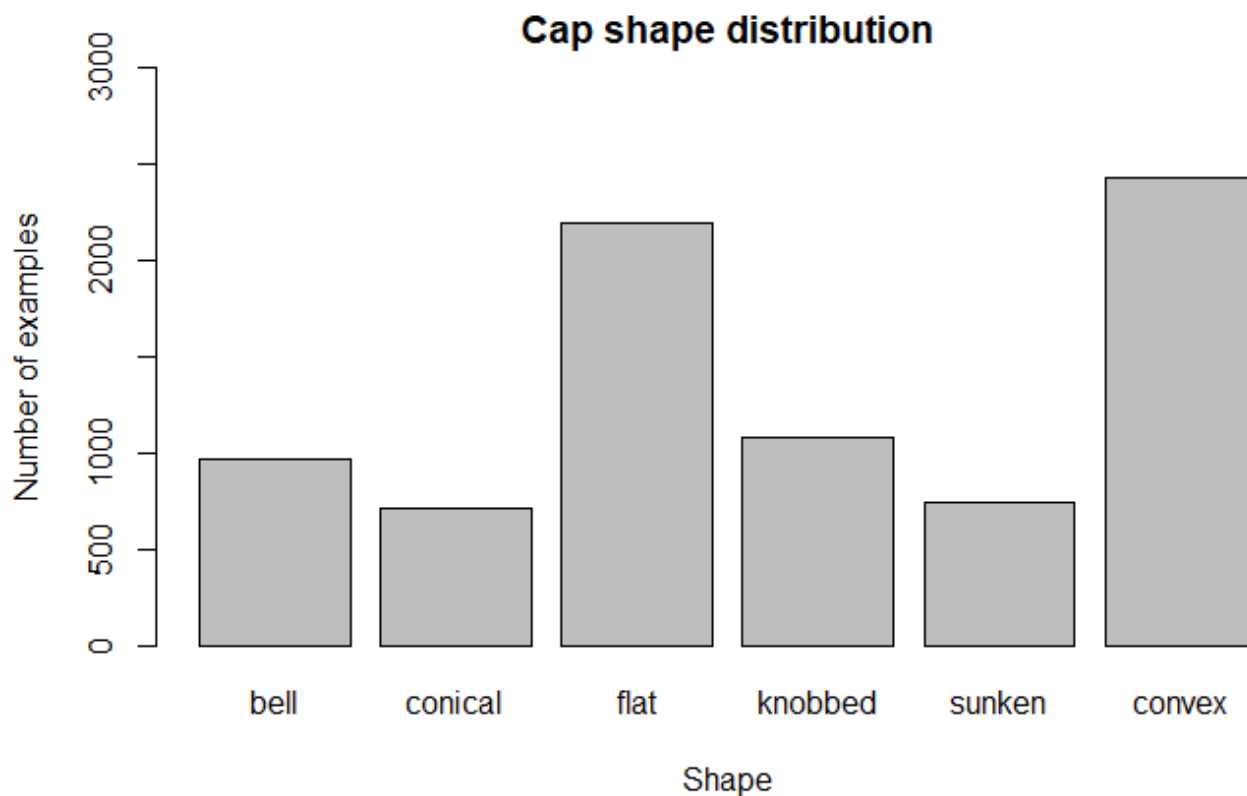
- En el olor destacan 'foul' y 'none'.
- Una gran mayoría tiene el valor 'free' para 'Gill attachment' y 'Close para Gill spacing'.
- Destaca la raíz con forma 'club' y el color morado tanto por encima como por debajo del anillo.
- Una gran mayoría tiene velo parcial y blanco.

[Hide](#)

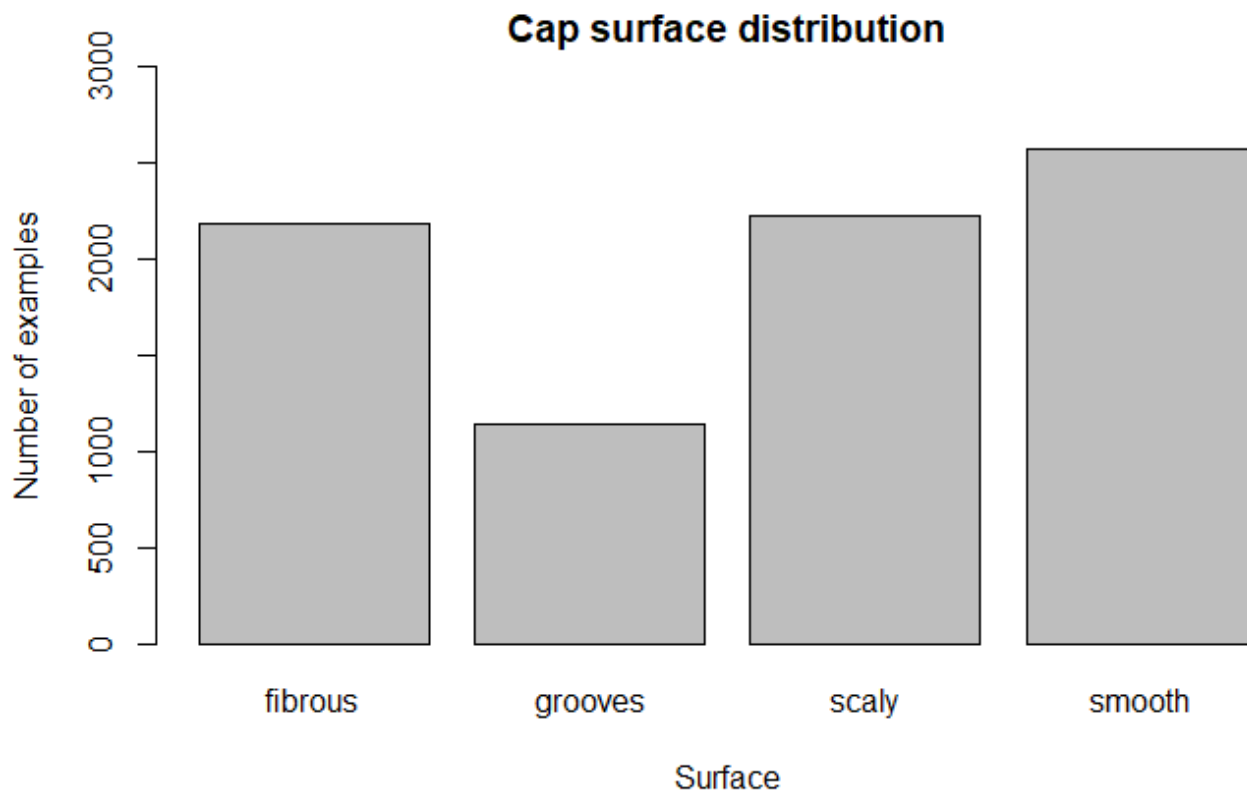
```
barplot(table(setas$edibility), xlab = "Edibility", ylab = "Number of examples",  
        main = "Edibility distribution",ylim=c(0,5000))
```

[Hide](#)

```
barplot(table(setas$cap_shape), xlab = "Shape", ylab = "Number of examples",  
        main = "Cap shape distribution",ylim=c(0,3000))
```

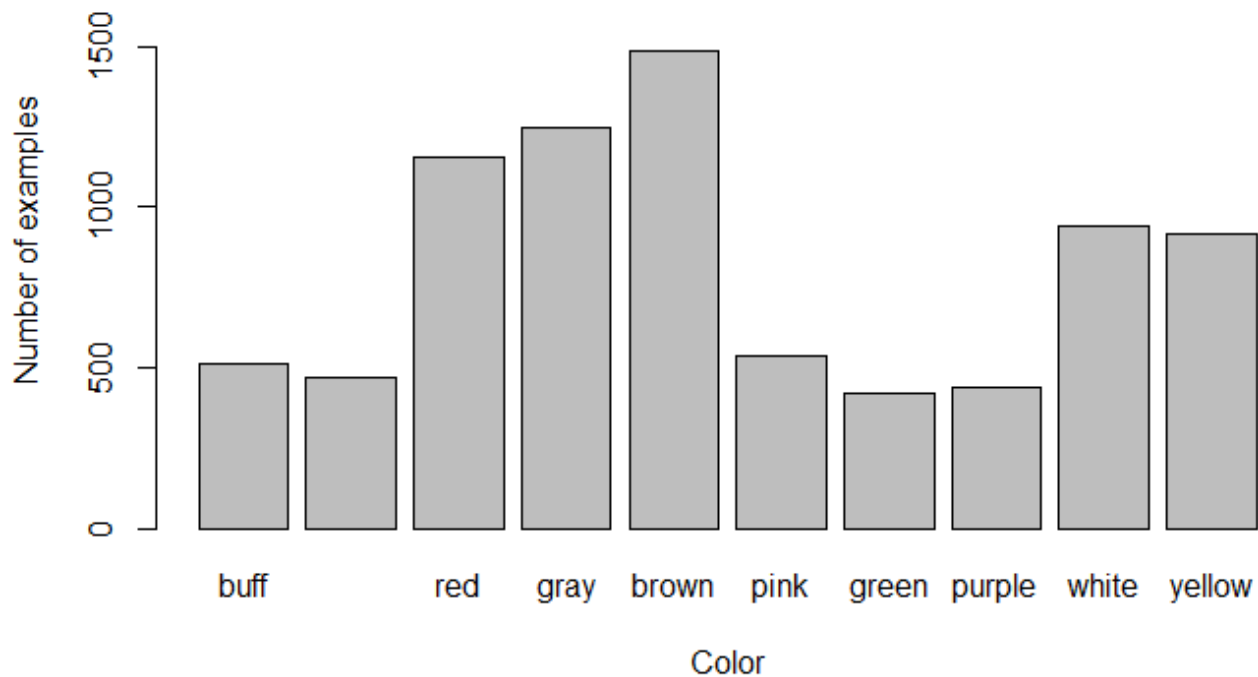
[Hide](#)

```
barplot(table(setas$cap_surface), xlab = "Surface", ylab = "Number of examples",  
        main = "Cap surface distribution",ylim=c(0,3000))
```

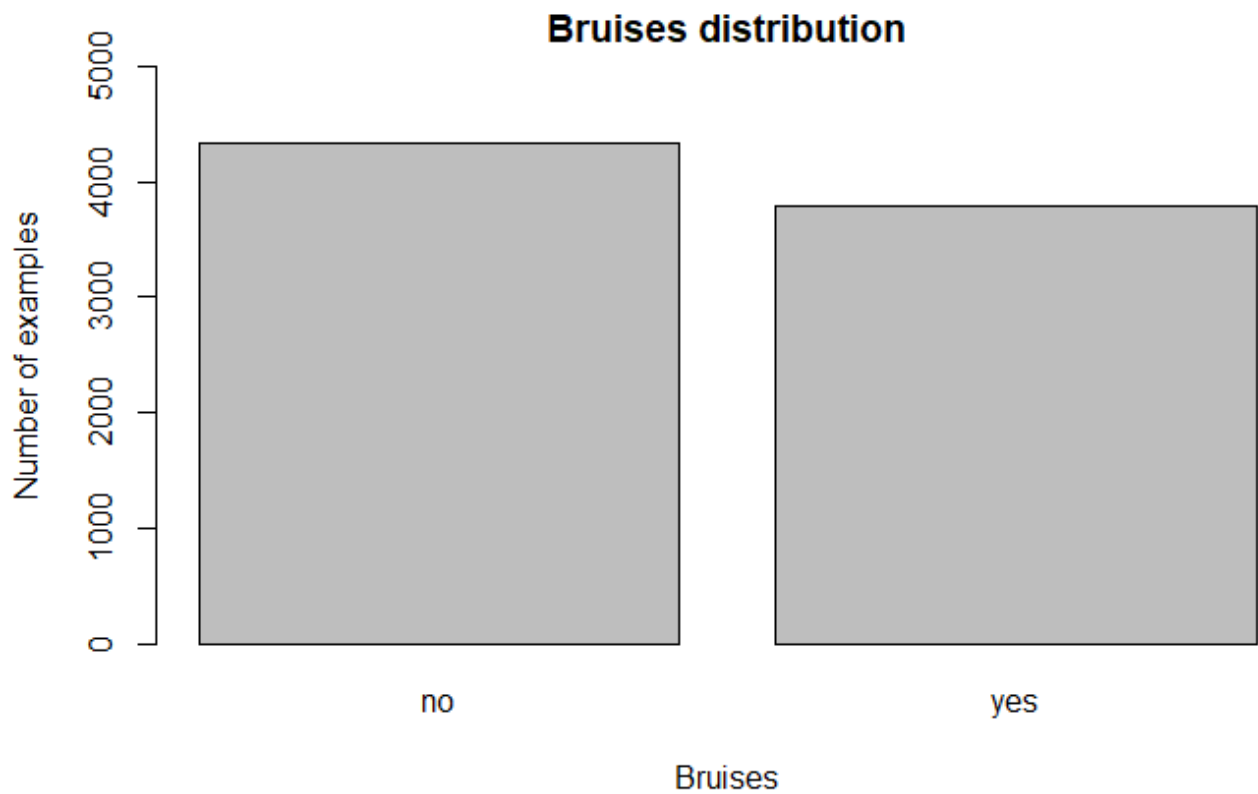
[Hide](#)

```
barplot(table(setas$cap_color), xlab = "Color", ylab = "Number of examples",  
        main = "Cap color distribution",ylim=c(0,1800))
```

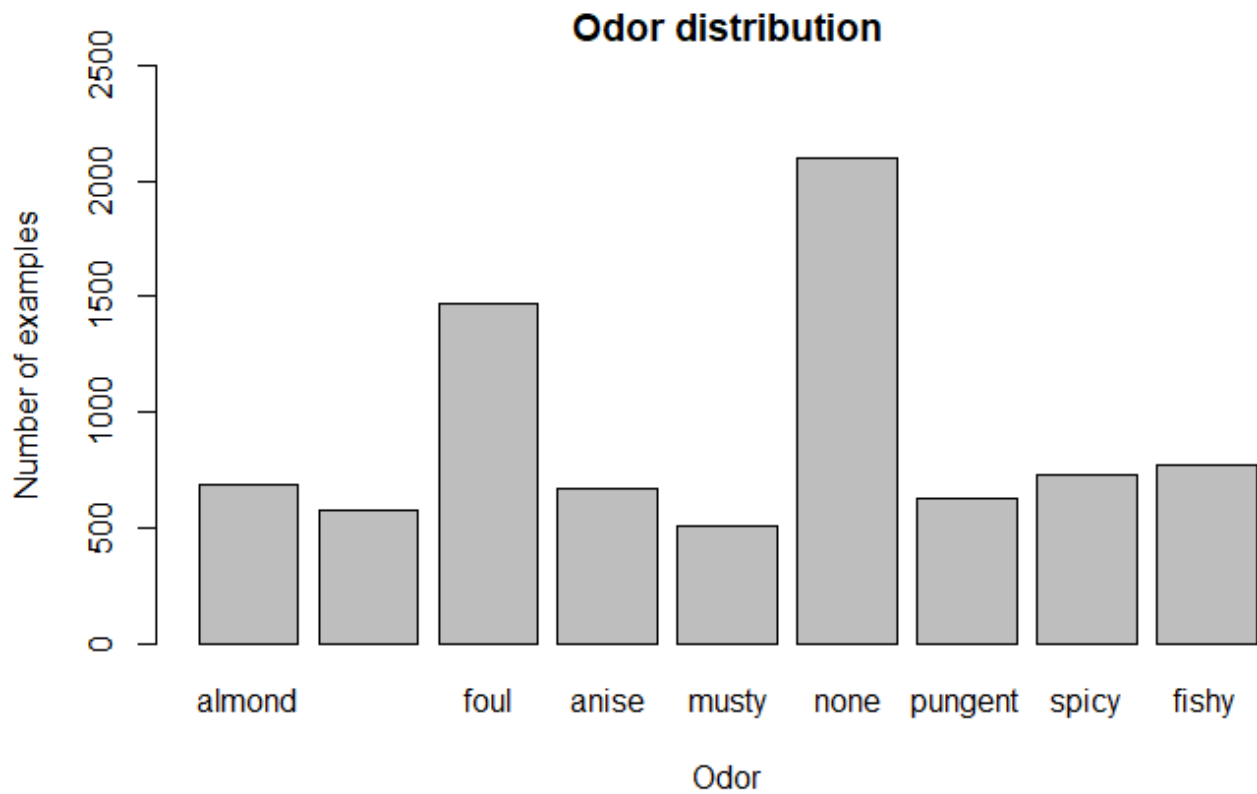
Cap color distribution

[Hide](#)

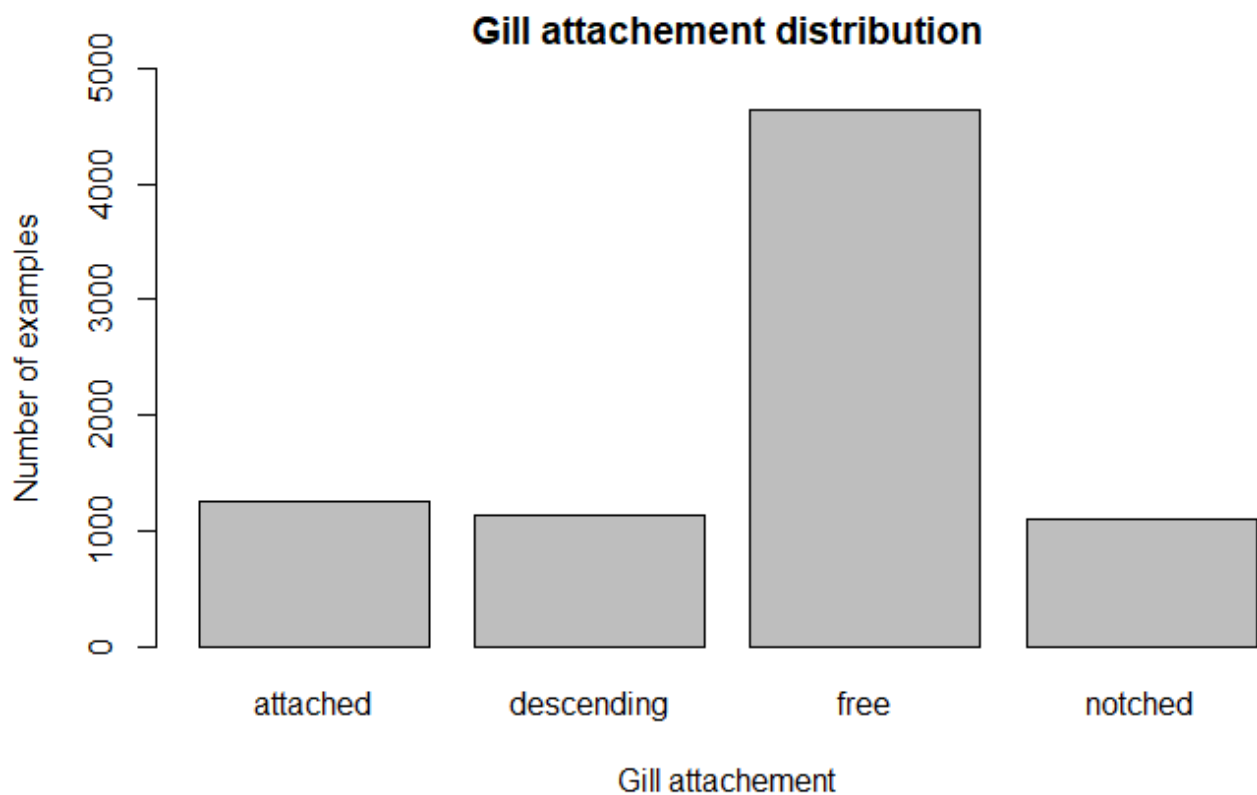
```
barplot(table(setas$bruises), xlab = "Bruises", ylab = "Number of examples",  
        main = "Bruises distribution",ylim=c(0,5000))
```

[Hide](#)

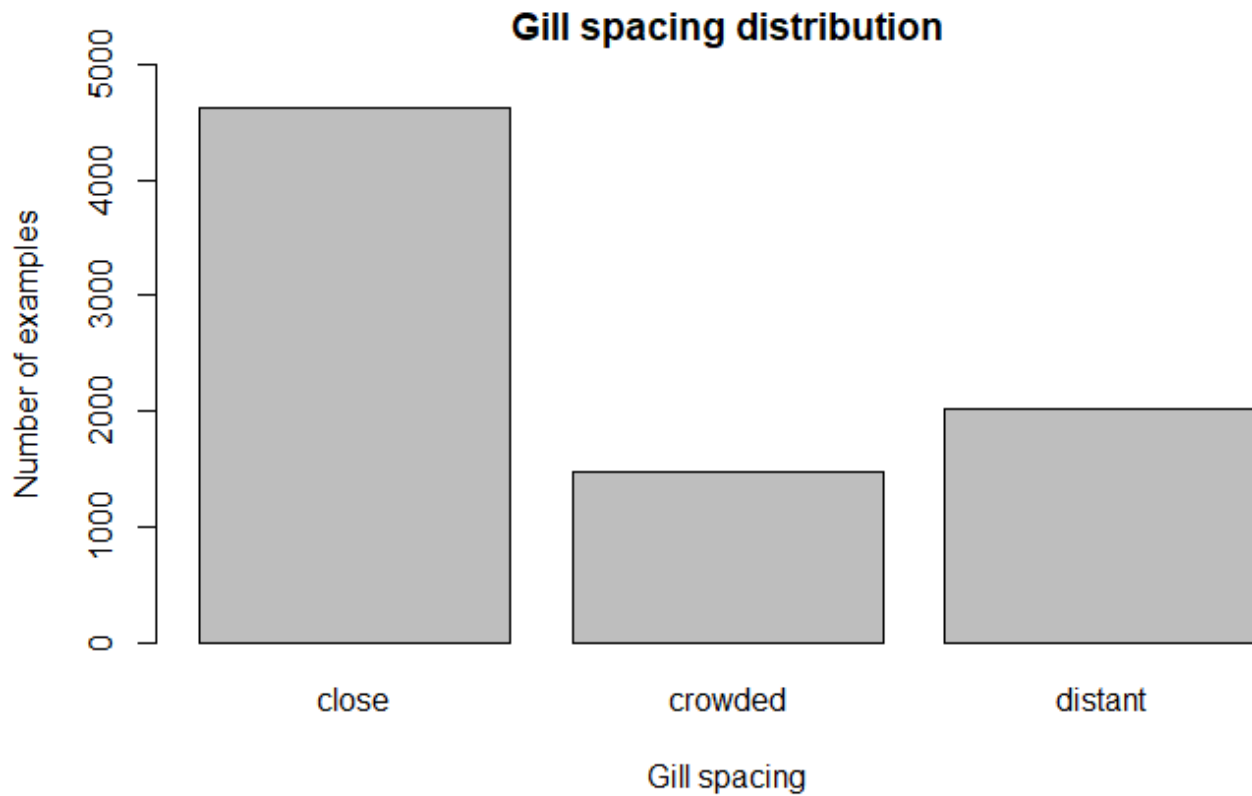
```
barplot(table(setas$odor), xlab = "Odor", ylab = "Number of examples",  
        main = "Odor distribution",ylim=c(0,2500))
```

[Hide](#)

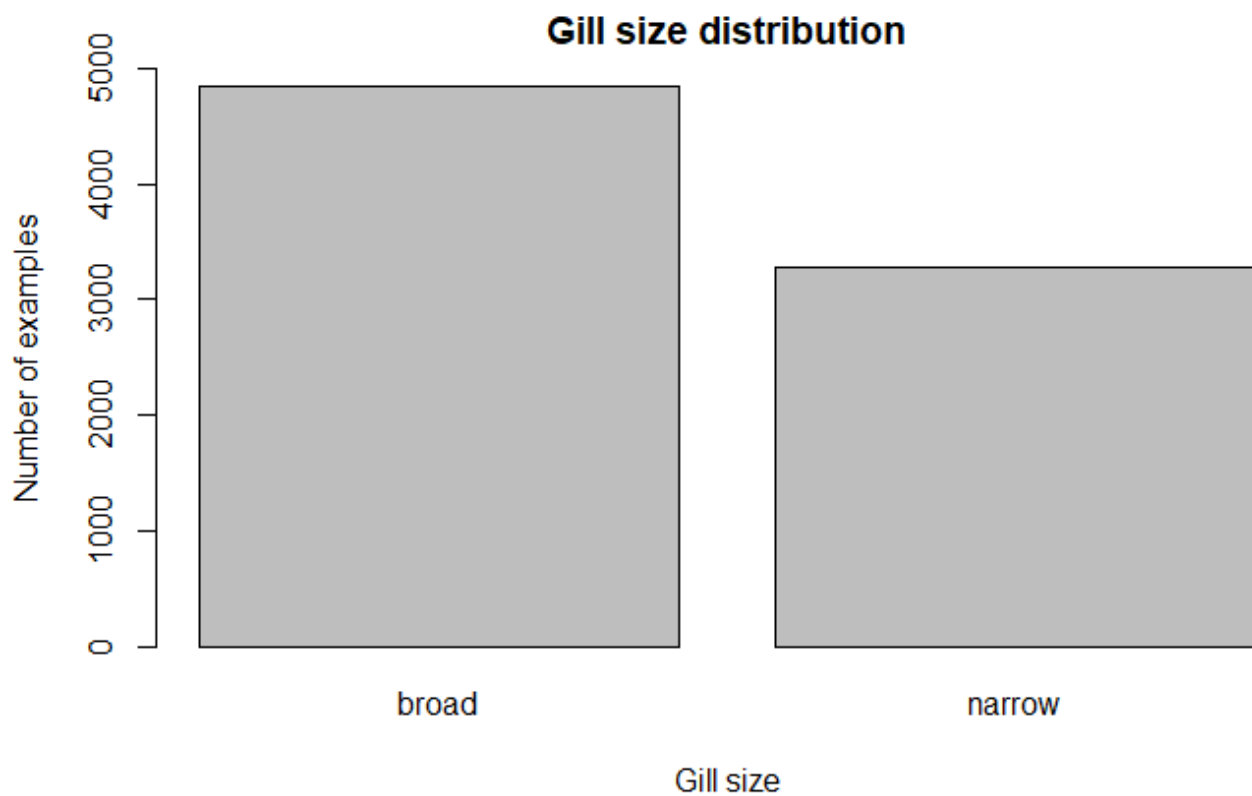
```
barplot(table(setas$gill_attachement), xlab = "Gill attachment", ylab = "Number of examples",  
,  
      main = "Gill attachment distribution",ylim=c(0,5000))
```

[Hide](#)

```
barplot(table(setas$gill_spacing), xlab = "Gill spacing", ylab = "Number of examples",  
        main = "Gill spacing distribution",ylim=c(0,5000))
```

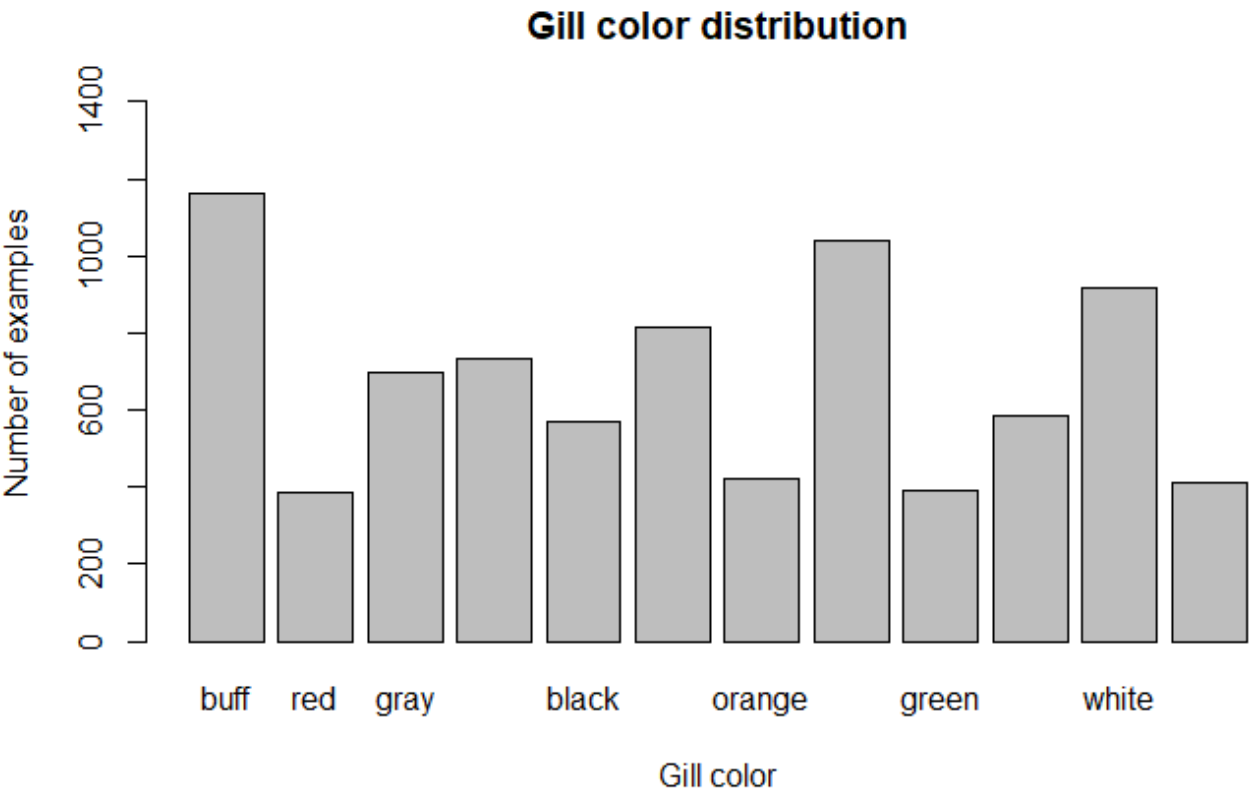
[Hide](#)

```
barplot(table(setas$gill_size), xlab = "Gill size", ylab = "Number of examples",  
        main = "Gill size distribution",ylim=c(0,5000))
```



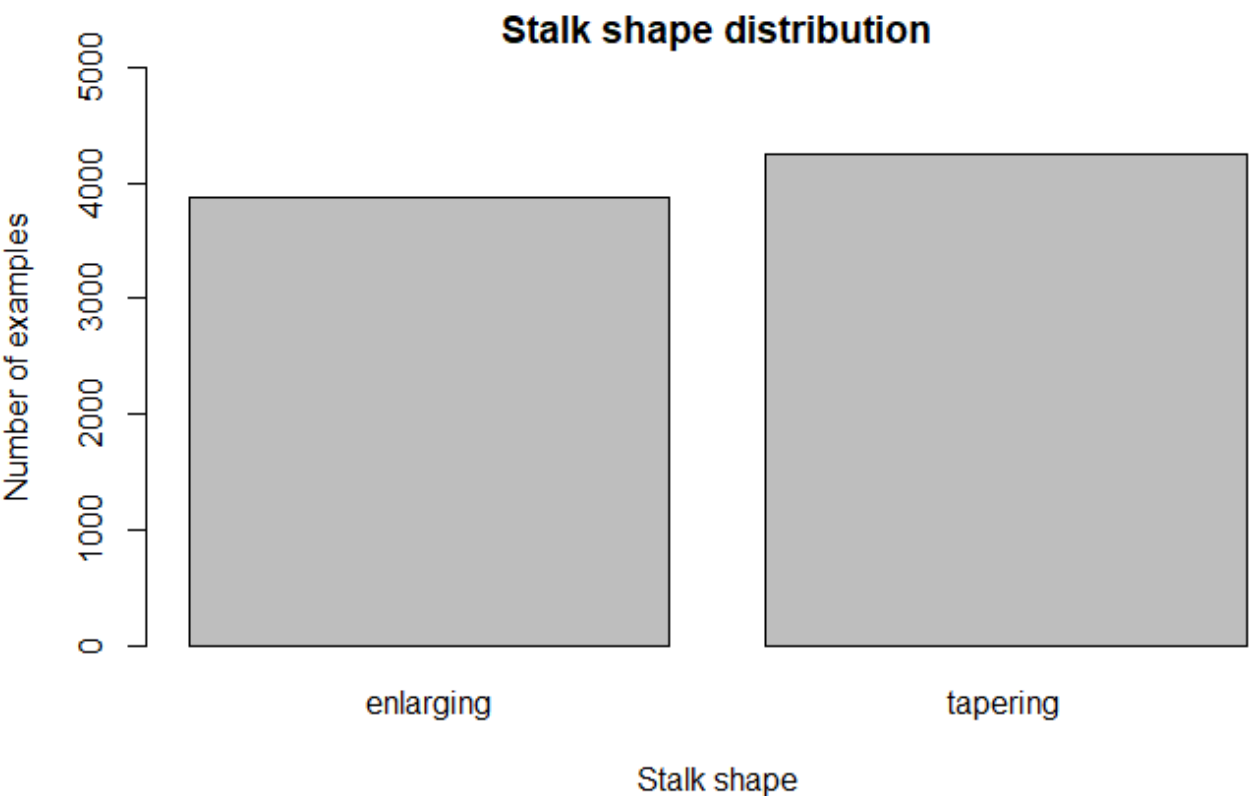
Hide

```
barplot(table(setas$gill_color), xlab = "Gill color", ylab = "Number of examples",
        main = "Gill color distribution",ylim=c(0,1500))
```



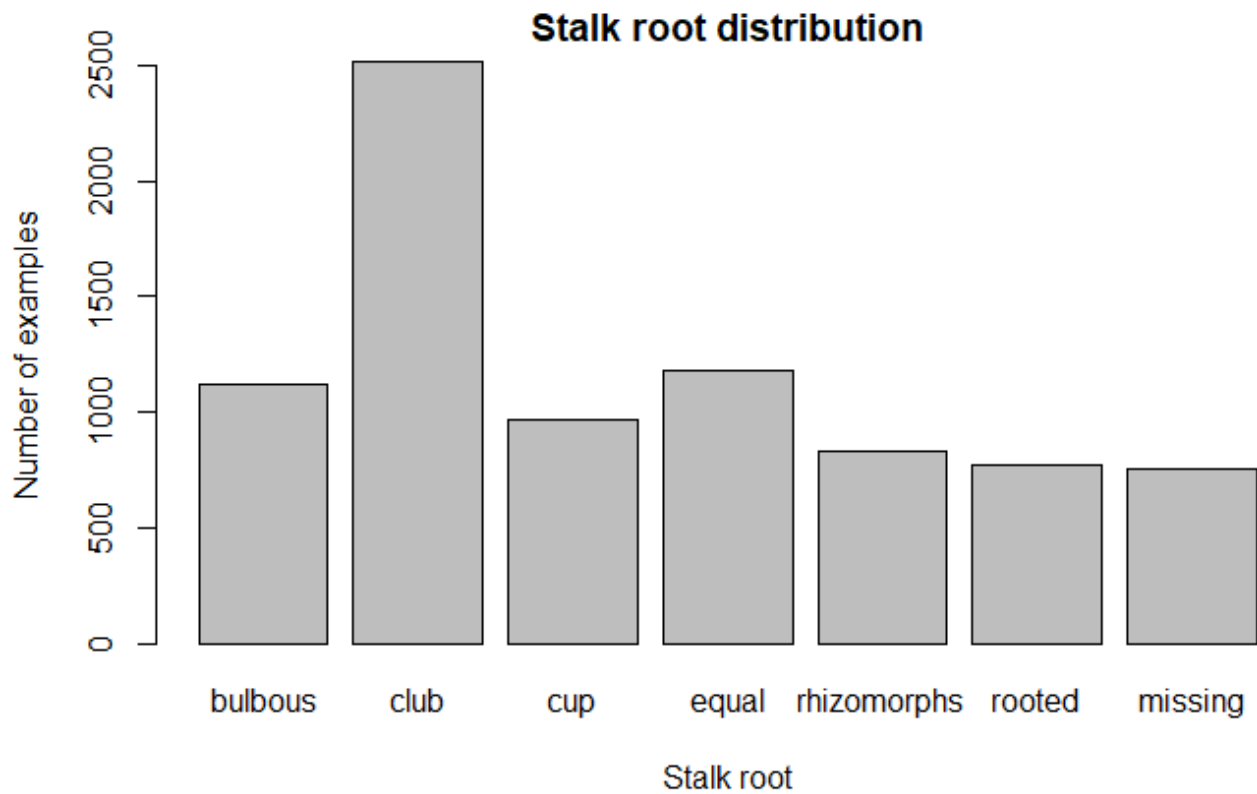
Hide

```
barplot(table(setas$stalk_shape), xlab = "Stalk shape", ylab = "Number of examples",
        main = "Stalk shape distribution",ylim=c(0,5000))
```

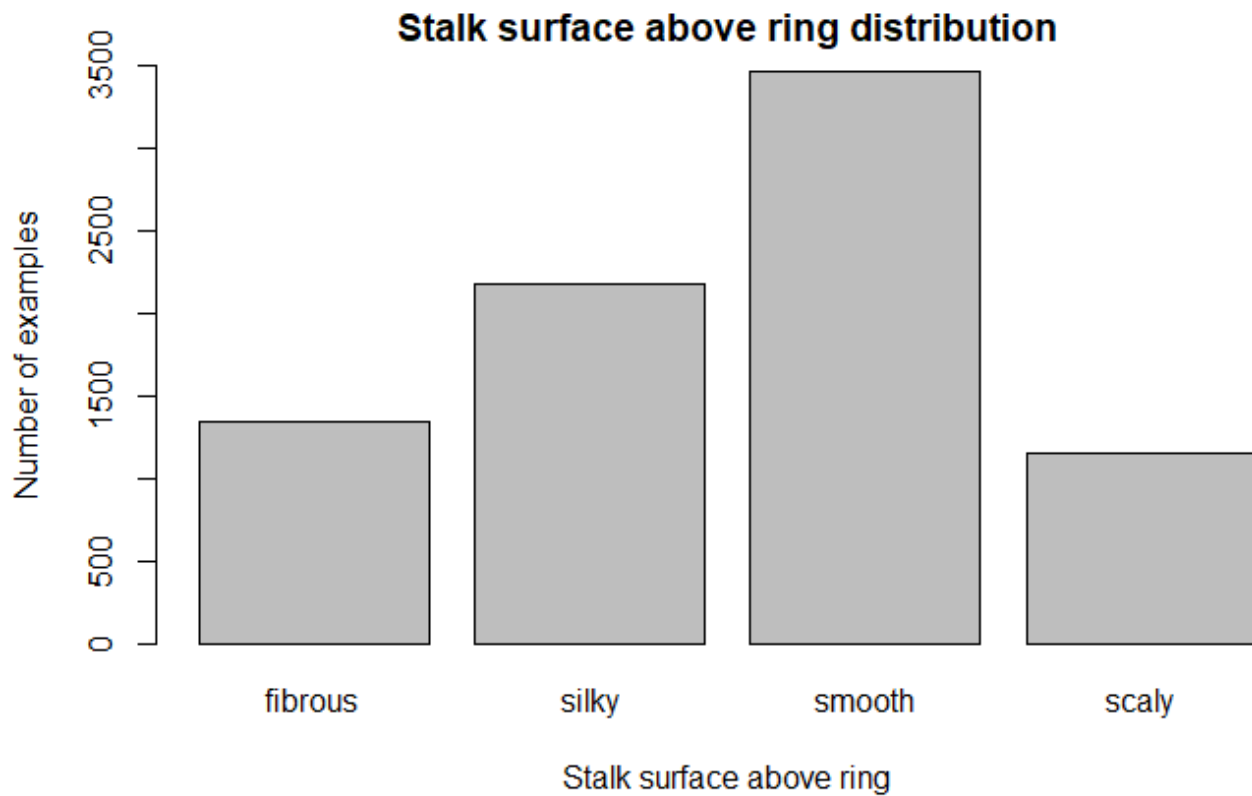


[Hide](#)

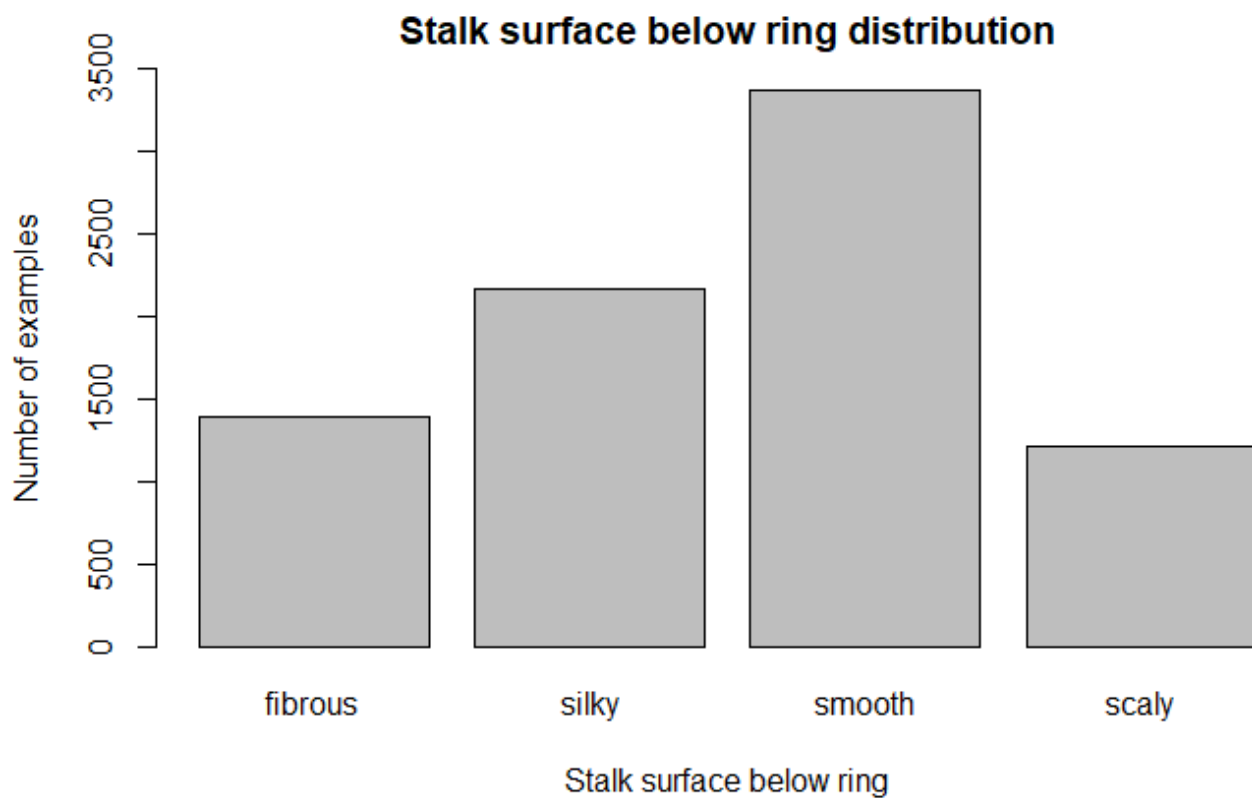
```
barplot(table(setas$stalk_root), xlab = "Stalk root", ylab = "Number of examples",  
        main = "Stalk root distribution",ylim=c(0,2500))
```

[Hide](#)

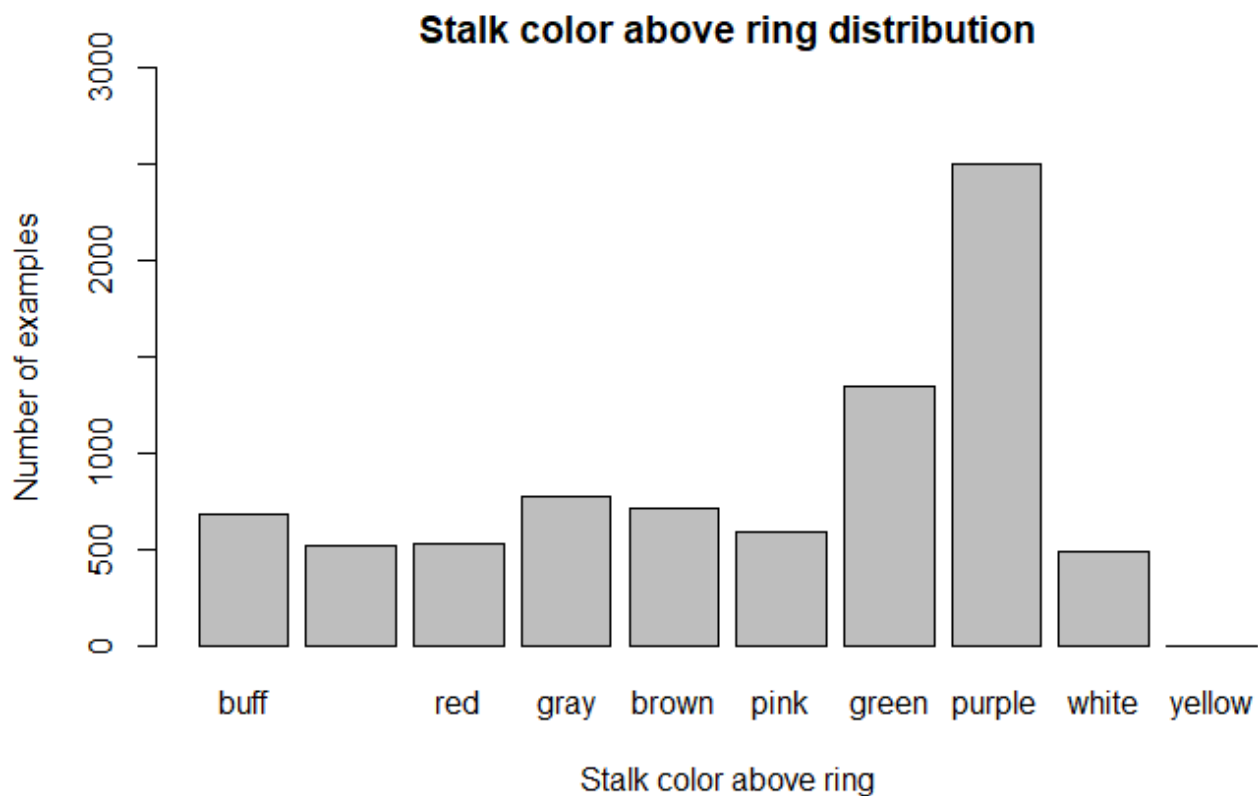
```
barplot(table(setas$stalk_surface_above_ring), xlab = "Stalk surface above ring", ylab = "Num  
ber of examples",  
        main = "Stalk surface above ring distribution",ylim=c(0,3500))
```

[Hide](#)

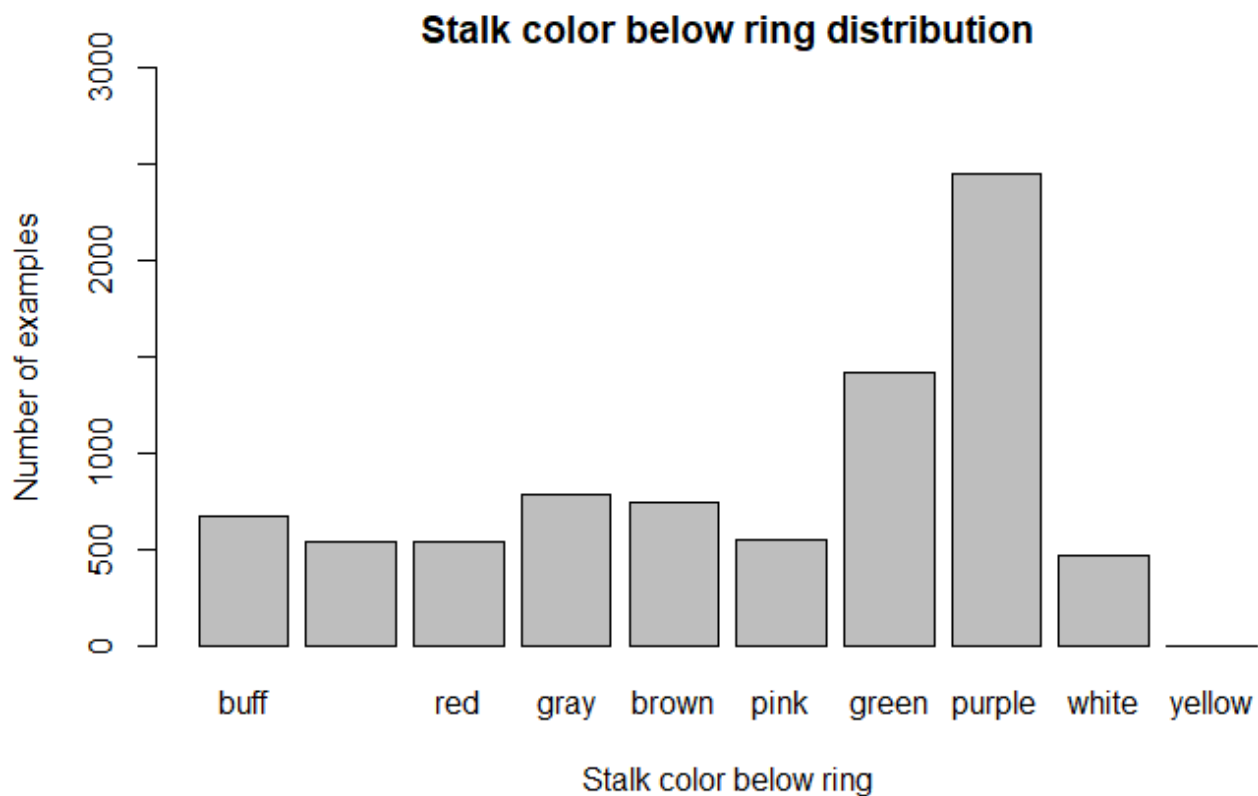
```
barplot(table(setas$stalk_surface_below_ring), xlab = "Stalk surface below ring", ylab = "Number of examples",  
        main = "Stalk surface below ring distribution",ylim=c(0,3500))
```

[Hide](#)

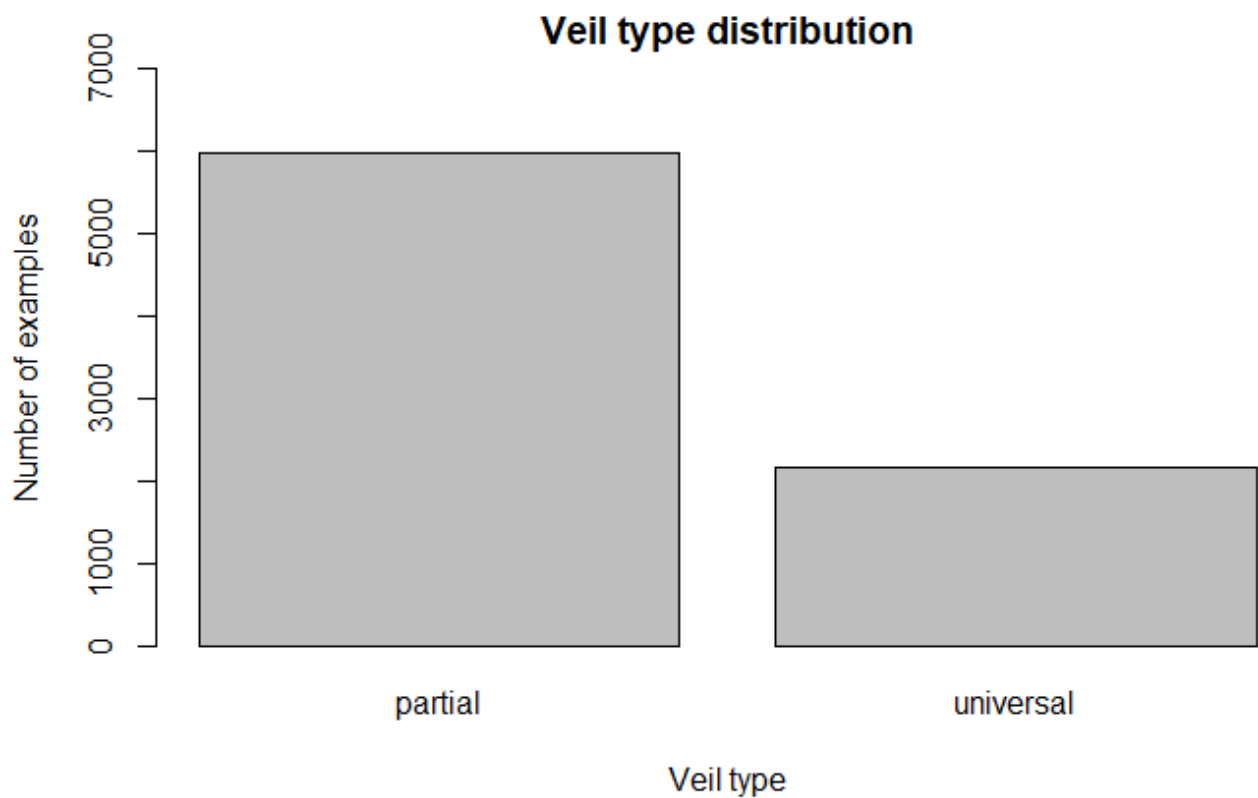

```
barplot(table(setas$stalk_color_above_ring), xlab = "Stalk color above ring", ylab = "Number  
of examples",  
        main = "Stalk color above ring distribution",ylim=c(0,3000))
```

[Hide](#)

```
barplot(table(setas$stalk_color_below_ring), xlab = "Stalk color below ring", ylab = "Number  
of examples",  
        main = "Stalk color below ring distribution",ylim=c(0,3000))
```

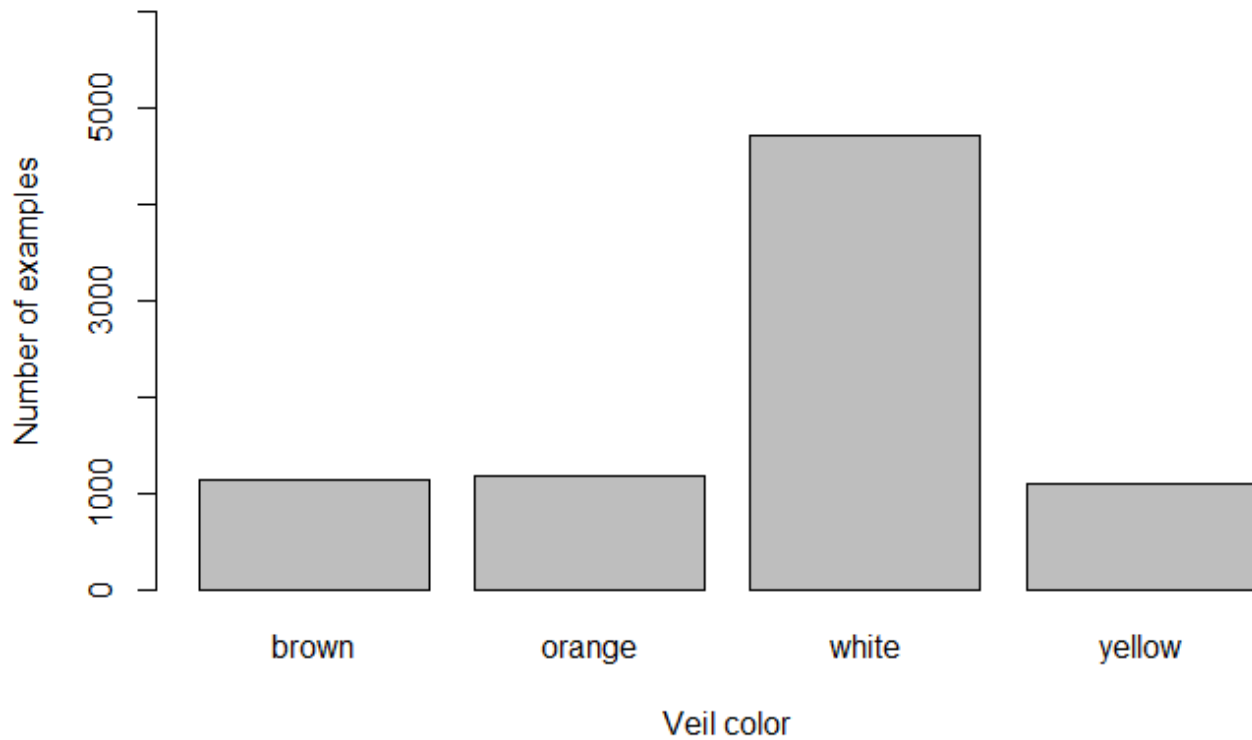
[Hide](#)

```
barplot(table(setas$veil_type), xlab = "Veil type", ylab = "Number of examples",  
        main = "Veil type distribution",ylim=c(0,7000))
```

[Hide](#)

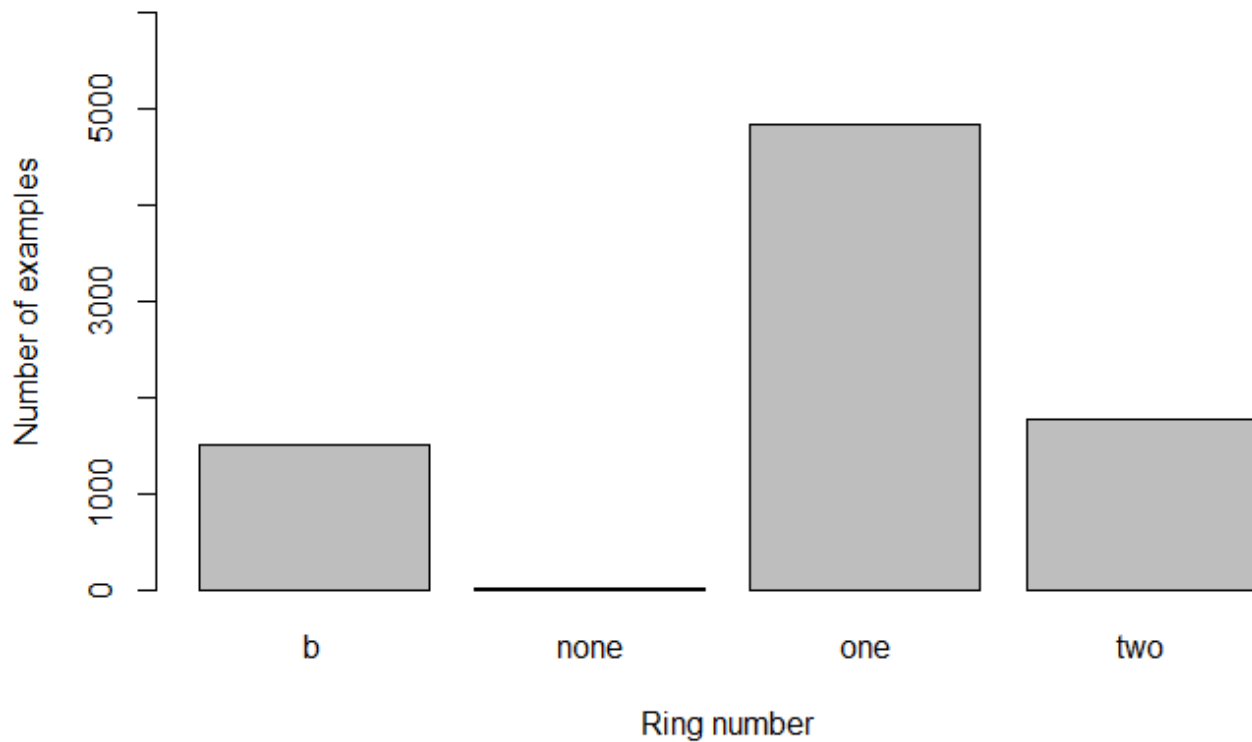
```
barplot(table(setas$veil_color), xlab = "Veil color", ylab = "Number of examples",  
        main = "Veil color distribution",ylim=c(0,6000))
```

Veil color distribution

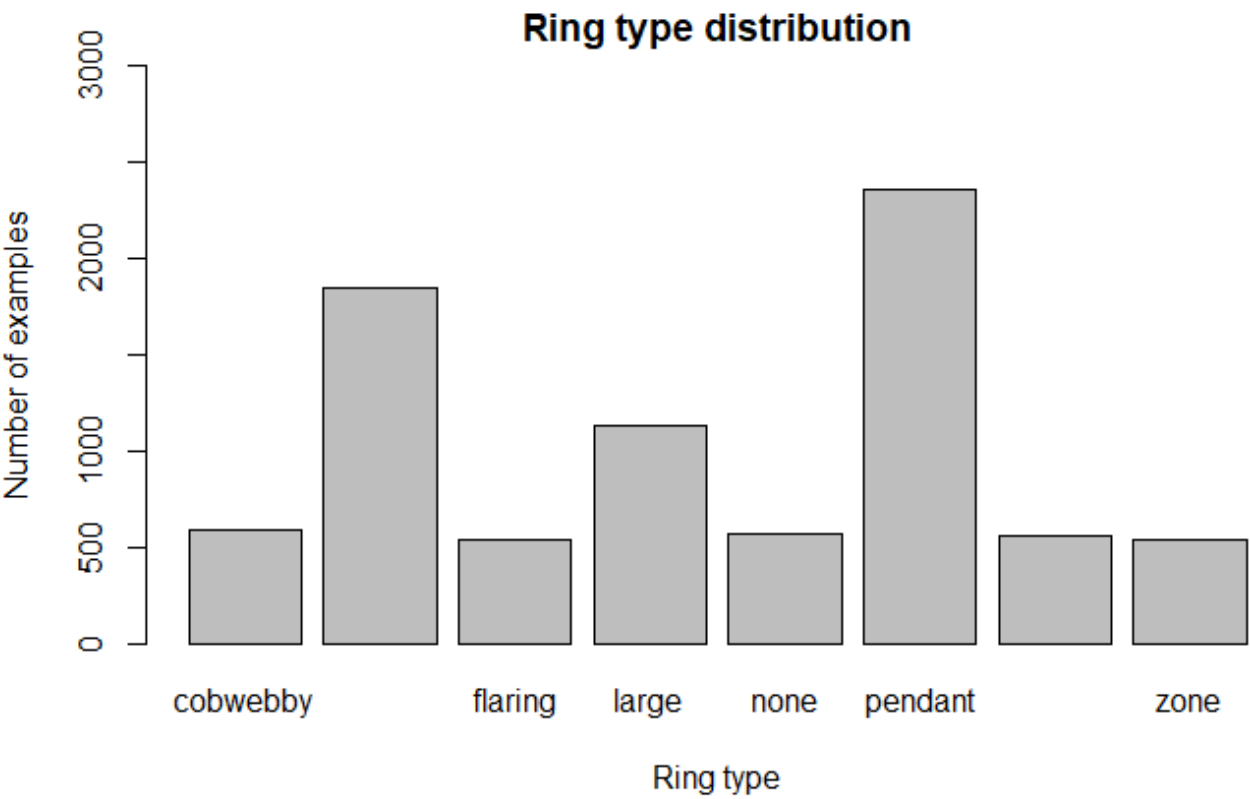
[Hide](#)

```
barplot(table(setas$ring_number), xlab = "Ring number", ylab = "Number of examples",  
        main = "Ring number distribution",ylim=c(0,6000))
```

Ring number distribution

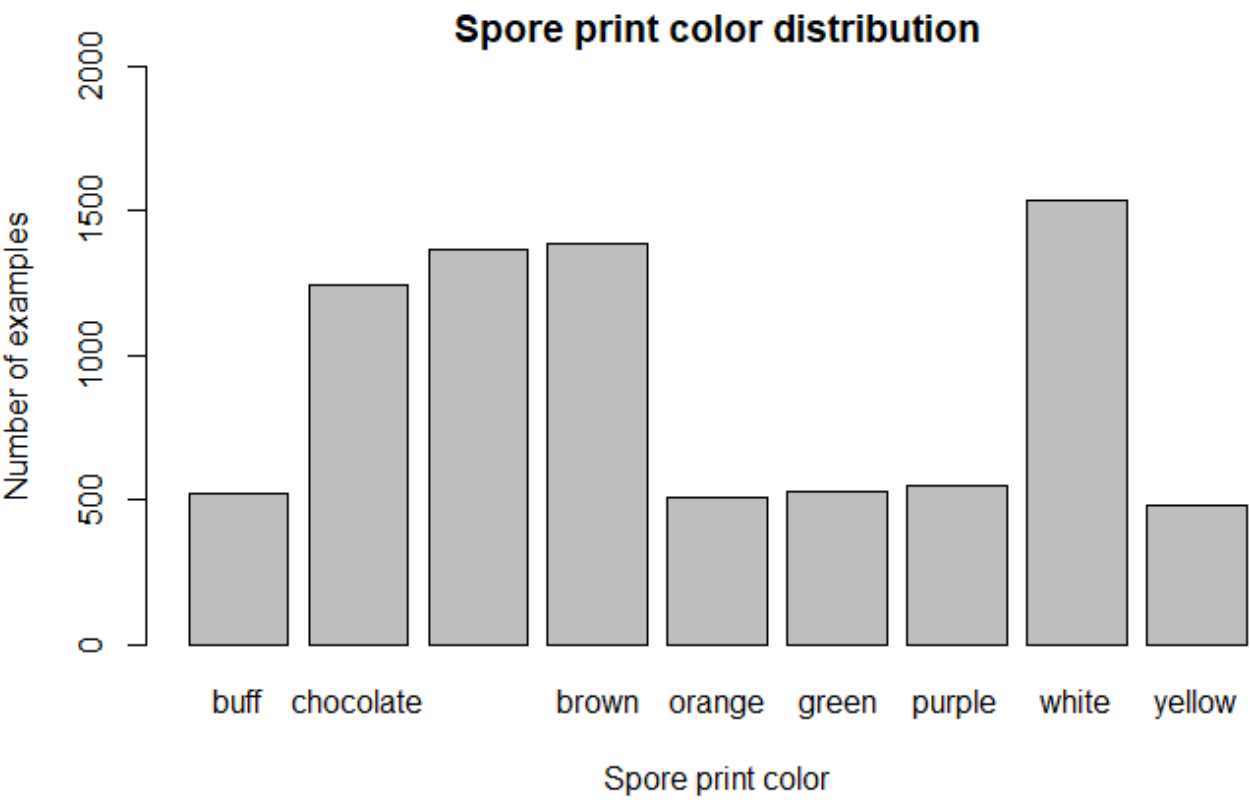
[Hide](#)

```
barplot(table(setas$ring_type), xlab = "Ring type", ylab = "Number of examples",  
        main = "Ring type distribution",ylim=c(0,3000))
```



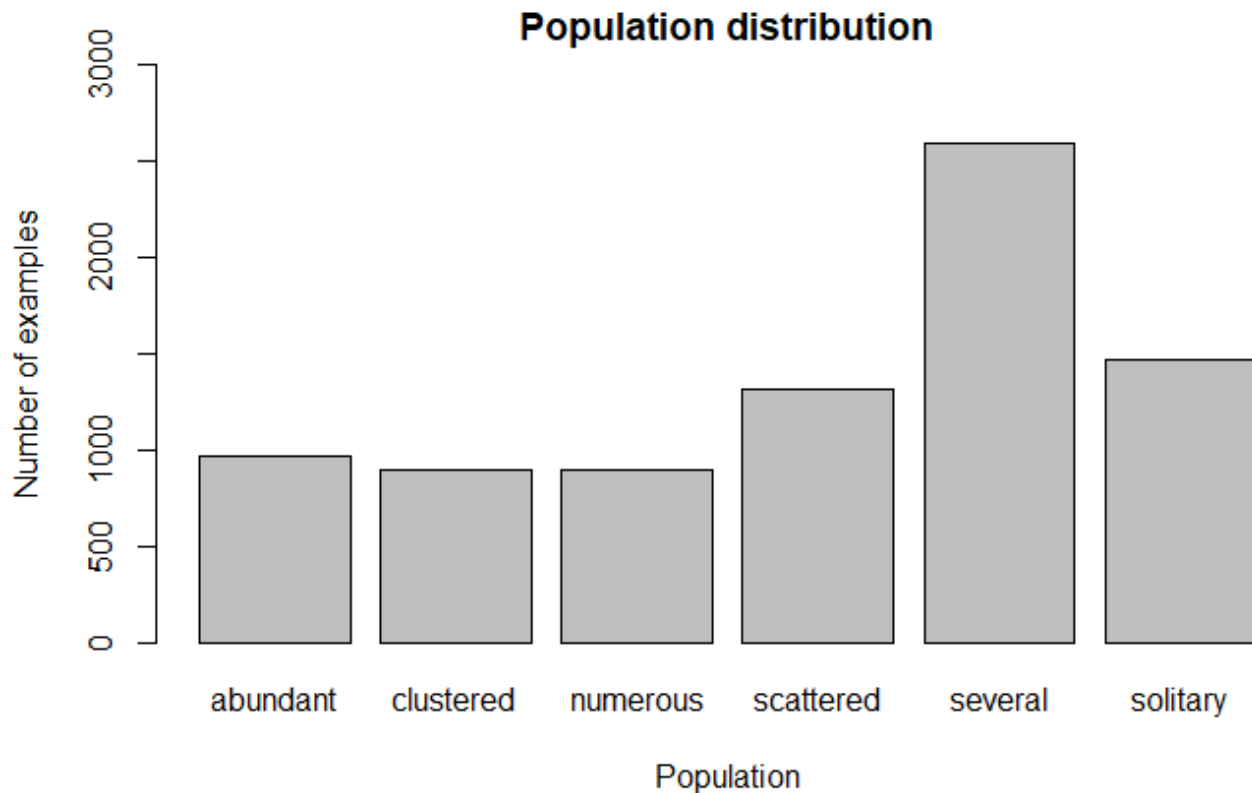
Hide

```
barplot(table(setas$spore_print_color), xlab = "Spore print color", ylab = "Number of examples",  
        main = "Spore print color distribution",ylim=c(0,2000))
```

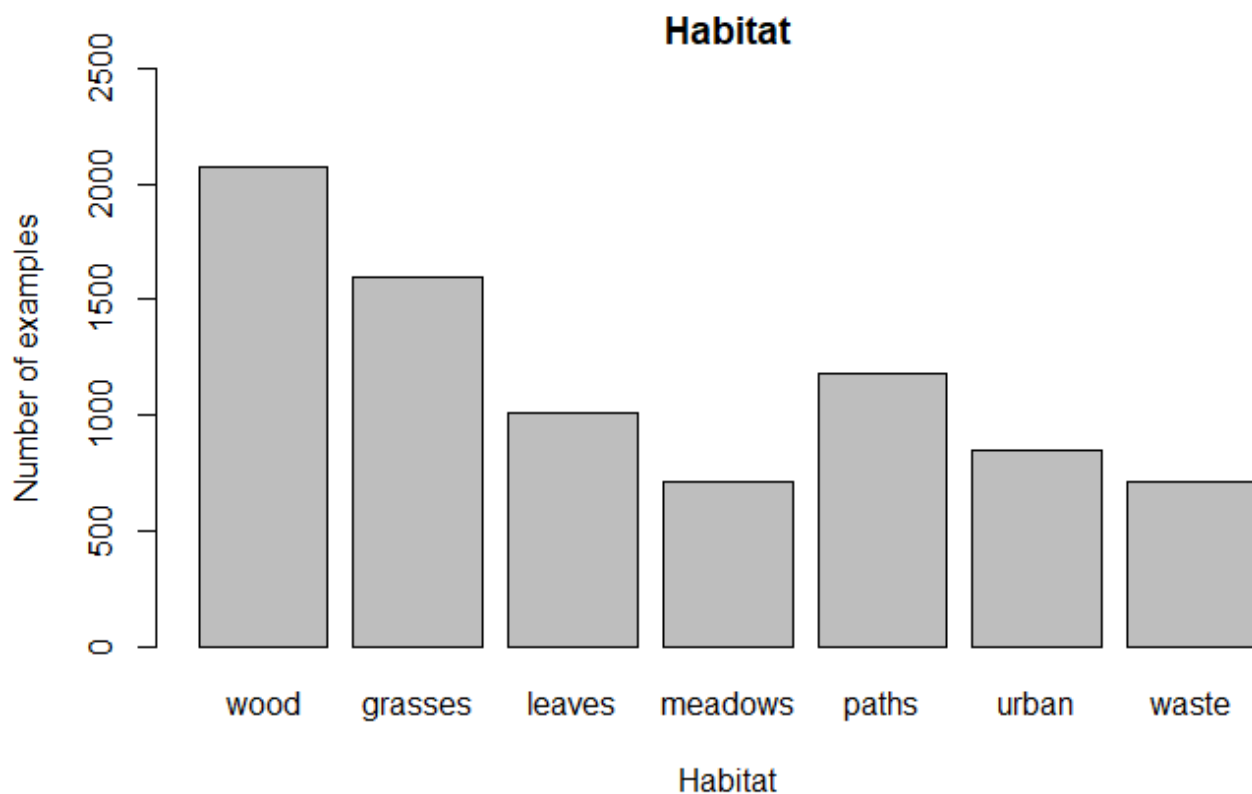


Hide

```
barplot(table(setas$population), xlab = "Population", ylab = "Number of examples",  
        main = "Population distribution",ylim=c(0,3000))
```

[Hide](#)

```
barplot(table(setas$habitat), xlab = "Habitat", ylab = "Number of examples",  
        main = "Habitat",ylim=c(0,2500))
```

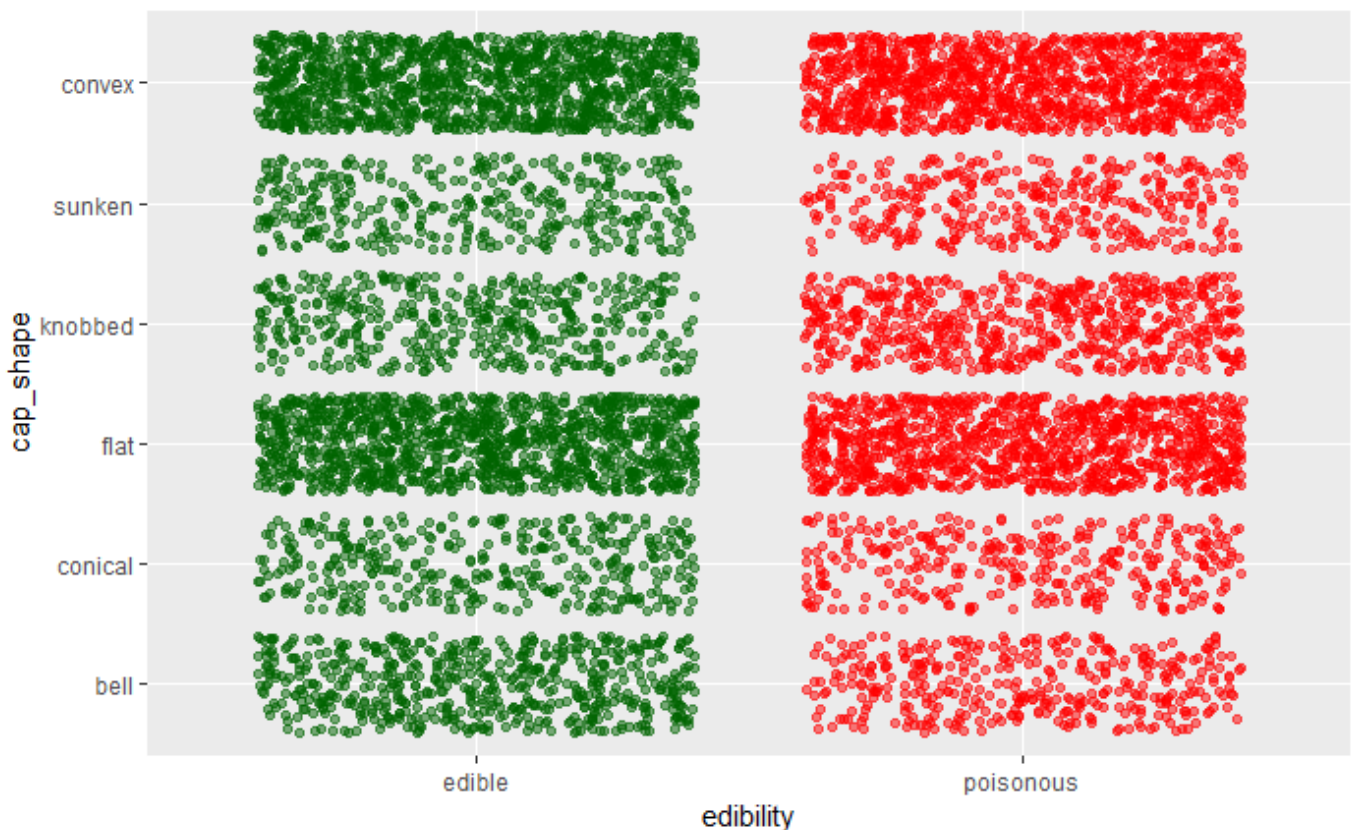


A continuación evaluamos la distribución de cada variable con respecto a la variable respuesta para ver si existen indicios de que alguna variable parece determinante a la hora de decidir si una seta es venenosa o no. A la vista de los resultados, las principales conclusiones extraídas han sido las siguientes:

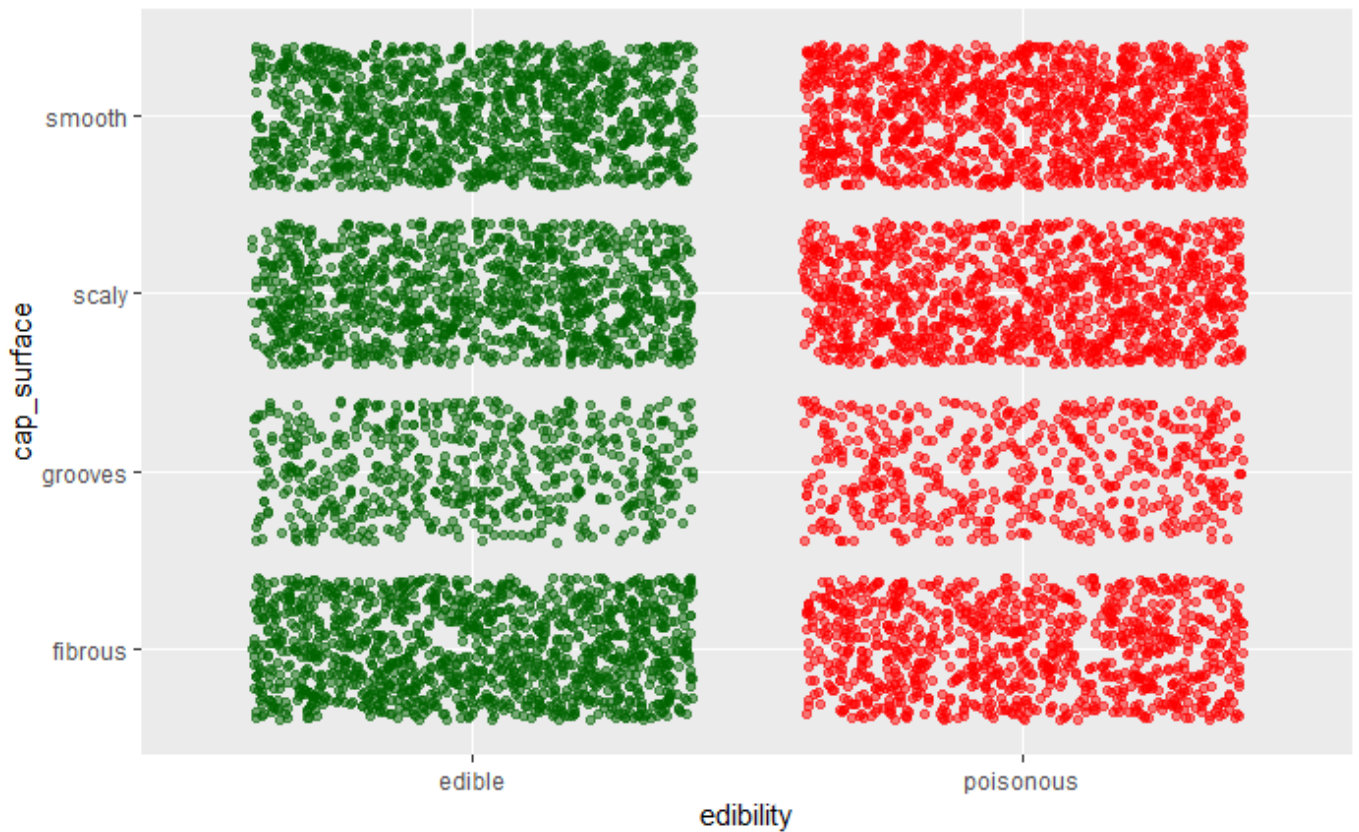
- Las setas con olor 'none' parecen mucho más tendentes a ser comestibles y las que tienen olor 'foul' a ser venenosas.
- Las setas con el anillo grande parecen tender a ser más venenosas.
- Las setas de color blanco y marron tienden a ser más comestibles mientras que las de color 'buff' tienden a ser venenosas.
- Las setas con las esporas color marrón y negro tienden a ser más comestibles mientras que las que las tienen color chocolate tienden a ser venenosas.
- Las setas que crecen en camino parece que tienden ligeramente a ser venenosas.
- Las setas que crecen en grandes poblaciones tienden a ser venenosas.

[Hide](#)

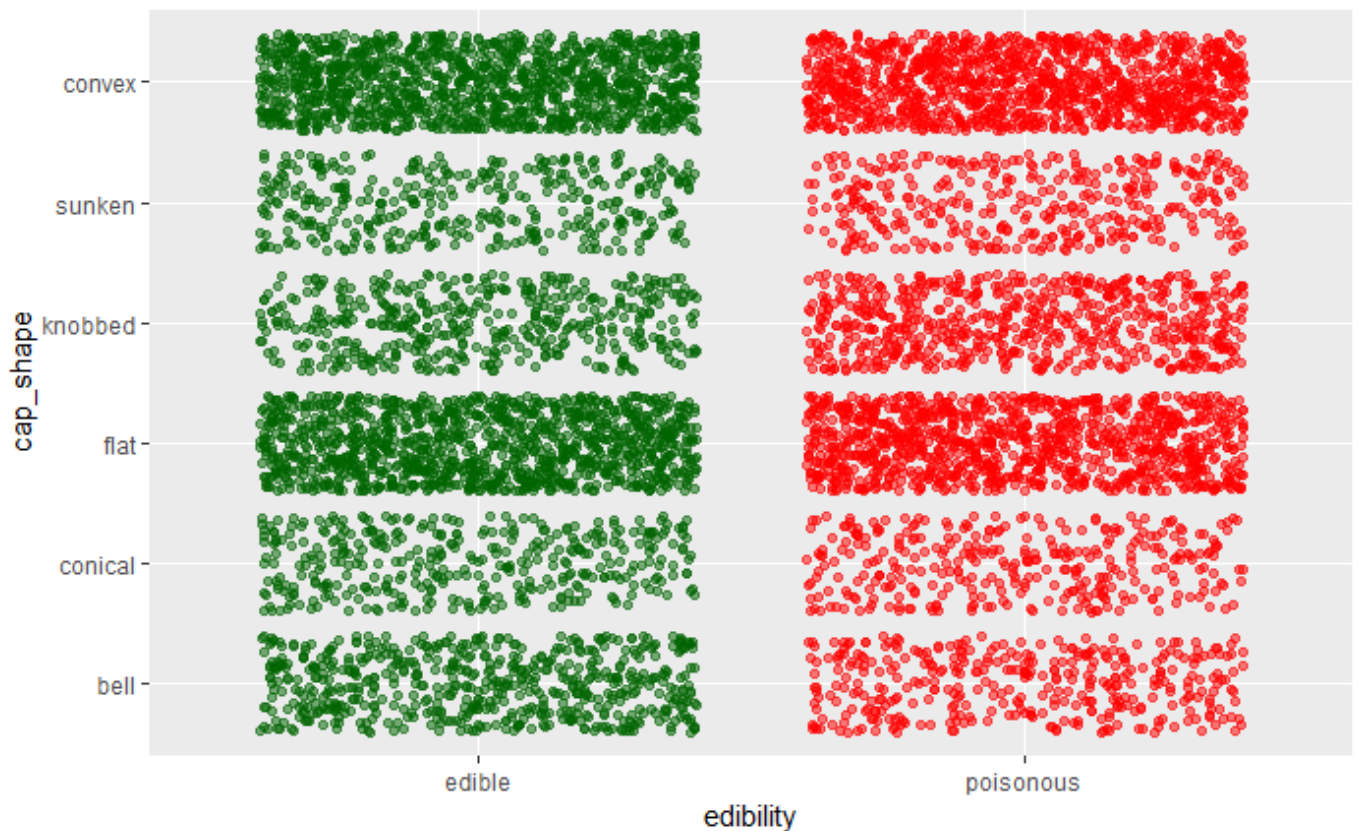
```
ggplot(setas, aes(x = edibility, y = cap_shape, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
                    values = c("dark green", "red") ) + theme(legend.position = "none")
```


[Hide](#)

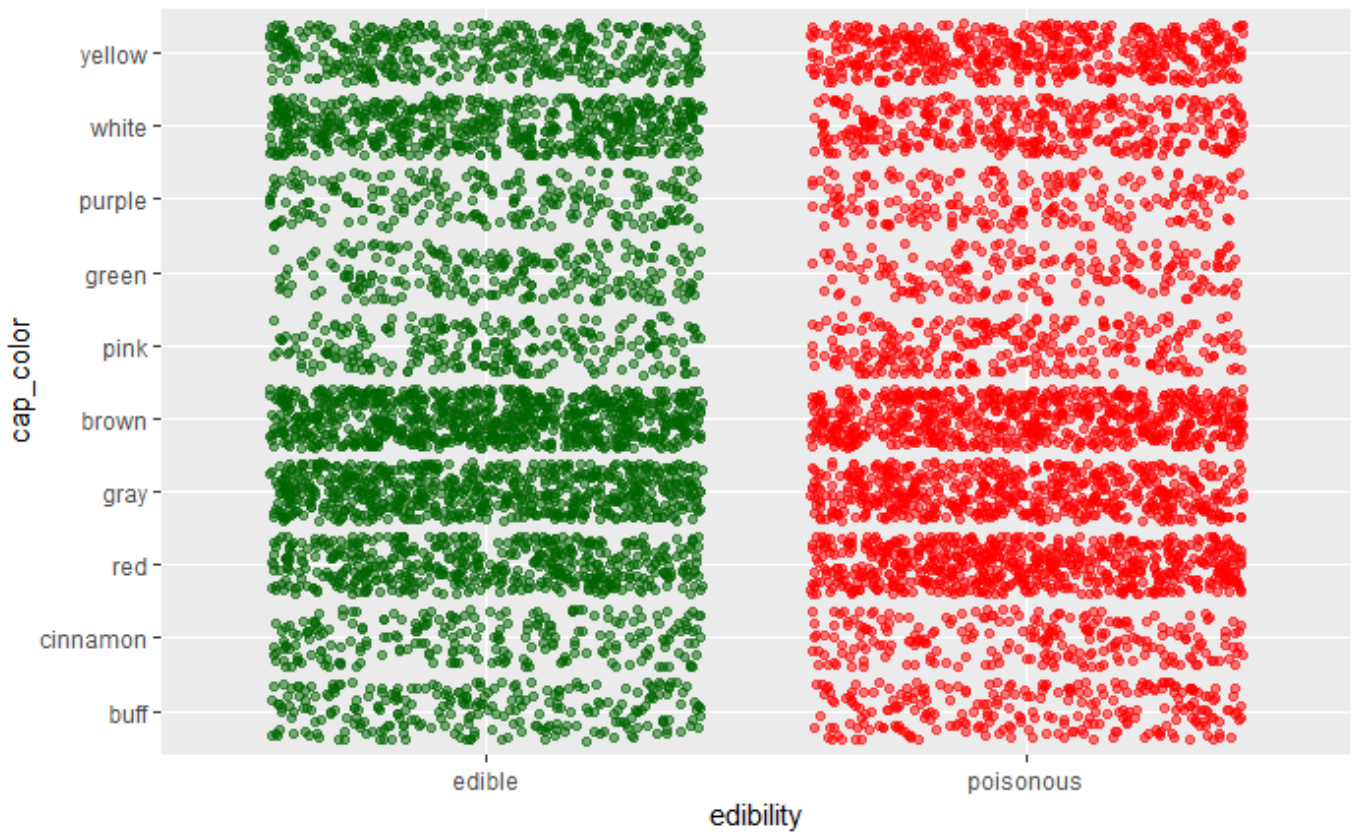
```
ggplot(setas, aes(x = edibility, y = cap_surface, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
                    values = c("dark green", "red") ) + theme(legend.position = "none")
```


[Hide](#)

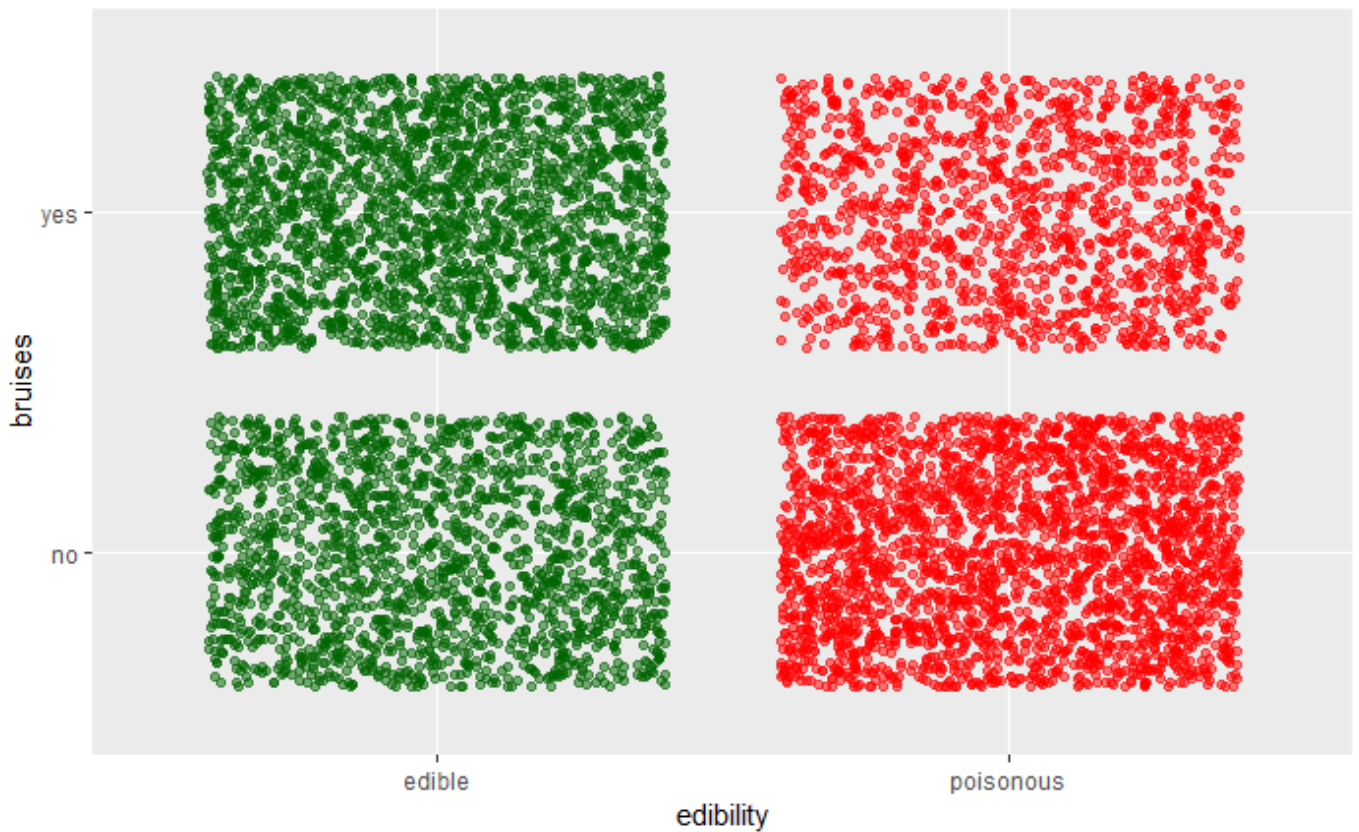
```
ggplot(setas, aes(x = edibility, y = cap_shape, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
                    values = c("dark green", "red") ) + theme(legend.position = "none")
```


[Hide](#)

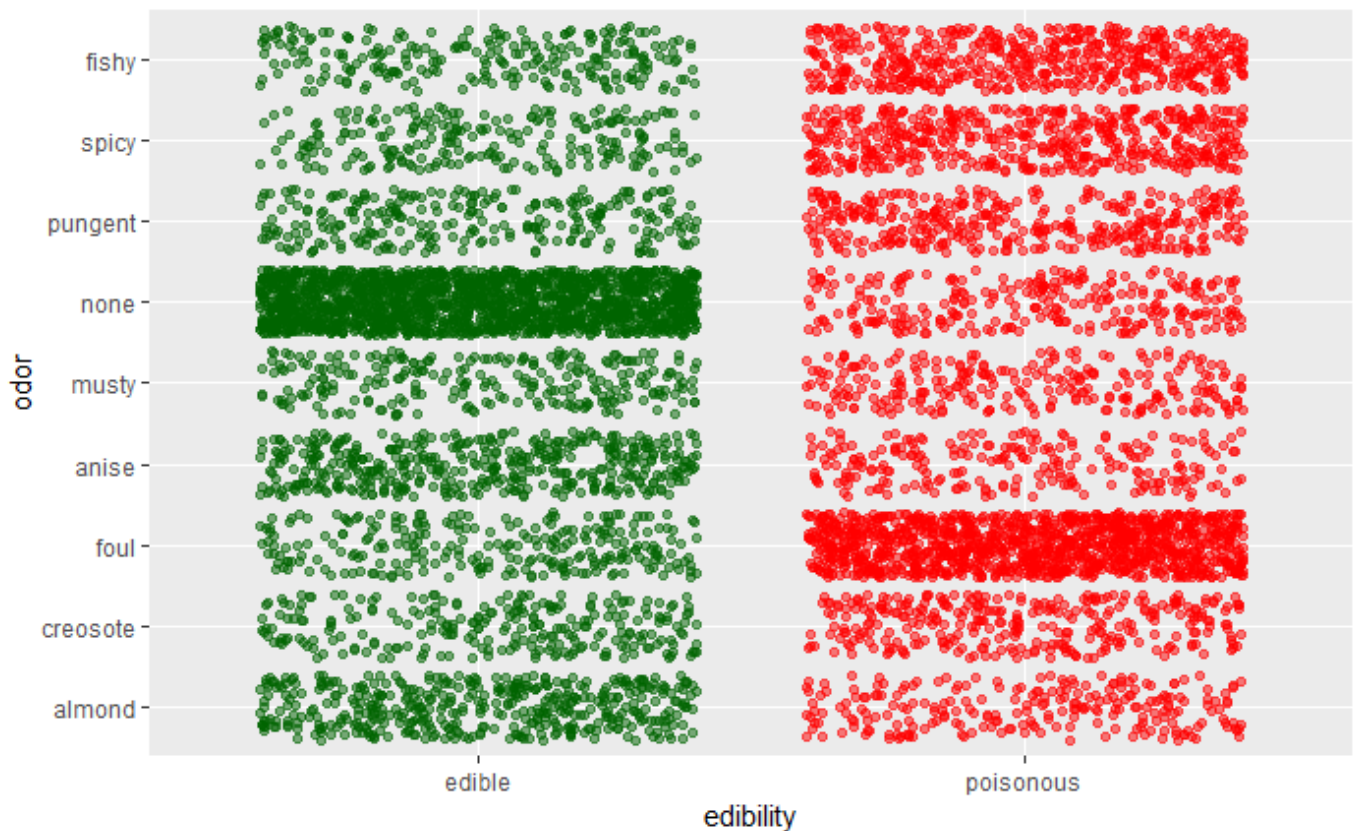

```
ggplot(setas, aes(x = edibility, y = cap_color, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
    values = c("dark green", "red") ) + theme(legend.position = "none")
```


[Hide](#)

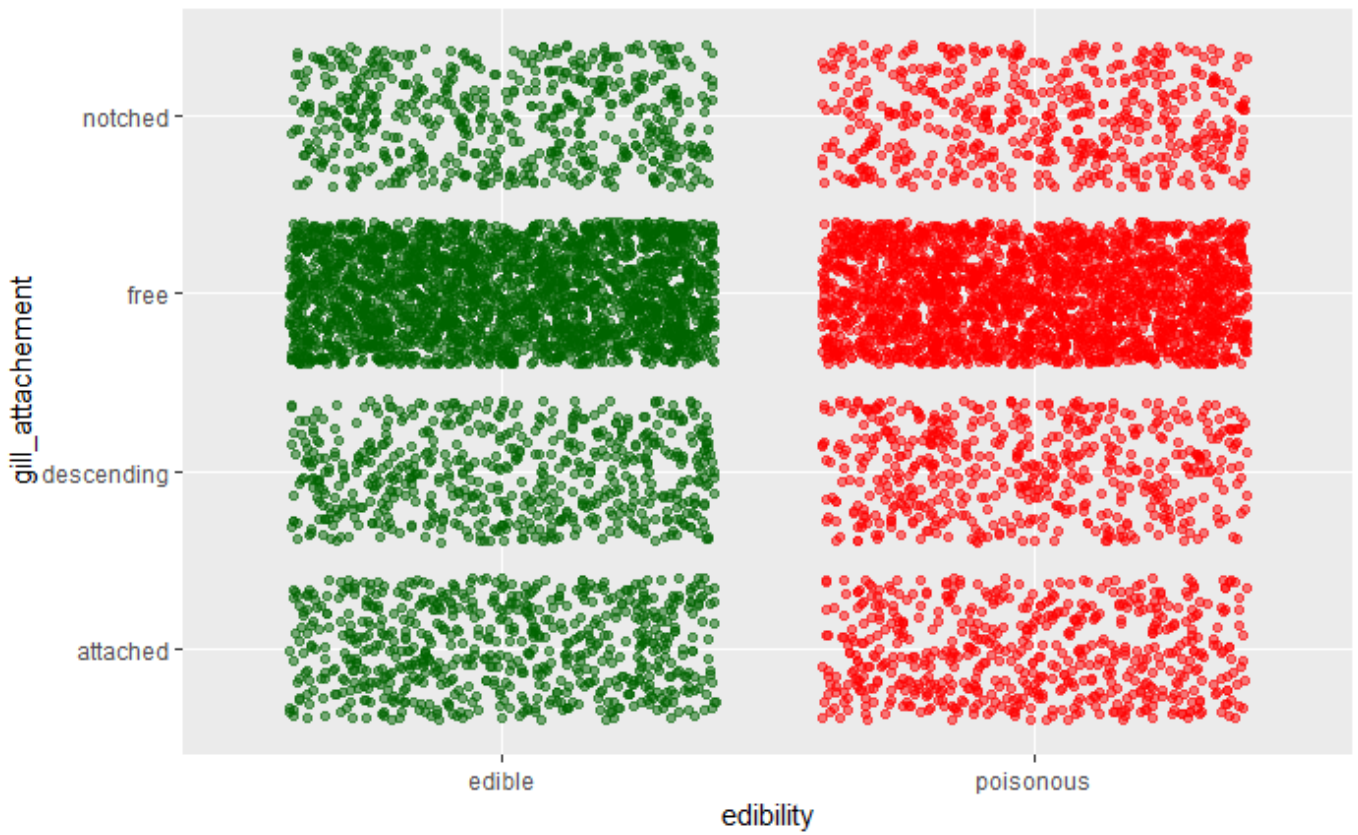
```
ggplot(setas, aes(x = edibility, y = bruises, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
    values = c("dark green", "red") ) + theme(legend.position = "none")
```



[Hide](#)

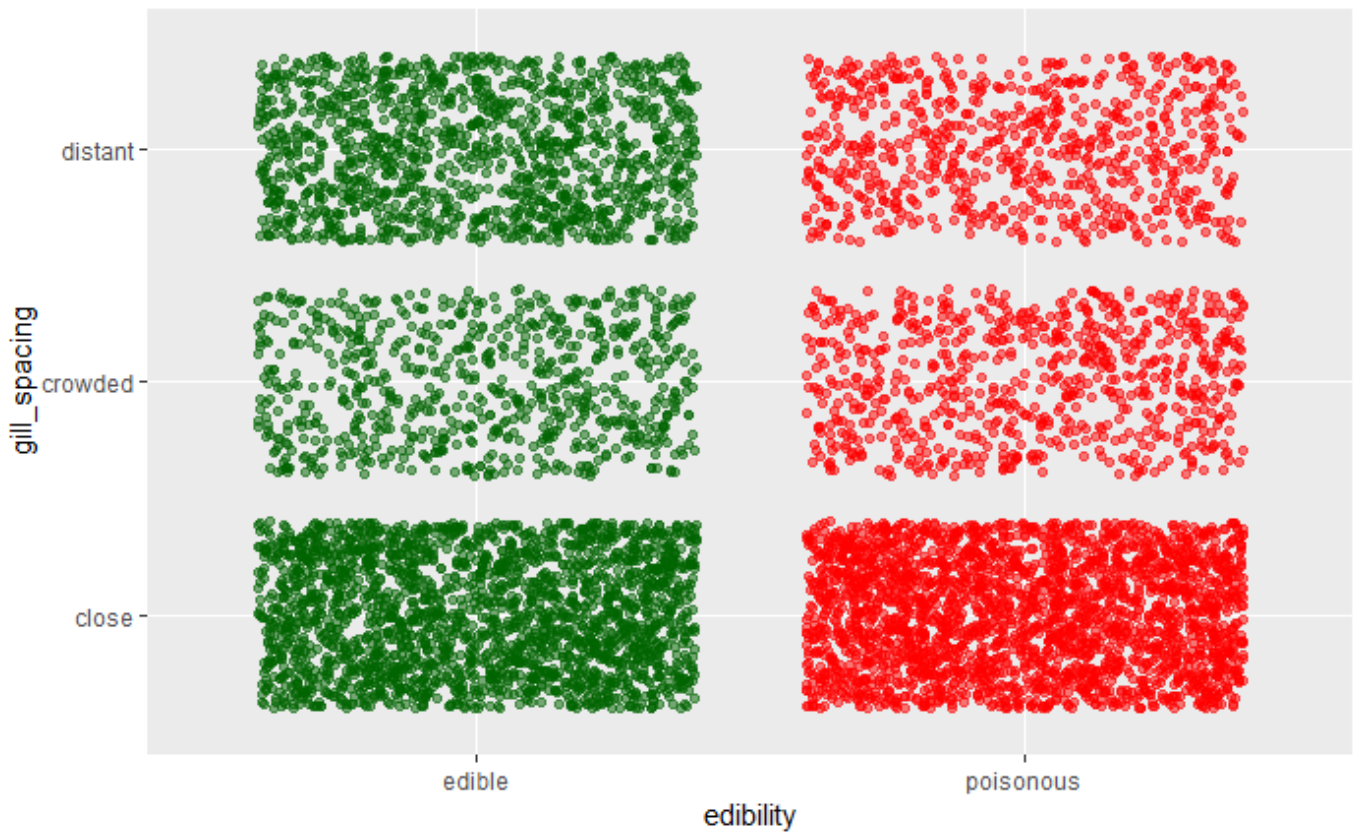
```
ggplot(setas, aes(x = edibility, y = odor, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
                    values = c("dark green", "red")) + theme(legend.position = "none")
```


[Hide](#)

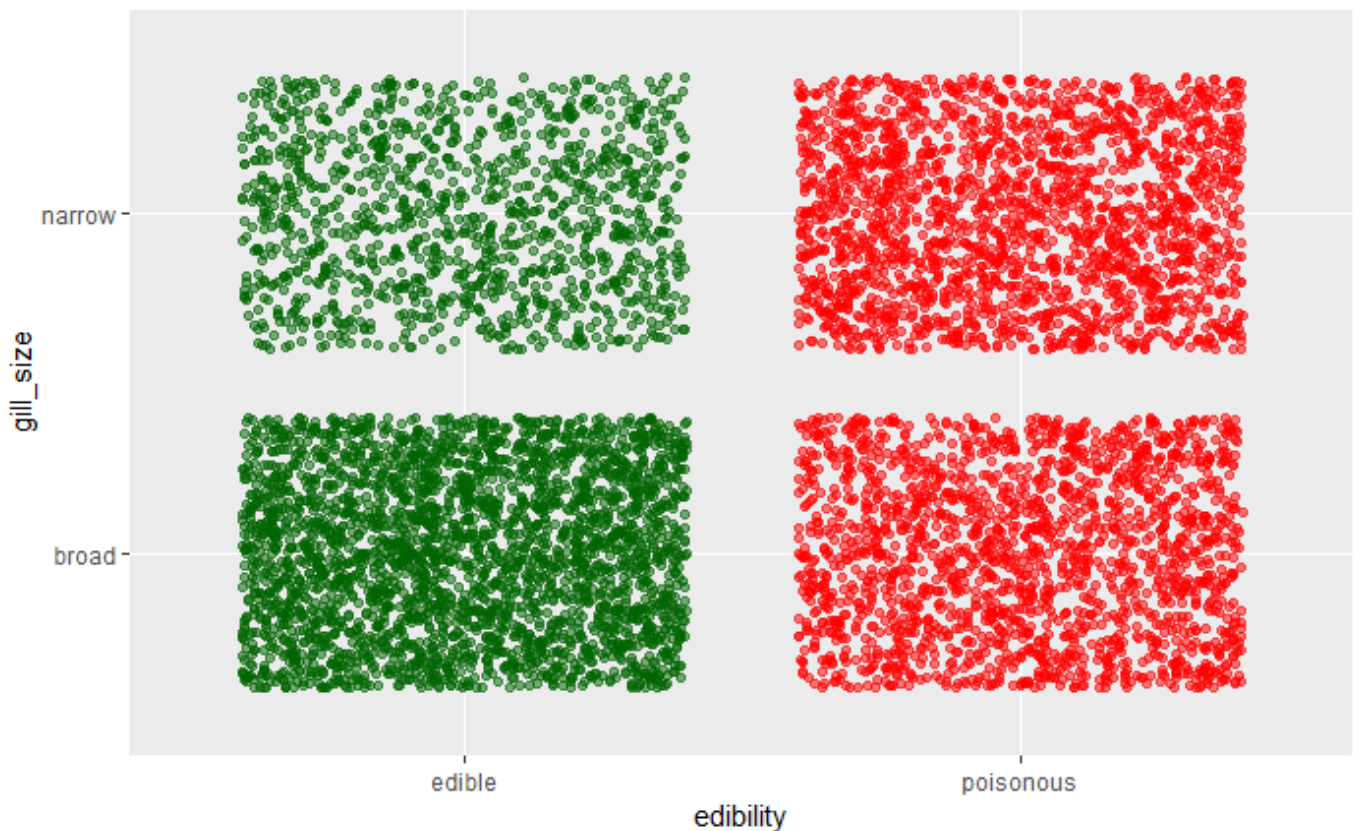
```
ggplot(setas, aes(x = edibility, y = gill_attachement, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
    values = c("dark green", "red") ) + theme(legend.position = "none")
```


[Hide](#)

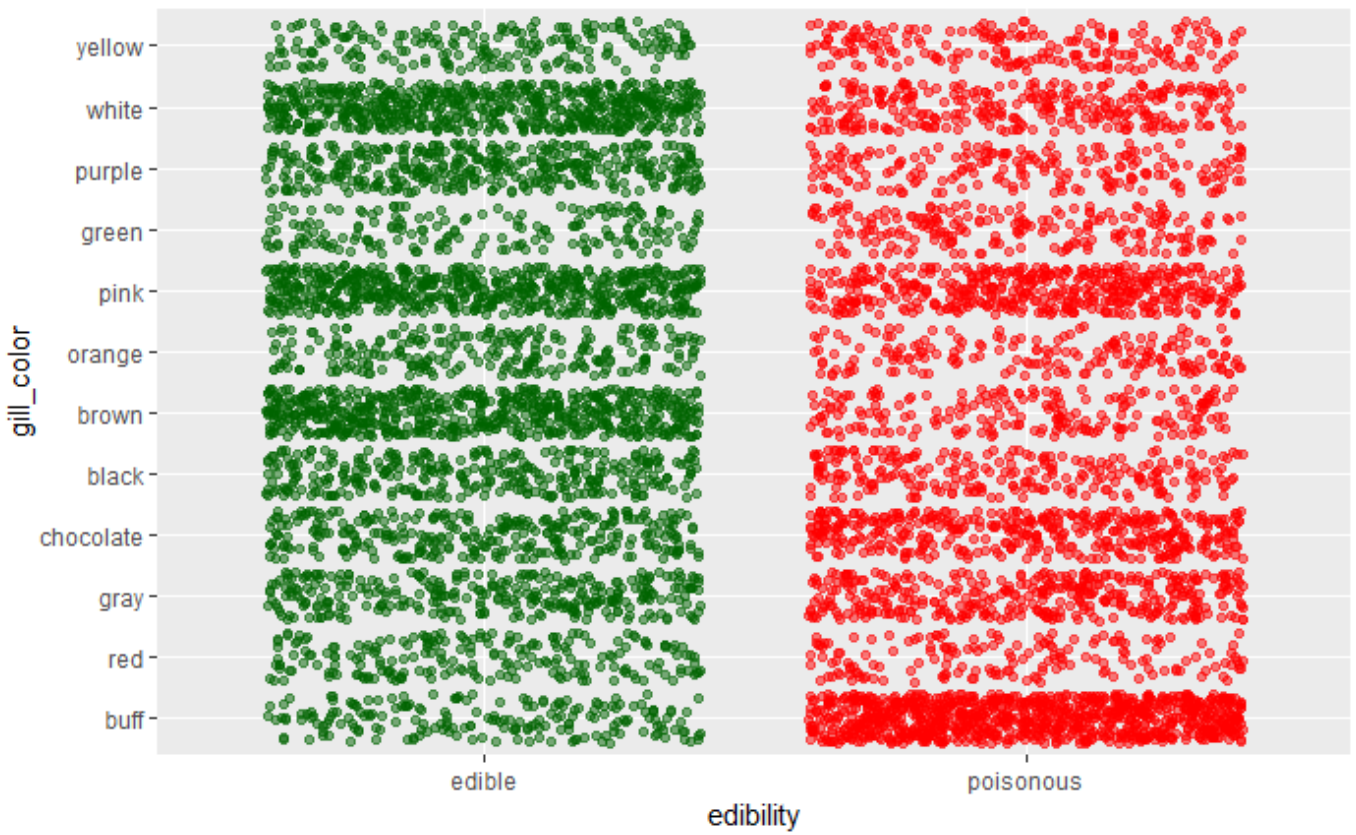
```
ggplot(setas, aes(x = edibility, y = gill_spacing, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
    values = c("dark green", "red") ) + theme(legend.position = "none")
```


[Hide](#)

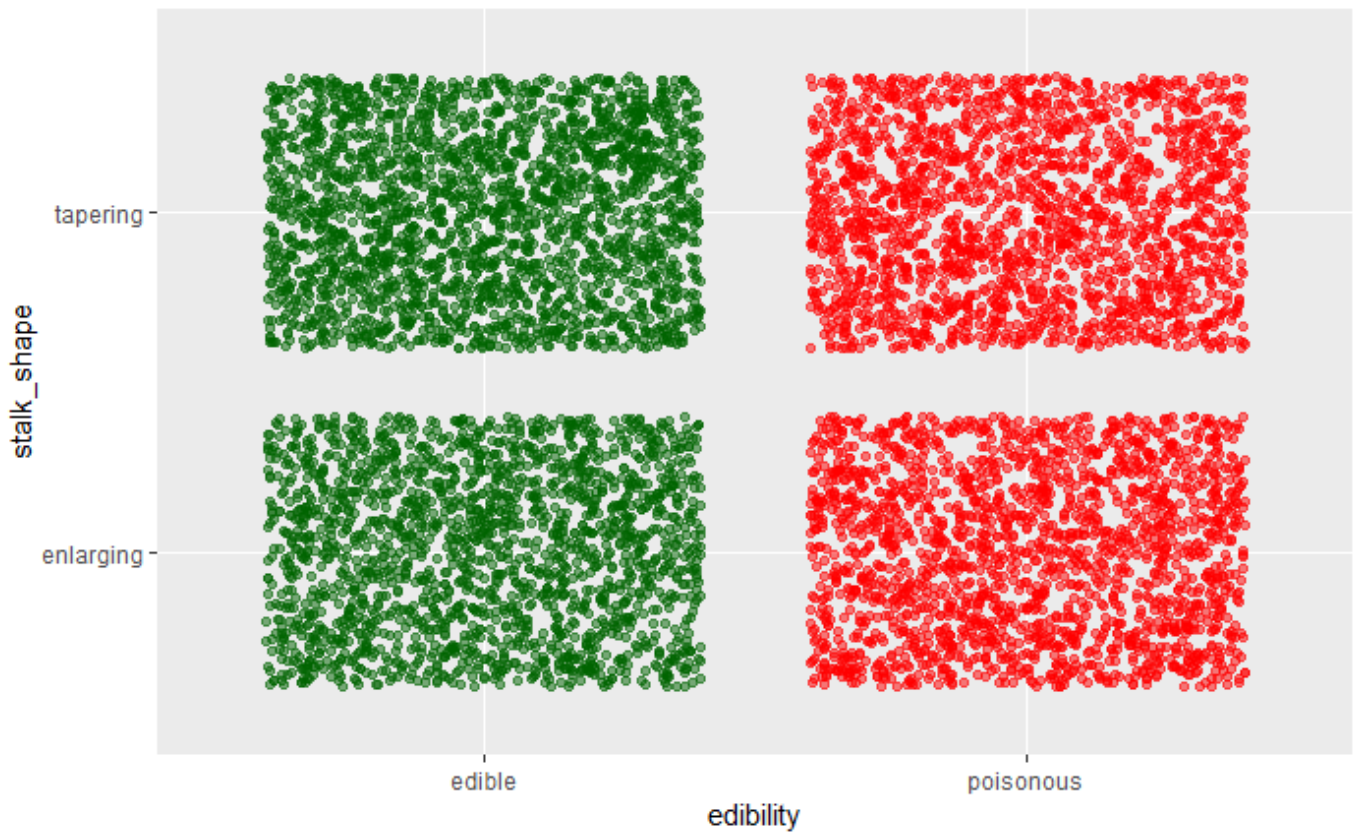
```
ggplot(setas, aes(x = edibility, y = gill_size, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
                    values = c("dark green", "red")) + theme(legend.position = "none")
```


[Hide](#)

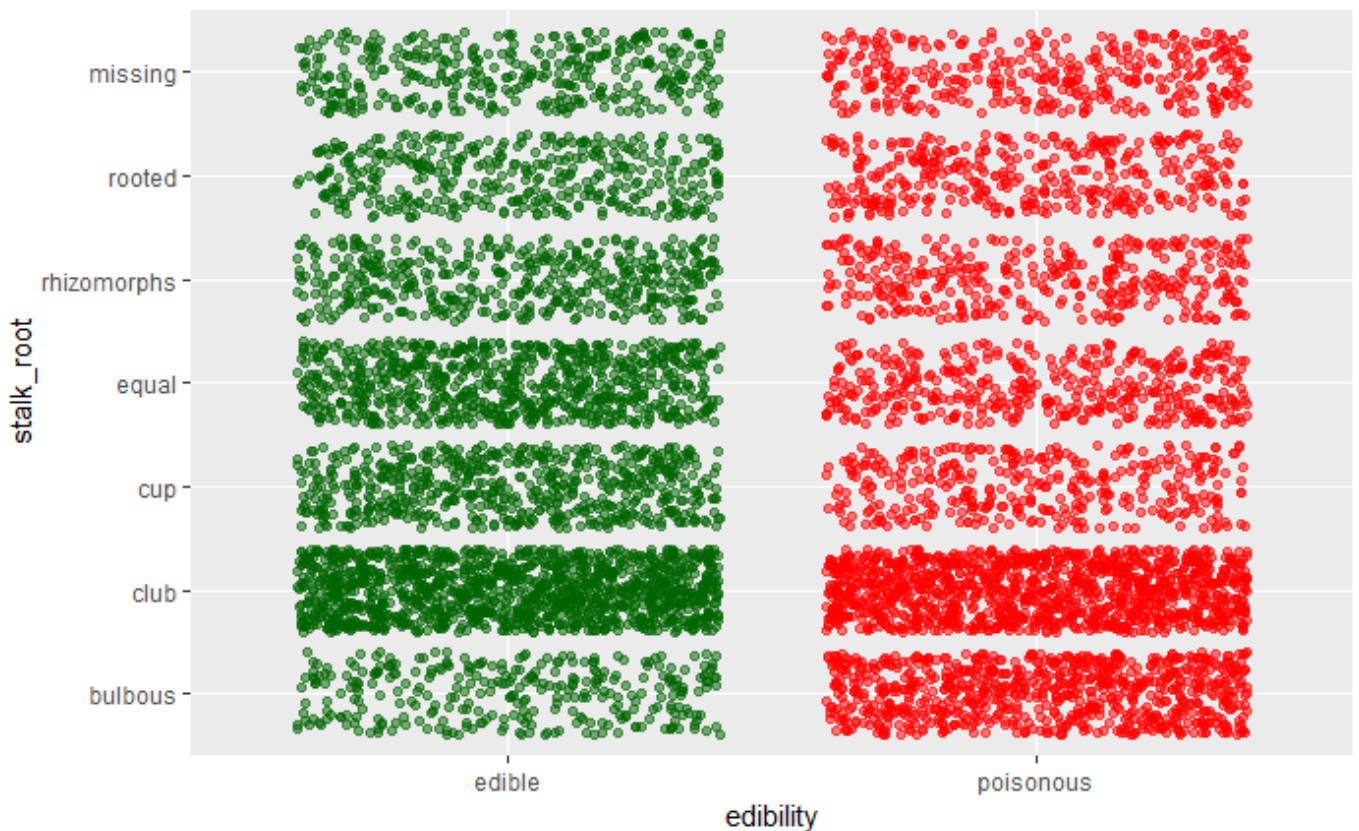

```
ggplot(setas, aes(x = edibility, y = gill_color, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
    values = c("dark green", "red") ) + theme(legend.position = "none")
```


[Hide](#)

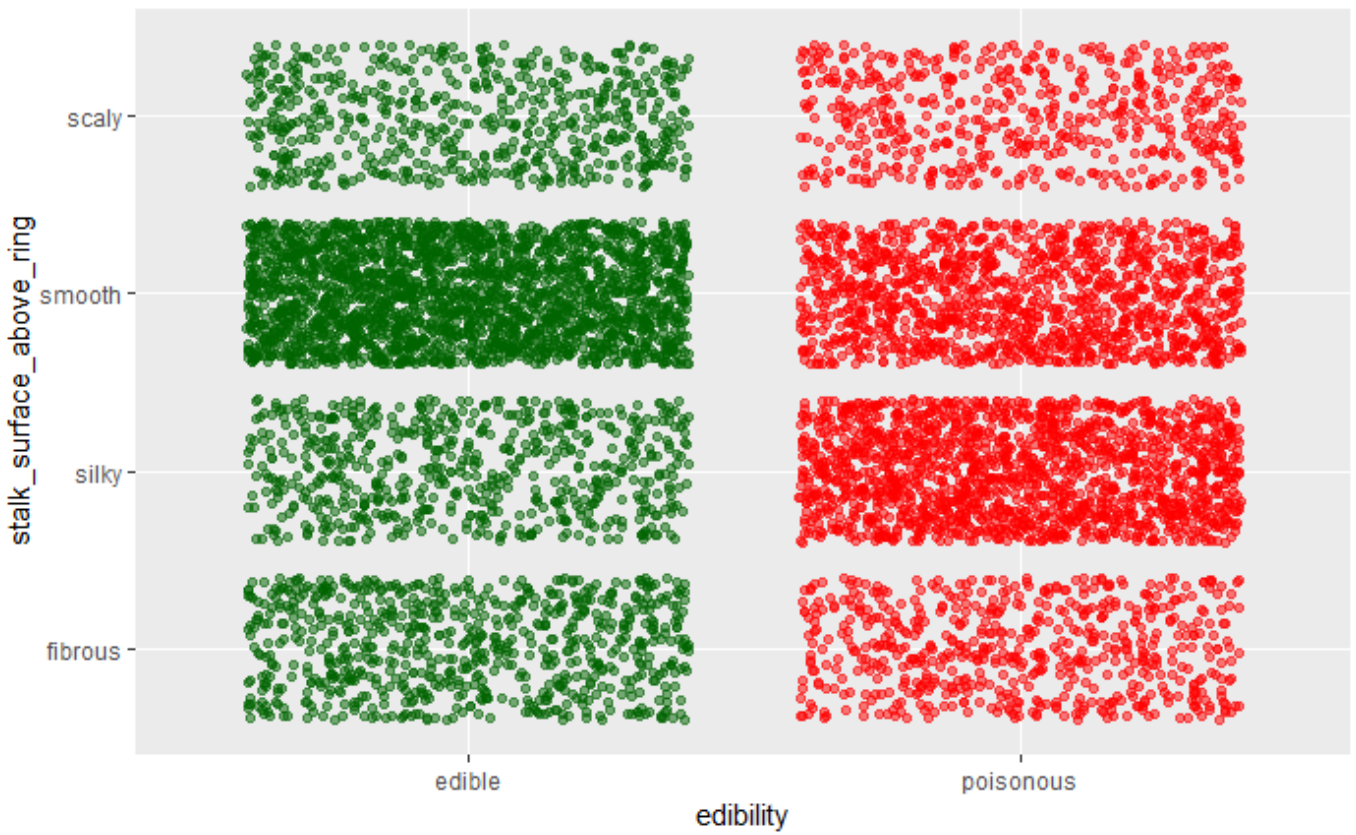
```
ggplot(setas, aes(x = edibility, y = stalk_shape, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
    values = c("dark green", "red") ) + theme(legend.position = "none")
```


[Hide](#)

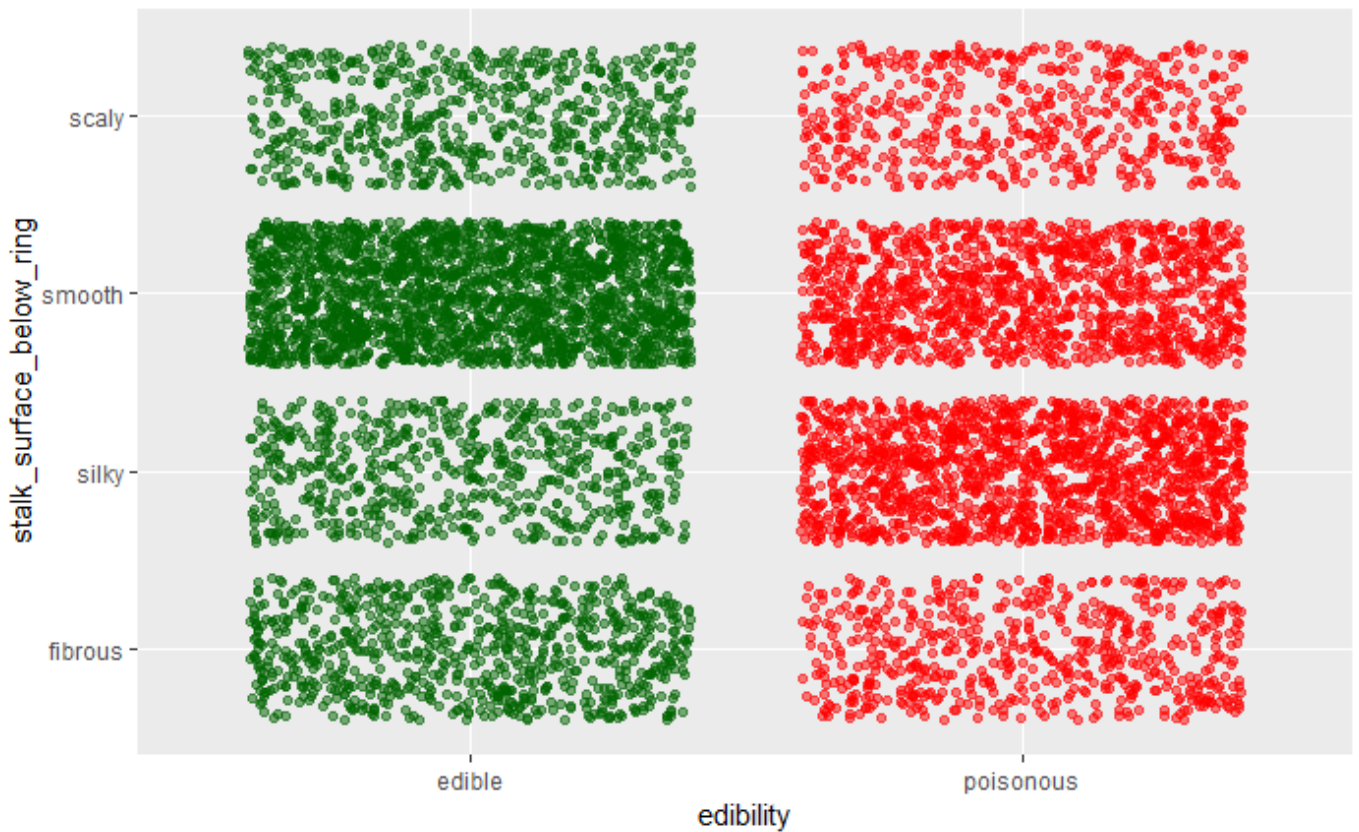
```
ggplot(setas, aes(x = edibility, y = stalk_root, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
                    values = c("dark green", "red") ) + theme(legend.position = "none")
```


[Hide](#)

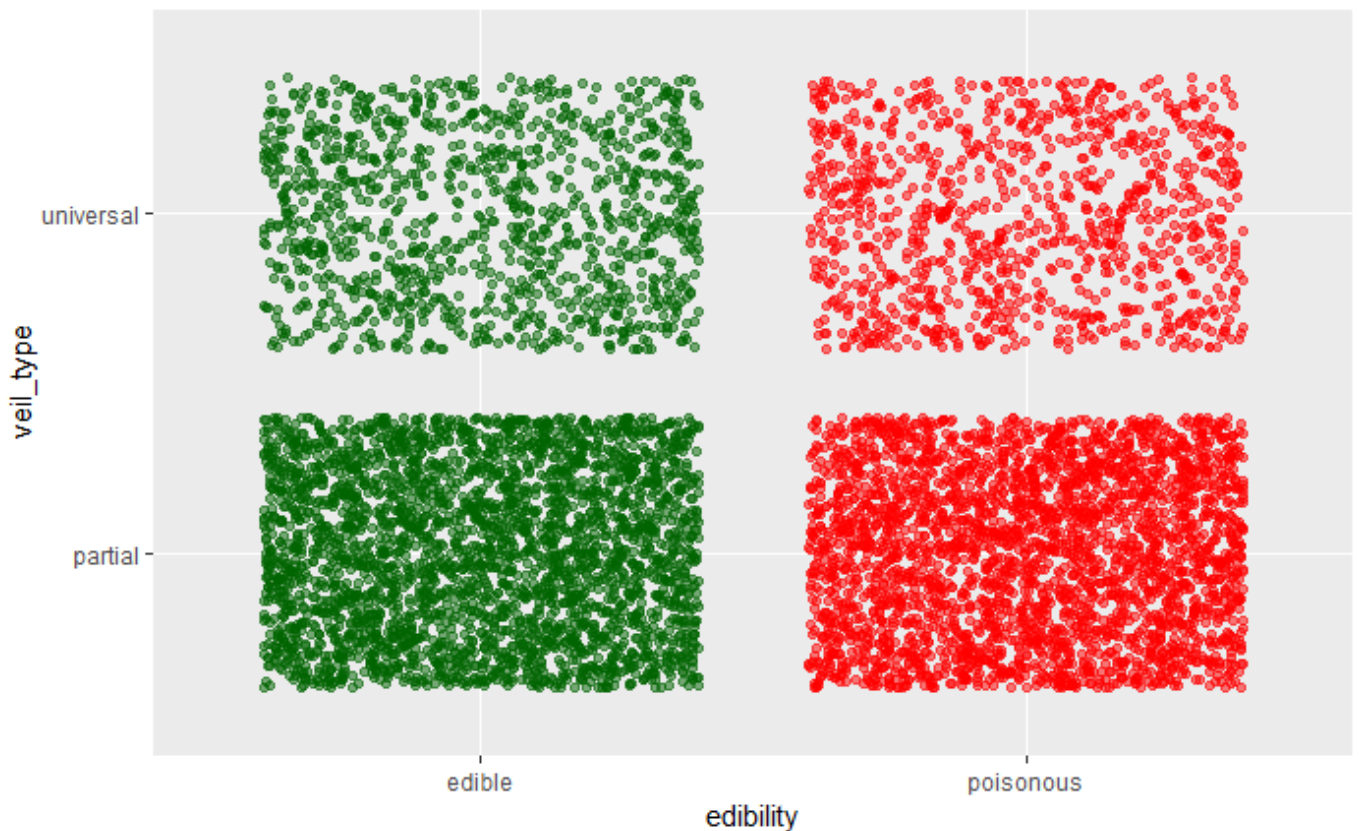
```
ggplot(setas, aes(x = edibility, y = stalk_surface_above_ring, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
    values = c("dark green", "red") ) + theme(legend.position = "none")
```


[Hide](#)

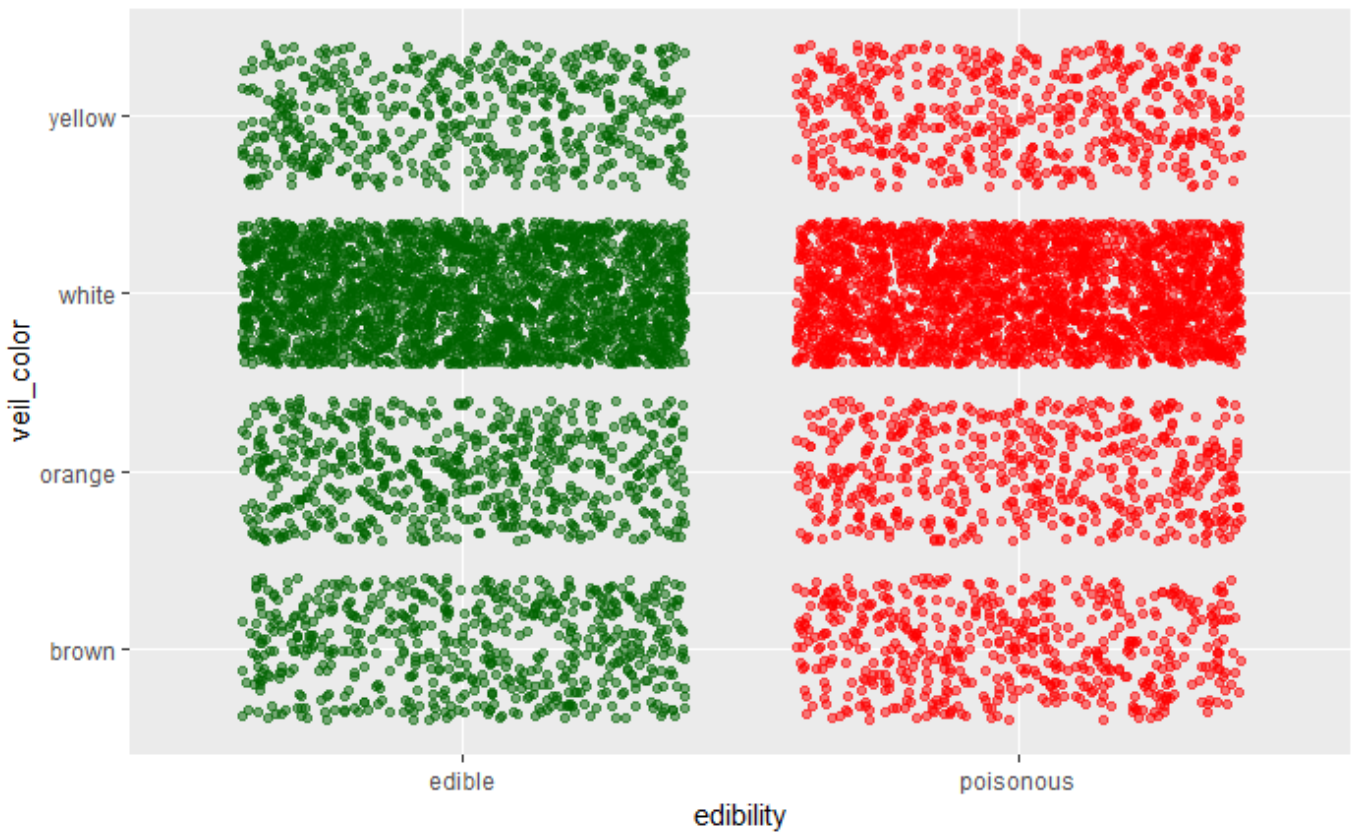
```
ggplot(setas, aes(x = edibility, y = stalk_surface_below_ring, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
    values = c("dark green", "red") ) + theme(legend.position = "none")
```



[Hide](#)

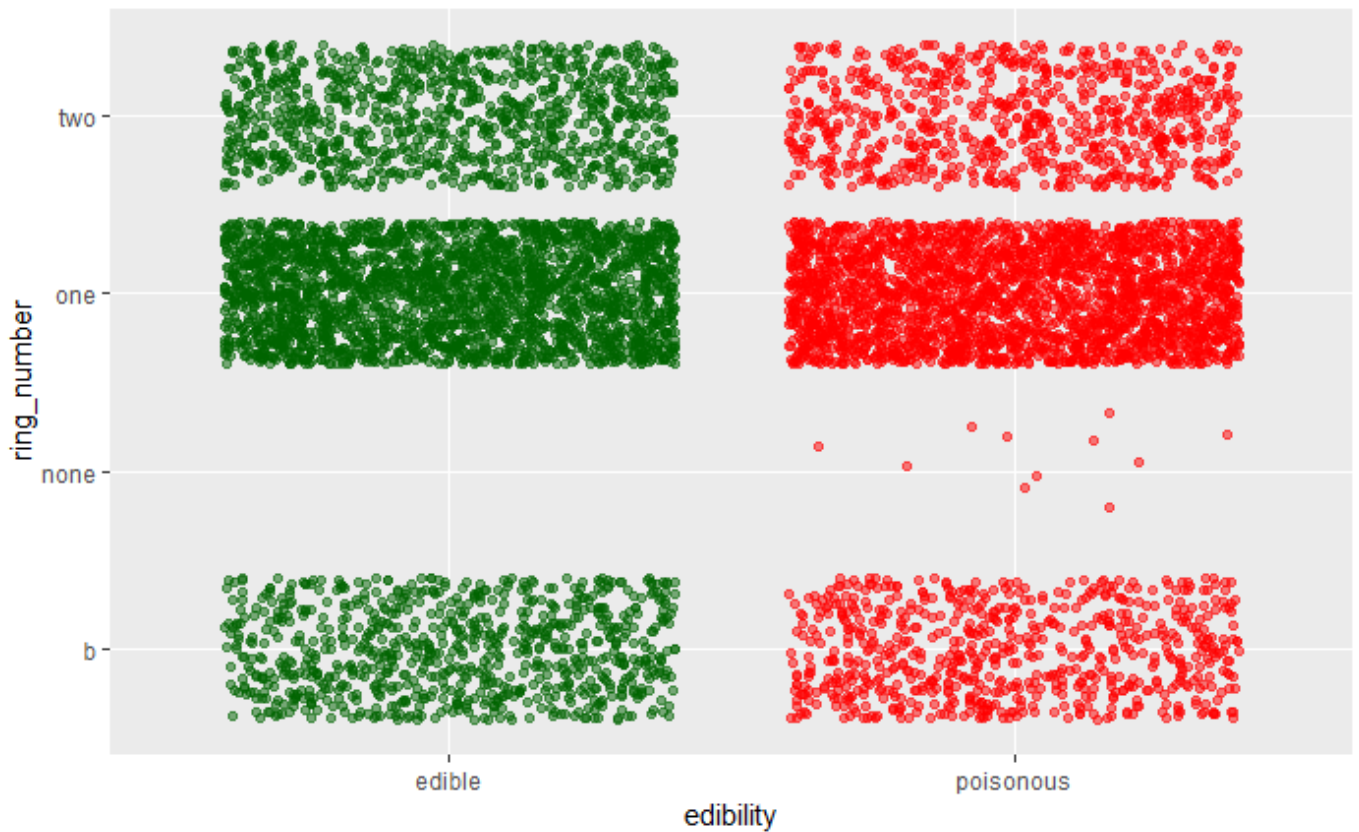
```
ggplot(setas, aes(x = edibility, y = veil_type, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
                    values = c("dark green", "red") ) + theme(legend.position = "none")
```


[Hide](#)

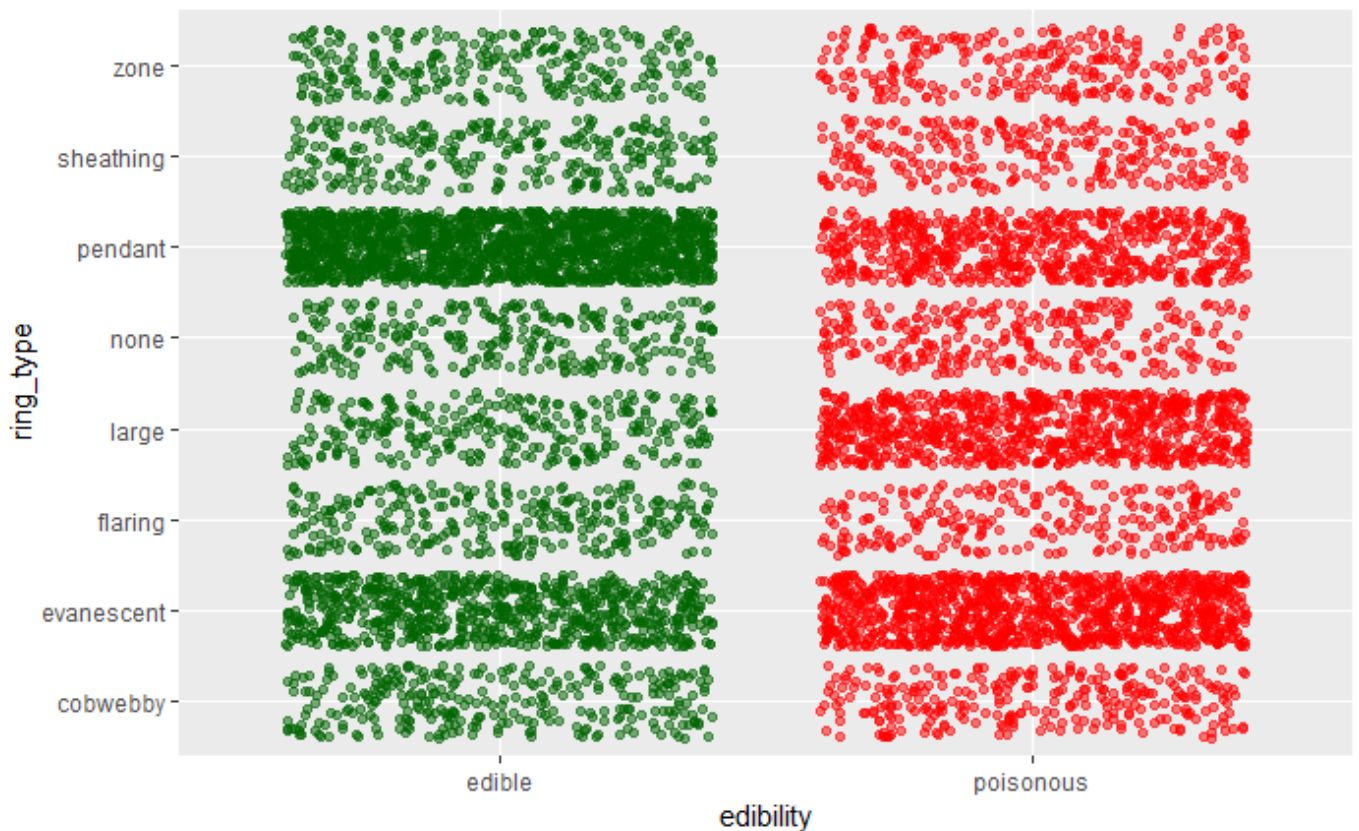
```
ggplot(setas, aes(x = edibility, y = veil_color, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
    values = c("dark green", "red") ) + theme(legend.position = "none")
```


[Hide](#)

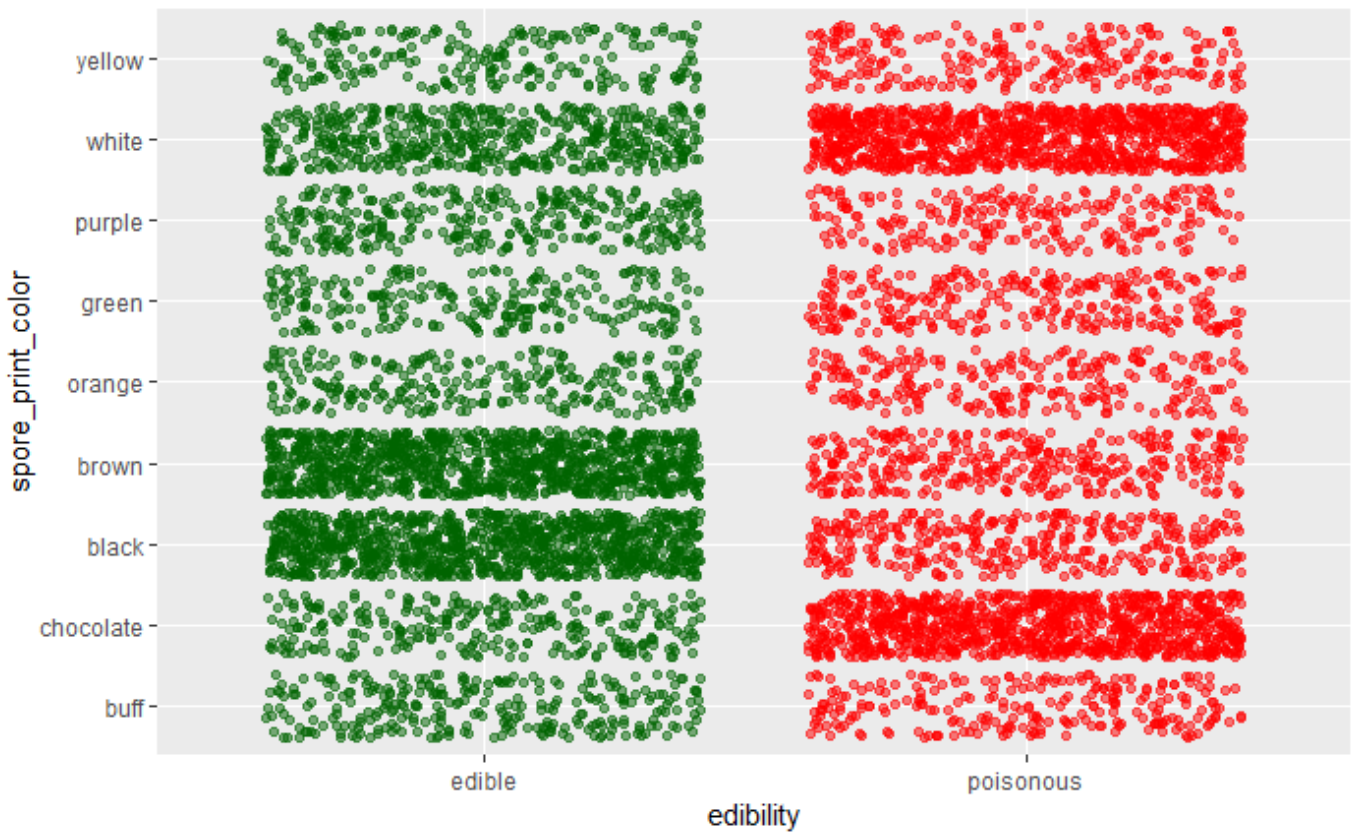
```
ggplot(setas, aes(x = edibility, y = ring_number, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
    values = c("dark green", "red") ) + theme(legend.position = "none")
```



[Hide](#)

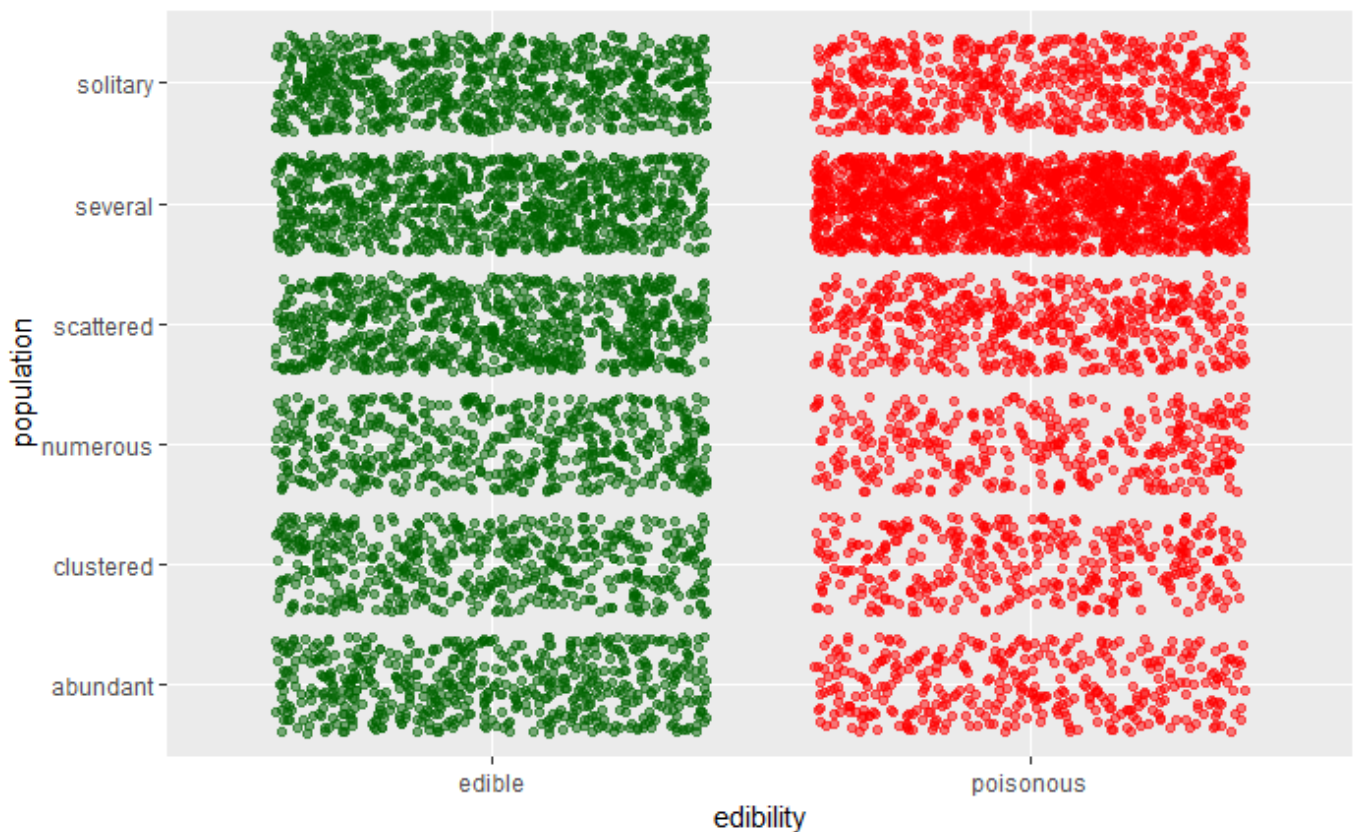
```
ggplot(setas, aes(x = edibility, y = ring_type, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
                    values = c("dark green", "red")) + theme(legend.position = "none")
```


[Hide](#)

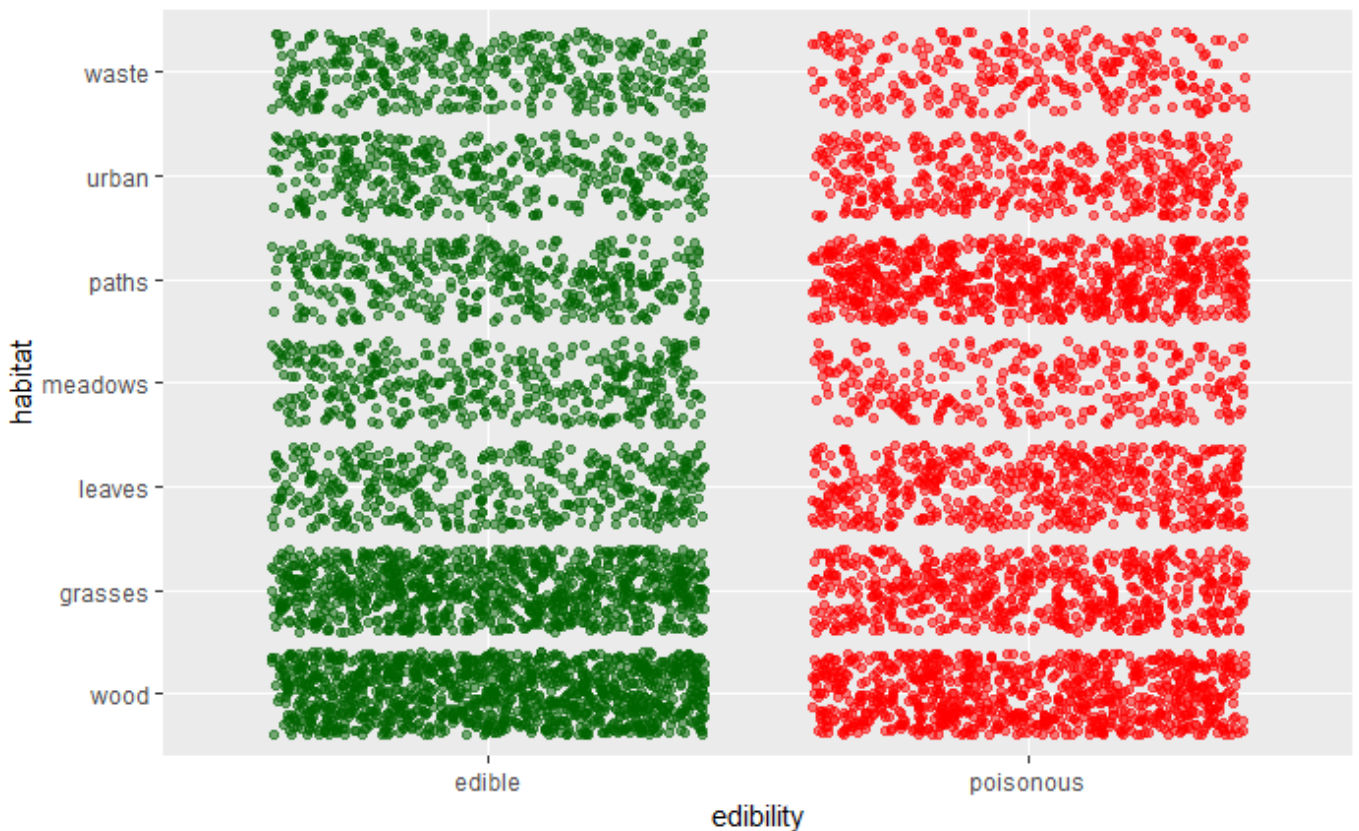
```
ggplot(setas, aes(x = edibility, y = spore_print_color, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
    values = c("dark green", "red") ) + theme(legend.position = "none")
```


[Hide](#)

```
ggplot(setas, aes(x = edibility, y = population, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
    values = c("dark green", "red") ) + theme(legend.position = "none")
```


[Hide](#)

```
ggplot(setas, aes(x = edibility, y = habitat, col = edibility,)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
                    values = c("dark green", "red")) + theme(legend.position = "none")
```



Analizamos un subconjunto de pares de variables que, por su significado, consideramos que podrían estar relacionados conjuntamente. Además también analizamos su influencia conjunta en la variable respuesta.

En cuanto a la relación entre las variables que conforman cada par observamos lo siguiente:

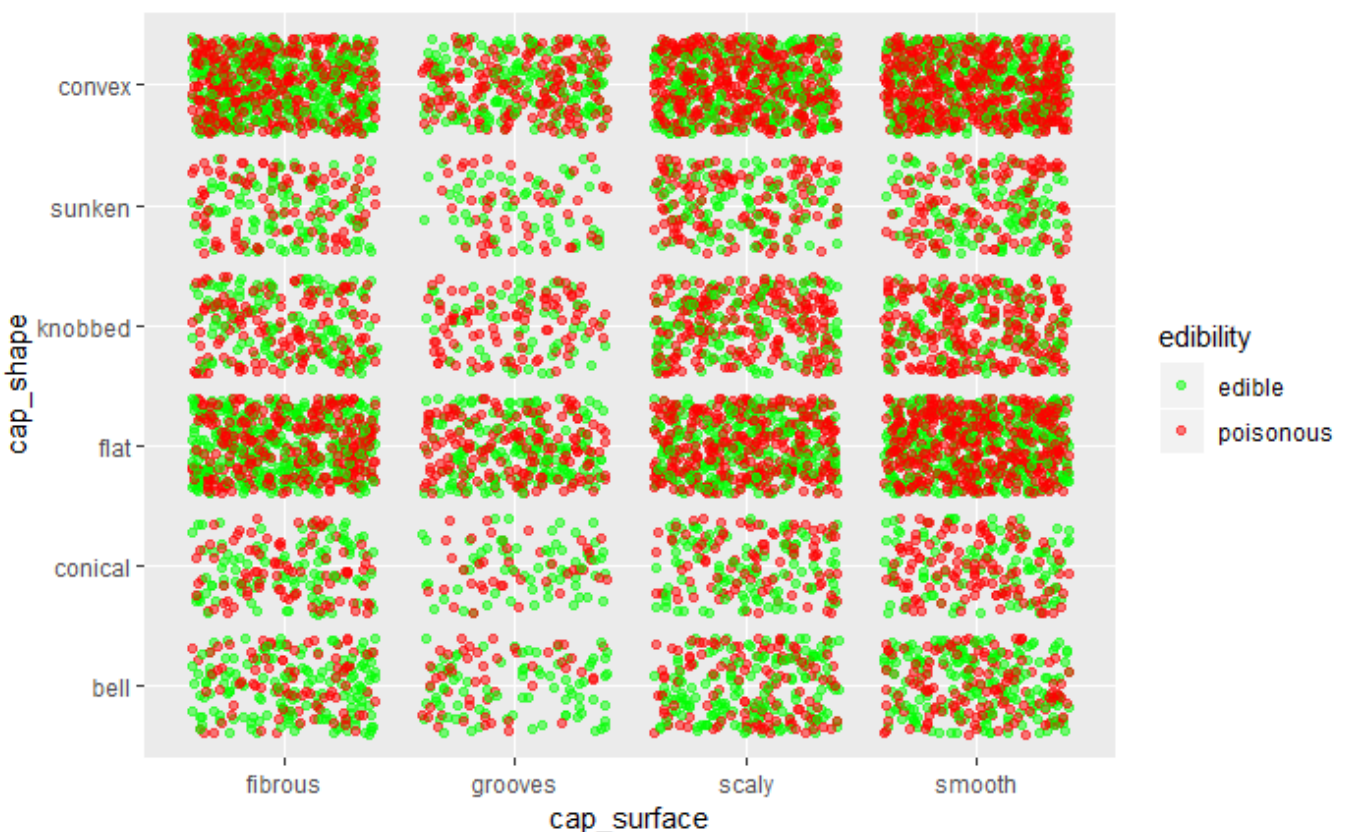
- Parece que en todos los pares existe cierta relación pero en algunos de densidad menor (superficie y forma del gorro, separación y tamaño de las agallas).
- Parece que el color del gorro y el color de las agallas están relacionados, así como también el olor con el lugar donde crece.

En cuanto a la influencia conjunta de las variables en si la seta es venenosa o no observamos lo siguiente:

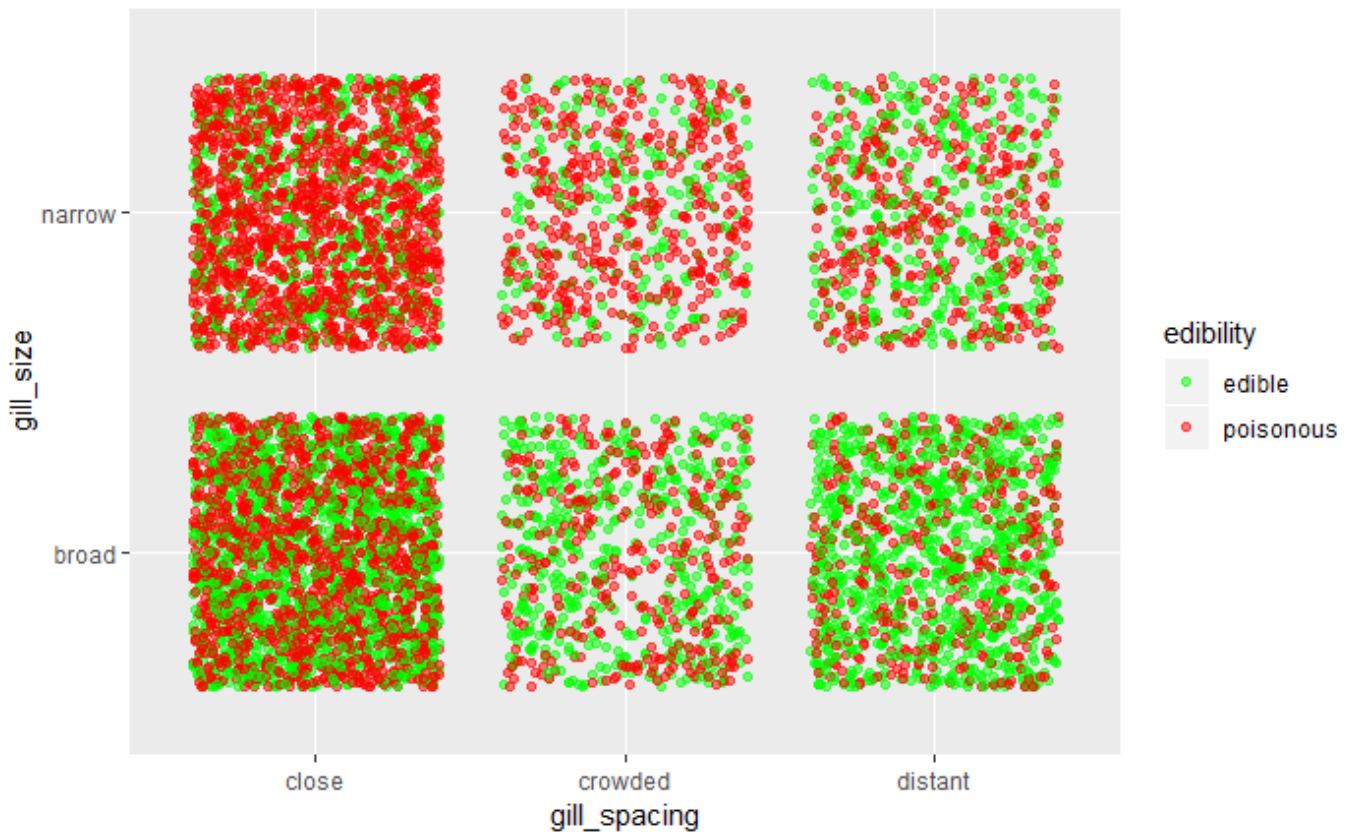
- Parece que las rojas que tienen las agallas color 'buff' y son rojas o marrones en el gorro tienen gran tendencia a ser venenosas.
- Por el contrario las que tienen agallas y gorro de color marrón tienden a ser comestibles.
- Las que tienen las agallas estrechas y cerca tienden a ser venenosas mientras que las que las tienen anchas y separadas parecen tender a ser comestibles.

[Hide](#)

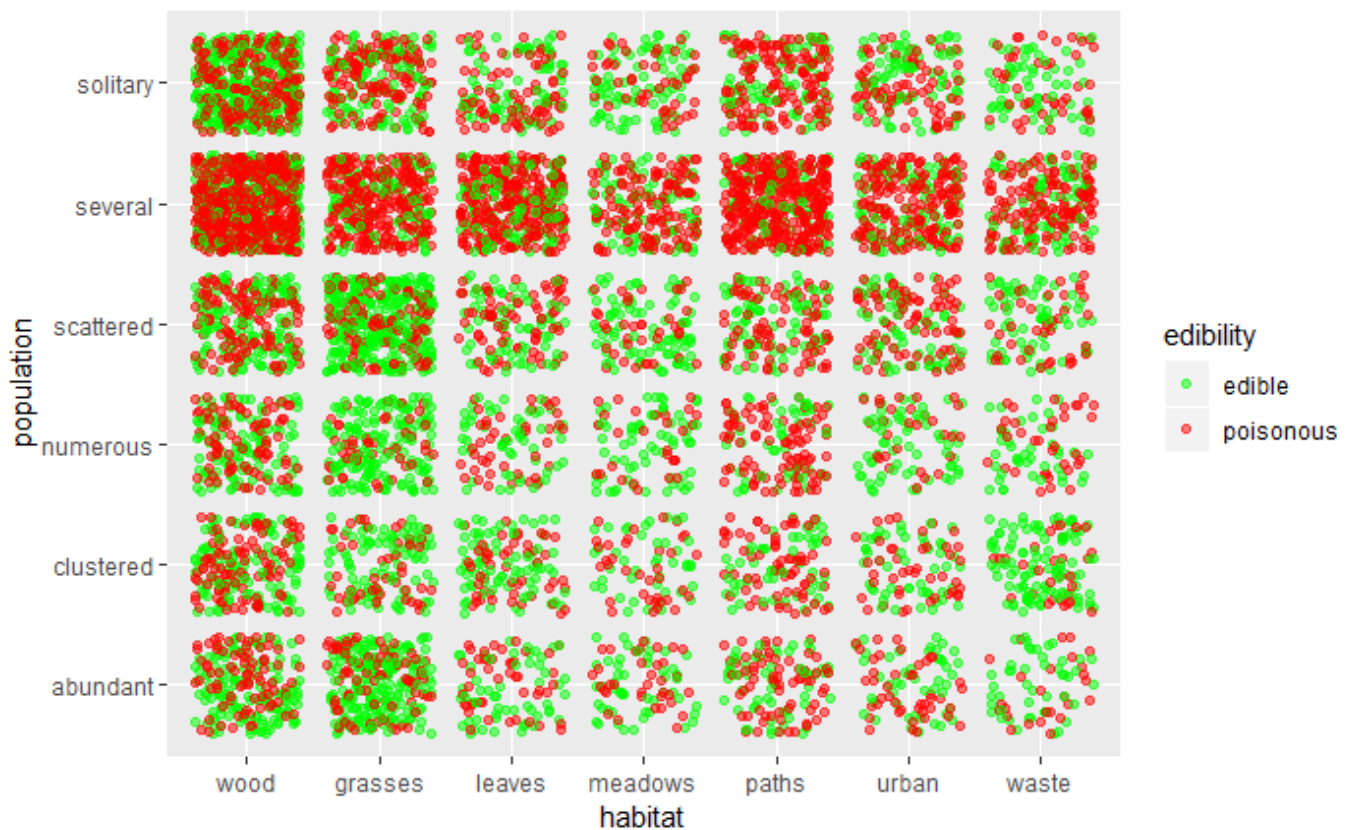
```
#cap_surface vs cap_shape
library(ggplot2)
ggplot(setas, aes(x = cap_surface, y = cap_shape, col = edibility)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
                    values = c("green", "red"))
```


[Hide](#)

```
#gill_spacing vs gill_size
ggplot(setas, aes(x = gill_spacing, y = gill_size, col = edibility)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
                    values = c("green", "red"))
```

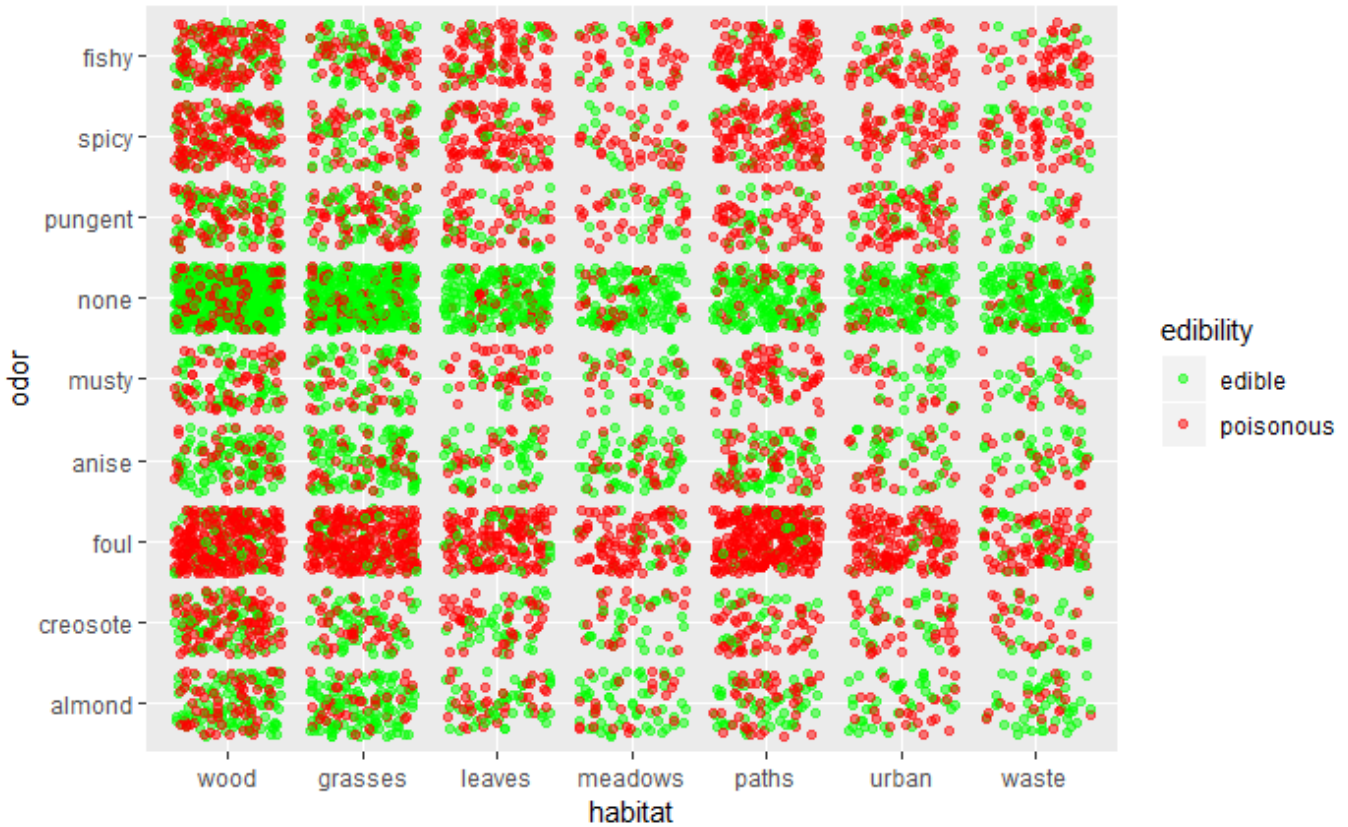

[Hide](#)

```
#habitat vs population
library(ggplot2)
ggplot(setas, aes(x = habitat, y = population, col = edibility)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
                    values = c("green", "red"))
```



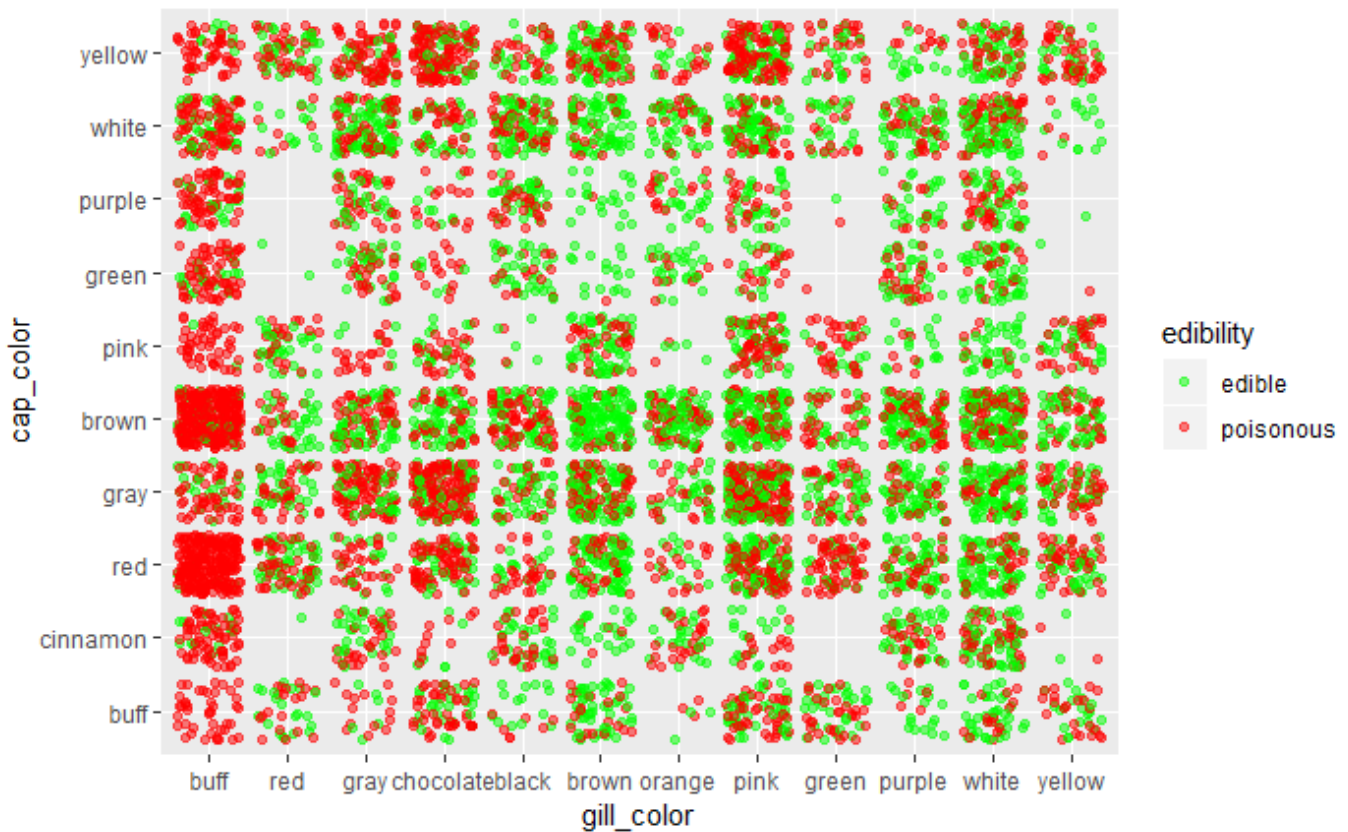
Hide

```
#habitat vs odor
ggplot(setas, aes(x = habitat, y = odor, col = edibility)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
                    values = c("green", "red"))
```



Hide

```
# gill_color y cap_color
ggplot(setas, aes(x = gill_color, y = cap_color, col = edibility)) +
  geom_jitter(alpha = 0.5) +
  scale_color_manual(breaks = c("edible", "poisonous"),
                    values = c("green", "red"))
```

Comenzaremos analizando de forma cuantitativa la correlación de los pares de variables anteriores utilizando tests de significancia: mediante el test de independencia de chi cuadrado de Pearson.

- Solo en un caso el p-value fue superior al nivel de significancia mínimo 0,05, el del par de variables cap_shape y cap_surface, por lo que solo en este caso no podemos negar la hipótesis nula de que ambas variables son independientes.
- Para los demás pares parece que no existe independencia entre sus miembros.

Hide

```
#cap_shape vs cap_surface
chisq.test(setas$cap_shape, setas$cap_surface, correct = FALSE)
```

Pearson's Chi-squared test

```
data: setas$cap_shape and setas$cap_surface
X-squared = 19.125, df = 15, p-value = 0.2081
```

Hide

```
#gill_spacing vs gill_size
chisq.test(setas$gill_spacing, setas$gill_size, correct = FALSE)
```

Pearson's Chi-squared test

```
data: setas$gill_spacing and setas$gill_size
X-squared = 20.789, df = 2, p-value = 3.06e-05
```

Hide

```
#habitat vs population
chisq.test(setas$habitat, setas$population, correct = FALSE)
```

Pearson's Chi-squared test

```
data: setas$habitat and setas$population
X-squared = 341.22, df = 30, p-value < 2.2e-16
```

[Hide](#)

```
#habitat vs odor
chisq.test(setas$habitat, setas$odor, correct = FALSE)
```

Pearson's Chi-squared test

```
data: setas$habitat and setas$odor
X-squared = 246.89, df = 48, p-value < 2.2e-16
```

[Hide](#)

```
# gill_color y cap_color
chisq.test(setas$gill_color, setas$cap_color, correct = FALSE)
```

Pearson's Chi-squared test

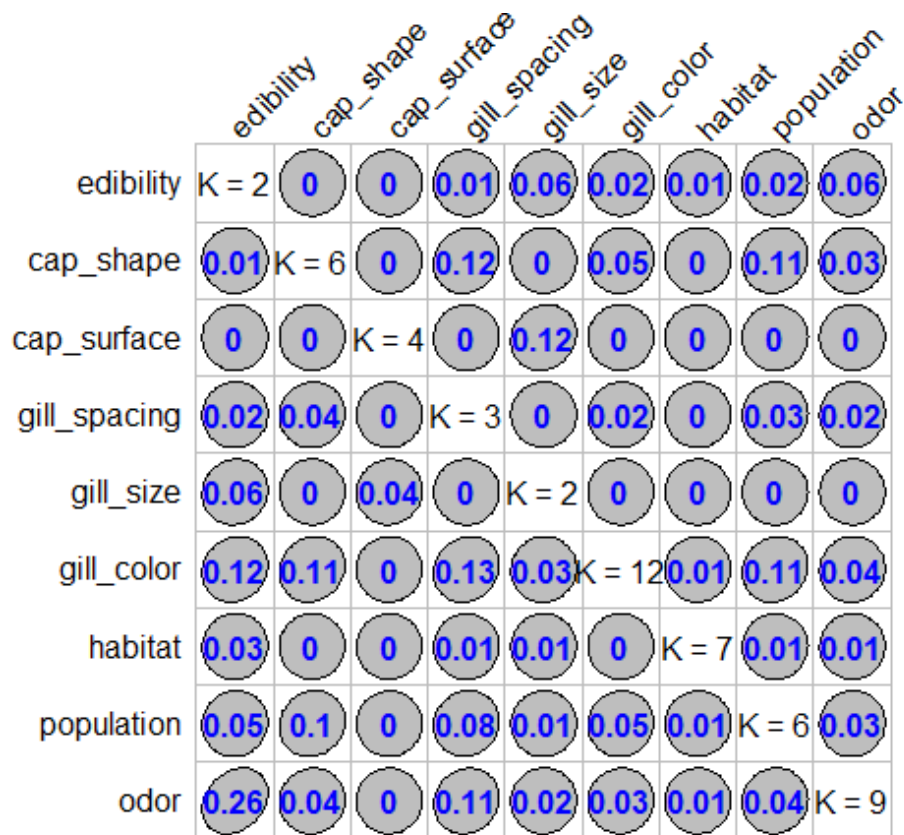
```
data: setas$gill_color and setas$cap_color
X-squared = 1422.4, df = 99, p-value < 2.2e-16
```

A continuación mediremos el grado de asociación entre todas las variables. Para ello utilizaremos Goodman-Kruskal Tay, que nos mide el porcentaje de mejora en predictabilidad de una variable dado el valor de la otra.

Los valores más altos de asociación los encontramos entre olor y comestibilidad (con gran distancia al resto), el color y el espaciado de las agallas y el color y la comestibilidad.

[Hide](#)

```
varset1<- c("edibility","cap_shape","cap_surface","gill_spacing","gill_size","gill_color",
"habitat", "population", "odor")
setas2<- subset(setas, select = varset1)
GKmatrix1<- GKtauDataframe(setas2)
plot(GKmatrix1, corrColors = "blue")
```

Outliers

Cuando todas las variables son cualitativas es más complicado encontrar outliers. Tras el análisis de las distribuciones de los valores de cada variable, solamente podemos considerar que los samples que tienen como valor de ring_number none (antes 'n') pueden ser considerados outliers al tratarse solo de 11 ejemplares de los más de 8000.

[Hide](#)

```
length (which (setas$ring_number=='none'))
```

```
[1] 11
```

Del análisis de los pares de variables juntos también se puede desprender que son muy infrecuentes y por lo tanto considerables outliers aquellos ejemplares con las agallas rojas y el gorro verde, así como también las combinaciones rojo-canela, verde-verde y amarillo-violeta.

[Hide](#)

```
length (which (setas$gill_color=='red' & setas$cap_color=='green'))
```

```
[1] 2
```

[Hide](#)

```
length (which (setas$gill_color=='red' & setas$cap_color=='cinnamon'))
```

```
[1] 1
```

[Hide](#)

```
length (which (setas$gill_color=='green' & setas$cap_color=='green'))
```

```
[1] 1
```

Hide

```
length (which (setas$gill_color=='yellow' & setas$cap_color=='purple'))
```

```
[1] 1
```

No podemos considerar outlier a aquella combinación infrecuente de todos los valores que toman las variables ya que al tener tantas variables y valores posibles de las mismas, solo hay una combinación que se repite.

Se procede a eliminar a todos los outliers.

Hide

```
setas_clean <- setas[which (!((setas$gill_color=='yellow' & setas$cap_color=='purple') | (setas$gill_color=='green' & setas$cap_color=='green') | (setas$gill_color=='red' & setas$cap_color=='cinamon') | (setas$gill_color=='red' & setas$cap_color=='green') | (setas$ring_number=='none'))),]
```

Finalmente exportamos el nuevo dataset para que pueda ser utilizado en la fase de modelado.

Hide

```
write.csv(setas_clean, file = "../inputs/mushroomClean.csv", row.names=FALSE)
```

Modelado de los datos y evaluación

En esta fase trataremos de extraer de los datos un modelo que sea capaz de realizar predicciones adecuadas sobre la comestibilidad de nuevas setas. Para ello utilizaremos varios algoritmos con el fin de quedarnos con el que muestre un rendimiento mayor. Los algoritmos escogidos a analizar y sus pros y contras son los siguientes:

- Naive Bayes: algoritmo que basa su decisión en las probabilidades condicionadas. Entre sus ventajas destacan su sencillez, su bajo consumo de recursos y alta velocidad. Estas dos últimas características lo hacen especialmente adecuado para nuestra situación ya que queremos una aplicación que responda con rapidez. Sin embargo tiene como inconveniente su incapacidad para considerar relaciones entre variables ya que asume su independencia. Esto podría traducirse en un mal rendimiento.
- Support Vector Machines: algoritmo clasificador basado en la creación de un hiperplano que separe las distintas muestras representadas en un espacio de tantas dimensiones como variables tiene el dataset. Gozan de gran precisión y evitan el overfitting pero puede ser costoso seleccionar los parámetros óptimos. Sin embargo, estos parámetros solo van a ser seleccionados una vez y no cada vez que se compruebe una nueva seta, por lo que puede ser un candidato.
- Random forest: consiste en el uso de árboles de decisión de tal forma que cada uno depende de valores de un vector aleatorio probado independientemente y con la misma distribución para cada uno de ellos. Nos proporciona información sobre cuáles son las variables que más afectan a la clasificación y es muy escalable, sin embargo es lento en algunos casos.

Tras obtener cada modelo de datos también lo evaluaremos en cada apartado para realizar finalmente una comparación en el apartado final de esta sección.

[Hide](#)

```
trainIndex<-createDataPartition(setas_clean$edibility, p=.8, list=FALSE)
train_set<-setas_clean[trainIndex,]
test_set<-setas_clean[-trainIndex,]
kfold_indexes = kfold(setas_clean,10);
```

Naive Bayes

El primer algoritmo a probar será el Naive Bayes con el que se ha obtenido una precisión del 84,2%. En la curva ROC vemos que, aunque no es mala, empieza a irse hacia la derecha ligeramente pronto. En cuanto a tiempo de ejecución, vemos que ha tardado 0.15 segundos en entrenarse y 1.25 segundos en dar la respuesta en el test. Como el algoritmo se ejecuta con rapidez, nos hemos permitido realizar una validación k-fold con k=10 obteniendo una precisión parecida del 83,7%. Vemos que el valor "precision" es algo mayor que el de accuracy, al contrario con lo que ocurre con el de recall.

[Hide](#)

```
t <- proc.time()
model_nbayes<-naiveBayes(edibility~., data=train_set)
proc.time()-t
```

user	system	elapsed
0.09	0.00	0.13

[Hide](#)

```
t <- proc.time()
prediction<-predict(model_nbayes, newdata=test_set)
proc.time()-t
```

user	system	elapsed
1.49	0.03	1.51

[Hide](#)

```
confusionMatrix(data=prediction, reference=test_set$edibility, positive="edible")
```

Confusion Matrix and Statistics

	Reference	
Prediction	edible	poisonous
edible	719	146
poisonous	121	634

Accuracy : 0.8352

95% CI : (0.8162, 0.8529)

No Information Rate : 0.5185

P-Value [Acc > NIR] : <2e-16

Kappa : 0.6695

McNemar's Test P-Value : 0.1419

Sensitivity : 0.8560

Specificity : 0.8128

Pos Pred Value : 0.8312

Neg Pred Value : 0.8397

Prevalence : 0.5185

Detection Rate : 0.4438

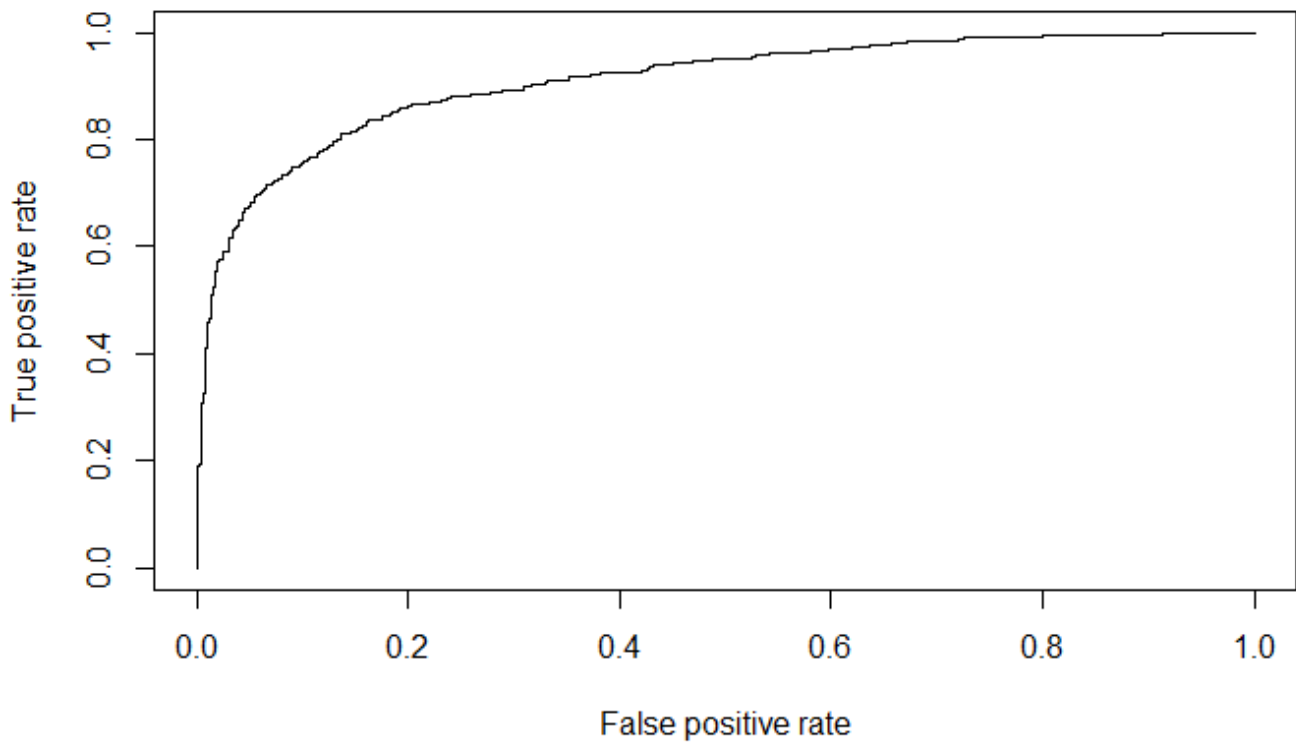
Detection Prevalence : 0.5340

Balanced Accuracy : 0.8344

'Positive' Class : edible

[Hide](#)

```
prediction2<-predict(model_nbayes, newdata=test_set, type="raw")
pred<-prediction(prediction2[,2], test_set$edibility)
perf<-performance(pred,"tpr","fpr")
plot(perf)
```



Hide

```

accuracy=0
recall=0
precision=0
sensitivity=0
specificity=0
for (i in 1:10){
  train_set2= setas_clean [kfold_indexes!=i,]
  test_set2= setas_clean [kfold_indexes==i,]
  model_nbayes2<-naiveBayes(edibility~., data=train_set2, laplace=1)
  prediction3<-predict(model_nbayes2, newdata=test_set2)
  accuracy= accuracy + Accuracy(prediction3, test_set2$edibility)
  recall= recall + Recall(prediction3, test_set2$edibility)
  precision= precision + Precision(prediction3, test_set2$edibility)
  sensitivity= sensitivity + Sensitivity(prediction3, test_set2$edibility)
  specificity= specificity + Specificity(prediction3, test_set2$edibility)
}
print("---10-FOLD METRICS---")

```

```
[1] "---10-FOLD METRICS---
```

Hide

```
cat("Accuracy: ", accuracy/10)
```

```
Accuracy:  0.836828
```

Hide

```
cat("\nPrecision: ", precision/10)
```

```
Precision: 0.8709822
```

Hide

```
cat("\nRecall: ", recall/10)
```

```
Recall: 0.8243937
```

Hide

```
cat("\nSpecificity: ", specificity/10)
```

```
Specificity: 0.8518822
```

Support Vector Machine

El segundo algoritmo que provaremos es el de las máquinas vectoriales de soporte. A este han de introducirse al menos tres parámetros: el tipo de kernel, el coste y gamma. Para encontrarlo hemos hecho uso de tune que nos proporciona los valores óptimos de cada parámetro haciendo una validación k-fold con k=10. Ha obtenido que el kernel ha de ser radial, el coste=100 y gamma=0.01. En lograr este “afinamiento” de los parámetros ha tardado 112,69 segundos.

Hide

```
t <- proc.time()
tuned = tune.svm(edibility~., data = train_set, gamma = 10^-2, cost = 10^2, tunecontrol=tune.control(cross=10))
print(tuned$best.model)
```

Call:

```
best.svm(x = edibility ~ ., data = train_set, gamma = 10^-2, cost = 10^2, tunecontrol = tune.control(cross = 10))
```

Parameters:

```
SVM-Type: C-classification
SVM-Kernel: radial
cost: 100
gamma: 0.01
```

Number of Support Vectors: 1955

Hide

```
proc.time() -t
```

```
user system elapsed
129.21 1.11 135.00
```

`` Tras ejecutar el algoritmo obtenemos una "accuracy" del 90,7%, una "precision" del 92,2 %, un "recall" del 90,1% y una "specificity" del 91,4%. Ha tardado 35 segundos en entrenarse y 0.75 en dar la respuesta sobre el conjunto de test. Vemos que la curva ROC es mejor que en el caso anterior.

Hide

```
t <- proc.time()
model_svm <- svm(edibility~. , data=train_set, kernel="radial", cost = 100, gamma = 0.01, probability = TRUE)
proc.time()-t
```

```
user  system elapsed
36.89   0.36   37.87
```

Hide

```
t <- proc.time()
test_svm <- predict(model_svm, newdata = test_set)
proc.time()-t
```

```
user  system elapsed
0.77   0.00   0.80
```

Hide

```
accuracy=0
recall=0
precision=0
sensitivity=0
specificity=0

table(test_svm, test_set$edibility)
```

```
test_svm    edible poisonous
edible      763          83
poisonous   77          697
```

Hide

```
accuracy= accuracy + Accuracy(test_svm, test_set$edibility)
recall= recall + Recall(test_svm, test_set$edibility)
precision= precision + Precision(test_svm, test_set$edibility)
sensitivity= sensitivity + Sensitivity(test_svm, test_set$edibility)
specificity= specificity + Specificity(test_svm, test_set$edibility)

cat("Accuracy: ", accuracy)
```

```
Accuracy:  0.9012346
```

Hide

```
cat("\nPrecision: ", precision)
```

```
Precision: 0.9083333
```

[Hide](#)

```
cat("\nRecall: ", recall)
```

```
Recall: 0.9018913
```

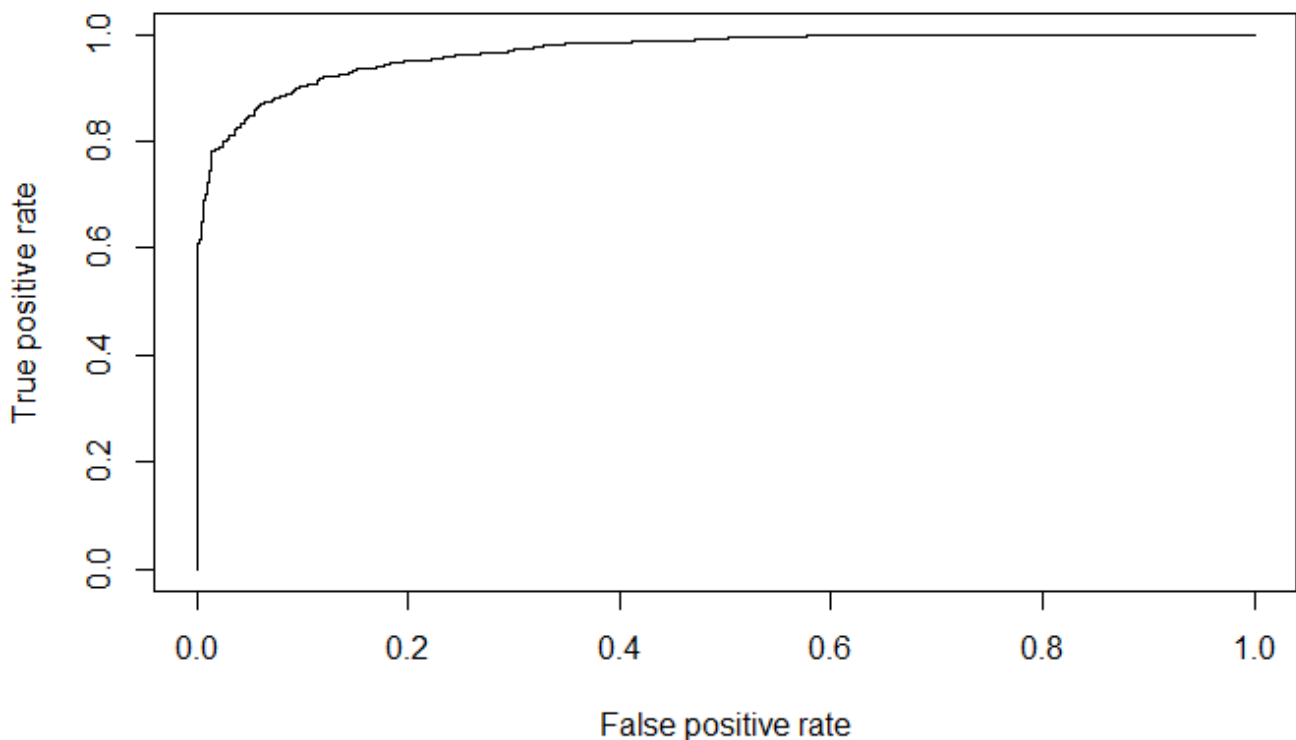
[Hide](#)

```
cat("\nSpecificity: ", specificity)
```

```
Specificity: 0.9005168
```

[Hide](#)

```
pred.output<-predict(model_svm, newdata=test_set, probability = TRUE, type="raw")
prob <- attr(pred.output, "probabilities")[,2]
pred<-prediction(prob, test_set$edibility)
perf<-performance(pred,"tpr","fpr")
plot(perf)
```



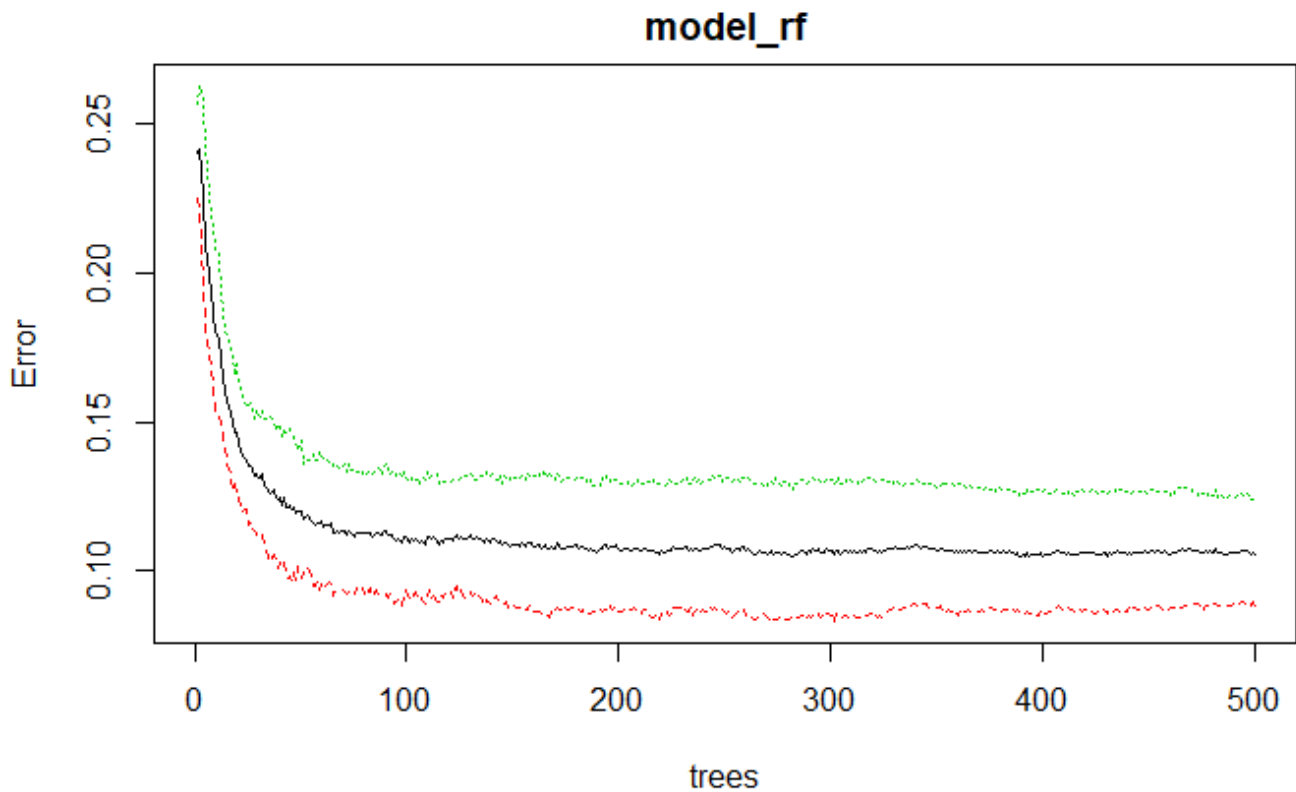
Random forest

A la hora de ejecutar el algoritmo Random forest es necesario determinar el parámetro número de árboles. Como se puede ver en la gráfica inferior a partir de (aproximadamente) 80 árboles, el error se reduce mucho más despacio en relación a la cantidad de árboles, cantidad que de ser excesiva podría ocupar mucho tiempo

de procesamiento.

[Hide](#)

```
model_rf <- randomForest(edibility ~ ., ntree = 500, data = train_set)
plot(model_rf)
```

[Hide](#)

```
t <- proc.time()
model_rf=randomForest(edibility ~ ., ntree = 80, data = train_set)
proc.time()-t
```

user	system	elapsed
3.25	0.03	3.36

[Hide](#)

```
t <- proc.time()
test_rf <- predict(model_rf, newdata = test_set)
proc.time()-t
```

user	system	elapsed
0.05	0.00	0.12

[Hide](#)

```
accuracy=0
recall=0
precision=0
sensitivity=0
specificity=0
```

```
table(test_rf, test_set$edibility)
```

```
test_rf      edible poisonous
edible       753         103
poisonous     87         677
```

Hide

```
accuracy= accuracy + Accuracy(test_rf, test_set$edibility)
recall= recall + Recall(test_rf, test_set$edibility)
precision= precision + Precision(test_rf, test_set$edibility)
sensitivity= sensitivity + Sensitivity(test_rf, test_set$edibility)
specificity= specificity + Specificity(test_rf, test_set$edibility)
```

```
cat("Accuracy: ", accuracy)
```

```
Accuracy:  0.882716
```

Hide

```
cat("\nPrecision: ", precision)
```

```
Precision:  0.8964286
```

Hide

```
cat("\nRecall: ", recall)
```

```
Recall:  0.8796729
```

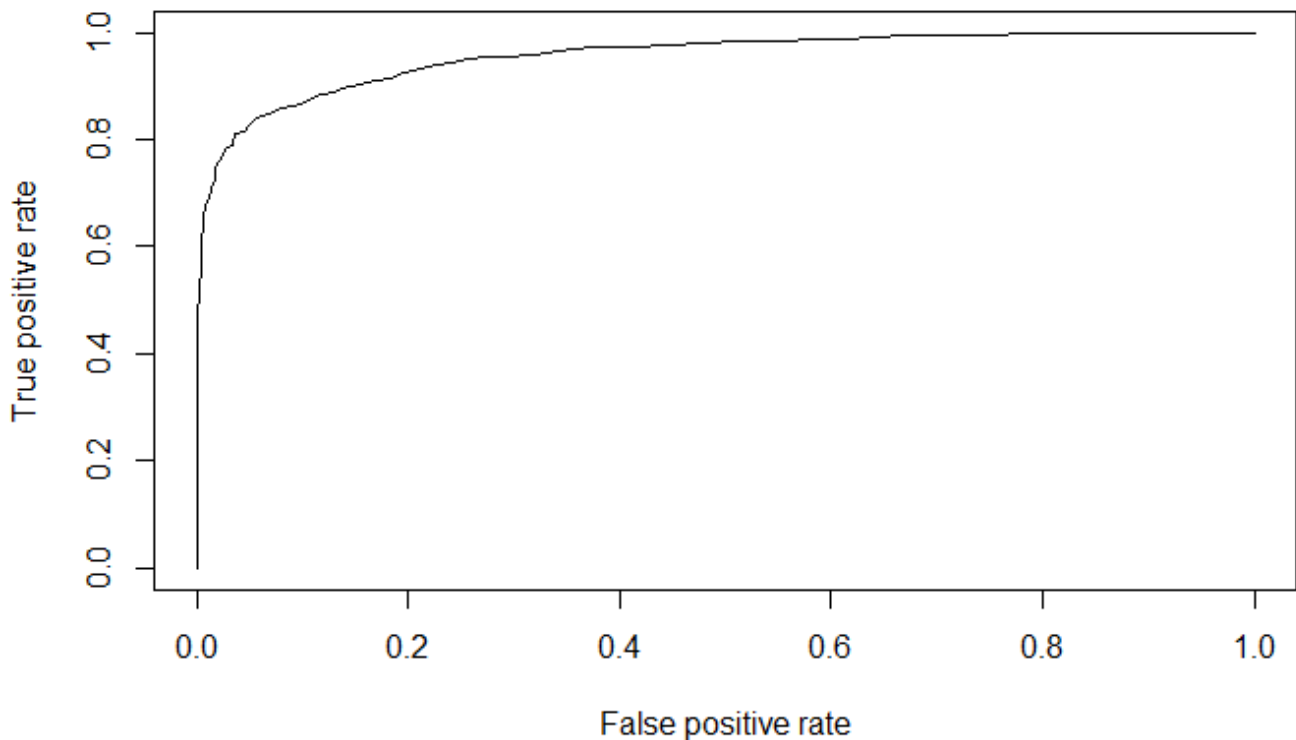
Hide

```
cat("\nSpecificity: ", specificity)
```

```
Specificity:  0.8861257
```

Hide

```
pred.output<-predict(model_rf, newdata=test_set, probability = TRUE, type="prob")
pred<-prediction(pred.output[,2], test_set$edibility)
perf<-performance(pred,"tpr","fpr")
plot(perf)
```



Evaluación

Los indicadores obtenidos del rendimiento de cada uno de los algoritmos ha sido el siguiente:

TÉCNICA	TIEMPO DE ENTRENAMIENTO (s)	TIEMPO DE EJECUCIÓN (s)	ACCURACY (%)	PRECISSION (%)	RECALL (%)	SPECIFICITY (%)
Naive Bayes	0,15	1,45	83,6	86,8	82,5	85,0
Support Vector Machines	35	0,75	90,7	92,3	90,1	91,4
Random Forest	3,15	0,06	89,1	91,1	88,2	90,1

Como podemos ver, el algoritmo Naive Bayes no solo es el que presenta un peor tiempo de ejecución sino que también es el que peores estadísticas de rendimiento tiene, por lo que es descartado.

El algoritmo basado en Support Vector Machines, aunque es el que mejor estadísticas presenta, tarda más de 10 veces más en ejecutarse que el Random Forest.

Por este último motivo, y porque presenta una accuracy y precision (este último parámetro es importante ya que recomendar una seta como comestible cuando es venenosa sería peligroso) bastante altas y cercanas al SVM (las curvas ROC son también bastante semejantes), el algoritmo escogido será Random Forest.

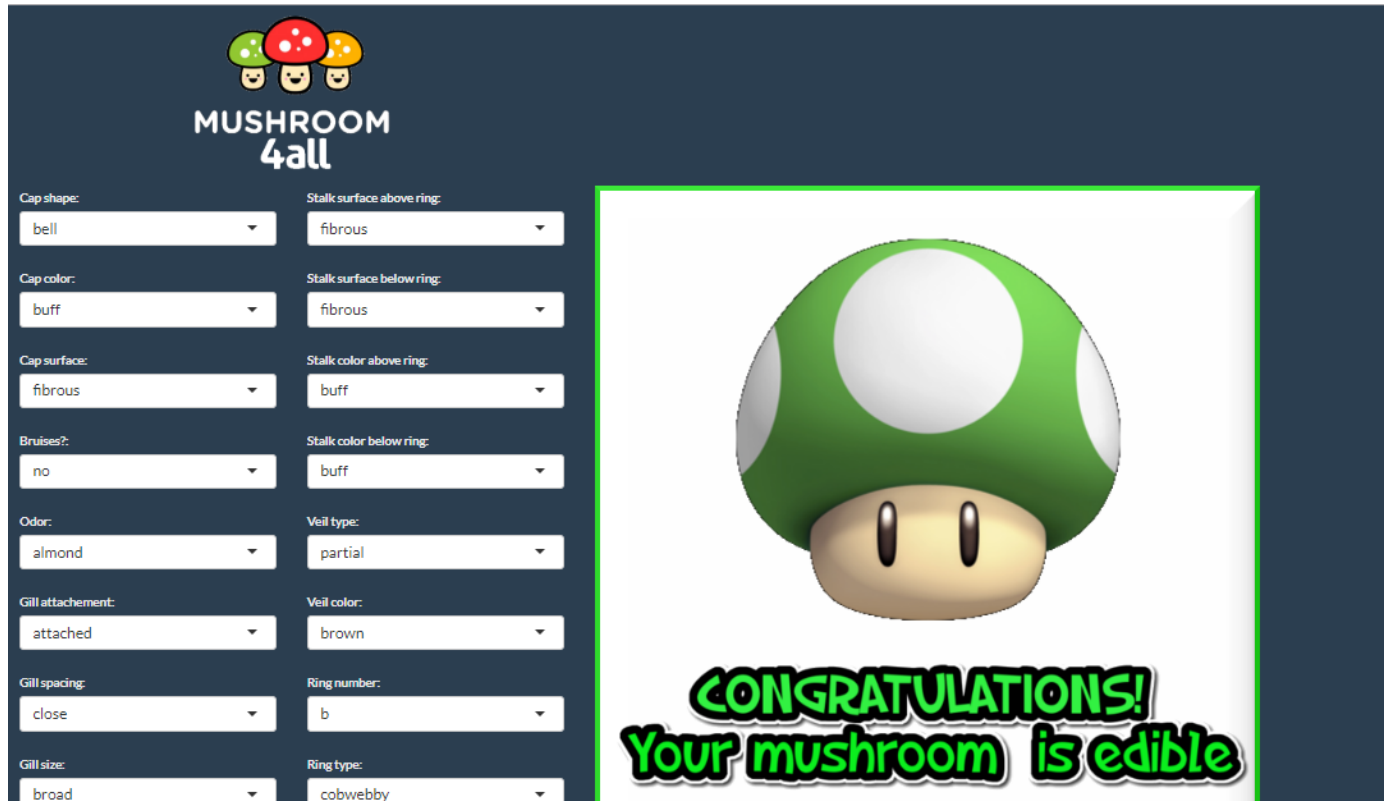
Despliegue

Para lograr hacer accesible nuestra herramienta de evaluación de setas se ha creado una API a la que poder consultar su comestibilidad introduciendo sus características. Para crear la herramienta se ha utilizado R Shiny, un paquete que nos permite crear fácilmente aplicaciones web basadas en R. Esta aplicación se puede

encontrar en el directorio code/API. Los datos se introducen utilizando la herramienta selectInput() de R shiny. Esta aplicación podría subirse a un servidor facilmente y hacer que fuese accesible por todos los usuarios de la web.

Utiliza un modelo previamente obtenido con el siguiente código incluido en el script code/modelCreation.R. En él se hace uso del algoritmo random forest con 80 árboles antes mencionado.

Para la ejecución de la API es necesario abrir el archivo api.R (necesitamos tener instalado R al ejecutarla en local) y hacer click en Run. Entonces nos aparecerá una pantalla en la que podremos seleccionar las características de nuestra seta. Cuando las hayamos introducido todas, pulsamos el botón 'Classify' y obtenemos en el lado derecho de la pantalla la clasificación de nuestra seta.



Se puede visualizar un video demostrativo del uso de la app en el archivo /docs/media/videos/tutorial.mp4.