semana_2

Luis Ambrocio

17/8/2021

Contents

graficacion en lattice **1**
comportamiento de lattice 3
Resumen 6

ggplot2 **6**
Grammer of Graphics 6
Componentes básicos de un gráfico ggplot2 7
Exjemplos 7
Resumen y recursos 34

graficacion en lattice

Funciones de lattice

- `xyplot`: esta es la función principal para crear diagramas de dispersión
- `bwplot`: diagramas de caja y bigotes (" diagramas de caja ")
- `histogram`: histogramas
- `stripplot`: como un diagrama de caja pero con puntos reales
- `dotplot`: traza puntos en "cuerdas de violín"
- `splom`: matriz de diagramas de dispersión; como "pares" en el sistema de trazado base
- `levelplot`,`contourplot`: para trazar datos de "imagen

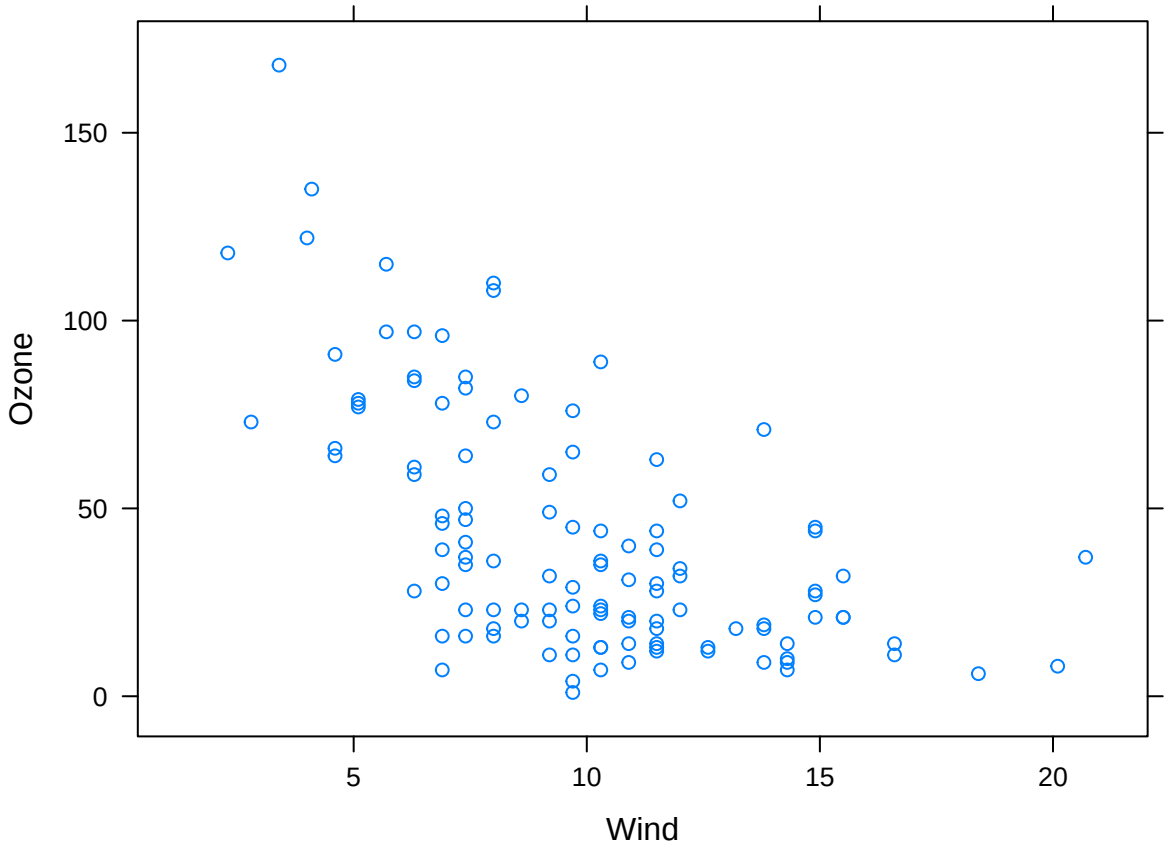
Las funciones de lattice generalmente toman una fórmula para su primer argumento, generalmente de la forma

```
xyplot(y ~ x | f * g, data)
```

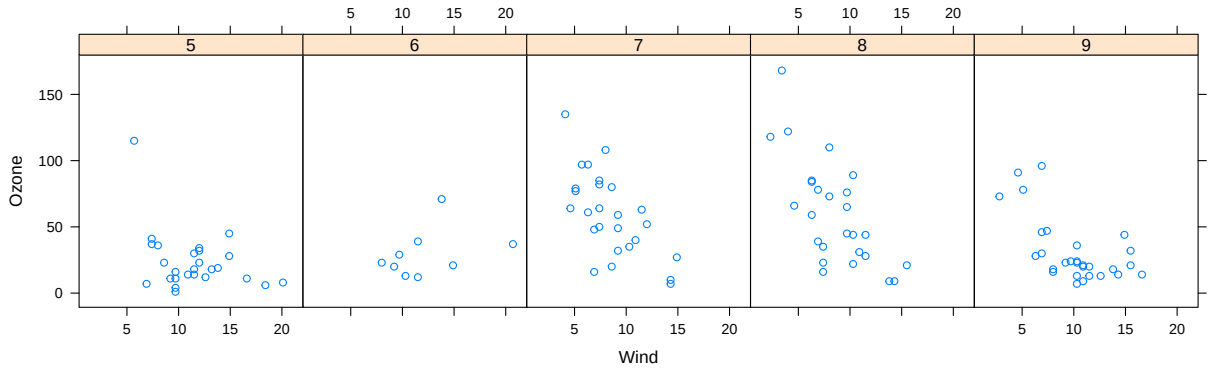
- Usamos la *notación de fórmula* aquí, de ahí el `~`.
- A la izquierda de ~ está la variable del eje y, a la derecha está la variable del eje x
- f y g son *variables condicionantes* - son opcionales
  - el * indica una interacción entre dos variables
- El segundo argumento es el marco de datos o la lista desde la que se deben buscar las variables en la fórmula.
  - Si no se pasa ningún marco de datos o lista, se utiliza el marco principal.
- Si no se pasan otros argumentos, hay valores predeterminados que se pueden utilizar.

```
library(lattice)
library(datasets)
```

```
## Simple scatterplot
xyplot(Ozone ~ Wind, data = airquality)
```




```
## Convert 'Month' to a factor variable
airquality <- transform(airquality, Month = factor(Month))
xyplot(Ozone ~ Wind | Month, data = airquality, layout = c(5, 1))
```

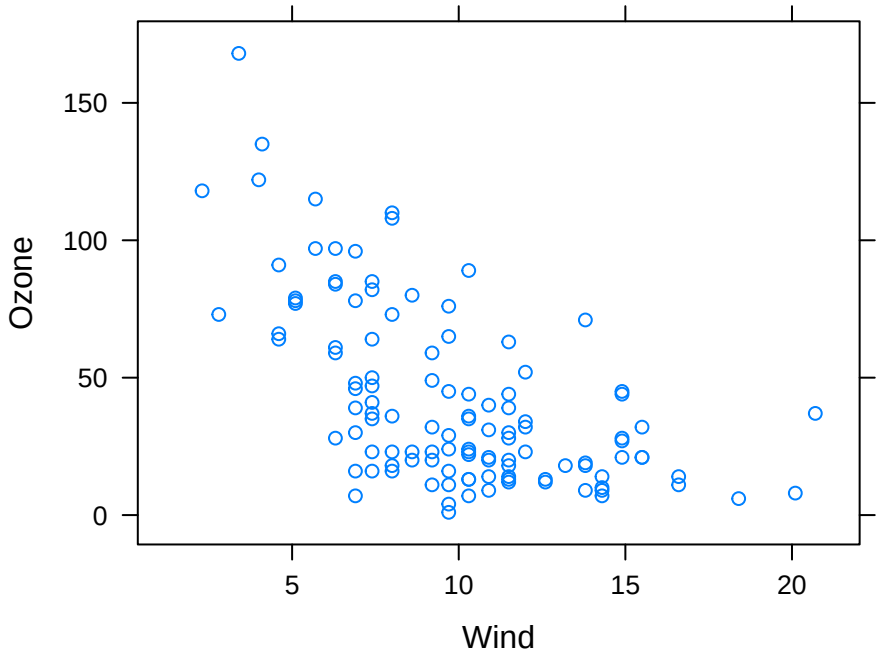


comportamiento de lattice

Las funciones de lattice se comportan de manera diferente a las funciones gráficas base de una manera crítica.

- Las funciones gráficas base trazan datos directamente en el dispositivo gráfico (pantalla, archivo PDF, etc.)
- Las funciones de gráficos de lattice devuelven un objeto de clase **trellis**
- Los métodos de impresión para las funciones de lattice realmente hacen el trabajo de trazar los datos en el dispositivo gráfico.
- Las funciones de lattice devuelven "objetos de trazado" que, en principio, pueden almacenarse (pero normalmente es mejor guardar el código y los datos).
- En la línea de comando, los objetos trellis se *imprimen automáticamente* para que parezca que la función está trazando los datos

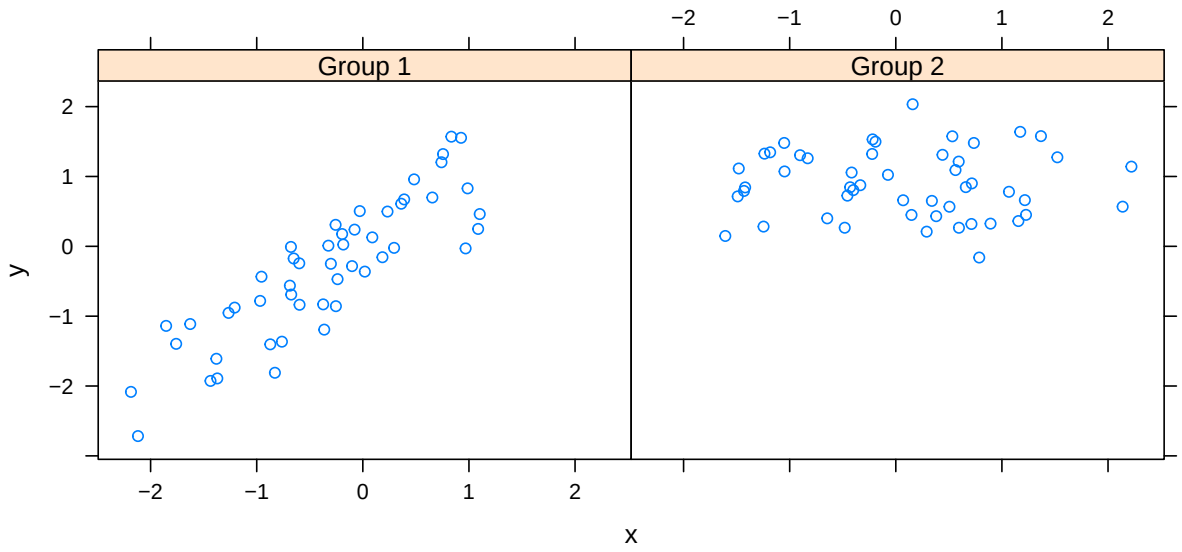
```
p <- xyplot(Ozone ~ Wind, data = airquality)  ## Nothing happens!
print(p)  ## Plot appears
```



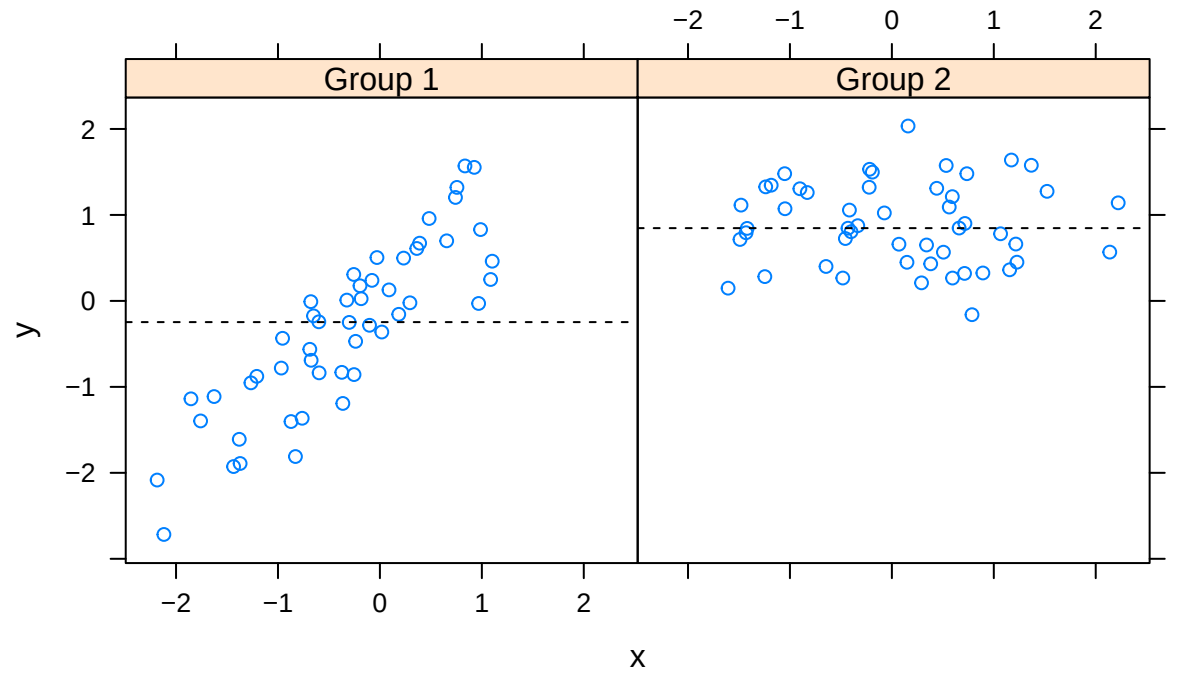
- Las funciones de lattice tienen una **función de panel** que controla lo que sucede dentro de cada panel de la trama.
- El paquete *lattice* viene con funciones de panel predeterminadas, pero puede proporcionar las suyas propias si desea personalizar lo que sucede en cada panel
- Las funciones del panel reciben las coordenadas x/y de los puntos de datos en su panel (junto con cualquier argumento opcional)

```
set.seed(10)
x <- rnorm(100)
f <- rep(0:1, each = 50)
y <- x + f - f * x+ rnorm(100, sd = 0.5)
```

```r
f <- factor(f, labels = c("Group 1", "Group 2"))
xyplot(y ~ x | f, layout = c(2, 1))  ## Plot with 2 panels
```




```r
## Custom panel function
xyplot(y ~ x | f, panel = function(x, y, ...) {
       panel.xyplot(x, y, ...)  ## First call the default panel function for 'xyplot'
       panel.abline(h = median(y), lty = 2)  ## Add a horizontal line at the median
})
```



```r
## Custom panel function
xyplot(y ~ x | f, panel = function(x, y, ...) {
               panel.xyplot(x, y, ...)  ## First call default panel function
               panel.lmline(x, y, col = 2)  ## Overlay a simple linear regression line
       })
```

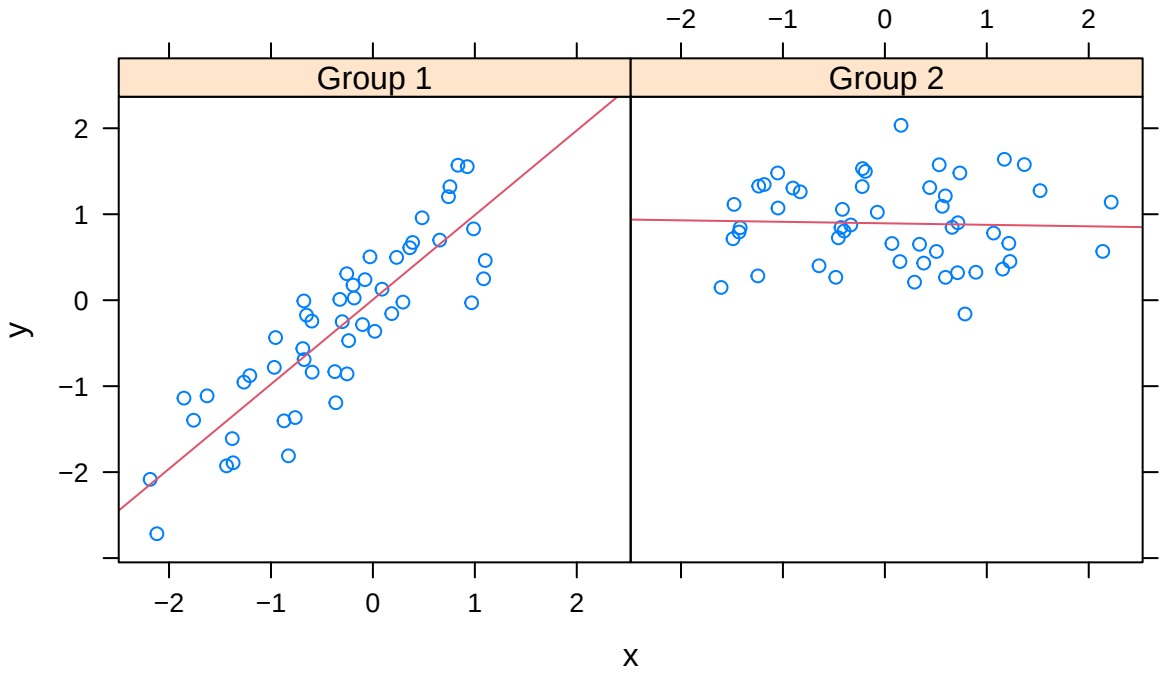
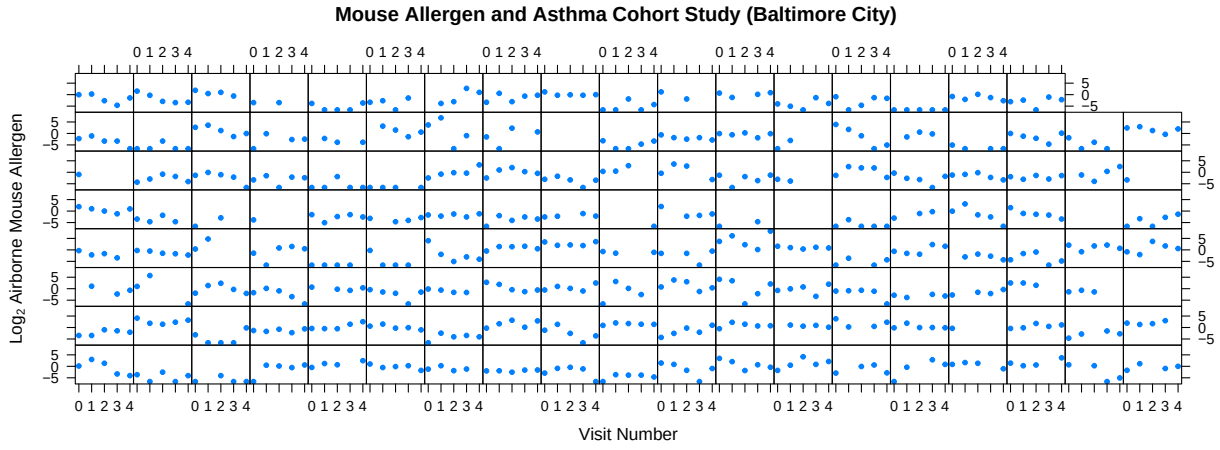


Gráfico de lattice de muchos paneles: ejemplo de MAACS

- Estudio: Estudio de cohorte de alérgenos en ratones y asma (MAACS)
- Sujetos del estudio: niños con asma que viven en la ciudad de Baltimore, muchos alérgicos al alérgeno del ratón
- Diseño: Estudio observacional, visita domiciliaria basal + cada 3 meses durante un año.
- Pregunta: ¿Cómo varía el alérgeno del ratón en el aire en interiores con el tiempo y entre sujetos?

Ahluwalia et al., *Journal of Allergy and Clinical Immunology*, 2013



Resumen

- Los diagramas de lattice se construyen con una sola llamada de función a una función de lattice central (por ejemplo, `xyplot`)
- Los aspectos como los márgenes y el espaciado se manejan automáticamente y los valores predeterminados suelen ser suficientes
- El sistema de lattice es ideal para crear parcelas de acondicionamiento en las que se examina el mismo tipo de parcela en muchas condiciones diferentes.
- Las funciones del panel se pueden especificar / personalizar para modificar lo que se traza en cada uno de los paneles de trazado.

ggplot2

¿Qué es ggplot2?

- Una implementación de *La gramática de los gráficos* por Leland Wilkinson
- Escrito por Hadley Wickham (mientras era estudiante de posgrado en Iowa State)
- Un "tercer" sistema de gráficos para R (junto con **base** y **lattice**)
- Disponible en CRAN a través de `install.packages ()`
- Sitio web: http://ggplot2.org (mejor documentación)
- La gramática de los gráficos representa una abstracción de ideas / objetos gráficos
- Piense en "verbo", "sustantivo", "adjetivo" para los gráficos
- Permite una "teoría" de gráficos sobre la que construir nuevos gráficos y objetos gráficos
- "Acorta la distancia de la mente a la página"

Grammer of Graphics

> "En resumen, la gramática nos dice que un gráfico estadístico es un **mapeo** de datos a atributos **estéticos** (color, forma, tamaño) de objetos **geométricos** (puntos, líneas, barras). El gráfico también puede contener transformaciones estadísticas de los datos y se dibuja en un sistema de coordenadas específico"

-del libro *ggplot2*

- toma lo mejor de base y lattice.
- Se ocupa automáticamente de los espacios, el texto y los títulos, pero también le permite hacer anotaciones "agregando"

- Similitud superficial con la lattice, pero generalmente más fácil / más intuitivo de usar
- El modo predeterminado le permite elegir entre muchas opciones (¡pero usted puede personalizarlo!)
- Funciona de manera muy similar a la función `plot` en el sistema de gráficos base
- Busca datos en un marco de datos, similar lattice o en el entorno principal
- Las parcelas se componen de *estética* (tamaño, forma, color) y *geoms* (puntos, líneas)
- Los factores son importantes para indicar subconjuntos de datos (si van a tener propiedades diferentes); deben estar **etiquetados**
- El `qplot ()` oculta lo que sucede debajo, lo cual está bien para la mayoría de las operaciones
- `ggplot ()` es la función principal y muy flexible para hacer cosas que `qplot ()` no puede hacer

Componentes básicos de un gráfico ggplot2

- Un *marco de datos*
- *mapeos estéticos*: cómo se asignan los datos al color, tamaño
- *geoms*: objetos geométricos como puntos, líneas, formas.
- *facets*: para gráficos condicionales.
- *stats*: transformaciones estadísticas como binning, cuantiles, suavizado.
- *escales*: qué escala usa un mapa estético (ejemplo: masculino = rojo, femenino = azul).
- *sistema coordinado*

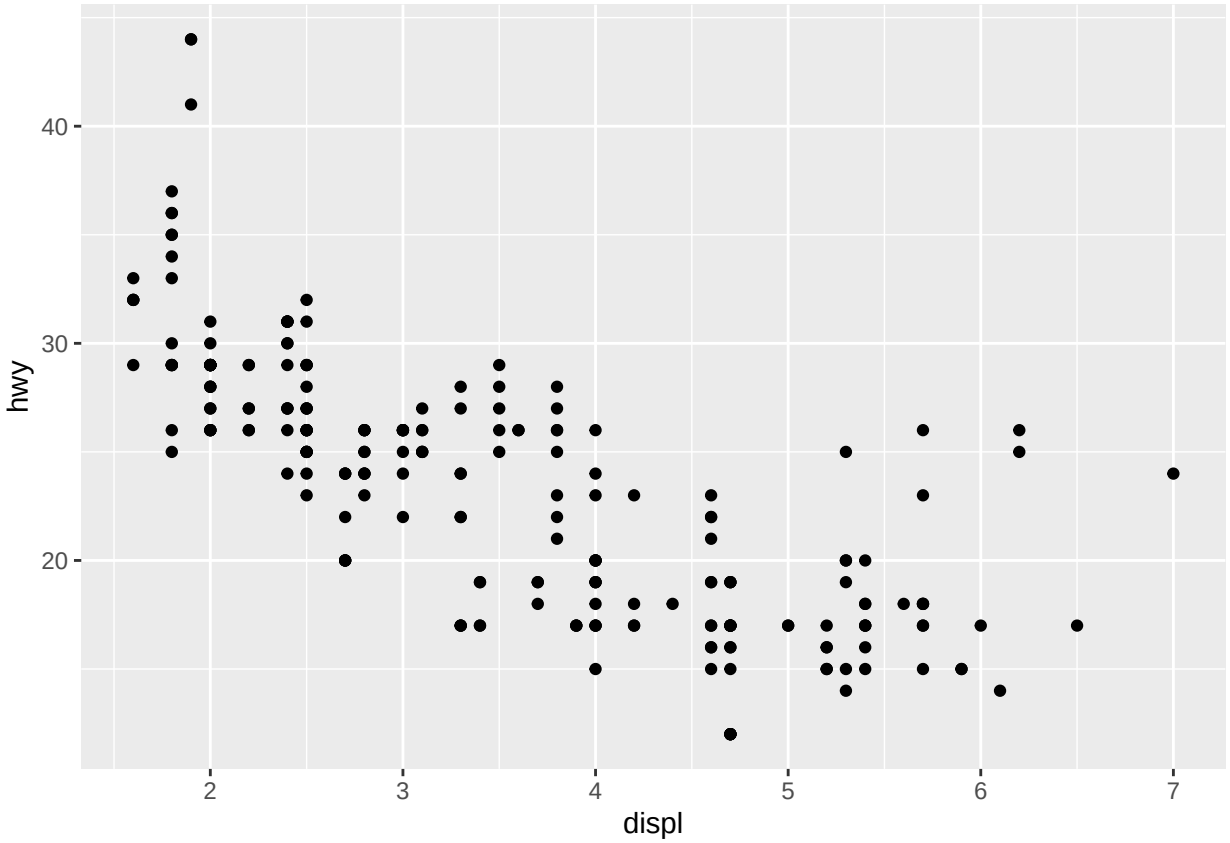
Parcelas edificables con ggplot2 - Al construir parcelas en ggplot2 (en lugar de usar qplot), el modelo de "paleta del artista" puede ser la analogía más cercana - Las parcelas se construyen en capas. - Trazar los datos - Superponer un resumen - Metadatos y anotación

Exjemplos

```
library(ggplot2)
str(mpg)
```

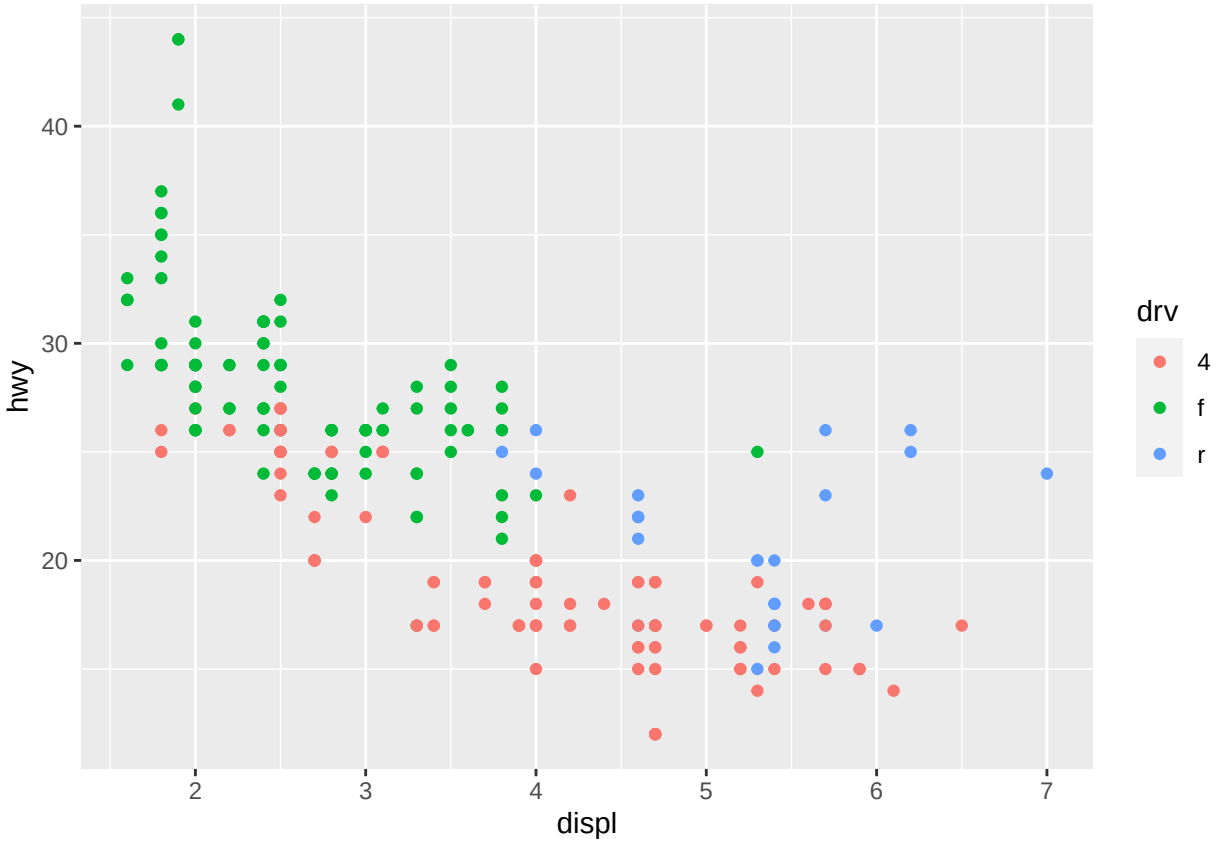
```
## tibble [234 x 11] (S3: tbl_df/tbl/data.frame)
##  $ manufacturer: chr [1:234] "audi" "audi" "audi" "audi" ...
##  $ model       : chr [1:234] "a4" "a4" "a4" "a4" ...
##  $ displ       : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cyl         : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
##  $ drv         : chr [1:234] "f" "f" "f" "f" ...
##  $ cty         : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : chr [1:234] "p" "p" "p" "p" ...
##  $ class       : chr [1:234] "compact" "compact" "compact" "compact" ...
```

```
qplot(displ, hwy, data = mpg)
```



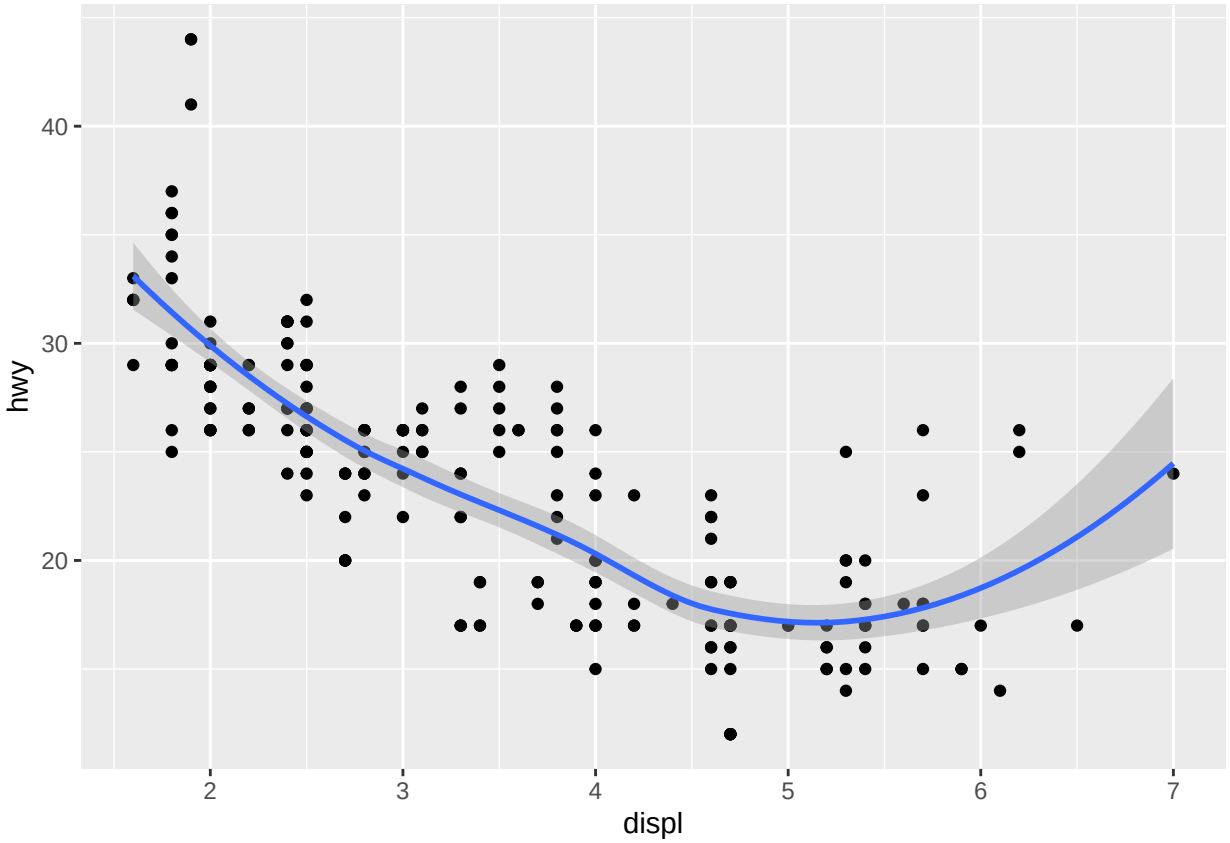
agregando una tercera variable factor

```
qplot(displ, hwy, data = mpg, color = drv)
```



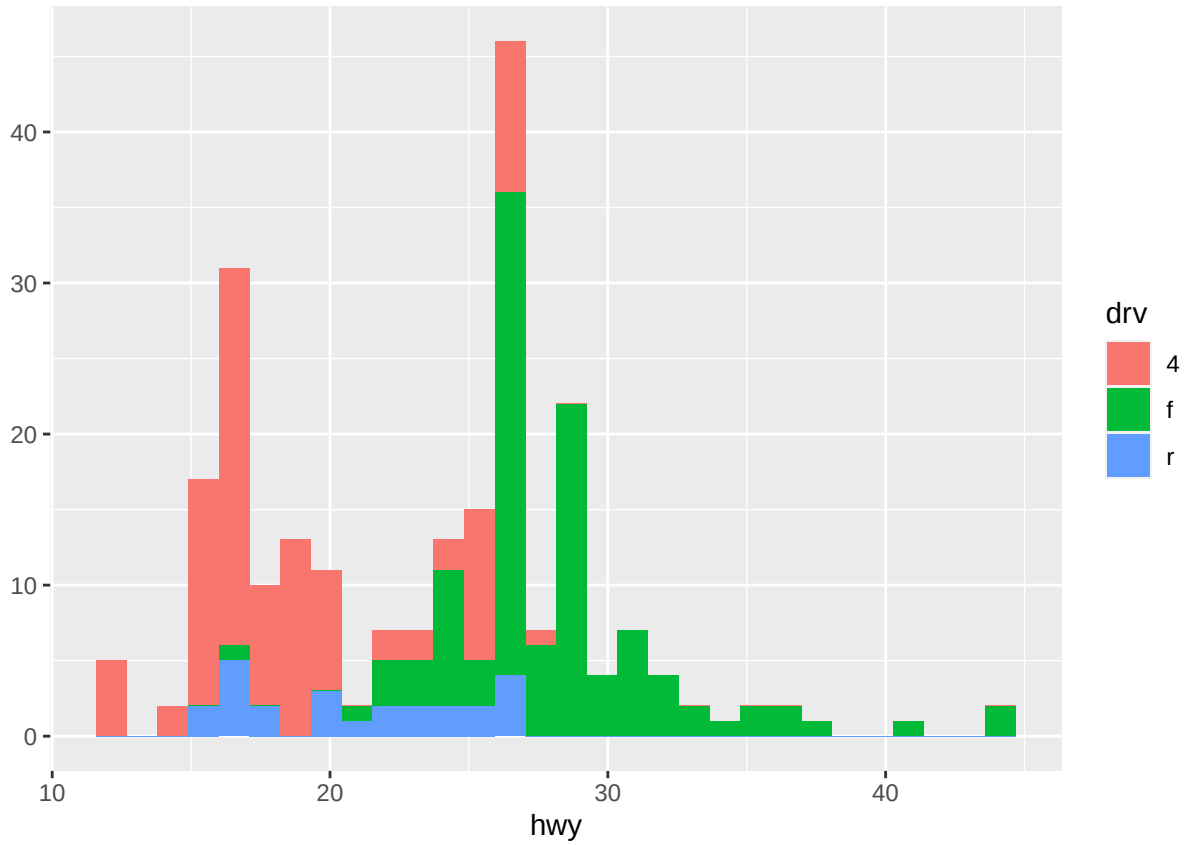
agregando “geoms”

```
qplot(displ, hwy, data = mpg, geom = c("point", "smooth"))
```



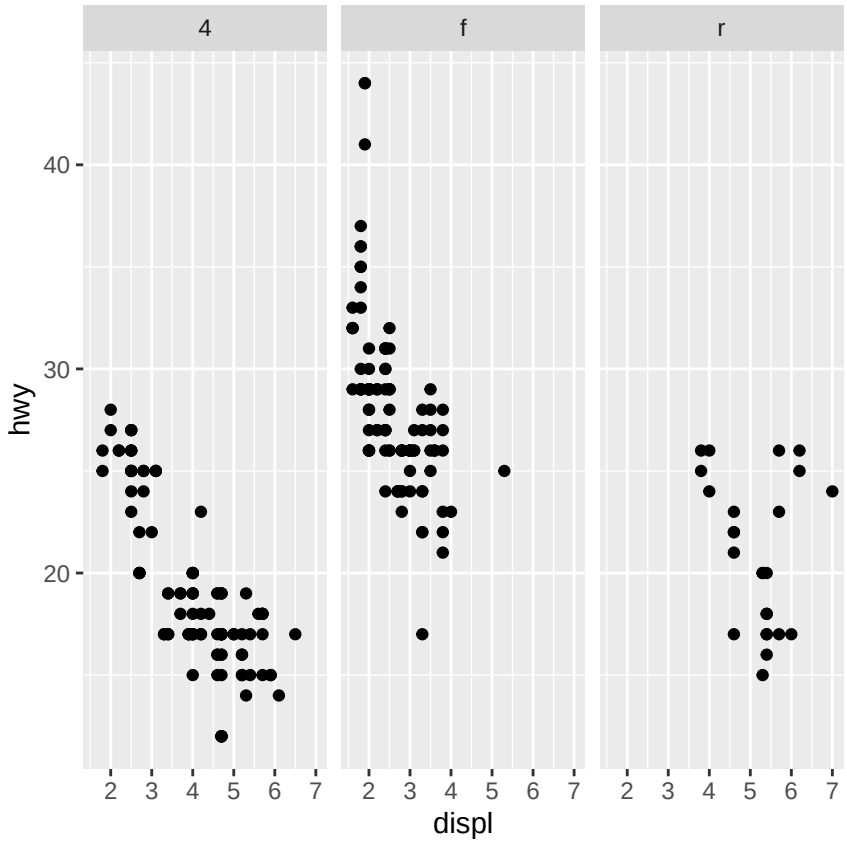
histogramas

```
qplot(hwy, data = mpg, fill = drv)
```

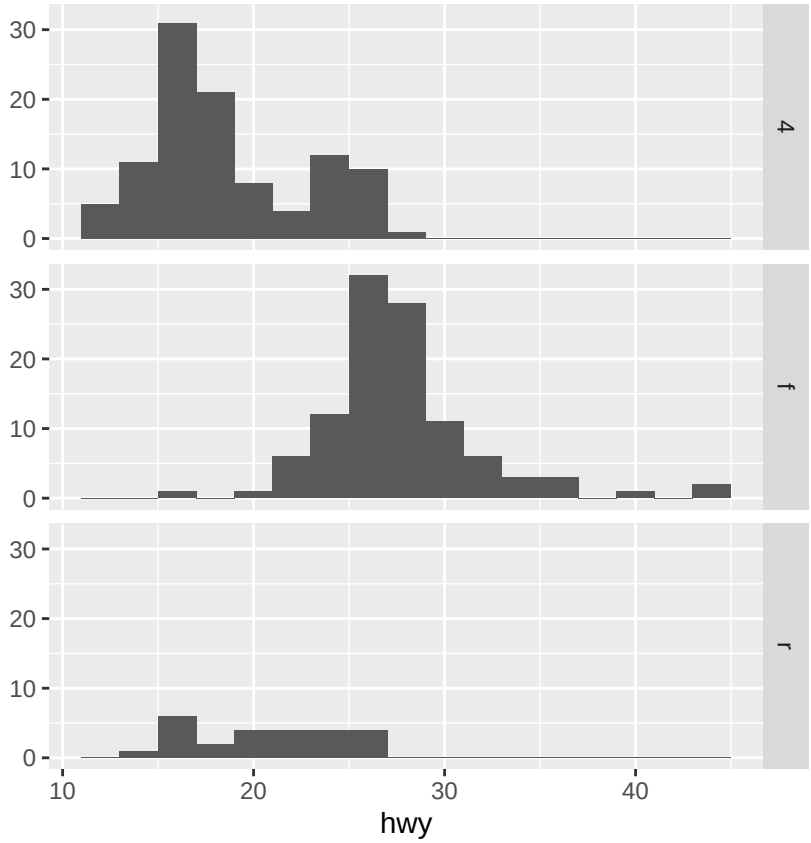


facetas

```
qplot(displ, hwy, data = mpg, facets = . ~ drv)
```




```
qplot(hwy, data = mpg, facets = drv ~ ., binwidth = 2)
```



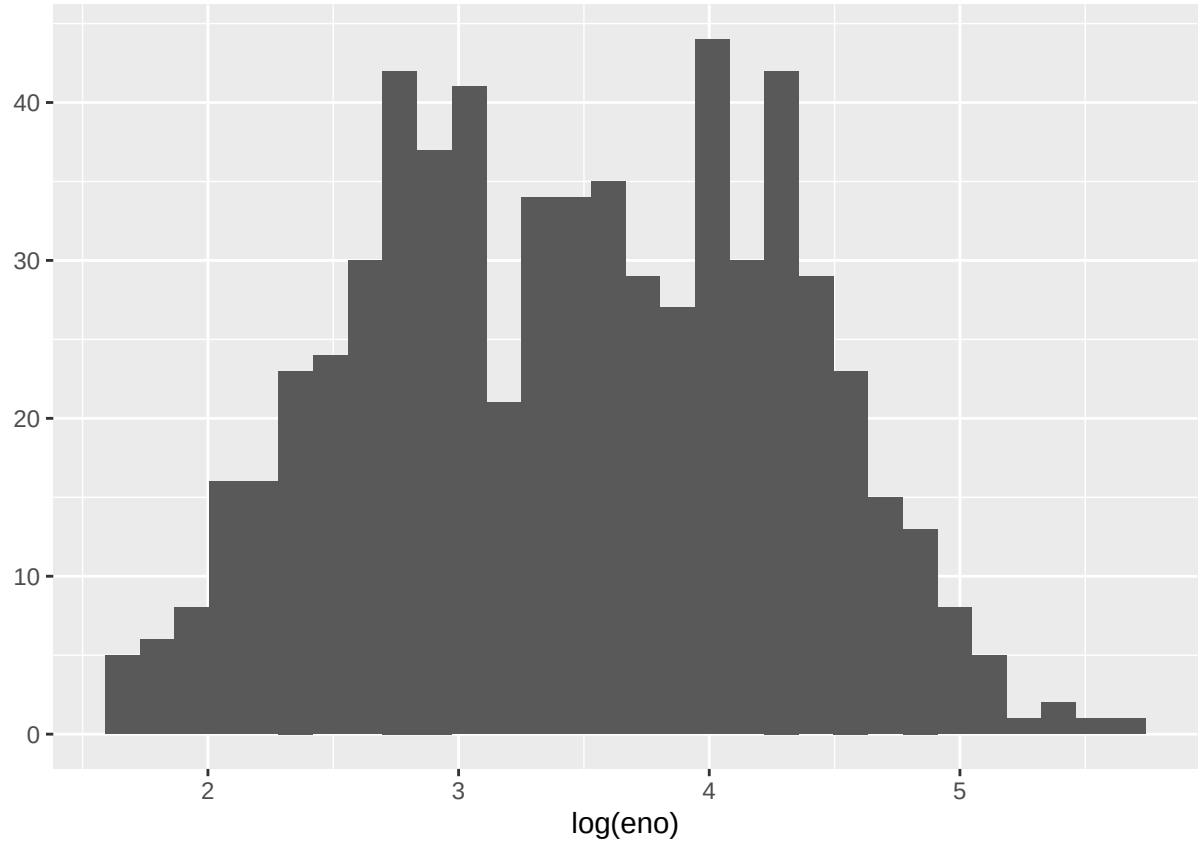
datos Cohorte MAACS

- Estudio de cohorte de alérgenos en ratones y asma
- Niños de Baltimore (de 5 a 17 años)
- Asma persistente, exacerbación en el último año
- Estudiar el ambiente interior y su relación con la morbilidad por asma.
- Publicación reciente: http://goo.gl/WqE9j8

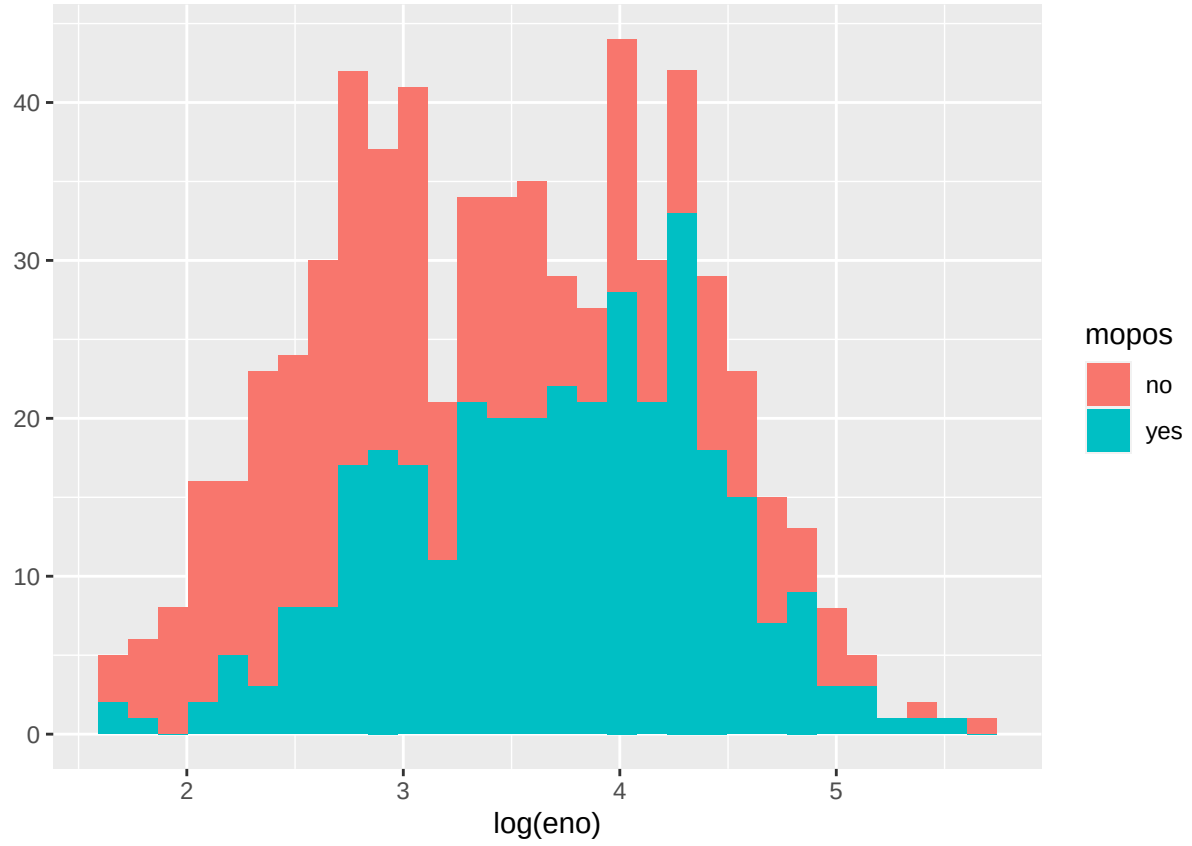
```
str(maacs)
```

```
## 'data.frame':    750 obs. of  9 variables:
##  $ id            : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ eno           : num  141 124 126 164 99 68 41 50 12 30 ...
##  $ duBedMusM     : num  2423 2793 3055 775 1634 ...
##  $ pm25          : num  15.6 34.4 39 33.2 27.1 ...
##  $ mopos         : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ logpm25       : num  1.19 1.54 1.59 1.52 1.43 ...
##  $ NocturnalSympt: int  0 0 2 2 2 2 0 1 0 0 ...
##  $ bmicat        : Factor w/ 2 levels "normal weight",..: 1 2 2 1 1 1 2 2 2 1 ...
##  $ logno2_new    : num  1.62 1.88 1.71 1.46 1.29 ...
```

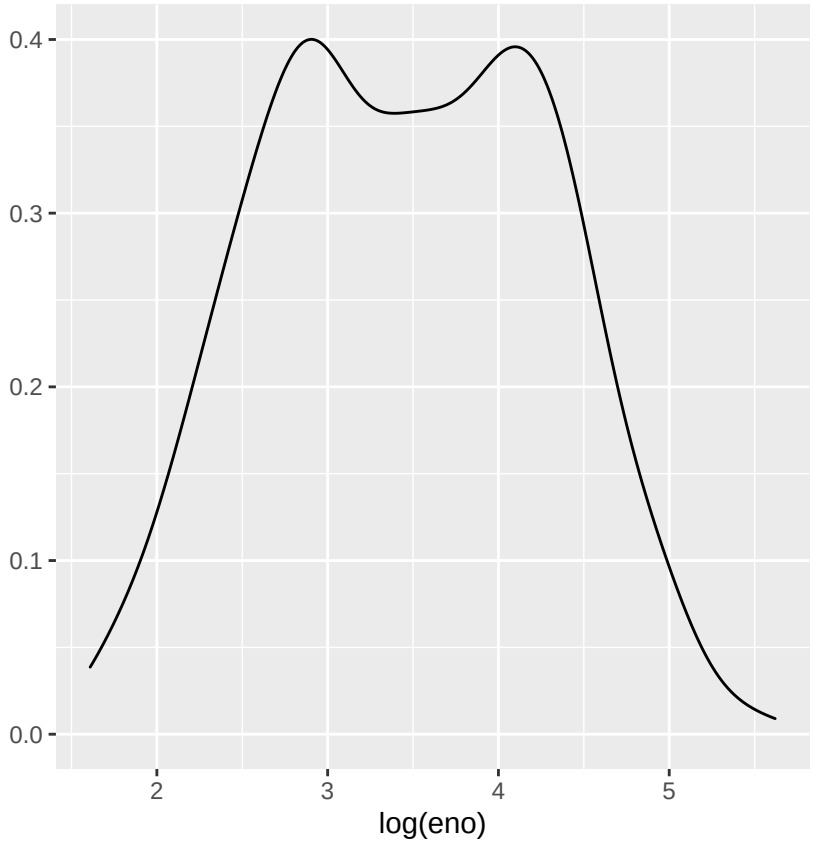
```
qplot(log(eno), data = maacs)
```



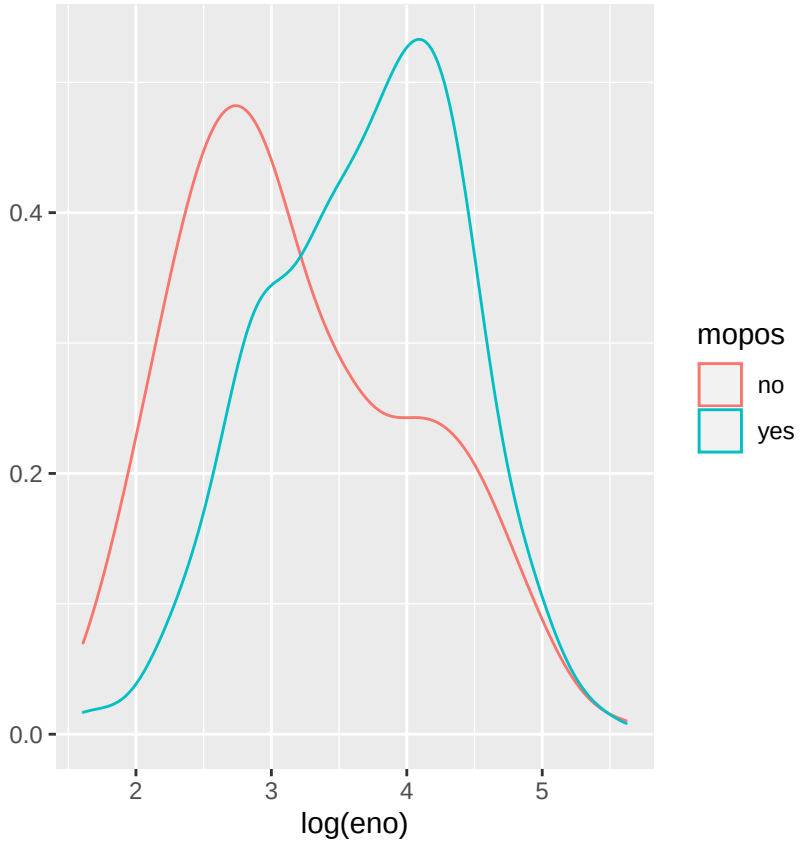

```
qplot(log(eno), data = maacs, fill = mopos)
```



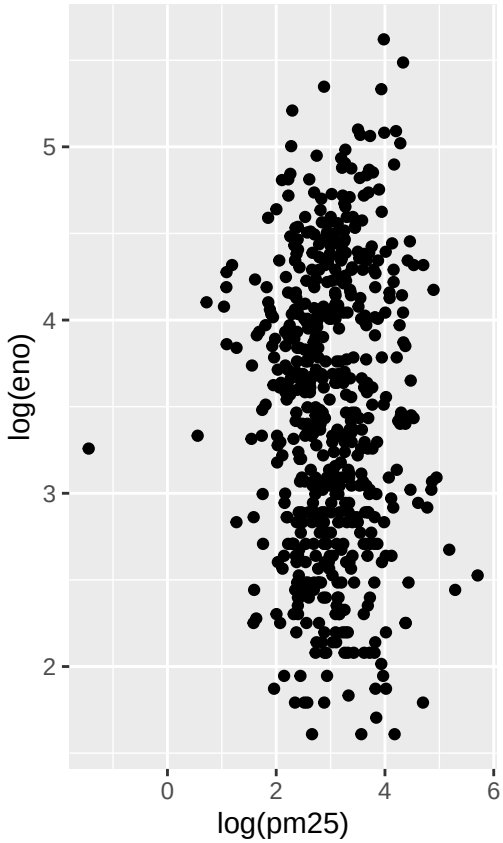

```
qplot(log(eno), data = maacs, geom = "density")
```



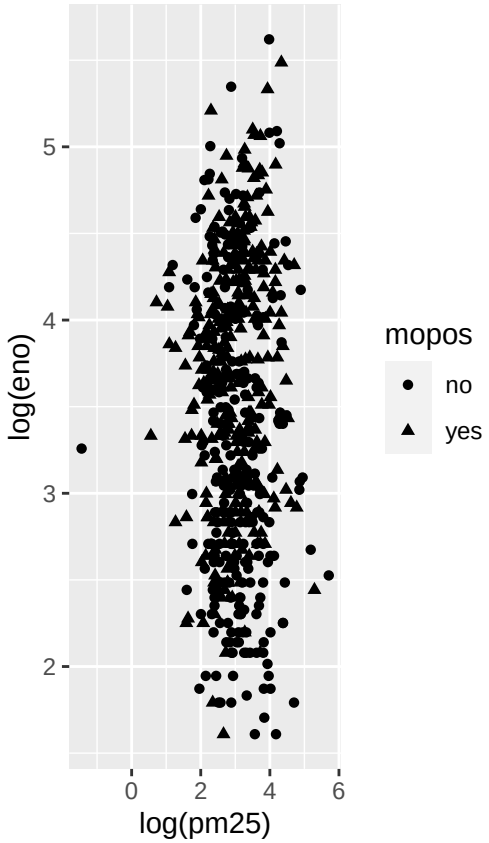

```
qplot(log(eno), data = maacs, geom = "density", color = mopos)
```



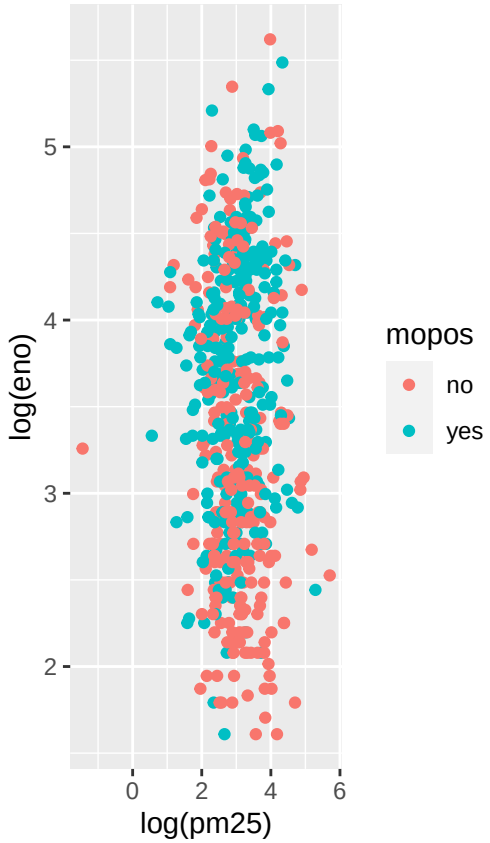

```
qplot(log(pm25), log(eno), data = maacs)
```



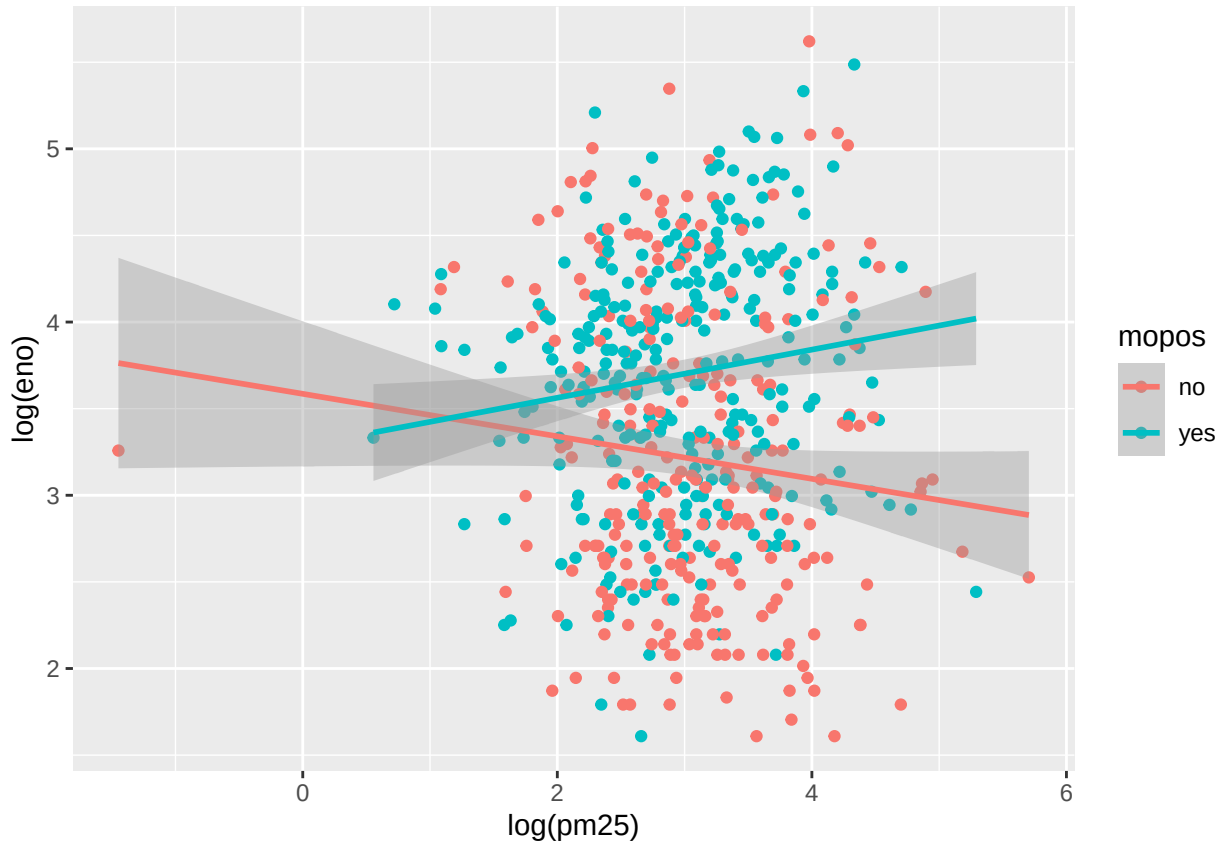

```
qplot(log(pm25), log(eno), data = maacs, shape = mopos)
```



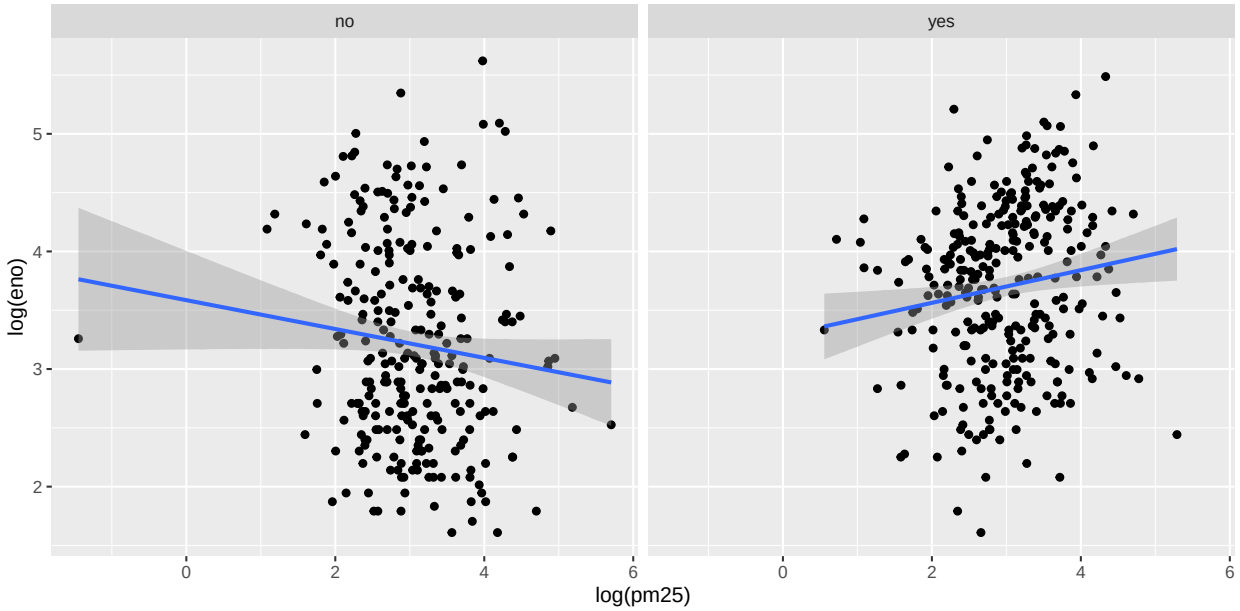

```
qplot(log(pm25), log(eno), data = maacs, color = mopos)
```




```
qplot(log(pm25), log(eno), data = maacs, color = mopos,
      geom = c("point", "smooth"), method = "lm")
```

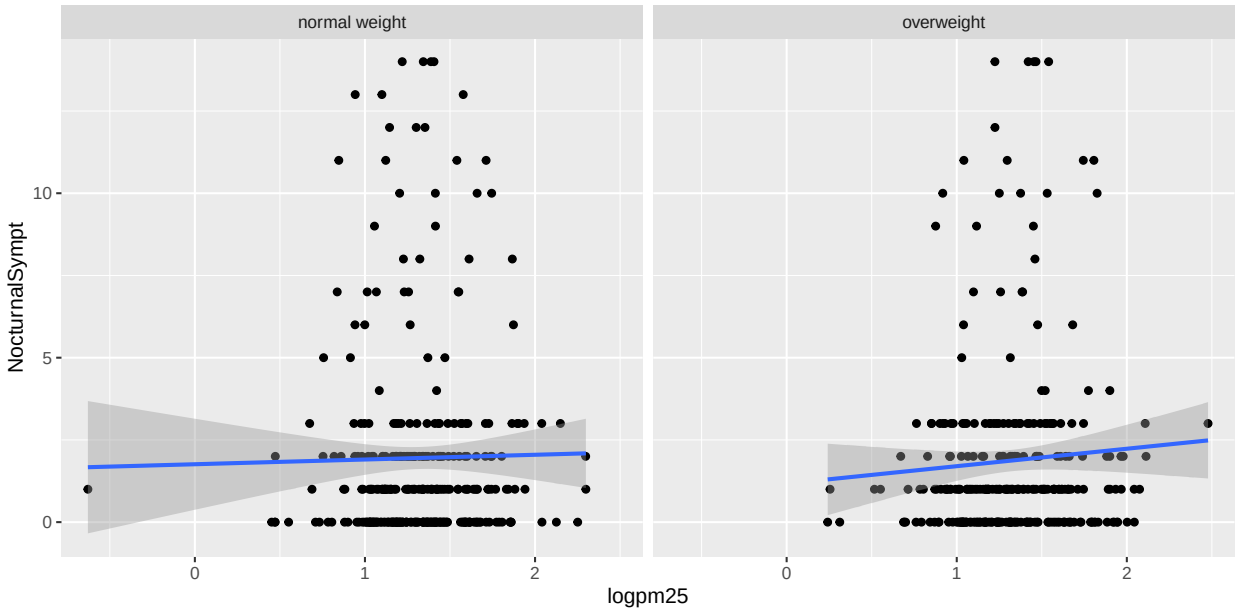



```
qplot(log(pm25), log(eno), data = maacs, geom = c("point", "smooth"),
      method = "lm", facets = . ~ mopos)
```




```
library(ggplot2)
qplot(logpm25, NocturnalSympt, data = maacs, facets = . ~ bmicat,
```

```
      geom = c("point", "smooth"), method = "lm")
```



Construyendo en capas

```
head(maacs)
```

```
##   id eno duBedMusM   pm25 mopos  logpm25 NocturnalSympt        bmicat
## 1  1 141      2423 15.560   yes 1.192010              0 normal weight
## 2  2 124      2793 34.370   yes 1.536180              0    overweight
## 3  3 126      3055 38.953   yes 1.590541              2    overweight
## 4  4 164       775 33.249   yes 1.521779              2 normal weight
## 5  5  99      1634 27.060   yes 1.432328              2 normal weight
## 6  6  68       939 18.890   yes 1.276232              2 normal weight
##   logno2_new
## 1   1.617849
## 2   1.884490
## 3   1.712953
## 4   1.458879
## 5   1.294510
## 6   1.468377
```

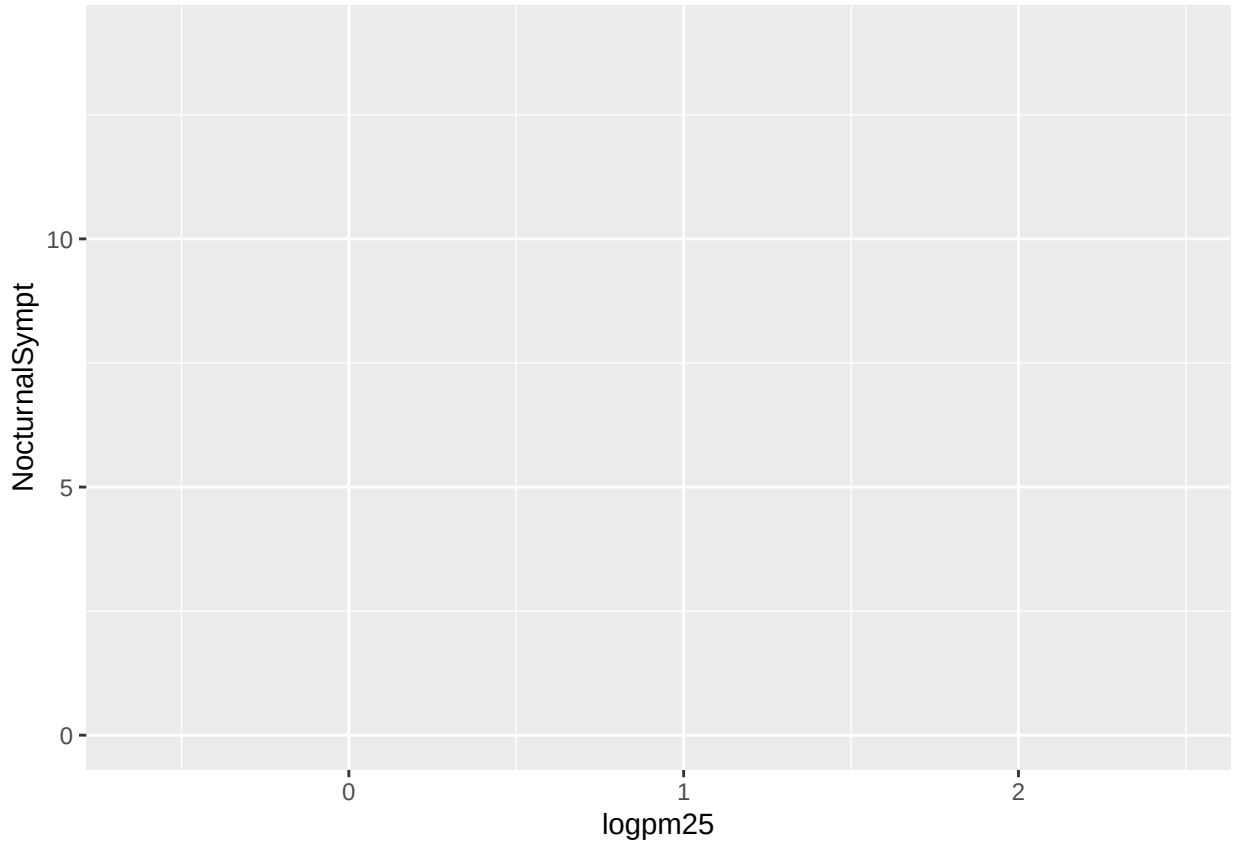
```
g <- ggplot(maacs, aes(logpm25, NocturnalSympt))
summary(g)
```

```
## data: id, eno, duBedMusM, pm25, mopos, logpm25, NocturnalSympt, bmicat,
##   logno2_new [750x9]
## mapping:  x = ~logpm25, y = ~NocturnalSympt
## faceting: <ggproto object: Class FacetNull, Facet, gg>
##     compute_layout: function
##     draw_back: function
##     draw_front: function
##     draw_labels: function
##     draw_panels: function
```

```
##     finish_data: function
##     init_scales: function
##     map_data: function
##     params: list
##     setup_data: function
##     setup_params: function
##     shrink: TRUE
##     train_scales: function
##     vars: function
##     super:  <ggproto object: Class FacetNull, Facet, gg>
```

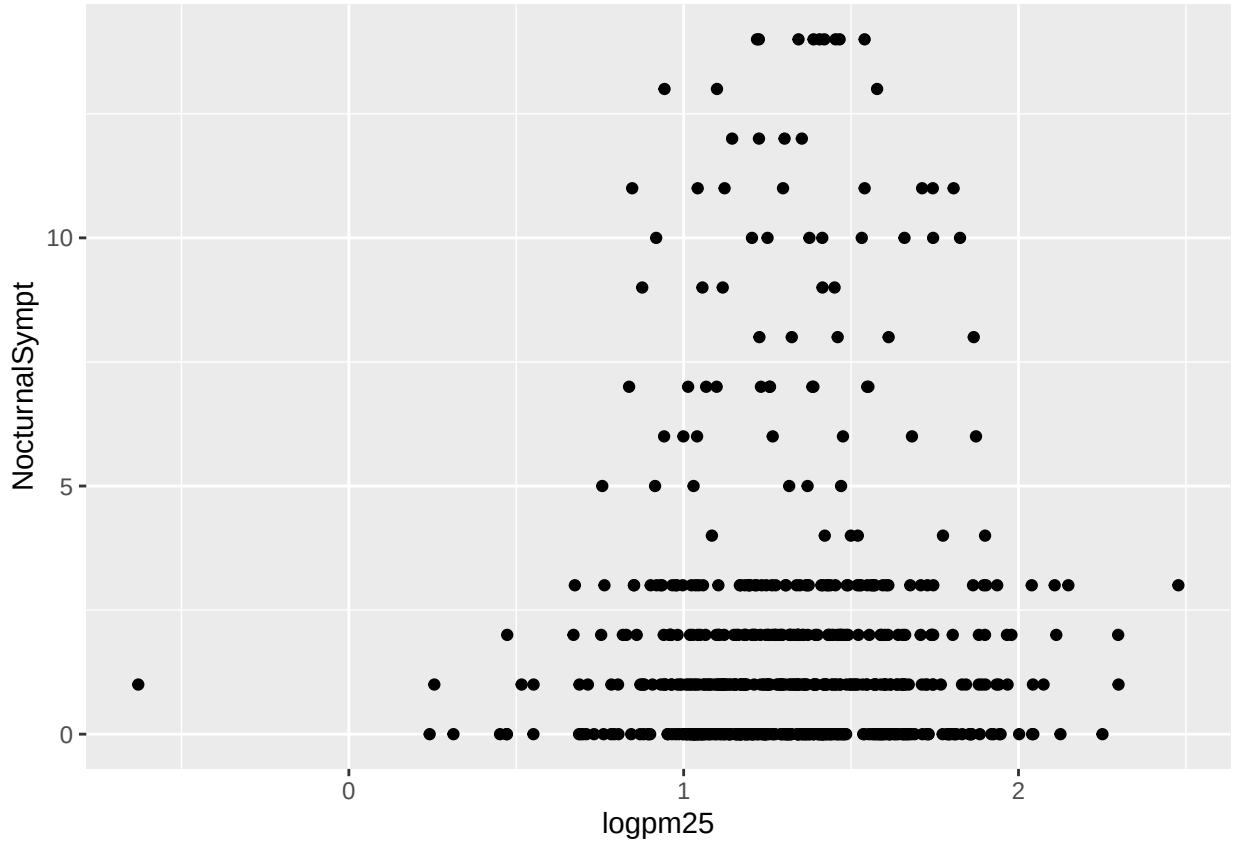
sin grafico aun

```
g <- ggplot(maacs, aes(logpm25, NocturnalSympt))
print(g)
```



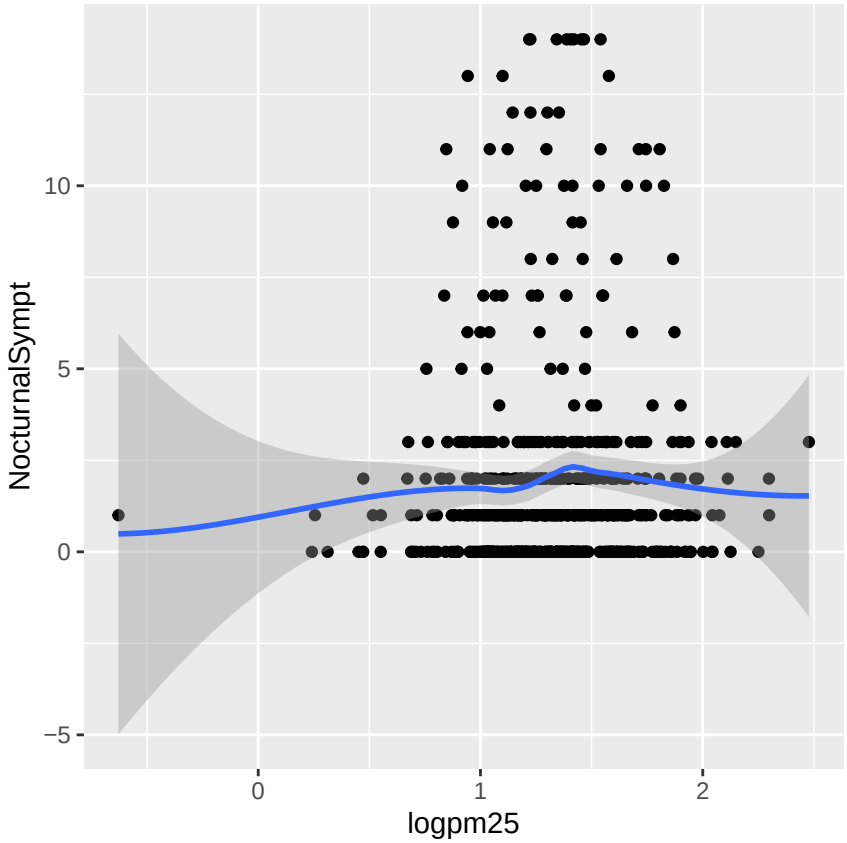
agregando puntos

```
g <- ggplot(maacs, aes(logpm25, NocturnalSympt))
g + geom_point()
```

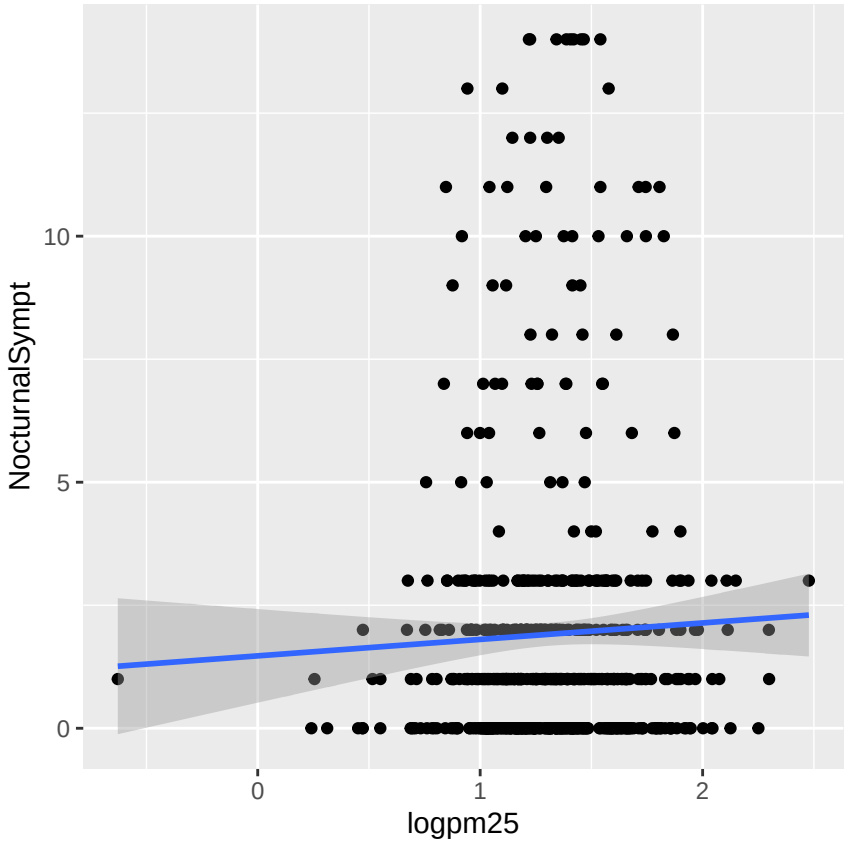


agregando mas capas

```
g + geom_point() + geom_smooth()
```

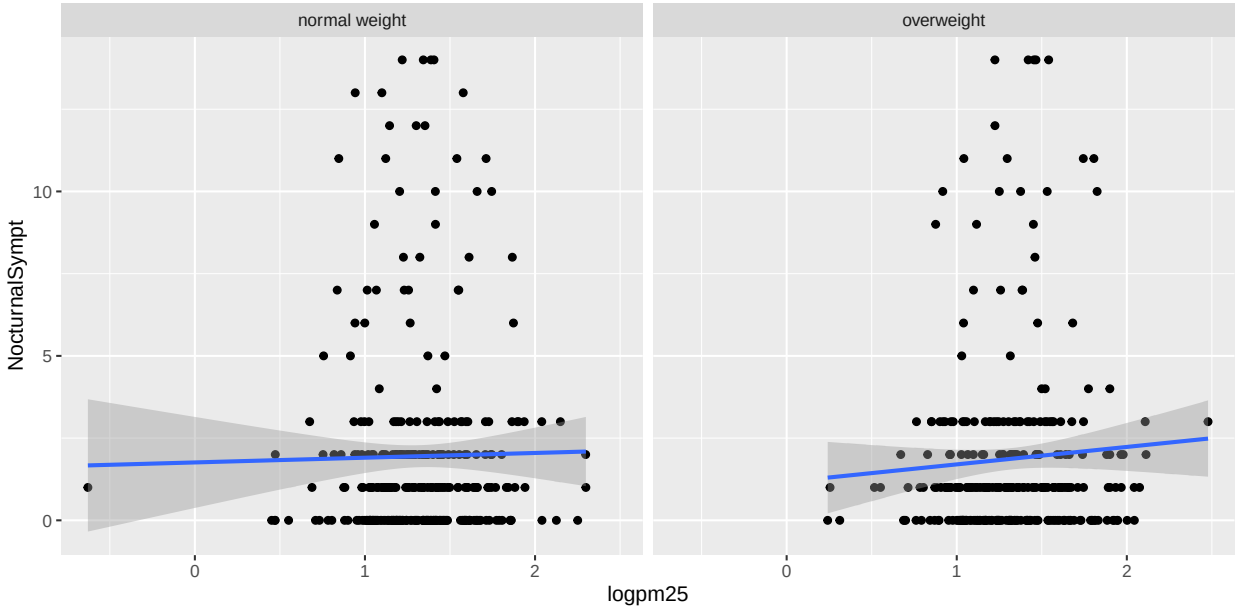



```
g + geom_point() + geom_smooth(method = "lm")
```



facetas

```
g + geom_point() + facet_grid(. ~ bmicat) + geom_smooth(method = "lm")
```

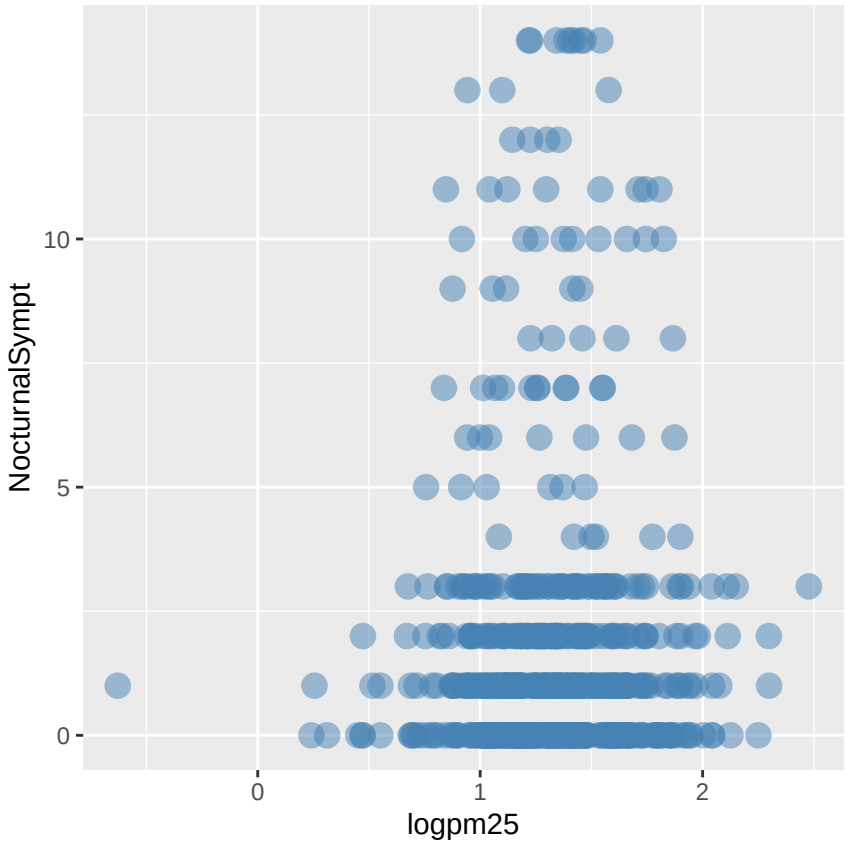


Anotación - Etiquetas: `xlab()`, `ylab()`, `labs()`, `ggtitle()` - Cada una de las funciones "geom" tiene opciones para modificar - Para cosas que solo tienen sentido globalmente, use `theme()` - Ejemplo:

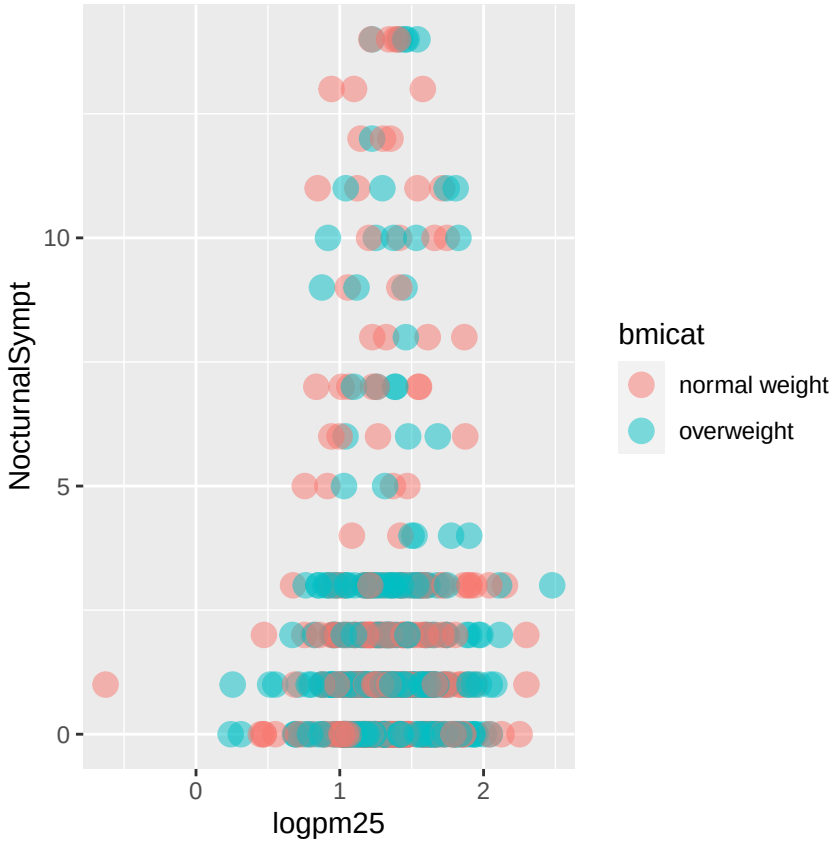
`theme(legend.position =" none ")` - Se incluyen dos temas de apariencia estándar - `theme_gray()`: El tema predeterminado (fondo gris) - `theme_bw()`: Más rígido / llano

modificando los esteticos

```
g + geom_point(color = "steelblue", size = 4, alpha = 1/2)
```




```
g + geom_point(aes(color = bmicat), size = 4, alpha = 1/2)
```

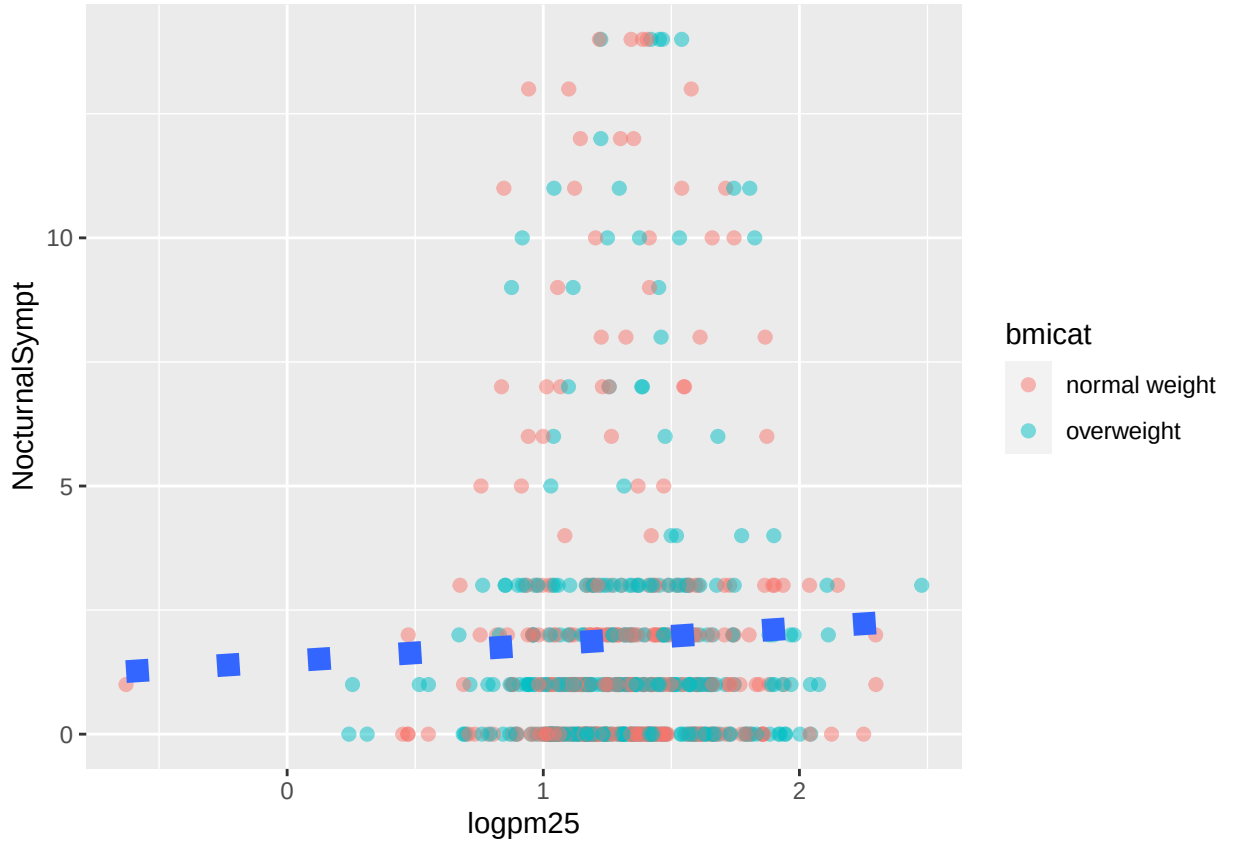


modificando las leyendas

```
g + geom_point(aes(color = bmicat)) + labs(title = "MAACS Cohort") +
  labs(x = expression("log " * PM[2.5]), y = "Nocturnal Symptoms")
```

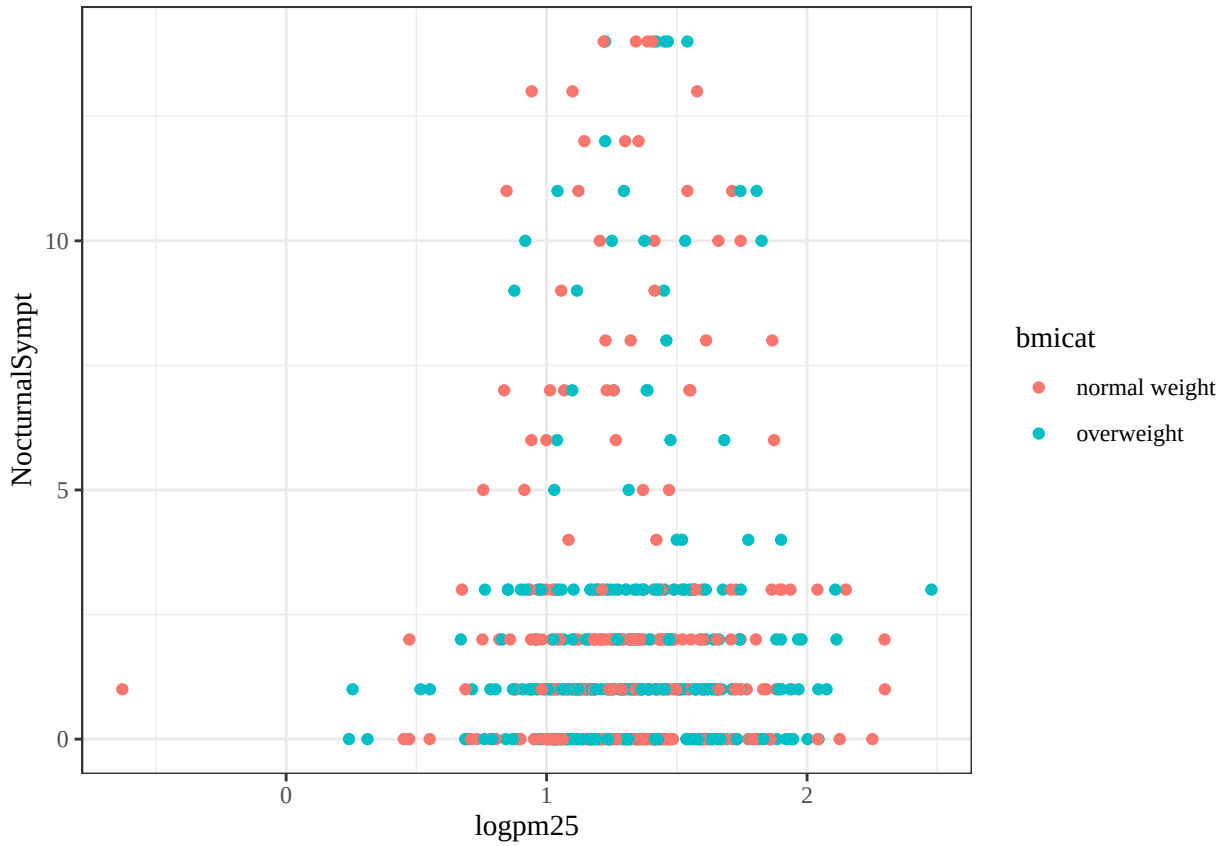
personalizacion de la linea

```
g + geom_point(aes(color = bmicat), size = 2, alpha = 1/2) +
  geom_smooth(size = 4, linetype = 3, method = "lm", se = FALSE)
```



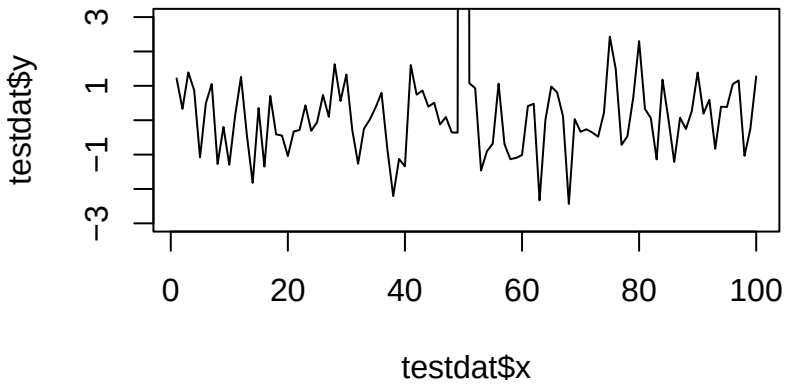
cambiando el tema

```
g + geom_point(aes(color = bmicat)) + theme_bw(base_family = "Times")
```

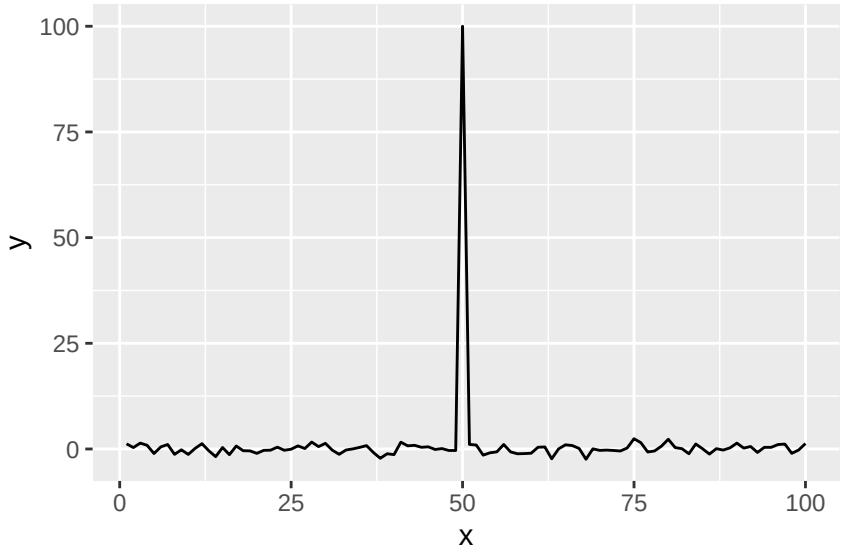


comparacion de el manejo de ejes entre base y ggplot2

```
testdat <- data.frame(x = 1:100, y = rnorm(100))
testdat[50,2] <- 100  ## Outlier!
plot(testdat$x, testdat$y, type = "l", ylim = c(-3,3))
```

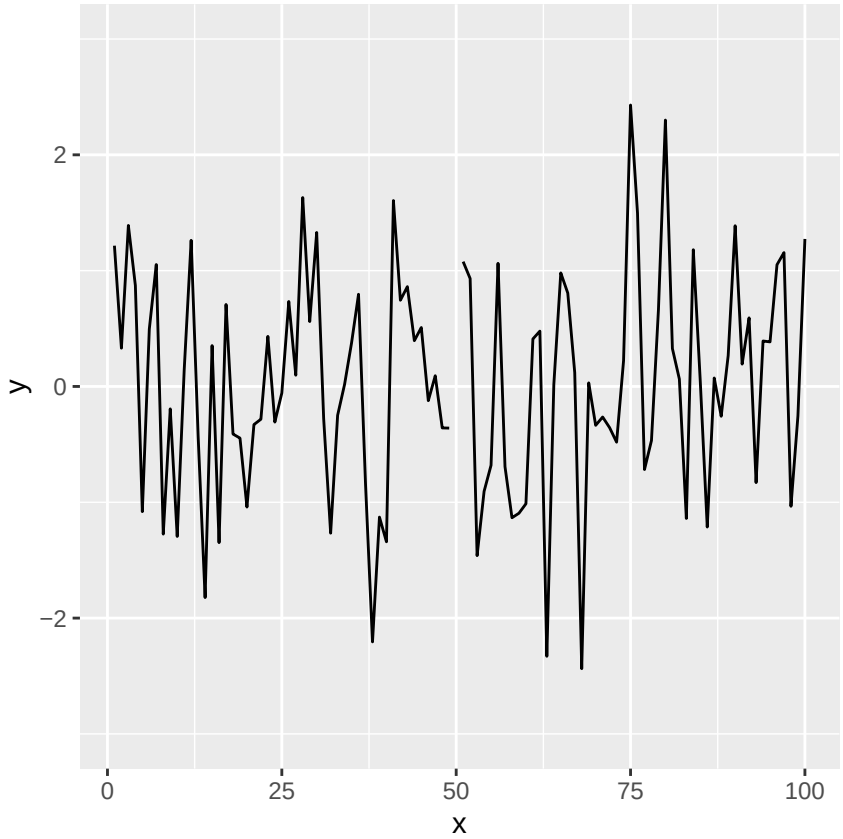



```
g <- ggplot(testdat, aes(x = x, y = y))
g + geom_line()
```

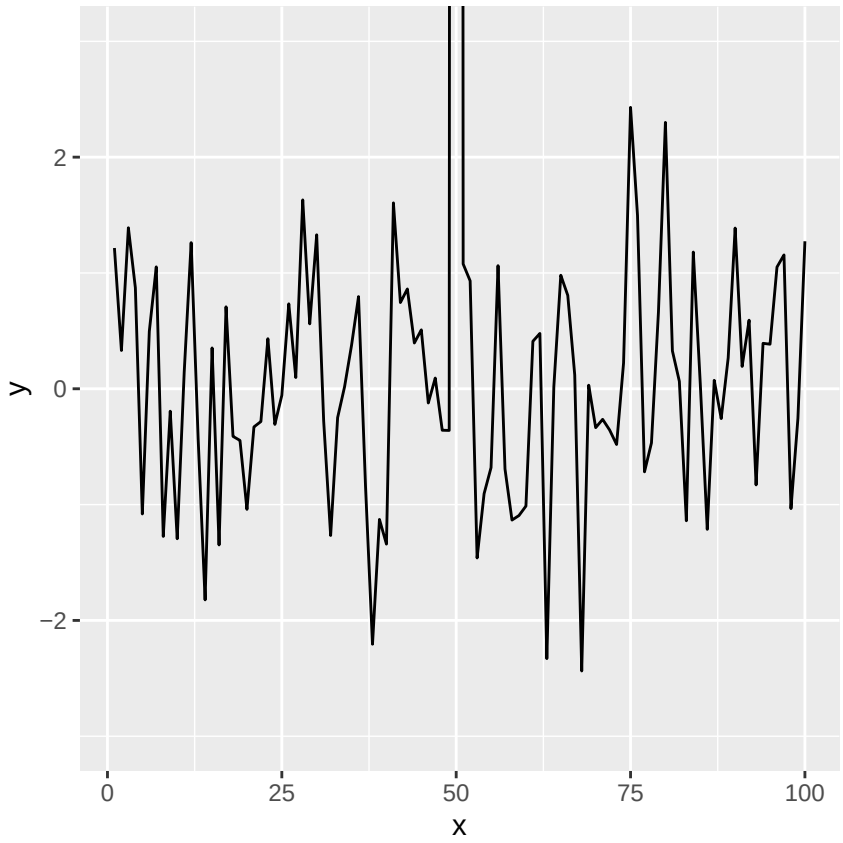


definiendo los limites

```
g + geom_line() + ylim(-3, 3)
```




```
g + geom_line() + coord_cartesian(ylim = c(-3, 3))
```



Ejemplo más complejo

- ¿Cómo varía la relación entre $PM_{2.5}$ y los síntomas nocturnos según el IMC y NO_2?
- A diferencia de nuestra variable anterior de IMC, NO_2 es continuo
- Necesitamos hacer que NO_2 sea categórico para que podamos condicionarlo en el trazado
- Utilice la función `cut()` para esto

```
## Calculate the tertiles of the data
cutpoints <- quantile(maacs$logno2_new, seq(0, 1, length = 4), na.rm = TRUE)

## Cut the data at the tertiles and create a new factor variable
maacs$no2tert <- cut(maacs$logno2_new, cutpoints)

## See the levels of the newly created factor variable
levels(maacs$no2tert)
```

```
## [1] "(-0.629,1.18]" "(1.18,1.44]"   "(1.44,2.48]"
```

```
## Setup ggplot with data frame
g <- ggplot(maacs, aes(logpm25, NocturnalSympt))

## Add layers
g + geom_point(alpha = 1/3) +
  facet_wrap(bmicat ~ no2tert, nrow = 2, ncol = 4) +
  geom_smooth(method="lm", se=FALSE, col="steelblue") +
  theme_bw(base_family = "Avenir", base_size = 10)
```

```
library(ggplot2)
```

Resumen y recursos

- La función `qplot ()` es análoga a `plot()` pero con muchas características integradas
- Sintaxis en algún lugar entre base / lattice
- Produce gráficos muy agradables, esencialmente listos para publicación (si te gusta el diseño)
- Difícil de ir contra la corriente / personalizar (no se moleste; use toda la potencia de ggplot2 en ese caso)
- El libro *ggplot2* de Hadley Wickham
- El *R Graphics Cookbook* de Winston Chang (ejemplos en gráficos base y en ggplot2)
- sitio web ggplot2 (http://ggplot2.org)
- lista de correo ggplot2 (http://goo.gl/OdW3uB), principalmente para desarrolladores