The Lattice Plotting System in R

Roger D. Peng, Associate Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

The Lattice Plotting System

The lattice plotting system is implemented using the following packages:

- *lattice*: contains code for producing Trellis graphics, which are independent of the “base” graphics system; includes functions like `xyplot`, `bwplot`, `levelplot`
- *grid*: implements a different graphing system independent of the “base” system; the *lattice* package builds on top of *grid*
  - We seldom call functions from the *grid* package directly
- The lattice plotting system does not have a "two-phase" aspect with separate plotting and annotation like in base plotting
- All plotting/annotation is done at once with a single function call

Lattice Functions

- `xyplot`: this is the main function for creating scatterplots
- `bwplot`: box-and-whiskers plots (“boxplots”)
- `histogram`: histograms
- `stripplot`: like a boxplot but with actual points
- `dotplot`: plot dots on "violin strings"
- `splom`: scatterplot matrix; like `pairs` in base plotting system
- `levelplot`, `contourplot`: for plotting "image" data

Lattice Functions

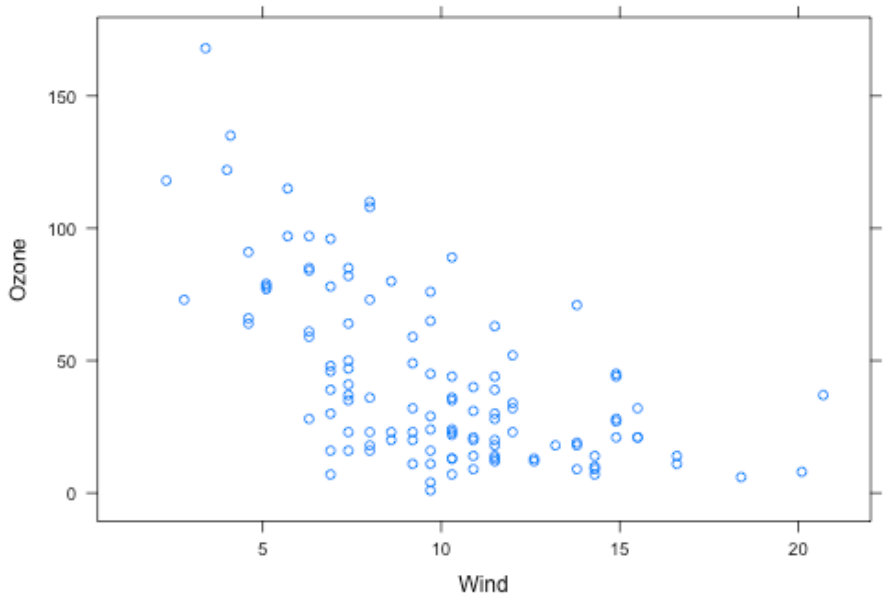
Lattice functions generally take a formula for their first argument, usually of the form

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

- We use the *formula notation* here, hence the ~.
- On the left of the ~ is the y-axis variable, on the right is the x-axis variable
- f and g are *conditioning variables* — they are optional
  - the * indicates an interaction between two variables
- The second argument is the data frame or list from which the variables in the formula should be looked up
  - If no data frame or list is passed, then the parent frame is used.
- If no other arguments are passed, there are defaults that can be used.

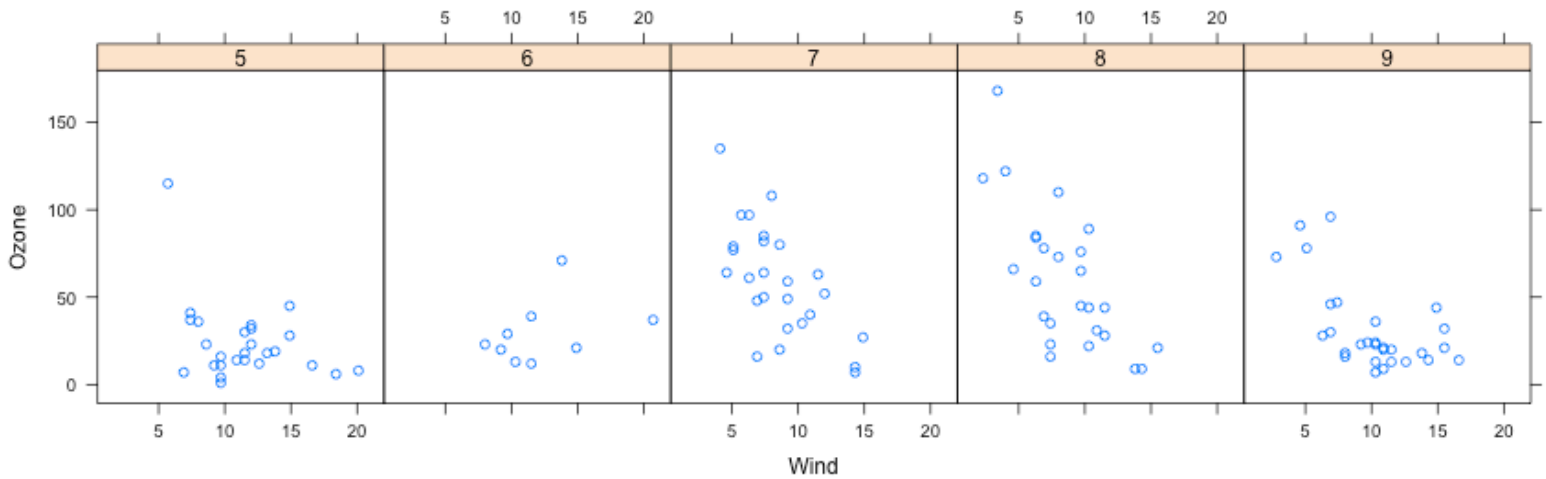
Simple Lattice Plot

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



Simple Lattice Plot

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



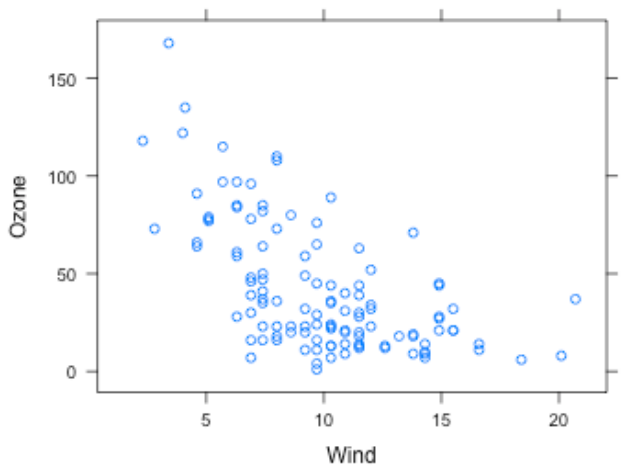
Lattice Behavior

Lattice functions behave differently from base graphics functions in one critical way.

- Base graphics functions plot data directly to the graphics device (screen, PDF file, etc.)
- Lattice graphics functions return an object of class **trellis**
- The print methods for lattice functions actually do the work of plotting the data on the graphics device.
- Lattice functions return "plot objects" that can, in principle, be stored (but it's usually better to just save the code + data).
- On the command line, trellis objects are *auto-printed* so that it appears the function is plotting the data

Lattice Behavior

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



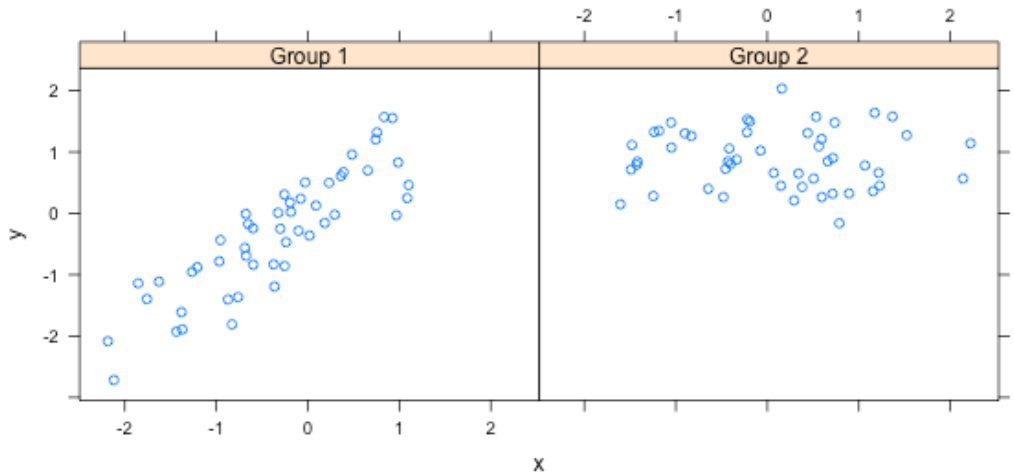

```
xyplot(Ozone ~ Wind, data = airquality)  ## Auto-printing
```

Lattice Panel Functions

- Lattice functions have a **panel function** which controls what happens inside each panel of the plot.
- The *lattice* package comes with default panel functions, but you can supply your own if you want to customize what happens in each panel
- Panel functions receive the x/y coordinates of the data points in their panel (along with any optional arguments)

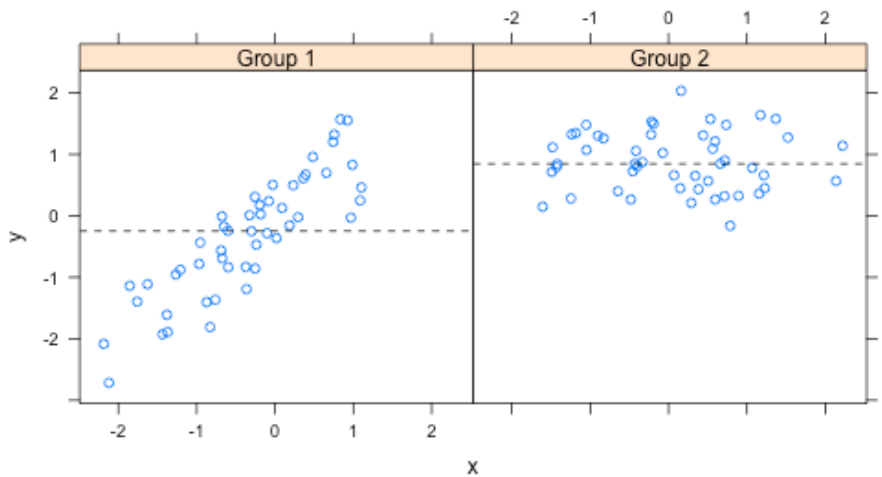
Lattice Panel Functions

```
set.seed(10)
x <- rnorm(100)
f <- rep(0:1, each = 50)
y <- x + f - f * x + rnorm(100, sd = 0.5)
f <- factor(f, labels = c("Group 1", "Group 2"))
xyplot(y ~ x | f, layout = c(2, 1))  ## Plot with 2 panels
```



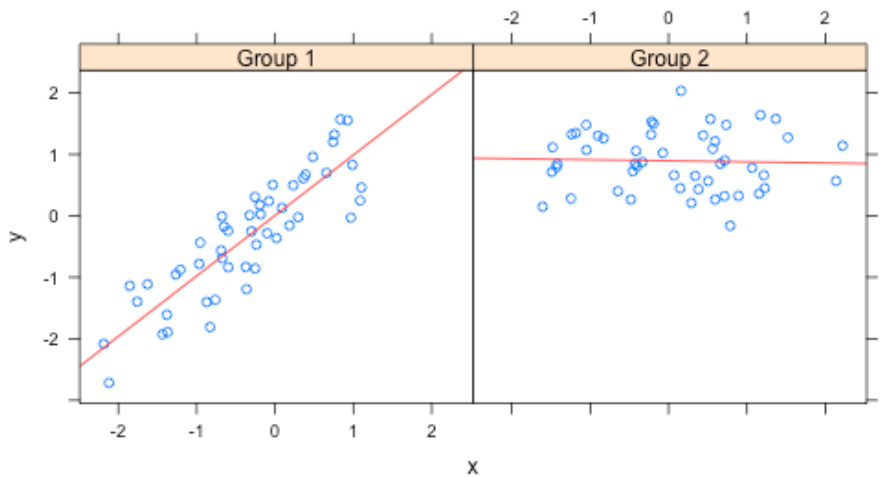
Lattice Panel Functions

```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
})
```



Lattice Panel Functions: Regression line

```
## 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
})
```

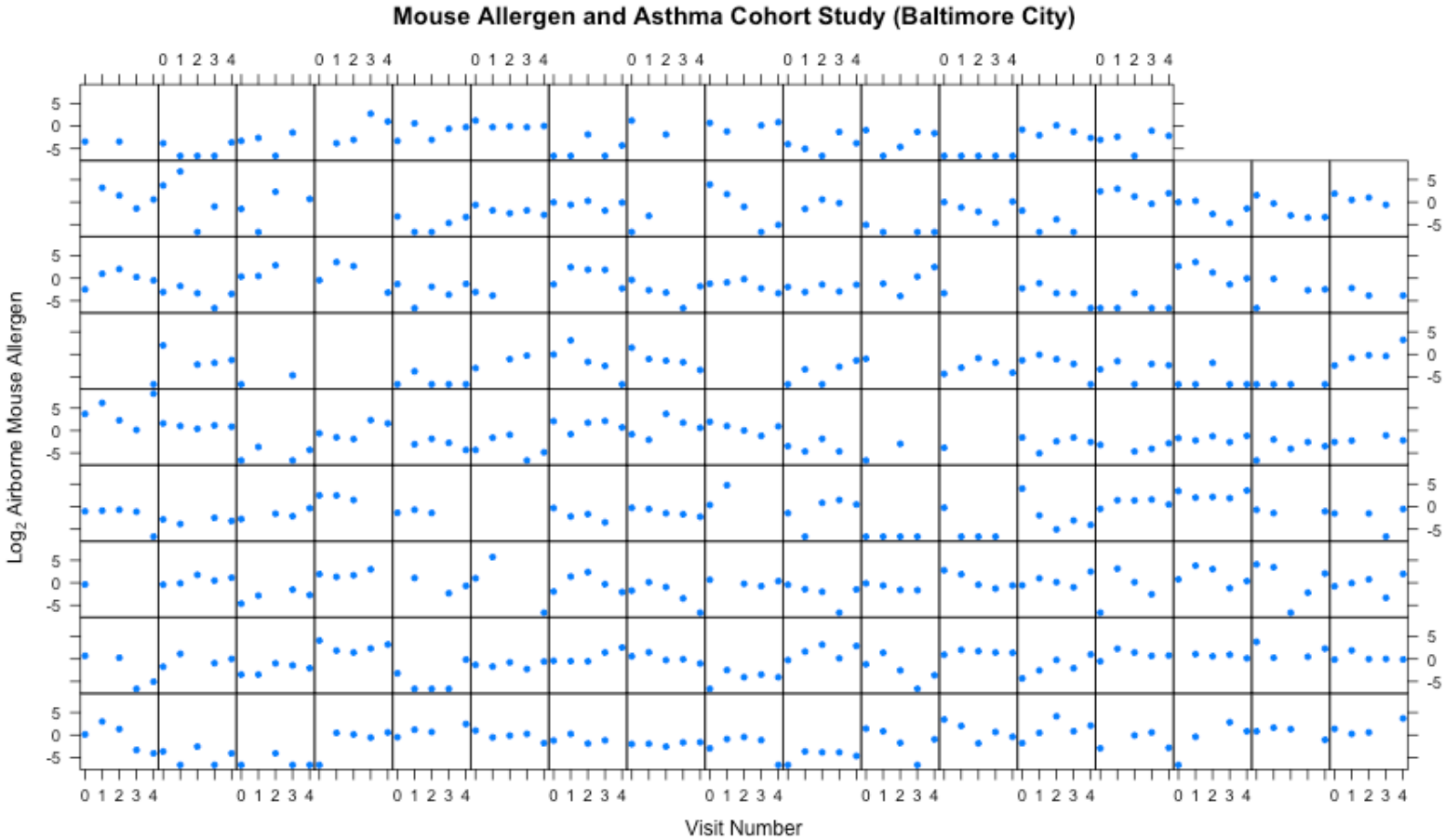


Many Panel Lattice Plot: Example from MAACS

- Study: Mouse Allergen and Asthma Cohort Study (MAACS)
- Study subjects: Children with asthma living in Baltimore City, many allergic to mouse allergen
- Design: Observational study, baseline home visit + every 3 months for a year.
- Question: How does indoor airborne mouse allergen vary over time and across subjects?

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

Many Panel Lattice Plot



Summary

- Lattice plots are constructed with a single function call to a core lattice function (e.g. `xyplot`)
- Aspects like margins and spacing are automatically handled and defaults are usually sufficient
- The lattice system is ideal for creating conditioning plots where you examine the same kind of plot under many different conditions
- Panel functions can be specified/customized to modify what is plotted in each of the plot panels

Plotting with ggplot2

Exploratory Data Analysis

Roger D. Peng, Associate Professor of Biostatistics
Johns Hopkins Bloomberg School of Public Health

What is ggplot2?

- An implementation of the *Grammar of Graphics* by Leland Wilkinson
- Written by Hadley Wickham (while he was a graduate student at Iowa State)
- A "third" graphics system for R (along with **base** and **lattice**)
- Available from CRAN via `install.packages()`
- Web site: http://ggplot2.org (better documentation)

What is ggplot2?

- Grammar of graphics represents and abstraction of graphics ideas/objects
- Think “verb”, “noun”, “adjective” for graphics
- Allows for a “theory” of graphics on which to build new graphics and graphics objects
- “Shorten the distance from mind to page”

Grammar of Graphics

"In brief, the grammar tells us that a statistical graphic is a **mapping** from data to **aesthetic** attributes (colour, shape, size) of **geometric** objects (points, lines, bars). The plot may also contain statistical transformations of the data and is drawn on a specific coordinate system"

from *ggplot2* book

The Basics: `qplot()`

- Works much like the `plot` function in base graphics system
- Looks for data in a data frame, similar to lattice, or in the parent environment
- Plots are made up of *aesthetics* (size, shape, color) and *geoms* (points, lines)

The Basics: `qplot()`

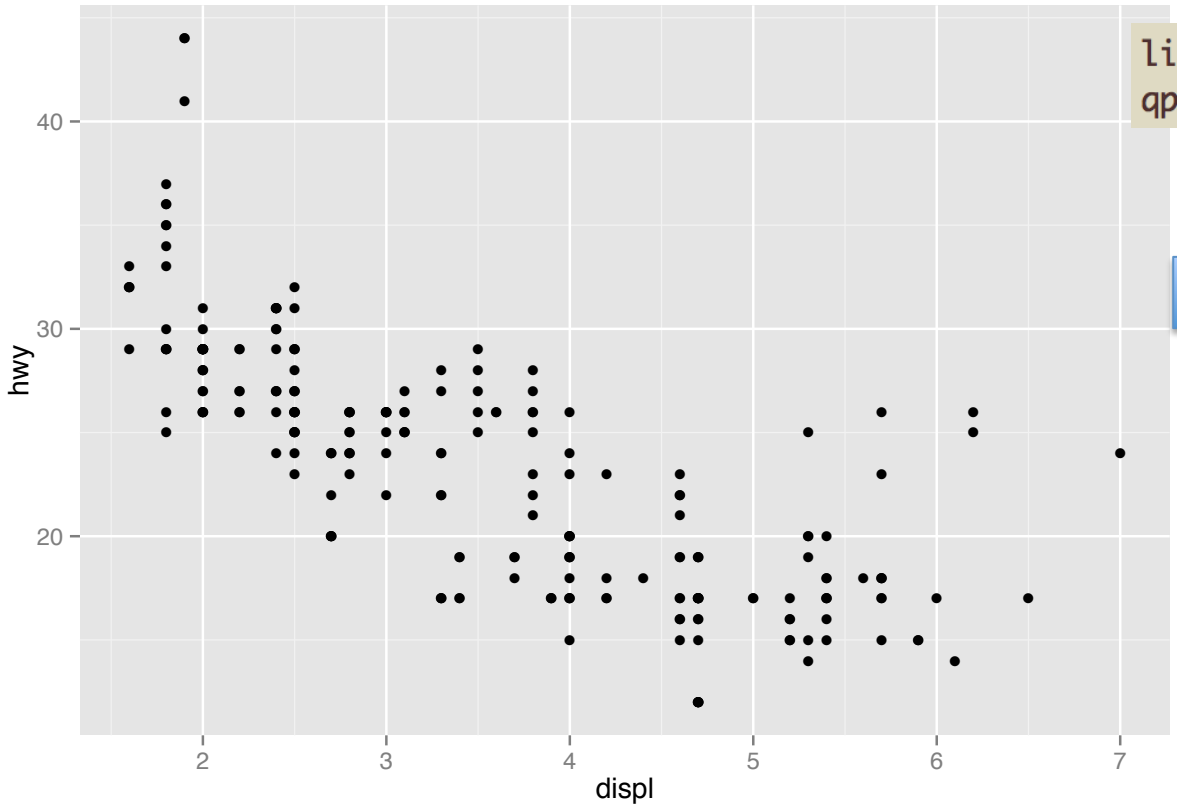
- Factors are important for indicating subsets of the data (if they are to have different properties); they should be **labeled**
- The qplot() hides what goes on underneath, which is okay for most operations
- ggplot() is the core function and very flexible for doing things qplot() cannot do

Example Dataset

```
> library(ggplot2)
> str(mpg)
'data.frame':   234 obs. of  11 variables:
 $ manufacturer: Factor w/ 15 levels "audi","chevrolet",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ model       : Factor w/ 38 levels "4runner 4wd",..: 2 2 2 2 2 2 2 3 3 3 ...
 $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
 $ year        : int  1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
 $ cyl         : int  4 4 4 4 6 6 6 4 4 4 ...
 $ trans       : Factor w/ 10 levels "auto(av)","auto(l3)",..: 4 9 10 1 4 9 1 9 4 10
...
 $ drv         : Factor w/ 3 levels "4","f","r": 2 2 2 2 2 2 2 1 1 1 ...
 $ cty         : int  18 21 20 21 16 18 18 18 16 20 ...
 $ hwy         : int  29 29 31 30 26 26 27 26 25 28 ...
 $ fl          : Factor w/ 5 levels "c","d","e","p",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ class       : Factor w/ 7 levels "2seater","compact",..: 2 2 2 2 2 2 2 2 2 2 ...
```

Factor label information important for annotation

ggplot2 “Hello, world!”



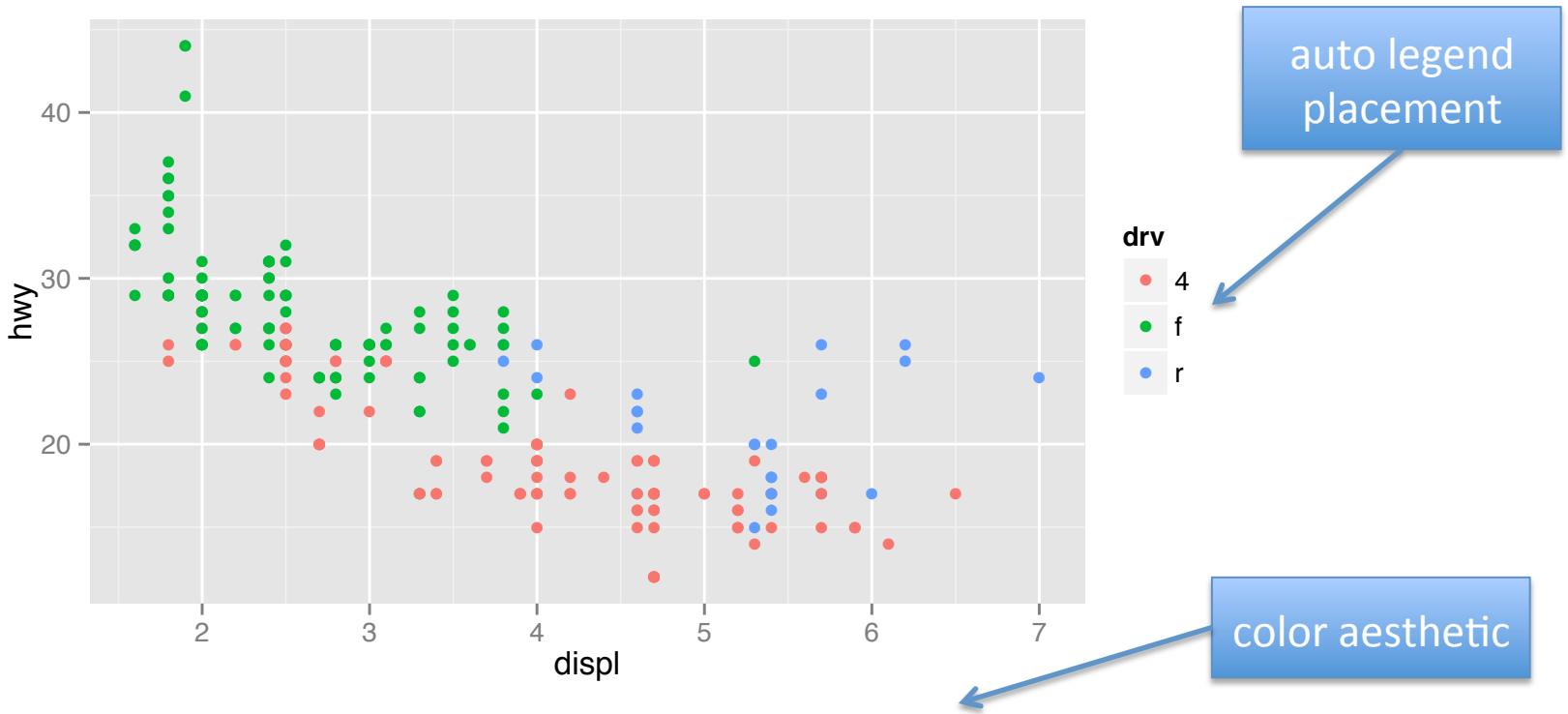

```
library(ggplot2)
qplot(displ, hwy, data = mpg)
```

x coord

y coord

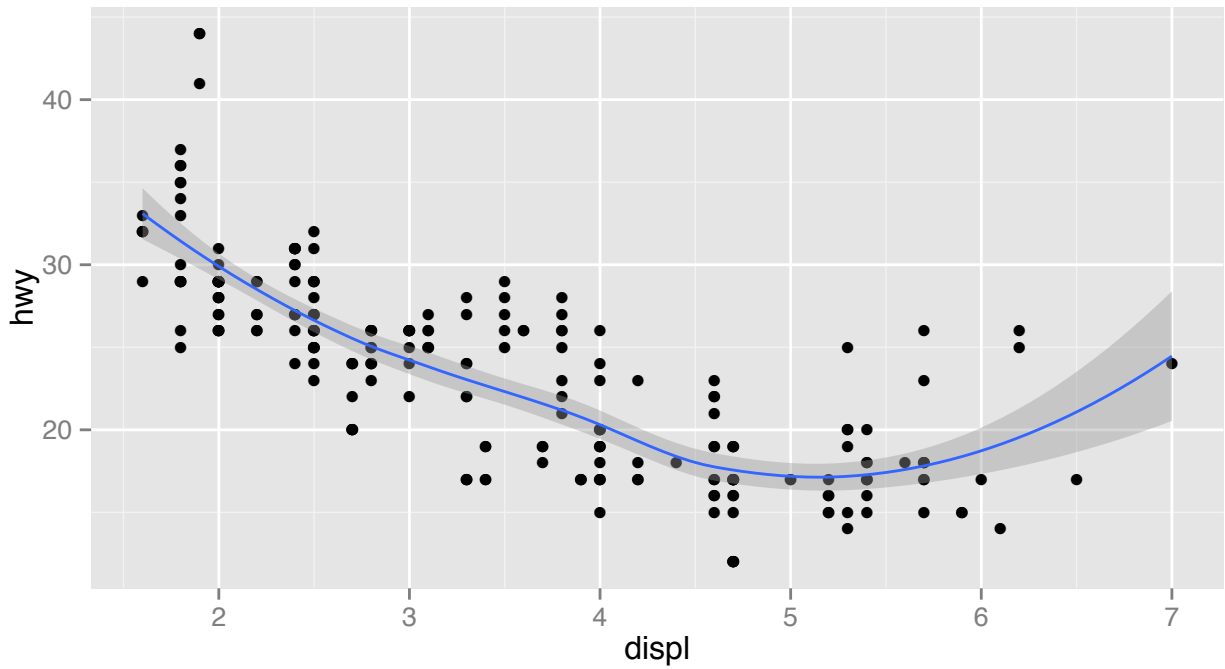
data frame

Modifying aesthetics



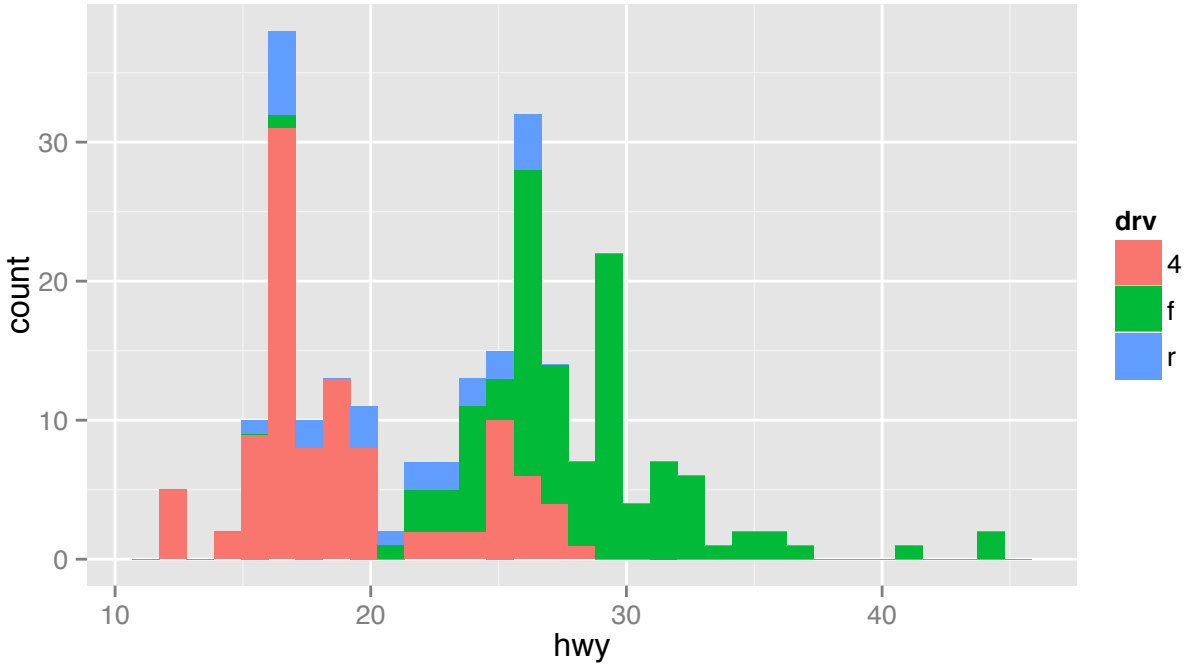
qplot(displ, hwy, data = mpg, color = drv)

Adding a geom



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

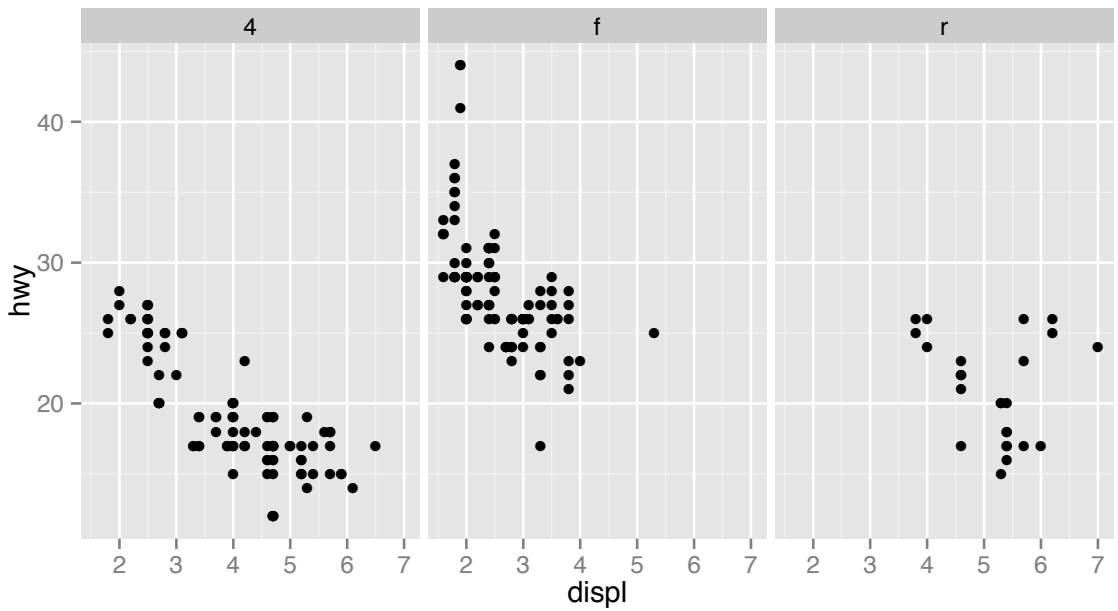
Histograms

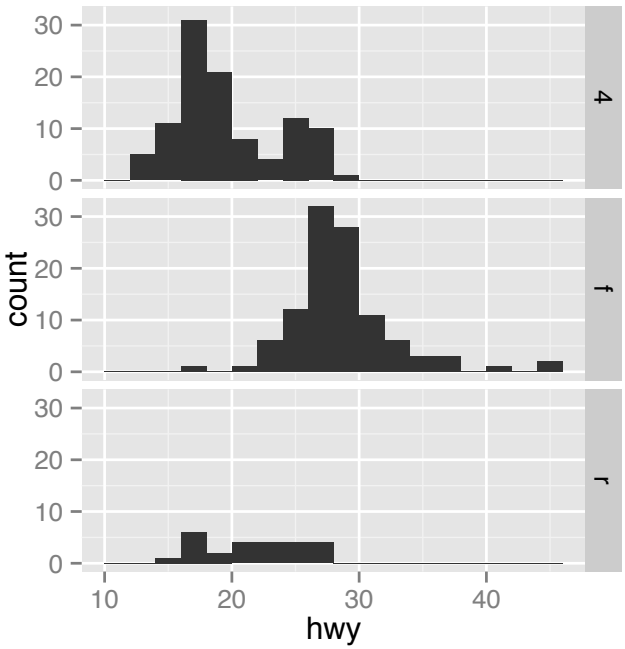


qplot(hwy, data = mpg, fill = drv)

Facets

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





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

MAACS Cohort

- Mouse Allergen and Asthma Cohort Study
- Baltimore children (aged 5—17)
- Persistent asthma, exacerbation in past year
- Study indoor environment and its relationship with asthma morbidity
- Recent publication: http://goo.gl/WqE9j8

Example: MAACS

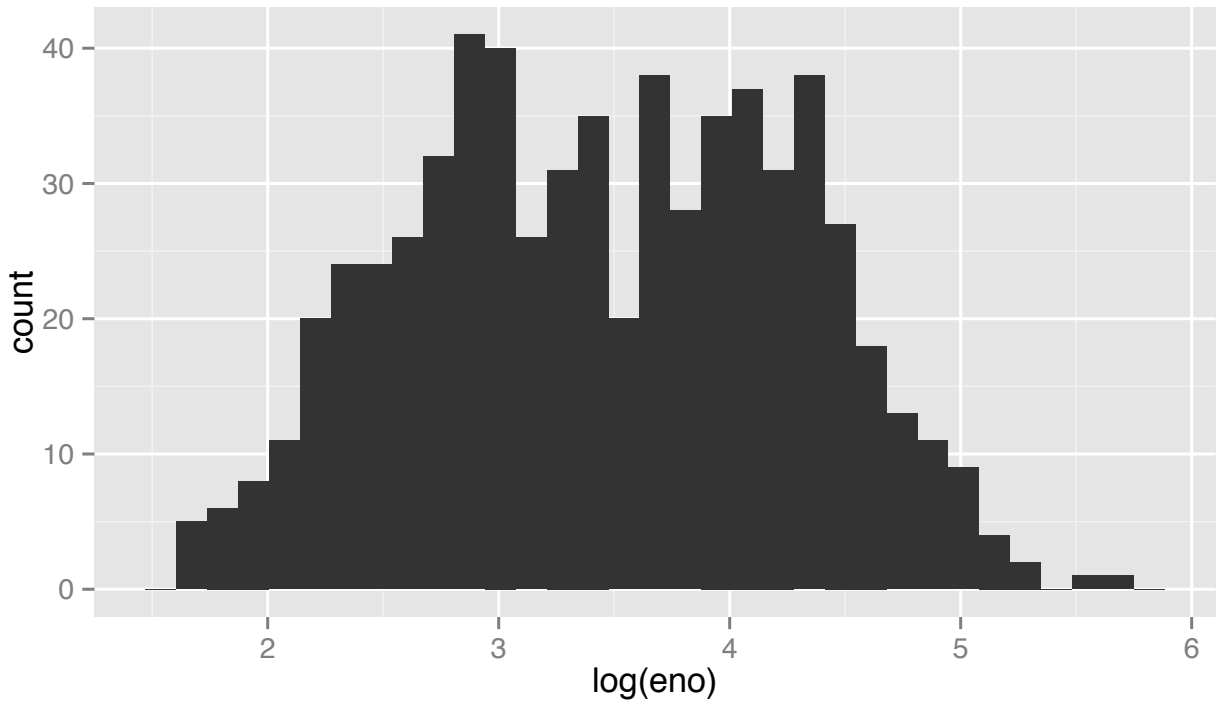
Exhaled nitric oxide

```
> str(maacs)
'data.frame':   750 obs. of  5 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 ...
```

Fine particulate matter

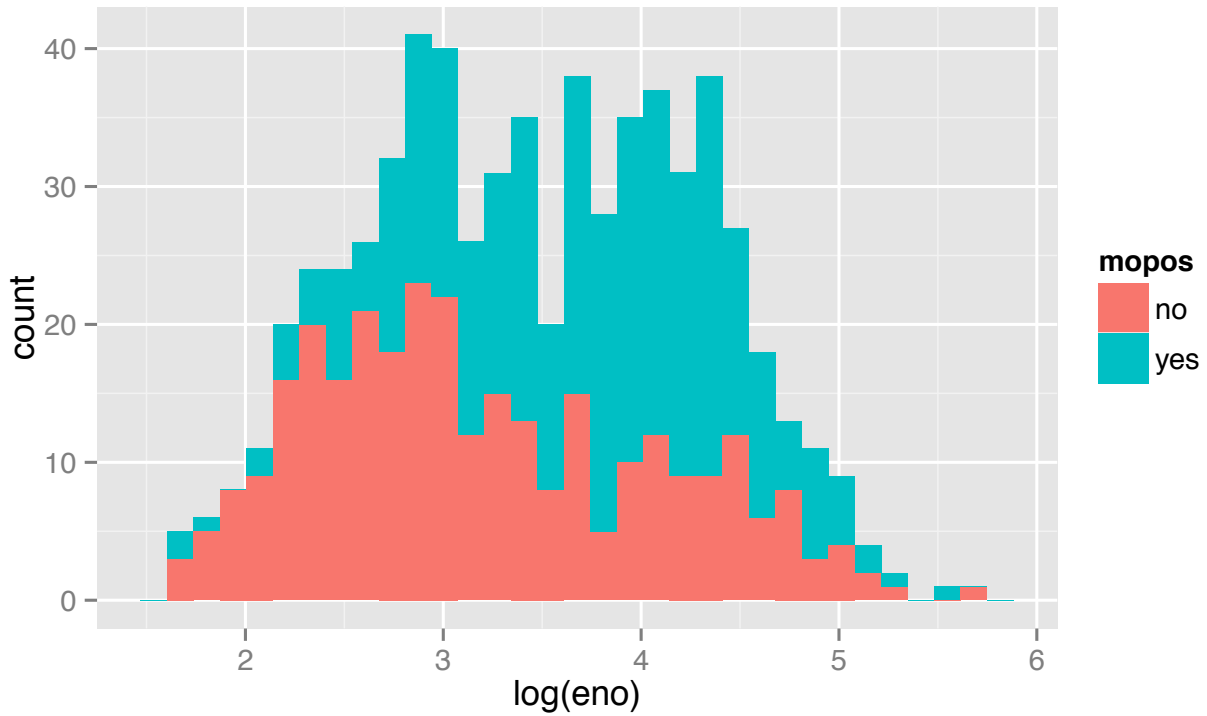
Sensitized to mouse allergen

Histogram of eNO



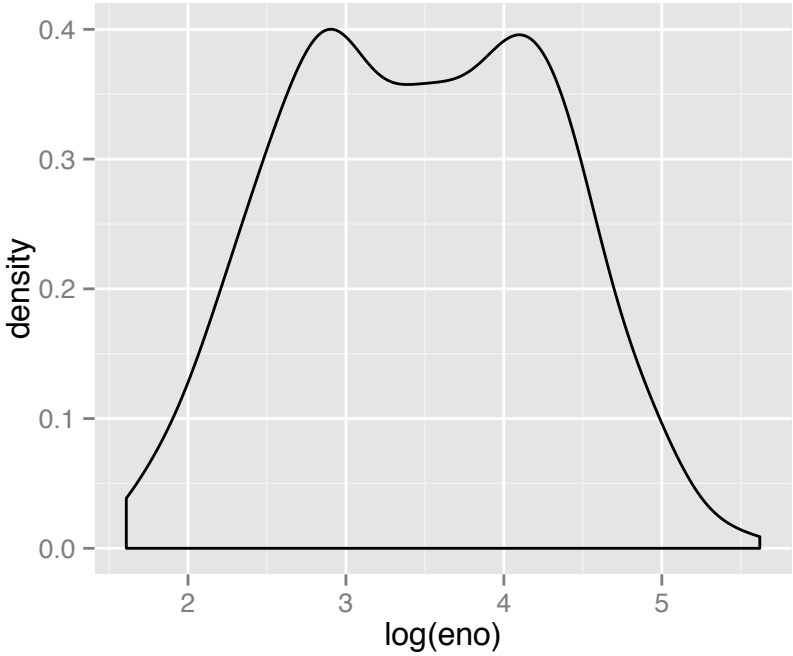
qplot(log(eno), data = maacs)

Histogram by Group

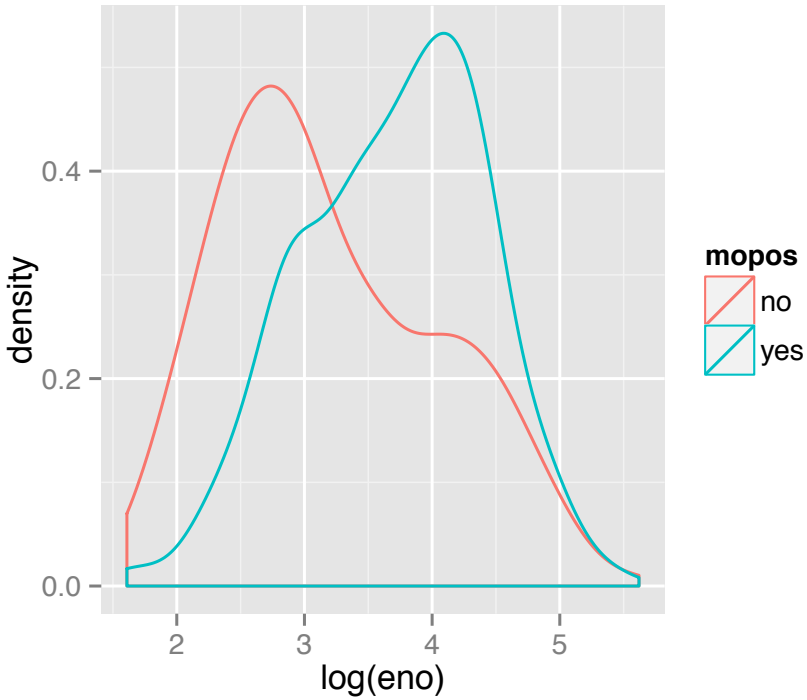


qplot(log(eno), data = maacs, fill = mopos)

Density Smooth

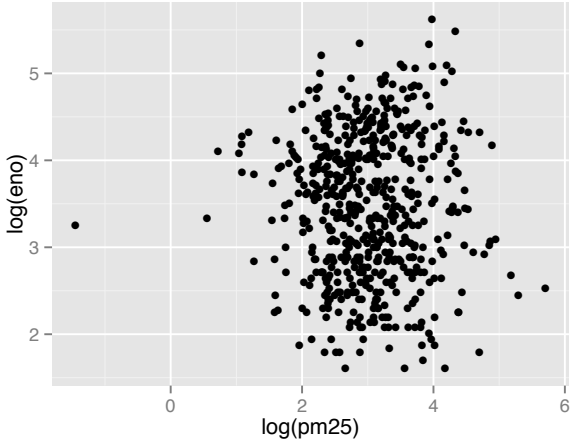


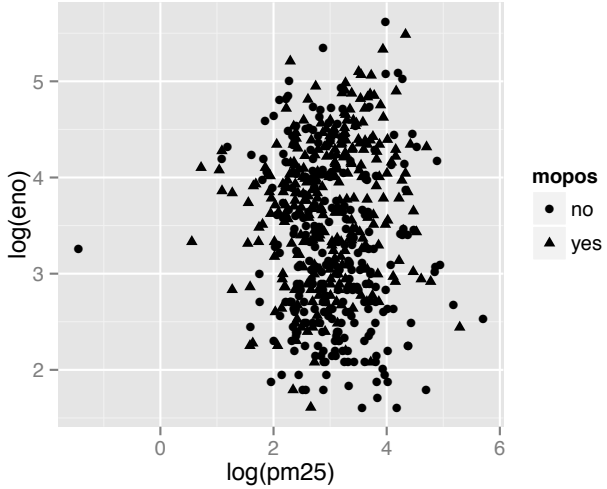
qplot(log(eno), data = maacs, geom = "density")

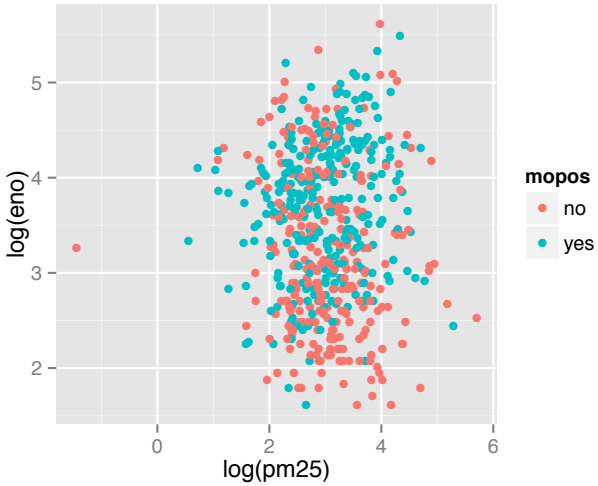


qplot(log(eno), data = maacs, geom = "density", color = mopos)

Scatterplots: eNO vs. $PM_{2.5}$





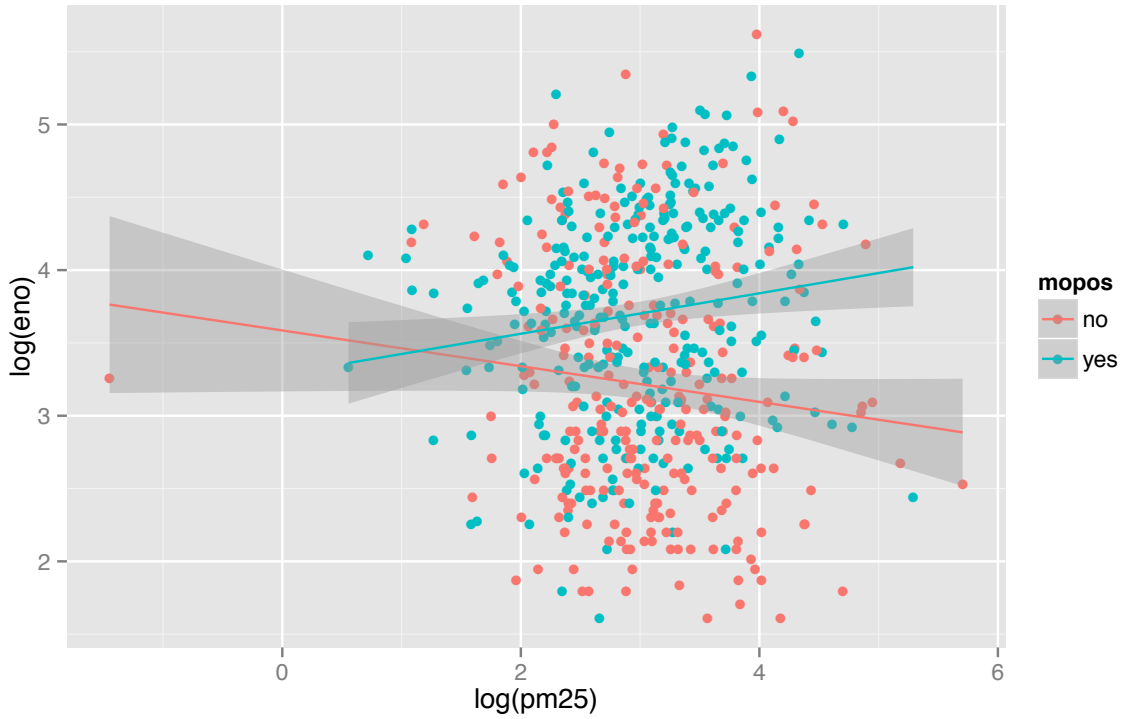



```
qplot(log(pm25), log(eno), data =
maacs)
```

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

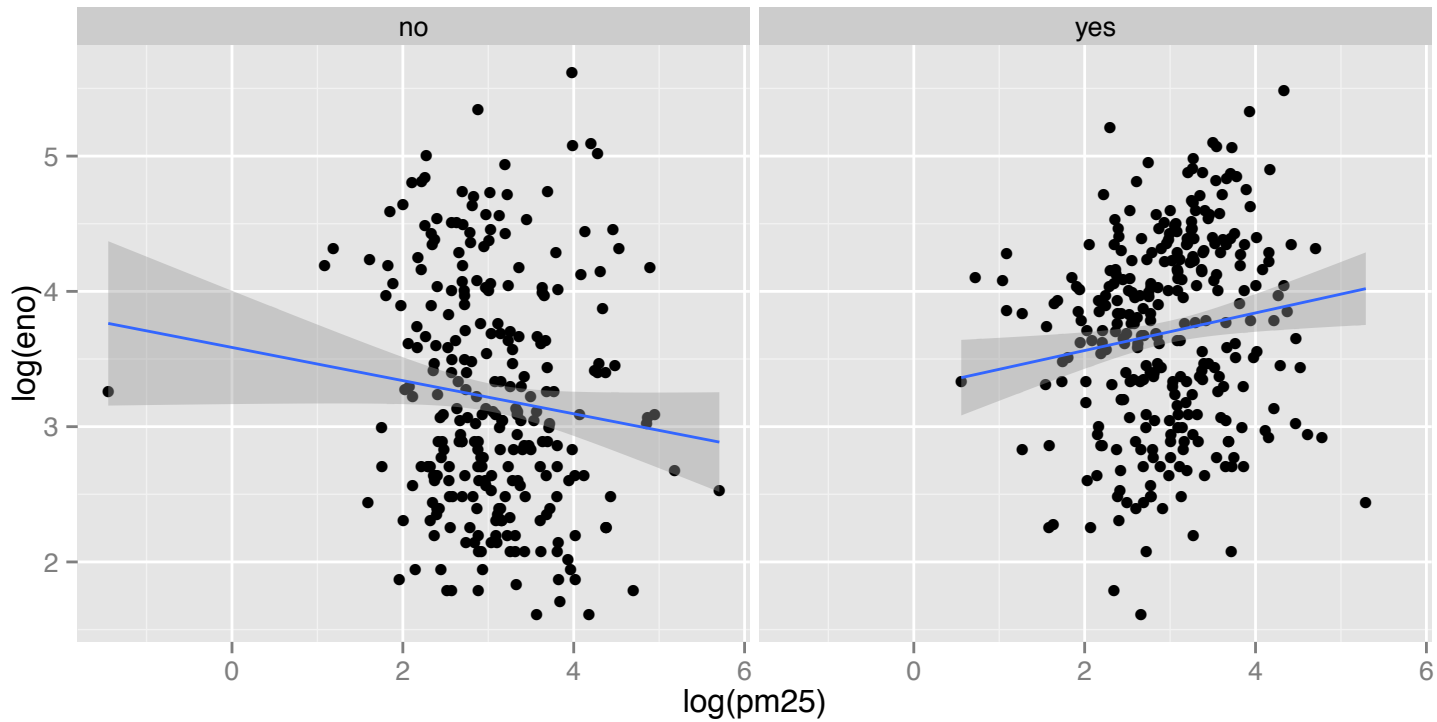
```
qplot(log(pm25), log(eno), data =
maacs, color = mopos)
```

Scatterplots: eNO vs. $PM_{2.5}$



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

Scatterplots: eNO vs. $PM_{2.5}$



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

Summary of qplot()

- The qplot() function is the analog to plot() but with many built-in features
- Syntax somewhere in between base/lattice
- Produces very nice graphics, essentially publication ready (if you like the design)
- Difficult to go against the grain/customize (don't bother; use full ggplot2 power in that case)

Resources

- The *ggplot2* book by Hadley Wickham
- The *R Graphics Cookbook* by Winston Chang (examples in base plots and in ggplot2)
- ggplot2 web site (http://ggplot2.org)
- ggplot2 mailing list (http://goo.gl/OdW3uB), primarily for developers

What is ggplot2?

- An implementation of the *Grammar of Graphics* by Leland Wilkinson
- Grammar of graphics represents and abstraction of graphics ideas/objects
- Think “verb”, “noun”, “adjective” for graphics
- Allows for a “theory” of graphics on which to build new graphics and graphics objects

Basic Components of a ggplot2 Plot

- A **data frame**
- **aesthetic mappings**: how data are mapped to color, size
- **geoms**: geometric objects like points, lines, shapes.
- **facets**: for conditional plots.
- **stats**: statistical transformations like binning, quantiles, smoothing.
- **scales**: what scale an aesthetic map uses (example: male = red, female = blue).
- **coordinate system**

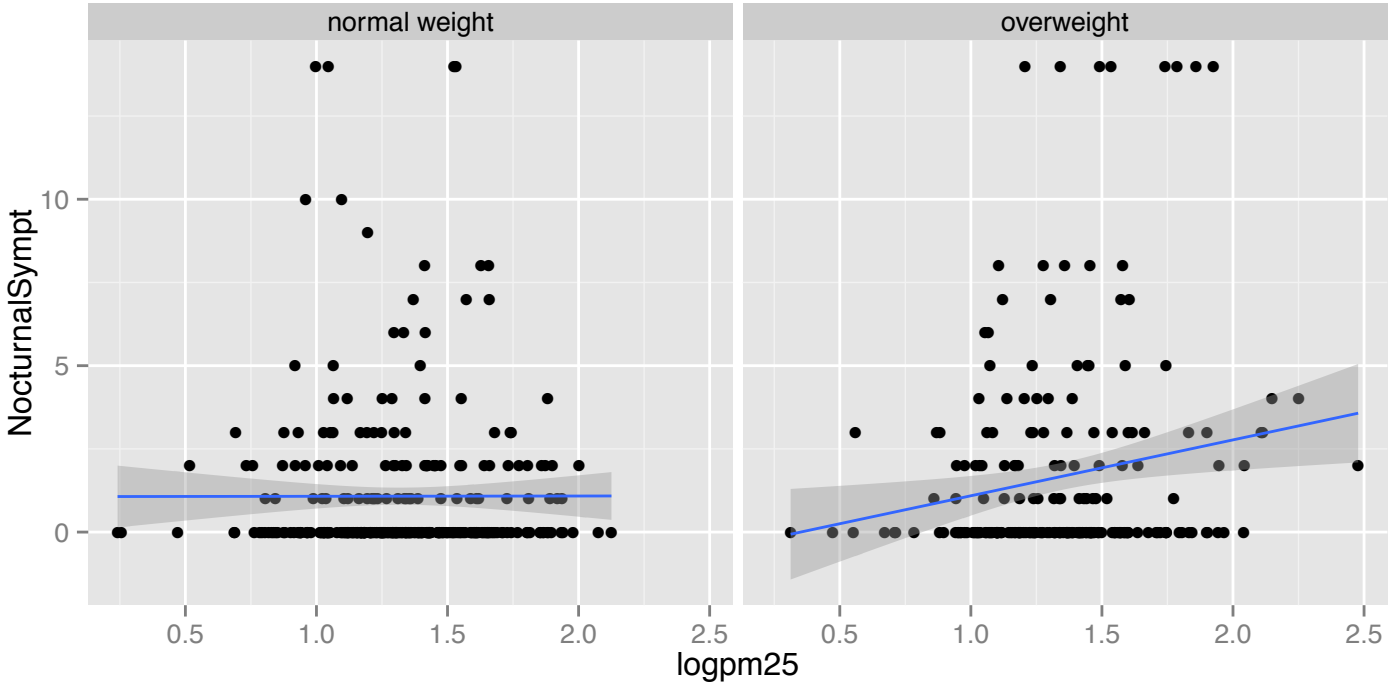
Building Plots with ggplot2

- When building plots in ggplot2 (rather than using qplot) the “artist’s palette” model may be the closest analogy
- Plots are built up in layers
  - Plot the data
  - Overlay a summary
  - Metadata and annotation

Example: BMI, $PM_{2.5}$, Asthma

- Mouse Allergen and Asthma Cohort Study
- Baltimore children (age 5-17)
- Persistent asthma, exacerbation in past year
- Does BMI (normal vs. overweight) modify the relationship between $PM_{2.5}$ and asthma symptoms?

Basic Plot




```
qplot(logpm25, NocturnalSympt, data = maacs, facets = . ~ bmicat, geom =
c("point", "smooth"), method = "lm”)
```

Building Up in Layers

```
> head(maacs)
    logpm25        bmicat NocturnalSympt
2 1.5361795 normal weight              1
3 1.5905409 normal weight              0
4 1.5217786 normal weight              0
5 1.4323277 normal weight              0
6 1.2762320    overweight              8
8 0.7139103    overweight              0

> g <- ggplot(maacs, aes(logpm25, NocturnalSympt))

> summary(g)
data: logpm25, bmicat, NocturnalSympt [554x3]
mapping:  x = logpm25, y = NocturnalSympt
faceting: facet_null()
```

Data Frame

Aesthetics

Initial call to ggplot

Summary of ggplot object

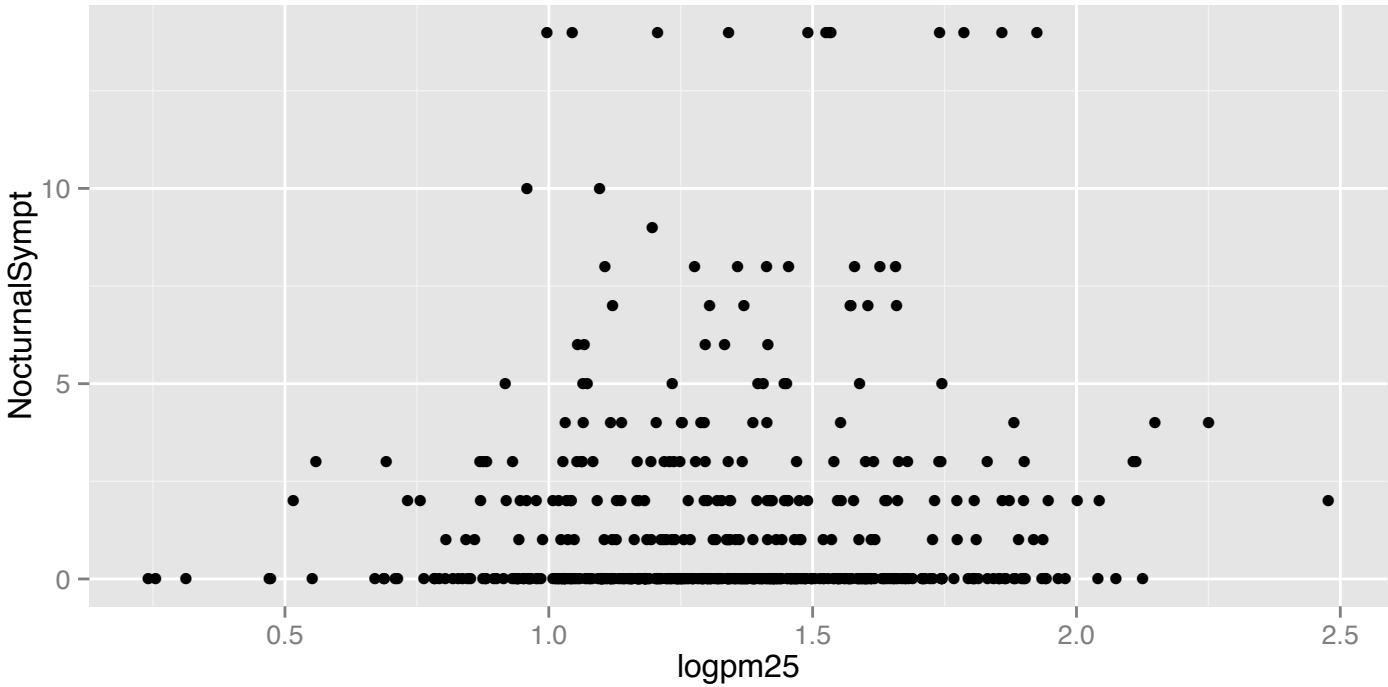
No Plot Yet!

```
> g <- ggplot(maacs, aes(logpm25, NocturnalSympt))
> print(g)
Error: No layers in plot

> p <- g + geom_point()
> print(p)

> g + geom_point()
```

Explicitly save and print ggplot object

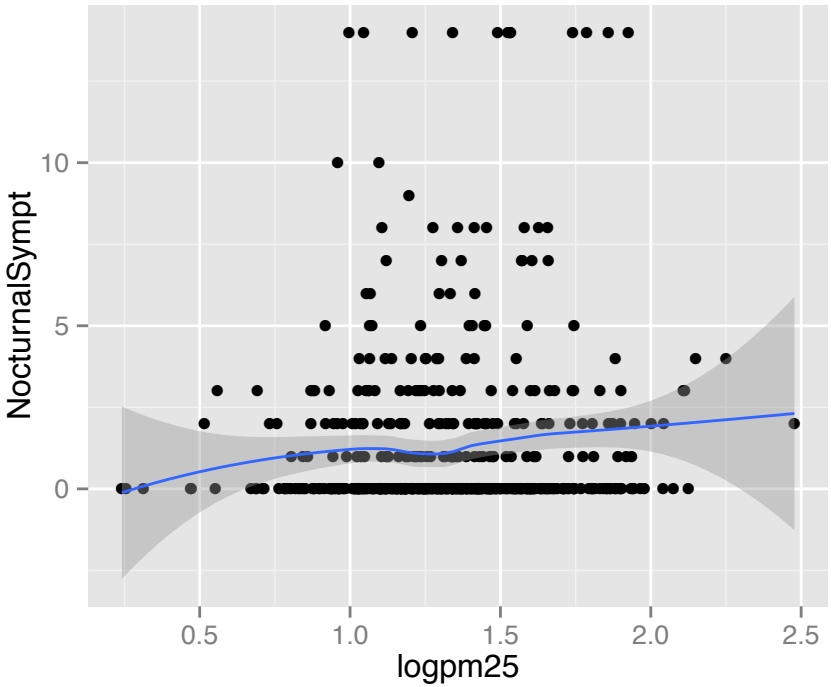
Auto-print plot object without saving

First Plot with Point Layer

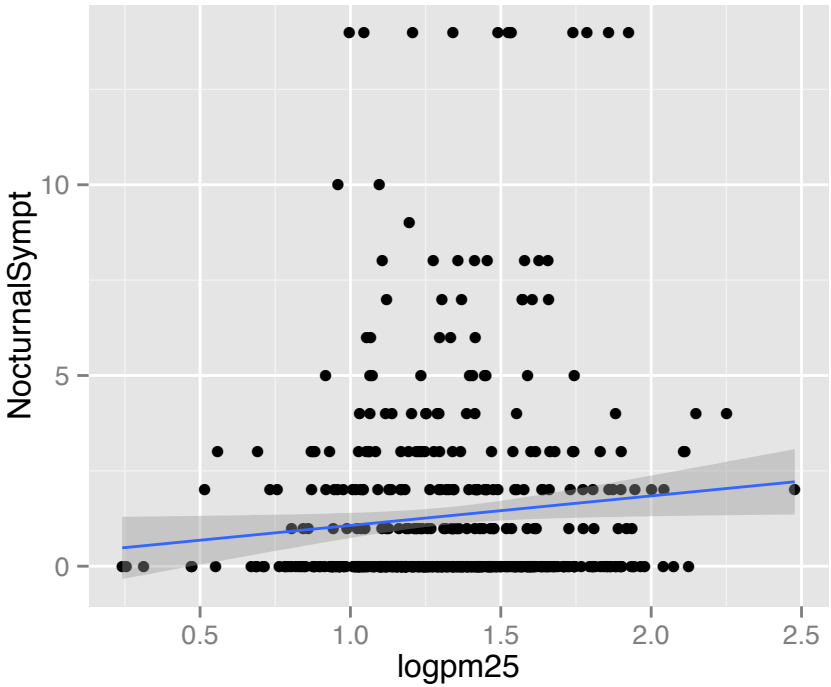



```
g <- ggplot(maacs, aes(logpm25, NocturnalSympt))
g + geom_point()
```

Adding More Layers: Smooth



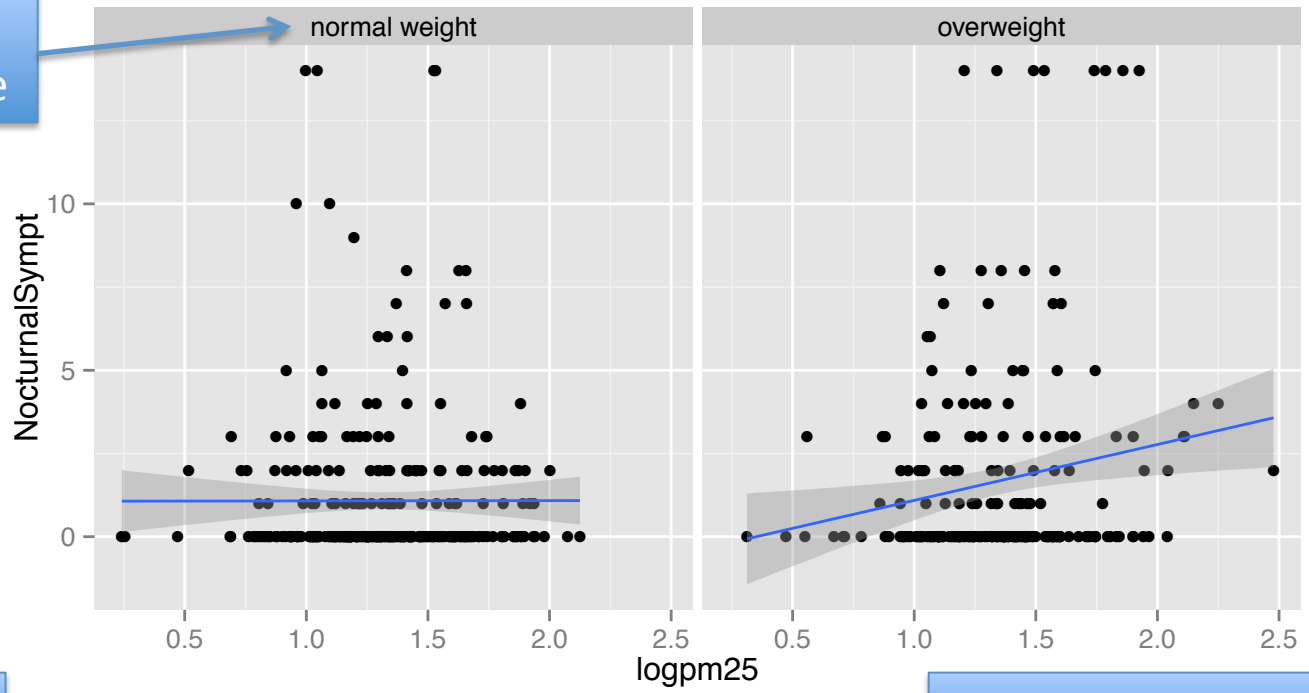

```
g + geom_point() + geom_smooth()
```




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

Adding More Layers: Facets

Labels from facet variable



Add facets

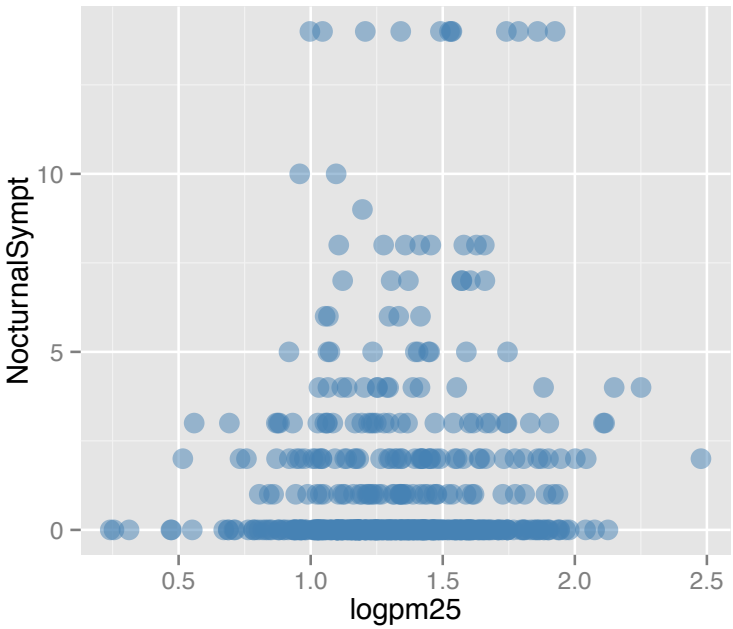
Faceting (factor) variable

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

Annotation

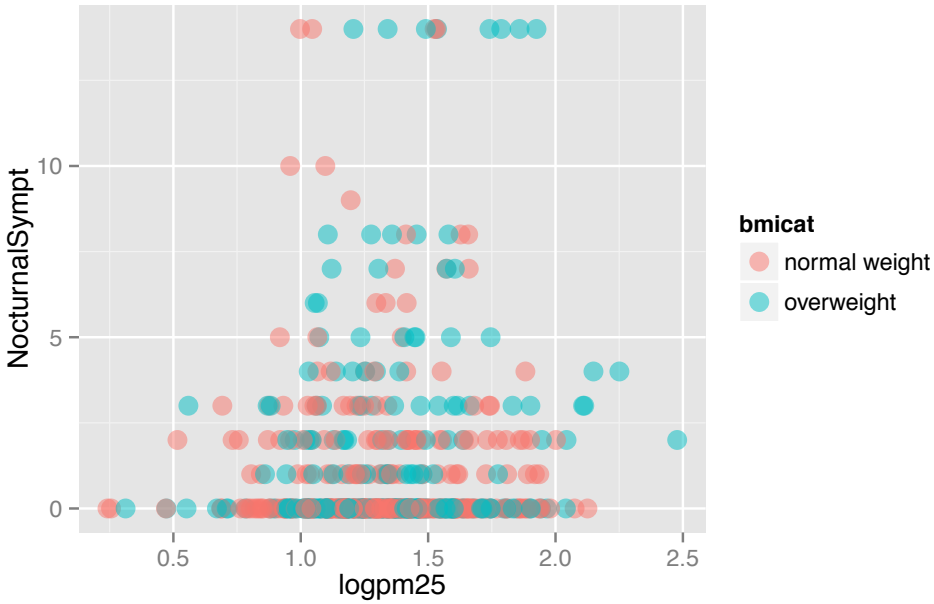
- Labels: xlab(), ylab(), labs(), ggtitle()
- Each of the “geom” functions has options to modify
- For things that only make sense globally, use theme()
  - Example: theme(legend.position = "none")
- Two standard appearance themes are included
  - theme_gray(): The default theme (gray background)
  - theme_bw(): More stark/plain

Modifying Aesthetics




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

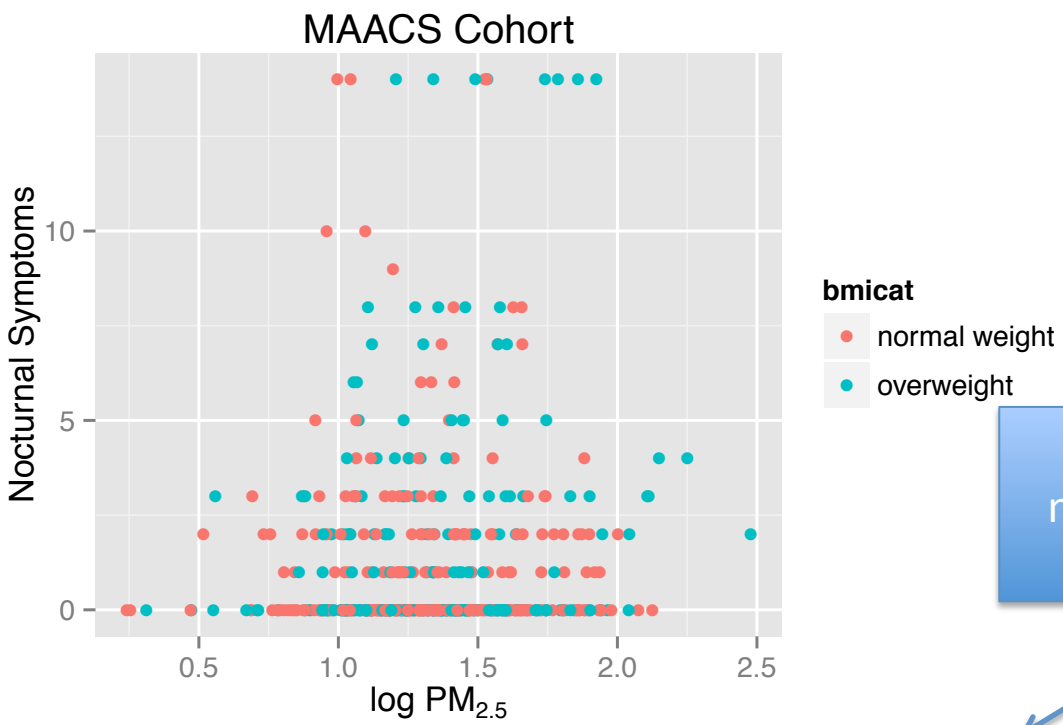
Constant values




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

Data variable

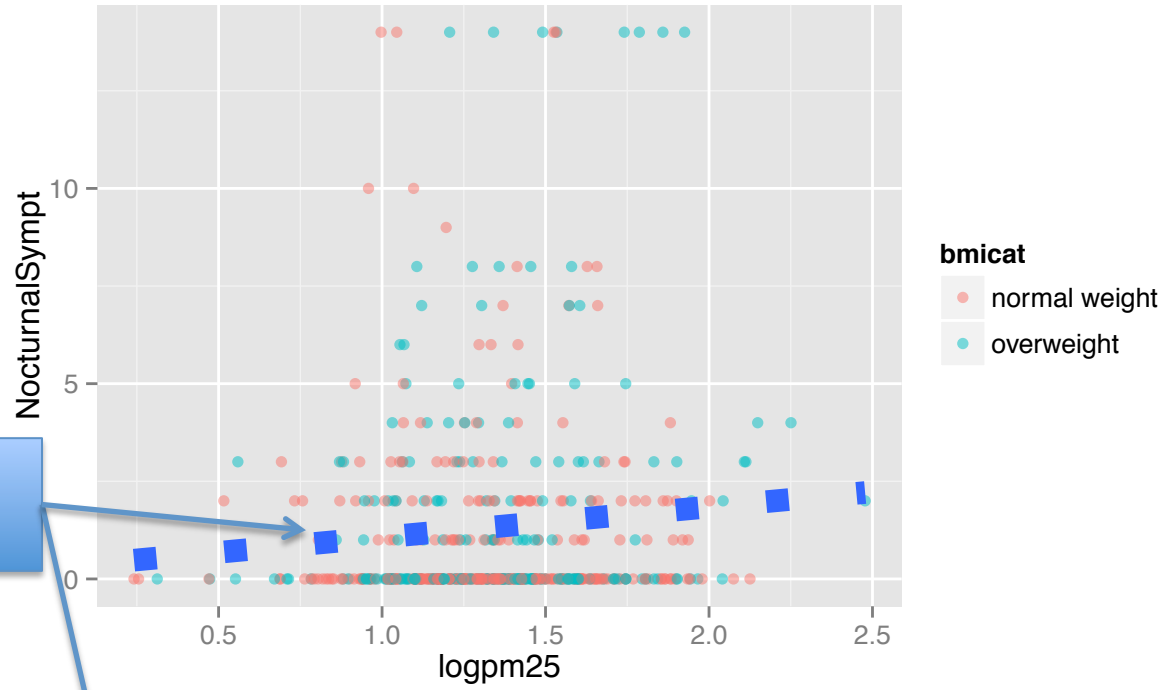
Modifying Labels



labs() function for modifying titles and x-, y-axis labels

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

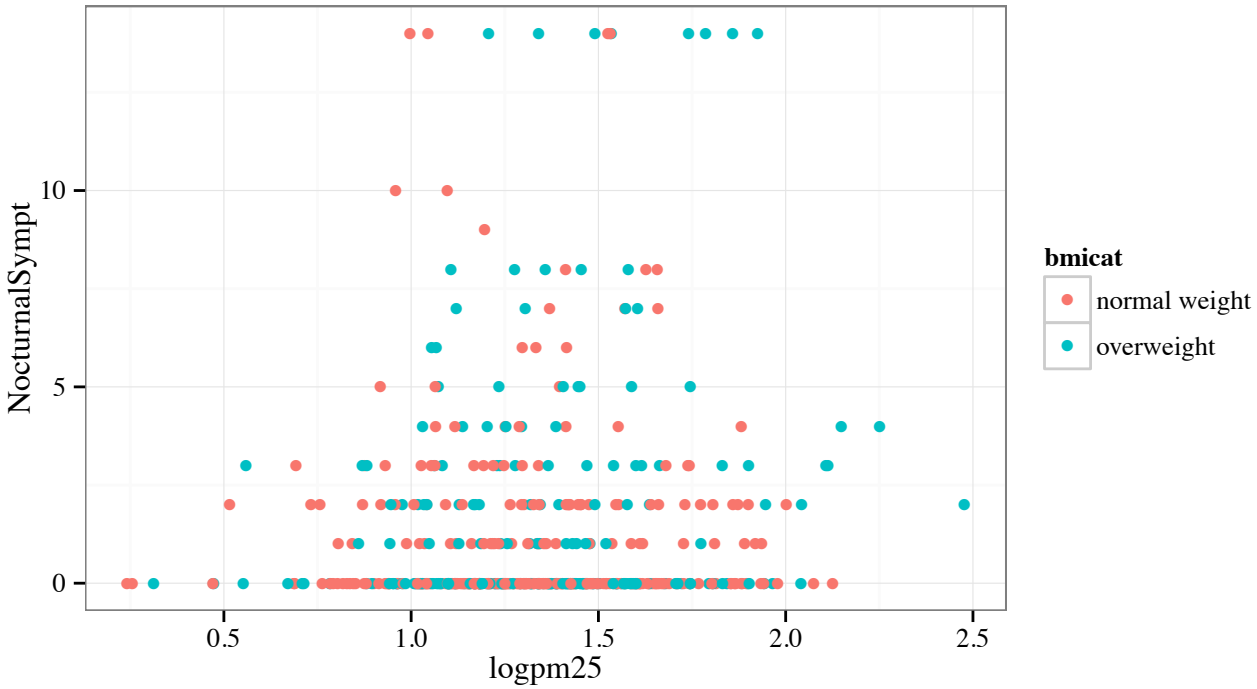
Customizing the Smooth



Modified smoother

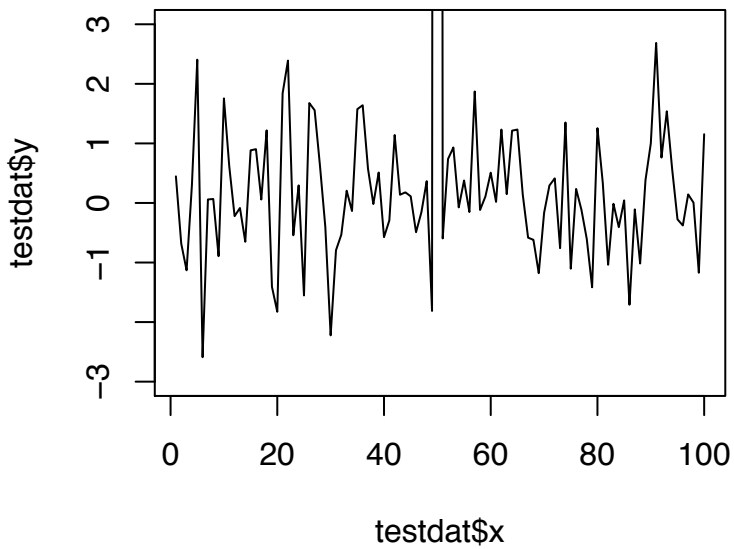
```
g + geom_point(aes(color = bmicat), size = 2, alpha = 1/2) +
geom_smooth(size = 4, linetype = 3, method = "lm", se = FALSE)
```

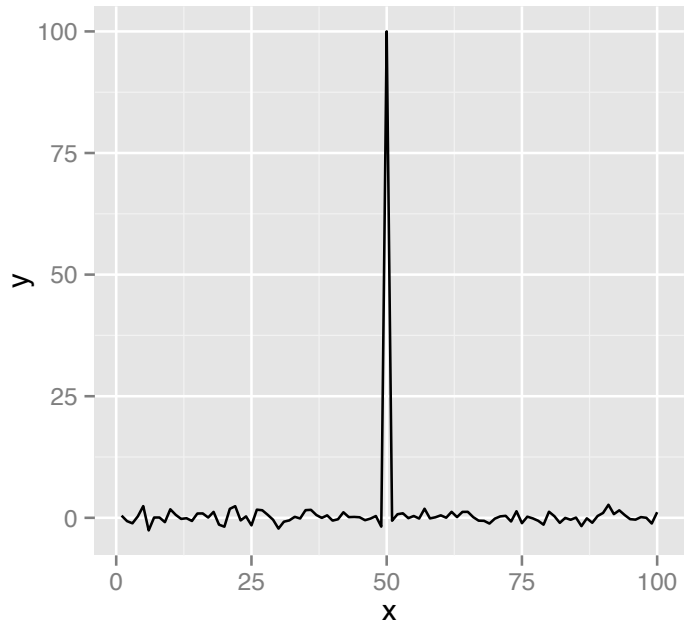
Changing the Theme




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

A Notes about Axis Limits



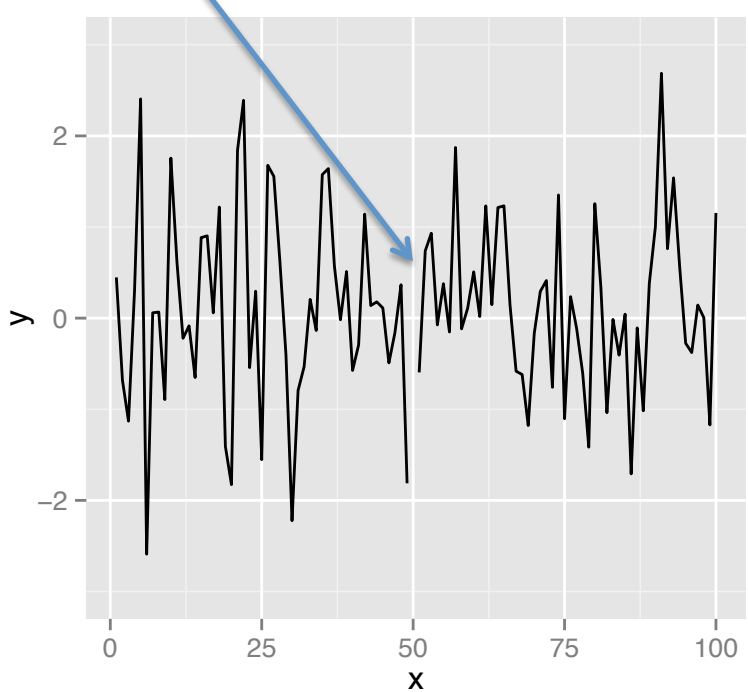



```
testdat <- data.frame(x = 1:100, y = rnorm(100))
testdat[50,2] <- 100  ## Outlier!
plot(testdat$x, testdat$y, type = "l", ylim = c(-3,3))
```

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

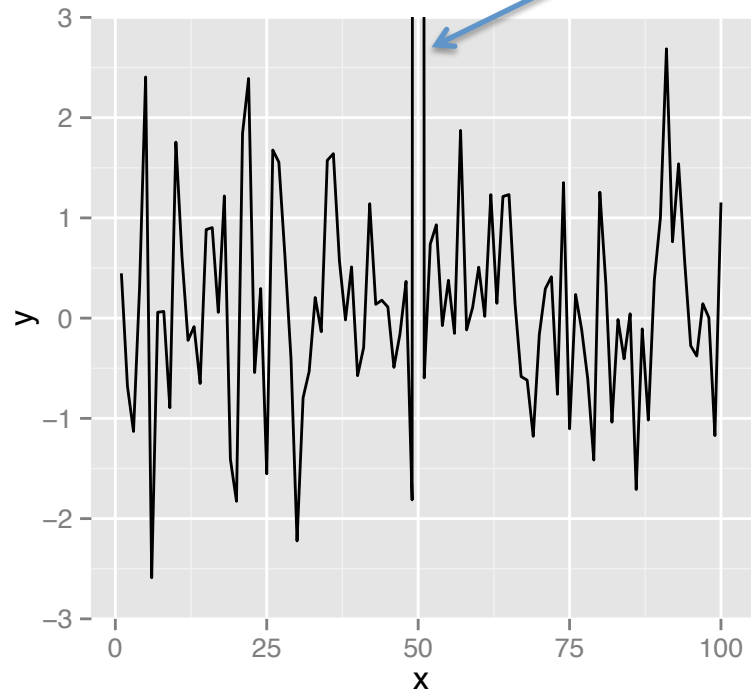
Axis Limits

Outlier missing




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

Outlier included




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

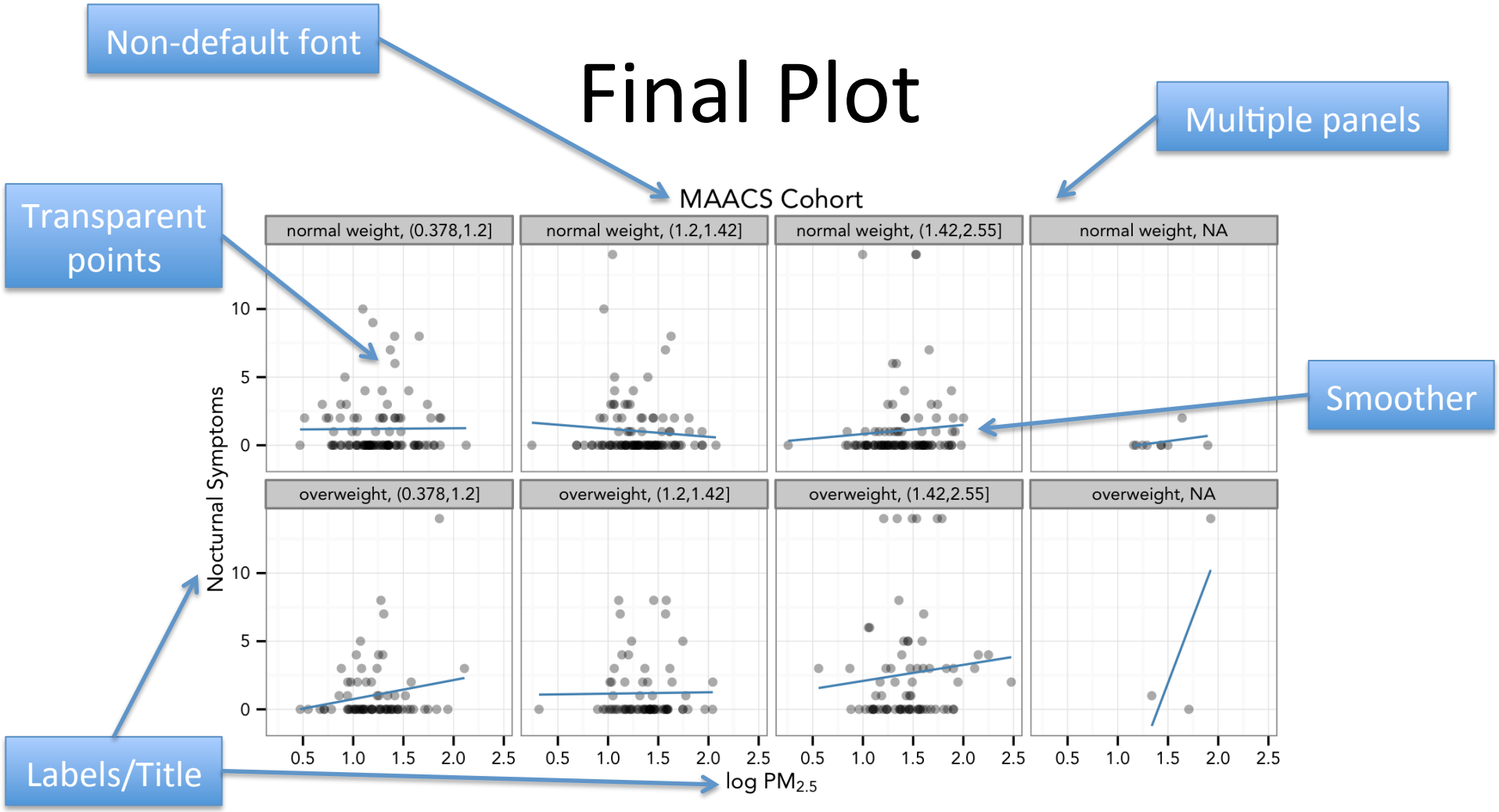
More Complex Example

- How does the relationship between $PM_{2.5}$ and nocturnal symptoms vary by BMI and NO_2?
- Unlike our previous BMI variable, NO_2 is continuous
- We need to make NO2 categorical so we can condition on it in the plotting
  - Use the cut() function for this

Making NO_2 Deciles

```
## Calculate the deciles of the data
> cutpoints <- quantile(maacs$logno2_new, seq(0, 1, length = 11), na.rm = TRUE)

## Cut the data at the deciles and create a new factor variable
> maacs$no2dec <- cut(maacs$logno2_new, cutpoints)

## See the levels of the newly created factor variable
> levels(maacs$no2dec)
 [1] "(0.378,0.969]" "(0.969,1.1]"   "(1.1,1.17]"    "(1.17,1.26]"
 [5] "(1.26,1.32]"   "(1.32,1.38]"   "(1.38,1.44]"   "(1.44,1.54]"
 [9] "(1.54,1.69]"   "(1.69,2.55]"
```

Final Plot


Non-default font
Multiple panels
Transparent points
Smoother
Labels/Title
MAACS Cohort
normal weight, (0.378,1.2]
normal weight, (1.2,1.42]
normal weight, (1.42,2.55]
normal weight, NA
overweight, (0.378,1.2]
overweight, (1.2,1.42]
overweight, (1.42,2.55]
overweight, NA
Nocturnal Symptoms
0
5
10
0.5
1.0
1.5
2.0
2.5
log PM2.5

Code for Final Plot

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

## Add layers
g + geom_point(alpha = 1/3)
  + facet_wrap(bmicat ~ no2dec, nrow = 2, ncol = 4)
  + geom_smooth(method="lm", se=FALSE, col="steelblue")
  + theme_bw(base_family = "Avenir", base_size = 10)
  + labs(x = expression("log " * PM[2.5])
  + labs(y = "Nocturnal Symptoms”)
  + labs(title = "MAACS Cohort”)
```

Add points

Make panels

Add smoother

Change theme

Add labels

Summary

- ggplot2 is very powerful and flexible if you learn the “grammar” and the various elements that can be tuned/modified
- Many more types of plots can be made; explore and mess around with the package (references mentioned in Part 1 are useful)