EXPLORATORY DATA ANALYSIS in R

21.

PRINCIPLES OF ANALYTIC GRAPHICS

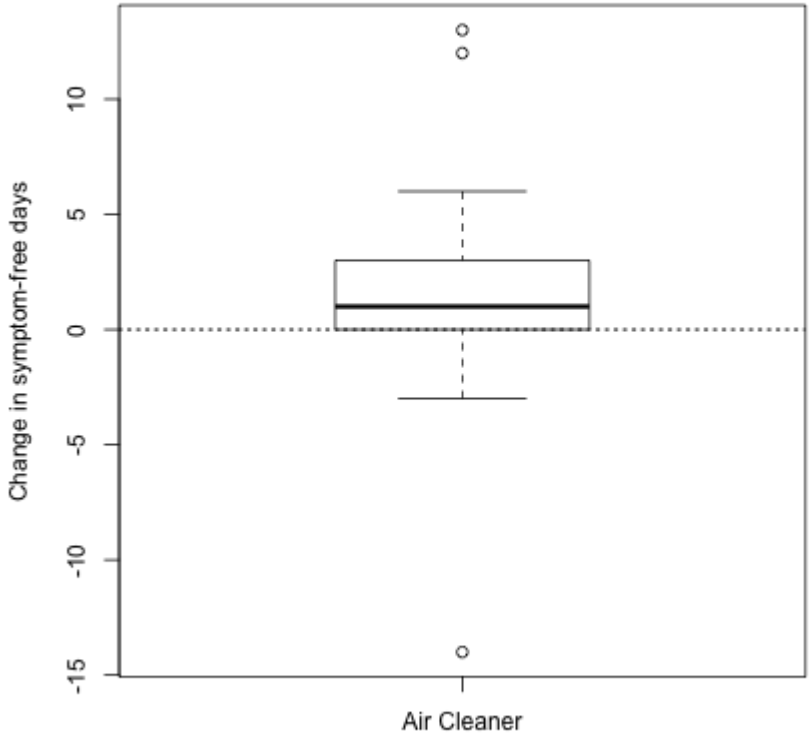
PRINCIPLES OF ANALYTIC GRAPHICS

Principle 1: Show comparisons

- Evidence for a hypothesis is always *relative* to another competing hypothesis.
- Always ask "Compared to What?"

PRINCIPLES OF ANALYTIC GRAPHICS

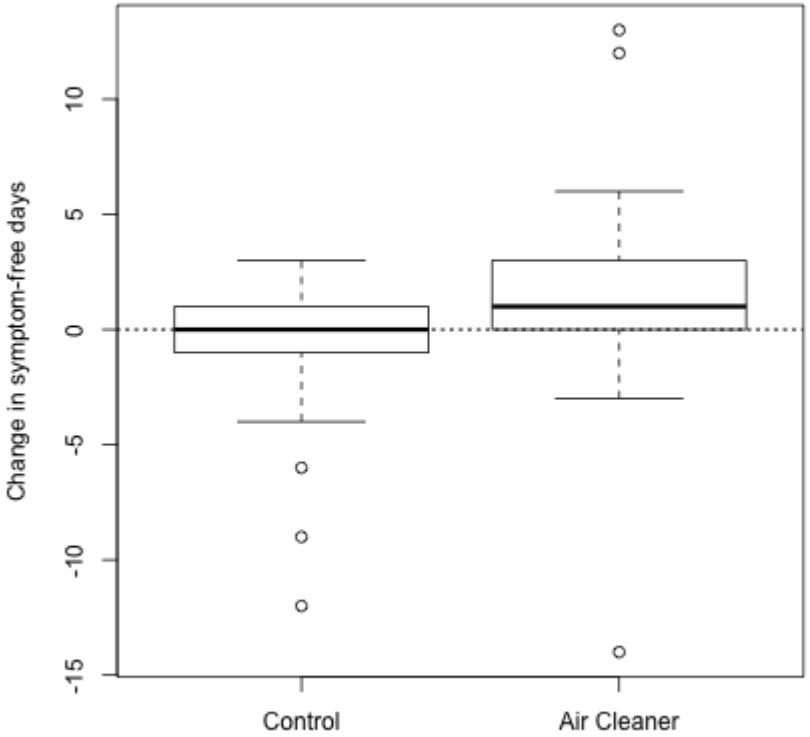
Principle 1: Show comparisons



Reference: Butz AM, *et al.*, *JAMA Pediatrics*, 2011.

PRINCIPLES OF ANALYTIC GRAPHICS

Principle 1: Show comparisons



Reference: Butz AM, *et al.*, *JAMA Pediatrics*, 2011.

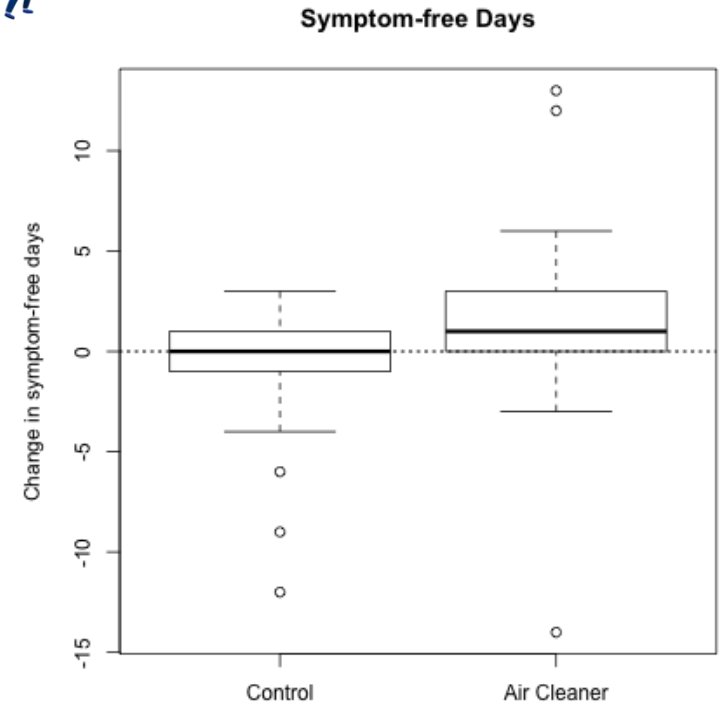
PRINCIPLES OF ANALYTIC GRAPHICS

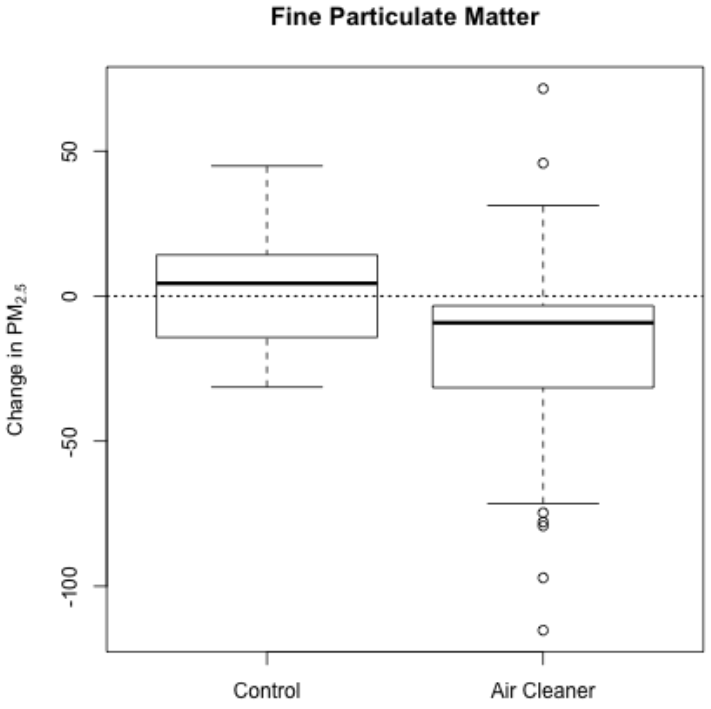
Principle 2: Show causality, mechanism, explanation, systematic structure

- What is your causal framework for thinking about a question?

PRINCIPLES OF ANALYTIC GRAPHICS

Principle 2: Show Causality, mechanism





Reference: Butz AM, *et al.*, *JAMA Pediatrics*, 2011.

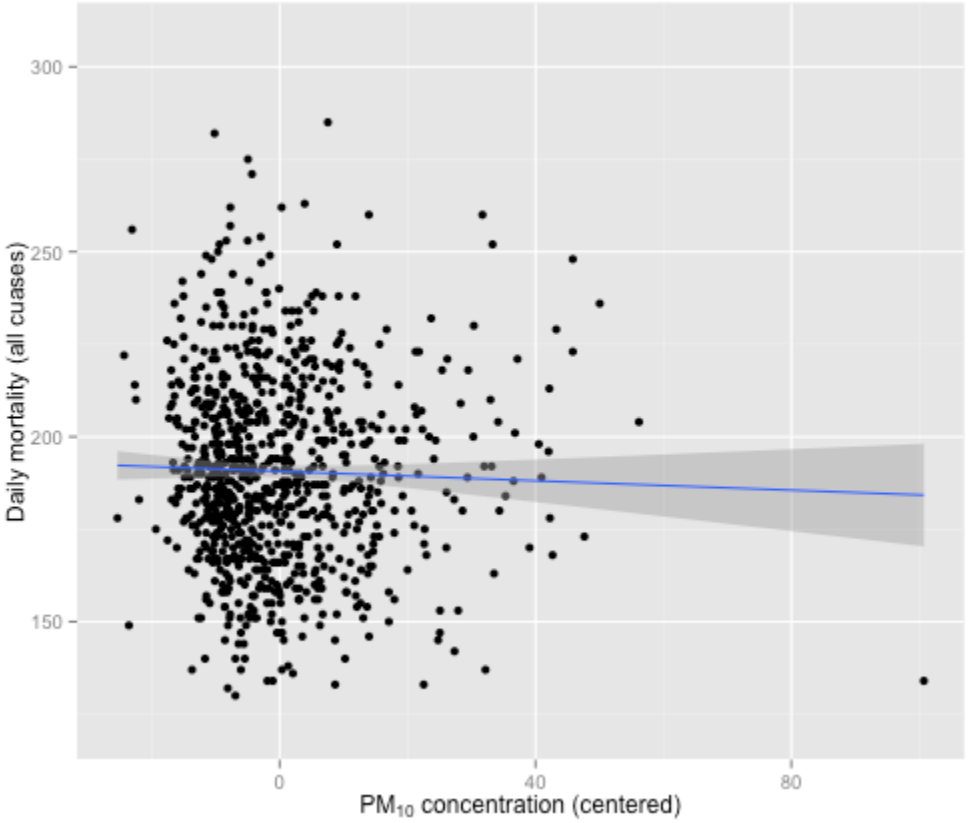
PRINCIPLES OF ANALYTIC GRAPHICS

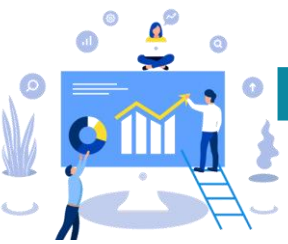
Principle 3: Show multivariate data

- Multivariate = more than 2 variables
- The real world is multivariate
- Need to "escape flatland"

PRINCIPLES OF ANALYTIC GRAPHICS

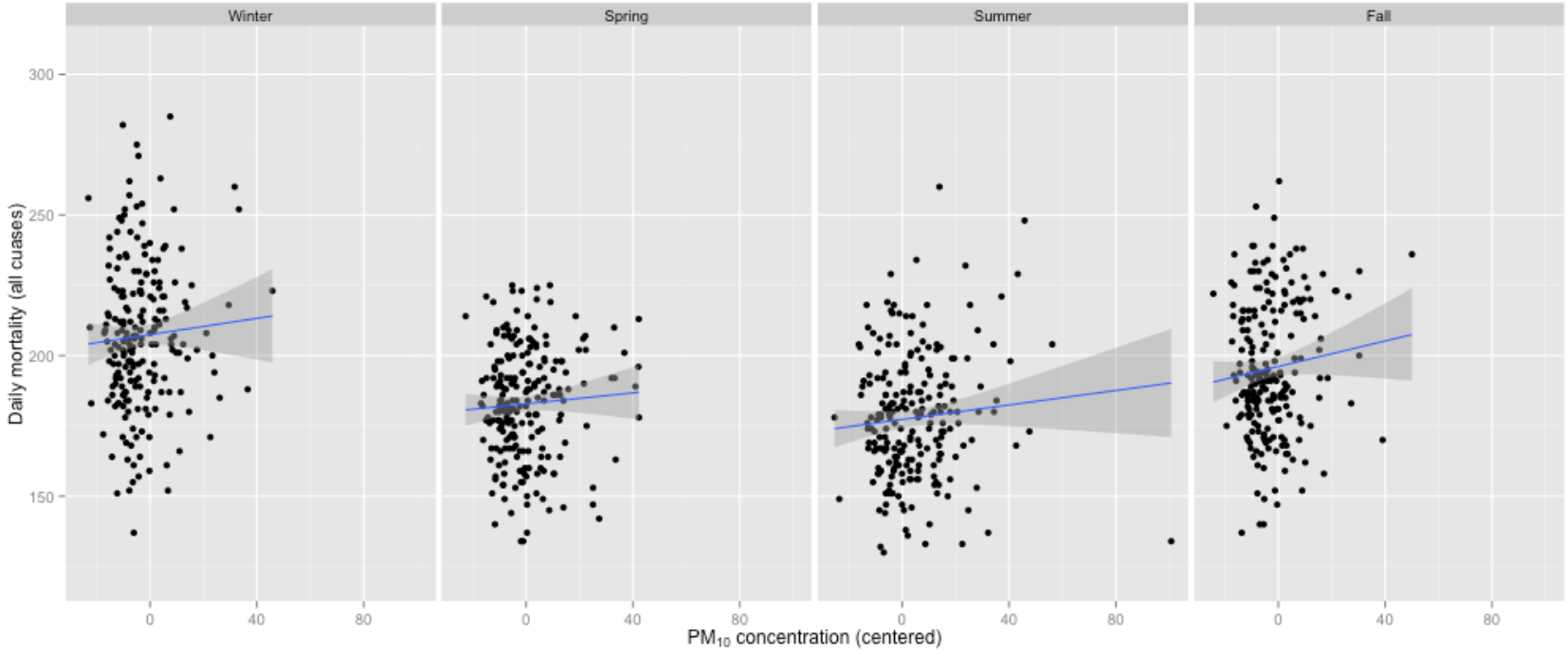
Principle 3: Show multivariate data





PRINCIPLES OF ANALYTIC GRAPHICS

Principle 3: Show multivariate data



PRINCIPLES OF ANALYTIC GRAPHICS

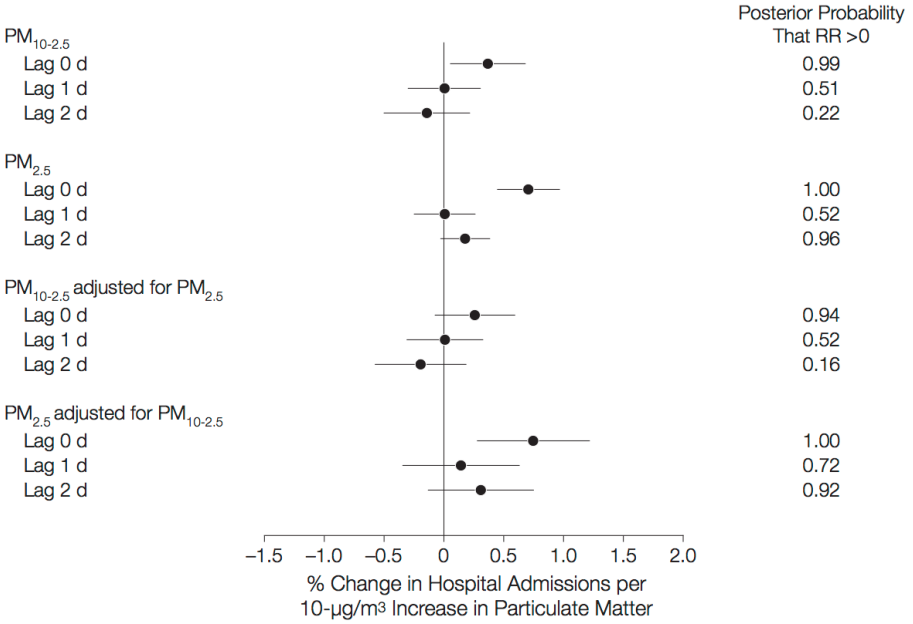
Principle 4: Integration of evidence

- Completely integrate words, numbers, images, diagrams
- Data graphics should make use of many modes of data presentation
- Don't let the tool drive the analysis

PRINCIPLES OF ANALYTIC GRAPHICS

Principle 4: Integrate Different Modes of Evidence

Figure 2. Percentage Change in Emergency Hospital Admissions Rate for Cardiovascular Diseases per a 10-µg/m³ Increase in Particulate Matter



Estimates are on average across 108 counties. $PM_{2.5}$ indicates particulate matter is 2.5 µm or less in aerodynamic diameter; PM_{10}, particulate matter is 10 µm or less in aerodynamic diameter; $PM_{10-2.5}$, particulate matter is greater than 2.5 µm and 10 µm or less in aerodynamic diameter; RR, relative risk. Error bars indicate 95% posterior intervals.

PRINCIPLES OF ANALYTIC GRAPHICS

Principle 5: Describe and document the evidence with appropriate labels, scales, sources, etc.

- A data graphic should tell a complete story that is credible

Principle 6: Content is King

- Analytical presentations ultimately stand or fall depending on the quality, relevance, and integrity of their content

Reference

Edward Tufte (2006). *Beautiful Evidence*, Graphics Press LLC.
www.edwardtufte.com

22.

EXPLORATORY GRAPHS

WHY DO WE USE GRAPHS IN DATA ANALYSIS?

- To understand data properties
- To find patterns in data
- To suggest modeling strategies
- To "debug" analyses
- To communicate results

CHARACTERISTICS OF EXPLORATORY GRAPHS

- They are made quickly
- A large number are made
- The goal is for personal understanding
- Axes/legends are generally cleaned up
- Color/size are primarily used for information

AIR POLLUTION

- The U.S. Environmental Protection Agency (EPA) sets national ambient air quality standards for outdoor air pollution.
- For fine particle pollution (PM2.5), the "annual mean, averaged over 3 years" cannot exceed 12 $\mu g/m^3$
- **Question**: Are there any counties in the U.S. that exceed that national standard for fine particle pollution?

DATA

- Annual average PM2.5 averaged over the period 2008-2010

```
pollution <- read.csv("avgpm25.csv",
                      colClasses = c("numeric", "character",
                                     "factor", "numeric", "numeric"))
```

```
> head(pollution)
       pm25  fips region longitude latitude
1  9.771185 01003   east -87.74826 30.59278
2  9.993817 01027   east -85.84286 33.26581
3 10.688618 01033   east -87.72596 34.73148
4 11.337424 01049   east -85.79892 34.45913
5 12.119764 01055   east -86.03212 34.01860
6 10.827805 01069   east -85.35039 31.18973
```

SIMPLE SUMMARIES OF DATA

One dimension

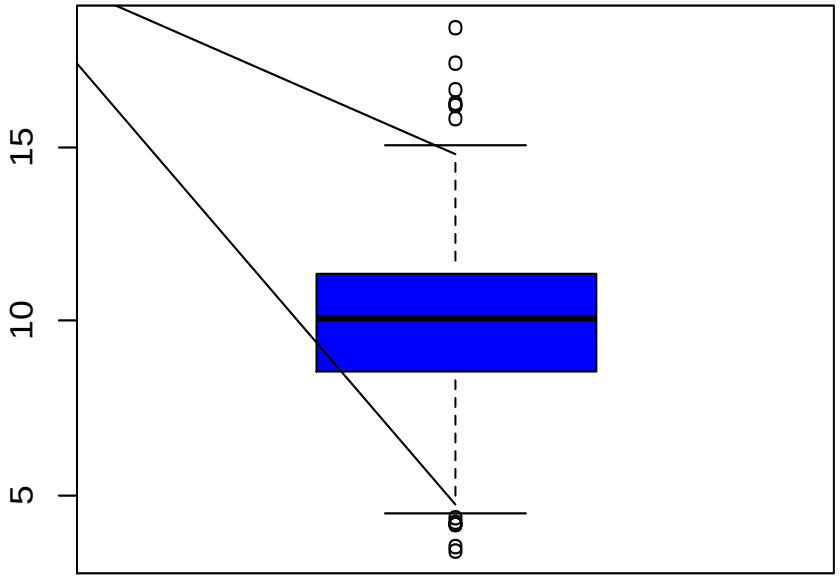
- Five-number summary
- Boxplots
- Histograms
- Density plot
- Bargraph

Five-Number Summary

```
> summary(pollution$pm25)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.383   8.549  10.047   9.836  11.356  18.441
```

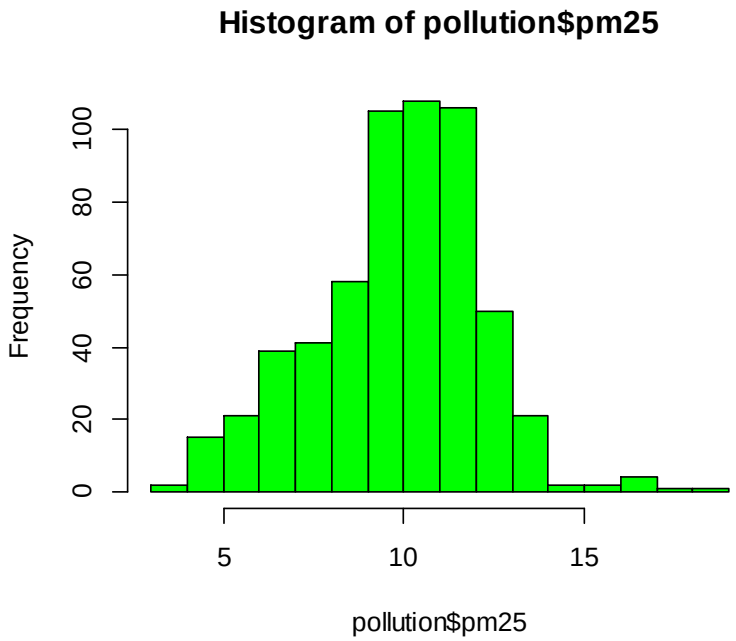
BOXPLOT

```
> boxplot(pollution$pm25, col = "blue")
```



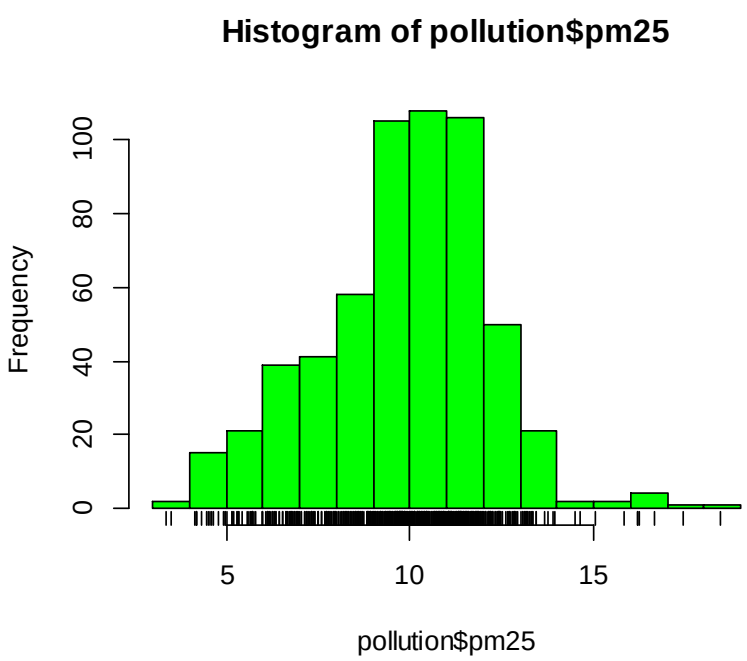
HISTOGRAM

```
> hist(pollution$pm25, col = "green")
```



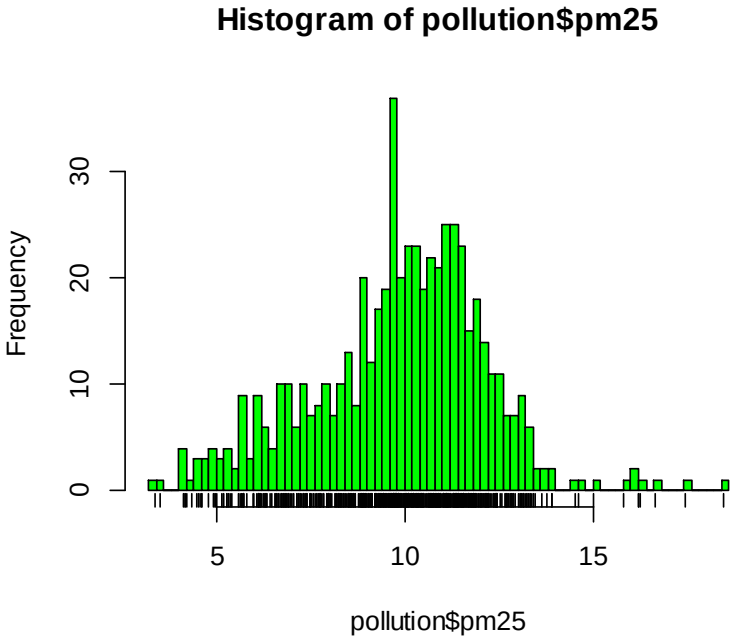
HISTOGRAM

```
> hist(pollution$pm25, col = "green")
> rug(pollution$pm25)
```



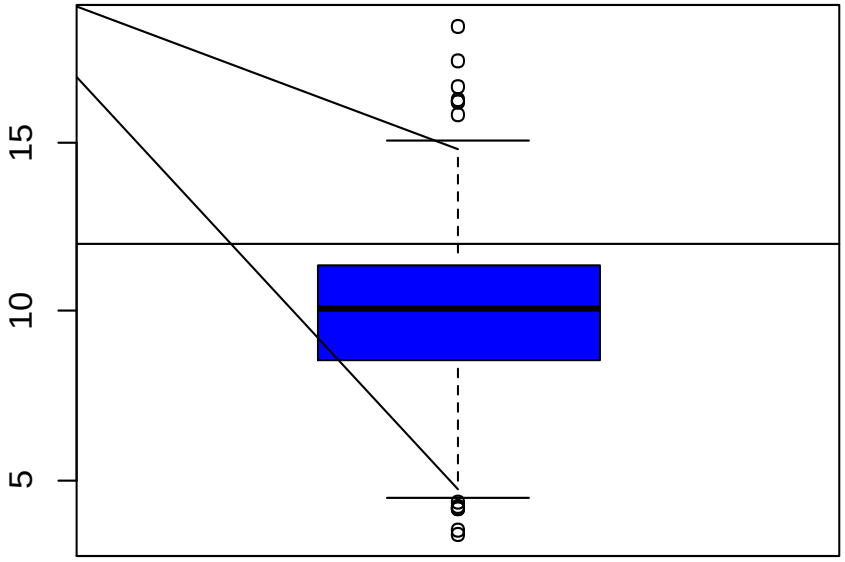
HISTOGRAM

```
> hist(pollution$pm25, col = "green", breaks = 100)
> rug(pollution$pm25)
```



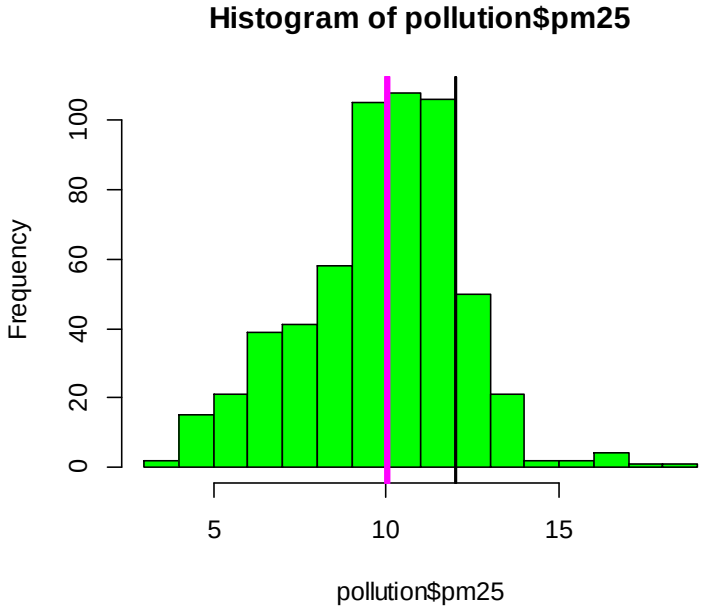
OVERLAYING FEATURES

```
> boxplot(pollution$pm25, col = "blue")
> abline(h = 12)
```



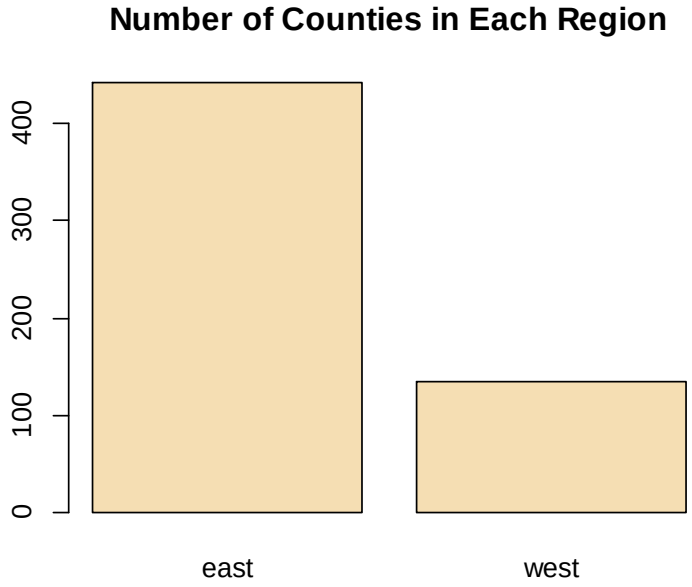
OVERLAYING FEATURES

```
> hist(pollution$pm25, col = "green")
> abline(v = 12, lwd = 2)
> abline(v = median(pollution$pm25), col = "magenta", lwd = 4)
```



BARGRAPH

```
> barplot(table(pollution$region), col = "wheat",
+         main = "Number of Counties in Each Region")
```



SIMPLE SUMMARIES OF DATA

Two dimensions

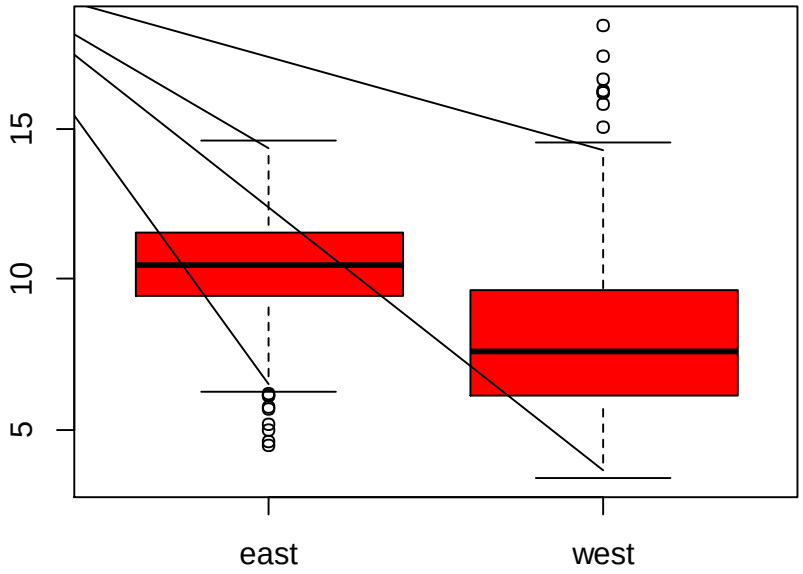
- Multiple/overlayed 1-D plots (`Lattice/ggplot2`)
- Scatterplots
- Smooth scatterplots

> 2 dimensions

- Overlayed/multiple 2-D plots; coplots
- Use color, size, shape to add dimensions
- Spinning plots
- Actual 3-D plots (not recommended)

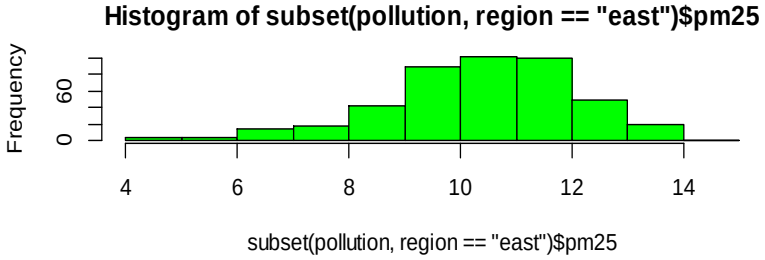
MULTIPLE BOXPLOTS

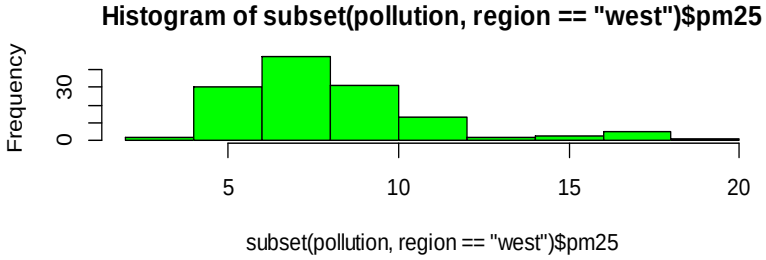
```
> boxplot(pm25 ~ region, data = pollution, col = "red")
```



MULTIPLE HISTOGRAMS

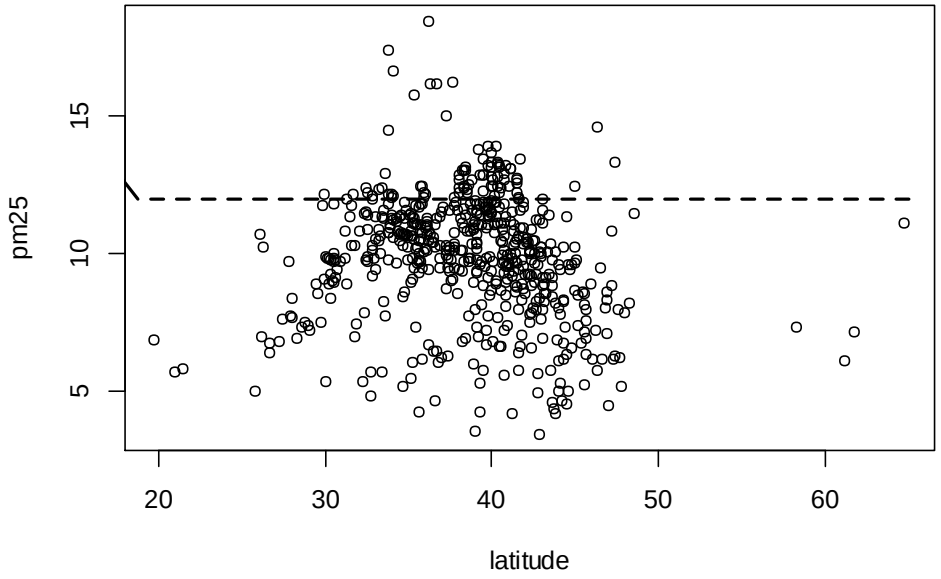
```
> par(mfrow = c(2, 1), mar = c(4, 4, 2, 1))
> hist(subset(pollution, region == "east")$pm25, col = "green")
> hist(subset(pollution, region == "west")$pm25, col = "green")
```





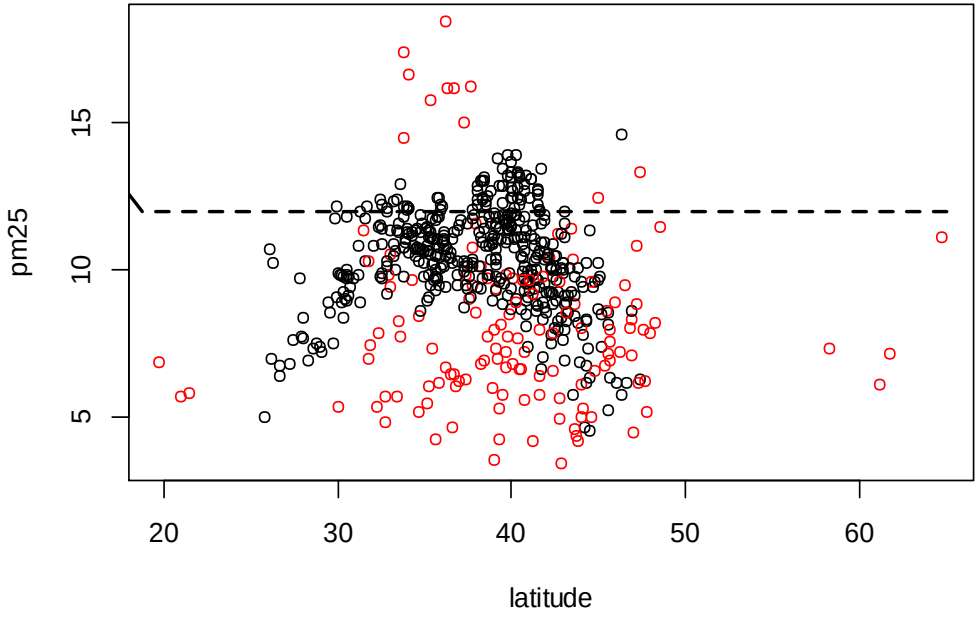
SCATTERPLOT

```
> with(pollution, plot(latitude, pm25))
> abline(h = 12, lwd = 2, lty = 2)
```



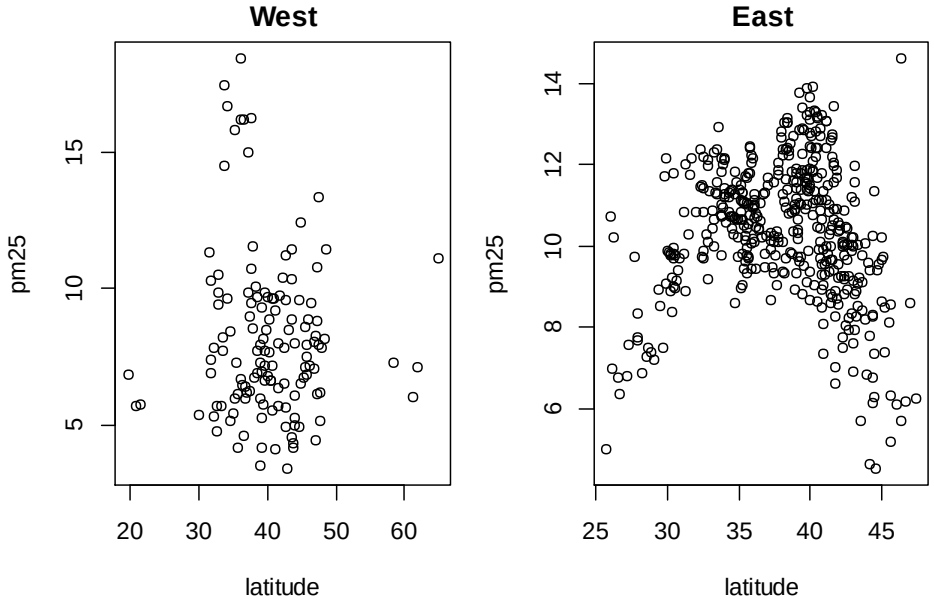
SCATTERPLOT – USING COLOR

```
> with(pollution, plot(latitude, pm25, col = region))
> abline(h = 12, lwd = 2, lty = 2)
```



MULTIPLE SCATTERPLOTS

```
> par(mfrow = c(1, 2), mar = c(5, 4, 2, 1))
> with(subset(pollution, region == "west"), plot(latitude, pm25, main = "West"))
> with(subset(pollution, region == "east"), plot(latitude, pm25, main = "East"))
```



COPYING PLOTS

```
with(faithful, plot(eruptions, waiting)) ## Create plot on screen device
title(main = "Old Faithful Geyser data") ## Add a main title
dev.copy(png, file = "geyserplot.png") ## Copy my plot to a PNG file
dev.off() ## Don't forget to close the PNG device!
```

- `dev.copy`: copy a plot from one device to another
- `dev.copy2pdf`: specifically copy a plot to a PDF file

NOTE: Copying a plot is not an exact operation, so the result may not be identical to the original.

23.

PLOTTING SYSTEMS IN R

PLOTTING SYSTEMS IN R

- The Base Plotting System
- The `Lattice` System
- The `ggplot2` System

THE BASE PLOTTING SYSTEM

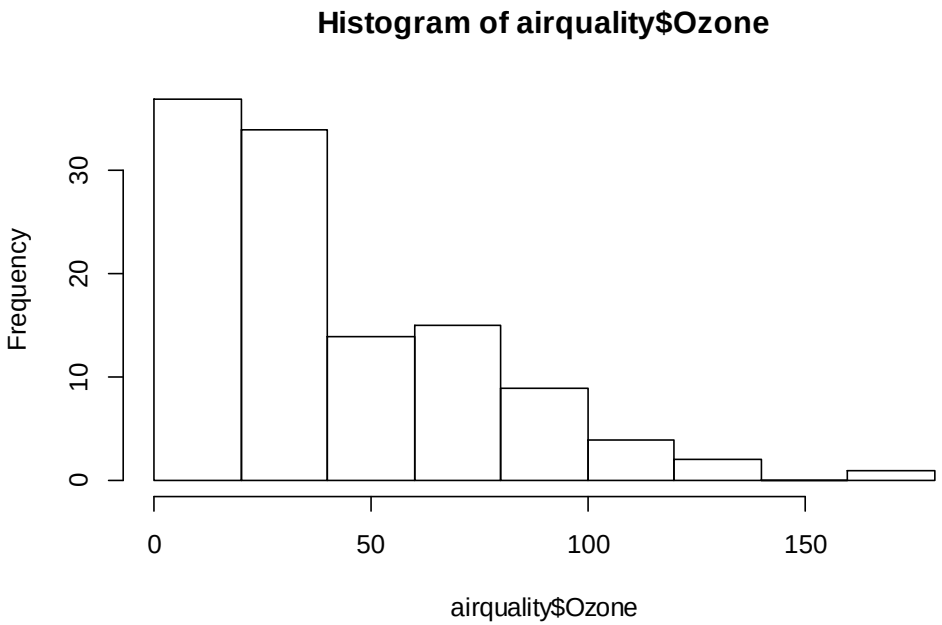
- "Artist's palette" model
- Start with blank canvas and build up from there
- Start with plot function (or similar)
- Use annotation functions to add/modify (`text, lines, points, axis`)

THE BASE PLOTTING SYSTEM

- Convenient, mirrors how we think of building plots and analyzing data
- Can’t go back once plot has started (i.e. to adjust margins); need to plan in advance
- Plot is just a series of R commands

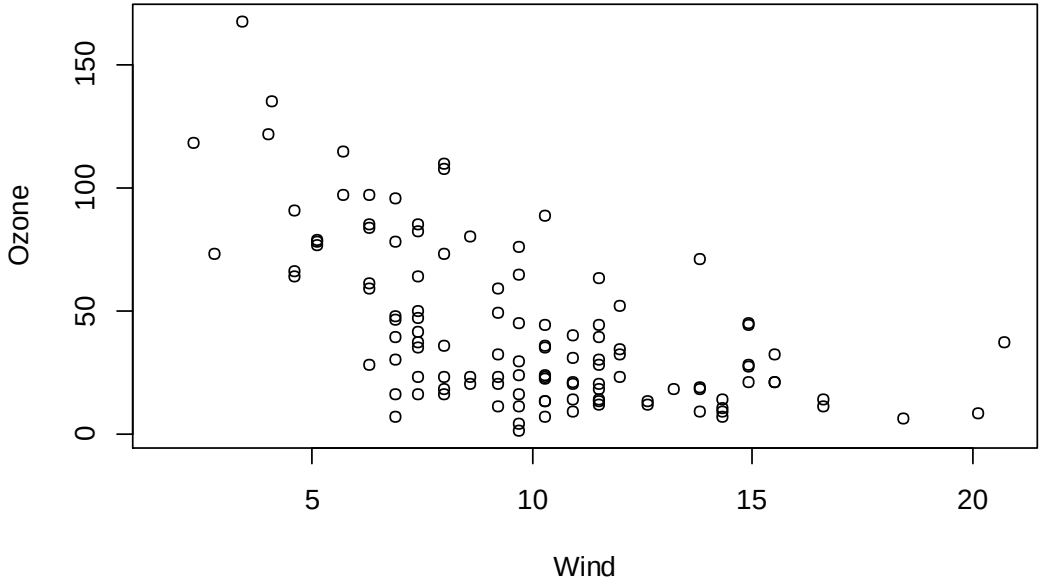
SIMPLE BASE GRAPHICS: Histogram

```
> hist(airquality$Ozone) ## Draw a new plot
```



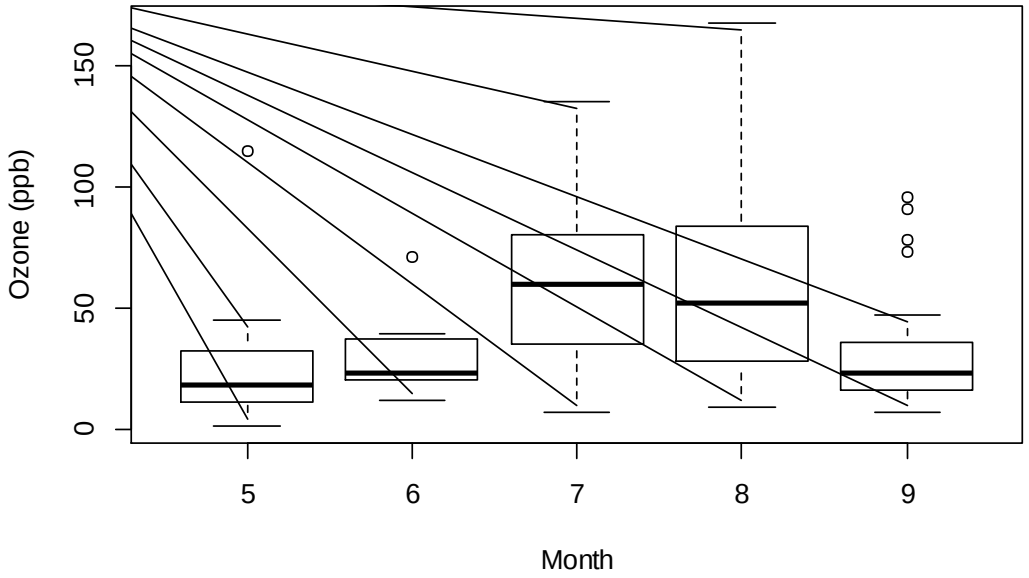
SIMPLE BASE GRAPHICS: Scatterplot

```
> with(airquality, plot(Wind, Ozone))
```



SIMPLE BASE GRAPHICS: Boxplot

```
airquality <- transform(airquality, Month = factor(Month))
boxplot(Ozone ~ Month, airquality,
        xlab = "Month", ylab = "Ozone (ppb)")
```



SOME IMPORTANT BASE GRAPHICS PARAMETERS

- `pch`: the plotting symbol (default is open circle)
- `lty`: the line type (default is solid line), can be dashed, dotted, etc.
- `lwd`: the line width, specified as an integer multiple
- `col`: the plotting color, specified as a number, string, or hex code; the `colors()` function gives you a vector of colors by name
- `xlab`: character string for the x-axis label
- `ylab`: character string for the y-axis label

SOME IMPORTANT BASE GRAPHICS PARAMETERS

The `par()` function is used to specify *global* graphics parameters that affect all plots in an R session. These parameters can be overridden when specified as arguments to specific plotting functions.

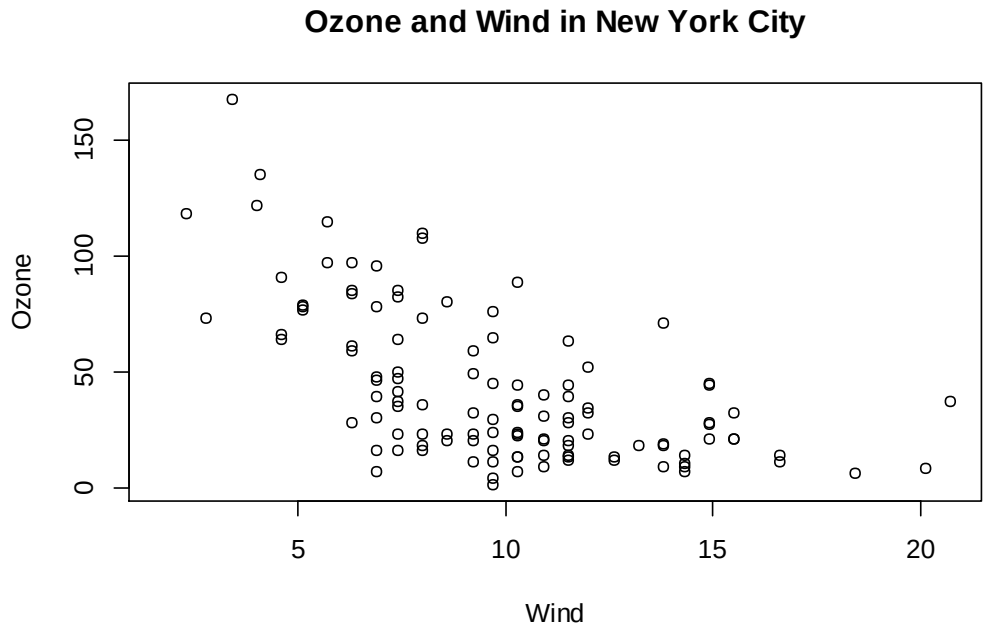
- `las`: the orientation of the axis labels on the plot
- `bg`: the background color
- `mar`: the margin size
- `oma`: the outer margin size (default is 0 for all sides)
- `mfrow`: number of plots per row, column (plots are filled row-wise)
- `mfcol`: number of plots per row, column (plots are filled column-wise)

BASE PLOTTING FUNCTIONS

- `plot`: make a scatterplot, or other type of plot depending on the class of the object being plotted
- `lines`: add lines to a plot, given a vector x values and a corresponding vector of y values (or a 2-column matrix); this function just connects the dots
- `points`: add points to a plot
- `text`: add text labels to a plot using specified x, y coordinates
- `title`: add annotations to x, y axis labels, title, subtitle, outer margin
- `mtext`: add arbitrary text to the margins (inner or outer) of the plot
- `axis`: adding axis ticks/labels

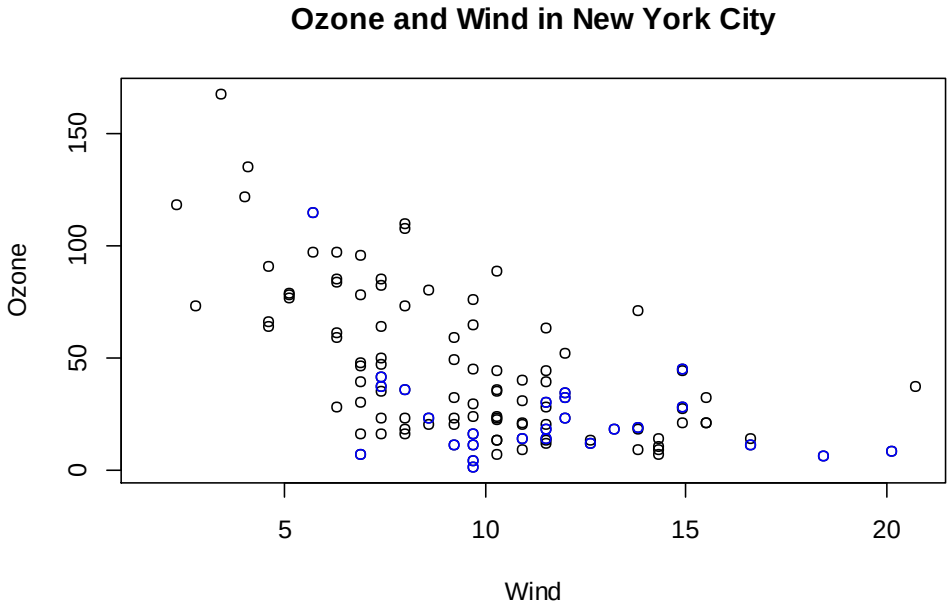
BASE PLOT WITH ANNOTATION

```
with(airquality, plot(Wind, Ozone))
title(main = "Ozone and Wind in New York City") ## Add a title
```



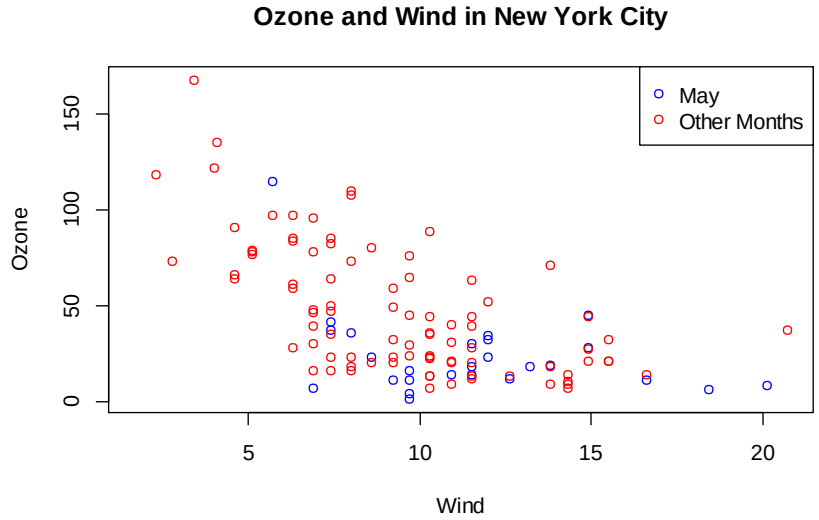
BASE PLOT WITH ANNOTATION

```
with(airquality, plot(Wind, Ozone,
                      main = "Ozone and Wind in New York City"))
with(subset(airquality, Month == 5), points(Wind, Ozone, col = "blue"))
```



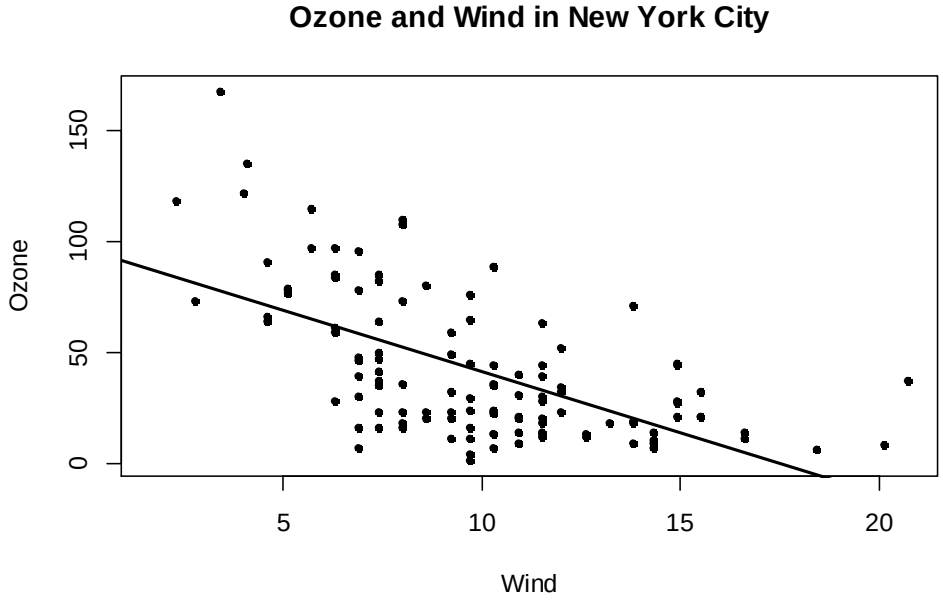
BASE PLOT WITH ANNOTATION

```
with(airquality, plot(Wind, Ozone, main = "Ozone and Wind in New York City",
                      type = "n"))
with(subset(airquality, Month == 5), points(Wind, Ozone, col = "blue"))
with(subset(airquality, Month != 5), points(Wind, Ozone, col = "red"))
legend("topright", pch = 1, col = c("blue", "red"),
       legend = c("May", "Other Months"))
```



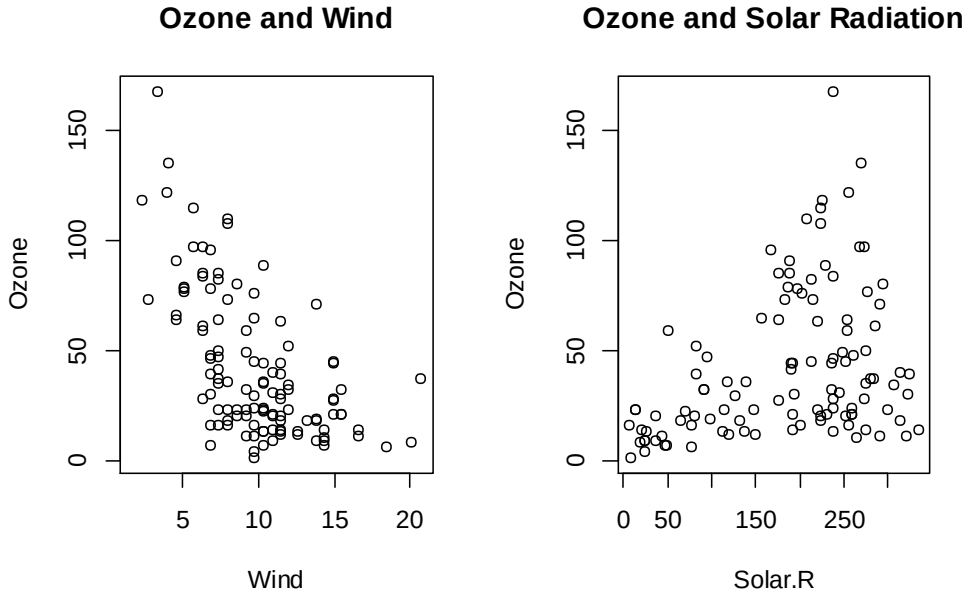
BASE PLOT WITH REGRESSION LINE

```
with(airquality, plot(Wind, Ozone, main = "Ozone and Wind in New York City",
                      pch = 20))
model <- lm(Ozone ~ Wind, airquality)
abline(model, lwd = 2)
```



MULTIPLE BASE PLOTS

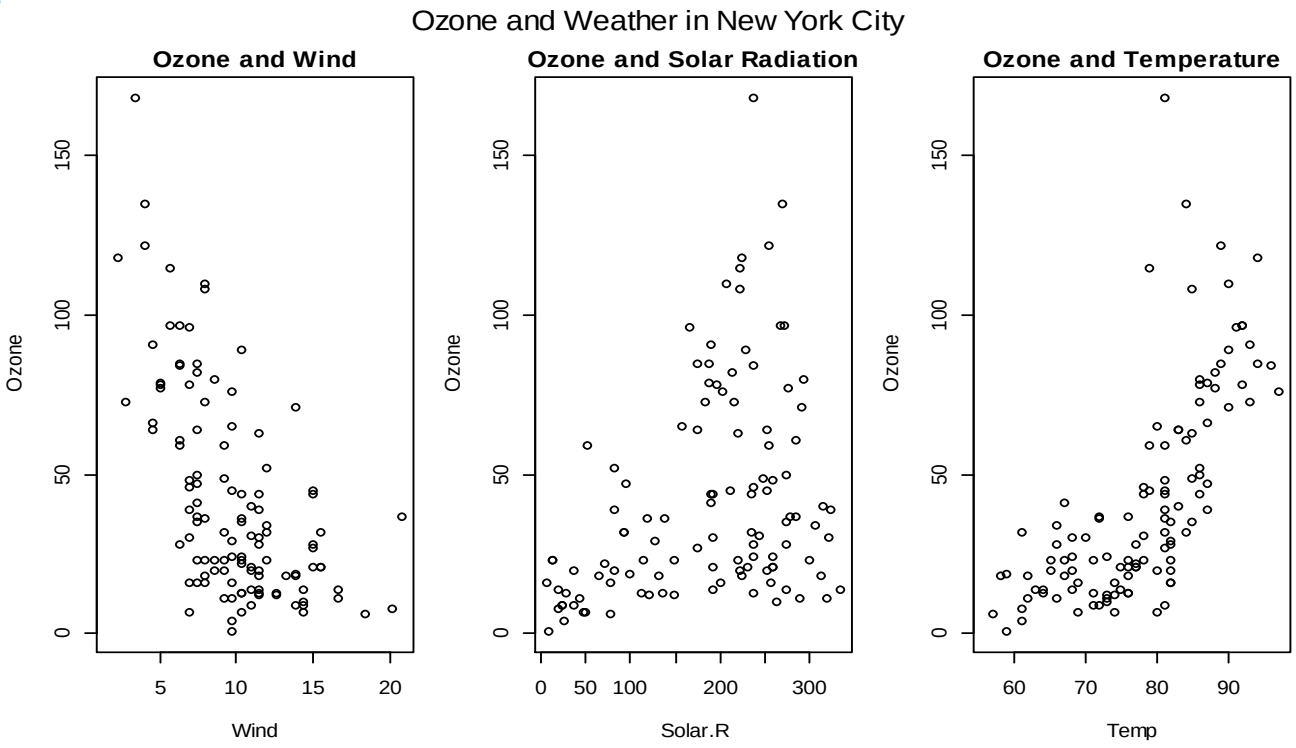
```
par(mfrow = c(1, 2))
with(airquality, {
    plot(Wind, Ozone, main = "Ozone and Wind")
    plot(Solar.R, Ozone, main = "Ozone and Solar Radiation")})
```



MULTIPLE BASE PLOTS

```
par(mfrow = c(1, 3), mar = c(4, 4, 2, 1), oma = c(0, 0, 2, 0))
with(airquality, {
    plot(Wind, Ozone, main = "Ozone and Wind")
    plot(Solar.R, Ozone, main = "Ozone and Solar Radiation")
    plot(Temp, Ozone, main = "Ozone and Temperature")
    mtext("Ozone and Weather in New York City", outer = TRUE)})
```

MULTIPLE BASE PLOTS



SUMMARY

- Plots in the base plotting system are created by calling successive R functions to "build up" a plot
- Plotting occurs in two stages:
  - Creation of a plot
  - Annotation of a plot (adding lines, points, text, legends)
- The base plotting system is very flexible and offers a high degree of control over plotting

THE LATTICE SYSTEM

- Plots are created with a single function call (`xyplot`, `bwplot`, etc.)
- Most useful for conditioning types of plots: Looking at how y changes with x across levels of z
- Things like margins/spacing set automatically because entire plot is specified at once
- Good for putting many plots on a screen

THE LATTICE SYSTEM

- Sometimes awkward to specify an entire plot in a single function call
- Annotation in plot is not especially intuitive
- Use of panel functions and subscripts difficult to wield and requires intense preparation
- Cannot "add" to the plot once it is created

THE LATTICE 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

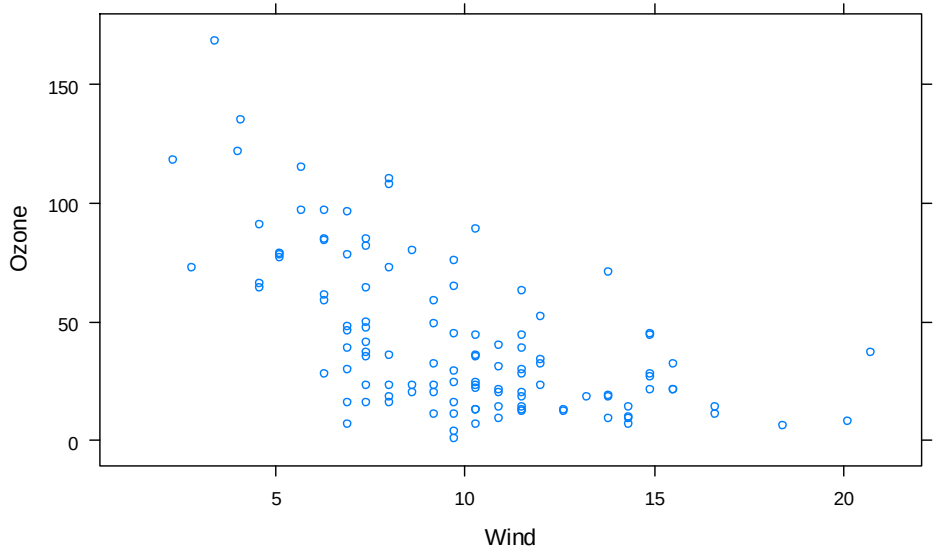
- The lattice plotting system does not have a "two-phase" aspect with separate plotting and annotation like in base plotting
- All plotting/annotation is done at once with a single function call

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

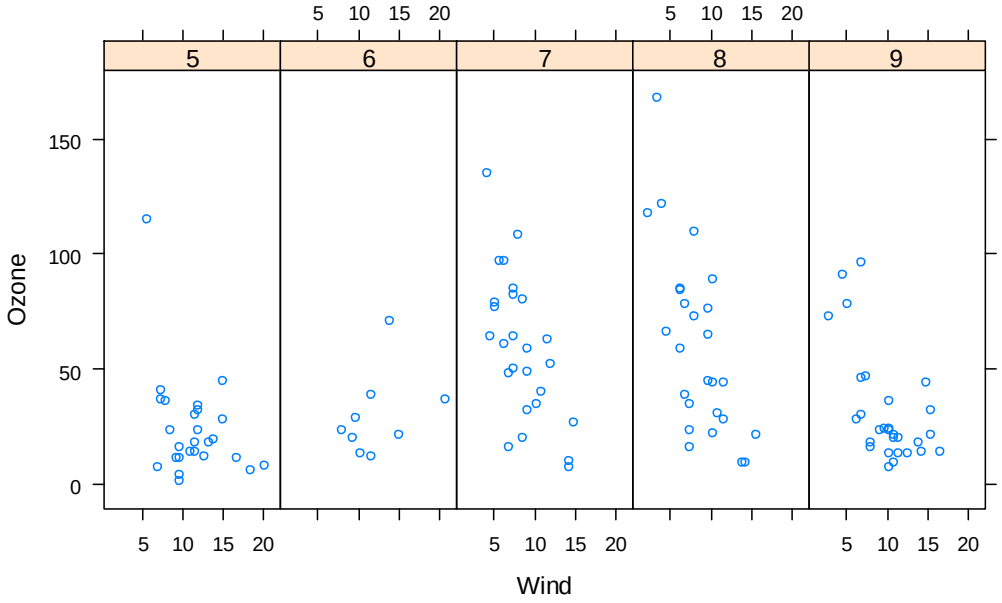
SIMPLE LATTICE PLOT

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



SIMPLE LATTICE PLOT

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



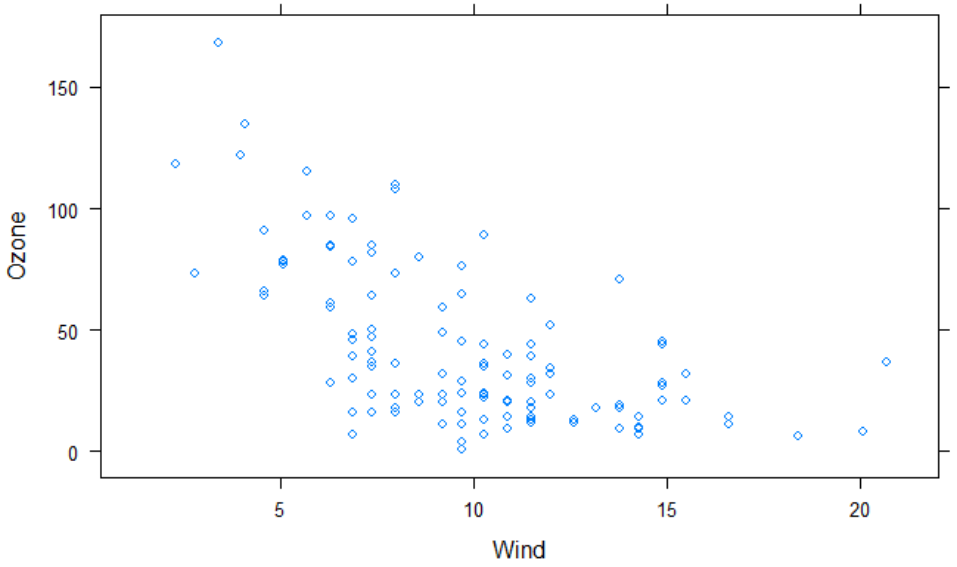
LATTICE BEHAVIOR

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

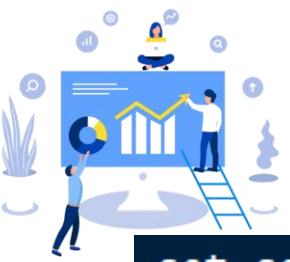
```
p <- xyplot(Ozone ~ Wind, data = airquality) ## Nothing happens!
print(p) ## Plot appears
```




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

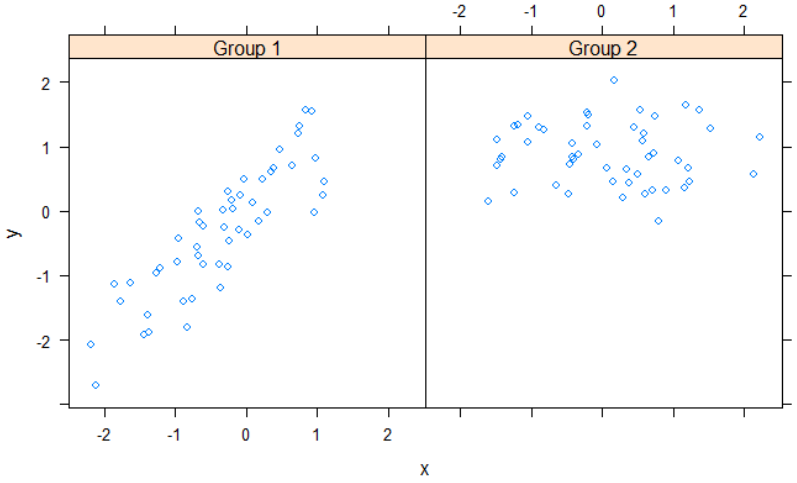
LATTICE PANEL FUNCTIONS

- 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)



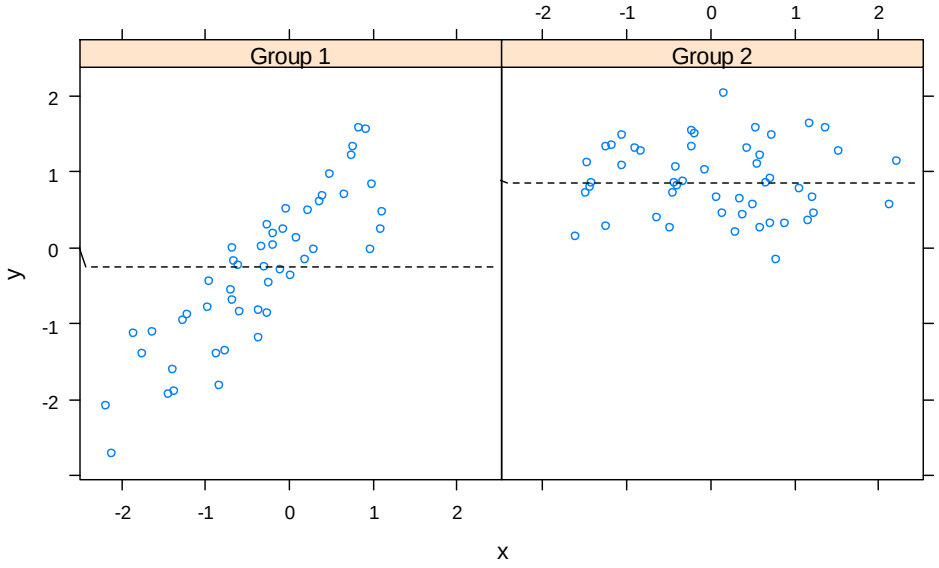
LATTICE PANEL FUNCTIONS

```
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
```



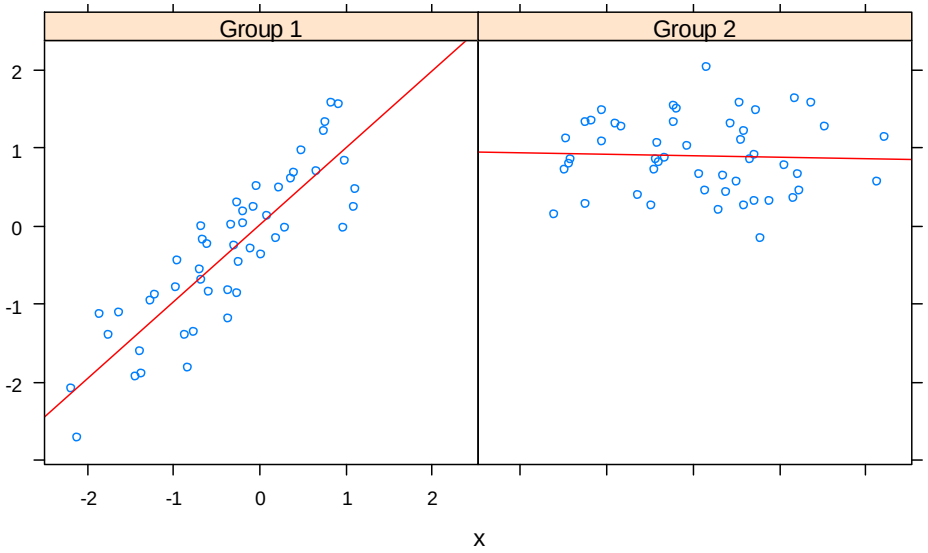
LATTICE PANEL FUNCTIONS

```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
})
```

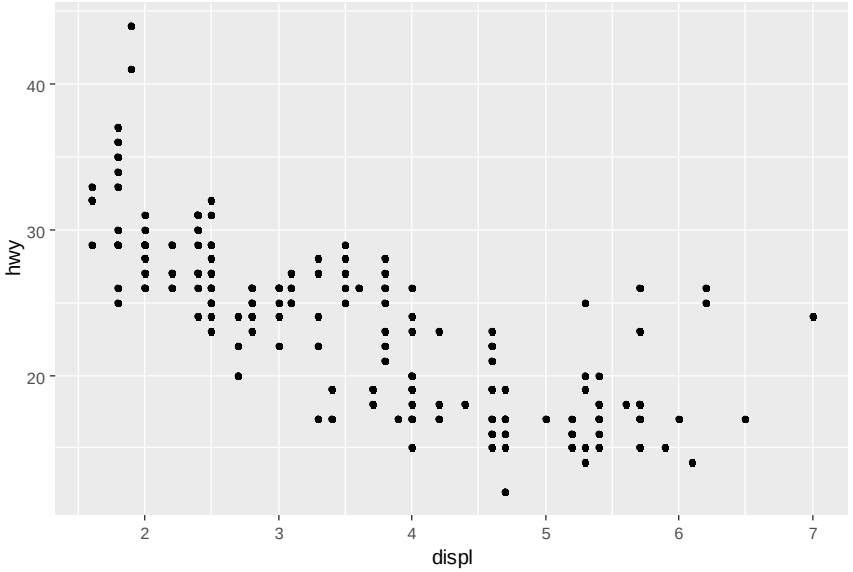


THE ggplot2 SYSTEM

- Splits the difference between base and lattice in a number of ways
- Automatically deals with spacings, text, titles but also allows you to annotate by "adding" to a plot
- Superficial similarity to lattice but generally easier/more intuitive to use
- Default mode makes many choices for you (but you can still customize to your heart's desire)

ggplot2 PLOT

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



REFERENCES

- Paul Murrell (2011). *R Graphics*, CRC Press.
- Hadley Wickham (2009). *ggplot2*, Springer.