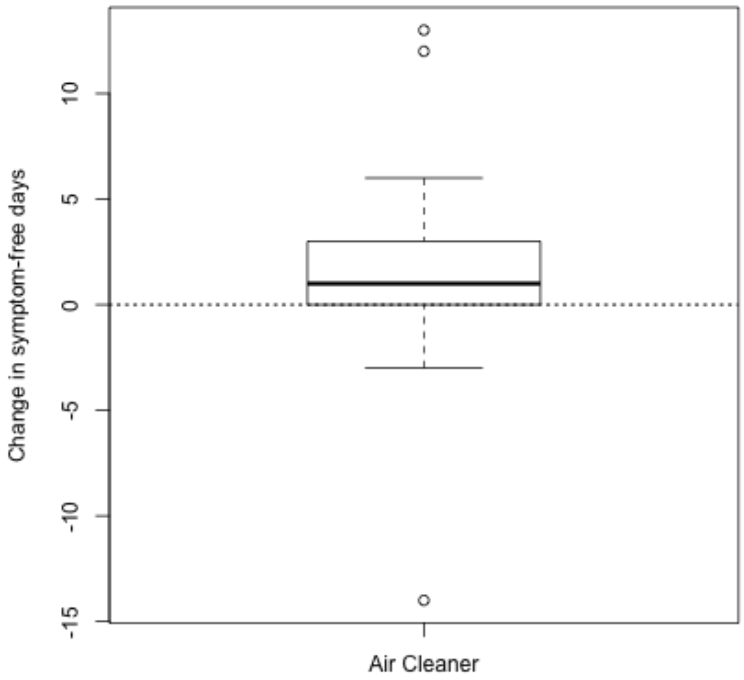
Principles of Analytic Graphics

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

Principles of Analytic Graphics

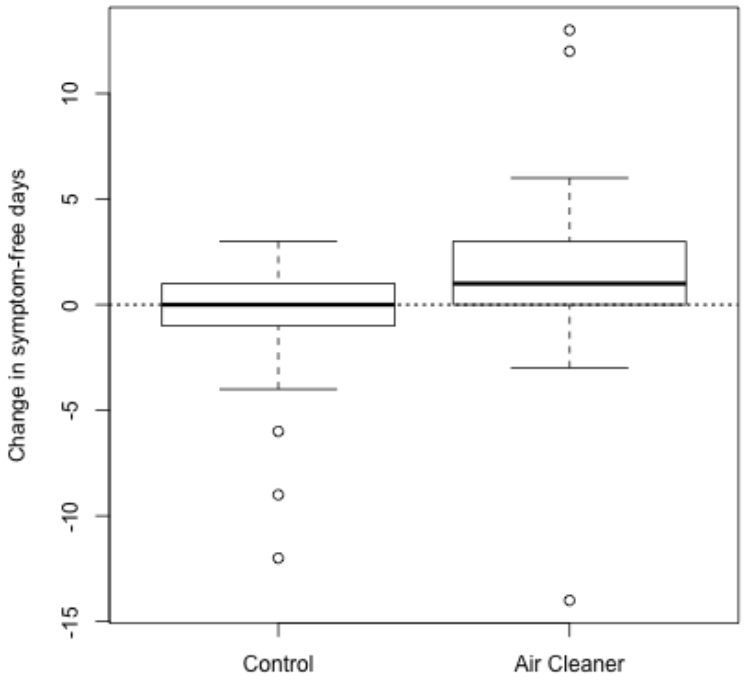
- Principle 1: Show comparisons
    - Evidence for a hypothesis is always *relative* to another competing hypothesis.
    - Always ask "Compared to What?"

Show Comparisons



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

Show Comparisons

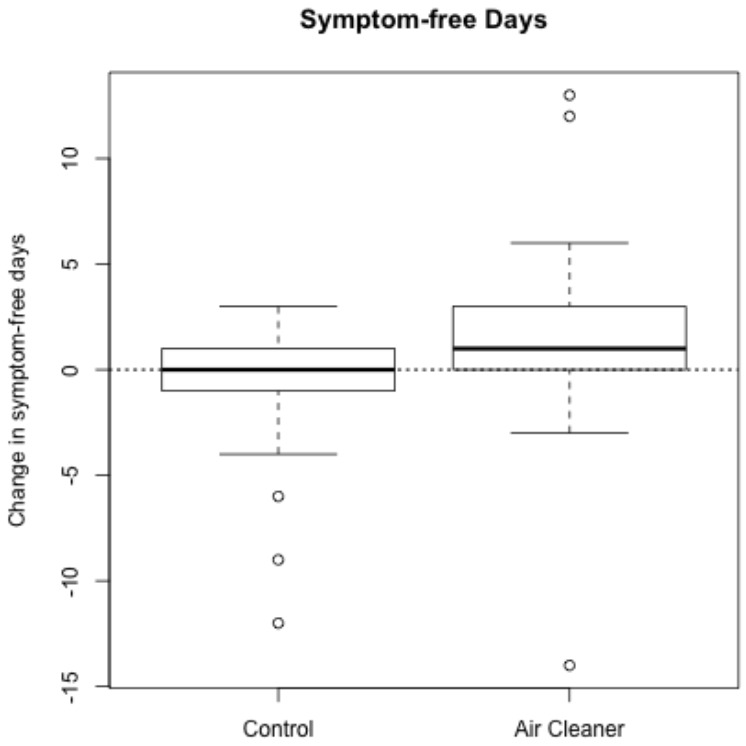


Reference: Butz AM, *et al.*, *JAMA Pediatrics*, 2011.

Principles of Analytic Graphics

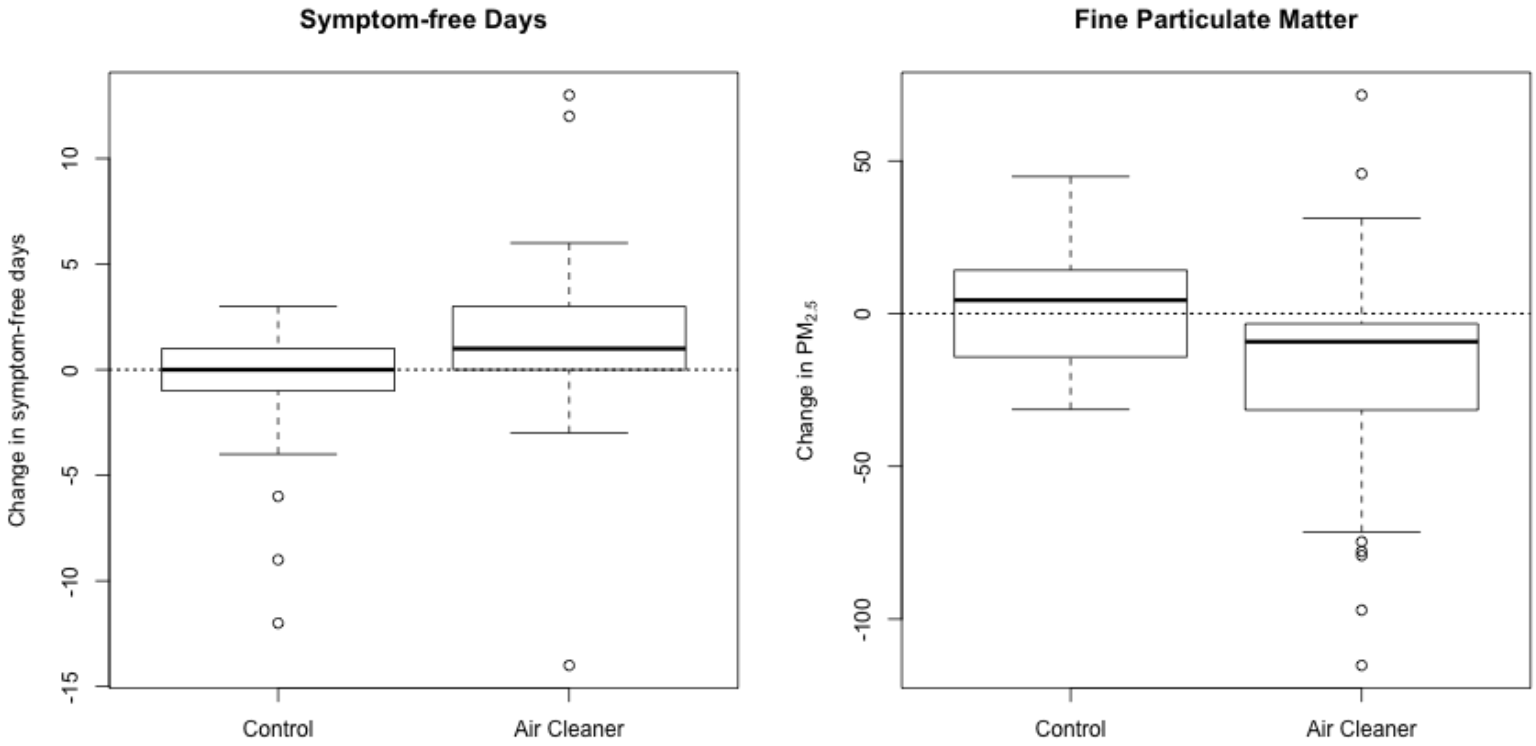
- Principle 1: Show comparisons
  - Evidence for a hypothesis is always *relative* to another competing hypothesis.
  - Always ask "Compared to What?"
- Principle 2: Show causality, mechanism, explanation, systematic structure
  - What is your causal framework for thinking about a question?

Show causality, mechanism



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

Show causality, mechanism

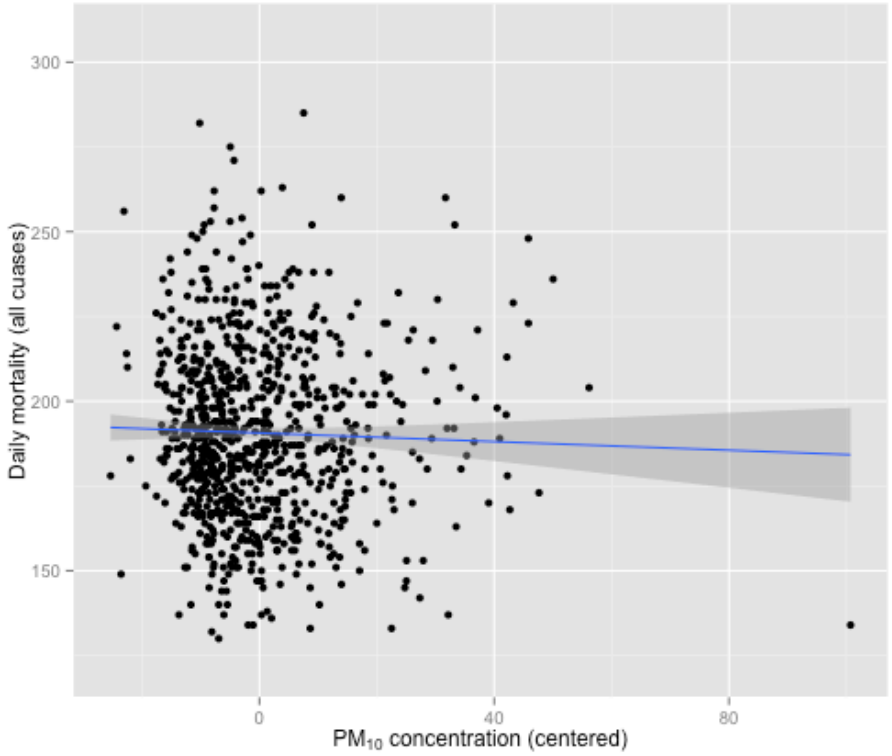


Reference: Butz AM, *et al.*, *JAMA Pediatrics*, 2011.

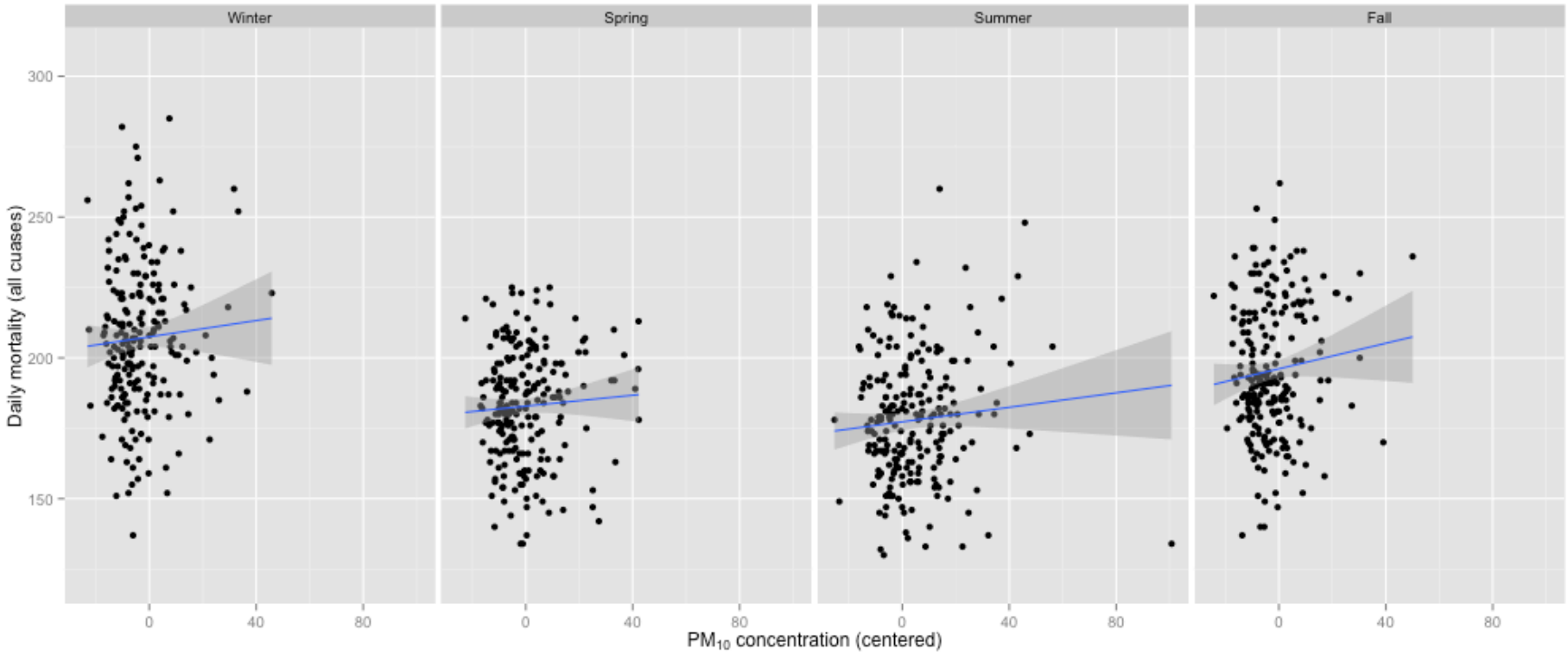
Principles of Analytic Graphics

- Principle 1: Show comparisons
  - Evidence for a hypothesis is always *relative* to another competing hypothesis.
  - Always ask "Compared to What?"
- Principle 2: Show causality, mechanism, explanation, systematic structure
  - What is your causal framework for thinking about a question?
- Principle 3: Show multivariate data
  - Multivariate = more than 2 variables
  - The real world is multivariate
  - Need to "escape flatland"

Show Multivariate Data



Show Multivariate Data

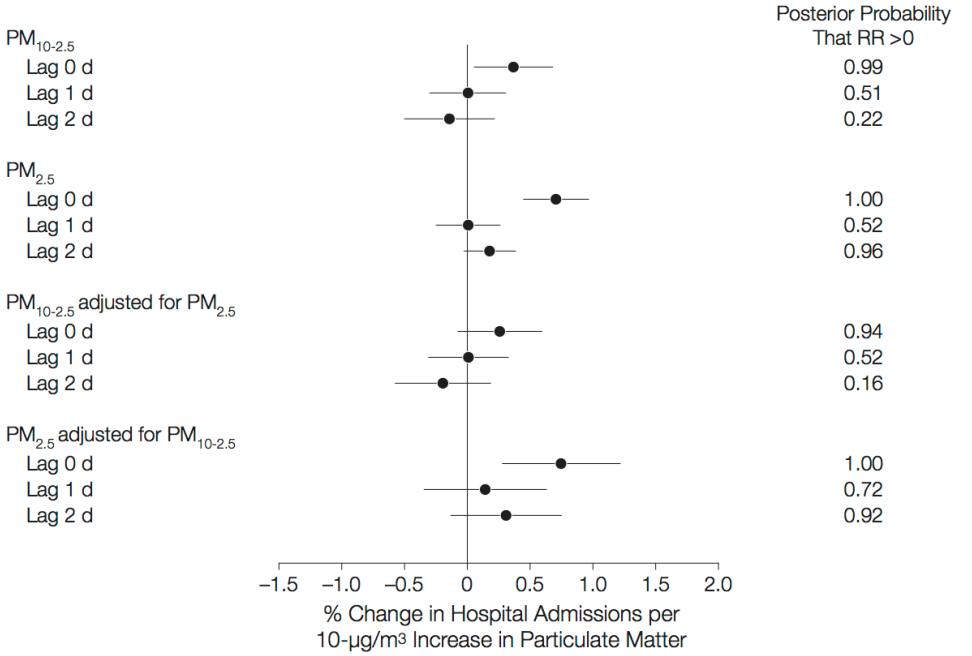


Principles of Analytic Graphics

- 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

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 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
- Principle 5: Describe and document the evidence with appropriate labels, scales, sources, etc.
  - A data graphic should tell a complete story that is credible

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
- 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

Summary

- Principle 1: Show comparisons
- Principle 2: Show causality, mechanism, explanation
- Principle 3: Show multivariate data
- Principle 4: Integrate multiple modes of evidence
- Principle 5: Describe and document the evidence
- Principle 6: Content is king

References

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

Exploratory Graphs

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

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

Exploratory graphs

- 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 (later)
- Color/size are primarily used for information

Air Pollution in the United States

- The U.S. Environmental Protection Agency (EPA) sets national ambient air quality standards for outdoor air pollution
  - U.S. National Ambient Air Quality Standards
- For fine particle pollution (PM2.5), the "annual mean, averaged over 3 years" cannot exceed $12\ \mu g/m^3$.
- Data on daily PM2.5 are available from the U.S. EPA web site
  - EPA Air Quality System
- **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 through 2010

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

```
##     pm25  fips region longitude latitude
## 1  9.771 01003   east    -87.75    30.59
## 2  9.994 01027   east    -85.84    33.27
## 3 10.689 01033   east    -87.73    34.73
## 4 11.337 01049   east    -85.80    34.46
## 5 12.120 01055   east    -86.03    34.02
## 6 10.828 01069   east    -85.35    31.19
```

Do any counties exceed the standard of $12\ \mu g/m^3$?

Simple Summaries of Data

One dimension

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

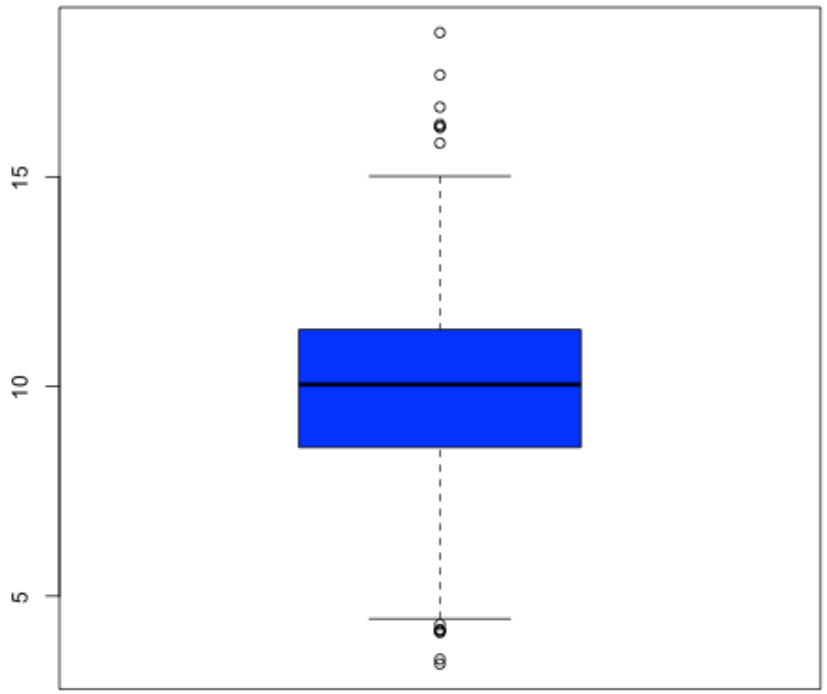
Five Number Summary

```
summary(pollution$pm25)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    3.38    8.55   10.00    9.84   11.40   18.40
```

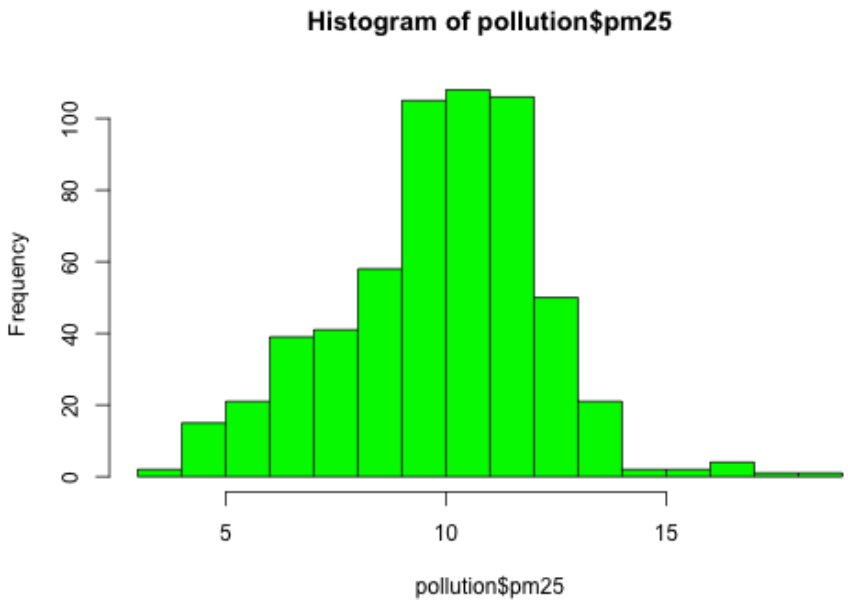
Boxplot

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



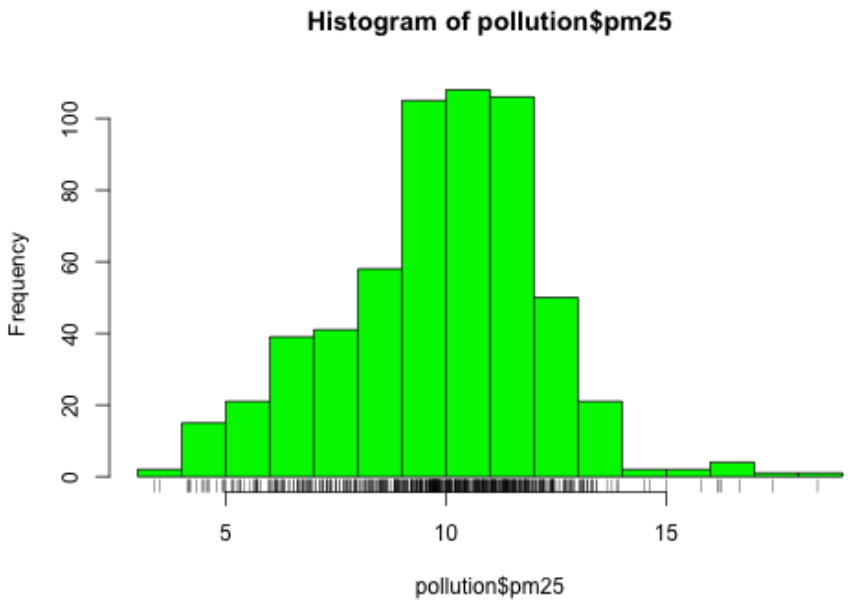
Histogram

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



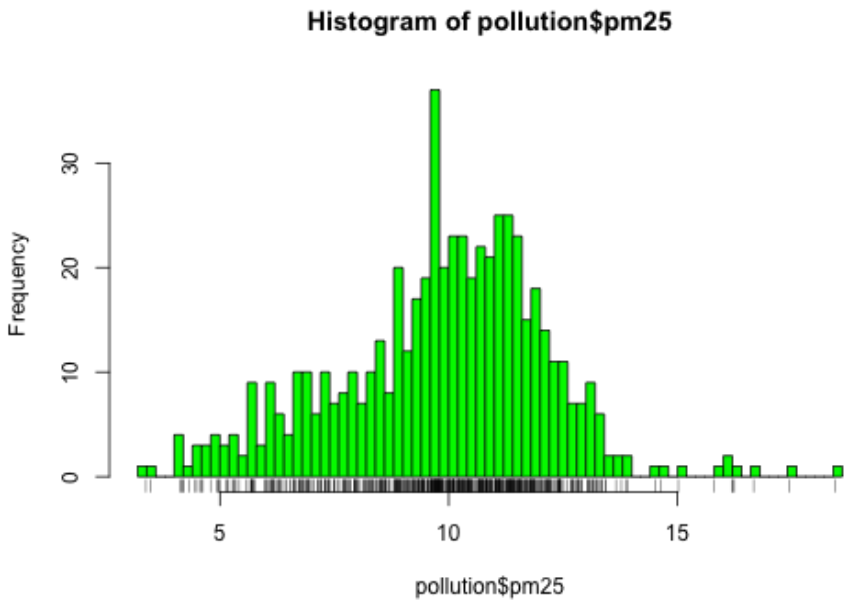
Histogram

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



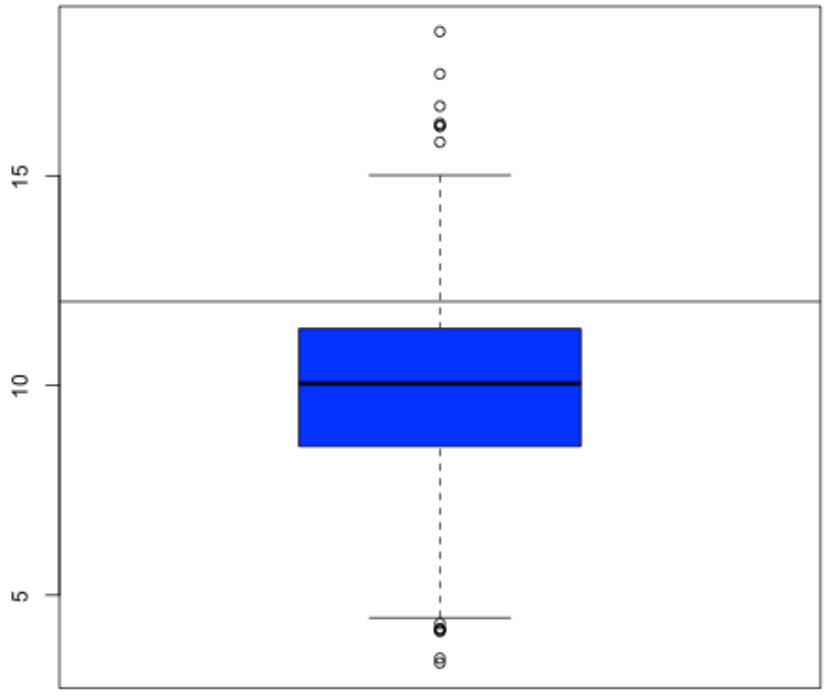
Histogram

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



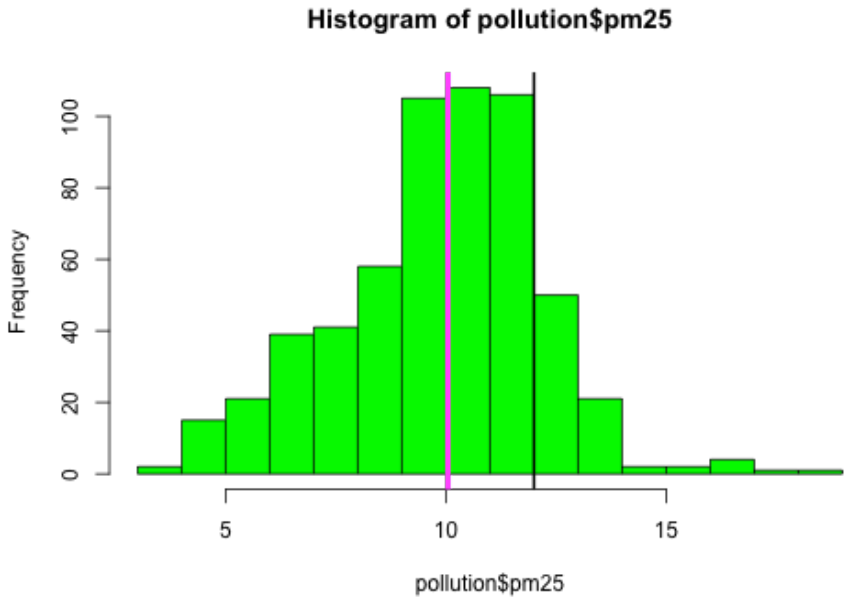
Overlaying Features

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



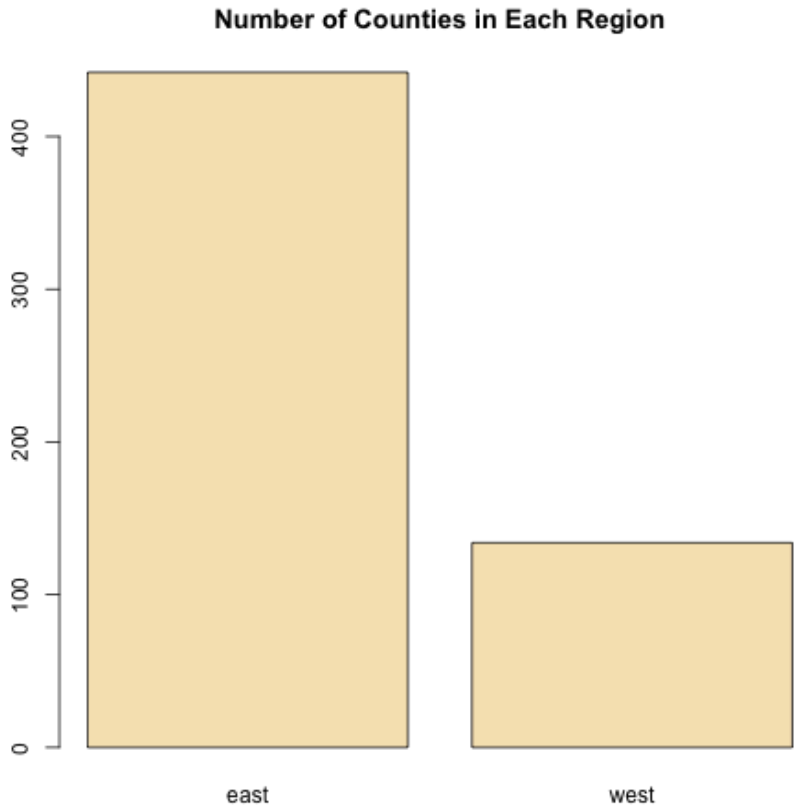
Overlaying Features

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



Barplot

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



Simple Summaries of Data

Two dimensions

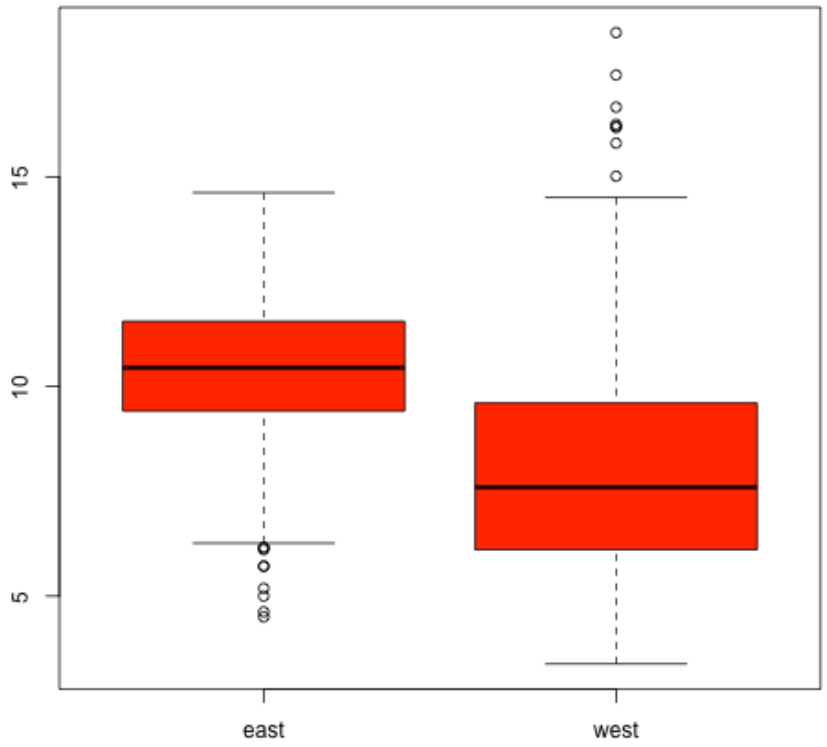
- Multiple/overlayed 1-D plots (Lattice/ggplot2)
- Scatterplots
- Smooth scatterplots

> 2 dimensions

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

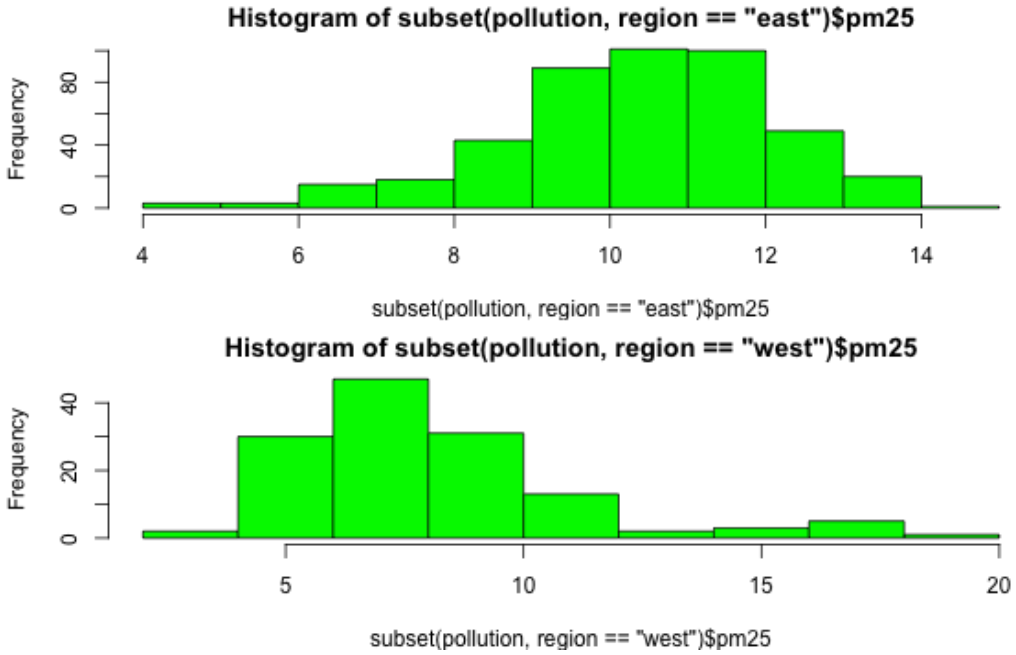
Multiple Boxplots

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



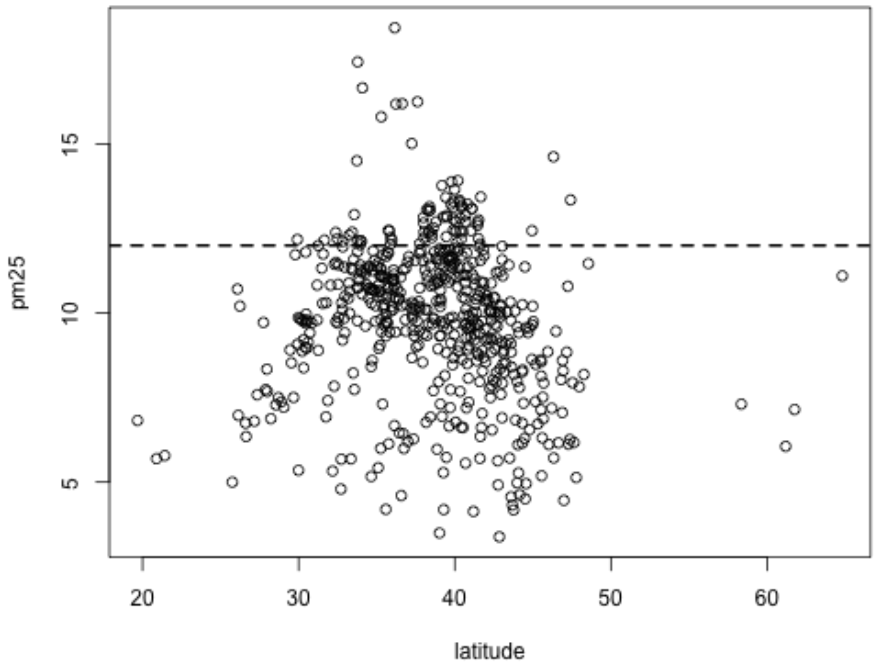
Multiple Histograms

```
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")
```



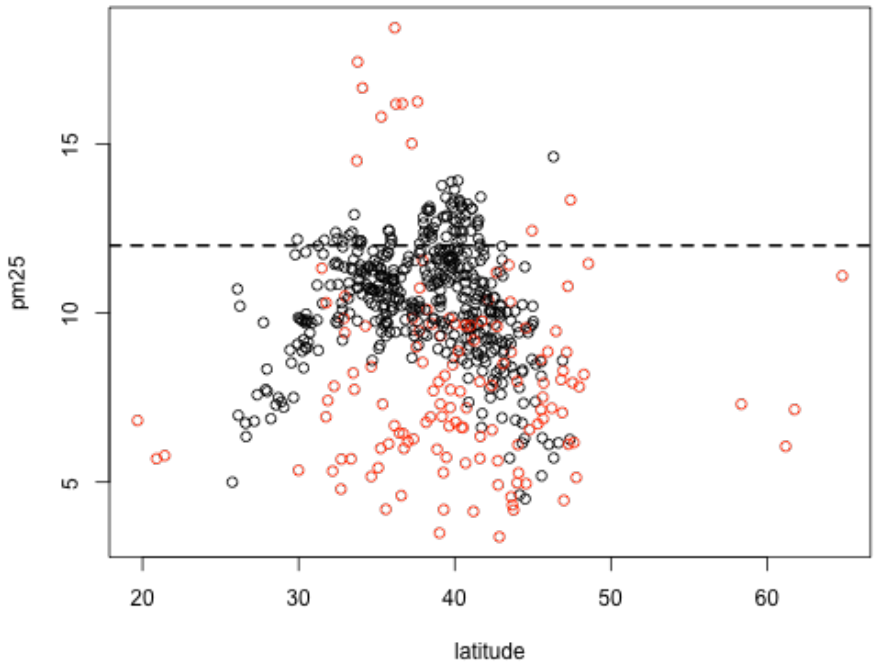
Scatterplot

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



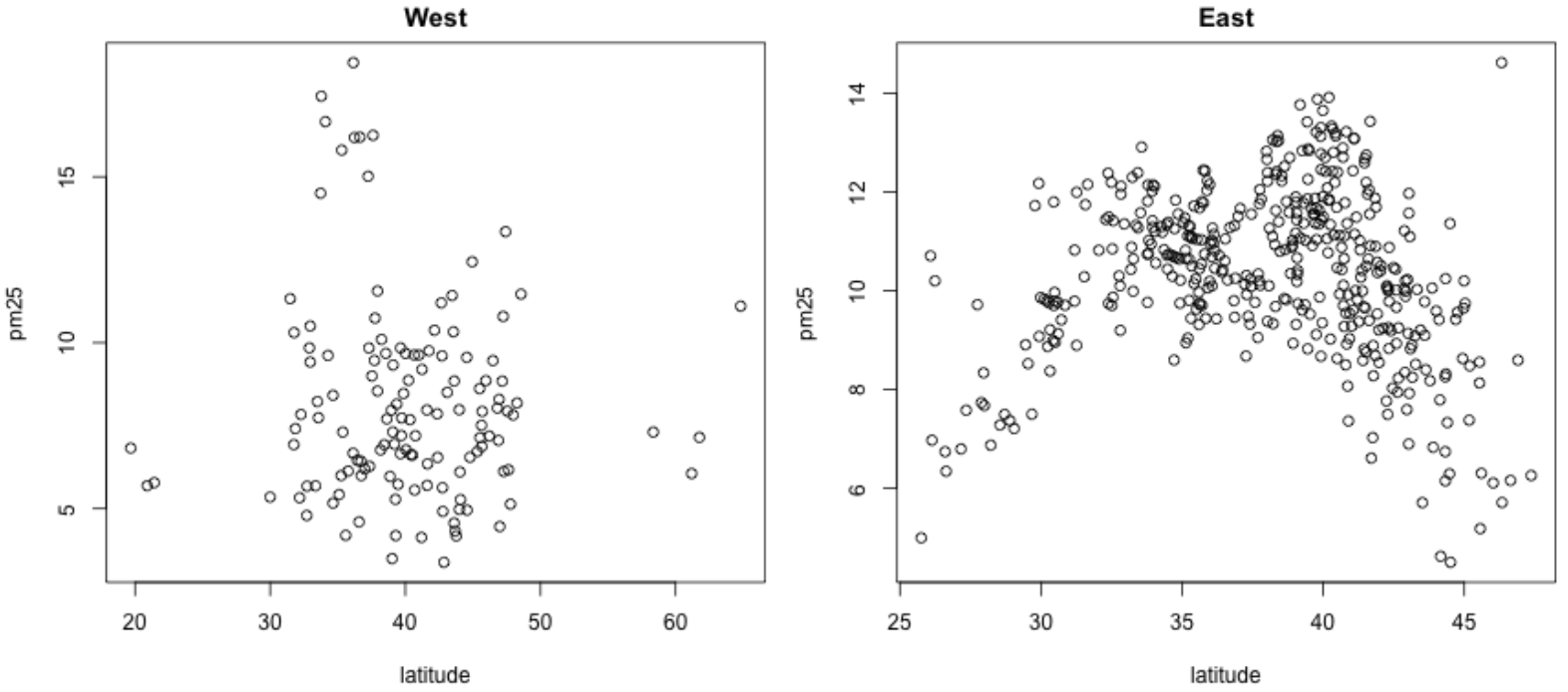
Scatterplot - Using Color

```
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"))
```



Summary

- Exploratory plots are "quick and dirty"
- Let you summarize the data (usually graphically) and highlight any broad features
- Explore basic questions and hypotheses (and perhaps rule them out)
- Suggest modeling strategies for the "next step"

Further resources

- R Graph Gallery
- R Bloggers

Plotting Systems in R

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

The Base Plotting System

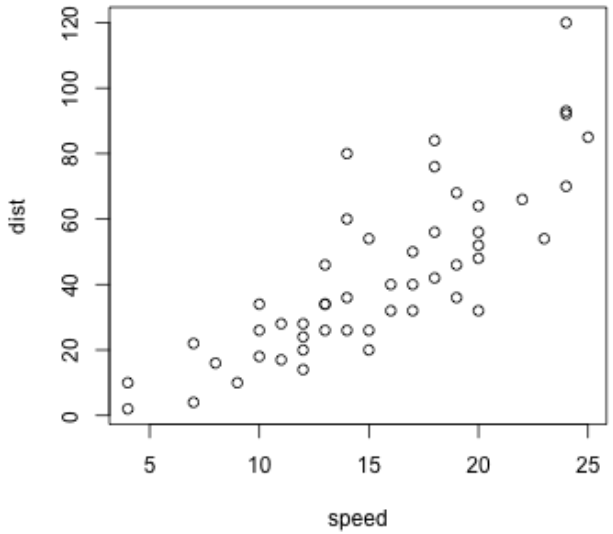
- "Artist's palette" model
- Start with blank canvas and build up from there
- Start with plot function (or similar)
- Use annotation functions to add/modify (`text`, `lines`, `points`, `axis`)

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
- Difficult to "translate" to others once a new plot has been created (no graphical "language")
- Plot is just a series of R commands

Base Plot

```
library(datasets)
data(cars)
with(cars, plot(speed, dist))
```



The Lattice System

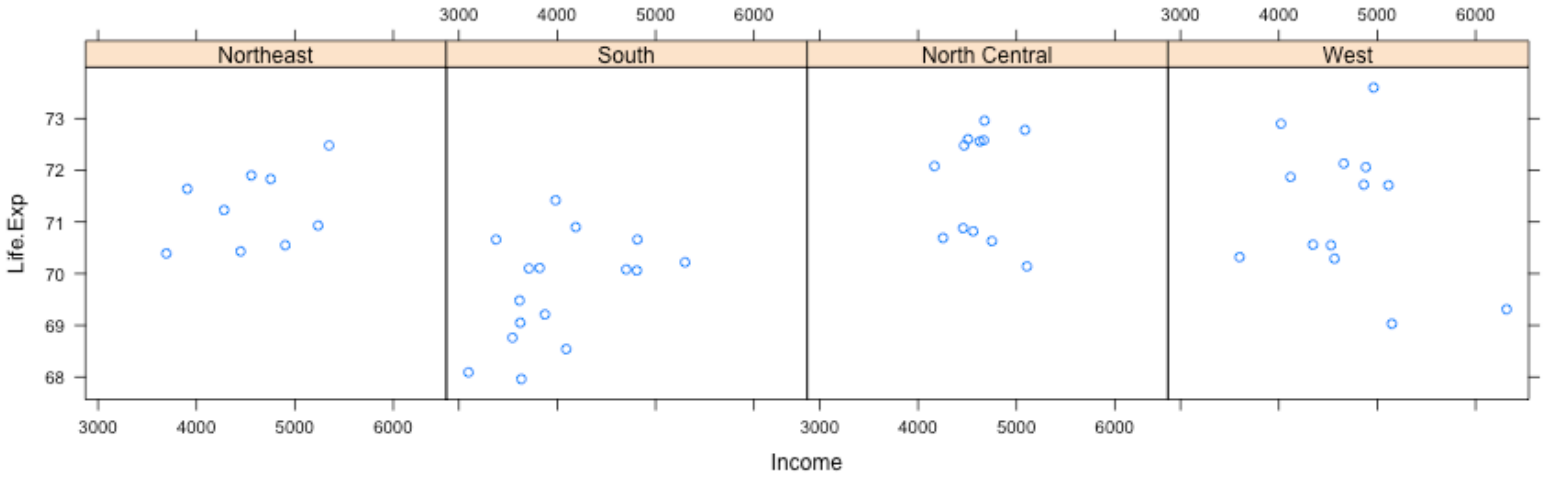
- Plots are created with a single function call (`xyplot`, `bwplot`, etc.)
- Most useful for conditioning types of plots: Looking at how y changes with x across levels of z
- Things like margins/spacing set automatically because entire plot is specified at once
- Good for puttng many many plots on a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

Lattice Plot

```
library(lattice)
state <- data.frame(state.x77, region = state.region)
xyplot(Life.Exp ~ Income | region, data = state, layout = c(4, 1))
```

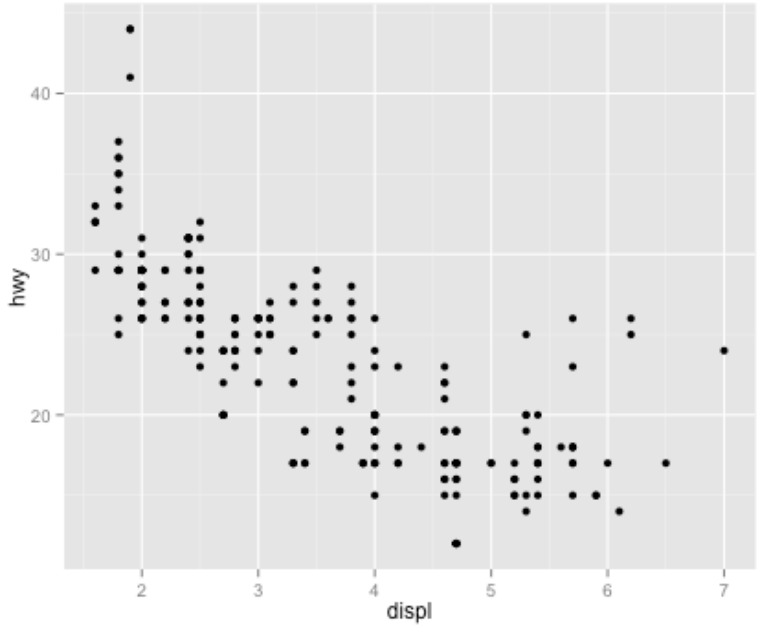


The ggplot2 System

- 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)
```



Summary

- Base: "artist's palette" model
- Lattice: Entire plot specified by one function; conditioning
- ggplot2: Mixes elements of Base and Lattice

References

Paul Murrell (2011). *R Graphics*, CRC Press.

Hadley Wickham (2009). *ggplot2*, Springer.

The Base Plotting System in R

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

Plotting System

The core plotting and graphics engine in R is encapsulated in the following packages:

- *graphics*: contains plotting functions for the "base" graphing systems, including `plot`, `hist`, `boxplot` and many others.
- *grDevices*: contains all the code implementing the various graphics devices, including X11, PDF, PostScript, PNG, etc.

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 Process of Making a Plot

When making a plot one must first make a few considerations (not necessarily in this order):

- Where will the plot be made? On the screen? In a file?
- How will the plot be used?
    - Is the plot for viewing temporarily on the screen?
    - Will it be presented in a web browser?
    - Will it eventually end up in a paper that might be printed?
    - Are you using it in a presentation?
- Is there a large amount of data going into the plot? Or is it just a few points?
- Do you need to be able to dynamically resize the graphic?

The Process of Making a Plot

- What graphics system will you use: base, lattice, or ggplot2? These generally cannot be mixed.
- Base graphics are usually constructed piecemeal, with each aspect of the plot handled separately through a series of function calls; this is often conceptually simpler and allows plotting to mirror the thought process
- Lattice graphics are usually created in a single function call, so all of the graphics parameters have to specified at once; specifying everything at once allows R to automatically calculate the necessary spacings and font sizes.
- ggplot2 combines concepts from both base and lattice graphics but uses an independent implementation

We focus on using the **base plotting system** to create graphics on the **screen device**.

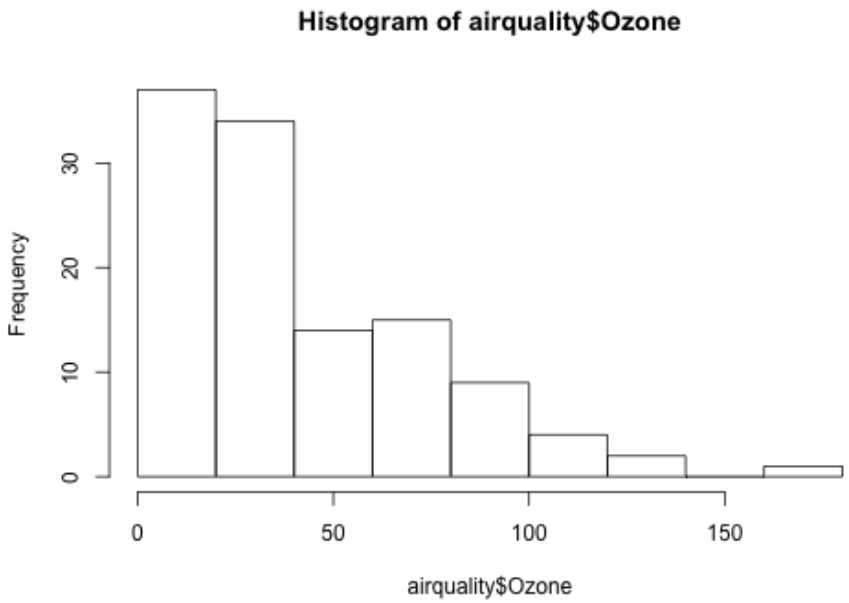
Base Graphics

Base graphics are used most commonly and are a very powerful system for creating 2-D graphics.

- There are two *phases* to creating a base plot
  - Initializing a new plot
  - Annotating (adding to) an existing plot
- Calling `plot(x, y)` or `hist(x)` will launch a graphics device (if one is not already open) and draw a new plot on the device
- If the arguments to `plot` are not of some special class, then the *default* method for `plot` is called; this function has *many* arguments, letting you set the title, x axis label, y axis label, etc.
- The base graphics system has *many* parameters that can set and tweaked; these parameters are documented in `?par`; it wouldn't hurt to try to memorize this help page!

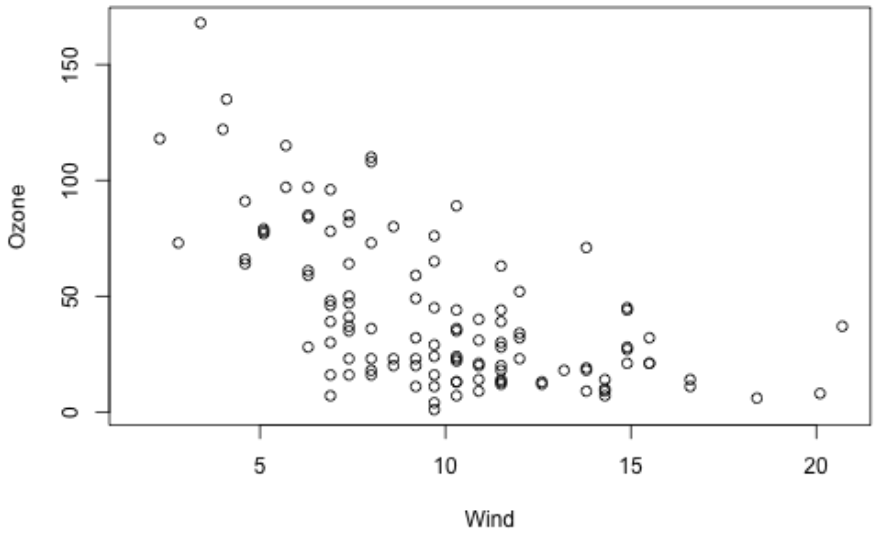
Simple Base Graphics: Histogram

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



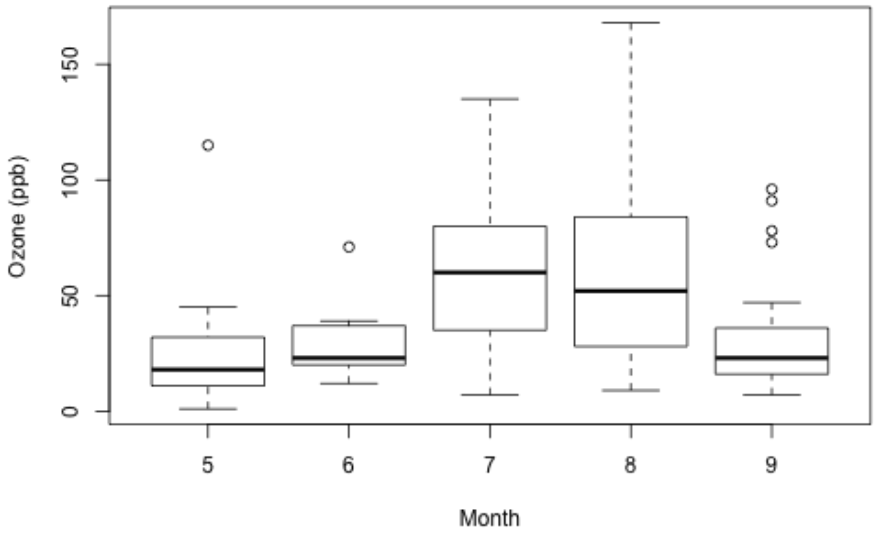
Simple Base Graphics: Scatterplot

```
library(datasets)
with(airquality, plot(Wind, Ozone))
```



Simple Base Graphics: Boxplot

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



Some Important Base Graphics Parameters

Many base plotting functions share a set of parameters. Here are a few key ones:

- `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)

Some Important Base Graphics Parameters

Default values for global graphics parameters

```
par("lty")
```

```
## [1] "solid"
```

```
par("col")
```

```
## [1] "black"
```

```
par("pch")
```

```
## [1] 1
```

Some Important Base Graphics Parameters

Default values for global graphics parameters

```
par("bg")
```

```
## [1] "transparent"
```

```
par("mar")
```

```
## [1] 5.1 4.1 4.1 2.1
```

```
par("mfrow")
```

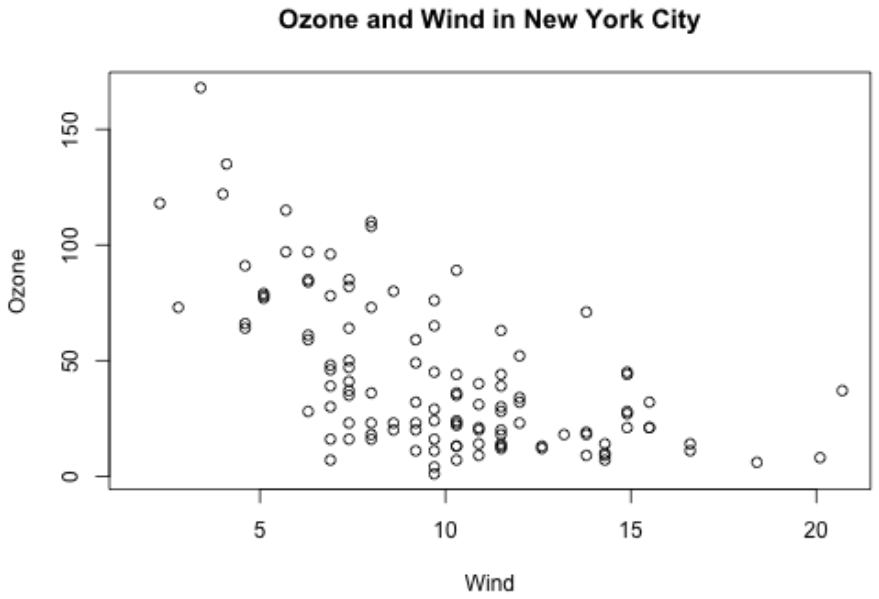
```
## [1] 1 1
```

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

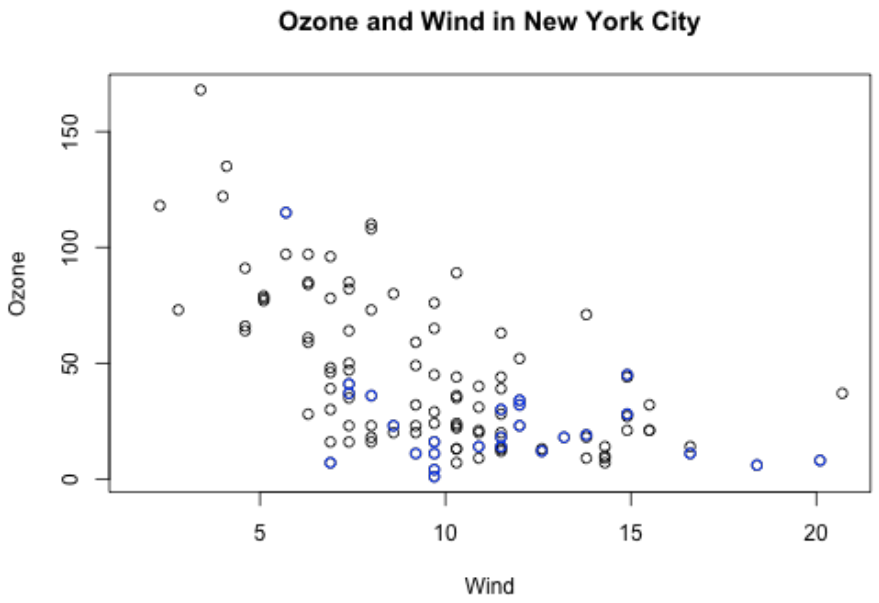
Base Plot with Annotation

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



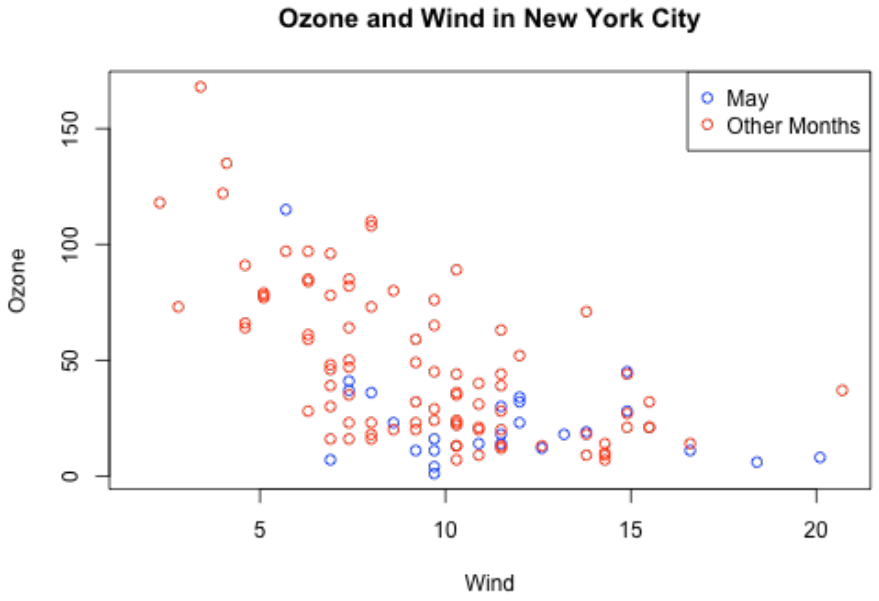
Base Plot with Annotation

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



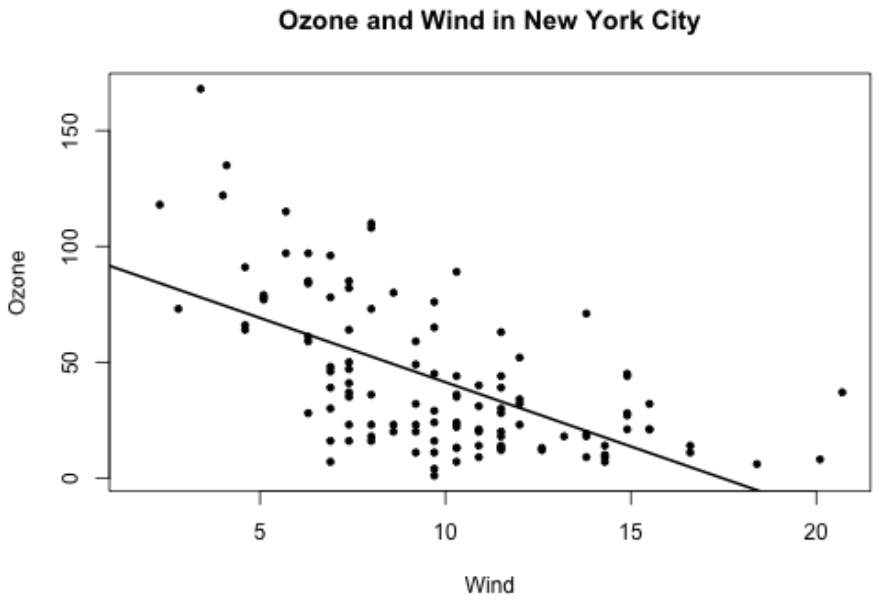
Base Plot with Annotation

```
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"))
```



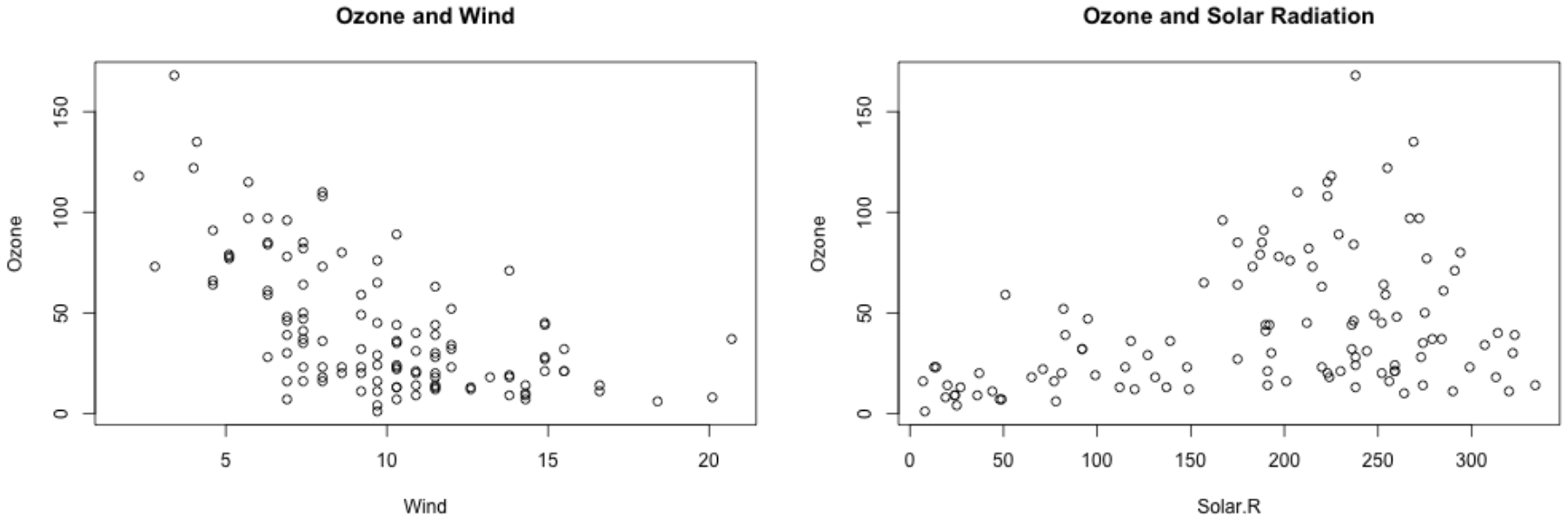
Base Plot with Regression Line

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



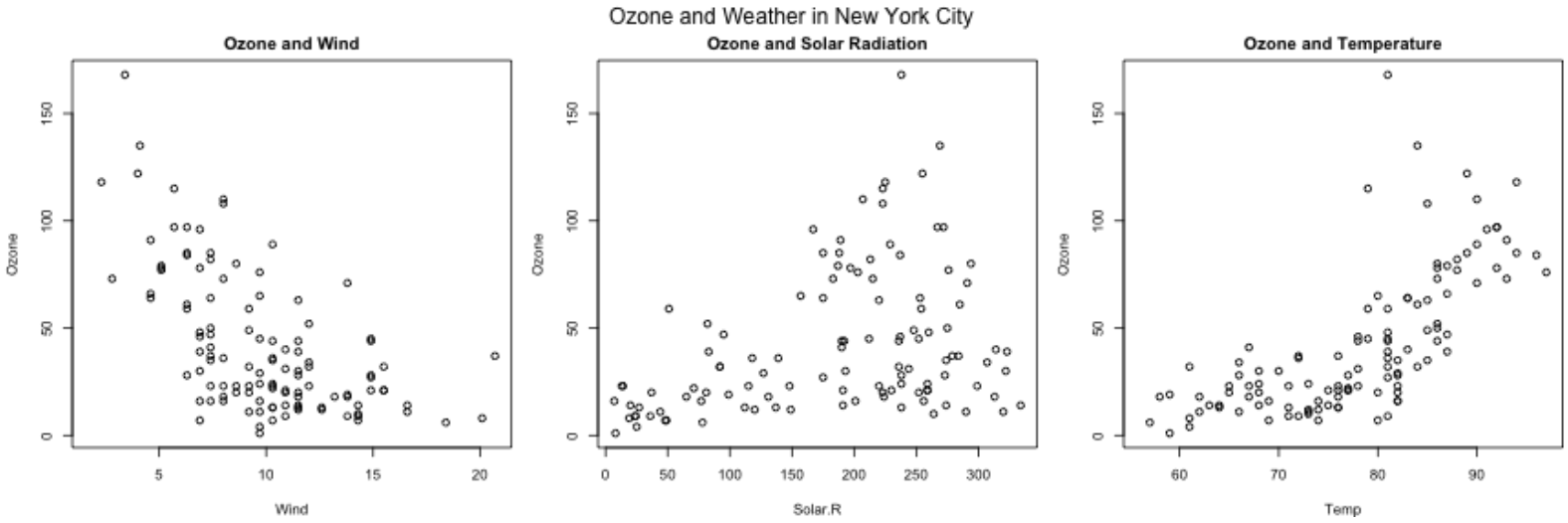
Multiple Base Plots

```
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)
})
```



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

Graphics Devices in R

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

What is a Graphics Device?

- A graphics device is something where you can make a plot appear
  - A window on your computer (screen device)
  - A PDF file (file device)
  - A PNG or JPEG file (file device)
  - A scalable vector graphics (SVG) file (file device)
- When you make a plot in R, it has to be "sent" to a specific graphics device
- The most common place for a plot to be "sent" is the *screen device*
  - On a Mac the screen device is launched with the `quartz()`
  - On Windows the screen device is launched with `windows()`
  - On Unix/Linux the screen device is launched with `x11()`

What is a Graphic Device?

- When making a plot, you need to consider how the plot will be used to determine what device the plot should be sent to.
  - The list of devices is found in `?Devices`; there are also devices created by users on CRAN
- For quick visualizations and exploratory analysis, usually you want to use the screen device
  - Functions like `plot` in base, `xyplot` in lattice, or `qplot` in ggplot2 will default to sending a plot to the screen device
  - On a given platform (Mac, Windows, Unix/Linux) there is only one screen device
- For plots that may be printed out or be incorporated into a document (e.g. papers/reports, slide presentations), usually a *file device* is more appropriate
  - There are many different file devices to choose from
- NOTE: Not all graphics devices are available on all platforms (i.e. you cannot launch the `windows()` on a Mac)

How Does a Plot Get Created?

There are two basic approaches to plotting. The first is most common:

1. Call a plotting function like `plot`, `xyplot`, or `qplot`
2. The plot appears on the screen device
3. Annotate plot if necessary
4. Enjoy

```
library(datasets)
with(faithful, plot(eruptions, waiting))  ## Make plot appear on screen device
title(main = "Old Faithful Geyser data")  ## Annotate with a title
```

How Does a Plot Get Created?

The second approach to plotting is most commonly used for file devices:

1. Explicitly launch a graphics device
2. Call a plotting function to make a plot (Note: if you are using a file device, no plot will appear on the screen)
3. Annotate plot if necessary
4. Explicitly close graphics device with `dev.off()` (this is very important!)

```
pdf(file = "myplot.pdf")  ## Open PDF device; create 'myplot.pdf' in my working directory
## Create plot and send to a file (no plot appears on screen)
with(faithful, plot(eruptions, waiting))
title(main = "Old Faithful Geyser data")  ## Annotate plot; still nothing on screen
dev.off()  ## Close the PDF file device
## Now you can view the file 'myplot.pdf' on your computer
```

Graphics File Devices

There are two basic types of file devices: *vector* and *bitmap* devices

Vector formats:

- `pdf`: useful for line-type graphics, resizes well, usually portable, not efficient if a plot has many objects/points
- `svg`: XML-based scalable vector graphics; supports animation and interactivity, potentially useful for web-based plots
- `win.metafile`: Windows metafile format (only on Windows)
- `postscript`: older format, also resizes well, usually portable, can be used to create encapsulated postscript files; Windows systems often don't have a postscript viewer

Graphics File Devices

Bitmap formats

- `png`: bitmapped format, good for line drawings or images with solid colors, uses lossless compression (like the old GIF format), most web browsers can read this format natively, good for plotting many many many points, does not resize well
- `jpeg`: good for photographs or natural scenes, uses lossy compression, good for plotting many many many points, does not resize well, can be read by almost any computer and any web browser, not great for line drawings
- `tiff`: Creates bitmap files in the TIFF format; supports lossless compression
- `bmp`: a native Windows bitmapped format

Multiple Open Graphics Devices

- It is possible to open multiple graphics devices (screen, file, or both), for example when viewing multiple plots at once
- Plotting can only occur on one graphics device at a time
- The **currently active** graphics device can be found by calling `dev.cur()`
- Every open graphics device is assigned an integer ≥ 2.
- You can change the active graphics device with `dev.set(<integer>)` where `<integer>` is the number associated with the graphics device you want to switch to

Copying Plots

Copying a plot to another device can be useful because some plots require a lot of code and it can be a pain to type all that in again for a different 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.

```
library(datasets)
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!
```

Summary

- Plots must be created on a graphics device
- The default graphics device is almost always the screen device, which is most useful for exploratory analysis
- File devices are useful for creating plots that can be included in other documents or sent to other people
- For file devices, there are vector and bitmap formats
    - Vector formats are good for line drawings and plots with solid colors using a modest number of points
    - Bitmap formats are good for plots with a large number of points, natural scenes or web-based plots