



EDA Case Study - Fine Particulate Matter Air Pollution in the United States

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



EDA Case Study - Understanding Human Activity with Smart Phones

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

Samsung Galaxy S3



<http://www.samsung.com/global/galaxys3/>

Samsung Data

UCI Machine Learning Repos X archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones



About Citation Policy Donate a Data Set Contact

Repository Web Google

[View ALL Data Sets](#)

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

Human Activity Recognition Using Smartphones Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: Human Activity Recognition database built from the recordings of 30 subjects performing activities of daily living (ADL) while carrying a waist-mounted smartphone with embedded inertial sensors.

Data Set Characteristics:	Multivariate, Time-Series	Number of Instances:	10299	Area:	Computer
Attribute Characteristics:	N/A	Number of Attributes:	561	Date Donated	2012-12-10
Associated Tasks:	Classification, Clustering	Missing Values?	N/A	Number of Web Hits:	5485

Source:

Jorge L. Reyes-Ortiz, Davide Anguita, Alessandro Ghio, Luca Oneto.
Smartlab - Non Linear Complex Systems Laboratory
DITEN - Università degli Studi di Genova, Genoa I-16145, Italy.
activityrecognition '@' smartlab.ws
www.smartlab.ws

<http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones>

Slightly processed data

Samsung data file

```
load("data/samsungData.rda")
names(samsungData)[1:12]
```

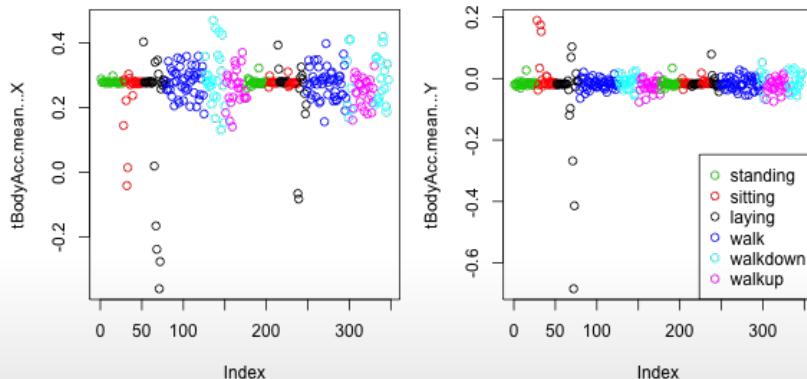
```
## [1] "tBodyAcc-mean()-X" "tBodyAcc-mean()-Y" "tBodyAcc-mean()-Z"
## [4] "tBodyAcc-std()-X"   "tBodyAcc-std()-Y"   "tBodyAcc-std()-Z"
## [7] "tBodyAcc-mad()-X"  "tBodyAcc-mad()-Y"  "tBodyAcc-mad()-Z"
## [10] "tBodyAcc-max()-X"  "tBodyAcc-max()-Y"  "tBodyAcc-max()-Z"
```

```
table(samsungData$activity)
```

```
##
##      laying   sitting standing      walk walkdown    walkup
##      1407     1286     1374     1226      986     1073
```

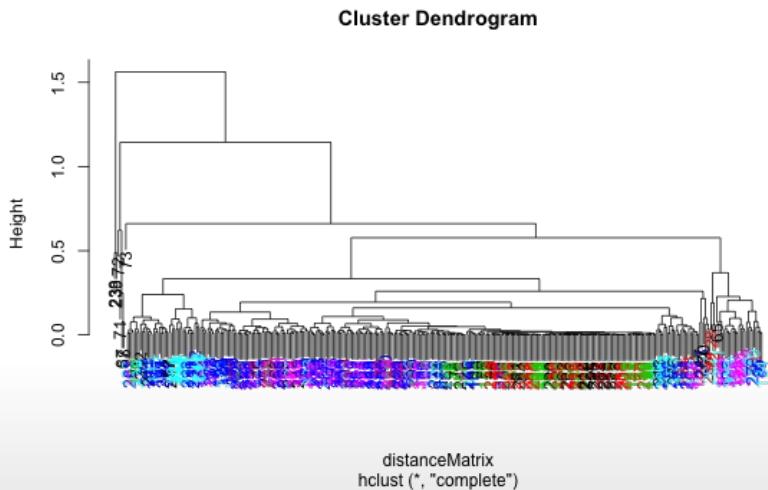
Plotting average acceleration for first subject

```
par(mfrow = c(1, 2), mar = c(5, 4, 1, 1))
samsungData <- transform(samsungData, activity = factor(activity))
sub1 <- subset(samsungData, subject == 1)
plot(sub1[, 1], col = sub1$activity, ylab = names(sub1)[1])
plot(sub1[, 2], col = sub1$activity, ylab = names(sub1)[2])
legend("bottomright", legend = unique(sub1$activity), col = unique(sub1$activity),
       pch = 1)
```



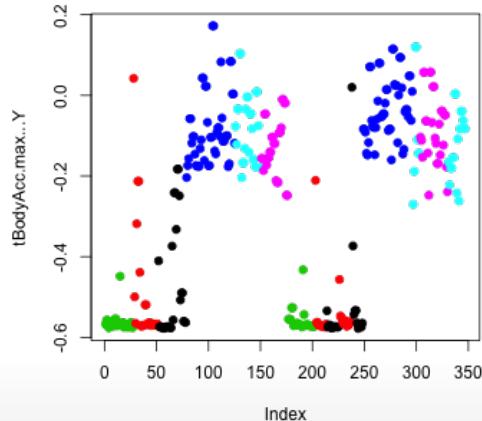
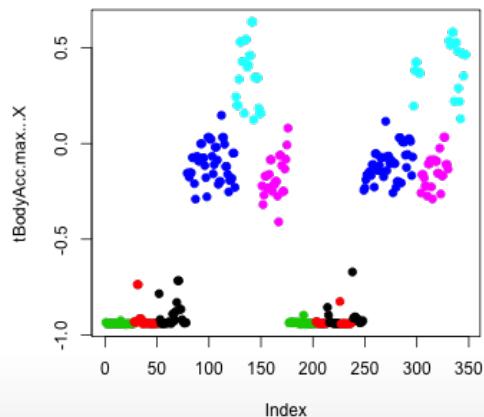
Clustering based just on average acceleration

```
source("myplclust.R")
distanceMatrix <- dist(sub1[, 1:3])
hclustering <- hclust(distanceMatrix)
myplclust(hclustering, lab.col = unclass(sub1$activity))
```



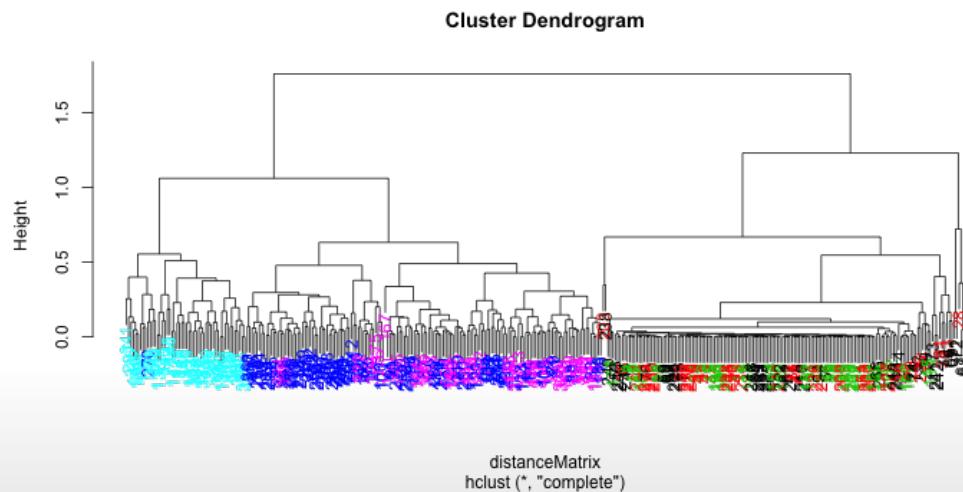
Plotting max acceleration for the first subject

```
par(mfrow = c(1, 2))
plot(sub1[, 10], pch = 19, col = sub1$activity, ylab = names(sub1)[10])
plot(sub1[, 11], pch = 19, col = sub1$activity, ylab = names(sub1)[11])
```



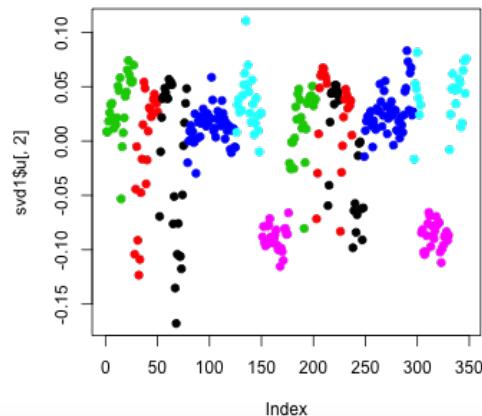
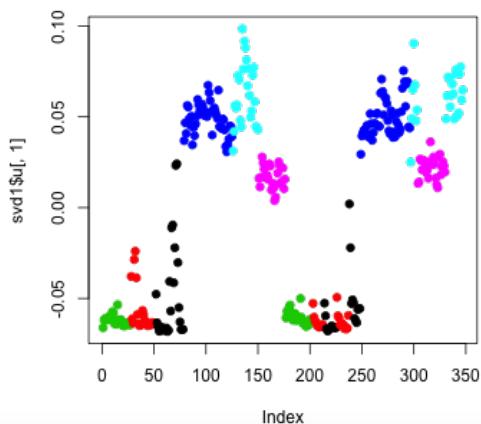
Clustering based on maximum acceleration

```
source("myplclust.R")
distanceMatrix <- dist(sub1[, 10:12])
hclustering <- hclust(distanceMatrix)
myplclust(hclustering, lab.col = unclass(sub1$activity))
```



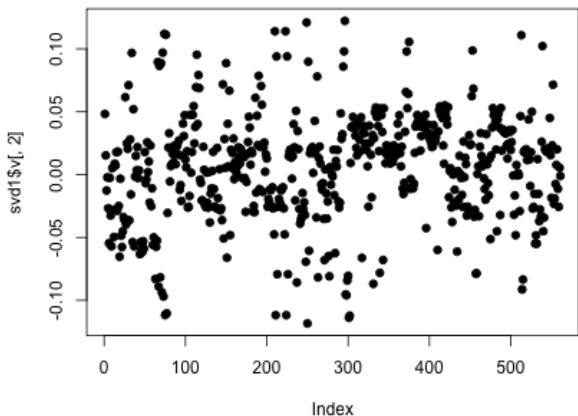
Singular Value Decomposition

```
svd1 = svd(scale(sub1[, -c(562, 563)]))  
par(mfrow = c(1, 2))  
plot(svd1$u[, 1], col = sub1$activity, pch = 19)  
plot(svd1$u[, 2], col = sub1$activity, pch = 19)
```



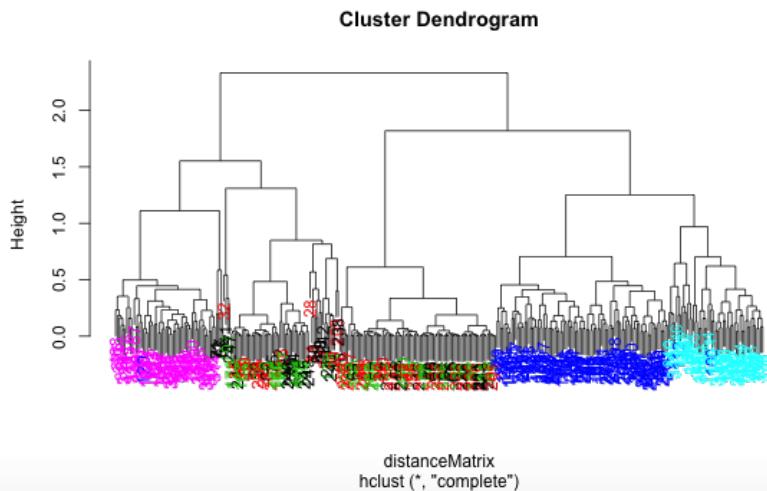
Find maximum contributor

```
plot(svd1$v[, 2], pch = 19)
```



New clustering with maximum contributer

```
maxContrib <- which.max(svd1$v[, 2])
distanceMatrix <- dist(sub1[, c(10:12, maxContrib)])
hclustering <- hclust(distanceMatrix)
myplclust(hclustering, lab.col = unclass(sub1$activity))
```



New clustering with maximum contributer

```
names(samsungData)[maxContrib]
```

```
## [1] "fBodyAcc.meanFreq...Z"
```

K-means clustering (nstart=1, first try)

```
kClust <- kmeans(sub1[, -c(562, 563)], centers = 6)
table(kClust$cluster, sub1$activity)
```

```
##
##      laying sitting standing walk walkdown walkup
## 1      0       0       0   50       1       0
## 2      0       0       0     0    48       0
## 3     27      37      51     0       0       0
## 4      3       0       0     0     0    53
## 5      0       0       0   45       0       0
## 6     20      10       2     0       0       0
```

K-means clustering (nstart=1, second try)

```
kClust <- kmeans(sub1[, -c(562, 563)], centers = 6, nstart = 1)
table(kClust$cluster, sub1$activity)
```

```
##
##      laying sitting standing walk walkdown walkup
## 1      0      0      0     0     0    49      0
## 2     18     10      2     0      0      0      0
## 3      0      0      0    95      0      0      0
## 4     29      0      0     0     0      0      0
## 5      0     37     51      0      0      0      0
## 6      3      0      0     0     0      0    53
```

K-means clustering (nstart=100, first try)

```
kClust <- kmeans(sub1[, -c(562, 563)], centers = 6, nstart = 100)
table(kClust$cluster, sub1$activity)
```

```
##
##      laying sitting standing walk walkdown walkup
## 1     18      10      2     0      0      0
## 2     29       0      0     0      0      0
## 3      0       0      0    95      0      0
## 4      0       0      0     0     49      0
## 5      3       0      0     0      0     53
## 6      0      37     51     0      0      0
```

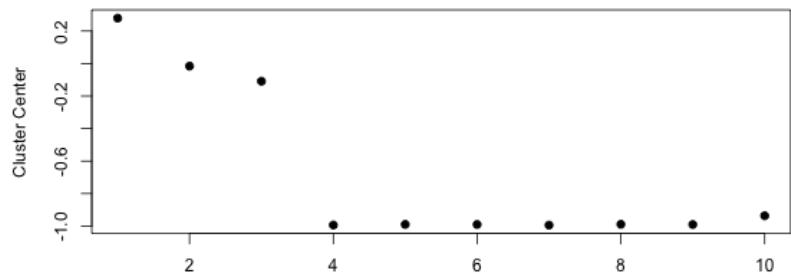
K-means clustering (nstart=100, second try)

```
kClust <- kmeans(sub1[, -c(562, 563)], centers = 6, nstart = 100)
table(kClust$cluster, sub1$activity)
```

```
##
##      laying sitting standing walk walkdown walkup
## 1     29       0       0   0       0       0
## 2      3       0       0   0       0      53
## 3      0       0       0   0      49       0
## 4      0       0       0  95       0       0
## 5      0      37      51   0       0       0
## 6     18      10       2   0       0       0
```

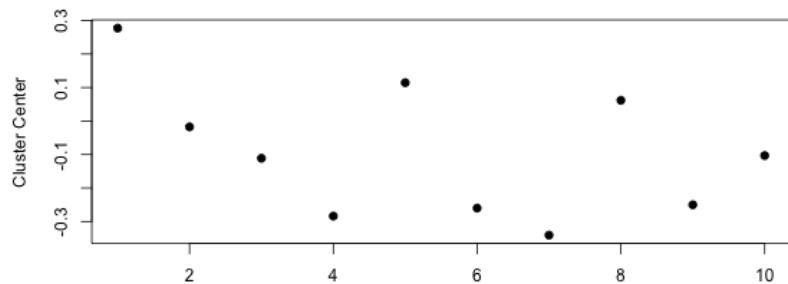
Cluster 1 Variable Centers (Laying)

```
plot(kClust$center[1, 1:10], pch = 19, ylab = "Cluster Center", xlab = "")
```



Cluster 2 Variable Centers (Walking)

```
plot(kClust$center[4, 1:10], pch = 19, ylab = "Cluster Center", xlab = "")
```





Plotting and Color in R

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

Plotting and Color

- The default color schemes for most plots in R are horrendous
 - I don't have good taste and even I know that
- Recently there have been developments to improve the handling/specification of colors in plots/graphs/etc.
- There are functions in R and in external packages that are very handy

Colors 1, 2, and 3



Default Image Plots in R

Color Utilities in R

- The `grDevices` package has two functions
 - `colorRamp`
 - `colorRampPalette`
- These functions take palettes of colors and help to interpolate between the colors
- The function `colors()` lists the names of colors you can use in any plotting function

Color Palette Utilities in R

- `colorRamp`: Take a palette of colors and return a function that takes values between 0 and 1, indicating the extremes of the color palette (e.g. see the 'gray' function)
- `colorRampPalette`: Take a palette of colors and return a function that takes integer arguments and returns a vector of colors interpolating the palette (like `heat.colors` or `topo.colors`)

colorRamp

[,1] [,2] [,3] corresponds to [Red] [Blue] [Green]

```
> pal <- colorRamp(c("red", "blue"))

> pal(0)
[,1] [,2] [,3]
[1,] 255     0     0

> pal(1)
[,1] [,2] [,3]
[1,] 0     0   255

> pal(0.5)
[,1] [,2] [,3]
[1,] 127.5  0 127.5
```

colorRamp

```
> pal(seq(0, 1, len = 10))  
      [,1] [,2]      [,3]  
[1,] 255.00000 0 0  
[2,] 226.66667 0 28.33333  
[3,] 198.33333 0 56.66667  
[4,] 170.00000 0 85.00000  
[5,] 141.66667 0 113.33333  
[6,] 113.33333 0 141.66667  
[7,] 85.00000 0 170.00000  
[8,] 56.66667 0 198.33333  
[9,] 28.33333 0 226.66667  
[10,] 0.00000 0 255.00000
```

colorRampPalette

```
> pal <- colorRampPalette(c("red", "yellow"))

> pal(2)
[1] "#FF0000" "#FFFF00"

> pal(10)
[1] "#FF0000" "#FF1C00" "#FF3800" "#FF5500" "#FF7100"
[6] "#FF8D00" "#FFAA00" "#FFC600" "#FFE200" "#FFFF00"
```

RColorBrewer Package

- One package on CRAN that contains interesting/useful color palettes
- There are 3 types of palettes
 - Sequential
 - Diverging
 - Qualitative
- Palette information can be used in conjunction with the `colorRamp()` and `colorRampPalette()`

RColorBrewer and colorRampPalette

```
> library(RColorBrewer)

> cols <- brewer.pal(3, "BuGn")

> cols
[1] "#E5F5F9" "#99D8C9" "#2CA25F"

> pal <- colorRampPalette(cols)

> image(volcano, col = pal(20))
```

RColorBrewer and colorRampPalette

The smoothScatter function

Some other plotting notes

- The `rgb` function can be used to produce any color via red, green, blue proportions
- Color transparency can be added via the `alpha` parameter to `rgb`
- The `colorspace` package can be used for a different control over colors

Scatterplot with no transparency



Scatterplot with transparency

Summary

- Careful use of colors in plots/maps/etc. can make it easier for the reader to get what you're trying to say (why make it harder?)
- The `RColorBrewer` package is an R package that provides color palettes for sequential, categorical, and diverging data
- The `colorRamp` and `colorRampPalette` functions can be used in conjunction with color palettes to connect data to colors
- Transparency can sometimes be used to clarify plots with many points

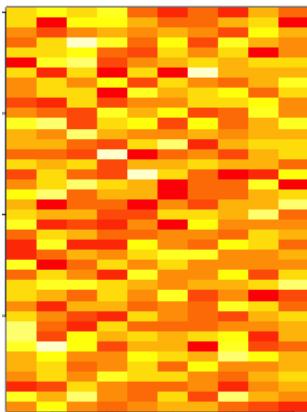


Principal Components Analysis and Singular Value Decomposition

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

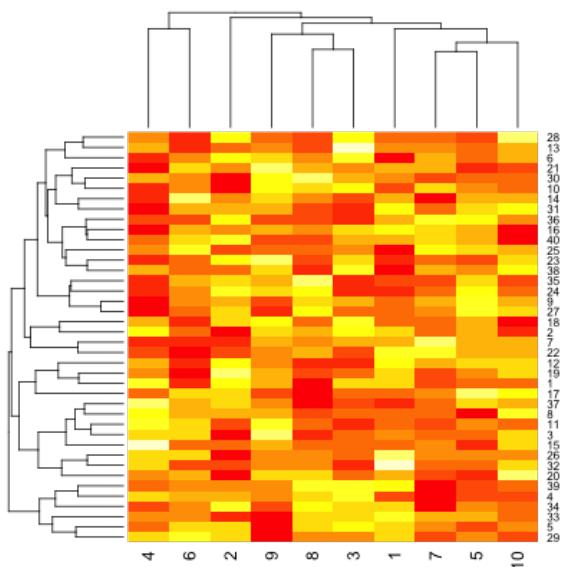
Matrix data

```
set.seed(12345)
par(mar = rep(0.2, 4))
dataMatrix <- matrix(rnorm(400), nrow = 40)
image(1:10, 1:40, t(dataMatrix)[, nrow(dataMatrix):1])
```



Cluster the data

```
par(mar = rep(0.2, 4))  
heatmap(dataMatrix)
```

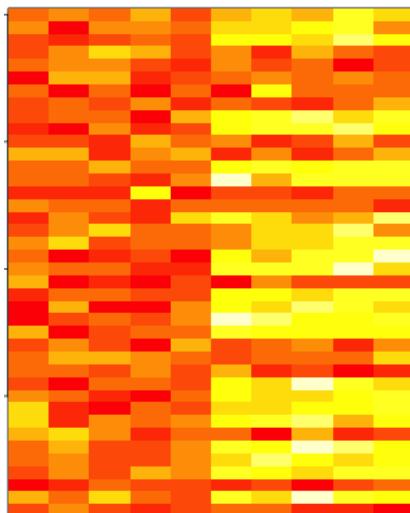


What if we add a pattern?

```
set.seed(678910)
for (i in 1:40) {
  # flip a coin
  coinFlip <- rbinom(1, size = 1, prob = 0.5)
  # if coin is heads add a common pattern to that row
  if (coinFlip) {
    dataMatrix[i, ] <- dataMatrix[i, ] + rep(c(0, 3), each = 5)
  }
}
```

What if we add a pattern? - the data

```
par(mar = rep(0.2, 4))
image(1:10, 1:40, t(dataMatrix)[, nrow(dataMatrix):1])
```

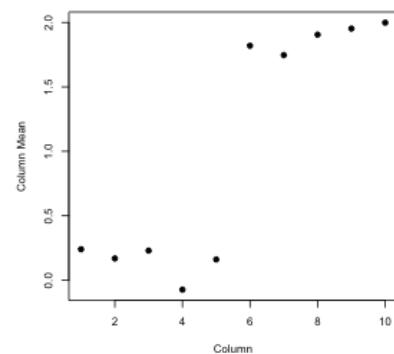
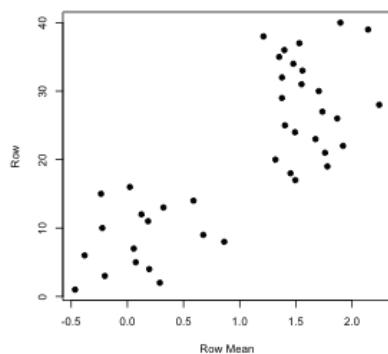
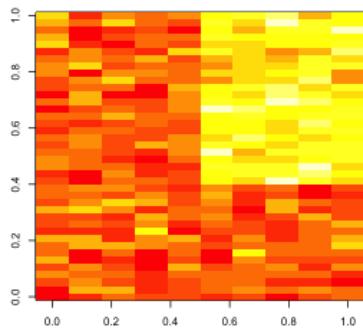


What if we add a pattern? - the clustered data

```
par(mar = rep(0.2, 4))
heatmap(dataMatrix)
```

Patterns in rows and columns

```
hh <- hclust(dist(dataMatrix))
dataMatrixOrdered <- dataMatrix[hh$order, ]
par(mfrow = c(1, 3))
image(t(dataMatrixOrdered)[, nrow(dataMatrixOrdered):1])
plot(rowMeans(dataMatrixOrdered), 40:1, , xlab = "Row Mean", ylab = "Row", pch = 19)
plot(colMeans(dataMatrixOrdered), xlab = "Column", ylab = "Column Mean", pch = 19)
```



Related problems

You have multivariate variables X_1, \dots, X_n so $X_1 = (X_{11}, \dots, X_{1m})$

- Find a new set of multivariate variables that are uncorrelated and explain as much variance as possible.
- If you put all the variables together in one matrix, find the best matrix created with fewer variables (lower rank) that explains the original data.

The first goal is **statistical** and the second goal is **data compression**.

Related solutions - PCA/SVD

SVD

If X is a matrix with each variable in a column and each observation in a row then the SVD is a "matrix decomposition"

$$X = UDV^T$$

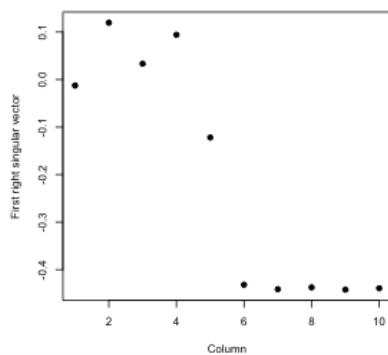
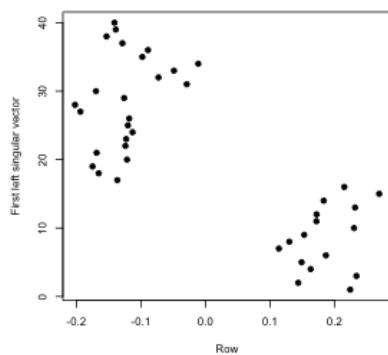
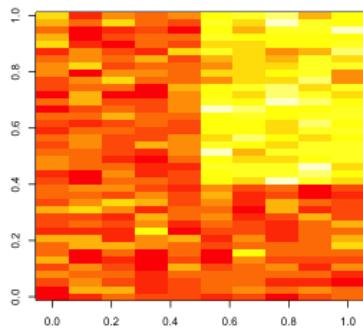
where the columns of U are orthogonal (left singular vectors), the columns of V are orthogonal (right singular vectors) and D is a diagonal matrix (singular values).

PCA

The principal components are equal to the right singular values if you first scale (subtract the mean, divide by the standard deviation) the variables.

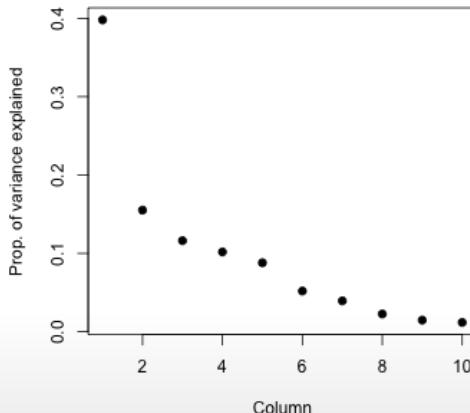
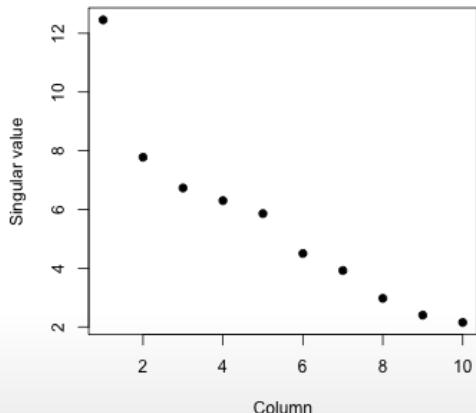
Components of the SVD - u and v

```
svd1 <- svd(scale(dataMatrixOrdered))
par(mfrow = c(1, 3))
image(t(dataMatrixOrdered)[, nrow(dataMatrixOrdered):1])
plot(svd1$u[, 1], 40:1, , xlab = "Row", ylab = "First left singular vector",
     pch = 19)
plot(svd1$v[, 1], xlab = "Column", ylab = "First right singular vector", pch = 19)
```



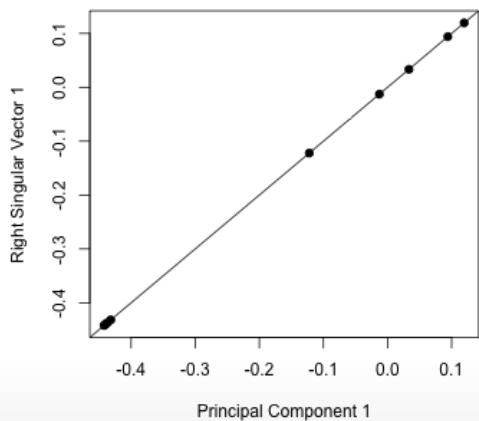
Components of the SVD - Variance explained

```
par(mfrow = c(1, 2))
plot(svd1$d, xlab = "Column", ylab = "Singular value", pch = 19)
plot(svd1$d^2/sum(svd1$d^2), xlab = "Column", ylab = "Prop. of variance explained",
     pch = 19)
```



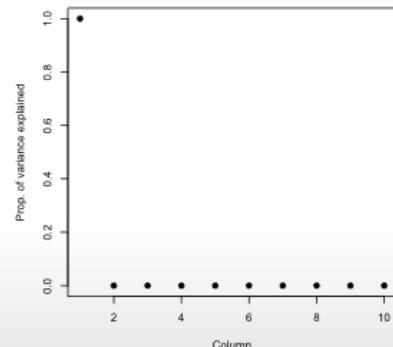
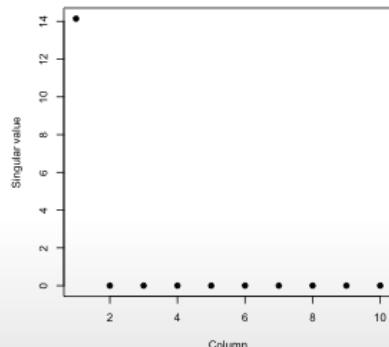
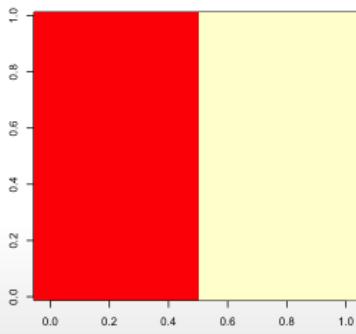
Relationship to principal components

```
svd1 <- svd(scale(dataMatrixOrdered))
pca1 <- prcomp(dataMatrixOrdered, scale = TRUE)
plot(pca1$rotation[, 1], svd1$v[, 1], pch = 19, xlab = "Principal Component 1",
     ylab = "Right Singular Vector 1")
abline(c(0, 1))
```



Components of the SVD - variance explained

```
constantMatrix <- dataMatrixOrdered*0  
for(i in 1:dim(dataMatrixOrdered)[1]) {constantMatrix[i,] <- rep(c(0,1),each=5)}  
svd1 <- svd(constantMatrix)  
par(mfrow=c(1,3))  
image(t(constantMatrix)[,nrow(constantMatrix):1])  
plot(svd1$d,xlab="Column",ylab="Singular value",pch=19)  
plot(svd1$d^2/sum(svd1$d^2),xlab="Column",ylab="Prop. of variance explained",pch=19)
```

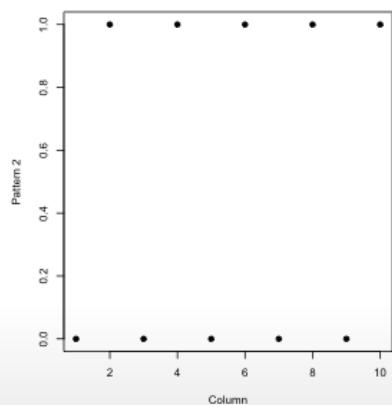
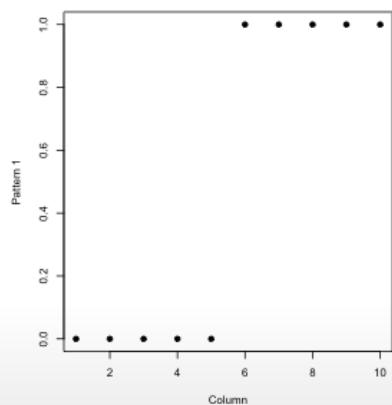
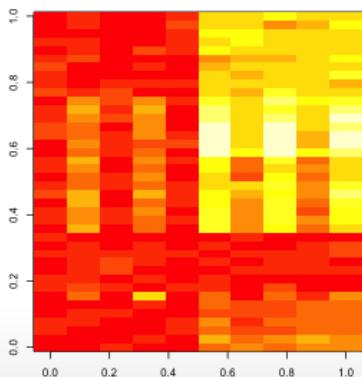


What if we add a second pattern?

```
set.seed(678910)
for (i in 1:40) {
  # flip a coin
  coinFlip1 <- rbinom(1, size = 1, prob = 0.5)
  coinFlip2 <- rbinom(1, size = 1, prob = 0.5)
  # if coin is heads add a common pattern to that row
  if (coinFlip1) {
    dataMatrix[i, ] <- dataMatrix[i, ] + rep(c(0, 5), each = 5)
  }
  if (coinFlip2) {
    dataMatrix[i, ] <- dataMatrix[i, ] + rep(c(0, 5), 5)
  }
}
hh <- hclust(dist(dataMatrix))
dataMatrixOrdered <- dataMatrix[hh$order, ]
```

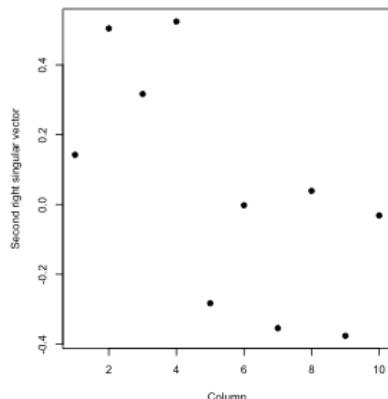
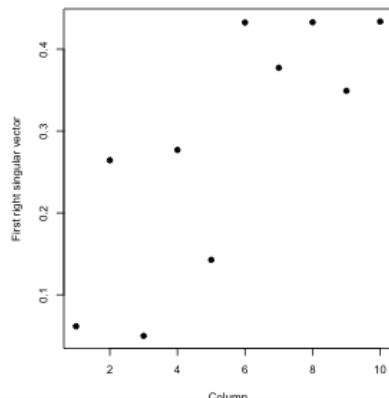
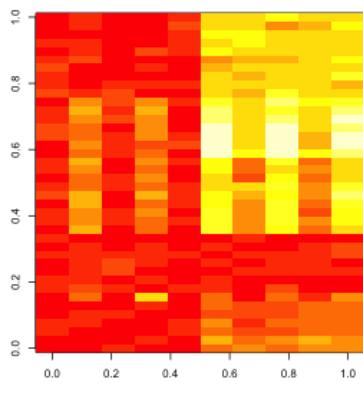
Singular value decomposition - true patterns

```
svd2 <- svd(scale(dataMatrixOrdered))
par(mfrow = c(1, 3))
image(t(dataMatrixOrdered)[, nrow(dataMatrixOrdered):1])
plot(rep(c(0, 1), each = 5), pch = 19, xlab = "Column", ylab = "Pattern 1")
plot(rep(c(0, 1), 5), pch = 19, xlab = "Column", ylab = "Pattern 2")
```



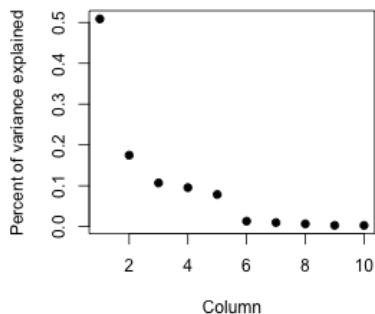
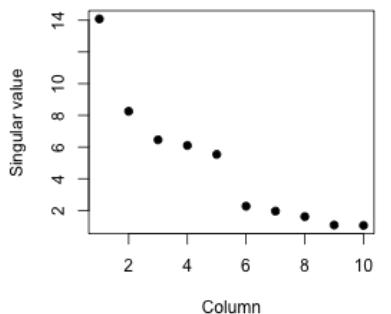
v and patterns of variance in rows

```
svd2 <- svd(scale(dataMatrixOrdered))
par(mfrow = c(1, 3))
image(t(dataMatrixOrdered)[, nrow(dataMatrixOrdered):1])
plot(svd2$v[, 1], pch = 19, xlab = "Column", ylab = "First right singular vector")
plot(svd2$v[, 2], pch = 19, xlab = "Column", ylab = "Second right singular vector")
```



d and variance explained

```
svd1 <- svd(scale(dataMatrixOrdered))  
par(mfrow = c(1, 2))  
plot(svd1$d, xlab = "Column", ylab = "Singular value", pch = 19)  
plot(svd1$d^2/sum(svd1$d^2), xlab = "Column", ylab = "Percent of variance explained",  
     pch = 19)
```



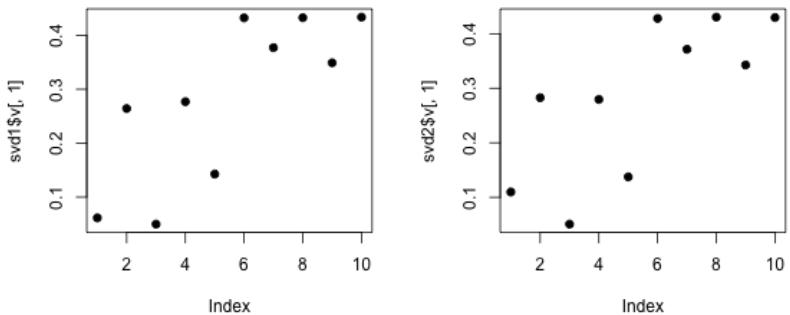
Missing values

```
dataMatrix2 <- dataMatrixOrdered  
## Randomly insert some missing data  
dataMatrix2[sample(1:100, size = 40, replace = FALSE)] <- NA  
svd1 <- svd(scale(dataMatrix2)) ## Doesn't work!
```

```
## Error: infinite or missing values in 'x'
```

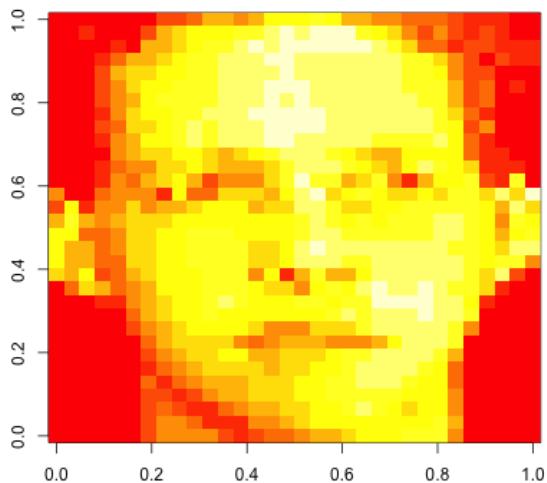
Imputing {impute}

```
library(impute) ## Available from http://bioconductor.org
dataMatrix2 <- dataMatrixOrdered
dataMatrix2[sample(1:100,size=40,replace=FALSE)] <- NA
dataMatrix2 <- impute.knn(dataMatrix2)$data
svd1 <- svd(scale(dataMatrixOrdered)); svd2 <- svd(scale(dataMatrix2))
par(mfrow=c(1,2)); plot(svd1$v[,1],pch=19); plot(svd2$v[,1],pch=19)
```



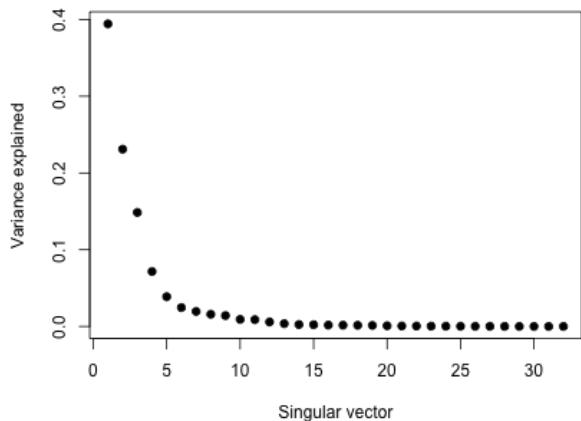
Face example

```
load("data/face.rda")
image(t(faceData)[, nrow(faceData):1])
```



Face example - variance explained

```
svd1 <- svd(scale(faceData))
plot(svd1$d^2/sum(svd1$d^2), pch = 19, xlab = "Singular vector", ylab = "Variance explained")
```



Face example - create approximations

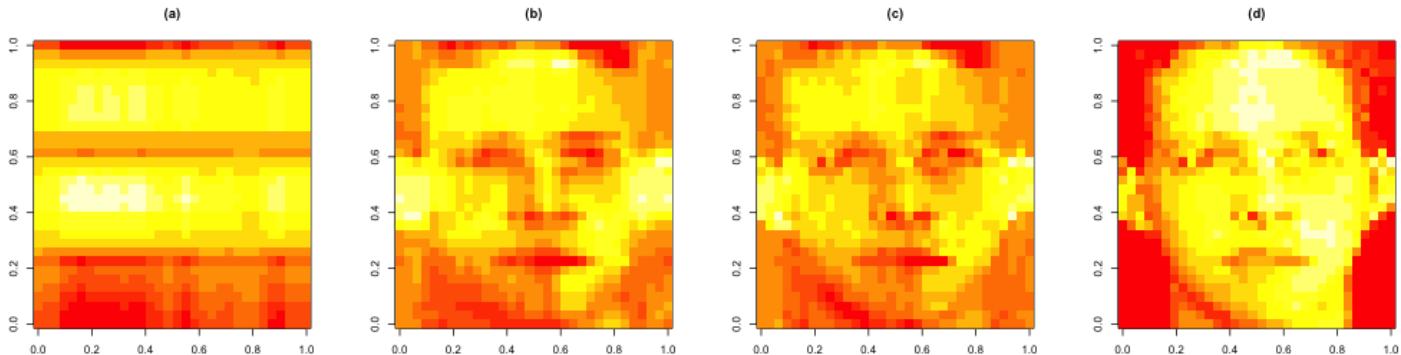
```
svd1 <- svd(scale(faceData))
## Note that %*% is matrix multiplication

# Here svd1$d[1] is a constant
approx1 <- svd1$u[, 1] %*% t(svd1$v[, 1]) * svd1$d[1]

# In these examples we need to make the diagonal matrix out of d
approx5 <- svd1$u[, 1:5] %*% diag(svd1$d[1:5]) %*% t(svd1$v[, 1:5])
approx10 <- svd1$u[, 1:10] %*% diag(svd1$d[1:10]) %*% t(svd1$v[, 1:10])
```

Face example - plot approximations

```
par(mfrow = c(1, 4))
image(t(approx1)[, nrow(approx1):1], main = "(a)")
image(t(approx5)[, nrow(approx5):1], main = "(b)")
image(t(approx10)[, nrow(approx10):1], main = "(c)")
image(t(faceData)[, nrow(faceData):1], main = "(d)") ## Original data
```



Notes and further resources

- Scale matters
- PC's/SV's may mix real patterns
- Can be computationally intensive
- [Advanced data analysis from an elementary point of view](#)
- [Elements of statistical learning](#)
- Alternatives
 - [Factor analysis](#)
 - [Independent components analysis](#)
 - [Latent semantic analysis](#)



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 $12 \mu\text{g}/\text{m}^3$.
- 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 $\mu\text{g}/\text{m}^3$?

Simple Summaries of Data

One dimension

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

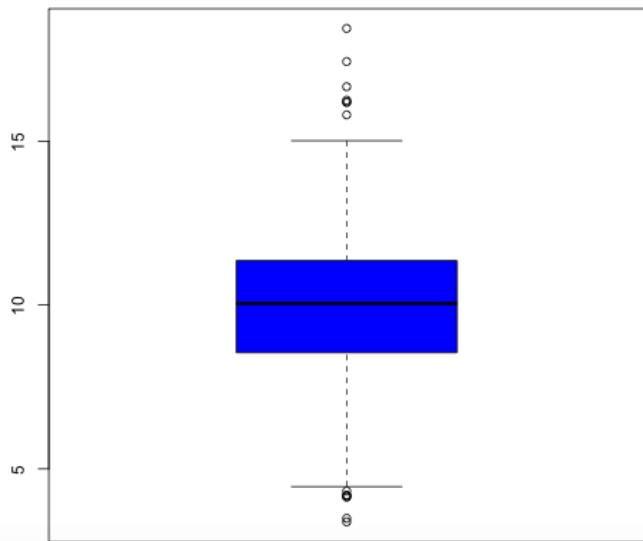
Five Number Summary

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

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

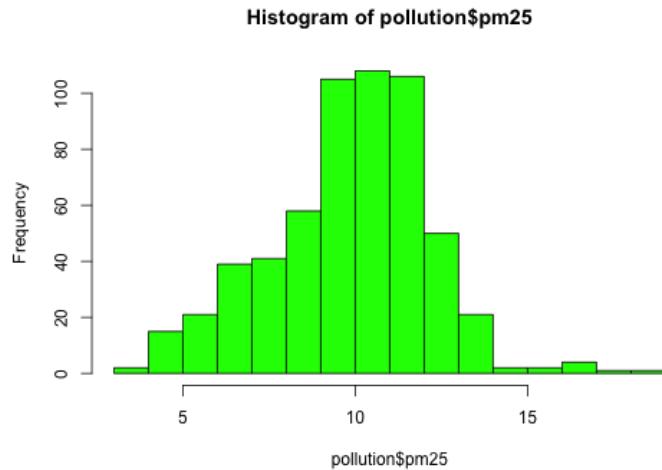
Boxplot

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



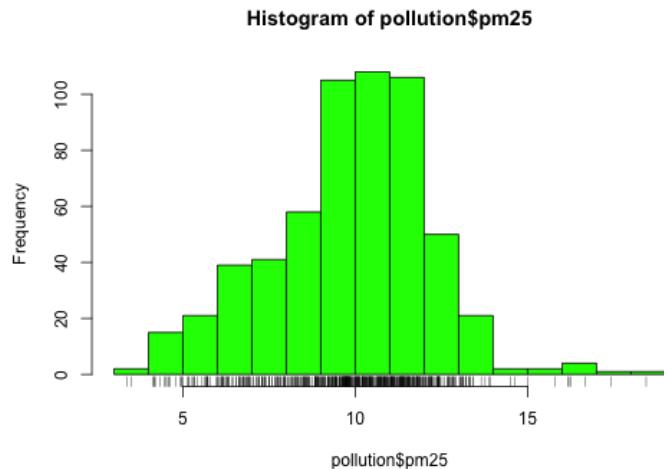
Histogram

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



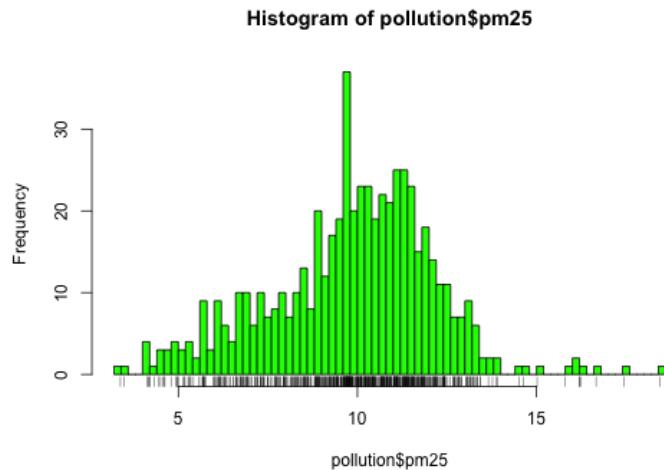
Histogram

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



Histogram

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

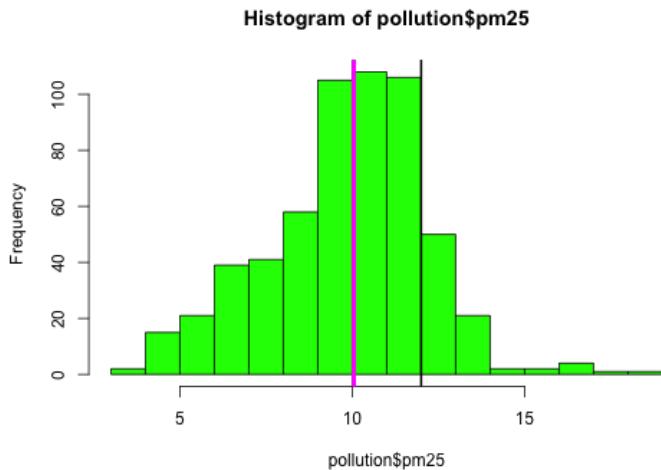


Overlaying Features

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

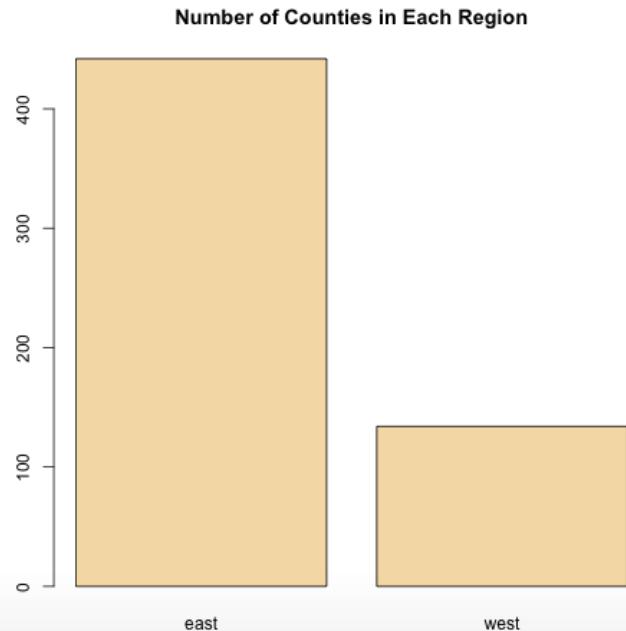
Overlaying Features

```
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

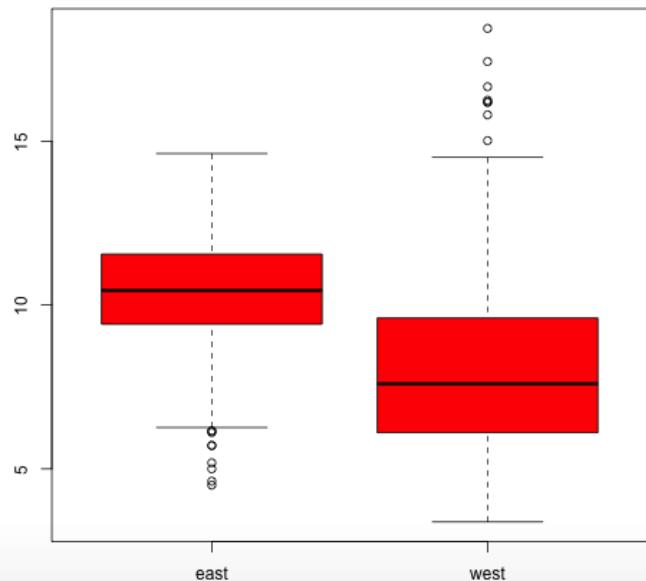
- Multiple/overlaid 1-D plots (Lattice/ggplot2)
- Scatterplots
- Smooth scatterplots

> 2 dimensions

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

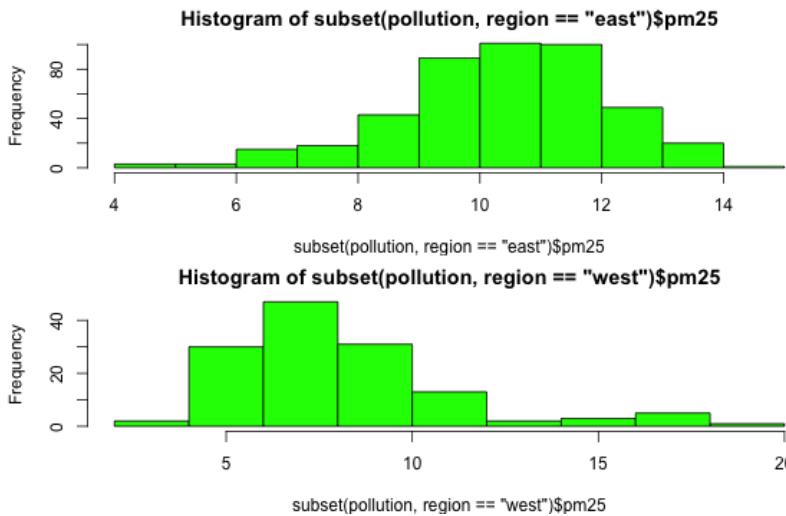
Multiple Boxplots

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



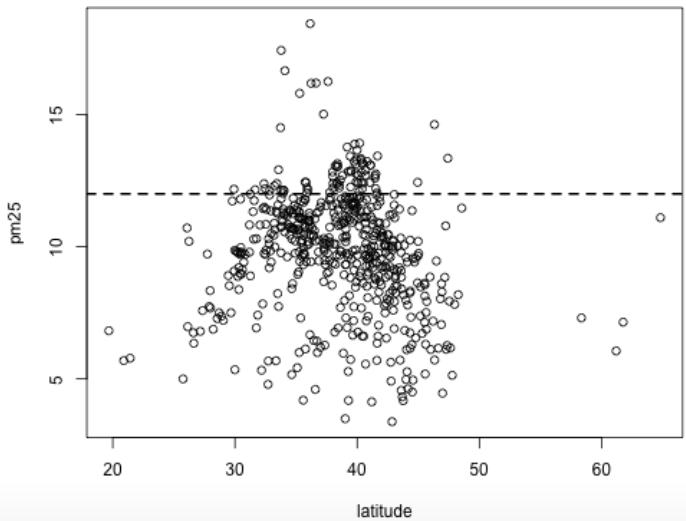
Multiple Histograms

```
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")
```



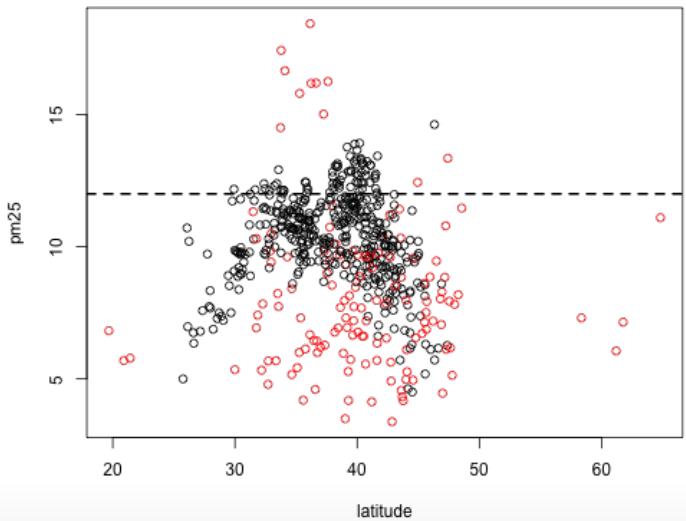
Scatterplot

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



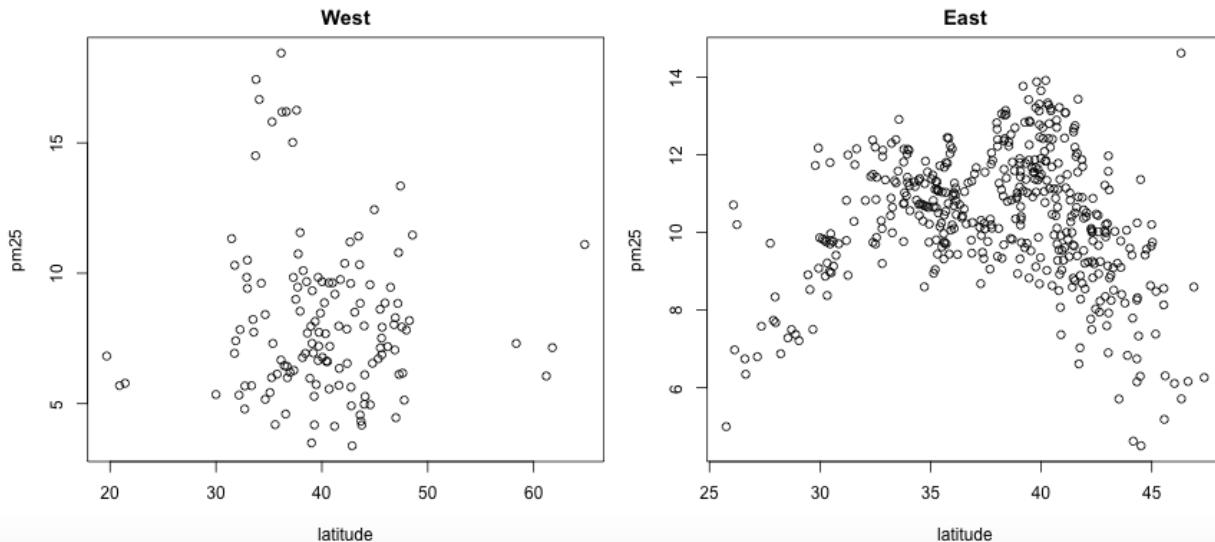
Scatterplot - Using Color

```
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](#)



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 points, does not resize well
- **jpeg**: good for photographs or natural scenes, uses lossy compression, good for plotting 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



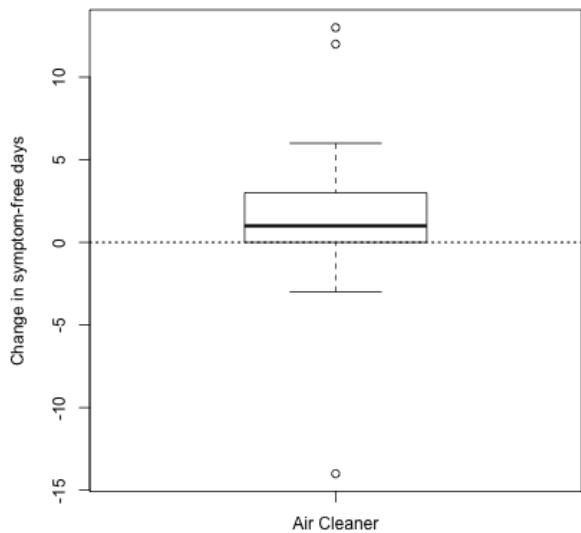
Principles of Analytic Graphics

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

Principles of Analytic Graphics

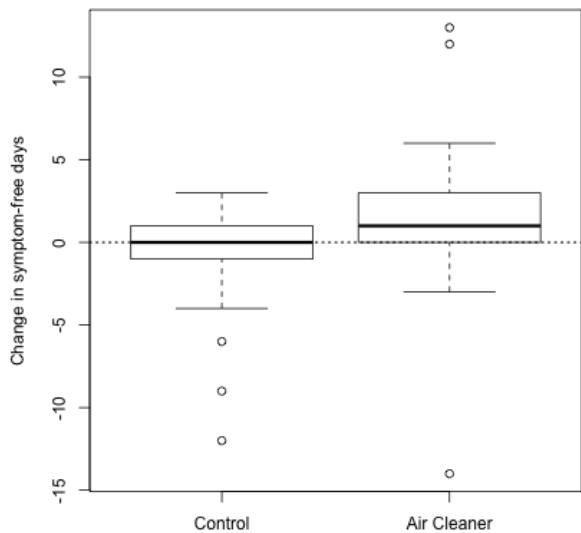
- Principle 1: Show comparisons
 - Evidence for a hypothesis is always *relative* to another competing hypothesis.
 - Always ask "Compared to What?"

Show Comparisons



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

Show Comparisons

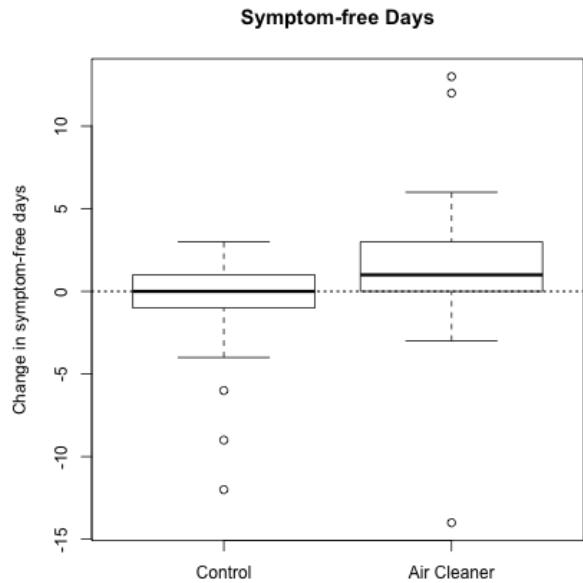


Reference: Butz AM, et al., *JAMA Pediatrics*, 2011.

Principles of Analytic Graphics

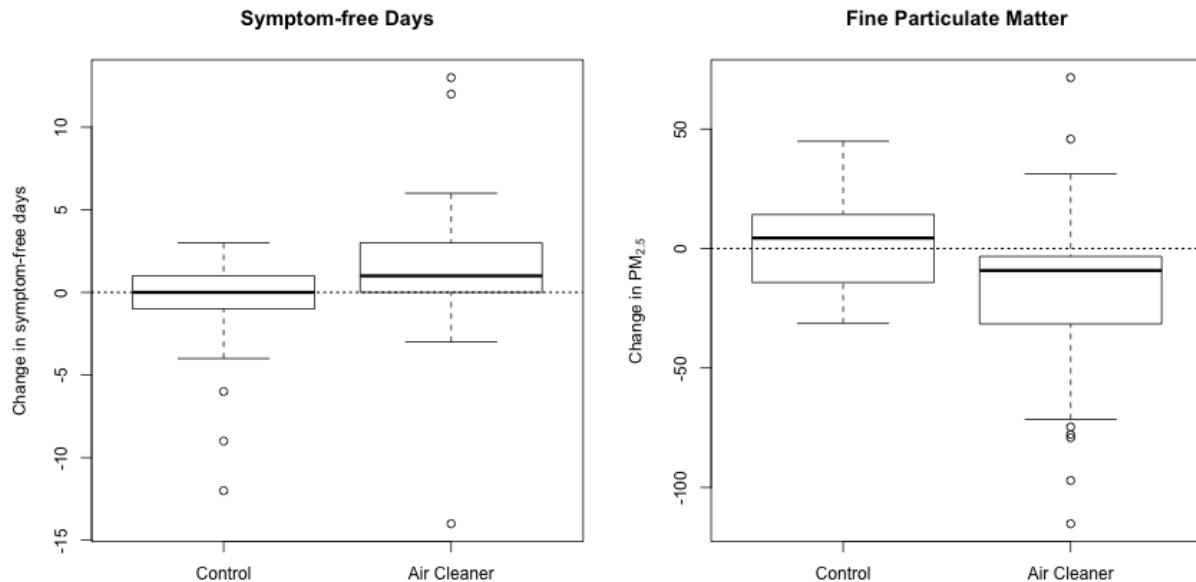
- Principle 1: Show comparisons
 - Evidence for a hypothesis is always *relative* to another competing hypothesis.
 - Always ask "Compared to What?"
- Principle 2: Show causality, mechanism, explanation, systematic structure
 - What is your causal framework for thinking about a question?

Show causality, mechanism



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

Show causality, mechanism

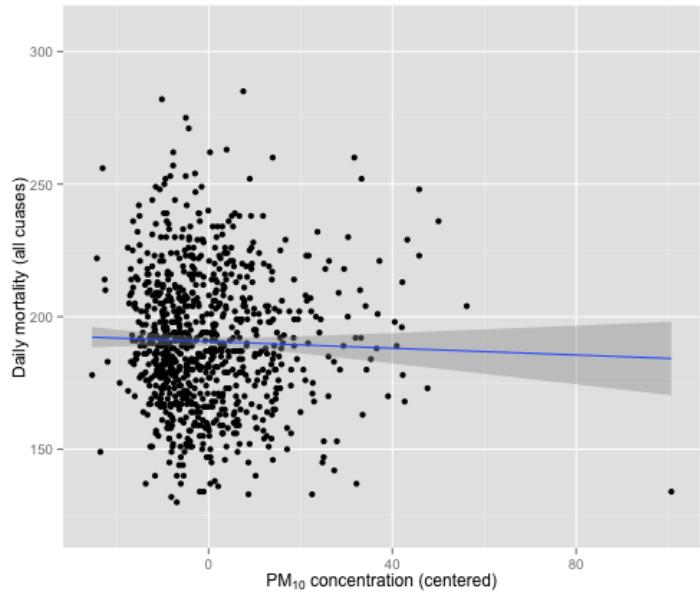


Reference: Butz AM, et al., *JAMA Pediatrics*, 2011.

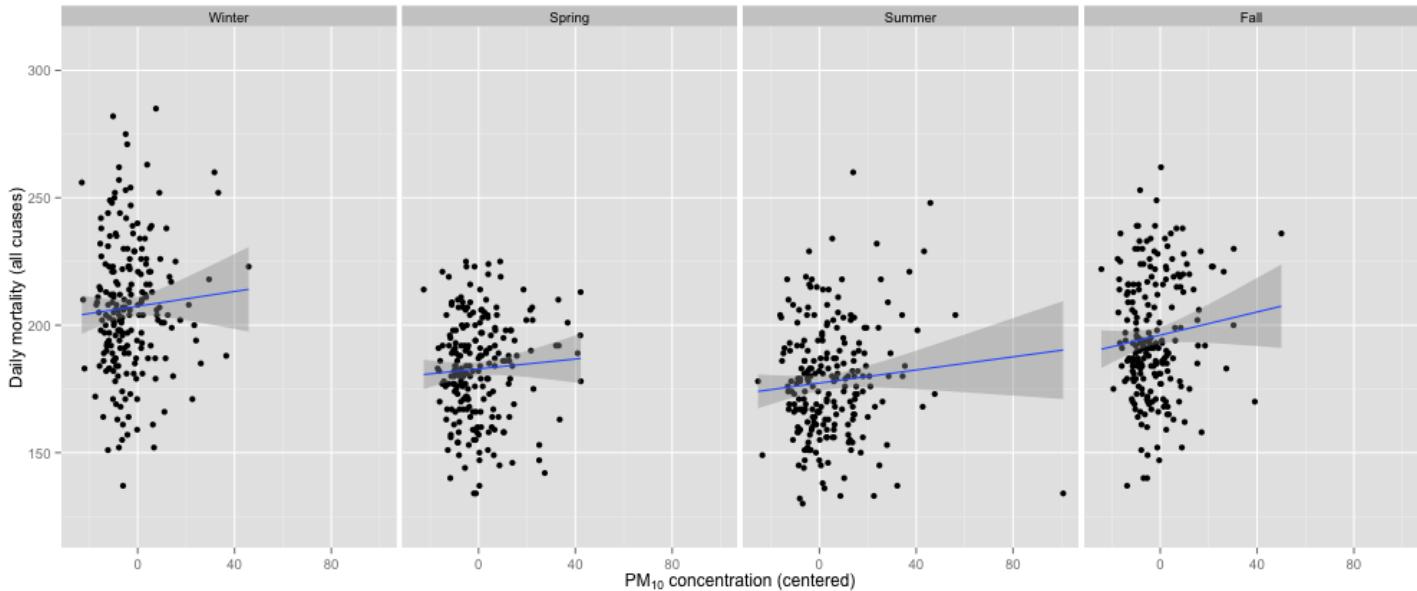
Principles of Analytic Graphics

- 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

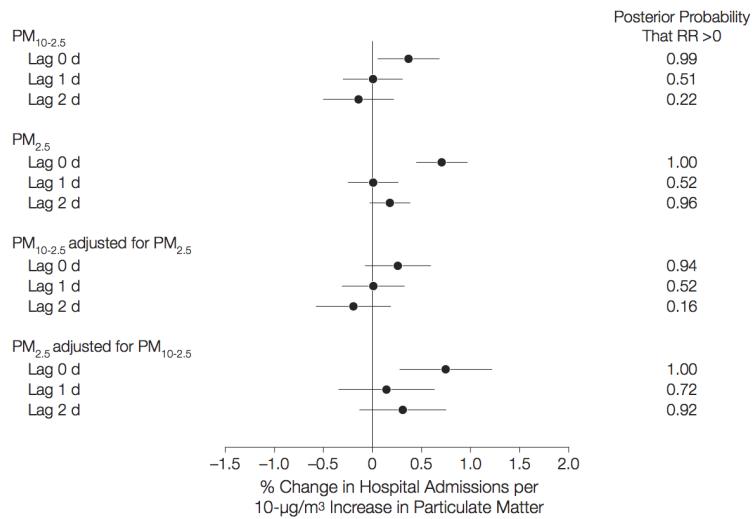


Principles of Analytic Graphics

- 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- $\mu\text{g}/\text{m}^3$ 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; PM₁₀, 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



Hierarchical Clustering

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

Can we find things that are close together?

Clustering organizes things that are **close** into groups

- How do we define close?
- How do we group things?
- How do we visualize the grouping?
- How do we interpret the grouping?

Hugely important/impactful

The screenshot shows a Google Scholar search results page for the query "cluster analysis". The search bar at the top contains "cluster analysis". Below the search bar, there are tabs for "Web", "Images", and "More...". On the right side of the header, there is a user profile for "jteek@gmail.com". The main search results area has a heading "Scholar" and displays "About 2,860,000 results (0.04 sec)". There are two main search results listed:

- Cluster analysis for applications**
MR Andeberg - 1973 - DTIC Document
Abstract: Cluster analysis is a collective term covering a wide variety of techniques for delineating natural groups or clusters in data sets. This book integrates the necessary elements of data analysis, cluster analysis, and computer implementation to cover the ...
Cited by 5438 Related articles All 12 versions Cite More▼
- Cluster analysis and display of genome-wide expression patterns**
MB Eisen, PT Spellman, PO Brown... - Proceedings of the ..., 1998 - National Acad Sciences
Abstract A system of cluster analysis for genome-wide expression data from DNA microarray hybridization is described that uses standard statistical algorithms to arrange genes according to similarity in pattern of gene expression. The output is displayed graphically, ...
Cited by 12537 Related articles BL Direct All 259 versions Cite [HTML] from nih.gov

On the left sidebar, there are filters and settings:

- Articles
- Legal documents
- Any time
- Since 2013
- Since 2012
- Since 2009
- Custom range...
- Sort by relevance
- Sort by date
- Include patents
- Include citations
- Create alert

http://scholar.google.com/scholar?hl=en&q=cluster+analysis&btnG=&as_sdt=1%2C21&as_sdtp=

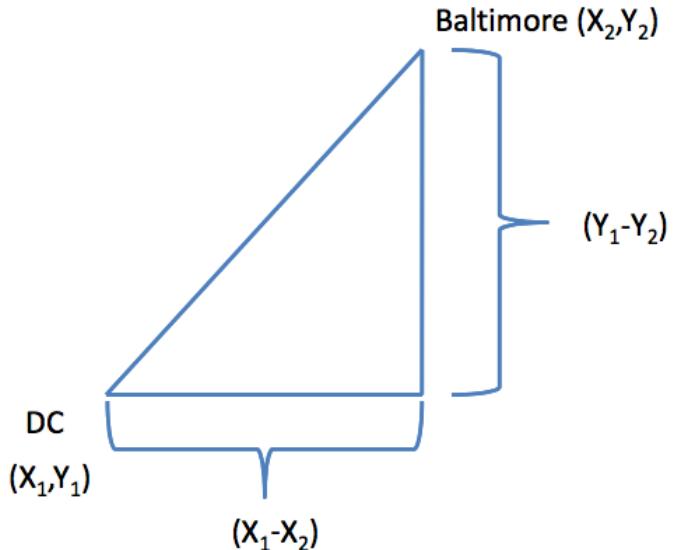
Hierarchical clustering

- An agglomerative approach
 - Find closest two things
 - Put them together
 - Find next closest
- Requires
 - A defined distance
 - A merging approach
- Produces
 - A tree showing how close things are to each other

How do we define close?

- Most important step
 - Garbage in -> garbage out
- Distance or similarity
 - Continuous - euclidean distance
 - Continuous - correlation similarity
 - Binary - manhattan distance
- Pick a distance/similarity that makes sense for your problem

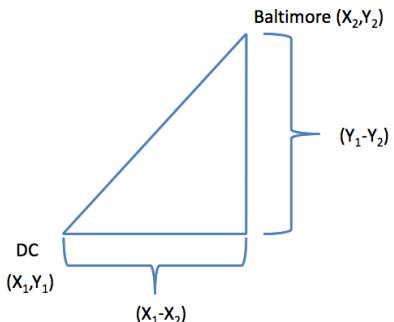
Example distances - Euclidean



<http://rafalab.jhsph.edu/688/lec/lecture5-clustering.pdf>

Example distances - Euclidean

$$\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$$

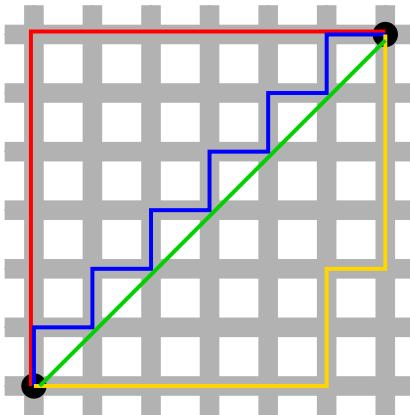


In general:

$$\sqrt{(A_1 - A_2)^2 + (B_1 - B_2)^2 + \dots + (Z_1 - Z_2)^2}$$

<http://rafalab.jhsph.edu/688/lec/lecture5-clustering.pdf>

Example distances - Manhattan



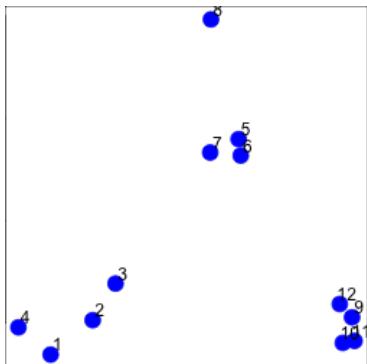
In general:

$$|A_1 - A_2| + |B_1 - B_2| + \dots + |Z_1 - Z_2|$$

http://en.wikipedia.org/wiki/Taxicab_geometry

Hierarchical clustering - example

```
set.seed(1234)
par(mar = c(0, 0, 0, 0))
x <- rnorm(12, mean = rep(1:3, each = 4), sd = 0.2)
y <- rnorm(12, mean = rep(c(1, 2, 1), each = 4), sd = 0.2)
plot(x, y, col = "blue", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
```



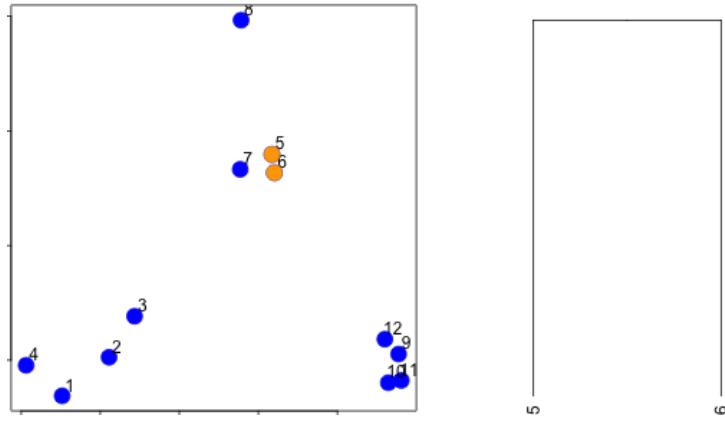
Hierarchical clustering - dist

- Important parameters: $x, method$

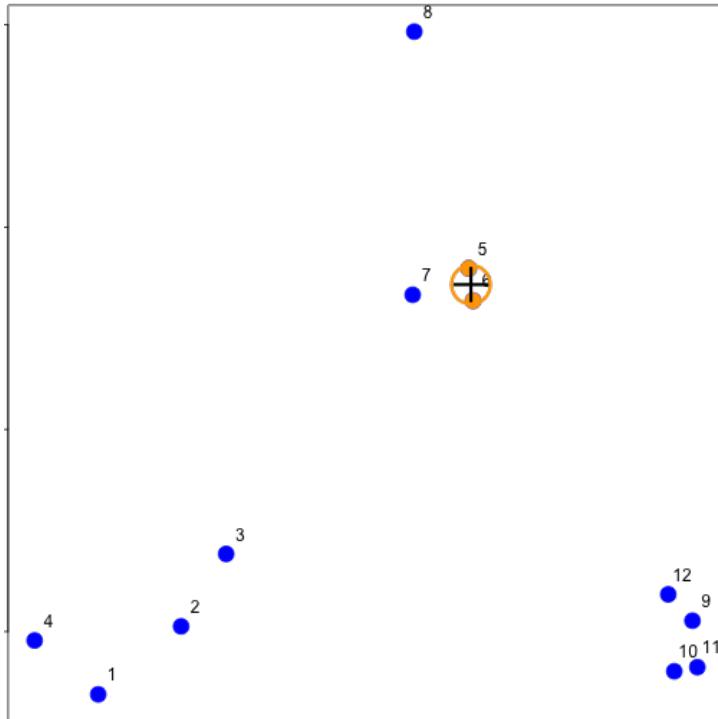
```
dataFrame <- data.frame(x = x, y = y)  
dist(dataFrame)
```

```
##           1         2         3         4         5         6         7         8         9  
## 2  0.34121  
## 3  0.57494 0.24103  
## 4  0.26382 0.52579 0.71862  
## 5  1.69425 1.35818 1.11953 1.80667  
## 6  1.65813 1.31960 1.08339 1.78081 0.08150  
## 7  1.49823 1.16621 0.92569 1.60132 0.21110 0.21667  
## 8  1.99149 1.69093 1.45649 2.02849 0.61704 0.69792 0.65063  
## 9  2.13630 1.83168 1.67836 2.35676 1.18350 1.11500 1.28583 1.76461  
## 10 2.06420 1.76999 1.63110 2.29239 1.23848 1.16550 1.32063 1.83518 0.14090  
## 11 2.14702 1.85183 1.71074 2.37462 1.28154 1.21077 1.37370 1.86999 0.11624  
## 12 2.05664 1.74663 1.58659 2.27232 1.07701 1.00777 1.17740 1.66224 0.10849  
##           10        11  
## 2  
## 3
```

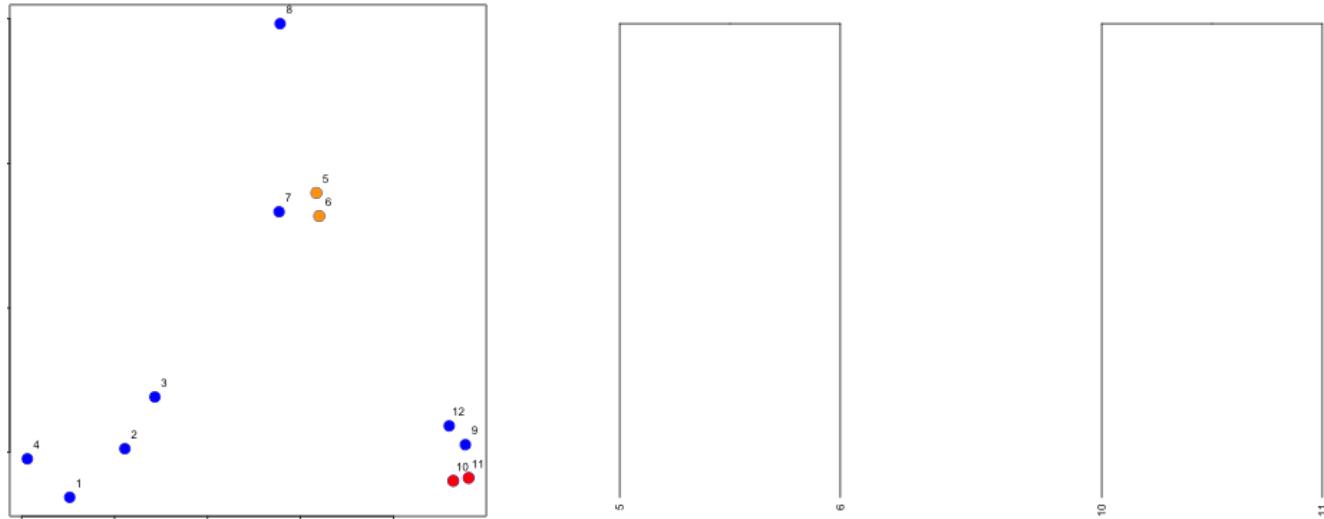
Hierarchical clustering - #1



Hierarchical clustering - #2

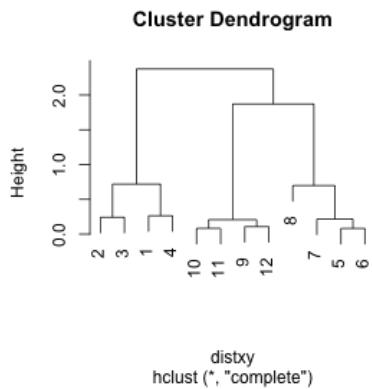


Hierarchical clustering - #3



Hierarchical clustering - hclust

```
dataFrame <- data.frame(x = x, y = y)
distxy <- dist(dataFrame)
hClustering <- hclust(distxy)
plot(hClustering)
```

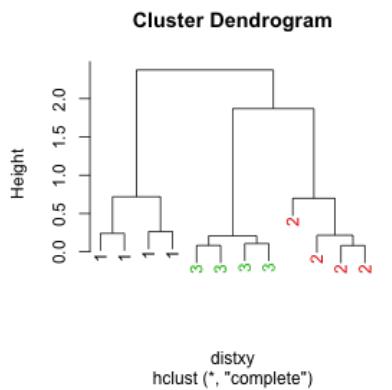


Prettier dendograms

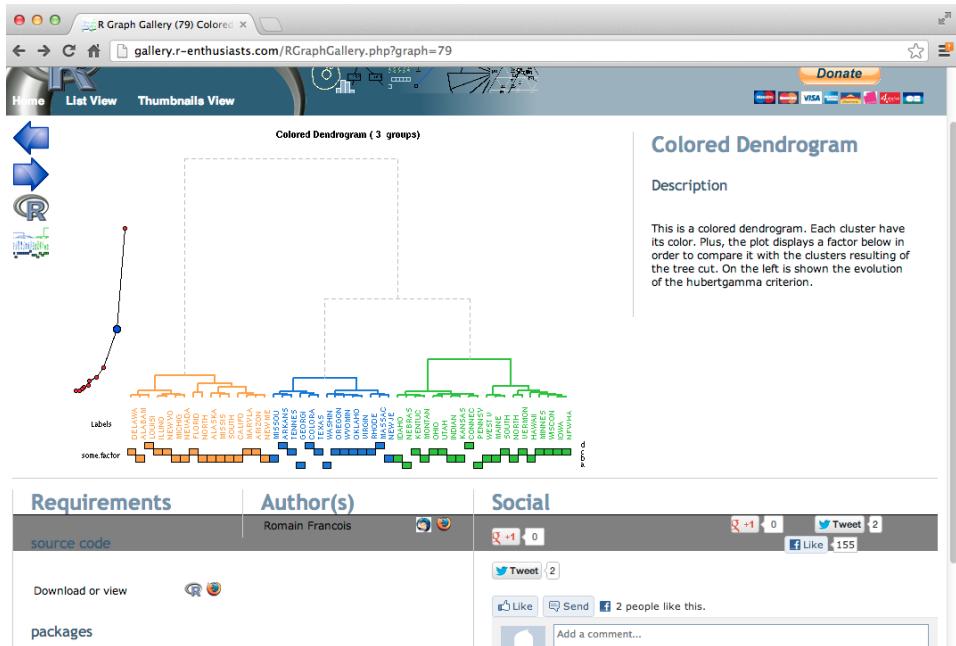
```
myplclust <- function(hclust, lab = hclust$labels, lab.col = rep(1, length(hclust$labels)),
hang = 0.1, ...) {
## modification of plclust for plotting hclust objects *in colour*! Copyright
## Eva KF Chan 2009 Arguments: hclust: hclust object lab: a character vector
## of labels of the leaves of the tree lab.col: colour for the labels;
## NA=default device foreground colour hang: as in hclust & plclust Side
## effect: A display of hierarchical cluster with coloured leaf labels.
y <- rep(hclust$height, 2)
x <- as.numeric(hclust$merge)
y <- y[which(x < 0)]
x <- x[which(x < 0)]
x <- abs(x)
y <- y[order(x)]
x <- x[order(x)]
plot(hclust, labels = FALSE, hang = hang, ...)
text(x = x, y = y[hclust$order] - (max(hclust$height) * hang), labels = lab[hclust$order],
col = lab.col[hclust$order], srt = 90, adj = c(1, 0.5), xpd = NA, ...)
}
```

Pretty dendograms

```
dataFrame <- data.frame(x = x, y = y)
distxy <- dist(dataFrame)
hClustering <- hclust(distxy)
myplclust(hClustering, lab = rep(1:3, each = 4), lab.col = rep(1:3, each = 4))
```

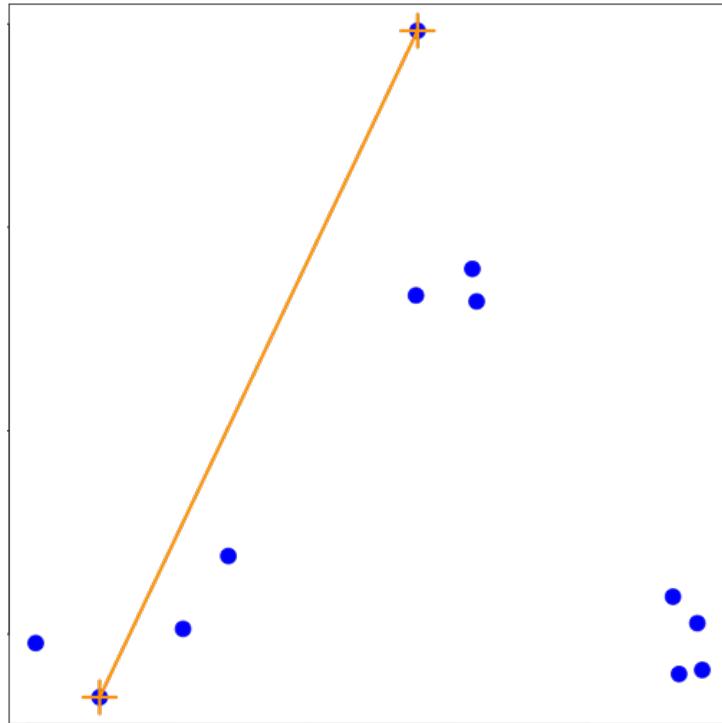


Even Prettier dendrograms

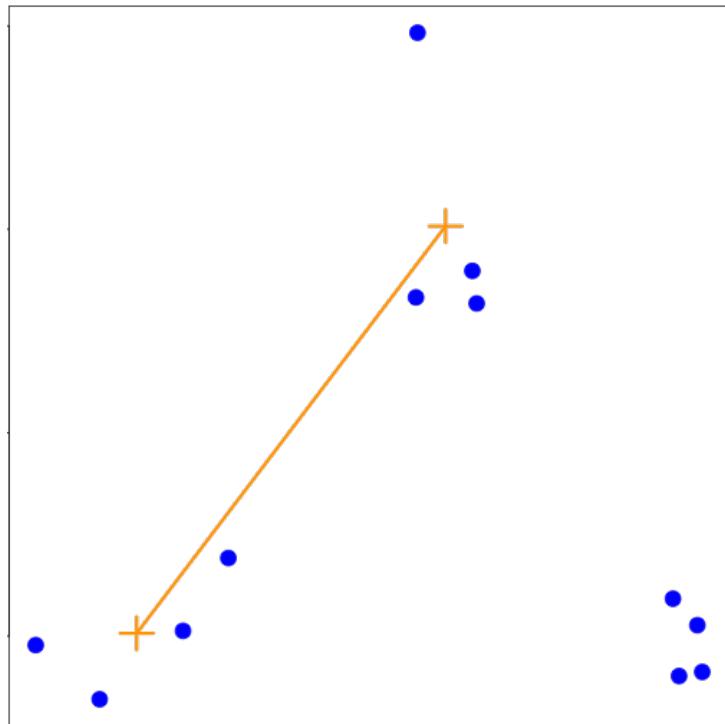


<http://gallery.r-enthusiasts.com/RGraphGallery.php?graph=79>

Merging points - complete

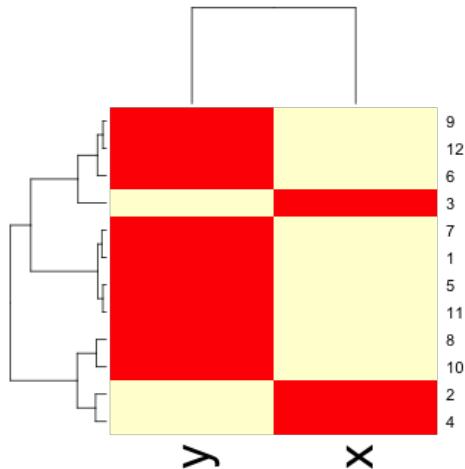


Merging points - average



heatmap()

```
dataFrame <- data.frame(x = x, y = y)
set.seed(143)
dataMatrix <- as.matrix(dataFrame)[sample(1:12), ]
heatmap(dataMatrix)
```



Notes and further resources

- Gives an idea of the relationships between variables/observations
- The picture may be unstable
 - Change a few points
 - Have different missing values
 - Pick a different distance
 - Change the merging strategy
 - Change the scale of points for one variable
- But it is deterministic
- Choosing where to cut isn't always obvious
- Should be primarily used for exploration
- [Rafa's Distances and Clustering Video](#)
- [Elements of statistical learning](#)



K-means Clustering

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

Can we find things that are close together?

- How do we define close?
- How do we group things?
- How do we visualize the grouping?
- How do we interpret the grouping?

How do we define close?

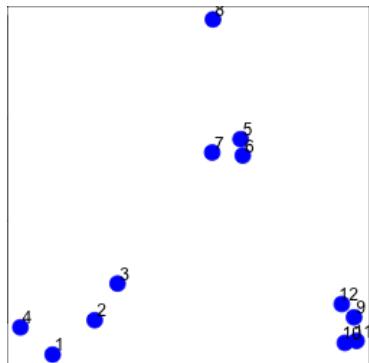
- Most important step
 - Garbage in → garbage out
- Distance or similarity
 - Continuous - euclidean distance
 - Continuous - correlation similarity
 - Binary - manhattan distance
- Pick a distance/similarity that makes sense for your problem

K-means clustering

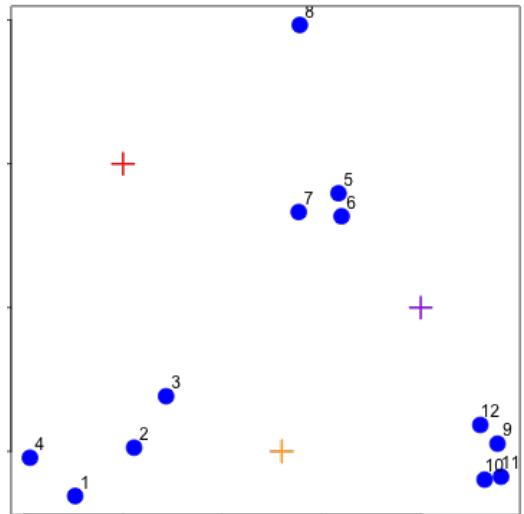
- A partitioning approach
 - Fix a number of clusters
 - Get "centroids" of each cluster
 - Assign things to closest centroid
 - Recalculate centroids
- Requires
 - A defined distance metric
 - A number of clusters
 - An initial guess as to cluster centroids
- Produces
 - Final estimate of cluster centroids
 - An assignment of each point to clusters

K-means clustering - example

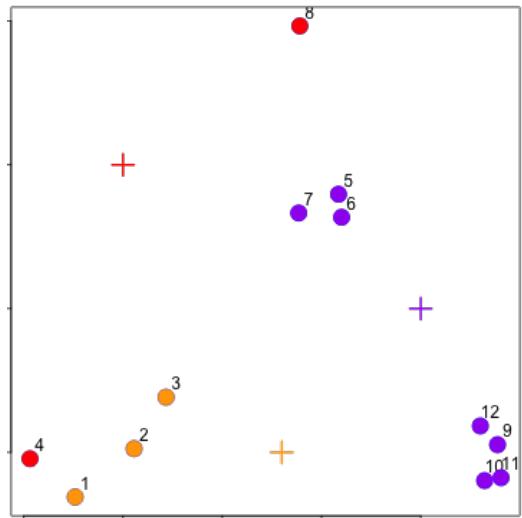
```
set.seed(1234)
par(mar = c(0, 0, 0, 0))
x <- rnorm(12, mean = rep(1:3, each = 4), sd = 0.2)
y <- rnorm(12, mean = rep(c(1, 2, 1), each = 4), sd = 0.2)
plot(x, y, col = "blue", pch = 19, cex = 2)
text(x + 0.05, y + 0.05, labels = as.character(1:12))
```



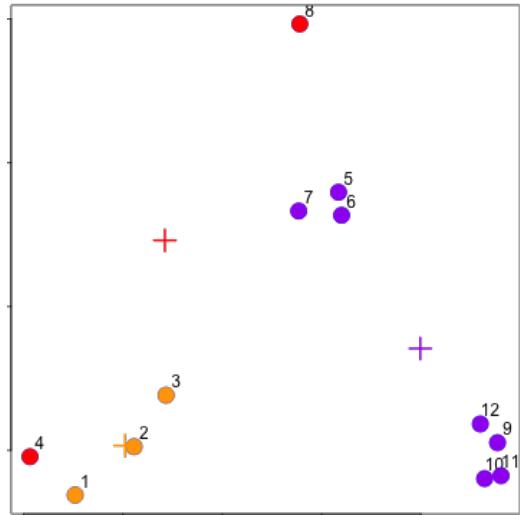
K-means clustering - starting centroids



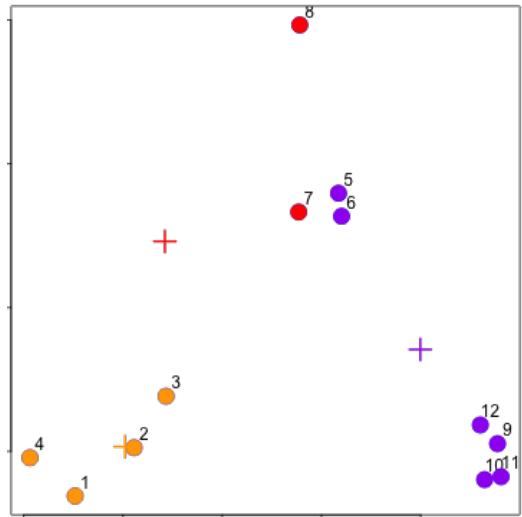
K-means clustering - assign to closest centroid



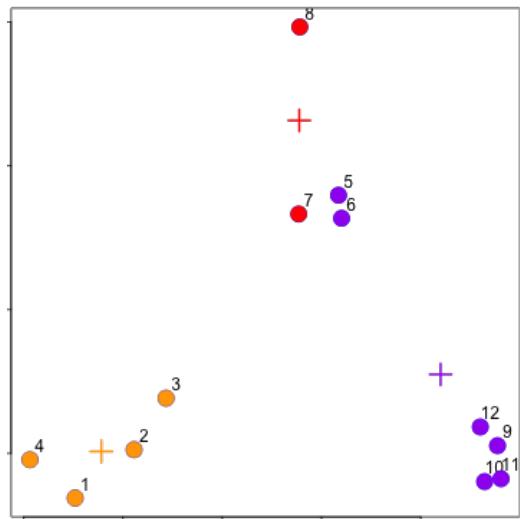
K-means clustering - recalculate centroids



K-means clustering - reassign values



K-means clustering - update centroids



kmeans()

- Important parameters: *x*, *centers*, *iter.max*, *nstart*

```
dataFrame <- data.frame(x, y)
kmeansObj <- kmeans(dataFrame, centers = 3)
names(kmeansObj)
```

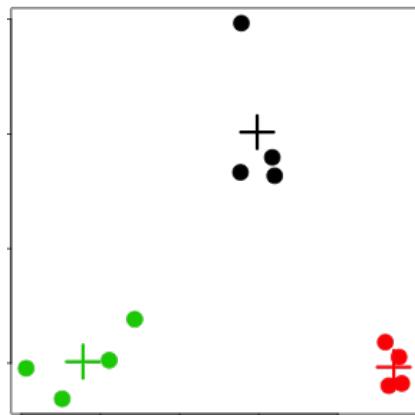
```
## [1] "cluster"      "centers"       "totss"        "withinss"
## [5] "tot.withinss" "betweenss"     "size"         "iter"
## [9] "ifault"
```

```
kmeansObj$cluster
```

```
## [1] 3 3 3 3 1 1 1 1 2 2 2 2
```

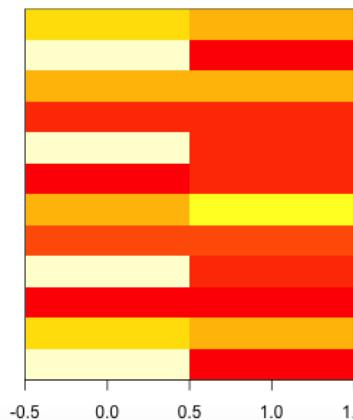
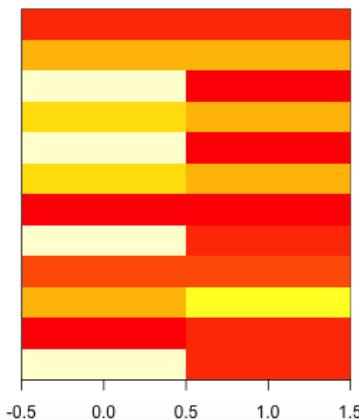
kmeans()

```
par(mar = rep(0.2, 4))
plot(x, y, col = kmeansObj$cluster, pch = 19, cex = 2)
points(kmeansObj$centers, col = 1:3, pch = 3, cex = 3, lwd = 3)
```



Heatmaps

```
set.seed(1234)
dataMatrix <- as.matrix(dataFrame)[sample(1:12), ]
kmeansObj2 <- kmeans(dataMatrix, centers = 3)
par(mfrow = c(1, 2), mar = c(2, 4, 0.1, 0.1))
image(t(dataMatrix)[, nrow(dataMatrix):1], yaxt = "n")
image(t(dataMatrix)[, order(kmeansObj2$cluster)], yaxt = "n")
```



Notes and further resources

- K-means requires a number of clusters
 - Pick by eye/intuition
 - Pick by cross validation/information theory, etc.
 - [Determining the number of clusters](#)
- K-means is not deterministic
 - Different # of clusters
 - Different number of iterations
- [Rafael Irizarry's Distances and Clustering Video](#)
- [Elements of statistical learning](#)



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 be 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**.

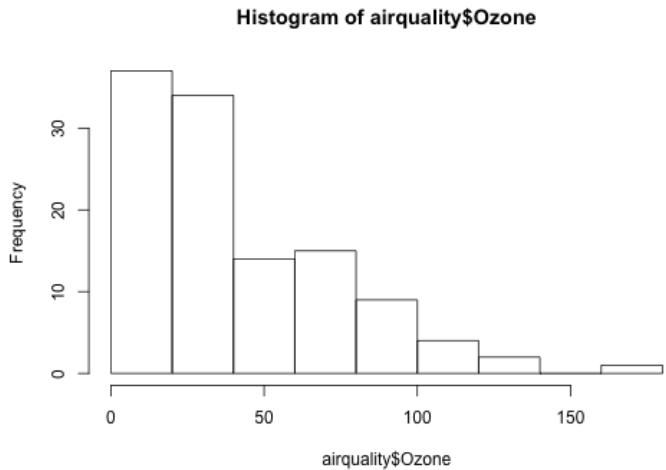
Base Graphics

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 be set and tweaked; these parameters are documented in `?par`; it wouldn't hurt to try to memorize this help page!

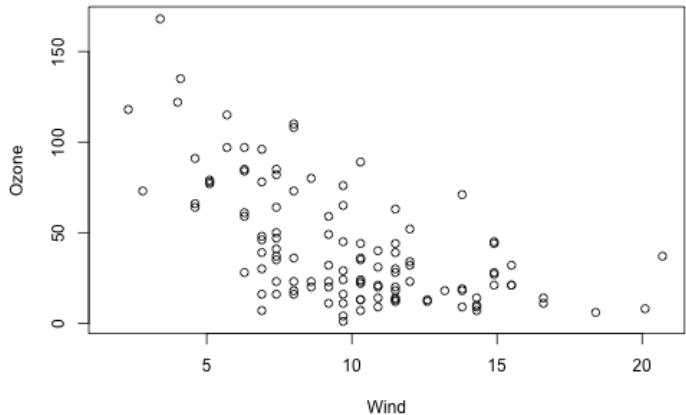
Simple Base Graphics: Histogram

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



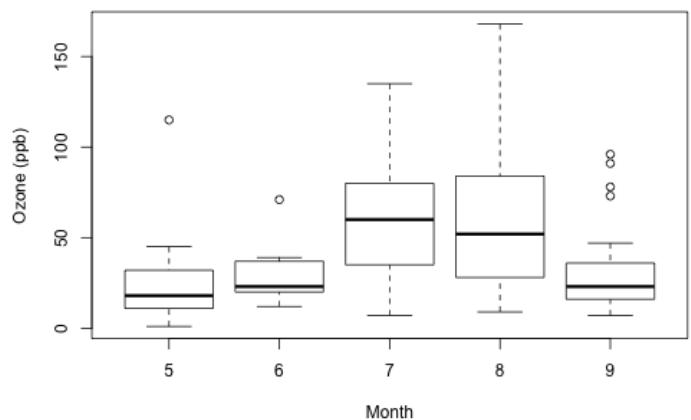
Simple Base Graphics: Scatterplot

```
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")
```

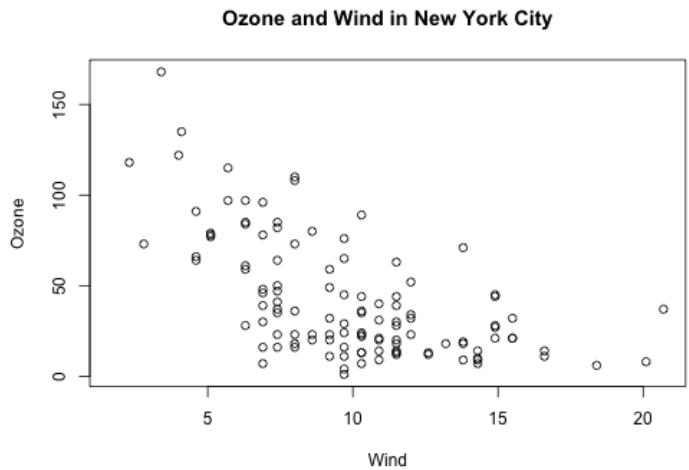
```
## [1] 1 1
```

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

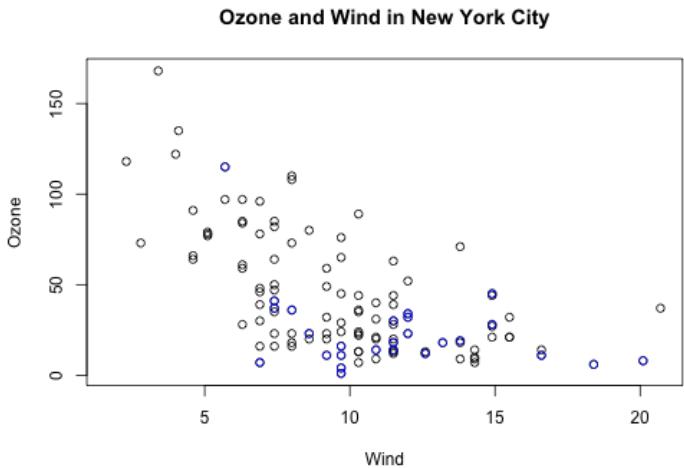
Base Plot with Annotation

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



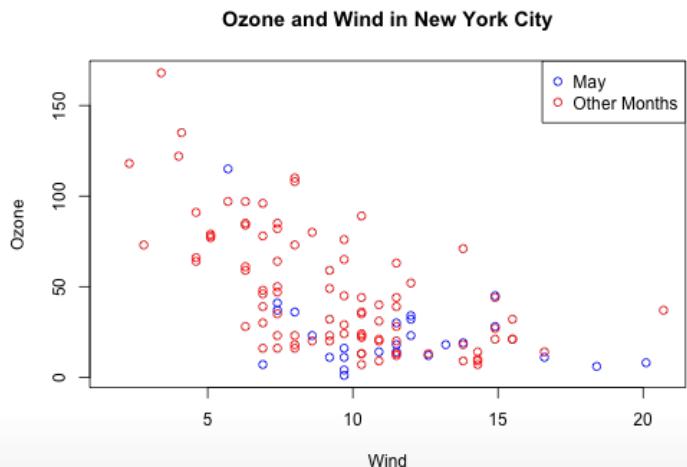
Base Plot with Annotation

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



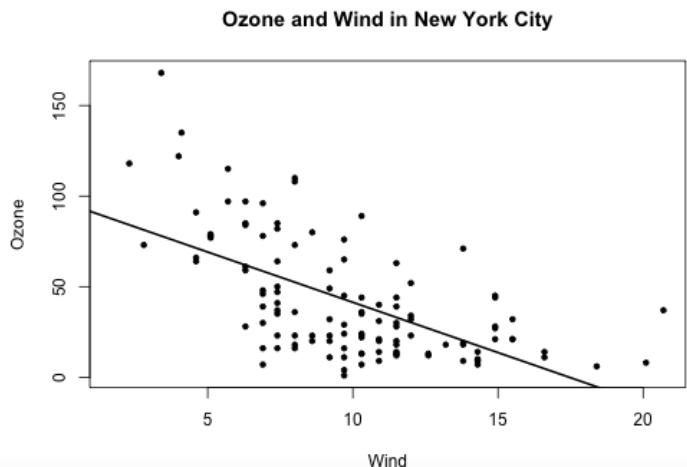
Base Plot with Annotation

```
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"))
```



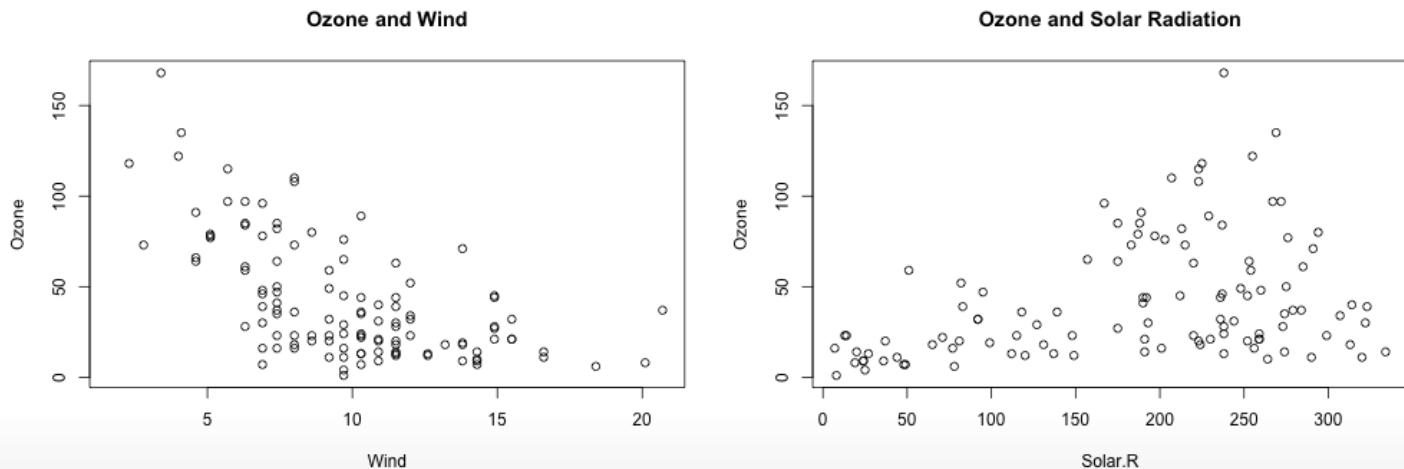
Base Plot with Regression Line

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



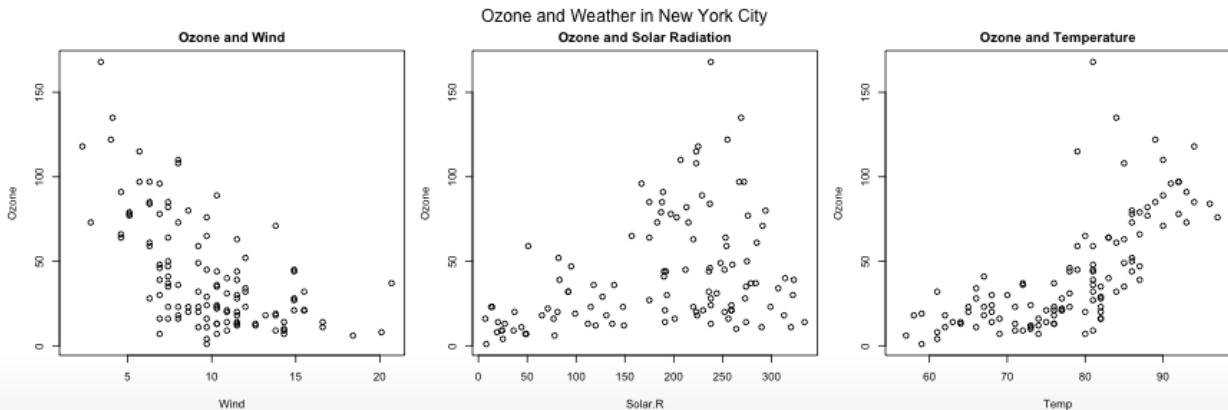
Multiple Base Plots

```
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



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

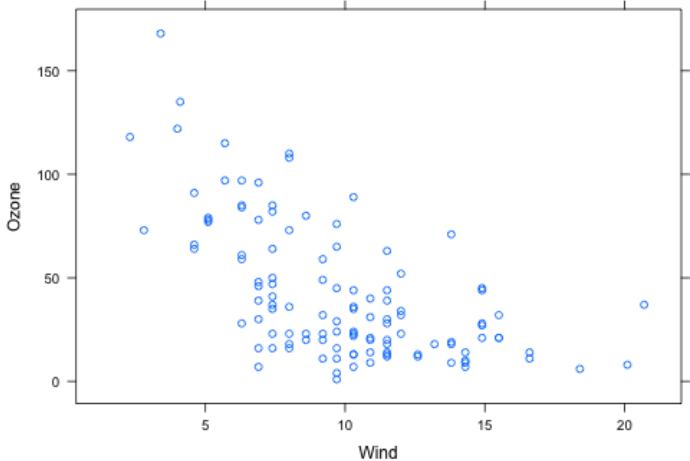
Lattice functions generally take a formula for their first argument, usually of the form

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

- We use the *formula notation* here, hence the \sim .
- On the left of the \sim 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.

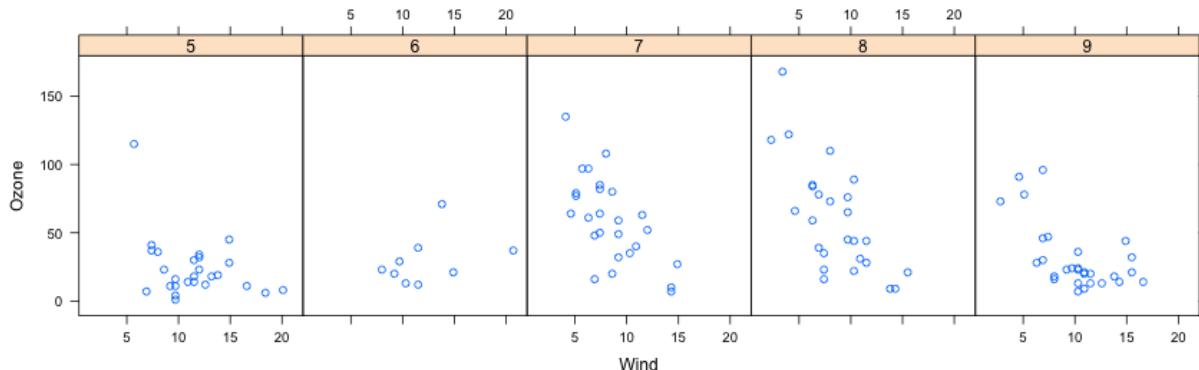
Simple Lattice Plot

```
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))
```



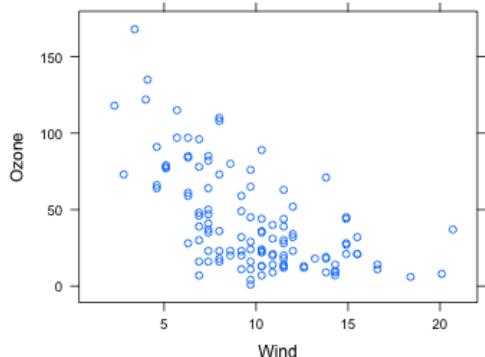
Lattice Behavior

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
```



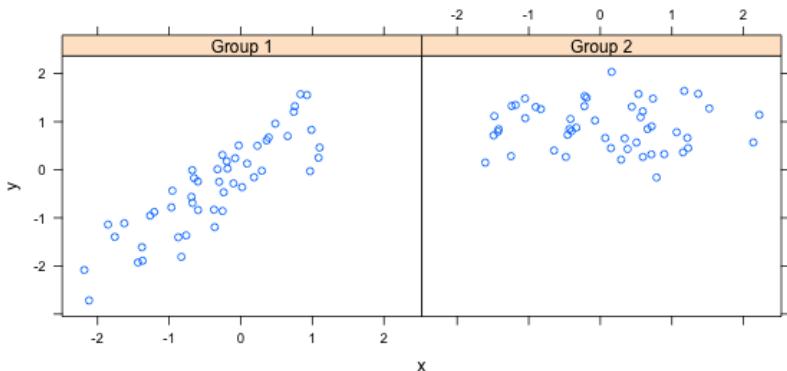
```
xyplot(Ozone ~ Wind, data = airquality) ## Auto-printing
```

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)

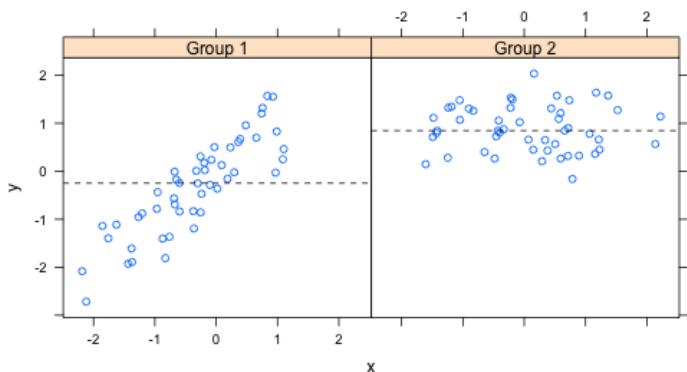
Lattice Panel Functions

```
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
```



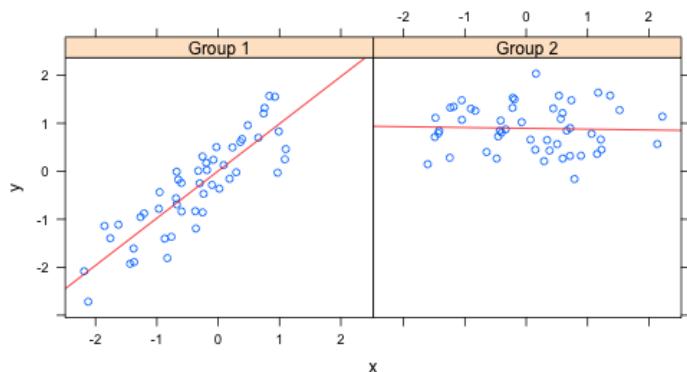
Lattice Panel Functions

```
## Custom panel function
xyplot(y ~ x | f, panel = function(x, y, ...) {
  panel.xyplot(x, y, ...)
  panel.abline(h = median(y), lty = 2)
})
```



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
})
```



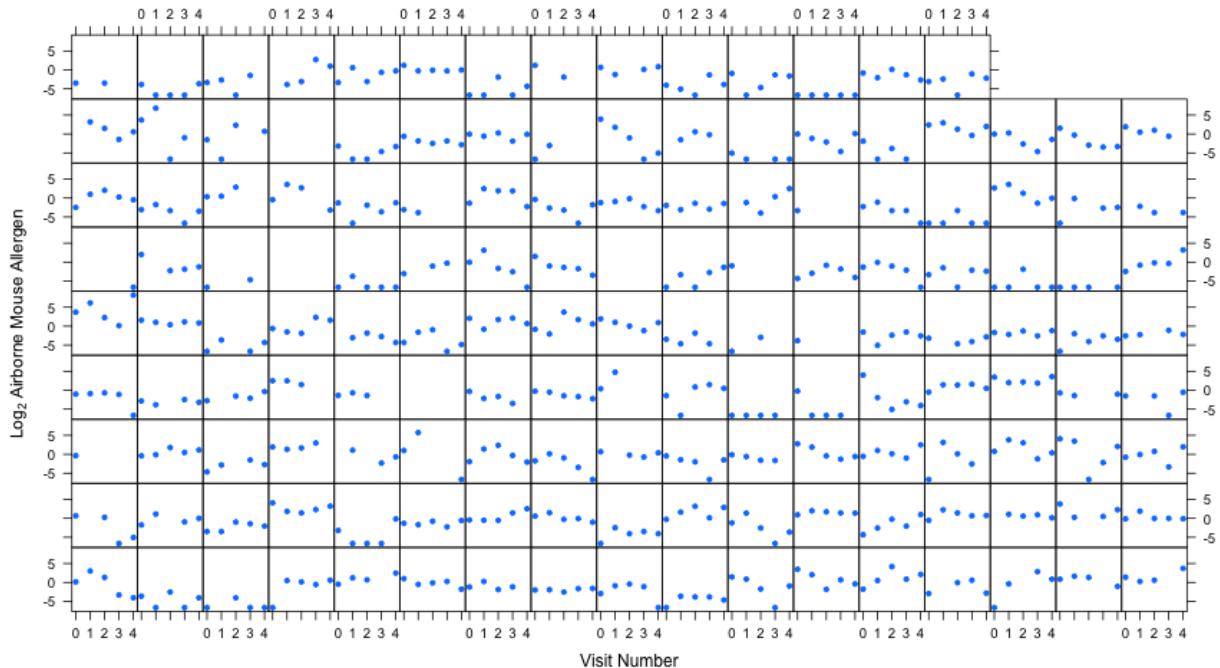
Many Panel Lattice Plot: Example from MAACS

- 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

Mouse Allergen and Asthma Cohort Study (Baltimore City)



Summary

- Lattice plots are constructed with a single function call to a core lattice function (e.g. `xypplot`)
- 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 - Math Functions

Computing for Data Analysis

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

Mathematical Annotation

R can produce LaTeX-like symbols on a plot for mathematical annotation. This is very handy and is useful for making fun of people who use other statistical packages.

- Math symbols are “expressions” in R and need to be wrapped in the `expression` function
- There is a set list of allowed symbols and this is documented in `?plotmath`
- Plotting functions that take arguments for text generally allow expressions for math symbols

Mathematical Annotation

Some examples.

```
plot(0, 0, main = expression(theta == 0),
     ylab = expression(hat(gamma) == 0),
     xlab = expression(sum(x[i] * y[i], i==1, n)))
```

Pasting strings together.

```
x <- rnorm(100)
hist(x,
      xlab=expression("The mean (" * bar(x) * ") is " *
                      sum(x[i]/n,i==1,n)))
```

Substituting

What if you want to use a computed value in the annotation?

```
x <- rnorm(100)
y <- x + rnorm(100, sd = 0.5)
plot(x, y,
      xlab=substitute(bar(x) == k, list(k=mean(x))),
      ylab=substitute(bar(y) == k, list(k=mean(y))))
    )
```

Or in a loop of plots

```
par(mfrow = c(2, 2))
for(i in 1:4) {
  x <- rnorm(100)
  hist(x, main=substitute(theta==num, list(num=i)))
}
```

Summary of Important Help Pages

- ?par
- ?plot
- ?xyplot
- ?plotmath
- ?axis



Plotting Systems in R

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

The Base Plotting System

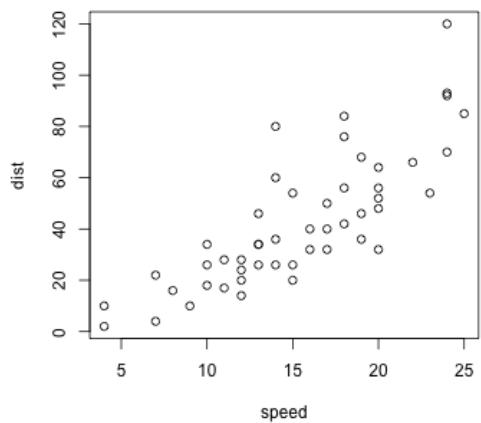
- "Artist's palette" model
- Start with blank canvas and build up from there
- Start with plot function (or similar)
- Use annotation functions to add/modify (`text, lines, points, axis`)

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

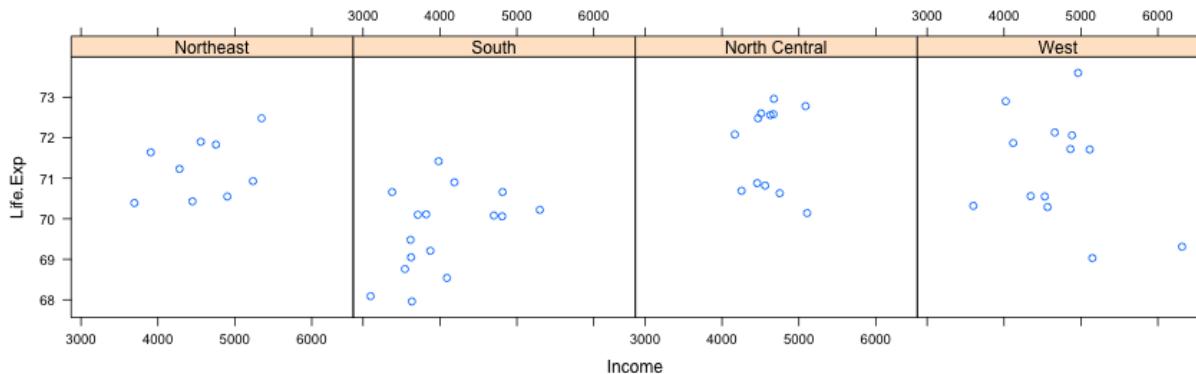
- Plots are created with a single function call (`xypplot`, `bwplot`, etc.)
- Most useful for conditioning types of plots: Looking at how y changes with x across levels of z
- Things like margins/spacing set automatically because entire plot is specified at once
- Good for putting many many plots on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

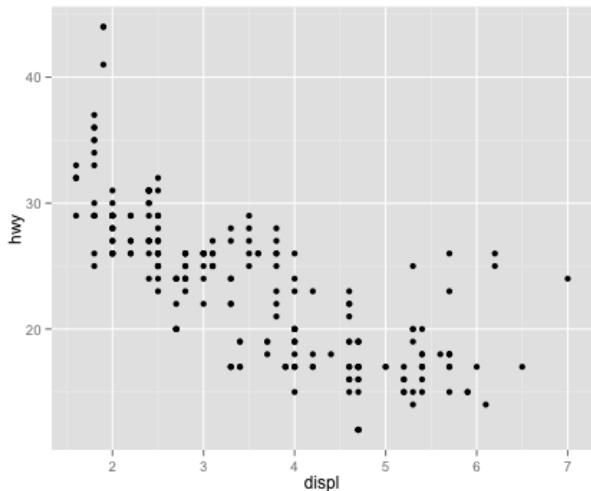


The ggplot2 System

- 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.

Feedback — Week 1 Quiz

[Help](#)

You submitted this quiz on **Fri 9 May 2014 12:45 PM PDT**. You got a score of **10.00** out of **10.00**.

Question 1

Which of the following is a principle of analytic graphics?

Your Answer	Score	Explanation
<input type="radio"/> Make judicious use of color in your scatterplots		
<input checked="" type="radio"/> Integrate multiple modes of evidence	✓ 1.00	
<input type="radio"/> Show box plots (univariate summaries)		
<input type="radio"/> Only do what your tools allow you to do		
<input type="radio"/> Don't plot more than two variables at at time		
Total	1.00 / 1.00	

Question 2

What is the role of exploratory graphs in data analysis?

Your Answer	Score	Explanation
<input checked="" type="radio"/> They are typically made very quickly.	✓ 1.00	
<input type="radio"/> They are made for formal presentations.		
<input type="radio"/> They are used in place of formal modeling.		
<input type="radio"/> Only a few are constructed.		
Total	1.00 / 1.00	

Question 3

Which of the following is true about the base plotting system?

Your Answer	Score	Explanation
<input checked="" type="radio"/> Plots are created and annotated with separate functions	✓ 1.00	Functions like 'plot' or 'hist' typically create the plot on the graphics device and functions like 'lines', 'text', or 'points' will annotate or add data to the plot.
<input type="radio"/> The system is most useful for conditioning plots		
<input type="radio"/> Margins and spacings are adjusted automatically depending on the type of plot and the data		
<input type="radio"/> Plots are typically created with a single function call		
Total	1.00 / 1.00	

Question 4

Which of the following is an example of a valid graphics device in R?

Your Answer	Score	Explanation
<input type="radio"/> The keyboard		
<input type="radio"/> A Microsoft Word document		
<input type="radio"/> A file folder		
<input checked="" type="radio"/> A PDF file	✓ 1.00	
Total	1.00 / 1.00	

Question 5

Which of the following is an example of a vector graphics device in R?

Your Answer	Score	Explanation
<input type="radio"/> PNG		
<input type="radio"/> GIF		
<input type="radio"/> TIFF		
<input checked="" type="radio"/> SVG	✓ 1.00	
Total	1.00 / 1.00	

Question 6

Bitmapped file formats can be most useful for

Your Answer	Score	Explanation
<input type="radio"/> Plots that are not scaled to a specific resolution		
<input type="radio"/> Plots that may need to be resized		
<input type="radio"/> Plots that require animation or interactivity		
<input checked="" type="radio"/> Scatterplots with many many points	✓ 1.00	
Total	1.00 / 1.00	

Question 7

Which of the following functions is typically used to add elements to a plot in the base graphics system?

Your Answer	Score	Explanation
<input checked="" type="radio"/> lines()	✓ 1.00	
<input type="radio"/> plot()		
<input type="radio"/> hist()		
<input type="radio"/> boxplot()		

Total

1.00 / 1.00

Question 8

Which function opens the screen graphics device for the Mac?

Your Answer**Score****Explanation** bitmap() quartz()

1.00

 pdf() png()

Total

1.00 / 1.00

Question 9

What does the 'pch' option to par() control?

Your Answer**Score****Explanation** the size of the plotting symbol in a scatterplot the orientation of the axis labels on the plot the plotting symbol/character in the base graphics system 1.00 the line width in the base graphics system

Total

1.00 / 1.00

Question 10

If I want to save a plot to a PDF file, which of the following is a correct way of doing that?

Your Answer**Score****Explanation**

- Open the PostScript device with postscript(), construct the plot, then close the device with dev.off().
- Construct the plot on the PNG device with png(), then copy it to a PDF with dev.copy2pdf().
- Open the screen device with quartz(), construct the plot, and then close the device with dev.off().
- Construct the plot on the screen device and then copy it to a PDF file with dev.copy2pdf() ✓ 1.00

Total	1.00 /
	1.00

Feedback — Week 2 Quiz

[Help](#)

You submitted this quiz on **Sun 18 May 2014 12:26 PM PDT**. You got a score of **10.00** out of **10.00**.

Question 1

Under the lattice graphics system, what do the primary plotting functions like xyplot() and bwplot() return?

Your Answer	Score	Explanation
-------------	-------	-------------

an object of class "plot"

an object of class "lattice"

an object of class "trellis"



1.00

nothing; only a plot is made

Total

1.00 / 1.00

Question 2

What is produced by the following code?

```
library(nlme)
library(lattice)
xyplot(weight ~ Time | Diet, BodyWeight)
```

Your Answer	Score	Explanation
-------------	-------	-------------

A set of 11 panels showing the relationship between weight and diet for each time.

A set of 16 panels showing the relationship between weight and time for each rat.

A set of 3 panels showing the relationship between weight and time for each rat.

- A set of 3 panels showing the relationship between weight and time for each diet.

✓ 1.00

Total

1.00 /

1.00

Question 3

Annotation of plots in any plotting system involves adding points, lines, or text to the plot, in addition to customizing axis labels or adding titles. Different plotting systems have different sets of functions for annotating plots in this way. Which of the following functions can be used to annotate the panels in a multi-panel lattice plot?

Your Answer**Score****Explanation** axis() points() panel.abline()

1.00

 lines()

Total

1.00 / 1.00

Question 4

The following code does NOT result in a plot appearing on the screen device.

```
library(lattice)
library(datasets)
data(airquality)
p <- xyplot(Ozone ~ Wind | factor(Month), data = airquality)
```

Which of the following is an explanation for why no plot appears?

Your Answer**Score****Explanation** The xyplot() function, by default, sends plots to the PDF device. There is a syntax error in the call to xyplot().

- The object 'p' has not yet been printed with the appropriate print method. ✓ 1.00

- The variables being plotted are not found in that dataset.

Total	1.00 /
	1.00

Question 5

In the lattice system, which of the following functions can be used to finely control the appearance of all lattice plots?

Your Answer	Score	Explanation
-------------	-------	-------------

`splom()`

`par()`

`trellis.par.set()` ✓ 1.00

`print.trellis()`

Total	1.00 / 1.00
-------	-------------

Question 6

What is ggplot2 an implementation of?

Your Answer	Score	Explanation
-------------	-------	-------------

the Grammar of Graphics developed by Leland Wilkinson ✓ 1.00

a 3D visualization system

the base plotting system in R

the S language originally developed by Bell Labs

Total	1.00 / 1.00
-------	-------------

Question 7

Load the `airquality` dataset from the datasets package in R.

```
library(datasets)  
data(airquality)
```

I am interested in examining how the relationship between ozone and wind speed varies across each month. What would be the appropriate code to visualize that using ggplot2?

Your Answer	Score	Explanation
<input checked="" type="radio"/> airquality = transform(airquality, Month = factor(Month)) qplot(Wind, Ozone, data = airquality, facets = . ~ Month)	✓ 1.00	
<input type="radio"/> qplot(Wind, Ozone, data = airquality, facets = . ~ factor(Month))		
<input type="radio"/> qplot(Wind, Ozone, data = airquality)		
<input type="radio"/> qplot(Wind, Ozone, data = airquality, geom = "smooth")		
Total	1.00 / 1.00	

Question 8

What is a **geom** in the ggplot2 system?

Your Answer	Score	Explanation
<input type="radio"/> a method for mapping data to attributes like color and size		
<input type="radio"/> a statistical transformation		
<input checked="" type="radio"/> a plotting object like point, line, or other shape	✓ 1.00	
<input type="radio"/> a method for making conditioning plots		
Total	1.00 / 1.00	

Question 9

When I run the following code I get an error:

```
library(ggplot2)
g <- ggplot(movies, aes(votes, rating))
print(g)
```

I was expecting a scatterplot of 'votes' and 'rating' to appear. What's the problem?

Your Answer	Score	Explanation
<input type="radio"/> There is a syntax error in the call to ggplot.		
<input type="radio"/> The dataset is too large and hence cannot be plotted to the screen.		
<input type="radio"/> The object 'g' does not have a print method.		
<input checked="" type="radio"/> ggplot does not yet know what type of layer to add to the plot.	✓ 1.00	
Total	1.00 / 1.00	

Question 10

The following code creates a scatterplot of 'votes' and 'rating' from the movies dataset in the ggplot2 package. After loading the ggplot2 package with the library() function, I can run

```
qplot(votes, rating, data = movies)
```

How can I modify the the code above to add a loess smoother to the scatterplot?

Your Answer	Score	Explanation
<input type="radio"/> qplot(votes, rating, data = movies) + stats_smooth("loess")		
<input type="radio"/> qplot(votes, rating, data = movies, smooth = "loess")		
<input checked="" type="radio"/> qplot(votes, rating, data = movies) + geom_smooth()	✓ 1.00	
<input type="radio"/> qplot(votes, rating, data = movies, panel = panel.loess)		

Total	1.00 /
	1.00

[Peer Assessments](https://class.coursera.org/exdata-002/human_grading/) (https://class.coursera.org/exdata-002/human_grading/) / Course Project 1

[Help](https://class.coursera.org/exdata-002/help/peergrading?url=https%3A%2F%2Fclass.coursera.org%2Fexdata-002%2Fhuman_grading%2Fview%2Fcourses%2F972082%2Fassessments%2F3%2Fresults%2Fmine) (https://class.coursera.org/exdata-002/help/peergrading?url=https%3A%2F%2Fclass.coursera.org%2Fexdata-002%2Fhuman_grading%2Fview%2Fcourses%2F972082%2Fassessments%2F3%2Fresults%2Fmine)

Submission Phase

1. Do assignment (/exdata-002/human_grading/view/courses/972082/assessments/3/submissions)

Evaluation Phase

2. Evaluate peers (/exdata-002/human_grading/view/courses/972082/assessments/3/peerGradingSets)

Results Phase

3. See results (/exdata-002/human_grading/view/courses/972082/assessments/3/results/mine)

Your effective grade is **12**

A 20% penalty has been applied because you did not complete the entire evaluation portion of the assessment.

Your unadjusted grade is 15, which is simply the grade you received from your peers.

See below for details.

Introduction

This assignment uses data from the [UC Irvine Machine Learning Repository](http://archive.ics.uci.edu/ml/) (<http://archive.ics.uci.edu/ml/>), a popular repository for machine learning datasets. In particular, we will be using the “Individual household electric power consumption Data Set” which I have made available on the course web site:

- **Dataset:** [Electric power consumption](https://d396qusza40orc.cloudfront.net/exdata%2Fdata%2Fhousehold_power_consumption.zip)
(https://d396qusza40orc.cloudfront.net/exdata%2Fdata%2Fhousehold_power_consumption.zip)
[20Mb]
- **Description:** Measurements of electric power consumption in one household with a one-minute sampling rate over a period of almost 4 years. Different electrical quantities and some sub-metering values are available.

The following descriptions of the 9 variables in the dataset are taken from the [UCI web site](http://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption) ([https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption](http://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption)):

1. **Date:** Date in format dd/mm/yyyy
2. **Time:** time in format hh:mm:ss
3. **Global_active_power:** household global minute-averaged active power (in kilowatt)

4. **Global_reactive_power**: household global minute-averaged reactive power (in kilowatt)
5. **Voltage**: minute-averaged voltage (in volt)
6. **Global_intensity**: household global minute-averaged current intensity (in ampere)
7. **Sub_metering_1**: energy sub-metering No. 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas powered).
8. **Sub_metering_2**: energy sub-metering No. 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing-machine, a tumble-drier, a refrigerator and a light.
9. **Sub_metering_3**: energy sub-metering No. 3 (in watt-hour of active energy). It corresponds to an electric water-heater and an air-conditioner.

Loading the data

When loading the dataset into R, please consider the following:

- The dataset has 2,075,259 rows and 9 columns. First calculate a rough estimate of how much memory the dataset will require in memory before reading into R. Make sure your computer has enough memory (most modern computers should be fine).
- We will only be using data from the dates 2007-02-01 and 2007-02-02. One alternative is to read the data from just those dates rather than reading in the entire dataset and subsetting to those dates.
- You may find it useful to convert the Date and Time variables to Date/Time classes in R using the `strptime()` and `as.Date()` functions.
- Note that in this dataset missing values are coded as `?`.

Making Plots

Our overall goal here is simply to examine how household energy usage varies over a 2-day period in February, 2007. Your task is to reconstruct the following plots below, all of which were constructed using the base plotting system.

First you will need to fork and clone the following GitHub repository:

https://github.com/rdpeng/ExData_Plotting1 (https://github.com/rdpeng/ExData_Plotting1)

For each plot you should

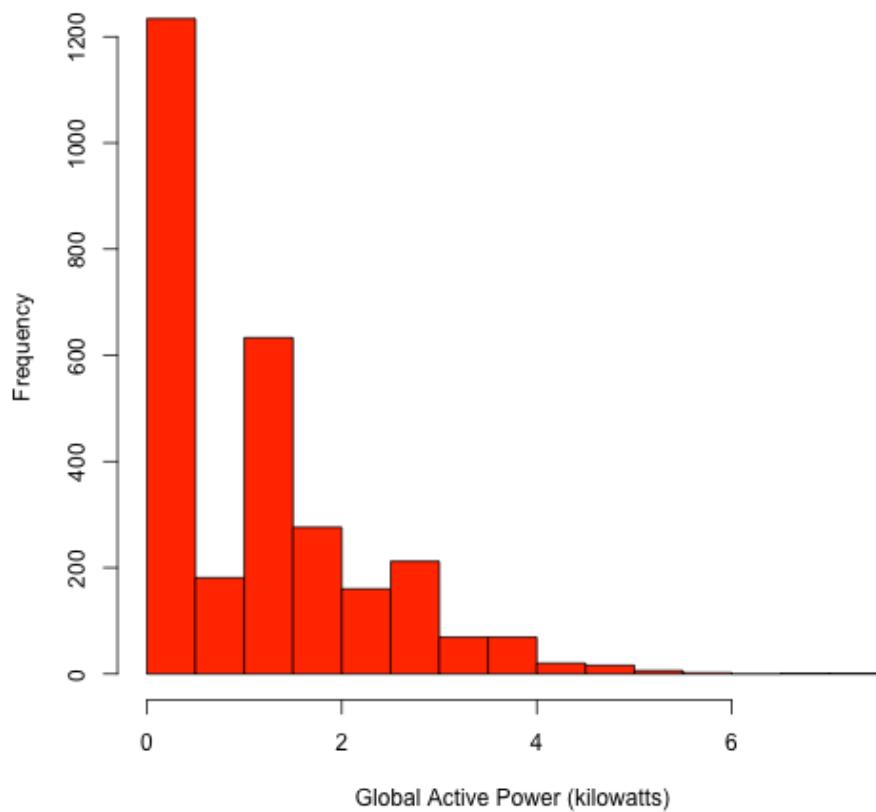
- Construct the plot and save it to a PNG file with a width of 480 pixels and a height of 480 pixels.
- Name each of the plot files as `plot1.png`, `plot2.png`, etc.
- Create a separate R code file (`plot1.R`, `plot2.R`, etc.) that constructs the corresponding plot, i.e. code in `plot1.R` constructs the `plot1.png` plot. Your code file **should include code for reading the data** so that the plot can be fully reproduced. You should also include the code that creates the PNG file.
- Add the PNG file and R code file to your git repository

When you are finished with the assignment, push your git repository to GitHub so that the GitHub version of your repository is up to date. There should be four PNG files and four R code files.

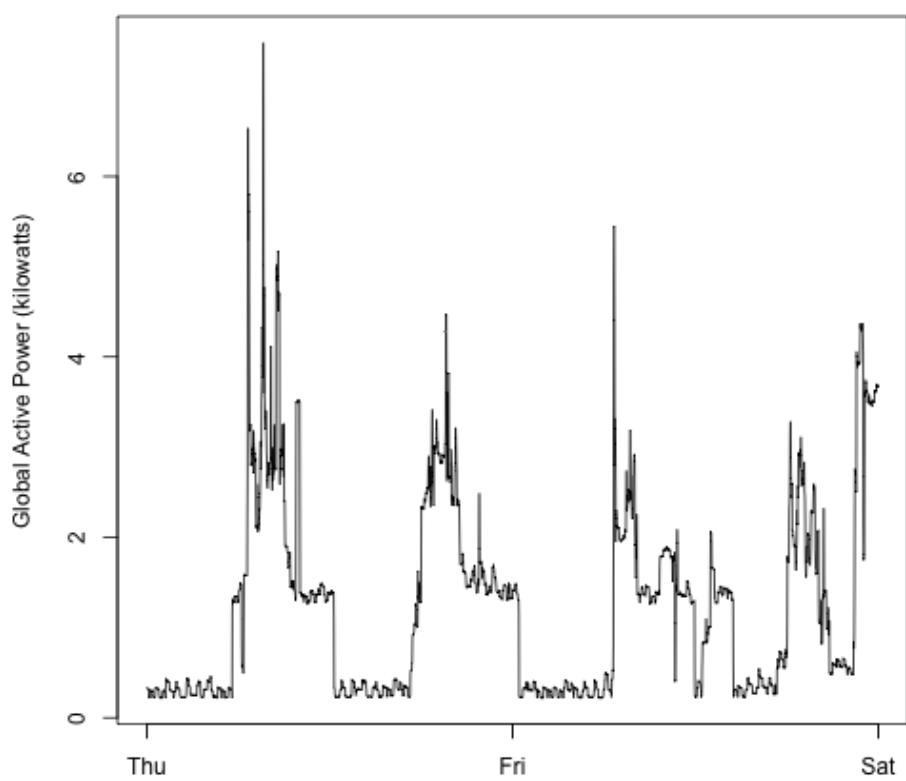
The four plots that you will need to construct are shown below.

Plot 1

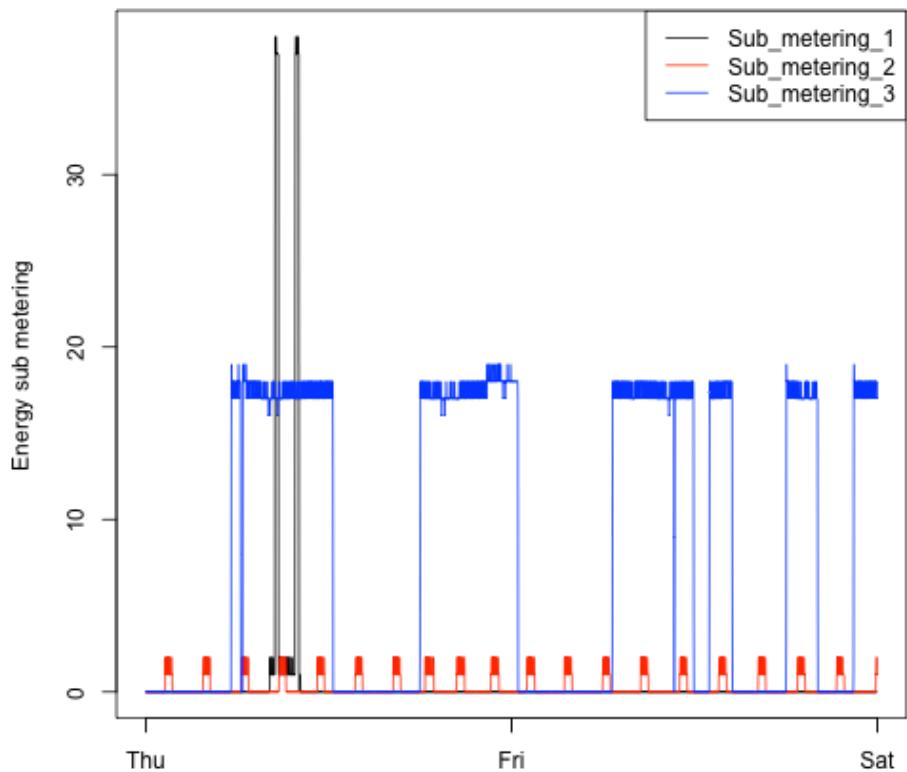
Global Active Power



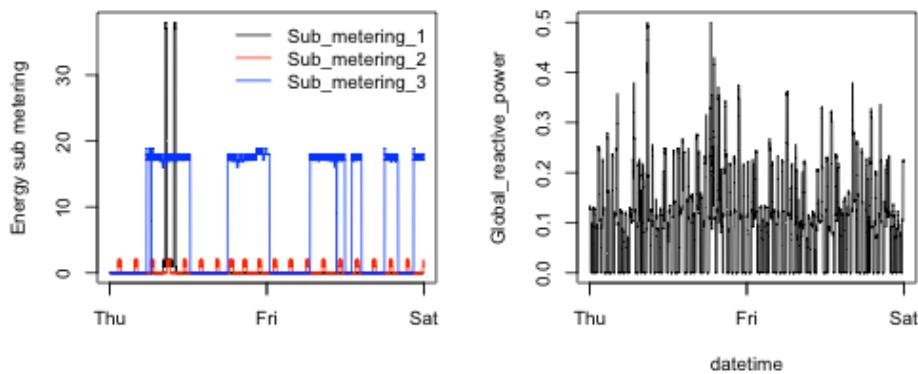
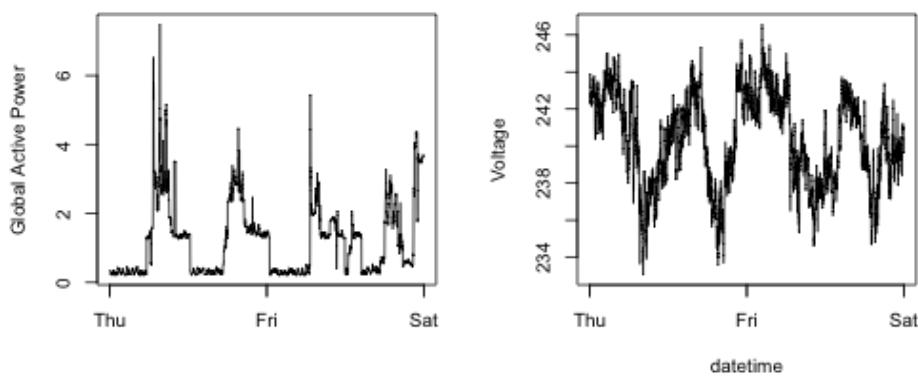
Plot 2



Plot 3



Plot 4



Please submit the URL pointing to your GitHub repository containing the completed R code for this assignment.

<https://github.com/sandipan/coursera/tree/exploratorydataanalysis/project1>
<https://github.com/sandipan/coursera/tree/exploratorydataanalysis/project1>

Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

Was a valid GitHub URL containing a git repository submitted?

- 0 points: A valid GitHub URL was NOT submitted (or URL is broken)
1 point: The submitted URL points to a GitHub repository

Score from your peers: **1**

Does the GitHub repository contain at least one commit beyond the original fork?

- 0 points: No, there are no commits beyond the original fork
1 point: Yes, there is at least one commit beyond the original fork

Score from your peers: **1**

Overall evaluation/feedback

Note: this section can only be filled out during the evaluation phase.

Please examine the plot files in the GitHub repository. Do the plot files appear to be of the correct graphics file format?

- 0 points:** No, at least one of the files appears to be in the wrong format
1 point: Yes, all of the files appear to be in the correct format

Score from your peers: **1**

Please view the image file for Plot 1 from the GitHub repository. Does the plot appear correct?

0 points: No, the plot appears incorrect in at least 1 major discrepancy from the reference plot (e.g. wrong data), or at least 2 minor discrepancies (e.g. x-label is incorrect, title is incorrect), or the plot was not viewable

1 point: The plot is mostly correct with at most one minor discrepancy from the reference plot

Score from your peers: 1

Please evaluate the code for Plot 1, but **do not run the code** on your computer. Does the code appear to create the plot reference plot given in the assignment?

0 points: The code does not create the reference plot, or is not viewable/present in repository

1 point: The code is mostly correct, but does not reproduce the reference plot exactly

2 points: The code reproduces the reference plot exactly

Score from your peers: 2

Please view the image file for Plot 2 from the GitHub repository. Does the plot appear correct?

0 points: No, the plot appears incorrect in at least 1 major discrepancy from the reference plot (e.g. wrong data), or at least 2 minor discrepancies (e.g. x-label is incorrect, title is incorrect), or the plot was not viewable

1 point: The plot is mostly correct with at most one minor discrepancy from the reference plot

Score from your peers: 1

Please evaluate the code for Plot 2, but **do not run the code** on your computer. Does the code appear to create the plot reference plot given in the assignment?

0 points: The code does not create the reference plot, or is not viewable/present in repository

1 point: The code is mostly correct, but does not reproduce the reference plot exactly

2 points: The code reproduces the reference plot exactly

Score from your peers: 2

Please view the image file for Plot 3 from the GitHub repository. Does the plot appear correct?

0 points: No, the plot appears incorrect in at least 1 major discrepancy from the reference plot (e.g. wrong data), or at least 2 minor discrepancies (e.g. x-label is incorrect, title is incorrect), or the plot was not viewable

1 point: The plot is mostly correct with at most one minor discrepancy from the reference plot

Score from your peers: 1

Please evaluate the code for Plot 3, but **do not run the code** on your computer. Does the code appear to create the plot reference plot given in the assignment?

0 points: The code does not create the reference plot, or is not viewable/present in repository

1 point: The code is mostly correct, but does not reproduce the reference plot exactly

2 points: The code reproduces the reference plot exactly

Score from your peers: 2

Please view the image file for Plot 4 from the GitHub repository. Does the plot appear correct?

0 points: No, the plot appears incorrect in at least 1 major discrepancy from the reference plot (e.g. wrong data), or at least 2 minor discrepancies (e.g. x-label is incorrect, title is incorrect), or the plot was not viewable

1 point: The plot is mostly correct with at most one minor discrepancy from the reference plot

Score from your peers: 1

Please evaluate the code for Plot 4, but **do not run the code** on your computer. Does the code appear to create the plot reference plot given in the assignment?

0 points: The code does not create the reference plot, or is not viewable/present in repository

1 point: The code is mostly correct, but does not reproduce the reference plot exactly

2 points: The code reproduces the reference plot exactly

Score from your peers: 2

[Peer Assessments \(https://class.coursera.org/exdata-002/human_grading/\)](https://class.coursera.org/exdata-002/human_grading/) / Course Project 2

[Help \(https://class.coursera.org/exdata-002/help/peergrading?url=https%3A%2F%2Fclass.coursera.org%2Fexdata-002%2Fhuman_grading%2Fview%2Fcourses%2F972082%2Fassessments%2F4%2Fresults%2Fmine\)](https://class.coursera.org/exdata-002/help/peergrading?url=https%3A%2F%2Fclass.coursera.org%2Fexdata-002%2Fhuman_grading%2Fview%2Fcourses%2F972082%2Fassessments%2F4%2Fresults%2Fmine)

Submission Phase

1. Do assignment (/exdata-002/human_grading/view/courses/972082/assessments/4/submissions)

Evaluation Phase

2. Evaluate peers (/exdata-002/human_grading/view/courses/972082/assessments/4/peerGradingSets)

Results Phase

3. See results (/exdata-002/human_grading/view/courses/972082/assessments/4/results/mine)

Your effective grade is **14.5**

Your unadjusted grade is 14.5, which is simply the grade you received from your peers.

See below for details.

Introduction

Fine particulate matter (PM_{2.5}) is an ambient air pollutant for which there is strong evidence that it is harmful to human health. In the United States, the Environmental Protection Agency (EPA) is tasked with setting national ambient air quality standards for fine PM and for tracking the emissions of this pollutant into the atmosphere. Approximately every 3 years, the EPA releases its database on emissions of PM_{2.5}. This database is known as the National Emissions Inventory (NEI). You can read more information about the NEI at the [EPA National Emissions Inventory web site \(http://www.epa.gov/ttn/chief/eiinformation.html\)](http://www.epa.gov/ttn/chief/eiinformation.html).

For each year and for each type of PM source, the NEI records how many tons of PM_{2.5} were emitted from that source over the course of the entire year. The data that you will use for this assignment are for 1999, 2002, 2005, and 2008.

Data

The data for this assignment are available from the course web site as a single zip file:

- [Data for Peer Assessment \(https://d396qusza40orc.cloudfront.net/exdata%2Fdata%2FNEI_data.zip\)](https://d396qusza40orc.cloudfront.net/exdata%2Fdata%2FNEI_data.zip) [29Mb]

The zip file contains two files:

PM_{2.5} Emissions Data (`summarySCC_PM25.rds`): This file contains a data frame with all of the PM_{2.5} emissions data for 1999, 2002, 2005, and 2008. For each year, the table contains number of **tons** of PM_{2.5} emitted from a specific type of source for the entire year. Here are the first few rows.

```
##      fips      SCC Pollutant Emissions   type year
## 4 09001 10100401  PM25-PRI    15.714 POINT 1999
## 8 09001 10100404  PM25-PRI   234.178 POINT 1999
## 12 09001 10100501  PM25-PRI     0.128 POINT 1999
## 16 09001 10200401  PM25-PRI    2.036 POINT 1999
## 20 09001 10200504  PM25-PRI    0.388 POINT 1999
## 24 09001 10200602  PM25-PRI    1.490 POINT 1999
```

- `fips` : A five-digit number (represented as a string) indicating the U.S. county
- `scc` : The name of the source as indicated by a digit string (see source code classification table)
- `Pollutant` : A string indicating the pollutant
- `Emissions` : Amount of PM_{2.5} emitted, in tons
- `type` : The type of source (point, non-point, on-road, or non-road)
- `year` : The year of emissions recorded

Source Classification Code Table (`Source_Classification_Code.rds`): This table provides a mapping from the SCC digit strings in the Emissions table to the actual name of the PM_{2.5} source. The sources are categorized in a few different ways from more general to more specific and you may choose to explore whatever categories you think are most useful. For example, source “10100101” is known as “Ext Comb /Electric Gen /Anthracite Coal /Pulverized Coal”.

You can read each of the two files using the `readRDS()` function in R. For example, reading in each file can be done with the following code:

```
## This first line will likely take a few seconds. Be patient!
NEI <- readRDS("summarySCC_PM25.rds")
SCC <- readRDS("Source_Classification_Code.rds")
```

as long as each of those files is in your current working directory (check by calling `dir()` and see if those files are in the listing).

Assignment

The overall goal of this assignment is to explore the National Emissions Inventory database and see what it says about fine particulate matter pollution in the United States over the 10-year period 1999–2008. You may use any R package you want to support your analysis.

Questions

You must address the following questions and tasks in your exploratory analysis. For each question/task you will need to make a single plot. Unless specified, you can use any plotting system in R to make your plot.

1. Have total emissions from PM_{2.5} decreased in the United States from 1999 to 2008? Using the **base** plotting system, make a plot showing the *total* PM_{2.5} emission from all sources for each of the years 1999, 2002, 2005, and 2008.

- Have total emissions from PM_{2.5} decreased in the **Baltimore City**, Maryland (`fips == "24510"`) from 1999 to 2008? Use the **base** plotting system to make a plot answering this question.
- Of the four types of sources indicated by the `type` (point, nonpoint, onroad, nonroad) variable, which of these four sources have seen decreases in emissions from 1999–2008 for **Baltimore City**? Which have seen increases in emissions from 1999–2008? Use the **ggplot2** plotting system to make a plot answer this question.
- Across the United States, how have emissions from coal combustion-related sources changed from 1999–2008?
- How have emissions from motor vehicle sources changed from 1999–2008 in **Baltimore City**?
- Compare emissions from motor vehicle sources in Baltimore City with emissions from motor vehicle sources in **Los Angeles County**, California (`fips == "06037"`). Which city has seen greater changes over time in motor vehicle emissions?

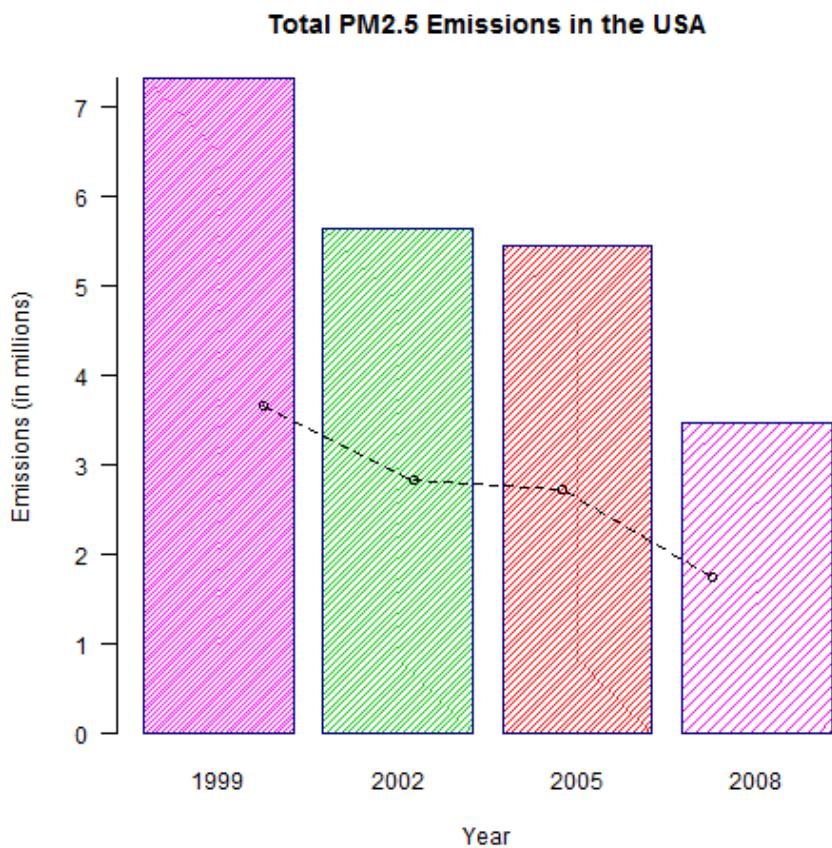
Making and Submitting Plots

For each plot you should

- Construct the plot and save it to a **PNG file**.
- Create a separate R code file (`plot1.R`, `plot2.R`, etc.) that constructs the corresponding plot, i.e. code in `plot1.R` constructs the `plot1.png` plot. Your code file should include code for reading the data so that the plot can be fully reproduced. You should also include the code that creates the PNG file. Only include the code for a single plot (i.e. `plot1.R` should only include code for producing `plot1.png`)
- Upload the PNG file on the Assignment submission page
- Copy and paste the R code from the corresponding R file into the text box at the appropriate point in the peer assessment.

Have total emissions from PM_{2.5} decreased in the United States from 1999 to 2008? Using the **base** plotting system, make a plot showing the *total* PM_{2.5} emission from all sources for each of the years 1999, 2002, 2005, and 2008.

Upload a PNG file containing your plot addressing this question.



Yes, PM2.5 has decreased in the United States from 1999 to 2008

Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

Please view the plot for this question. Does the plot appear to address the question being asked? In other words, can you answer the question using the information shown in the plot?

Score from your peers: **2**

Upload the R code file for the plot uploaded in the previous question.

```
# read the RDS files
NEI <- readRDS("summarySCC_PM25.rds")
SCC <- readRDS("Source_Classification_Code.rds")

# yearwise total emission
summaryEmissions <- tapply(NEI$Emissions, NEI$year, sum)
# plot
png(filename = "plot1.png", width = 480, height = 480)
barplot(summaryEmissions, col = 2*summaryEmissions/10^6, border = "dark blue", density =
```

```
5*summaryEmissions/10^6, yaxt = "n") #col = heat.colors(6), log = "y"  
axis(2, at = seq(0, 8000000, 1000000), labels = 0:8, las = 2)  
lines(summaryEmissions / 2, lty=2)  
points(summaryEmissions / 2)  
title("Total PM2.5 Emissions in the USA", xlab="Year", ylab="Emissions (in millions)")  
dev.off()
```

Evaluation/feedback on the above work

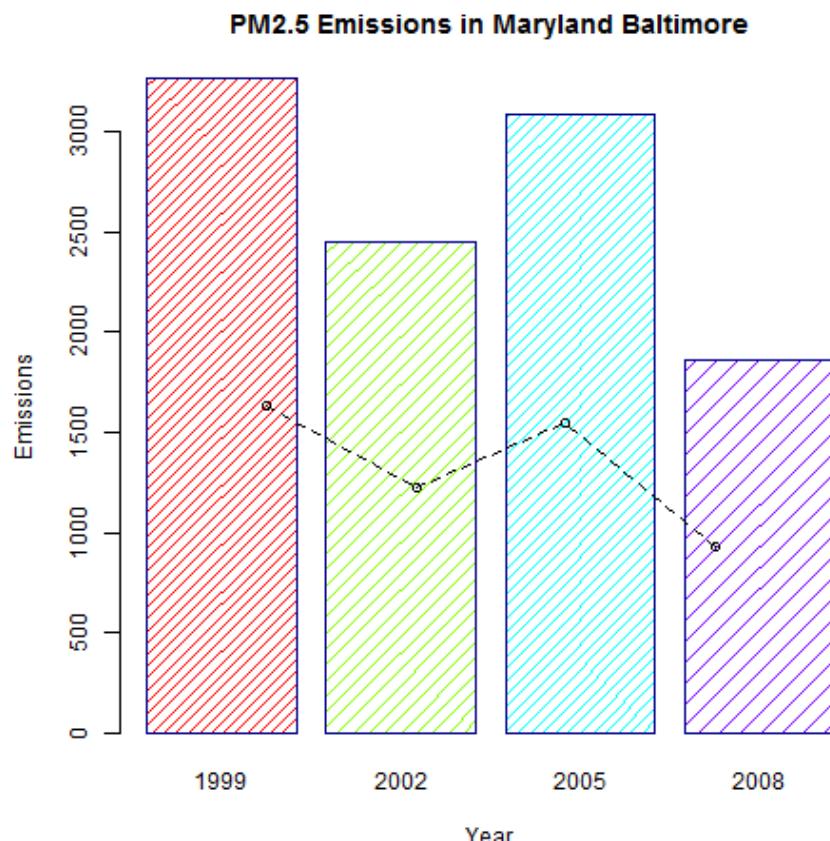
Note: this section can only be filled out during the evaluation phase.

Examine the submitted R code file. Does the R code appear to construct the plot shown in the previous question? NOTE: Do not run the code on your own computer.

Score from your peers: 1

Have total emissions from PM_{2.5} decreased in the **Baltimore City, Maryland** (`fips == 24510`) from 1999 to 2008? Use the **base** plotting system to make a plot answering this question.

Upload a PNG file containing your plot addressing this question.



No, PM2.5 has not decreased steadily in Baltimore from 1999 to 2008

Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

Please view the plot for this question. Does the plot appear to address the question being asked? In other words, can you answer the question using the information shown in the plot?

Score from your peers: **2**

Upload the R code file for the plot uploaded in the previous question.

```
# read the RDS files
NEI <- readRDS("summarySCC_PM25.rds")
SCC <- readRDS("Source_Classification_Code.rds")

# yearwise emission at Baltimore
NEIBaltimore <- subset(NEI, fips == "24510")
summaryEmissions <- tapply(NEIBaltimore$Emissions, NEIBaltimore$year, sum)
# plot
png(filename = "plot2.png", width = 480, height = 480)
barplot(summaryEmissions, col = rainbow(4), border = "dark blue", density =
5*summaryEmissions/10^3)
lines(summaryEmissions / 2, lty=2)
points(summaryEmissions / 2)
title("PM2.5 Emissions in Maryland Baltimore", xlab="Year", ylab="Emissions")
dev.off()
```

Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

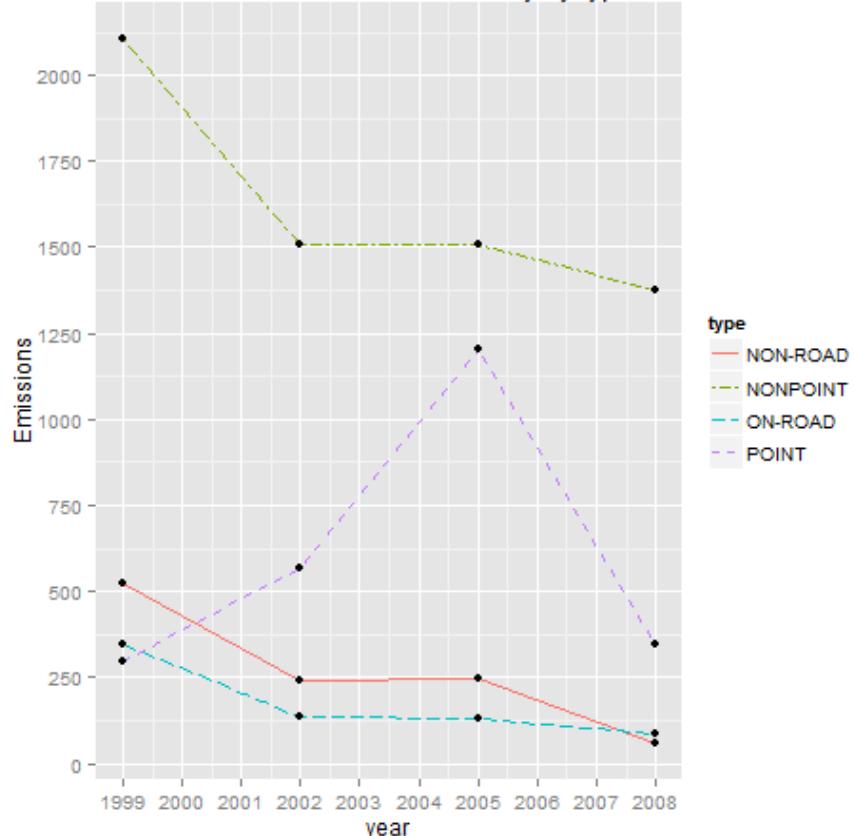
Examine the submitted R code file. Does the R code appear to construct the plot shown in the previous question? NOTE: Do not run the code on your own computer.

Score from your peers: **1**

Of the four types of sources indicated by the `type` (point, nonpoint, onroad, nonroad) variable, which of these four sources have seen decreases in emissions from 1999–2008 for **Baltimore City**? Which have seen increases in emissions from 1999–2008? Use the **ggplot2** plotting system to make a plot answer this question.

Upload a PNG file containing your plot addressing this question.

Emissions from 1999–2008 for Baltimore City by types of sources



As can be seen, NON-ROAD, NON-POINT and ON-ROAD types have shown decrease in Emission, while
POINT type has increased from 1999-2005 and then decreased.

Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

Please view the plot for this question. Does the plot appear to address the question being asked? In other words, can you answer the question using the information shown in the plot?

Score from your peers: 2

Upload the R code file for the plot uploaded in the previous question.

```
library(ggplot2)

# read the RDS files
NEI <- readRDS("summarySCC_PM25.rds")
SCC <- readRDS("Source_Classification_Code.rds")

NEIB <- subset(NEI, fips == "24510")
NEIBty <- aggregate(NEIB$Emissions, list(NEIB$year, NEIB$type), sum)
names(NEIBty) <- c("year", "type", "Emissions")
png(filename = "plot3.png", width = 480, height = 480)
ggplot(NEIBty, aes(x=year, y=Emissions, group=type)) +
  geom_line(aes(colour=type, linetype=type)) +
  geom_point() +
  scale_x_continuous(breaks=seq(1999, 2008, 1)) +
  scale_y_continuous(breaks=seq(0, 2500, 250)) +
  scale_linetype_manual(values=c("solid", "twodash", "longdash", "dashed")) +
  ggtitle("Emissions from 1999–2008 for Baltimore City by types of sources")
dev.off()
```

Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

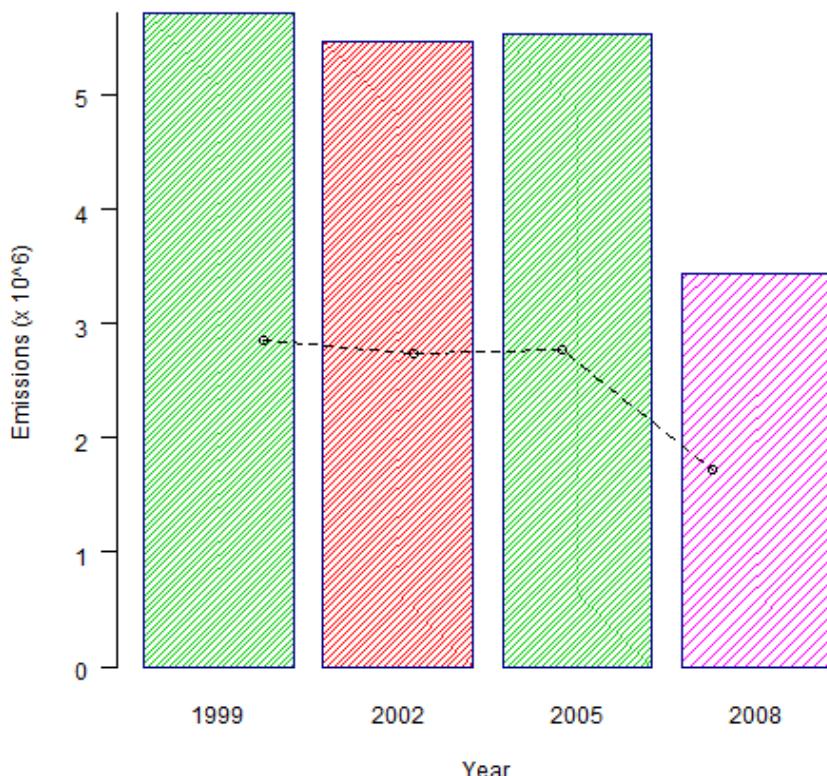
Examine the submitted R code file. Does the R code appear to construct the plot shown in the previous question? NOTE: Do not run the code on your own computer.

Score from your peers: 1

Across the United States, how have emissions from coal combustion-related sources changed from 1999–2008?

Upload a PNG file containing your plot addressing this question.

Emissions from coal combustion-related sources in the USA



As can be seen there is an overall decrease in emission from 1999-2008 (although there is a slight increase in between)

Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

Please view the plot for this question. Does the plot appear to address the question being asked? In other words, can you answer the question using the information shown in the plot?

Score from your peers: 1

Examine the submitted R code file. Does the R code appear to construct the plot shown in the previous question? NOTE: Do not run the code on your own computer.

```
# read the RDS files  
NEI <- readRDS("summarySCC_PM25.rds")  
SCC <- readRDS("Source_Classification_Code.rds")  
  
#unique(SCC$EI.Sector)  
SCC <- SCC[SCC$EI.Sector %in% grep("Coal", unique(SCC$EI.Sector), value=TRUE),]
```

```
NEISCC <- merge(NEI, SCC, by.x="SCC", by.y="SCC")

summaryEmissions <- tapply(NEISCC$Emissions, NEISCC$year, sum)
# plot
png(filename = "plot4.png", width = 480, height = 480)
barplot(summaryEmissions, col = 2*summaryEmissions/10^5, border = "dark blue", density =
5*summaryEmissions/10^5, yaxt = "n")
axis(2, at = seq(0, 700000, 100000), labels = 0:7, las = 2)
lines(summaryEmissions / 2, lty=2)
points(summaryEmissions / 2)
title("Emissions from coal combustion-related sources in the USA", xlab="Year", ylab="Emissions (x
10^6)")
dev.off()
```

Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

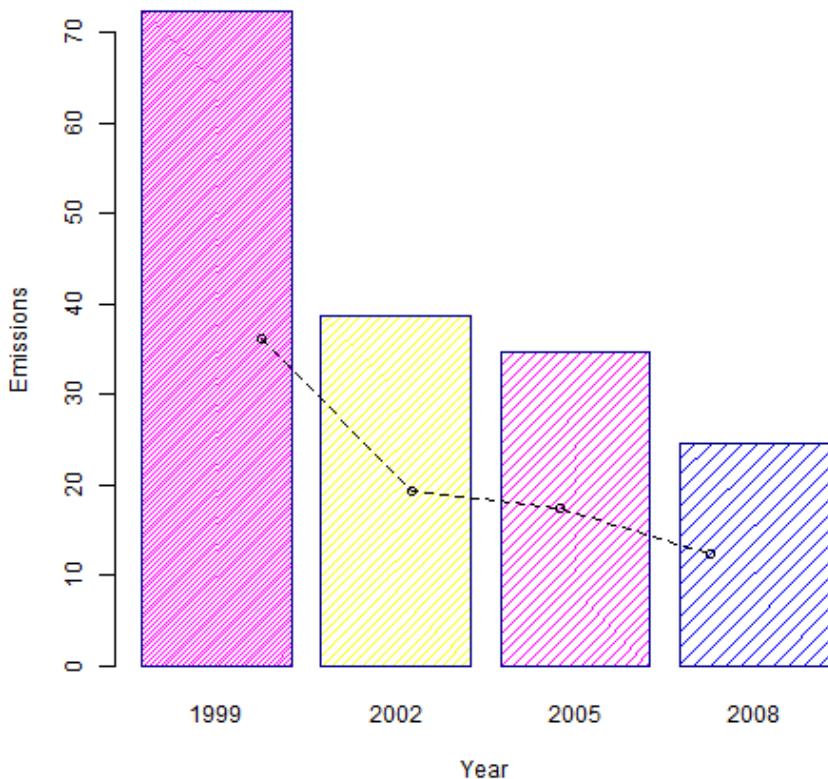
Examine the submitted R code file. Does the R code appear to construct the plot shown in the previous question? NOTE: Do not run the code on your own computer.

Score from your peers: **1**

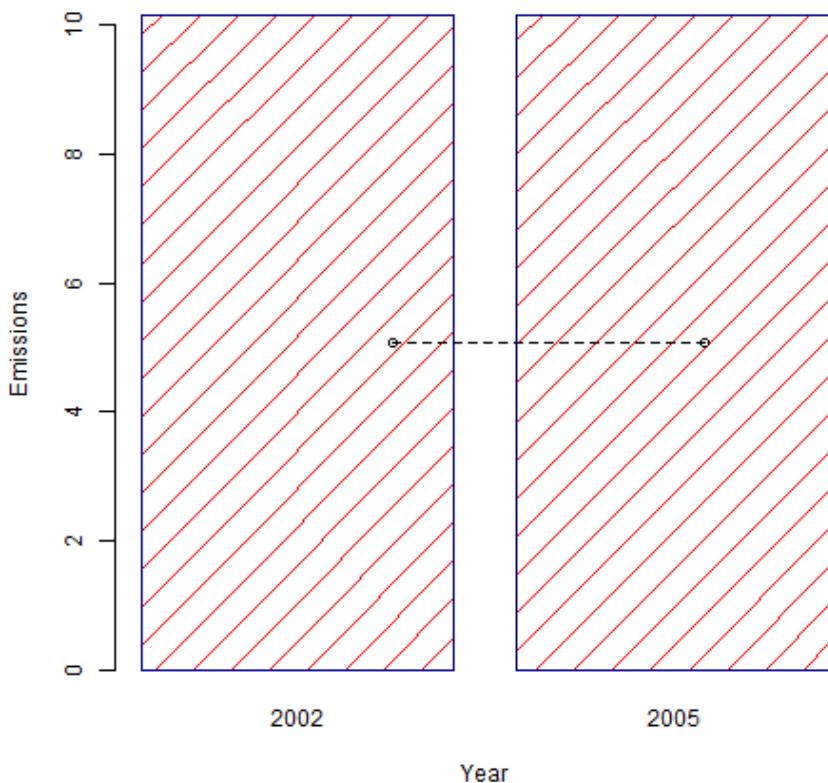
How have emissions from motor vehicle sources changed from 1999–2008 in **Baltimore City**?

Upload a PNG file containing your plot addressing this question.

Emissions from Vehicle sources at Baltimore



Emissions from Motor Vehicle sources at Baltimore



As can be seen, Emissions from Motor Vehicle sources was same in 2002 and 2005, while emission from all vehicle sources have undergone a steady decrease at Baltimore from 1999 to 2008.

Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

Please view the plot for this question. Does the plot appear to address the question being asked? In other words, can you answer the question using the information shown in the plot?

Score from your peers: 1

Upload the R code file for the plot uploaded in the previous question.

```
# read the RDS files
NEI <- readRDS("summarySCC_PM25.rds")
SCC <- readRDS("Source_Classification_Code.rds")

NEIBaltimore <- subset(NEI, fips == "24510")
NEIBSCC <- merge(NEIBaltimore, SCC, by.x="SCC", by.y="SCC")
#sort(unique(NEIBSCC$Short.Name))
NEIBSCC <- NEIBSCC[NEIBSCC$Short.Name %in% grep("Vehicle", unique(NEIBSCC$Short.Name),
value=TRUE),]

summaryEmissions <- tapply(NEIBSCC$Emissions, NEIBSCC$year, sum)
# plot
png(filename = "plot5.png", width = 480, height = 480)
barplot(summaryEmissions, col = 2*summaryEmissions/10, border = "dark blue", density =
5*summaryEmissions/10)
lines(summaryEmissions / 2, lty=2)
points(summaryEmissions / 2)
title("Emissions from Vehicle sources at Baltimore", xlab="Year", ylab="Emissions")
dev.off()

NEIBSCC <- merge(NEIBaltimore, SCC, by.x="SCC", by.y="SCC")
#sort(unique(NEIBSCC$Short.Name))
NEIBSCC <- NEIBSCC[NEIBSCC$Short.Name %in% grep("Motor Vehicle",
unique(NEIBSCC$Short.Name), value=TRUE),]

summaryEmissions <- tapply(NEIBSCC$Emissions, NEIBSCC$year, sum)
# plot
png(filename = "plot5.1.png", width = 480, height = 480)
barplot(summaryEmissions, col = 2*summaryEmissions/10, border = "dark blue", density =
5*summaryEmissions/10)
lines(summaryEmissions / 2, lty=2)
points(summaryEmissions / 2)
title("Emissions from Motor Vehicle sources at Baltimore", xlab="Year", ylab="Emissions")
dev.off()
```

Evaluation/feedback on the above work

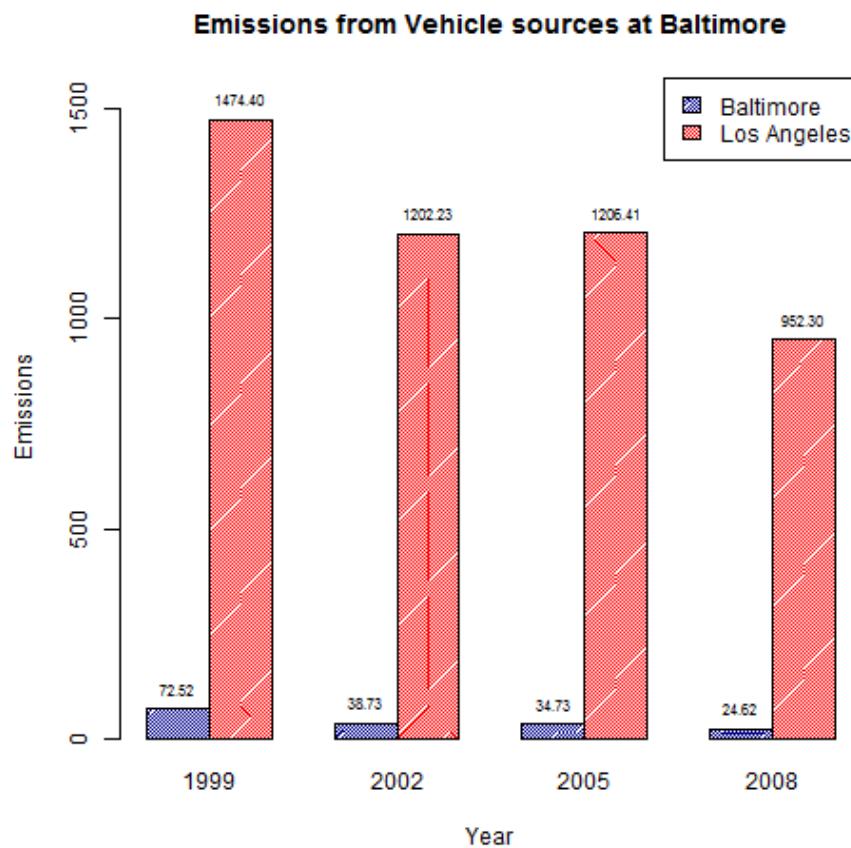
Note: this section can only be filled out during the evaluation phase.

Examine the submitted R code file. Does the R code appear to construct the plot shown in the previous question? NOTE: Do not run the code on your own computer.

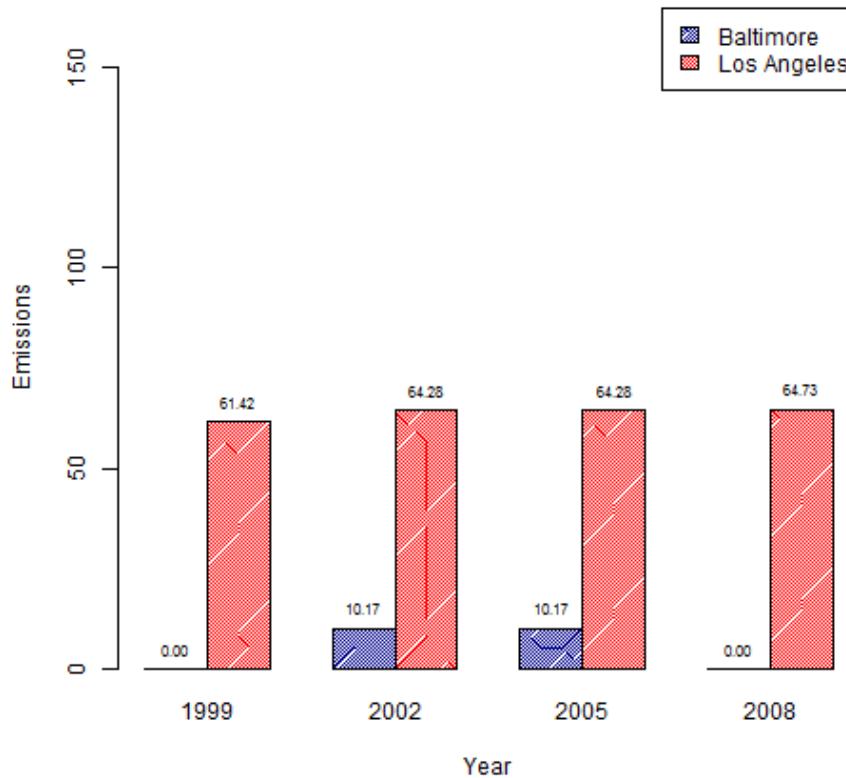
Score from your peers: 1

Compare emissions from motor vehicle sources in Baltimore City with emissions from motor vehicle sources in **Los Angeles County**, California (`fips == 06037`). Which city has seen greater changes over time in motor vehicle emissions?

Upload a PNG file containing your plot addressing this question.



Emissions from Motor Vehicle sources at Baltimore



As can be seen, Los Angeles has greater changes in Emission over time.

Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

Please view the plot for this question. Does the plot appear to address the question being asked? In other words, can you answer the question using the information shown in the plot?

Score from your peers: **0.5**

Upload the R code file for the plot uploaded in the previous question.

```
# read the RDS files
NEI <- readRDS("summarySCC_PM25.rds")
SCC <- readRDS("Source_Classification_Code.rds")

NEIBaltimore <- subset(NEI, fips == "24510")
NEILosAngeles <- subset(NEI, fips == "06037")
NEIBSCC <- merge(NEIBaltimore, SCC, by.x="SCC", by.y="SCC")
NEIBSCC <- NEIBSCC[NEIBSCC$Short.Name %in% grep("Vehicle", unique(NEIBSCC$Short.Name),
value=TRUE),]
```

```

NEILSCC <- merge(NEILosAngeles, SCC, by.x="SCC", by.y="SCC")
NEILSCC <- NEILSCC[NEILSCC$Short.Name %in% grep("Vehicle", unique(NEILSCC$Short.Name),
value=TRUE),]

summaryBEmissions <- tapply(NEIBSCC$Emissions, NEIBSCC$year, sum)
summaryLEmissions <- tapply(NEILSCC$Emissions, NEILSCC$year, sum)
# plot
png(filename = "plot6.png", width = 480, height = 480)
height <- rbind(summaryBEmissions, summaryLEmissions)
mp <- barplot(height, ylim = c(0, max(height)+100), names.arg = names(height),
col=c("darkblue","red"), density = c(50, 50), beside = TRUE)
text(mp, height, labels = format(round(height,2), 4), pos = 3, cex = .7)
legend("topright", legend = c("Baltimore", "Los Angeles"), fill = c("darkblue", "red"), density = c(50,
50))
title("Emissions from Vehicle sources at Baltimore", xlab="Year", ylab="Emissions")
dev.off()

NEIBaltimore <- subset(NEI, fips == "24510")
NEILosAngeles <- subset(NEI, fips == "06037")
NEIBSCC <- merge(NEIBaltimore, SCC, by.x="SCC", by.y="SCC")
NEIBSCC <- NEIBSCC[NEIBSCC$Short.Name %in% grep("Motor Vehicle",
unique(NEIBSCC$Short.Name), value=TRUE),]
NEILSCC <- merge(NEILosAngeles, SCC, by.x="SCC", by.y="SCC")
NEILSCC <- NEILSCC[NEILSCC$Short.Name %in% grep("Motor Vehicle",
unique(NEILSCC$Short.Name), value=TRUE),]

summaryBEmissions <- NULL
summaryBEmissions[c('1999', '2002', '2005', '2008')] <- 0
tmp <- tapply(NEIBSCC$Emissions, NEIBSCC$year, sum)
summaryBEmissions[names(tmp)] <- tmp
summaryBEmissions["2008"] <- 0
summaryLEmissions <- tapply(NEILSCC$Emissions, NEILSCC$year, sum)
# plot
png(filename = "plot6.1.png", width = 480, height = 480)
height <- rbind(summaryBEmissions, summaryLEmissions)
mp <- barplot(height, ylim = c(0, max(height)+100), names.arg = names(height),
col=c("darkblue","red"), density = c(50, 50), beside = TRUE)
text(mp, height, labels = format(round(height,2), 4), pos = 3, cex = .7)
legend("topright", legend = c("Baltimore", "Los Angeles"), fill = c("darkblue", "red"), density = c(50,
50))
title("Emissions from Motor Vehicle sources at Baltimore", xlab="Year", ylab="Emissions")
dev.off()

```

Evaluation/feedback on the above work

Note: this section can only be filled out during the evaluation phase.

Examine the submitted R code file. Does the R code appear to construct the plot shown in the previous question? NOTE: Do not run the code on your own computer.

Score from your peers: **1**

Confidence intervals, t tests, P values

Joe Felsenstein

Department of Genome Sciences and Department of Biology

Normality

Everybody believes in the normal approximation, the experimenters because they think it is a mathematical theorem, the mathematicians because they think it is an experimental fact!

G. Lippman

- We can use the Gaussian (normal) distribution, assumed correct, and estimate the mean (which is the expectation). It turns out that, not surprisingly, the best estimate of the mean is the mean of the sample.

Normality

Everybody believes in the normal approximation, the experimenters because they think it is a mathematical theorem, the mathematicians because they think it is an experimental fact!

G. Lippman

- We can use the Gaussian (normal) distribution, assumed correct, and estimate the mean (which is the expectation). It turns out that, not surprisingly, the best estimate of the mean is the mean of the sample.
- (The median of the sample is a legitimate estimate too, but it is noisier).

Normality

Everybody believes in the normal approximation, the experimenters because they think it is a mathematical theorem, the mathematicians because they think it is an experimental fact!

G. Lippman

- We can use the Gaussian (normal) distribution, assumed correct, and estimate the mean (which is the expectation). It turns out that, not surprisingly, the best estimate of the mean is the mean of the sample.
- (The median of the sample is a legitimate estimate too, but it is noisier).
- The sample mean is the optimal estimate as it is the Maximum Likelihood Estimate – for which see later in the course.

Normality

Everybody believes in the normal approximation, the experimenters because they think it is a mathematical theorem, the mathematicians because they think it is an experimental fact!

G. Lippman

- We can use the Gaussian (normal) distribution, assumed correct, and estimate the mean (which is the expectation). It turns out that, not surprisingly, the best estimate of the mean is the mean of the sample.
- (The median of the sample is a legitimate estimate too, but it is noisier).
- The sample mean is the optimal estimate as it is the Maximum Likelihood Estimate – for which see later in the course.
- But how do we figure out how noisy the estimate is?

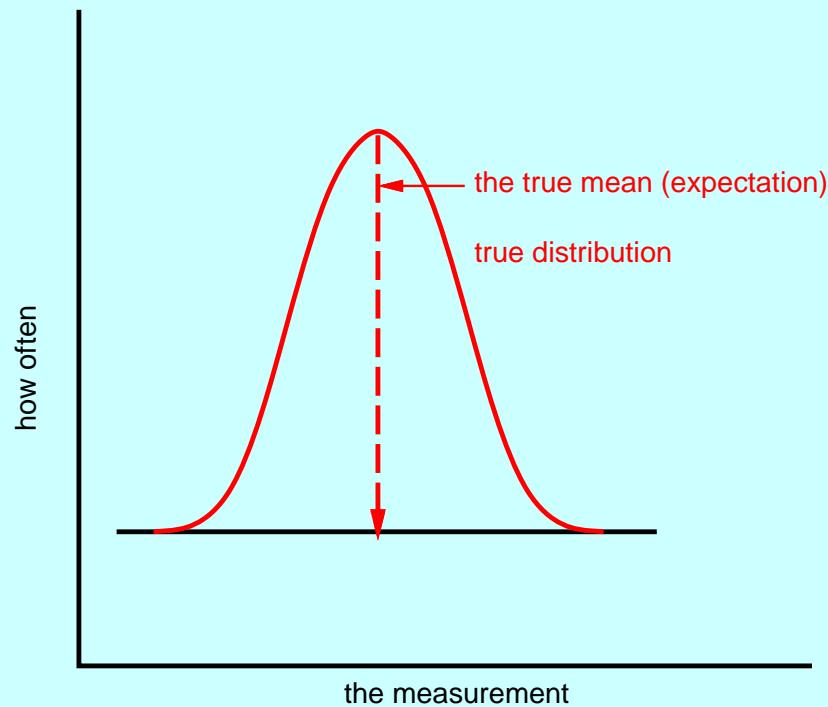
Normality

Everybody believes in the normal approximation, the experimenters because they think it is a mathematical theorem, the mathematicians because they think it is an experimental fact!

G. Lippman

- We can use the Gaussian (normal) distribution, assumed correct, and estimate the mean (which is the expectation). It turns out that, not surprisingly, the best estimate of the mean is the mean of the sample.
- (The median of the sample is a legitimate estimate too, but it is noisier).
- The sample mean is the optimal estimate as it is the Maximum Likelihood Estimate – for which see later in the course.
- But how do we figure out how noisy the estimate is?
- Can we make an *interval estimate*?

A normal distribution (artist's conception)



Uncertainty of the mean

- Let's go forward (distribution to data).

Uncertainty of the mean

- Let's go forward (distribution to data).
- if the (unknown) standard deviation of the true distribution is σ , the variance is σ^2 .

Uncertainty of the mean

- Let's go forward (distribution to data).
- if the (unknown) standard deviation of the true distribution is σ , the variance is σ^2 .
- The variance of the mean of a sample of n points is σ^2/n ,

Uncertainty of the mean

- Let's go forward (distribution to data).
- if the (unknown) standard deviation of the true distribution is σ , the variance is σ^2 .
- The variance of the mean of a sample of n points is σ^2/n ,
- so its standard deviation is σ/\sqrt{n} .

Uncertainty of the mean

- Let's go forward (distribution to data).
- if the (unknown) standard deviation of the true distribution is σ , the variance is σ^2 .
- The variance of the mean of a sample of n points is σ^2/n ,
- so its standard deviation is σ/\sqrt{n} .
- The 2.5% point of a normal distribution is 1.95996 standard deviations below the mean.

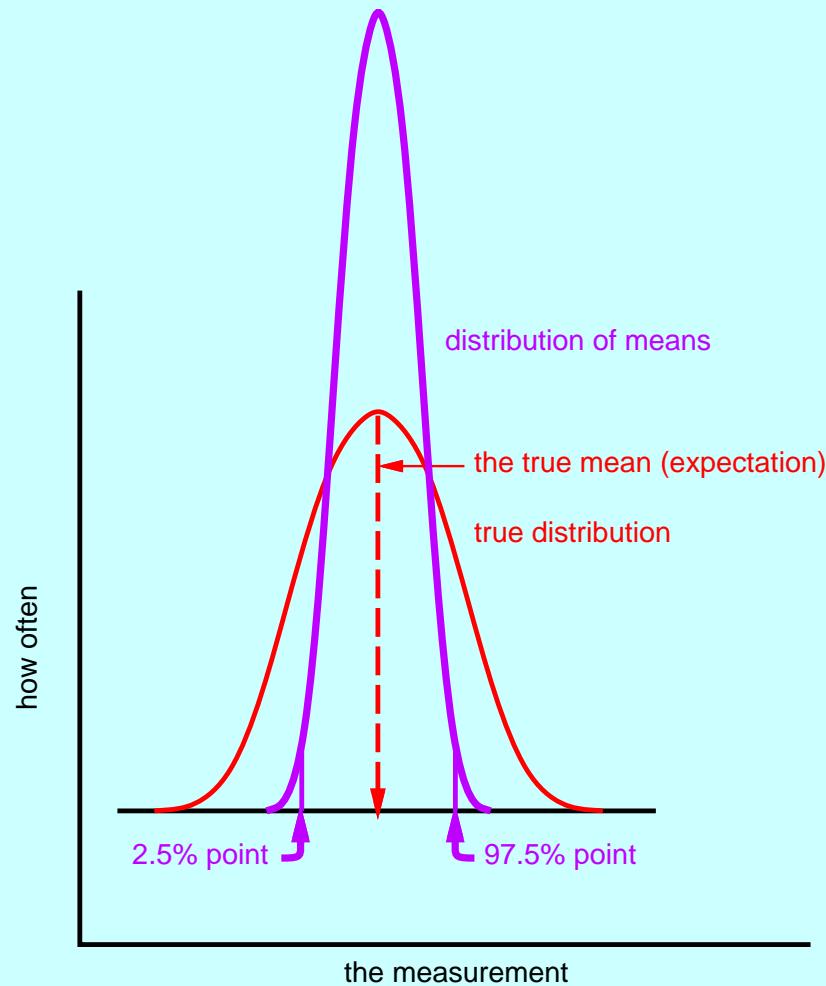
Uncertainty of the mean

- Let's go forward (distribution to data).
- if the (unknown) standard deviation of the true distribution is σ , the variance is σ^2 .
- The variance of the mean of a sample of n points is σ^2/n ,
- so its standard deviation is σ/\sqrt{n} .
- The 2.5% point of a normal distribution is 1.95996 standard deviations below the mean.
- The 97.5% point of a normal distribution is 1.95996 standard deviations above the mean.

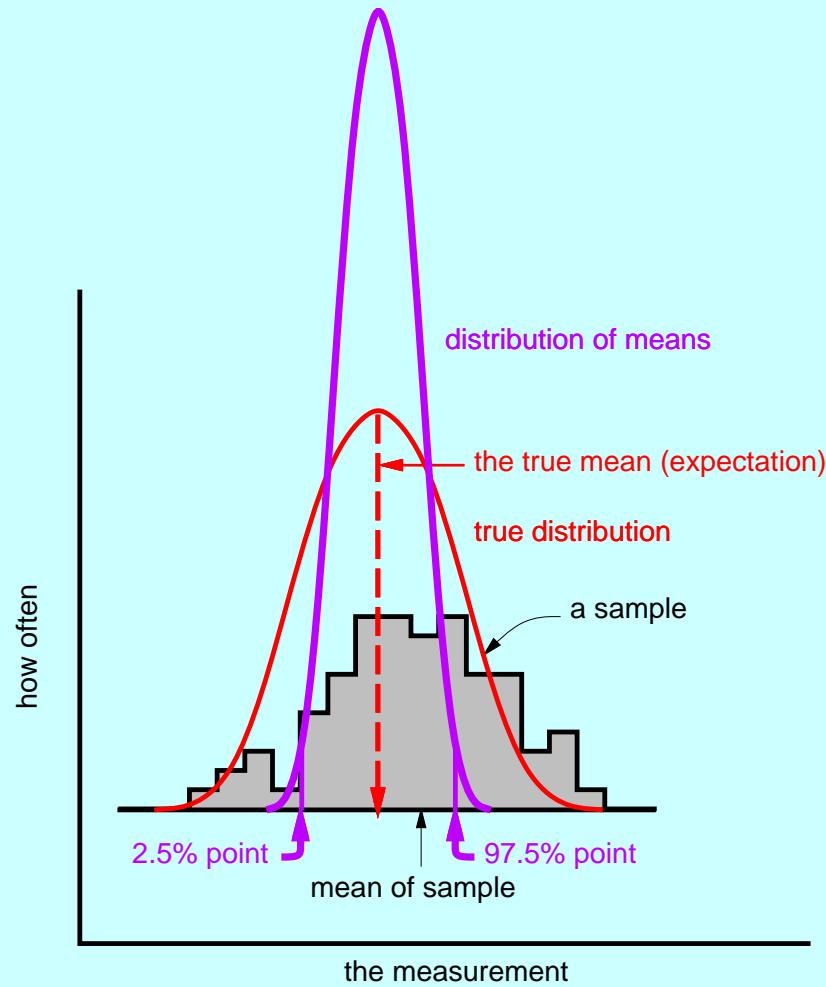
Uncertainty of the mean

- Let's go forward (distribution to data).
- if the (unknown) standard deviation of the true distribution is σ , the variance is σ^2 .
- The variance of the mean of a sample of n points is σ^2/n ,
- so its standard deviation is σ/\sqrt{n} .
- The 2.5% point of a normal distribution is 1.95996 standard deviations below the mean.
- The 97.5% point of a normal distribution is 1.95996 standard deviations above the mean.
- So if the (unknown) true mean is called μ , 95% of the time the mean you calculate from a sample, will lie between $\mu - 1.95996 \sigma/\sqrt{n}$ and $\mu + 1.95996 \sigma/\sqrt{n}$.

The distribution of means of n points



A particular sample



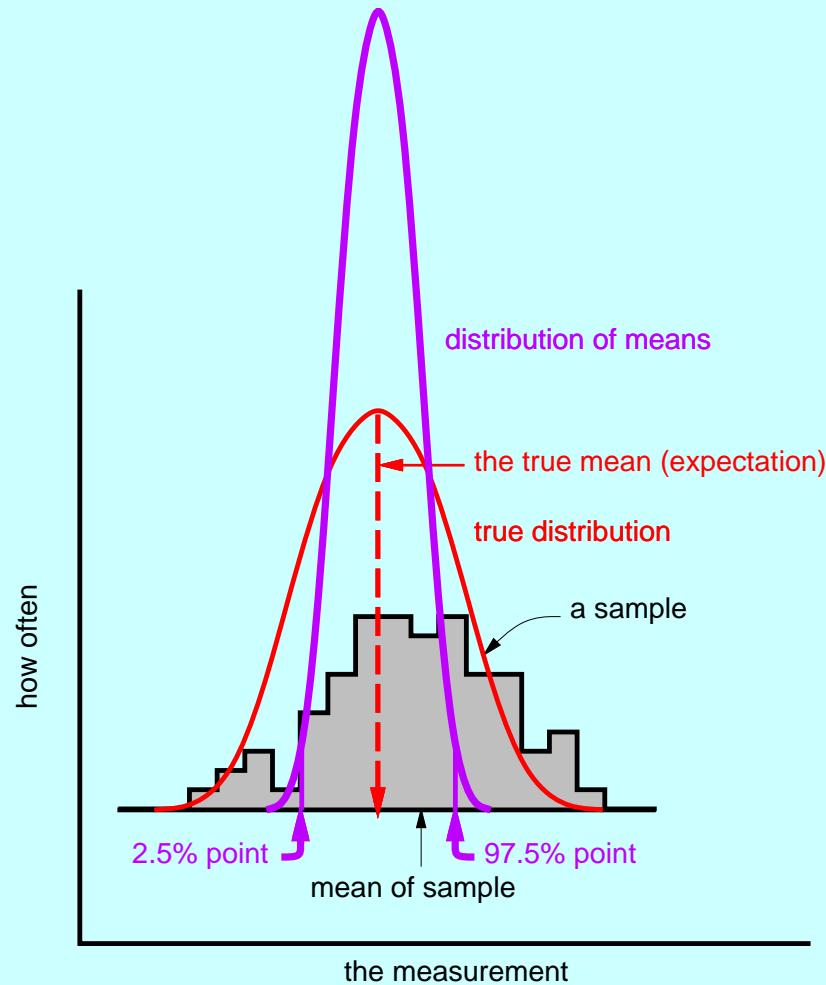
Confidence interval

So, that solves it, right? No! We don't know μ (which is what we want to know). We have calculated how the limits on the sample mean, not the limits on the true mean.

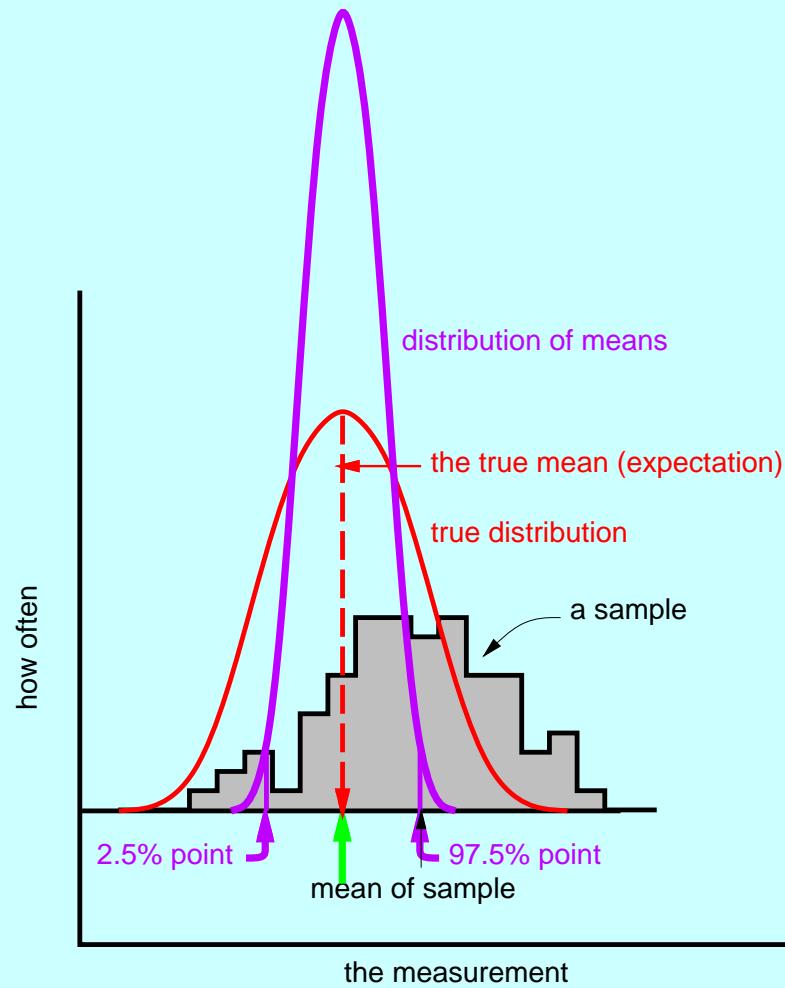
But we can say that, 95% of the time, the empirical mean \bar{x} that we calculate is below that upper limit, and above that lower limit.

In that sense (*in what sense?*) the true mean is ...

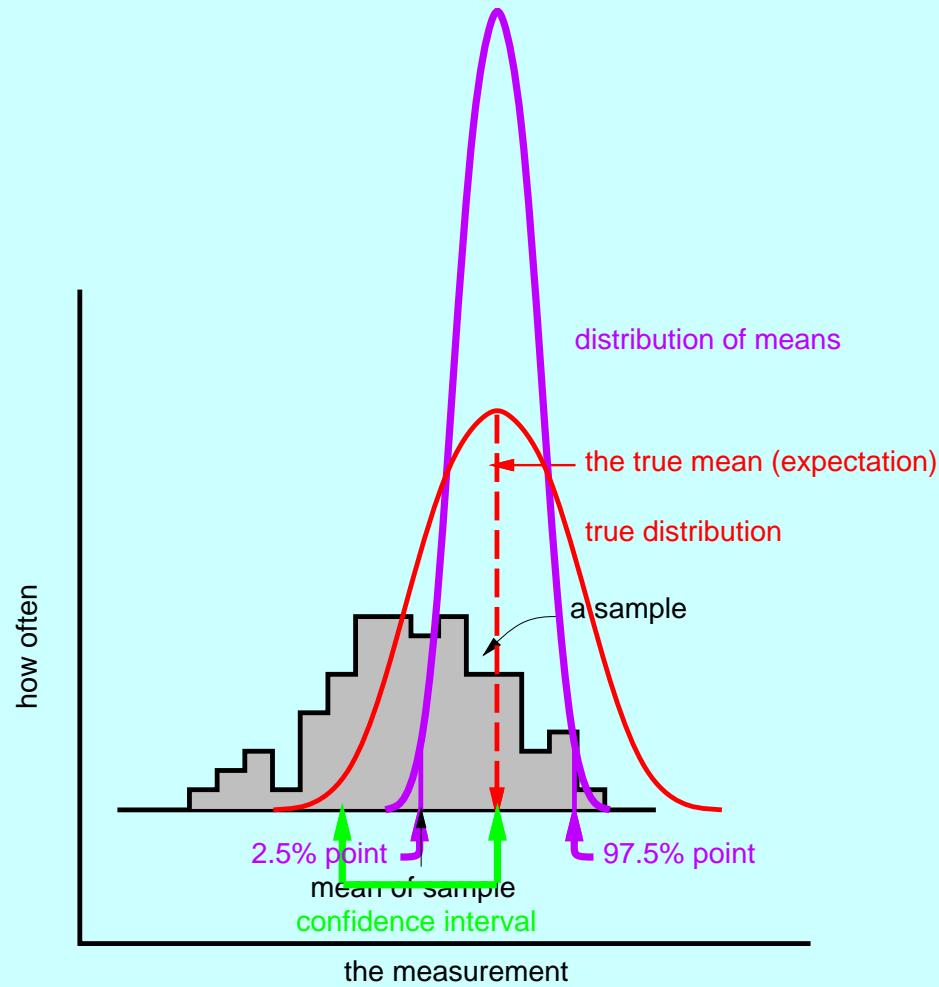
Let's get ready to slide the true stuff left



Not any lower than this ...

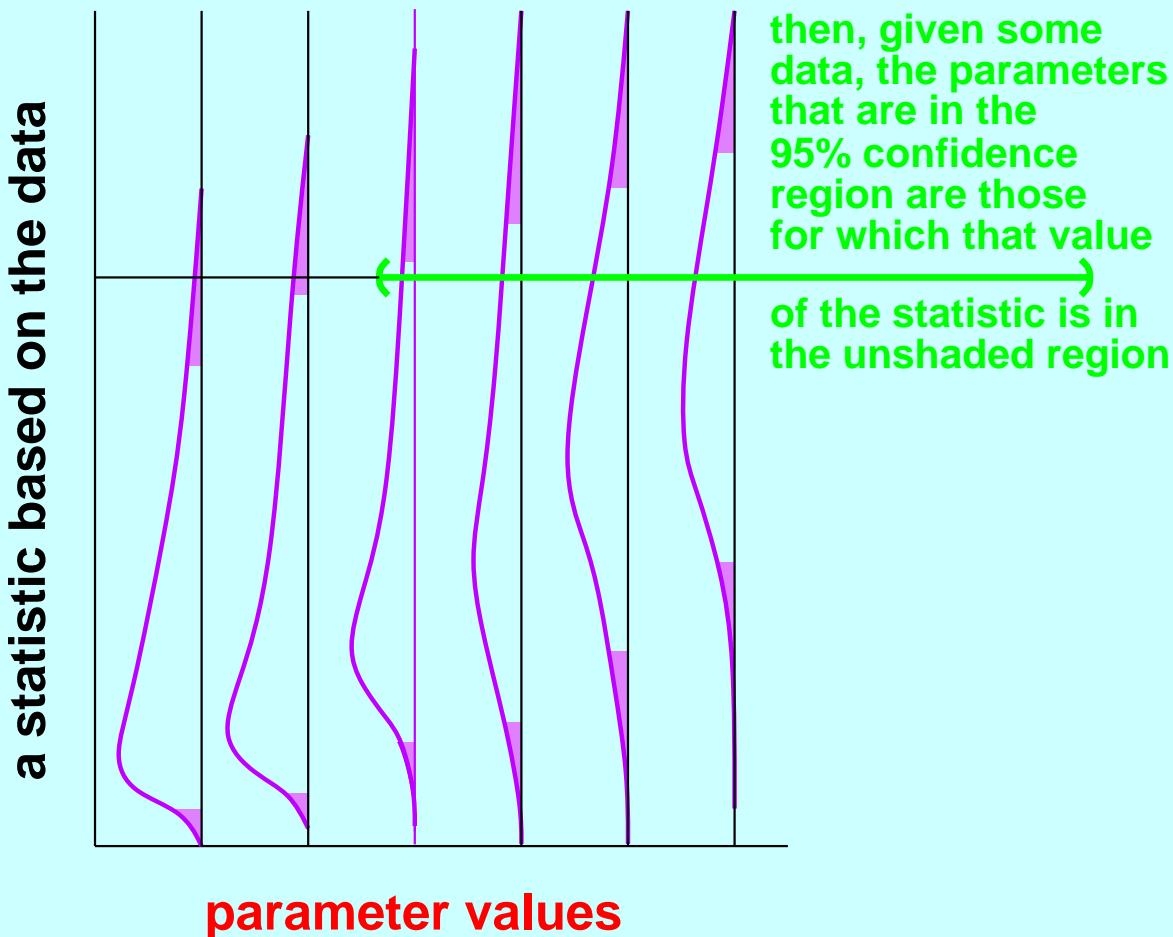


Not any higher than this ...



Here we see true values, and data statistics

for each parameter value, find data values (unshaded) that account for 95% of the probability



so 95% of the time the statistic is in the region where the confidence interval based on it contains the truth.

In what sense??

You could say that if you do this throughout your career, 95% of these intervals will contain the true value.

- This is fairly unsatisfactory, as it acts as if the outcome of some other, unrelated, experiment is relevant.

In what sense??

You could say that if you do this throughout your career, 95% of these intervals will contain the true value.

- This is fairly unsatisfactory, as it acts as if the outcome of some other, unrelated, experiment is relevant.
- Maybe best this way: if you did this experiment again and again and again, each time making a confidence interval for the mean, 95% of those intervals would contain the true value.

The Bayesian alternative

We will cover this later (lecture 6). It involves assuming that we know a prior probability of the parameters (in this case of the mean μ of the true distribution and the standard deviation σ).

Then, using Bayes' Formula (which we see in that lecture) we can compute the posterior probability of the parameter, given the data.

This gives us what we really wanted: the probabilities (or probability densities) of different values of the parameters. But it is achieved at the cost of having to have a prior distribution for them that we are assuming is true.

The present approach instead makes a different assumption, that we can regard the experiment as a repeatable trial. This Frequentist approach and the Bayesian approach both have supporters among statisticians – the issue is one's philosophy of science, not a technical disagreement about statistics.

Coping with not knowing the standard deviation

So far we have casually assumed we know the standard deviation σ of the true distribution. But generally we don't. The upper (lower) 2.5% point of the sample means is 1.95996σ . But if we look at the estimate of σ^2 , its unbiased estimate is (for reasons connected with also estimating the mean)

$$\hat{s}^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}$$

(the $n - 1$ instead of n is to correct for x_i being a bit closer to \bar{x} than it should be, because x_i is part of \bar{x} as well).

The standard deviation we use is the (estimated) standard deviation of the mean, which is \hat{s}/\sqrt{n} .

The quantity 1.95996 needs to be replaced by something that corrects for us sometimes having a \hat{s} that is smaller than σ , and sometimes larger.

Here's why, in this case "Guinness is good for you":



Student's t distribution



W. S. Gosset (1876-1937) was a modest, well-liked Englishman who was a brewer and agricultural statistician for the famous Guinness brewing company in Dublin. It insisted that its employees keep their work secret, so he published under the pseudonym 'Student' the distribution in 1908. This was one of the first results in modern small-sample statistics.

Why we have to make the confidence interval wider

We can't just say that the estimated $\hat{\sigma}$ is sometimes 20% smaller than the true value, and sometimes 20% larger, so it all cancels out.

It doesn't, because σ doesn't have a symmetrical distribution, but something derived from a chi-square distribution (which we have not introduced yet).

The t statistic

This is the number of (estimated) standard deviations of the mean that the mean deviates from its expected value.

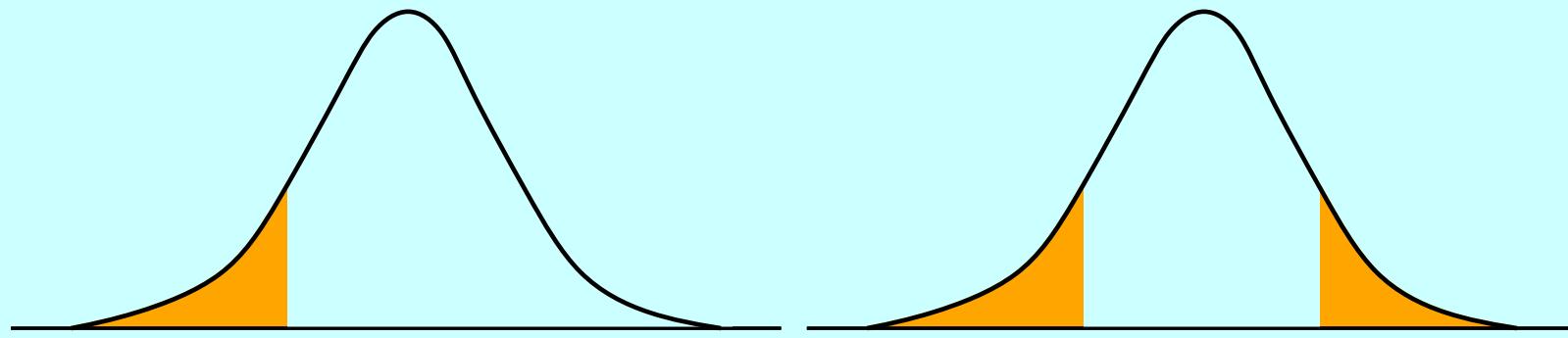
If \hat{s} is the estimated standard deviation, from a sample of n values, and the mean of the sample is \bar{x} , while the expectation of that mean is μ , then

$$t = \frac{\bar{x} - \mu}{\hat{s}/\sqrt{n}}$$

This does not have a normal distribution but it is closer to normal the bigger n is. The quantity $n - 1$ is called the *degrees of freedom* of the t value.

The P value is the (one- or two-tailed) tail probability

The P value for a t-test is the probability of the value of a t variable falling farther from the mean than the value of t that we observed.



This can be one-tailed or two-tailed (these differ by a factor of 2 since the t distribution is symmetrical) depending on whether we are interested in departures in both directions.

Values of t for 0.05 tail probability

Degrees of freedom ($n - 1$)	t	Degrees of freedom ($n - 1$)	t
1	12.71	9	2.262
2	4.303	10	2.228
3	3.182	15	2.131
4	2.776	20	2.086
5	2.571	30	2.042
6	2.447	40	2.021
7	2.365	50	2.009
8	2.306		

Note that, as the number of degrees of freedom ($n - 1$) rises, the value of t falls toward 1.95996 which would be the multiple to use for the true σ . This is because our estimated standard deviation \hat{s} is getting close to the true value.

Elaborate tables exist to compute values of t for other values of P . Or of course you can just let R calculate it for you.

Why 0.05?

No real reason.

It is mostly an historical accident of the values R. A. Fisher chose to use in his incredibly influential 1922 book *Statistical Methods for Research Workers*.

The more cautious you need to be the smaller tail probability P you should choose. (Note: sometimes people describe the non-tail probability, such as 95%, instead).

Statistical tests and confidence intervals

If someone suggests that the true μ is (say) 3.14159, we can test this by seeing whether 3.14159 is within the confidence interval. If it isn't, we reject that hypothesis (the *null hypothesis*).

Alternatively, we can calculate for what value of P the suggested value (3.14159 in this case) is just within the confidence interval. We might get, for example, 0.09. That means it is within the interval calculated for a more extreme tail probability 0.05, Departure of the mean from 3.14159 is *not significant*. We can report the 0.09 to indicate to the reader how close to significance it was.

Getting all worried about whether 0.09 is “significantly larger than 0.05” is not worth it. If we think about the “significance of the significance”, that way lies madness!

The 0.05 is the “type I error” of the test. It is the fraction of the time (when the null hypothesis is true) that we get a “false positive” and reject it.

One-tailed test

If we just want to know whether our mean is significantly bigger than 3.14159, but do not want to get excited if it is smaller, we do a one-tailed t test. We get the tail probability of 0.05 on one side, and 0.05 on the other (before we used 0.025 on each). Then we ignore the uninteresting tail (in this case the upper tail – think about it ...)

This will make sense, for example, if we want to get excited about a drug and develop it further if it causes us to lose weight, but not if it causes us to gain weight.

Paired two-sample t -test

Suppose we have two samples of the same size (say 25 numbers) and the corresponding numbers pair naturally. That is, there are some sources of error (“noise”) that are shared by both members of a pair. For example, we might do before-and-after pairs of measurements after giving a drug. Or measure gene expression levels of a gene on two samples (one European and one African) on the same day, and do pairs of individuals, one from each continent, on 25 different days.

The difference between two independent normally-distributed quantities is itself normally distributed. So to measure whether the “after” member of the pair is different from the “before” member, we just subtract them, and test whether the mean of this sample of differences is 0. It is then just a one-sample t test of the sort we used above. The standard deviation you use is the standard deviation of the differences, not of the individual members of the pair.

If the shared noise is large, this can be incredibly more appropriate than the next test ...

(Unpaired) two-sample t test

Suppose the two samples, which might even have different numbers of points, are drawn independently – there is no connection between point 18 from one sample, and point 18 from another. We want to compare the means of the two samples.

Assuming both are drawn from normal distributions *with the same (unknown) true variance* we can do a t test this way:

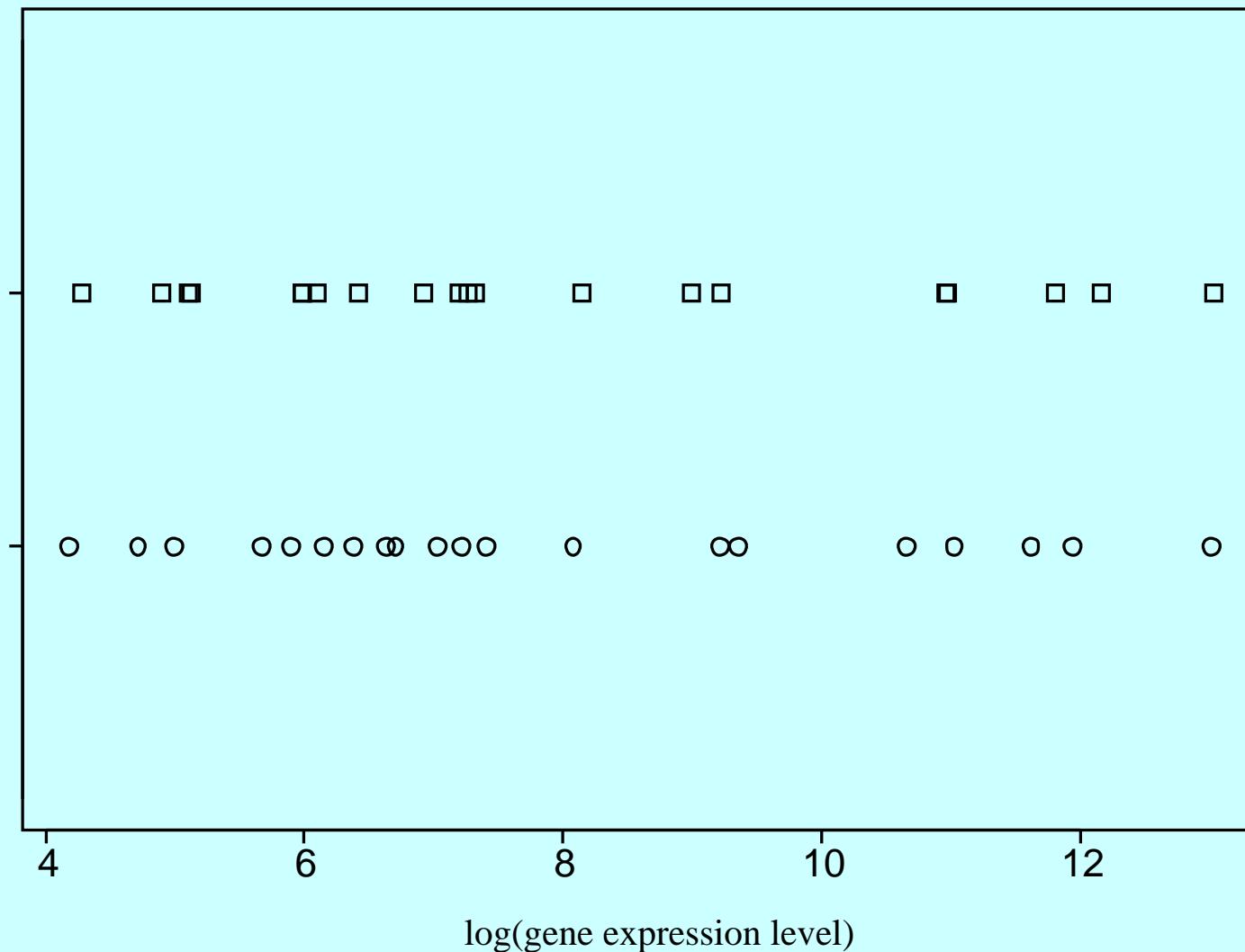
- Suppose the samples are x_1, x_2, \dots, x_m and y_1, y_2, \dots, y_n , compute the means of the two samples (\bar{x} and \bar{y}).
- Compute a pooled estimate of the variance:

$$\hat{s}^2 = \frac{\sum_{i=1}^m (x_i - \bar{x})^2 + \sum_{j=1}^n (y_j - \bar{y})^2}{n + m - 2}$$

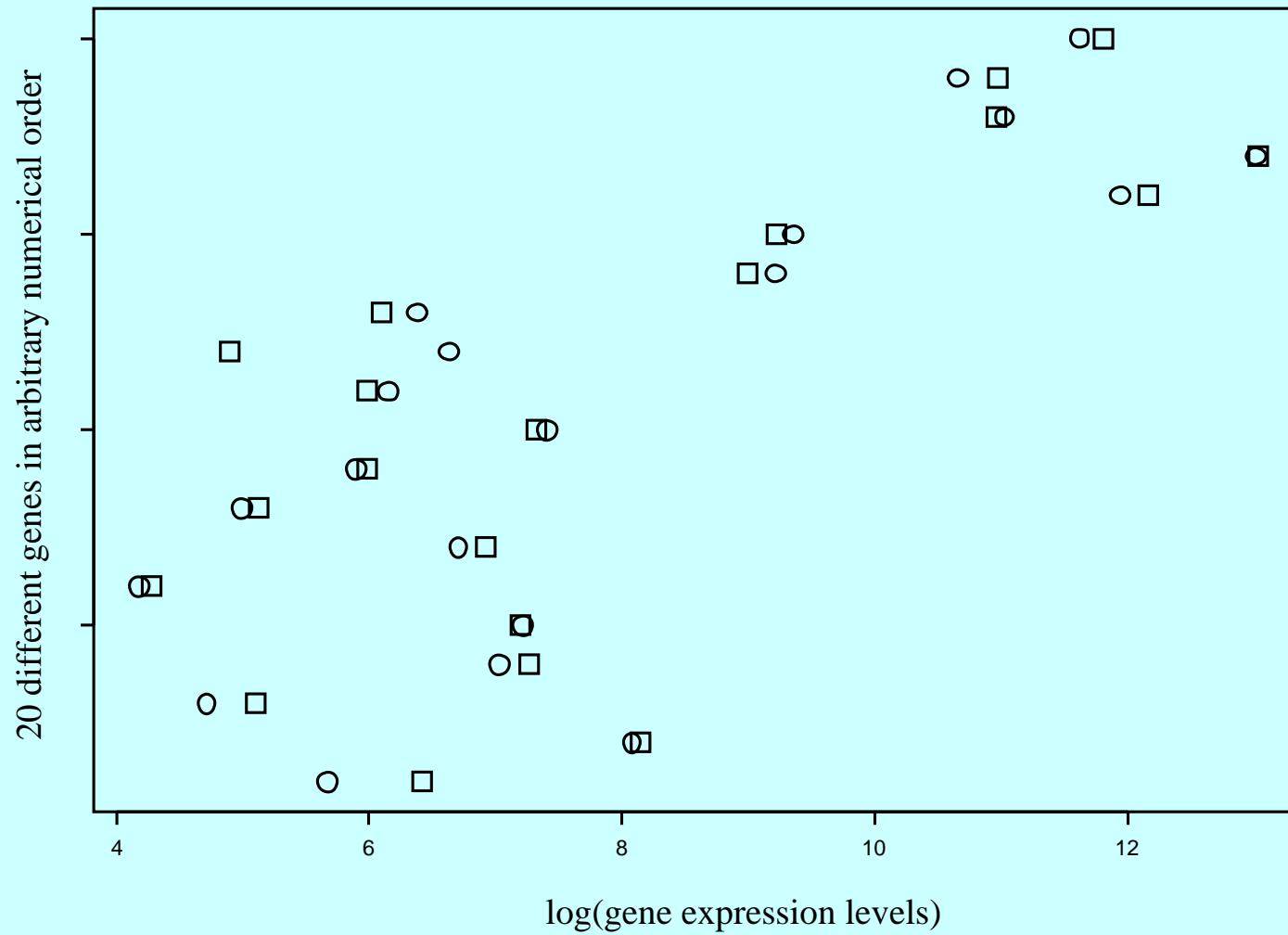
- The degrees of freedom is that denominator: $(n - 1) + (m - 1)$ (or $n + m - 2$).
- Compute the confidence interval on $\bar{x} - \bar{y}$ (using that estimate of the variance, and its square root times $\sqrt{1/m + 1/n}$ as the standard deviation). That is appropriate under the null hypothesis that the difference is 0. See whether 0 is in it. You can do either a two-tailed or a one-tailed test.

Gene expression levels in two individuals

Again from the Storer and Akey results, here are the gene expression levels for 20 genes, shown as unpaired. One individual is squares, the other is circles. (Storey et al., 2007, *Amer. J. Human Genet.*)



The same gene expression levels – paired



Using R to do a one-sample t test

```
> t.test(x[,1])
```

One Sample t-test

```
data: x[, 1]
t = 13.5484, df = 19, p-value = 3.251e-11
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 6.674685 9.113771
sample estimates:
mean of x
7.894228
```

A two-sample t-test, unpaired

```
> t.test(x[,1],x[,2])
```

Welch Two Sample t-test

data: x[, 1] and x[, 2]

t = -8e-04, df = 37.984, p-value = 0.9994

alternative hypothesis: true difference in means is
not equal to 0

95 percent confidence interval:

-1.686258 1.684984

sample estimates:

mean of x mean of y

7.894228 7.894865

A two-sample t-test, which pairs corresponding values

```
> t.test(x[,1],x[,2],paired=TRUE)
```

Paired t-test

data: x[, 1] and x[, 2]

t = -0.006, df = 19, p-value = 0.9953

alternative hypothesis: true difference in means is
not equal to 0

95 percent confidence interval:

-0.2223615 0.2210875

sample estimates:

mean of the differences

-0.000637

From ScienceNews, last year

Statistical insignificance

Nowhere are the problems with statistics more blatant than in studies of genetic influences on disease. In 2007, for instance, researchers combing the medical literature found numerous studies linking a total of 85 genetic variants in 70 different genes to acute coronary syndrome, a cluster of heart problems. When the researchers compared genetic tests of 811 patients that had the syndrome with a group of 650 (matched for sex and age) that didn't, only one of the suspect gene variants turned up substantially more often in those with the syndrome — a number to be expected by chance.

“Our null results provide no support for the hypothesis that any of the 85 genetic variants tested is a susceptibility factor” for the syndrome, the researchers reported in the *Journal of the American Medical Association*.

How could so many studies be wrong? Because their conclusions relied on “statistical significance,” a concept at the heart of the mathematical analysis of modern scientific experiments.

Tom Siegfried. 2010. Odds are, it's wrong. *ScienceNews*, March 27.

Miscellaneous

If the two samples seem to have different variances, maybe you are on the wrong scale. Maybe their logarithms are normally distributed and have equal variances (more nearly anyway). If we are dealing with measurements that can't be negative, such as weights, that is a more natural assumption, and sometimes it homogenizes the variances nicely.

Get familiar with the lognormal distribution, which just means the logs are normally distributed. (It doesn't matter whether natural logs or common logs as those are just multiples of each other).



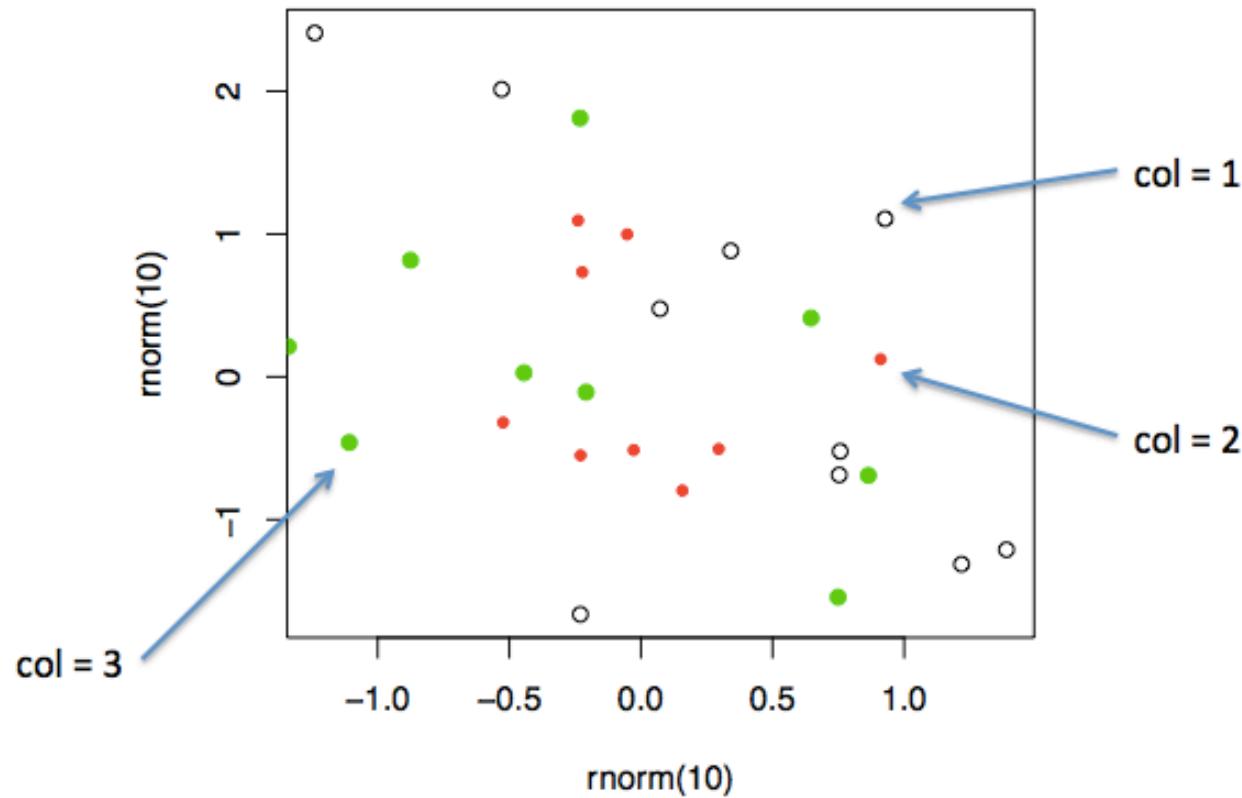
Plotting and Color in R

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

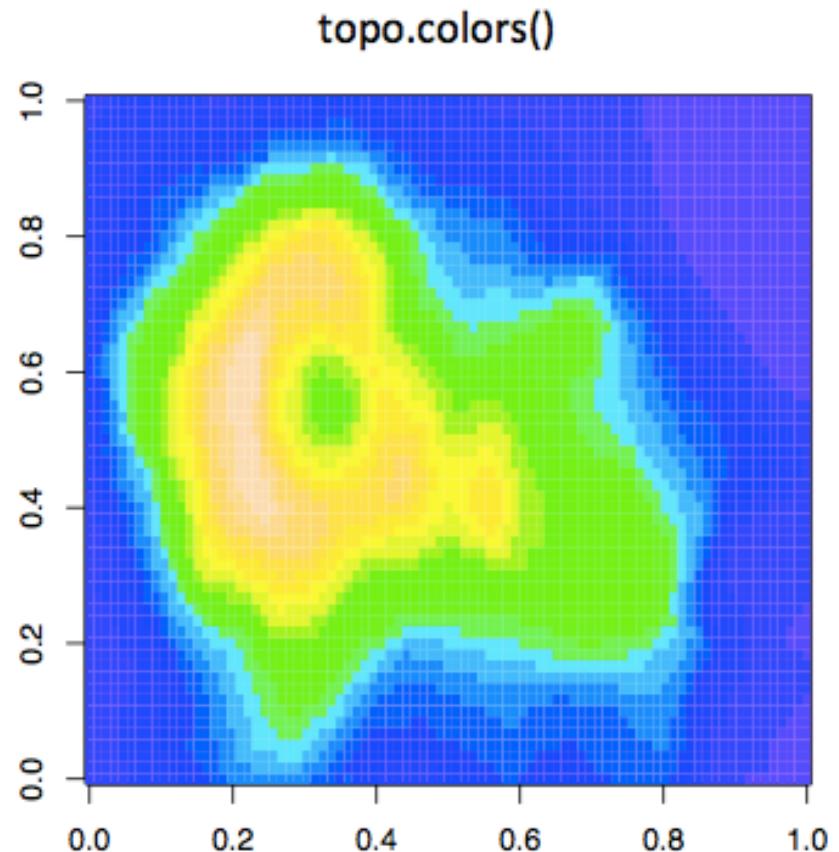
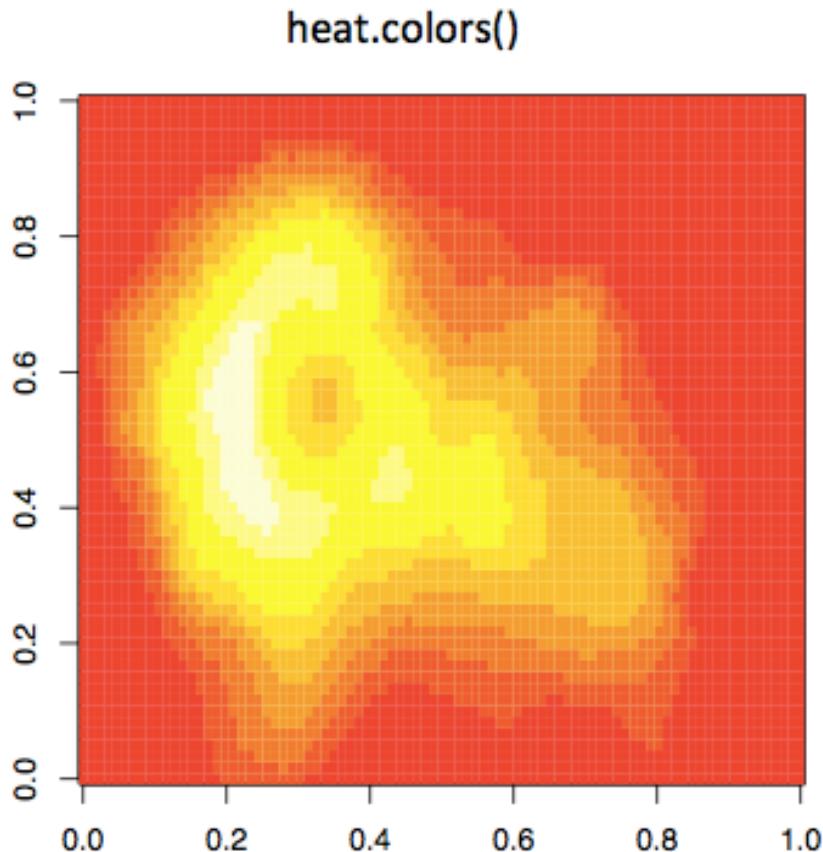
Plotting and Color

- The default color schemes for most plots in R are horrendous
 - I don't have good taste and even I know that
- Recently there have been developments to improve the handling/specification of colors in plots/graphs/etc.
- There are functions in R and in external packages that are very handy

Colors 1, 2, and 3



Default Image Plots in R



Color Utilities in R

- The `grDevices` package has two functions
 - `colorRamp`
 - `colorRampPalette`
- These functions take palettes of colors and help to interpolate between the colors
- The function `colors()` lists the names of colors you can use in any plotting function

Color Palette Utilities in R

- `colorRamp`: Take a palette of colors and return a function that takes values between 0 and 1, indicating the extremes of the color palette (e.g. see the 'gray' function)
- `colorRampPalette`: Take a palette of colors and return a function that takes integer arguments and returns a vector of colors interpolating the palette (like `heat.colors` or `topo.colors`)

colorRamp

[,1] [,2] [,3] corresponds to [Red] [Blue] [Green]

```
> pal <- colorRamp(c("red", "blue"))

> pal(0)
[,1] [,2] [,3]
[1,] 255     0     0

> pal(1)
[,1] [,2] [,3]
[1,] 0     0   255

> pal(0.5)
[,1] [,2] [,3]
[1,] 127.5  0 127.5
```

colorRamp

```
> pal(seq(0, 1, len = 10))  
      [,1] [,2]      [,3]  
[1,] 255.00000 0 0  
[2,] 226.66667 0 28.33333  
[3,] 198.33333 0 56.66667  
[4,] 170.00000 0 85.00000  
[5,] 141.66667 0 113.33333  
[6,] 113.33333 0 141.66667  
[7,] 85.00000 0 170.00000  
[8,] 56.66667 0 198.33333  
[9,] 28.33333 0 226.66667  
[10,] 0.00000 0 255.00000
```

colorRampPalette

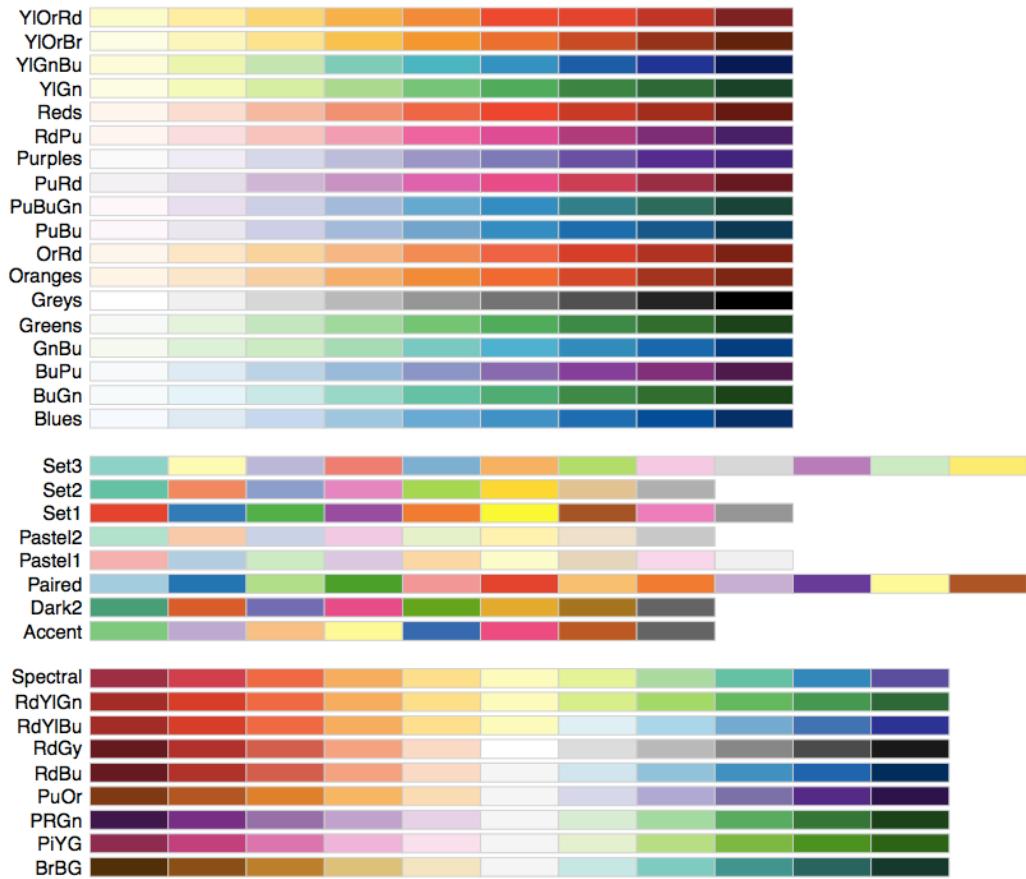
```
> pal <- colorRampPalette(c("red", "yellow"))

> pal(2)
[1] "#FF0000" "#FFFF00"

> pal(10)
[1] "#FF0000" "#FF1C00" "#FF3800" "#FF5500" "#FF7100"
[6] "#FF8D00" "#FFAA00" "#FFC600" "#FFE200" "#FFFF00"
```

RColorBrewer Package

- One package on CRAN that contains interesting/useful color palettes
- There are 3 types of palettes
 - Sequential
 - Diverging
 - Qualitative
- Palette information can be used in conjunction with the `colorRamp()` and `colorRampPalette()`



RColorBrewer and colorRampPalette

```
> library(RColorBrewer)

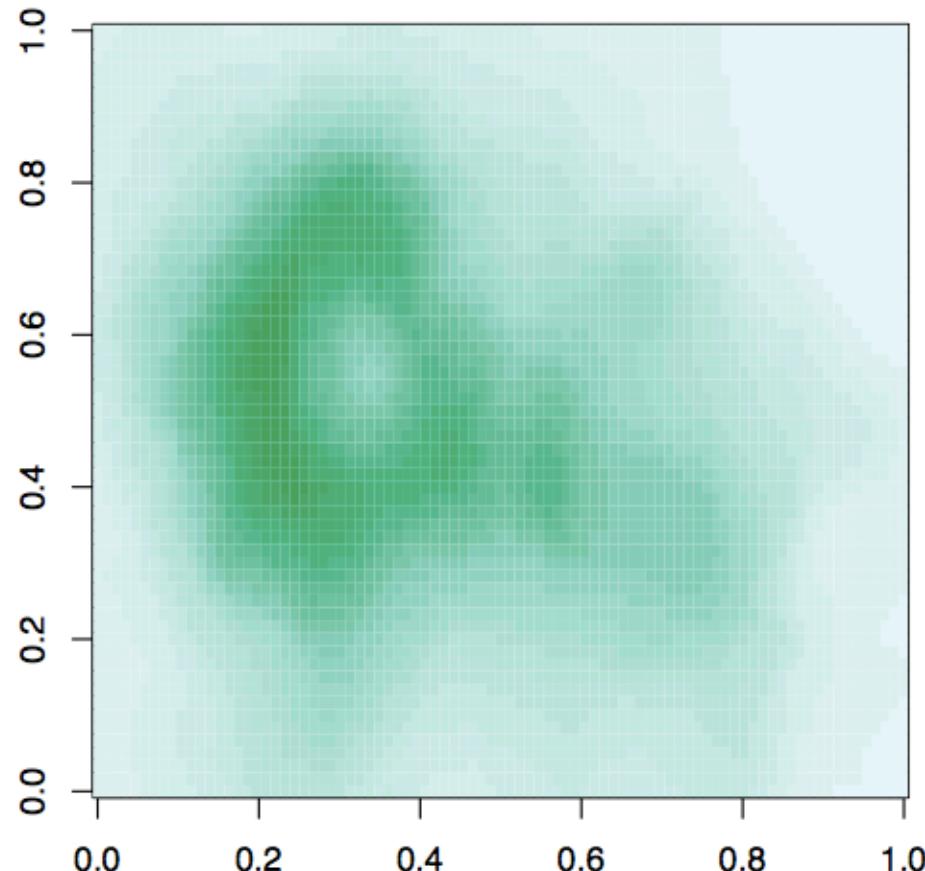
> cols <- brewer.pal(3, "BuGn")

> cols
[1] "#E5F5F9" "#99D8C9" "#2CA25F"

> pal <- colorRampPalette(cols)

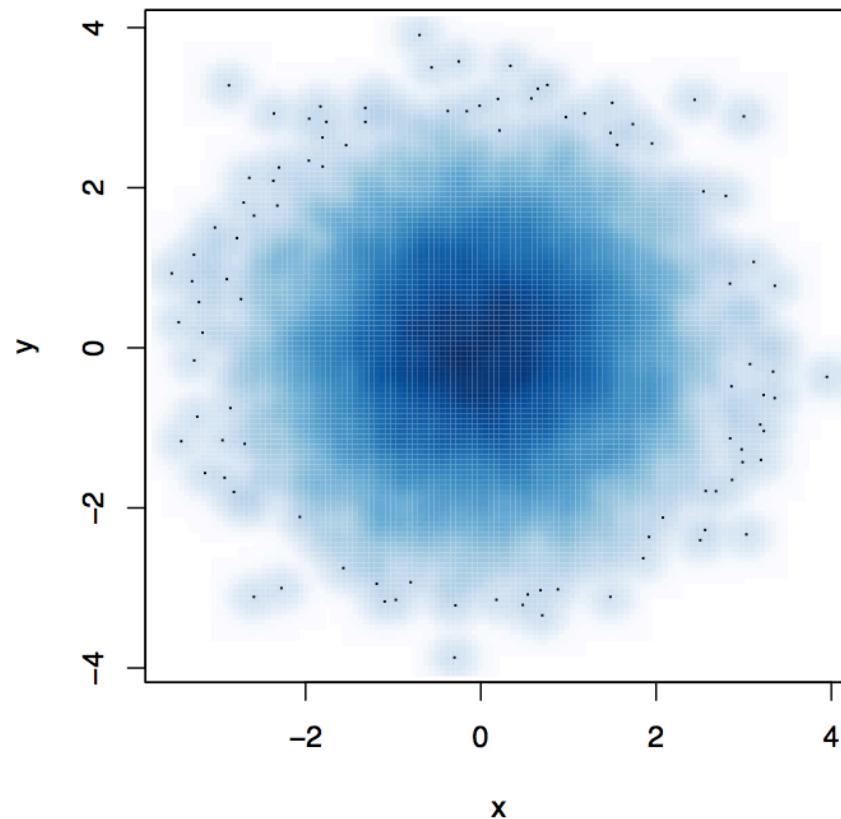
> image(volcano, col = pal(20))
```

RColorBrewer and colorRampPalette



The smoothScatter function

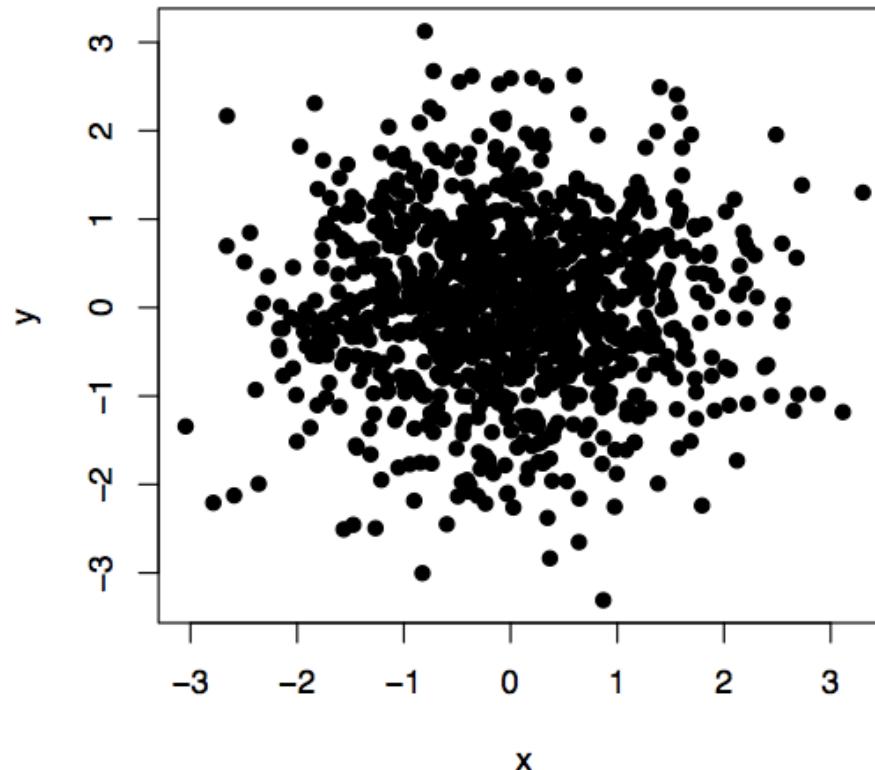
```
x <- rnorm(10000)
y <- rnorm(10000)
smoothScatter(x, y)
```



Some other plotting notes

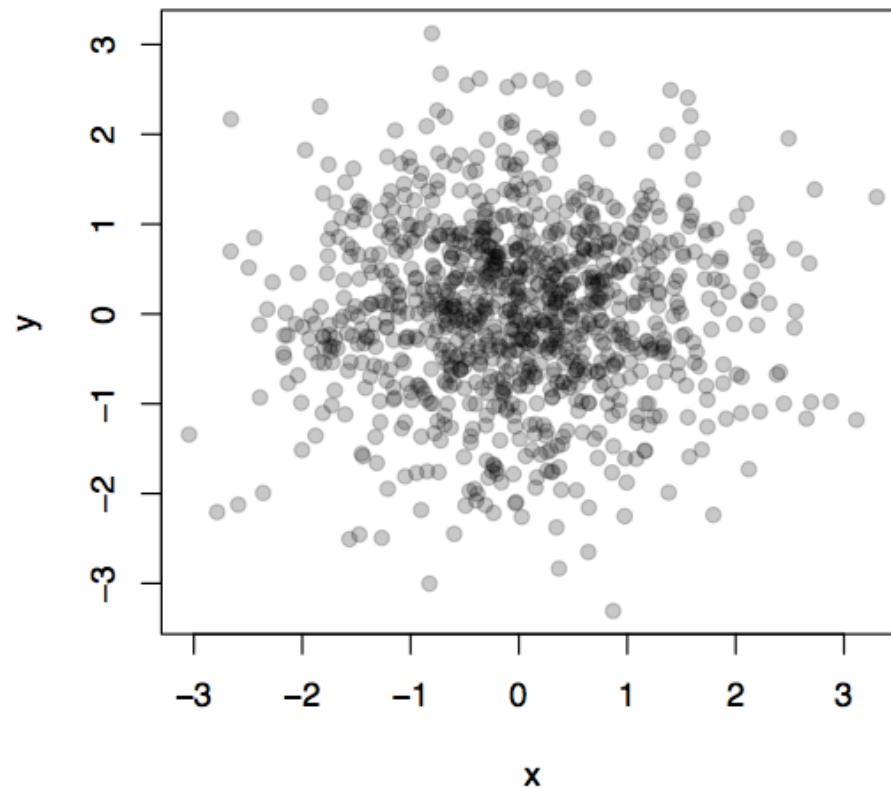
- The `rgb` function can be used to produce any color via red, green, blue proportions
- Color transparency can be added via the `alpha` parameter to `rgb`
- The `colorspace` package can be used for a different control over colors

Scatterplot with no transparency



```
plot(x, y, pch = 19)
```

Scatterplot with transparency



```
plot(x, y, col = rgb(0, 0, 0, 0.2), pch = 19)
```

Summary

- Careful use of colors in plots/maps/etc. can make it easier for the reader to get what you're trying to say (why make it harder?)
- The `RColorBrewer` package is an R package that provides color palettes for sequential, categorical, and diverging data
- The `colorRamp` and `colorRampPalette` functions can be used in conjunction with color palettes to connect data to colors
- Transparency can sometimes be used to clarify plots with many points

Sums of Independent Random Variables

Consider the sum of two independent discrete random variables X and Y whose values are restricted to the non-negative integers. Let $f_X(\cdot)$ denote the probability distribution of X and $f_Y(\cdot)$ denote the probability distribution of Y . The distribution of their sum $Z = X + Y$ is given by the **discrete convolution formula**.

Theorem Discrete Convolution Formula. The random variable $Z = X + Y$ has probability distribution $f_Z(\cdot)$ given by

$$f_Z(z) = f_{X+Y}(z) = P(Z = z) = \sum_{x=0}^z f_X(x)f_Y(z-x)$$

for $z = 0, 1, \dots$.

Proof: For each z , the event $[Z = z]$ is the union of the disjoint events $[X = x \text{ and } Y = z - x]$ for $x = 0, 1, \dots, z$. Consequently,

$$\begin{aligned} P(Z = z) &= f_Z(z) = \sum_{x=0}^z P(X = x \text{ and } Y = z - x) \\ &= \sum_{x=0}^z f_X(x)f_Y(z-x) \end{aligned}$$

where the last step follows by independence.

Let X_1 and X_2 be independent binomial random variables having the same probability of success. Their sum is again binomial.

Corollary 1 Sum of Binomial Random Variables. Let X_1 and X_2 be independent binomial random variables where X_i has a $\text{Binomial}(n_i, p)$ distribution for $i = 1, 2$. Then

$X_1 + X_2$ has a binomial distribution with $n_1 + n_2$ trials and probability of success p

Let X_1, X_2, \dots, X_k be independent binomial random variables where X_i has a $\text{Binomial}(n_i, p)$ distribution for $i = 1, 2, \dots, k$. Then

$X_1 + X_2 + \dots + X_k$ has a $\text{Binomial}(n_1 + n_2 + \dots + n_k, p)$ distribution.

Proof: By the discrete convolution formula, $Z = X_1 + X_2$ has probability distribution

$$P(X_1 + X_2 = z) = f_Z(z) = \sum_{x=0}^z f_{X_1}(x)f_{X_2}(z-x)$$

so

$$\begin{aligned} f_Z(z) &= \sum_{x=0}^z \binom{n_1}{x} p^x (1-p)^{n_1-x} \binom{n_2}{z-x} p^{z-x} (1-p)^{n_2-(z-x)} \\ &= p^z (1-p)^{n_1+n_2-z} \sum_{x=0}^z \binom{n_1}{x} \binom{n_2}{z-x} \end{aligned}$$

Now, equating the coefficients of s^y in the binomial expansion of both sides of

$$(1+s)_1^n (1+s)_2^n = (1+s)^{n_1+n+2}$$

we conclude that

$$\sum_{x=0}^z \binom{n_1}{x} \binom{n_2}{z-x} = \binom{n_1+n_2}{z}$$

The case for several binomial random variables follows by induction.

Remark: Note that the sample sizes add but the success probability remains the same.

Corollary 2 Sum of Poisson Random Variables. Let X_1 and X_2 be independent Poisson random variables where X_i has a Poisson (λ_i) distribution for $i = 1, 2$. Then

$$X_1 + X_2 \quad \text{has a Poisson distribution with } \lambda_1 + \lambda_2$$

Let X_1, X_2, \dots, X_k be independent Poisson random variables where X_i has a Poisson (λ_i) distribution for $i = 1, 2, \dots, k$. Then

$$X_1 + X_2 + \dots + X_k \quad \text{has a Poisson}(\lambda_1 + \lambda_2 + \dots + \lambda_k) \quad \text{distribution.}$$

Proof: By the discrete convolution formula, $Z = X_1 + X_2$ has probability distribution

$$P(X_1 + X_2 = z) = f_Z(z) = \sum_{x=0}^z f_{X_1}(x)f_{X_2}(z-x)$$

so

$$\begin{aligned} f_Z(z) &= \sum_{x=0}^z \frac{\lambda_1^x}{x!} e^{-\lambda_1} \frac{\lambda_2^{z-x}}{(z-x)!} e^{-\lambda_2} \\ &= e^{-(\lambda_1 + \lambda_2)} \sum_{x=0}^z \frac{\lambda_1^x}{x!} \frac{\lambda_2^{z-x}}{(z-x)!} \end{aligned}$$

Use the binomial formula

$$(a+b)^m = \sum_{x=0}^m \binom{m}{x} a^x b^{m-x}$$

with $m = z, a = \lambda_1$ and $b = \lambda_2$ after multiplying and dividing by $z!$, to conclude that

$$\sum_{x=0}^z \frac{\lambda_1^x}{x!} \frac{\lambda_2^{z-x}}{(z-x)!} = \frac{(\lambda_1 + \lambda_2)^z}{z!}$$

and the result is established.

Remark: Note that the rate parameters λ_i add.

t Table

cum. prob	<i>t_{.50}</i>	<i>t_{.75}</i>	<i>t_{.80}</i>	<i>t_{.85}</i>	<i>t_{.90}</i>	<i>t_{.95}</i>	<i>t_{.975}</i>	<i>t_{.99}</i>	<i>t_{.995}</i>	<i>t_{.999}</i>	<i>t_{.9995}</i>
one-tail	0.50	0.25	0.20	0.15	0.10	0.05	0.025	0.01	0.005	0.001	0.0005
two-tails	1.00	0.50	0.40	0.30	0.20	0.10	0.05	0.02	0.01	0.002	0.001
df											
1	0.000	1.000	1.376	1.963	3.078	6.314	12.71	31.82	63.66	318.31	636.62
2	0.000	0.816	1.061	1.386	1.886	2.920	4.303	6.965	9.925	22.327	31.599
3	0.000	0.765	0.978	1.250	1.638	2.353	3.182	4.541	5.841	10.215	12.924
4	0.000	0.741	0.941	1.190	1.533	2.132	2.776	3.747	4.604	7.173	8.610
5	0.000	0.727	0.920	1.156	1.476	2.015	2.571	3.365	4.032	5.893	6.869
6	0.000	0.718	0.906	1.134	1.440	1.943	2.447	3.143	3.707	5.208	5.959
7	0.000	0.711	0.896	1.119	1.415	1.895	2.365	2.998	3.499	4.785	5.408
8	0.000	0.706	0.889	1.108	1.397	1.860	2.306	2.896	3.355	4.501	5.041
9	0.000	0.703	0.883	1.100	1.383	1.833	2.262	2.821	3.250	4.297	4.781
10	0.000	0.700	0.879	1.093	1.372	1.812	2.228	2.764	3.169	4.144	4.587
11	0.000	0.697	0.876	1.088	1.363	1.796	2.201	2.718	3.106	4.025	4.437
12	0.000	0.695	0.873	1.083	1.356	1.782	2.179	2.681	3.055	3.930	4.318
13	0.000	0.694	0.870	1.079	1.350	1.771	2.160	2.650	3.012	3.852	4.221
14	0.000	0.692	0.868	1.076	1.345	1.761	2.145	2.624	2.977	3.787	4.140
15	0.000	0.691	0.866	1.074	1.341	1.753	2.131	2.602	2.947	3.733	4.073
16	0.000	0.690	0.865	1.071	1.337	1.746	2.120	2.583	2.921	3.686	4.015
17	0.000	0.689	0.863	1.069	1.333	1.740	2.110	2.567	2.898	3.646	3.965
18	0.000	0.688	0.862	1.067	1.330	1.734	2.101	2.552	2.878	3.610	3.922
19	0.000	0.688	0.861	1.066	1.328	1.729	2.093	2.539	2.861	3.579	3.883
20	0.000	0.687	0.860	1.064	1.325	1.725	2.086	2.528	2.845	3.552	3.850
21	0.000	0.686	0.859	1.063	1.323	1.721	2.080	2.518	2.831	3.527	3.819
22	0.000	0.686	0.858	1.061	1.321	1.717	2.074	2.508	2.819	3.505	3.792
23	0.000	0.685	0.858	1.060	1.319	1.714	2.069	2.500	2.807	3.485	3.768
24	0.000	0.685	0.857	1.059	1.318	1.711	2.064	2.492	2.797	3.467	3.745
25	0.000	0.684	0.856	1.058	1.316	1.708	2.060	2.485	2.787	3.450	3.725
26	0.000	0.684	0.856	1.058	1.315	1.706	2.056	2.479	2.779	3.435	3.707
27	0.000	0.684	0.855	1.057	1.314	1.703	2.052	2.473	2.771	3.421	3.690
28	0.000	0.683	0.855	1.056	1.313	1.701	2.048	2.467	2.763	3.408	3.674
29	0.000	0.683	0.854	1.055	1.311	1.699	2.045	2.462	2.756	3.396	3.659
30	0.000	0.683	0.854	1.055	1.310	1.697	2.042	2.457	2.750	3.385	3.646
40	0.000	0.681	0.851	1.050	1.303	1.684	2.021	2.423	2.704	3.307	3.551
60	0.000	0.679	0.848	1.045	1.296	1.671	2.000	2.390	2.660	3.232	3.460
80	0.000	0.678	0.846	1.043	1.292	1.664	1.990	2.374	2.639	3.195	3.416
100	0.000	0.677	0.845	1.042	1.290	1.660	1.984	2.364	2.626	3.174	3.390
1000	0.000	0.675	0.842	1.037	1.282	1.646	1.962	2.330	2.581	3.098	3.300
Z	0.000	0.674	0.842	1.036	1.282	1.645	1.960	2.326	2.576	3.090	3.291
	0%	50%	60%	70%	80%	90%	95%	98%	99%	99.8%	99.9%
	Confidence Level										

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)

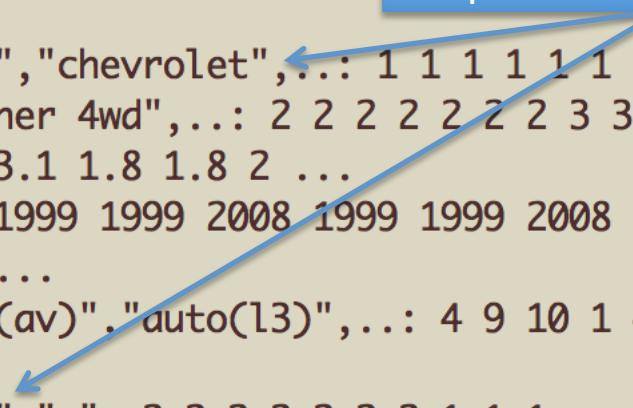
The Basics: qplot()

- 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

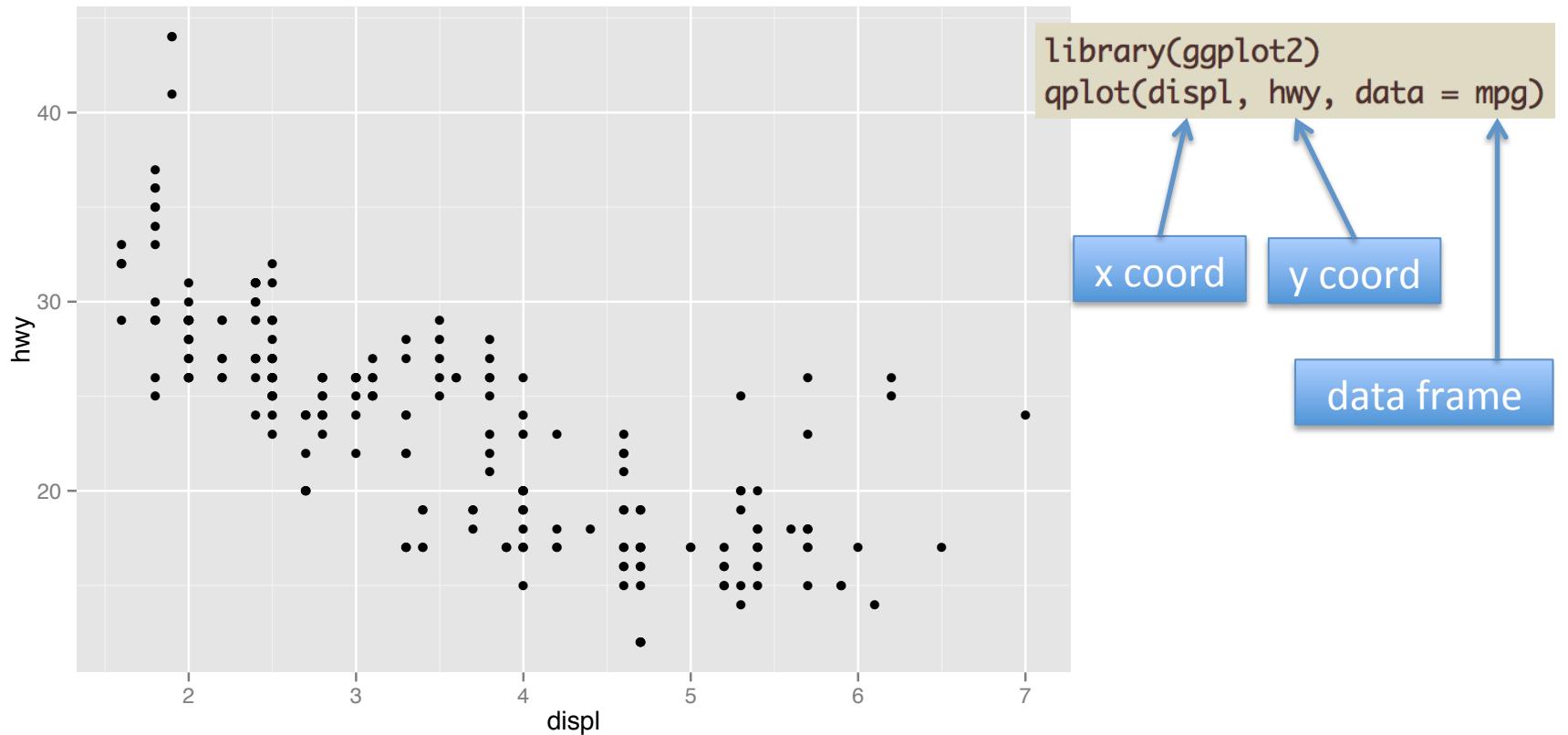
Example Dataset

```
> 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 ...
```

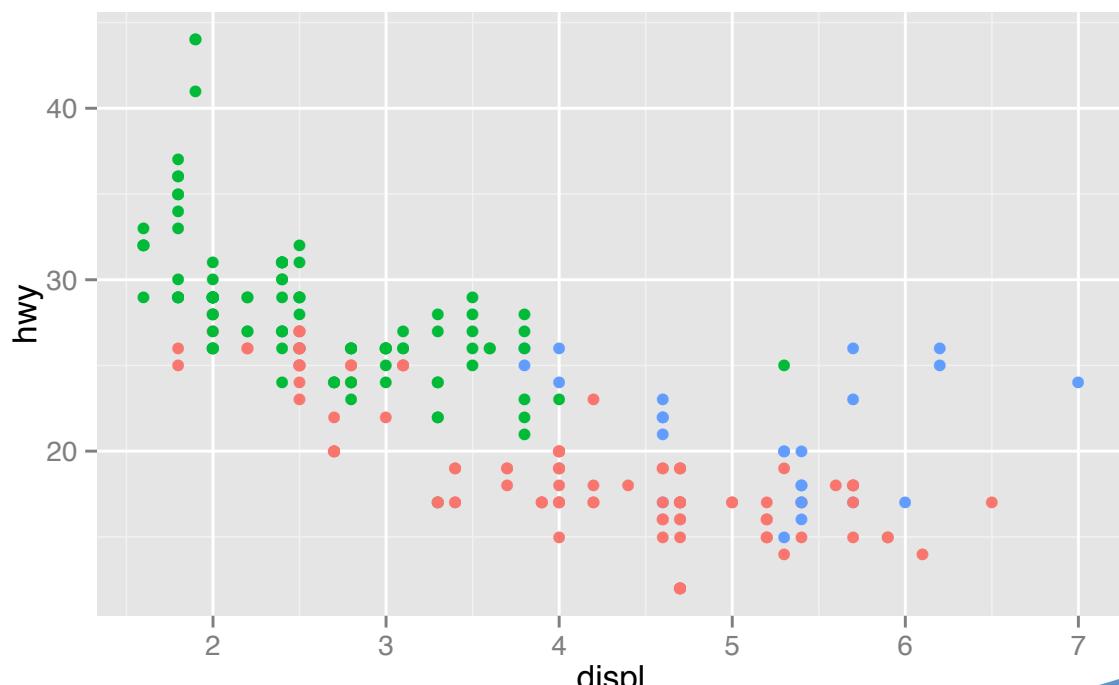
Factor label information
important for annotation



ggplot2 “Hello, world!”



Modifying aesthetics



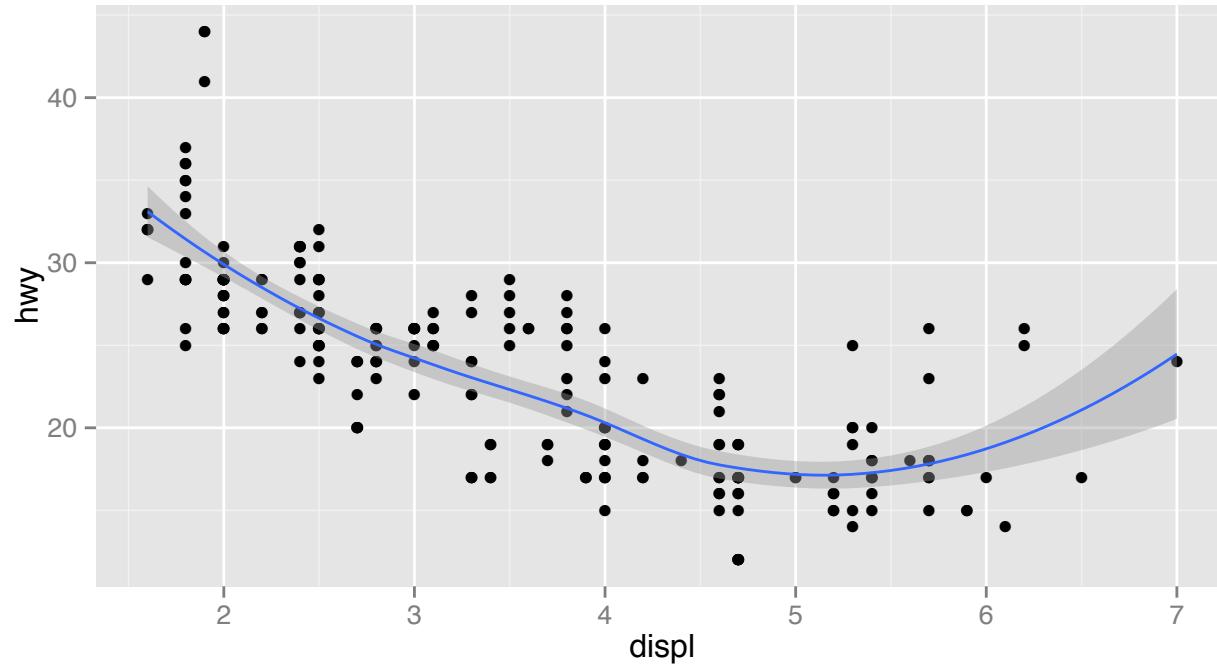
auto legend placement

drv
4
f
r

color aesthetic

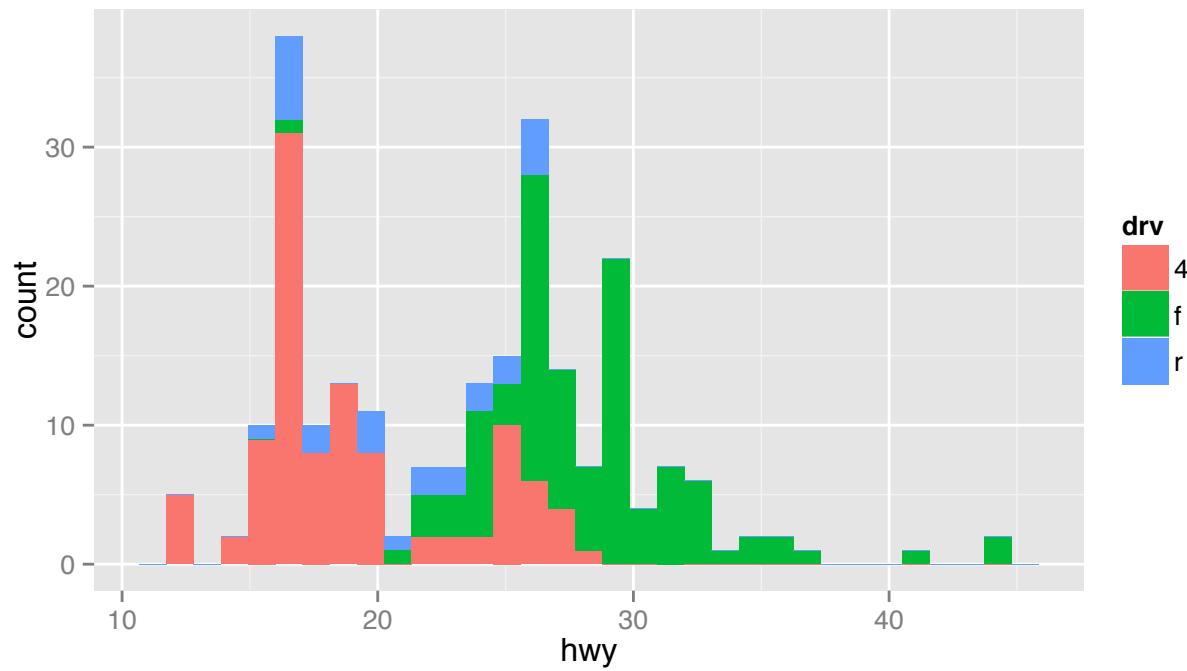
qplot(displ, hwy, data = mpg, color = drv)

Adding a geom



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

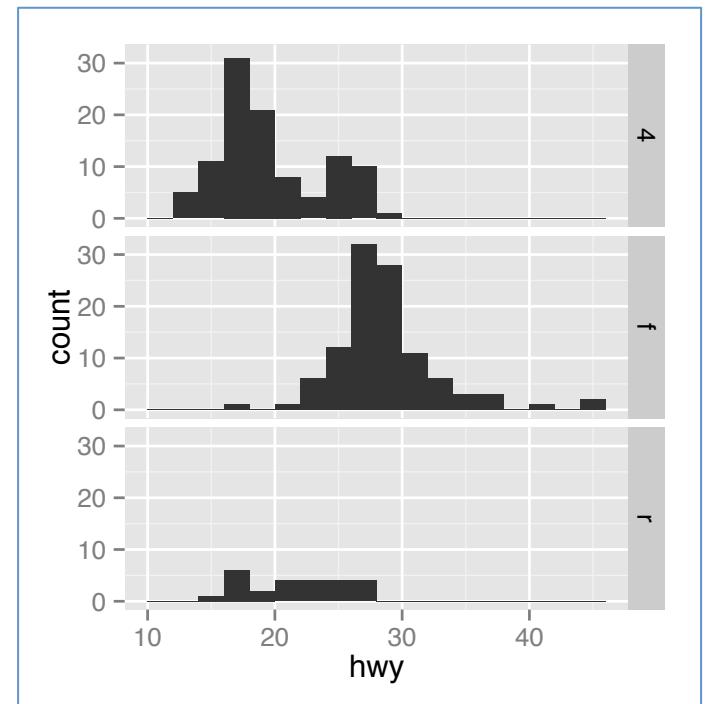
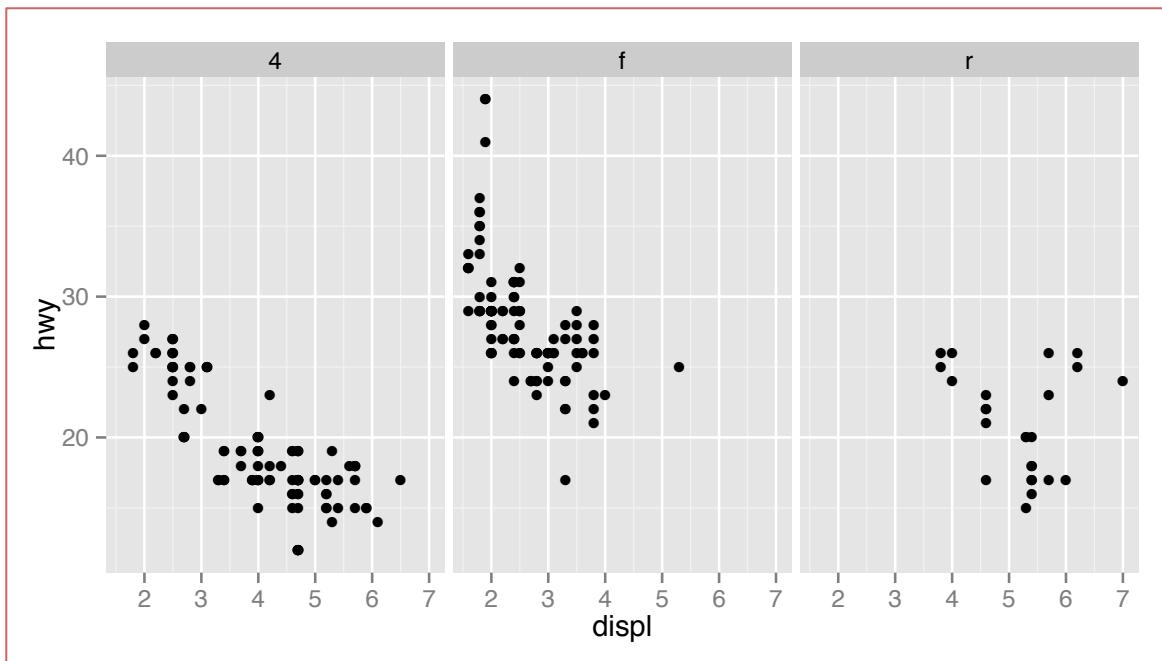
Histograms



```
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>

Exhaled nitric
oxide

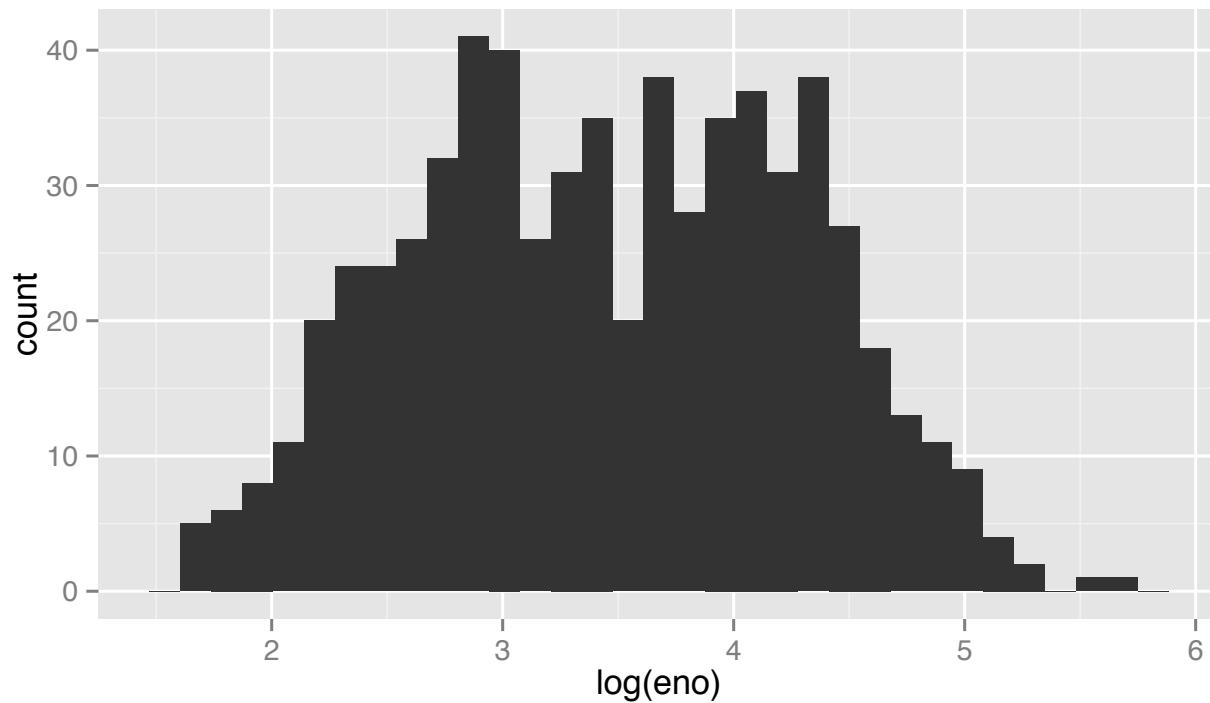
Example: MAACS

```
> 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 ...
```

Sensitized to
mouse allergen

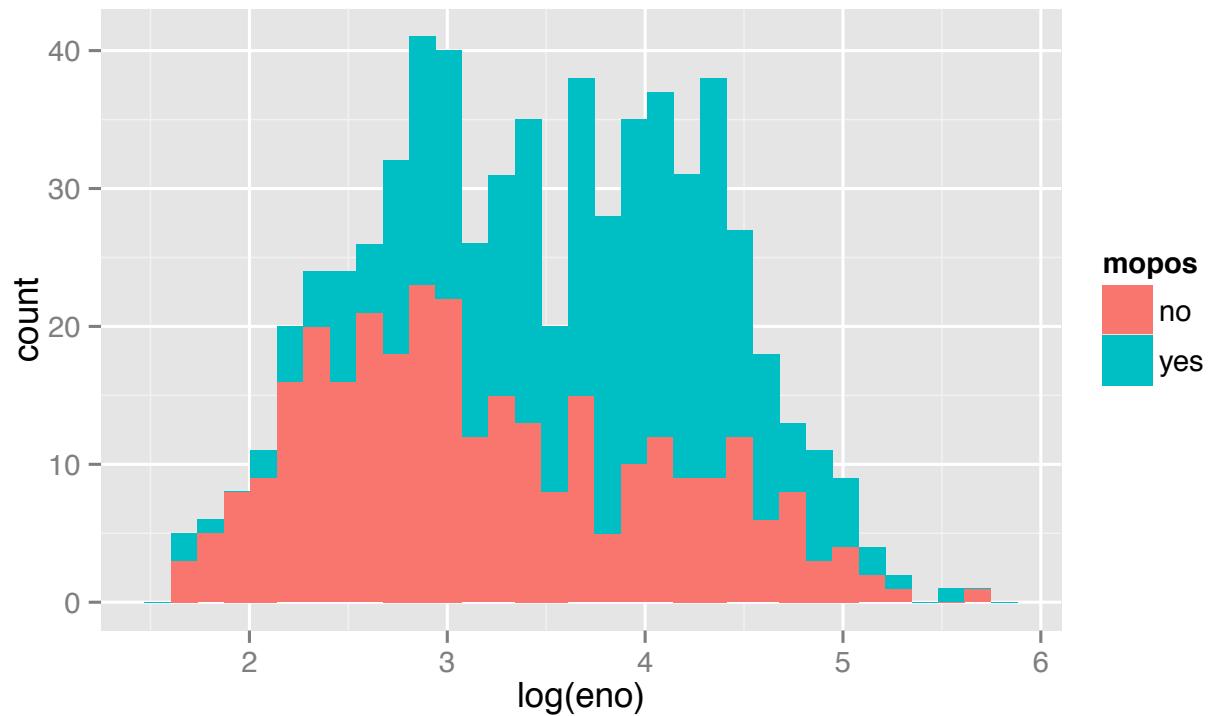
Fine particulate
matter

Histogram of eNO



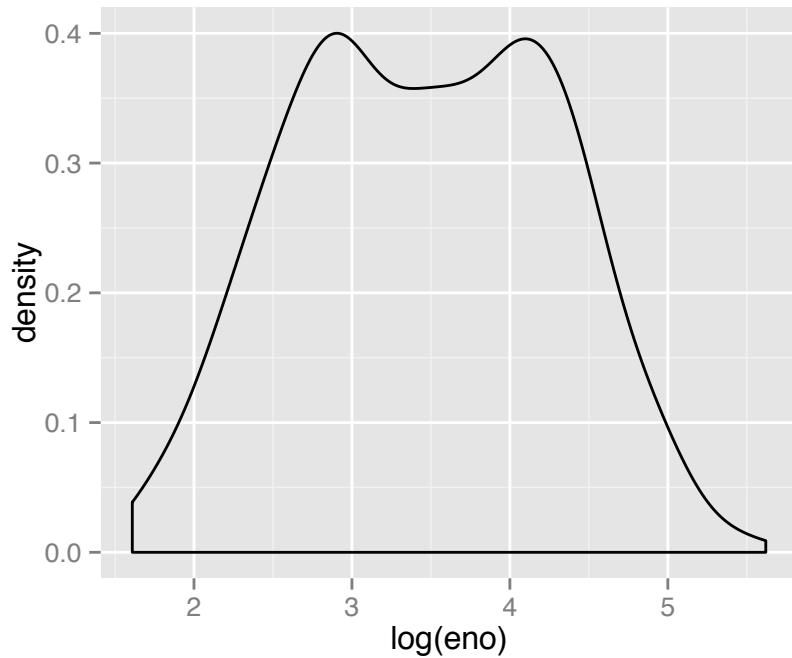
```
qplot(log(eno), data = maacs)
```

Histogram by Group

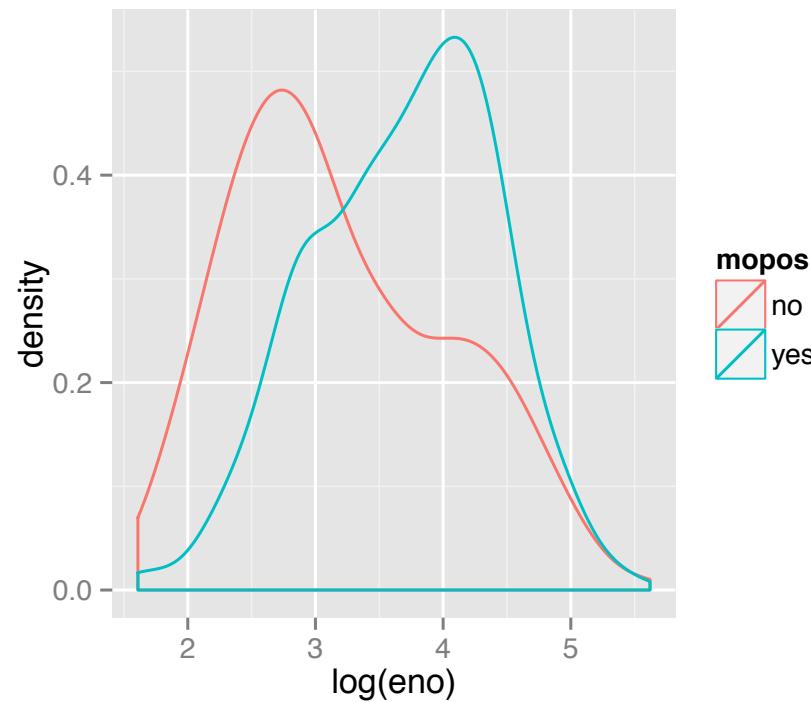


```
qplot(log(eno), data = maacs, fill = mopos)
```

Density Smooth

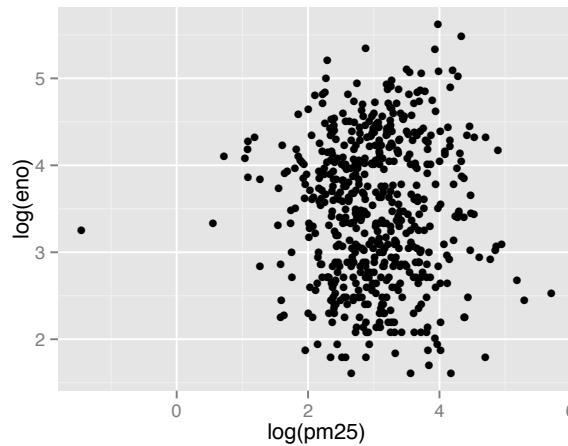


```
qplot(log(eno), data = maacs, geom = "density")
```

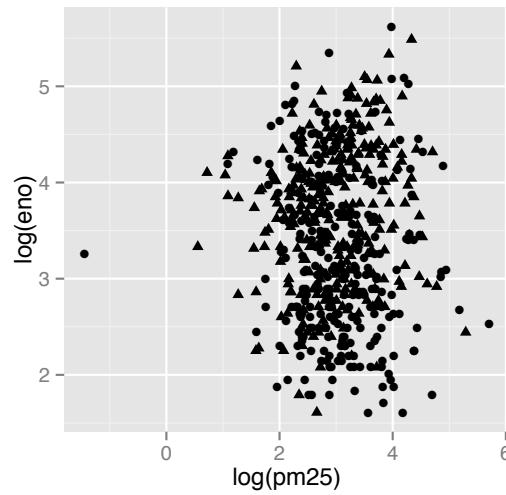


```
qplot(log(eno), data = maacs, geom = "density", color = mpos)
```

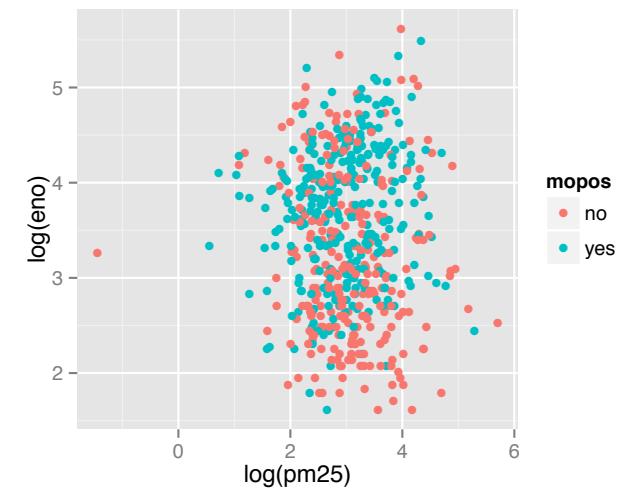
Scatterplots: eNO vs. PM_{2.5}



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

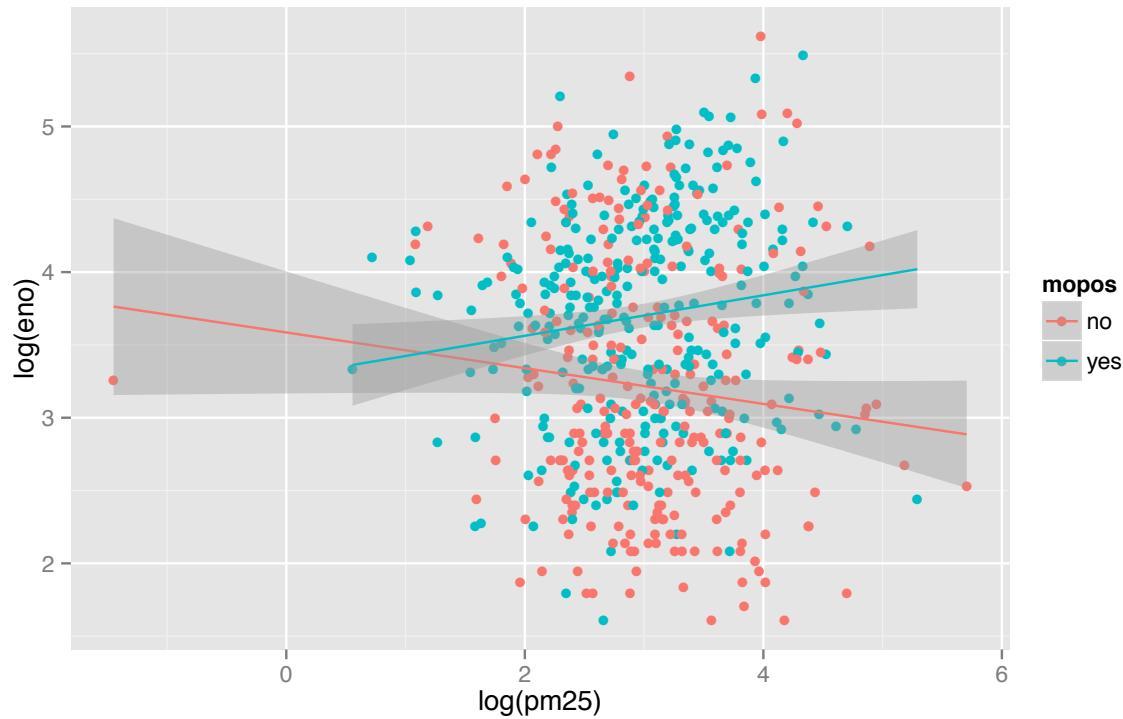


```
qplot(log(pm25), log(eno), data =  
maacs, shape = mpos)
```



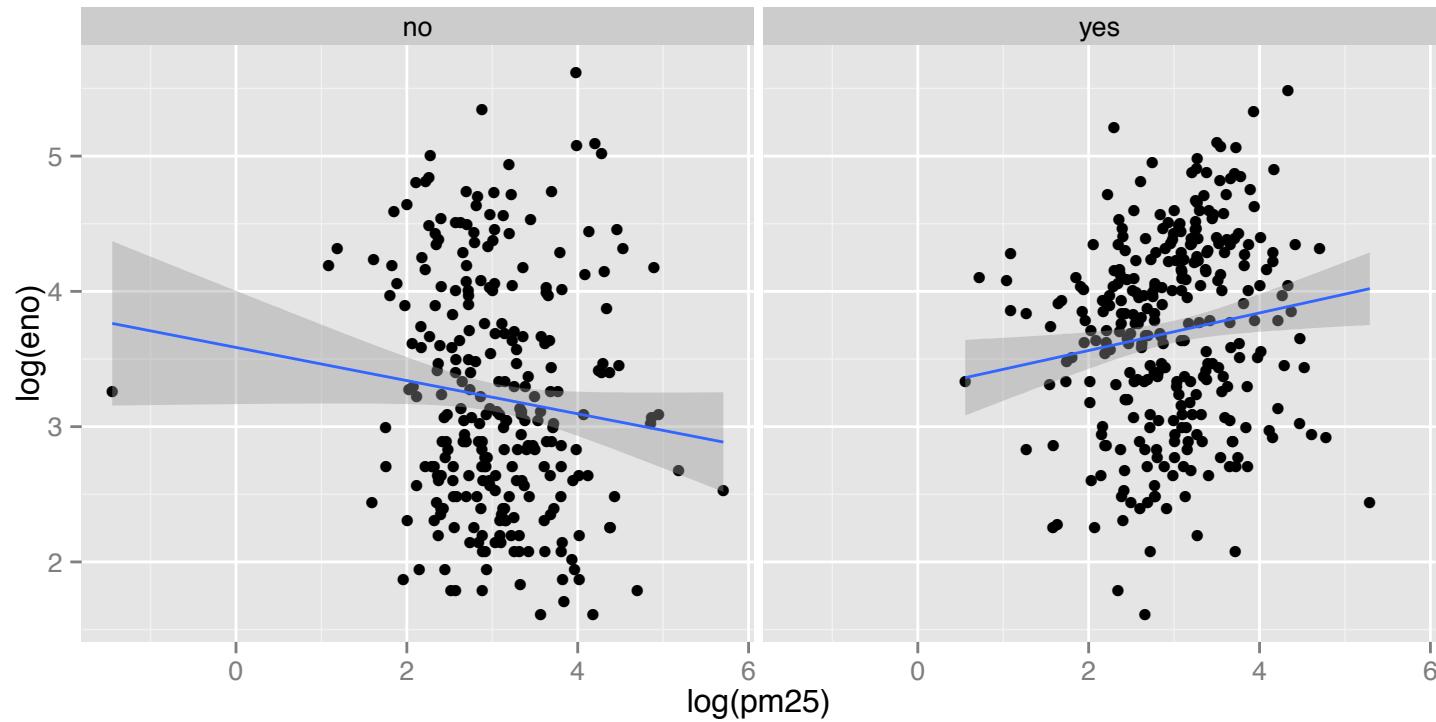
```
qplot(log(pm25), log(eno), data =  
maacs, color = mpos)
```

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 = . ~ mpos)
```

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**

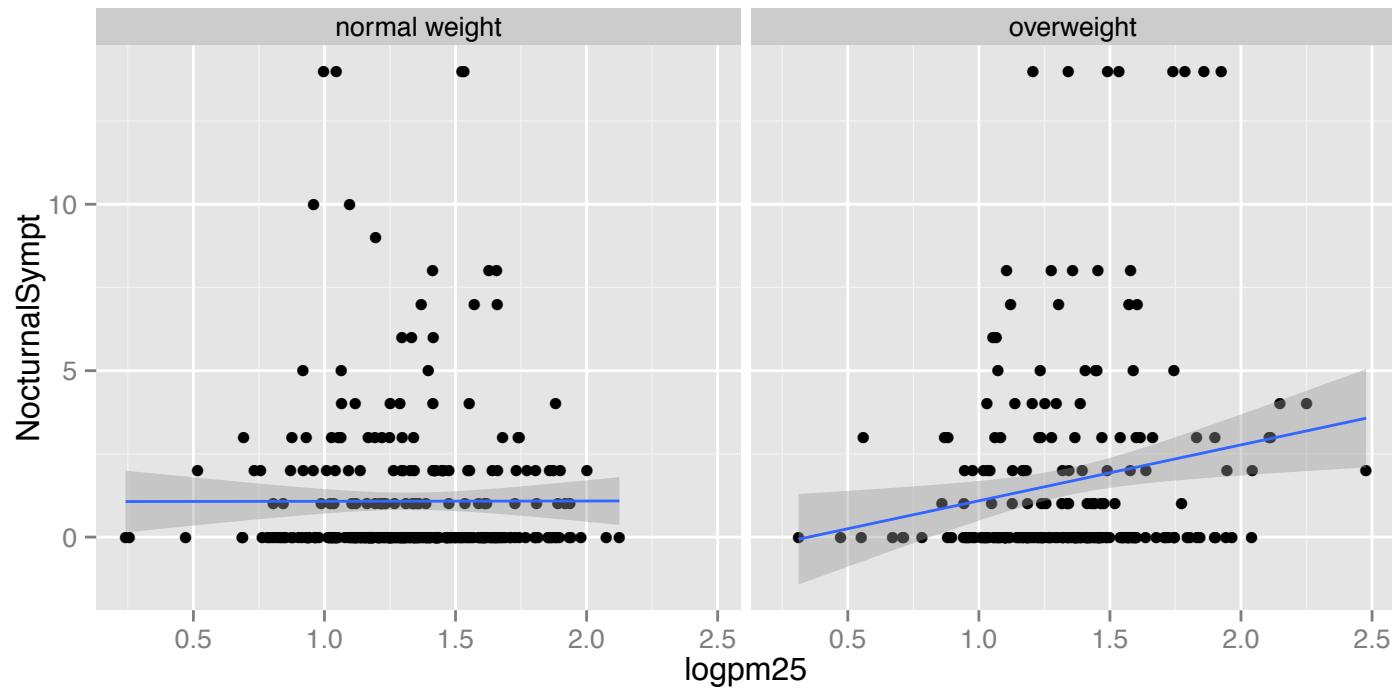
Building Plots with ggplot2

- 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



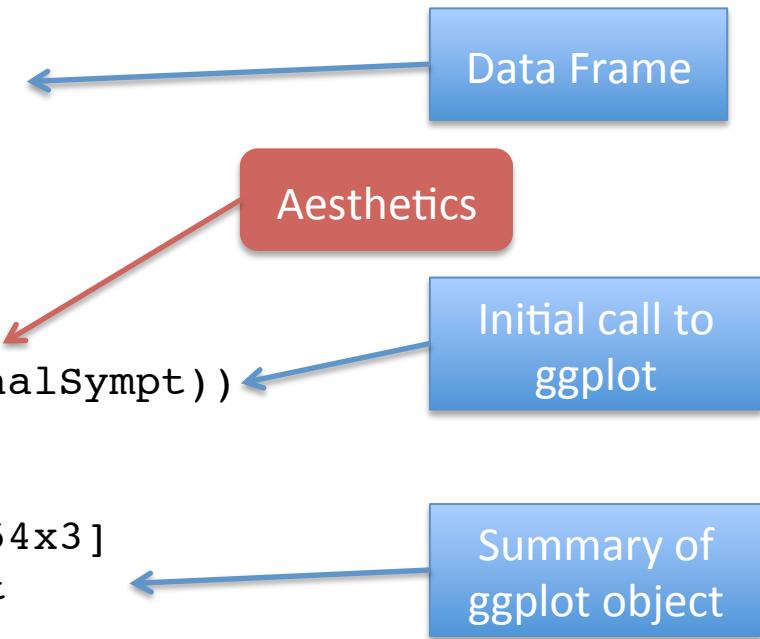
```
qplot(logpm25, NocturnalSympt, data = maacs, facets = . ~ bmicat, geom =  
c("point", "smooth"), method = "lm")
```

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()
```



No Plot Yet!

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

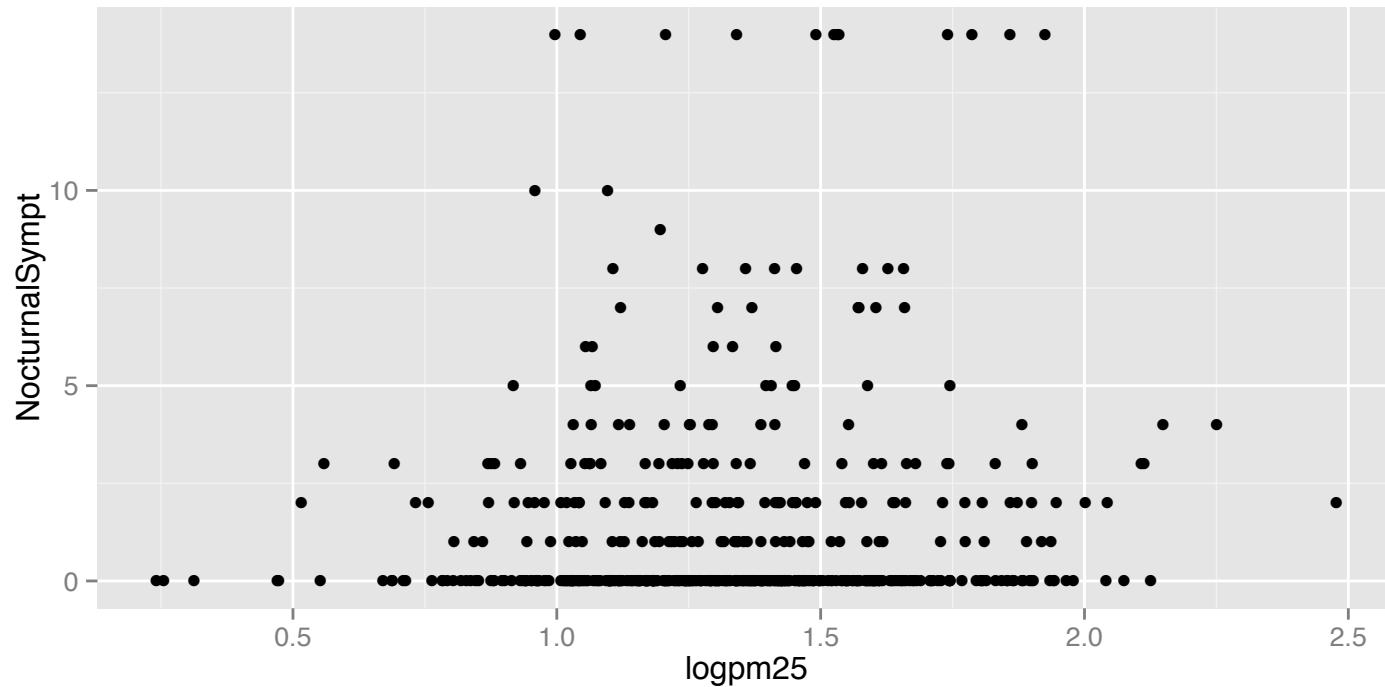
```
> p <- g + geom_point()  
> print(p)
```

Explicitly save and print
ggplot object

```
> g + geom_point()
```

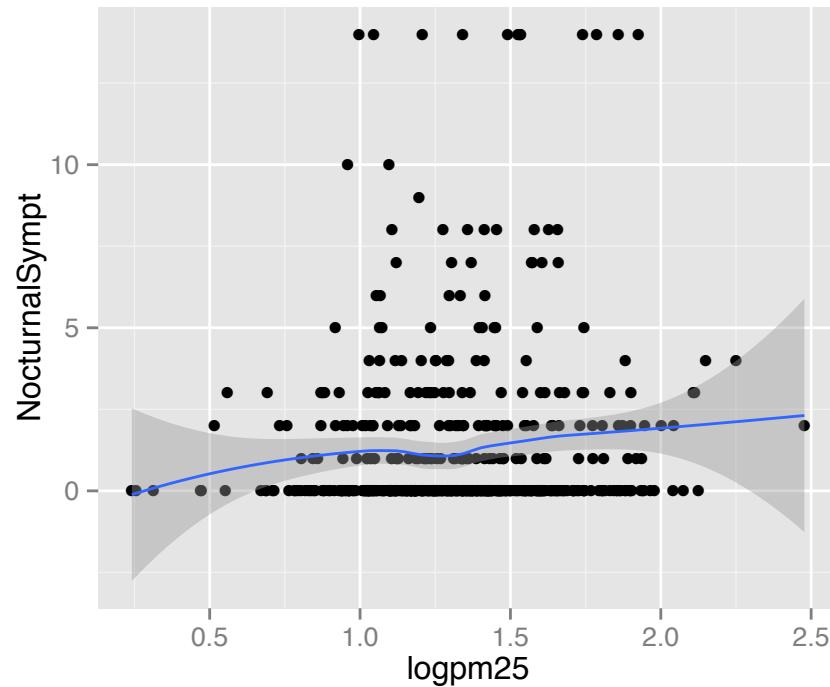
Auto-print plot object
without saving

First Plot with Point Layer

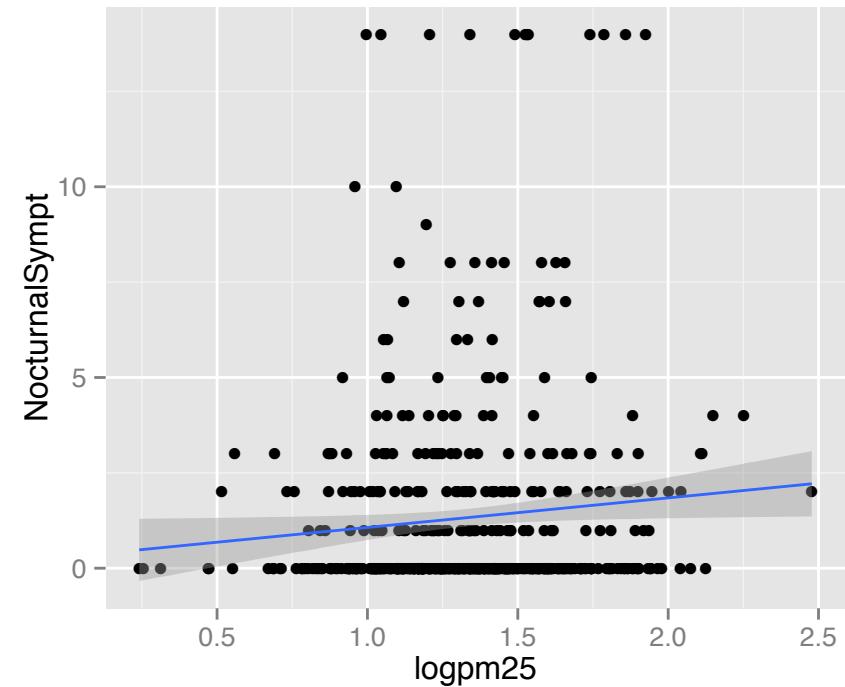


```
g <- ggplot(maacs, aes(logpm25, NocturnalSympt))  
g + geom_point()
```

Adding More Layers: Smooth

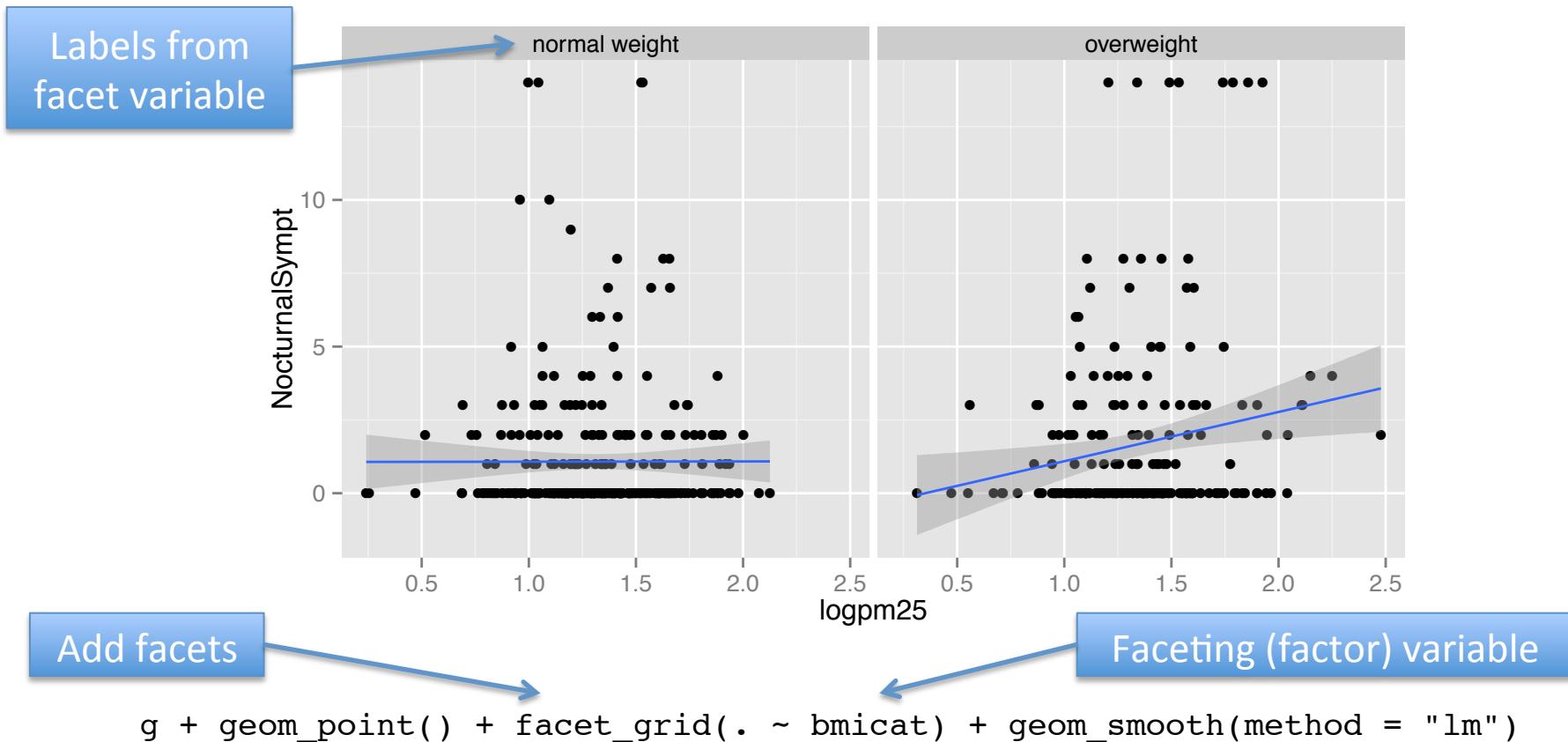


```
g + geom_point() + geom_smooth()
```



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

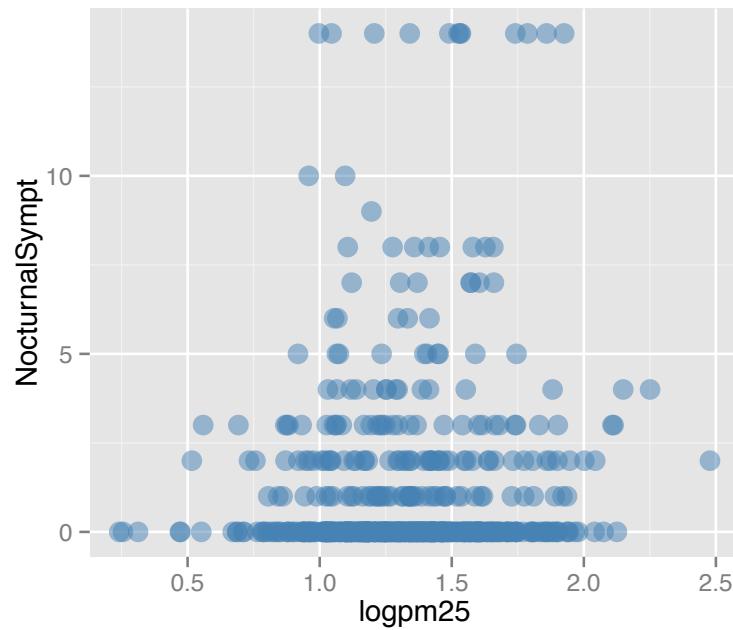
Adding More Layers: Facets



Annotation

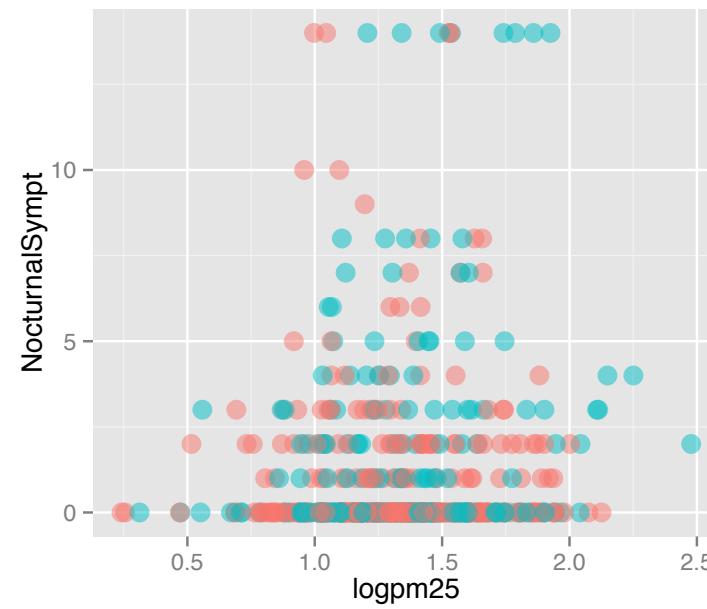
- Labels: xlab(), ylab(), labs(), ggtitle()
- Each of the “geom” functions has options to modify
- For things that only make sense globally, use theme()
 - Example: theme(legend.position = "none")
- Two standard appearance themes are included
 - theme_gray(): The default theme (gray background)
 - theme_bw():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

Modifying Aesthetics



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

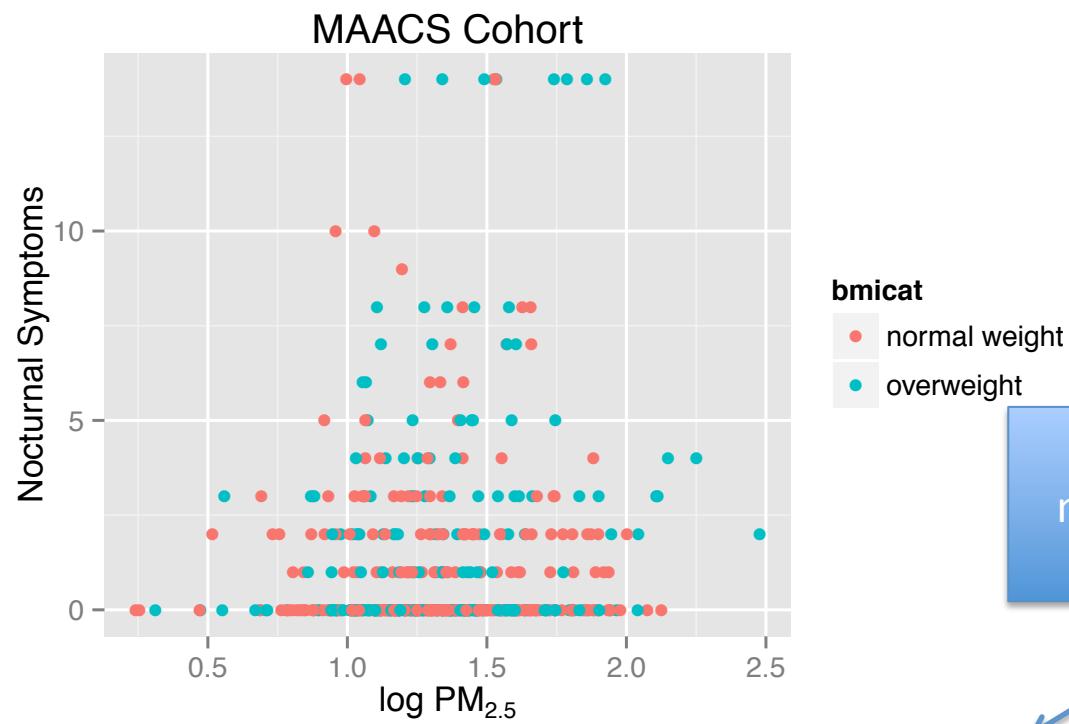
Constant values



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

Data variable

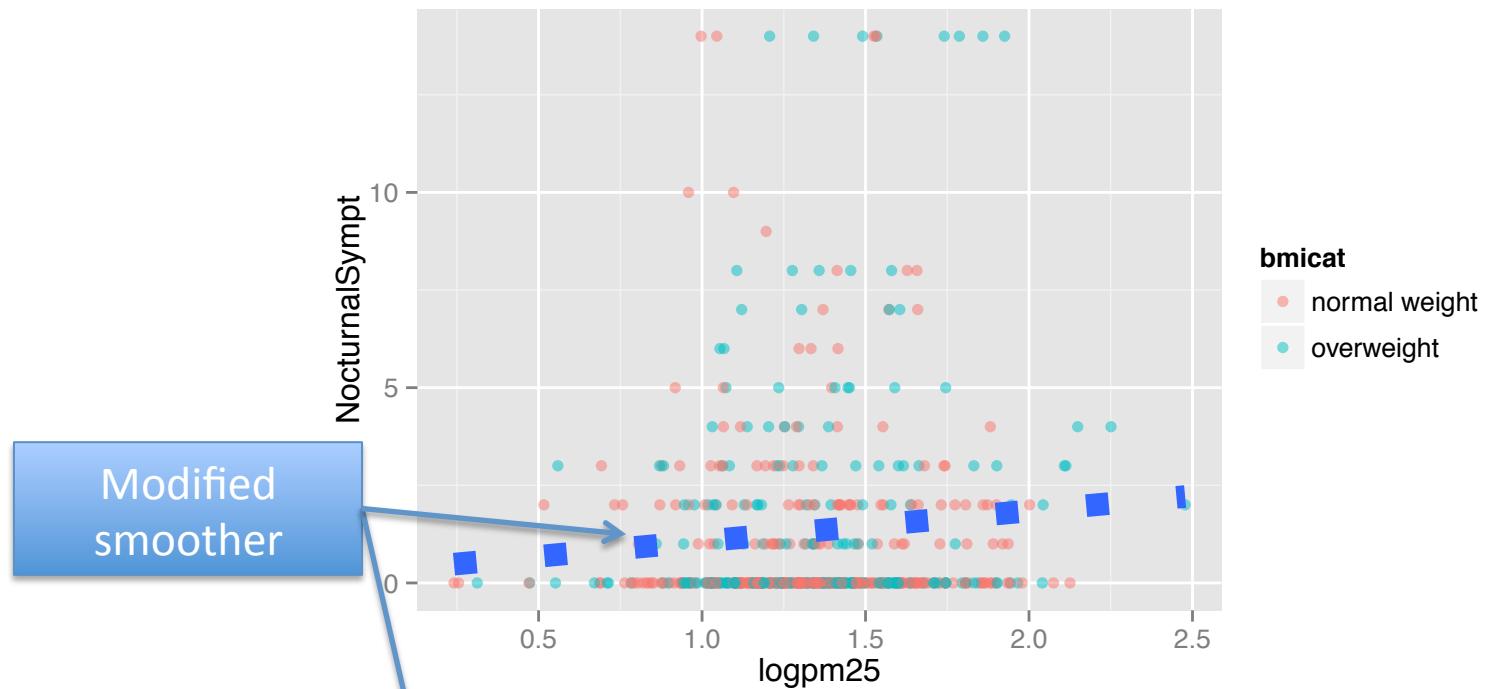
Modifying Labels



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

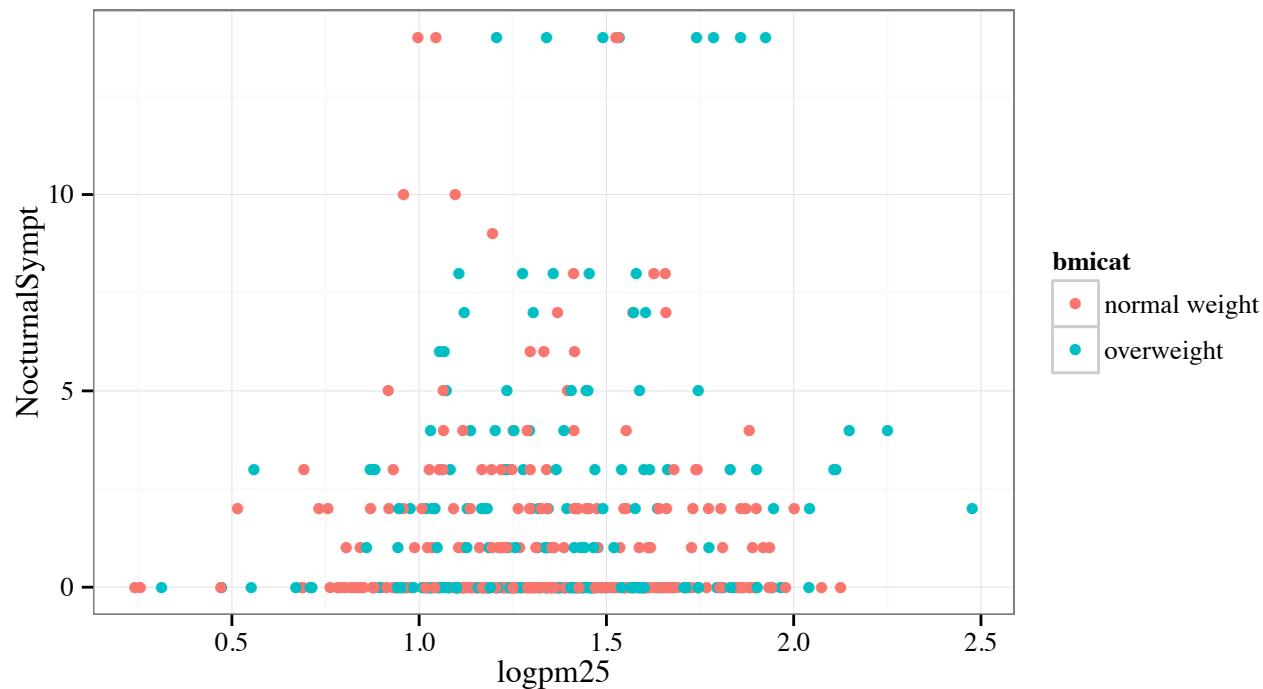
labs() function for
modifying titles and
x-, y-axis labels

Customizing the Smooth



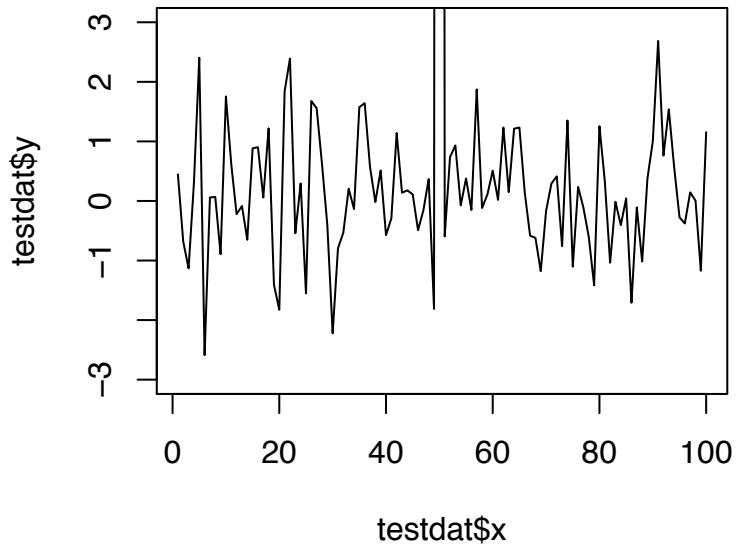
```
g + geom_point(aes(color = bmicat), size = 2, alpha = 1/2) +  
  geom_smooth(size = 4, linetype = 3, method = "lm", se = FALSE)
```

Changing the Theme

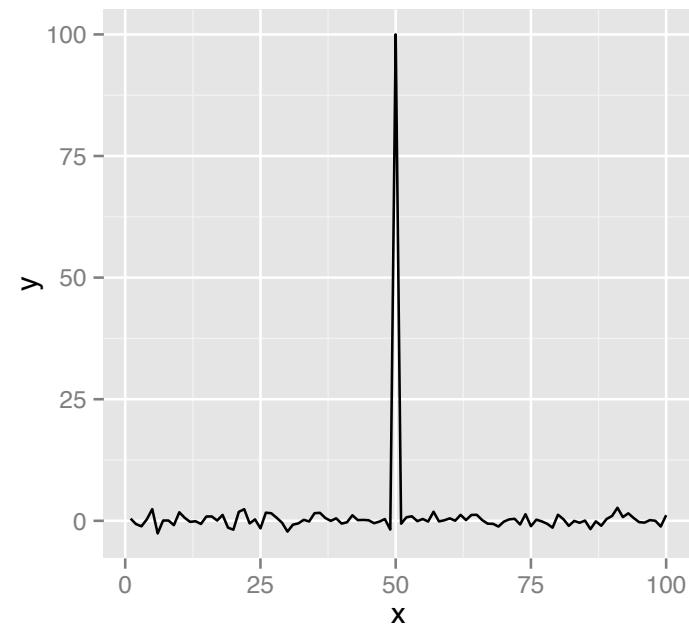


```
g + geom_point(aes(color = bmicat)) + theme_bw(base_family = "Times")
```

A Notes about Axis Limits

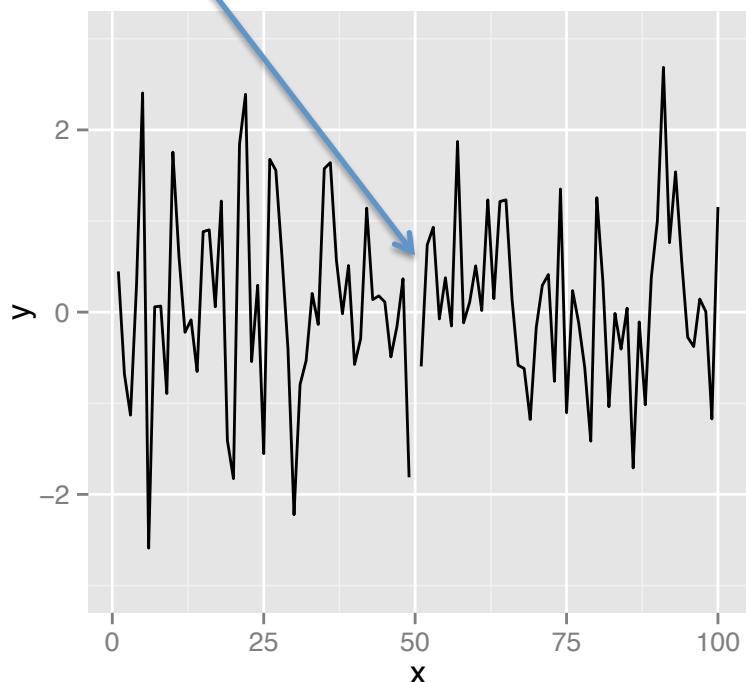


```
testdat <- data.frame(x = 1:100, y = rnorm(100))
testdat[50,2] <- 100 ## Outlier!
plot(testdat$x, testdat$y, type = "l", ylim = c(-3,3))
```



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

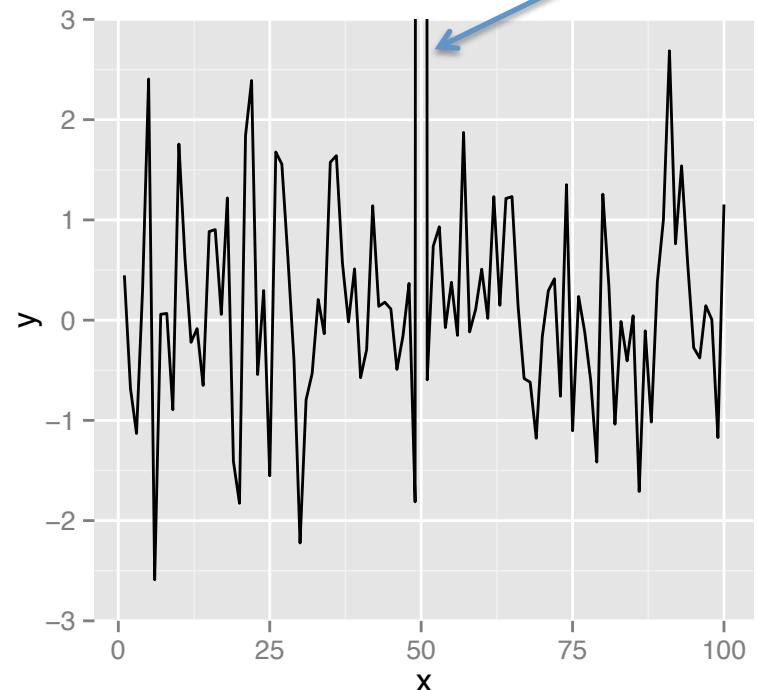
Outlier missing



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

Axis Limits

Outlier included



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

More Complex Example

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

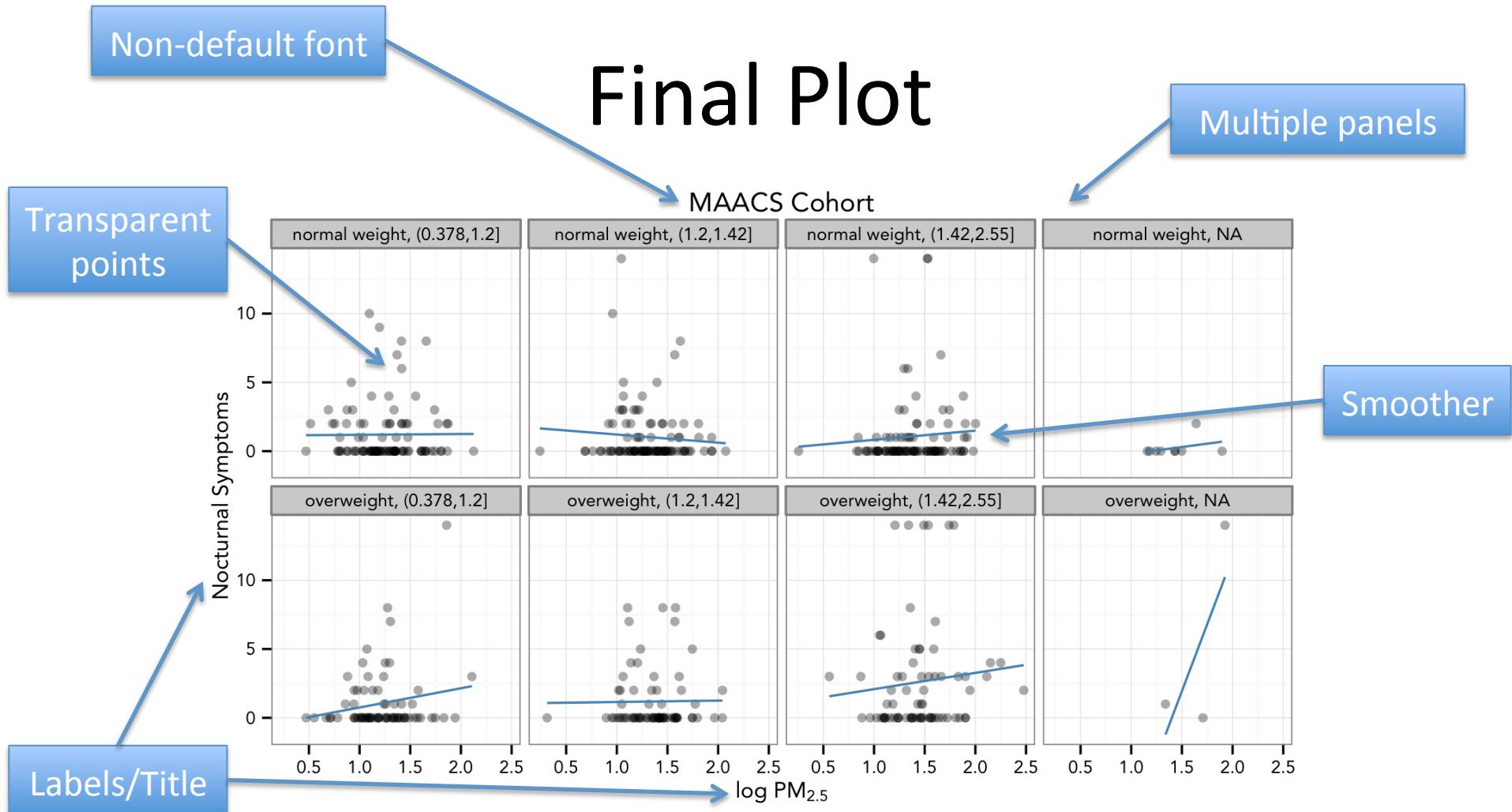
Making NO₂ Deciles

```
## 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



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")
```

The diagram illustrates the steps involved in creating the final plot. It shows arrows pointing from specific lines of R code to blue callout boxes containing descriptive text:

- An arrow points from the first line of the code (`geom_point(alpha = 1/3)`) to the box labeled "Add points".
- An arrow points from the `facet_wrap` line to the box labeled "Make panels".
- An arrow points from the `geom_smooth` line to the box labeled "Add smoother".
- An arrow points from the `theme_bw` line to the box labeled "Change theme".
- Three arrows point from the three `labs` lines to the box labeled "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)

Example: PAIRED T-TEST

It is thought that the glycaemic index (GI) of food is an indicator of how sustaining or satisfying it is and may influence appetite.

A sample of children were provided with a breakfast of low GI foods on one day and high GI foods on another. The two breakfasts contained the same quantities of carbohydrate, fat and protein. On each day a buffet lunch was provided and the number of calories eaten at lunchtime were recorded.

On the first day the children ate a low GI breakfast and on the second day a high GI breakfast. The objective is to determine whether the kind of breakfast eaten has an effect on mean calorie intake.

The table below summarises the data for children in the sample. This shows that for the children in the experiment, mean calorie intake at lunchtime was higher after the high GI breakfast. A hypothesis test is needed to determine whether these results mean that we can predict that there would be differences in mean calorie intake for other children who ate low and high GI breakfasts.

Paired Samples Statistics

	Mean	N	Std. Deviation	Std. Error Mean
kcal after low gi breakfast	607.9697	33	182.82445	31.82565
kcal after high gi breakfast	771.6061	33	222.10779	38.66400

The hypotheses to test are:

Null hypothesis (H_0): Mean calorie intake is the same following low or high GI breakfast

Alternative: (H_1): Mean calorie intake depends on whether a low or high GI breakfast was eaten

A paired t-test is appropriate because:

- (1) two related samples are being compared. The samples are related because the same group of children tested both breakfasts
- (2) the dependent variable, that is, the calorie intake, is a continuous variable

The next table shows the result of the paired t-test – this allows us to draw conclusions about the population of children from which the sample was drawn. The key outputs are the test statistic, the degrees of freedom and the P-value.

Paired Samples Test											
	Paired Differences					t	df	Sig. (2-tailed)			
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference							
				Lower	Upper						
Pair 1	kcal after low gi breakfast - kcal after high gi breakfast	-163.636	186.40762	32.44940	-229.734	-97.53910	-5.043	.000			

SPSS calculates the P-value from the test statistic and the degrees of freedom and reports it to 3 decimal places. The actual value of P is 0.0000175. This is so small that when reported to three decimal places, it is shown as 0.000 which we report as $P < 0.001$. Because the P-value is less than 0.05, we have statistically significant evidence that mean calorie intake at lunchtime depends on whether a child's breakfast had a low or high GI.

Although our conclusion is based wholly on the P-value, the convention is to report the value of t and the df as well.

The hypothesis test allows you to say:

The kind of food eaten at breakfast has a statistically significant effect on mean calorie consumption ($t = -5.043$, $df = 32$, $P < 0.001$).

The confidence interval allows you to say:

We are 95% confident that eating a low GI breakfast reduces children's mean calorie consumption at lunchtime by between 98 and 230 kcal (to the nearest whole number).

Assumptions underlying the test

The paired t-test is based on the assumption that if you calculate the difference in calorie intake following low and high GI breakfasts for each child, the results are Normally distributed. If there are major problems then the results of the test can be unreliable. This can be checked by plotting the differences between intakes following low and high GI breakfasts on a P-P plot. This is a graph that is specially designed to detect departures from the Normal distribution (which appear as deviations from the 45 degree line). If the data does not support the Normality assumption, then the problem can be solved either by using an alternative test, such as the Wilcoxon test or by transforming the data.

Standard Normal Probabilities

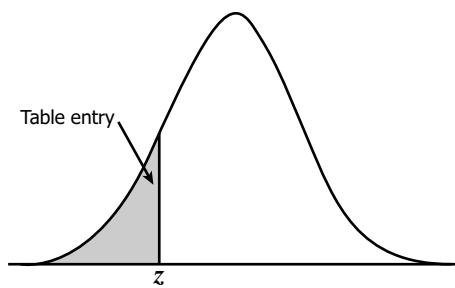


Table entry for z is the area under the standard normal curve to the left of z .

z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
-3.4	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0003	.0002
-3.3	.0005	.0005	.0005	.0004	.0004	.0004	.0004	.0004	.0004	.0003
-3.2	.0007	.0007	.0006	.0006	.0006	.0006	.0006	.0005	.0005	.0005
-3.1	.0010	.0009	.0009	.0009	.0008	.0008	.0008	.0008	.0007	.0007
-3.0	.0013	.0013	.0013	.0012	.0012	.0011	.0011	.0011	.0010	.0010
-2.9	.0019	.0018	.0018	.0017	.0016	.0016	.0015	.0015	.0014	.0014
-2.8	.0026	.0025	.0024	.0023	.0023	.0022	.0021	.0021	.0020	.0019
-2.7	.0035	.0034	.0033	.0032	.0031	.0030	.0029	.0028	.0027	.0026
-2.6	.0047	.0045	.0044	.0043	.0041	.0040	.0039	.0038	.0037	.0036
-2.5	.0062	.0060	.0059	.0057	.0055	.0054	.0052	.0051	.0049	.0048
-2.4	.0082	.0080	.0078	.0075	.0073	.0071	.0069	.0068	.0066	.0064
-2.3	.0107	.0104	.0102	.0099	.0096	.0094	.0091	.0089	.0087	.0084
-2.2	.0139	.0136	.0132	.0129	.0125	.0122	.0119	.0116	.0113	.0110
-2.1	.0179	.0174	.0170	.0166	.0162	.0158	.0154	.0150	.0146	.0143
-2.0	.0228	.0222	.0217	.0212	.0207	.0202	.0197	.0192	.0188	.0183
-1.9	.0287	.0281	.0274	.0268	.0262	.0256	.0250	.0244	.0239	.0233
-1.8	.0359	.0351	.0344	.0336	.0329	.0322	.0314	.0307	.0301	.0294
-1.7	.0446	.0436	.0427	.0418	.0409	.0401	.0392	.0384	.0375	.0367
-1.6	.0548	.0537	.0526	.0516	.0505	.0495	.0485	.0475	.0465	.0455
-1.5	.0668	.0655	.0643	.0630	.0618	.0606	.0594	.0582	.0571	.0559
-1.4	.0808	.0793	.0778	.0764	.0749	.0735	.0721	.0708	.0694	.0681
-1.3	.0968	.0951	.0934	.0918	.0901	.0885	.0869	.0853	.0838	.0823
-1.2	.1151	.1131	.1112	.1093	.1075	.1056	.1038	.1020	.1003	.0985
-1.1	.1357	.1335	.1314	.1292	.1271	.1251	.1230	.1210	.1190	.1170
-1.0	.1587	.1562	.1539	.1515	.1492	.1469	.1446	.1423	.1401	.1379
-0.9	.1841	.1814	.1788	.1762	.1736	.1711	.1685	.1660	.1635	.1611
-0.8	.2119	.2090	.2061	.2033	.2005	.1977	.1949	.1922	.1894	.1867
-0.7	.2420	.2389	.2358	.2327	.2296	.2266	.2236	.2206	.2177	.2148
-0.6	.2743	.2709	.2676	.2643	.2611	.2578	.2546	.2514	.2483	.2451
-0.5	.3085	.3050	.3015	.2981	.2946	.2912	.2877	.2843	.2810	.2776
-0.4	.3446	.3409	.3372	.3336	.3300	.3264	.3228	.3192	.3156	.3121
-0.3	.3821	.3783	.3745	.3707	.3669	.3632	.3594	.3557	.3520	.3483
-0.2	.4207	.4168	.4129	.4090	.4052	.4013	.3974	.3936	.3897	.3859
-0.1	.4602	.4562	.4522	.4483	.4443	.4404	.4364	.4325	.4286	.4247
-0.0	.5000	.4960	.4920	.4880	.4840	.4801	.4761	.4721	.4681	.4641

Standard Normal Probabilities

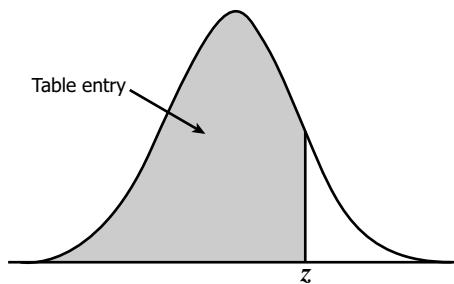


Table entry for z is the area under the standard normal curve to the left of z .