ggplot Cheat Sheets:

Amazing/Best: <http://zevross.com/blog/2014/08/04/beautiful-plotting-in-r-a-ggplot2-cheatsheet-3/>

List/Comparison of color options: <https://learnr.wordpress.com/2009/04/15/ggplot2-qualitative-colour-palettes/>

The Official ggplot2 cheat sheet from rStudio:

file:///Users/bnorgeot/Downloads/ggplot2-cheatsheet%20(2).pdf

Stats for Genomics::Distance and Clustering:

<https://www.youtube.com/watch?v=wQhVWUcXM0A>

Princliples of Analytic Graphics

1. Show comparisons
   1. Evidence for a Hypo is always relative to another competing Hypo, like the Null
   2. Always be asking/answering, “compared to what?”
   3. If kids w/asthma given air filter in house, don’t just show the number of symptom free days w/ air cleaner data, show it in comparison the control group data w/o air cleaner
2. Show causal framework, mechanism, explanation, system structure for thinking about a/the question
   1. Show the outcome and data from the study that could possible explain the outcome
   2. If removing particles from air is what might cause an increase in symptom free days, so a side by side comparison of amount of particulants for the air cleaner group vs the control
3. Show multivariate data (more than 2 variables at once). Show as much data on a sinlge plot as you can
   1. The Real World is multivariate
4. Integration of Evidence. Integrate as many different modes of describing, show, explaining evidence as you can into one plot.
   1. Words, numbers, images, diagrams
   2. Data graphics should be make use of many different modes of data presentation
5. Document the evidence w/labels, scales, sources(of the data), etc
   1. **A graphic should tell a complete story that is credible**
6. Content is king
   1. Start from the ground up. What is the story that I want to tell with the data? what’s the best way to present the evidence etc, so that others can quickly understand it?

Summary Plotting to explore data:

1-Dimensional Summaries

* histograms
* boxplot
* barplot

hist(dataframe$variable, col=”colorYoudlike”,breaks=Number/WidthOfBars)

rug(dataframe$variable) #puts tallys beneath the histogram of the counts

#overlay a line on a plot

boxplot(dataframe$variable,col=”blue)

abline(h=12) # overlay a horizontal line on the boxplot at y=12

barplot(table(dataframe$FACTOR/categoricalVariable),col=”poo”,main=”Title for plot”)

2-Dimensional Summaries

* multiple, overlayed, 1-D plots
* scatterplots (to make a scatterplot a line plot, plot(type=”l”)
* smooth scatterplot

#Multiple boxplots that compare categorical/factor/independent variables(x values) to 1 dependent/outcome variable (the y value)

boxplot(y ~ x, data = yourDataFrame, col=”something”)# compare y as function of x.

#Multiple Histograms

par(mfrow = c(2, 1), mar = c(4, 4, 2, 1)) #DON’T KNOW WHAT DOES

hist(subset(dataFrame, YourFactorVariableColumnName==”factor1”)$theDFcolumnYouWantToPlotOnY-axis), col=””)

hist(subset(dataFrame, YourFactorVariableColumnName==”factor2”)$theDFcolumnYouWantToPlotOnY-axis), col=””)

#scatterplot w/multiple factors

with(dataFrame, plot(nameOfColumnYouWantOnX,nameOfColumnYouWantOnY,col =nameOfFactorVariable-aThirdColumn-soYouCanViewXvsYbyGroup)

#Scatterplot, w/lines, each line representing a different variable/column in a different color

with(days2, {

plot(Sub\_metering\_1 ~ posDT, type = "l",

ylab = "Global Active Power (kilowatts)", xlab = "")

lines(Sub\_metering\_2 ~ posDT, col = 'Red')

lines(Sub\_metering\_3 ~ posDT, col = 'Blue')

})

Sub\_m’s are different variables, each in their own columns and are plotted as a function of posDT, which is a pasted date\_time ins POSIX.ct

#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"))

//region is name of FactorVar, pollution is name df, lattitude on x, pm25 on y

#Add x/y labels

xlab = "Month", ylab = "Ozone (ppb)"

#Add a Legend in the Top Right corner of the plot

legend("topright", pch = 1, col = c("blue", "red"), legend = c("May", "Other Months"))

#if using lines instead of dots, you need to specify line type and width in the legend

legend("topright", col = c("black", "red", "blue"), **lty = 1, lwd = 2**,legend = c("Sub\_metering\_1", "Sub\_metering\_2","Sub\_metering\_3"))

Basic plotting parameters:

pch: the plotting symbol (default is open circle)

lty: the line type (default is solid line), can be dashed, dotted, etc.

lwd: the line width, specified as an integer multiple

col: the plotting color, specified as a number, string, or hex code; the colors() function gives you a vector of colors by name

xlab: character string for the x-axis label

ylab: character string for the y-axis label

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)

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.

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

#Give special color to only certain elements: here, plot all Ozone and Wind data, but color the points (data) from the month of May a different color than all of the other data points.

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"))

#Add a Regression Line to the plot above

model <- lm(Ozone ~ Wind, airquality)

abline(model, lwd = 2)

**Multiple Base Plots**

par(mfrow = c(1, 2)) #make a plot w/1 row and 2 columns, ie 2 plots side by side

with(airquality, {

plot(Wind, Ozone, main = "Ozone and Wind") #wind as function of ozone

plot(Solar.R, Ozone, main = "Ozone and Solar Radiation") radiation as function of ozone

})

par(mfrow = c(1, 3), mar = c(4, 4, 2, 1), oma = c(0, 0, 2, 0)) #oma, outer margin must be set to #allow room for a title that will be for the whole panel

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) #Title for whole panel

})

#These two elements, (combined w/a plot) produced a ‘good’ sized legend where I wanted it.

par(mfrow = c(2,2))

legend("topright",

c("Sub\_metering\_1", "Sub\_metering\_2", "Sub\_metering\_3"),

col = c("black", "red", "blue"),

lty = 1,

box.lty = 0)

GGPLOT2

FOR A COMPLETE WALKTHROUGH, SEE MY ggplot2.tutorialComplete.R script

Great collection of resources here:

<http://stackoverflow.com/questions/3446495/ggplot2-cheat-sheet>

Other 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/> )

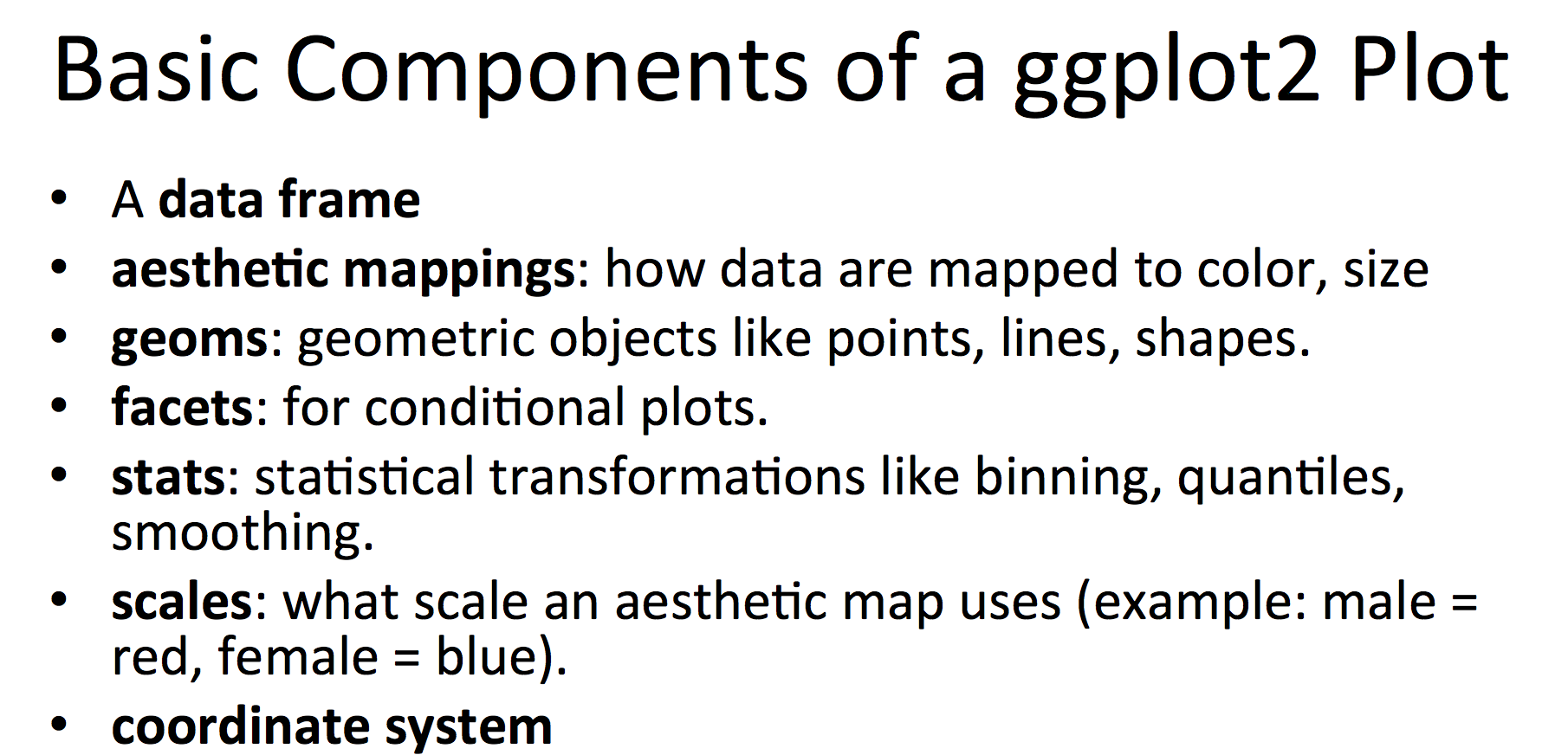
4 elements: Aesthetics, geometrics, statistics, coordinate system

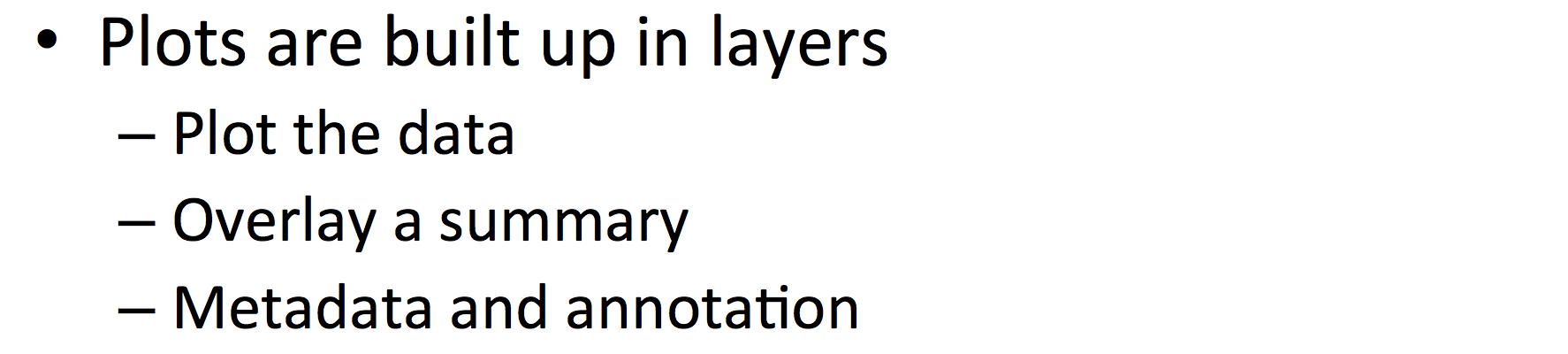
Grammar of Graphics

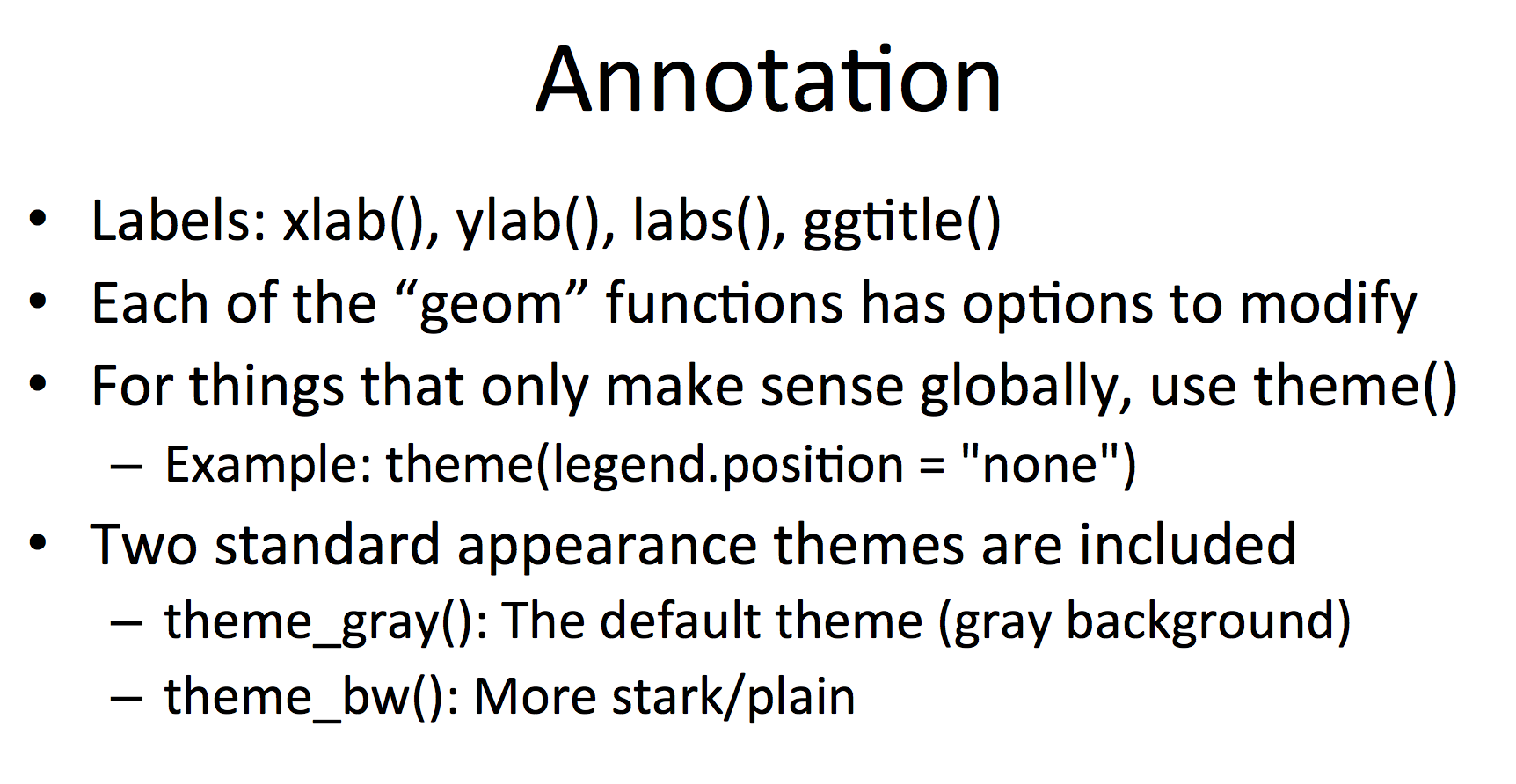
“In brief, the grammar tells us that a statistical graphic is a mapping from data to aesthetic

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”







Plots are made up of *aesthetics* (size,shape,color) and *geoms* (points, lines, bars)

Factors are important for indicating subsets of the data; they should be labeled (not just numbered)

A statistic is a summary function of the data

QPLOT()

Always must tell the function where the data is coming from and it must come from a data.frame.

Smoothing:

Understanding the difference between parametric and non-parametric fitting: <http://www.statisticshowto.com/lowess-smoothing/>

LOWESS, and least squares fitting in general, are non-parametric strategies for fitting a smooth curve to data points. “Parametric” means that the researcher or analyst assumes in advance that the data fits some type of distribution (i.e. the normal distribution). Because some type of distribution is assumed in advance, parametric fitting can lead to fitting a smooth curve that misrepresents the data. In those cases, non-parametric smoothers may be a better choice. Non-parametric smoothers like LOESS try to find a curve of best fit without assuming the data must fit some distribution shape. In general, both types of smoothers are used for the same set of data to offset the advantages and disadvantages of each type of smoother.

Benefits of Non-Parametric Smoothing

Provides a flexible approach to representing data.

Ease of use.

Computations are relatively easy.

Disadvantages of Non-Parametric Smoothing

Can’t be used to obtain a simple equation for a set of data.

Less well understood than parametric smoothers.

Requires the analyst to use a little guesswork to obtain a result.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_CLUSTERING\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DEALING W/HIGH DIMENSIONAL DATA (many variables)**

Clustering organizes things that are close together into groups.

Great overview of clustering: <https://www.youtube.com/watch?v=wQhVWUcXM0A>

I have a detailed overview of the above video (classifying distance and grouping and methods at the bottom)

How do we define close?

How do we group things?

How do we visualize the grouping?

How do we interpret the grouping?

**HIERARCHICAL CLUSTERING**

Two approaches Top-down(Divisive) and Bottom-Up(agglomerative). Distance from one point to another is determined by vertical height from the node of one point to the node where the branches of the 2 points intersect. Creating cluster is done based on heights. Draw a horiztonal line somewhere on the dendogram, the stems that are severed create the groups/clusters.

Top-down: Divide all points into 2 groups. Then divide each group into 2 groups. Keep going until you have groups of 1 and can’t divide further

An agglomerative approach (bottom up approach starting w/the individual data points and start lumping them together until all of your data is 1 big cluster)

Find closest two things

Put them together: into a new (imaginary) ‘merged point’. Then remove the original 2 points. The location of the new merged super point can be done in one of two ways. Merging methods

* Average Linkage: new point x-coordinate is the just the average of the 2 original points x-coordinates, and the new y-coordinate is avg of og points y-coordiantes. For 2 clusters, the new super-cluster x is the the avg of all x points, the y is the avg of all y values
* Complete Linkage: Assuming you have 2 clusters of points, the distance between the 2 points (1 from each cluster) that are the furthest away from each other.The new point for the new cluster is just the avg of the x-values for these 2 points (not all the points in the clusters)

Find next 2 closest things.

Requires

A defined distance or similarity metric. Pick the type that makes sense for your problem. 3 commons types

Continuous - euclidean distance

Continuous - correlation similarity

Binary - manhattan distance. Can’t always travel in straight line, so euclidean dist not always meaningful. Manhattan distance, can’t walk through obstacles, like buildings, must walk around them, like a grid where you must walk along the lines and can’t pass through the center of a square, like blocks.To find, calculate the absolute value of the distance that you travel in each dimension.(eg: abs(x-distance traveled) + abs(y--distance traveled)

A merging approach: how do you merge 2 things

Produces

A tree (dendrogram) showing how close things are to each other. the points that are further down the tree, towards the bottom, are closer and got clustered/merged first. those that are higher up are further from each other and got merged/clustered last.

Heir-clust Example

set.seed(1234)

par(mar = c(0, 0, 0, 0))

#create x and y coordinates to make 12 points

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 the points

plot(x, y, col = "blue", pch = 19, cex = 2)

#label each of the points using text()

text(x + 0.05, y + 0.05, labels = as.character(1:12))

#calculate the distance between the points using dist() which creates a distance matrix giving all of the pairwise distances. Uses Euclidean distance by default, but other distance options are possible.

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

distxy <- dist(dataFrame)

#cluster using hclust(). Note that this doesn’t tell or show how **many** clusters there are

hClustering <- hclust(distxy)

plot(hClustering)

# to cut the tree into clusters, imagine drawing a horizontal line along the y-axis at a given point, the number of branches on the tree that the line intersects/cuts will determine the number of groups/clusters that are created. You choose how to set where this line falls.

A Heat Map, heatmap(), runs a heirarchical clustering on the rows of the table and on the columns of the table. Re-orders the columns and the rows according the the heirarchichal clustering.

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

set.seed(143)

dataMatrix <- as.matrix(dataFrame)[sample(1:12), ]

heatmap(dataMatrix)

**K-means CLUSTERING**

A partioning approach

Determine a number of clusters (before starting). How do you choose how many clusters?<http://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set>

Get "centroids" of each cluster

Assign observations/elements to closest centroid

Reclaculate centroids: often by taking the mean (x,y) for the points that were just assigned to the cluster/centroid (Then you can reassign the observations.) There can be this iteration back and forth between calculating centroids and assigning observations until a ‘good’ solution is reached.

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

kmeans(): Important parameters: x, centers (number of centroids), iter.max, nstart

this clustering approach can be sensitive to the initial selection of centroids. The kmeans() function has an nstart option that attempts multiple initial configurations and reports on the best one. For example, adding nstart=25 will generate 25 initial configurations. This approach is often recommended. <http://www.r-statistics.com/2013/08/k-means-clustering-from-r-in-action/>

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 #For each point, shows which cluster the point has been assigned to

kmeansObj$centers #The location of the centroids.

#plot the results of the k-means clustering

par(mar = rep(0.2, 4))

plot(x, y, col = kmeansObj$cluster, pch = 19, cex = 2) # color the points depending on which cluster they’re in

points(kmeansObj$centers, col = 1:3, pch = 3, cex = 3, lwd = 3) #Add the centroids, and give them the ‘+’ symbol, pch=3.

**DIMENSION REDUCTION:**

What do you do when you have a ,overwhelming, very high number of variables? With too many variables, it can be impossible to find mappings from variables to outcomes. Principal Components Analysis and Singular Value Decomposition.

1. Statistical approach/Principal Components Analysis(PCA):Create a set of variables that is smaller than the original set of variables. Clustering or grouping variables together that might be related/correlated (eg height and weight). The new set of variables (which are each a group of variables that are correlated to each other) will be uncorrelated to eachother.
2. Data compression approach/Singular Value Decomposition(SVD): If you put all the variables together in one matrix, find the best matrix created with fewer variables (lower rank matrix) that explains the original data

DEVICES:

Copy a plot from your screen to another device (or from dev1 to dev2)

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!

**Graphics File Devices**

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

Vector formats:good for line drawings and plots with solid colors using a modest number of points, resizes well

* 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

Bitmap format: good for plots with a large number of points, natural scenes or web-based plots, don’t resize well.

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

CLUSTERING CASE STUDY: PREDICTING HUMAN ACTVITY W/SAMSUNG S2

Each phone has: 1 accelerometer, 1 gyroscope

downloaded the ‘training’ data from: <http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones>

Plotting average acceleration for first subject

par(mfrow = c(1, 2), mar = c(5, 4, 1, 1))

# Convert the ‘activity’ column into a factor variable

samsungData <- transform(samsungData, activity = factor(activity))

sub1 <- subset(samsungData, subject == 1)

# column1 is the mean body accleration in the x-direction

plot(sub1[, 1], col = sub1$activity, ylab = names(sub1)[1])

# column2 is the mean body accleration in the y-direction (col3 = z direction)

plot(sub1[, 2], col = sub1$activity, ylab = names(sub1)[2])

#Use col= to color code each activity

legend("bottomright", legend = unique(sub1$activity), col = unique(sub1$activity),

pch = 1)

#NOTE: average acceleration doesn’t seem to show much variation based on the different types of activities, this might not be a good predictor

Clustering based just on average acceleration

source("myplclust.R")

#create a distance matrix to calculate pair-wise distance between all points

distanceMatrix <- dist(sub1[, 1:3])

#make the cluster

hclustering <- hclust(distanceMatrix)

#visualize the cluster

myplclust(hclustering, lab.col = unclass(sub1$activity))

#NOTE: Sure enough, average acceleration produced a jumbled cluter

Plotting max acceleration for the first subject

par(mfrow = c(1, 2))

#column 10 is max acceleration x direction

plot(sub1[, 10], pch = 19, col = sub1$activity, ylab = names(sub1)[10])

#column 11 is max acceleration y direction

plot(sub1[, 11], pch = 19, col = sub1$activity, ylab = names(sub1)[11])

#NOTE: max acceleration has more variation, this might be more predictive of body position

Clustering based on maximum acceleration

source("myplclust.R")

distanceMatrix <- dist(sub1[, 10:12])

hclustering <- hclust(distanceMatrix)

myplclust(hclustering, lab.col = unclass(sub1$activity))

#NOTE: yay, distinct, seperate clusters. On the left hand side we see the various walking activities, on the right hand side we see the various non-moving activities

Singular Value Decomposition

# The last 2 columns are just the activity identifer and subject identifer. remove them

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

# Which feature in the whole data set is contributing the most to the overall variation? This one will/should be the most predictive.

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

New clustering with maximum contributer

maxContrib <- which.max(svd1$v[, 2])

# create new cluster w/max acceleration PLUS whichever feature contributed the most variation.

distanceMatrix <- dist(sub1[, c(10:12, maxContrib)])

hclustering <- hclust(distanceMatrix)

myplclust(hclustering, lab.col = unclass(sub1$activity))

#NOTE: the 3 movement activities have now been completely seperated from each other and the non-moving activities. Thought the non-moving activities are still all clustered together.

#The max contributor was mean body accleration, frequency domain, in the z direction:

names(samsungData)[maxContrib]

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

TRY USING K-MEANS CLUSTERING INSTEAD

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

# Since there’s 6 activities, choose 6 centers/centroids

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 #Clus1 is clearly walking

## 2 0 0 0 0 48 0 #clus2 is clearly walking down

## 3 27 37 51 0 0 0 #Notice clus3 is a combo of laying, sitting, standing

## 4 3 0 0 0 0 53 # clus4 is walking up

## 5 0 0 0 45 0 0 # clus5 is ALSO clearly walking

## 6 20 10 2 0 0 0 # clus6 is mostly laying w/some sitting

Try AGAIN

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 49 0

## 2 18 10 2 0 0 0

## 3 0 0 0 95 0 0

## 4 29 0 0 0 0 0

## 5 0 37 51 0 0 0

## 6 3 0 0 0 0 53

THIS WAS BETTER. Clus1,3,4,6 very clear now. Only 5 and 2 problematic

Try AGAIN, this time w/100 different starting values (centroid values, try 100 diff, report the one ones that do the best)

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

NOT MUCH DIFFERENT. clus2 and 5 now clear. But 6 and 1 have issues

EXAMINE the centers of the resulting clusters: What features have interesting values that drive the location of that center? this gives hint as to which features might drive prediction for a particular activity.

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 = "")

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_DISTANCE AND CLUSTERING\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

<https://www.youtube.com/watch?v=wQhVWUcXM0A>

MicroArrays(SNP Chips):

Genotyping: answering the question about the 2 copies of the chromosome on which the SNP is located.

At each of the 1-million or so sites of variation that the chip examines, there are always/only 3 possible genotypes (HH, Hh, hh). Since you only have 2 strands of dna, you can be homozygous in 2 different ways, or heterozygous.

If you’re probing for a position that contain either an A or a G, the 3 Genotypes would be AA, AG, GG

<https://www.23andme.com/en-ca/more/genotyping/>

The BeadChip is a small glass slide with millions of tiny beads on its surface. Attached to each bead are probes - bits of DNA complementary to sites in your genome where SNPs of interest are located. Your DNA will stick to the probe that matches whichever SNP you happen to have.

In the genotyping process, a person's DNA is chopped up into pieces and washed over the chip, where each fragment sticks to any probes that are complementary to it. Then fragments of DNA that have been specially tagged are introduced to the chip in such a way that they stick to any probes that are themselves paired perfectly with sample DNA. At that point, the tagged DNA fragments are triggered to glow indicating which version of each SNP is present in the sample.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_