Superficial Data Analysis – Exploring Millions of Social Stereotypes

Brendan O'Connor and Lukas Biewald
Dolores Labs, Inc. http://doloreslabs.com

Appears in "Beautiful Data", ed. Toby Segaran and Jeff Hammerbacher. 2009 O'Reilly Media.

Introduction

How do we perceive age, gender, intelligence, and attractiveness? What insight can we extract from millions of anonymous opinions?

Last year we, with Chris Van Pelt, built the website FaceStat.com, where users can upload their own photos, as well as look at and judge photos of other people. The site became surprisingly popular. More than 100,000 brave users have uploaded pictures of themselves, friends, relatives, enemies, etc., and more than 10 million judgments have been collected for pre-selected questions such as:

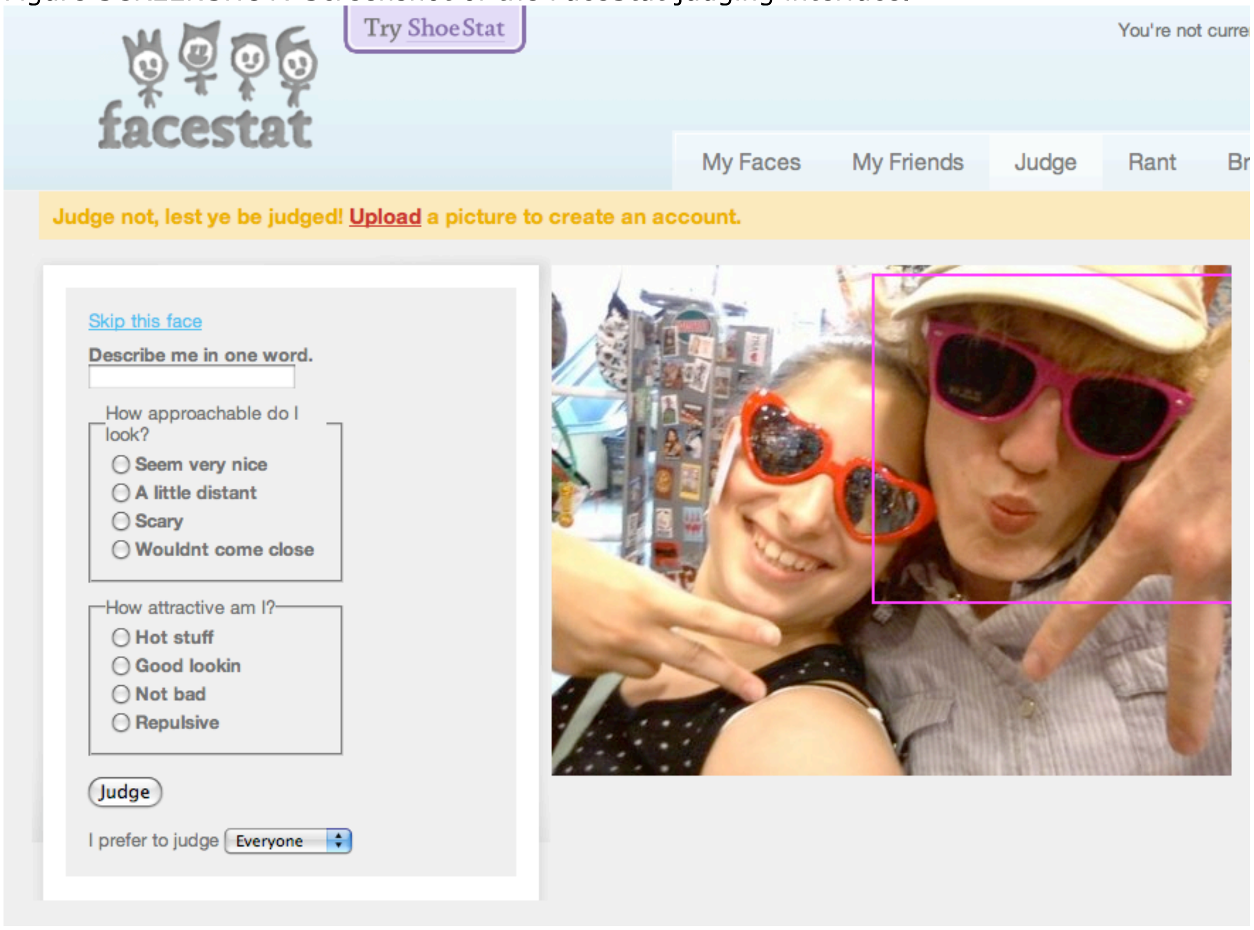
- *How old do I look?*
- *Do you think I look smart?*
- *Do you think I could win a fight with a medium sized dog?*
- *Describe me in one word*.

We like to call it "multivariate Hot-or-Not."

Researchers in psychology and sociology have extensively studied stereotypes and how our appearances influence the way we are perceived. But no one has had access to such a large pool of data from such a diverse group of people. This data is much messier than a typical lab experiment, but can volume make up for a lack of control? In fact, real-world data might be most revealing: someone who thinks they're playing a game could be more honest than a college sophomore taking a survey in their Psych 101 course.

We love exploring big datasets. Rather than confirm pre-baked hypotheses, we'll search for interesting patterns and correlations. We won't try to hide or gloss over the messy outliers and missing values; instead, we'll show you explicitly the choices we're forced to make. We will refrain from drawing grand or controversial conclusions about stereotyping and let the data speak for itself.

Figure SCREENSHOT: Screenshot of the FaceStat judging interface.



I prefer to judge Everyone

Preprocessing the Data

We'll start from the beginning: like many websites, FaceStat runs on an SQL database. The judgment interface takes user judgments and saves them as a set of (face ID, attribute, judgment) triples. The first thing we do is extract those 10 million rows from the database. This gives us a file that looks like:

```
face_id    key          value
149777     describe     serious
18717      trustworthy  3
140467     attractive   2
149777     describe     five-head
...
```

We're interested in exploring the relationships between different types of perceived attributes. One interesting question is, "How old do I look?" The very first thing to do is to look at the responses that people have given. Unix command-line tools make it easy to quickly see a histogram of responses. The most common responses look like reasonable ages, but we also see a problem:

Look at age judgments'

```
$ cat data.tsv  |
  grep "age"  |
```

values	*cut -f3* |
and count how many times	*sort* |
each value occurs,	*uniq -c* |
and order by this count.	*sort -nr*

Here's the output of this shell pipeline. For each line, the first number is the frequency count. The second string is the response value -- exactly what the user typed in the web form in response to the question *How old do I look?* Most often, they typed in a number, but there are some issues:

```
70472 19
70021 22
69387 18
68423 17
...
27 24\r\n
27 17\r\n
23 01
21 16\r\n
...
1 old enough to know better
1 hopefully over 21
1 e
1 ??
...
```

FaceStat has existed for 8 months and undergone many changes, so data has been collected under different circumstances. Some weird web browsers seem to add whitespace control codes \r\n. At some point there was a bug and users slipped in textual responses and other problematic data. Looking at rare values from the bottom of the *sort | uniq -c | sort -nr* histogram is an easy way to reveal data bugs, since they often manifest as outliers. We have to write some regular expressions that can clean out bad values like this.

It would be tedious to go into detail about all of the sanity checks and data cleanup, but they are a crucial first step for any data analysis. With any human-generated data set, there's bound to be messy outliers. For example, we found one person who figured out a way to circumvent the randomness in the selection of which face to judge, and labeled one face "mr. cool" hundreds of times.

Besides cleanup, some critical decisions to make for this particular data set are (1) how to map from multiple choice responses like "very trustworthy" vs. "not to be trusted" to a numerical value, and (2) how to aggregate results from multiple people into a single description of a face. Every face has some 100 judgments among several different attributes. We'll simply average the numeric judgments. (Under this paradigm, we ignore textual judgments; we'll get to those later.) So each face has an average perceived age, perceived intelligence, etc. Using SQL and Python scripts, we eventually end up with a file with one row per face. It looks something like:

```
  male      age intelligence attractive political_affiliation
  TRUE 24.26667           NA   2.800000                    NA
  TRUE 47.00000     3.400000   2.120000                   3.2
  TRUE 29.27273     2.700000   2.083333                   1.8
 FALSE 17.63636     3.111111   2.428571                    NA
```

```
FALSE 19.58333        NA   2.750000                 NA
 TRUE 22.80953        NA   2.250000                 NA
 TRUE 29.77778  1.833333   1.900000                 NA
FALSE 18.16667        NA   2.571429                 NA
 TRUE 46.60000  3.200000   2.120000                3.4
 TRUE 52.06667  3.000000   2.080000                 NA
```

In all, there are tens of thousands of faces with about 20 different attributes. There are many missing values: different questions were asked of different people. With those caveats in mind, we're ready to load the data into a package for more detailed analysis. If you want to follow along, we've made a subset of the data and useful code available at http://data.doloreslabs.com.

Exploring the Data

There are many great tools for data analysis. Some of the most commonly used are compared in table COMP.

Table COMP: Comparison of Data Analysis Packages

Name	**Advantages**	**Disadvantages**	**Open source?**	**Typical users**
R	Library support; visualization	Steep learning curve	Yes	Statistics
Matlab	Elegant matrix support; visualization	Expensive; incomplete statistics support	No	Engineering
SciPy/NumPy/ Matplotlib	Python (general-purpose programming language)	Less mature	Yes	Engineering
Excel	Easy; visual; flexible	Large datasets; weak numeric support	No	Business
SAS	Large datasets	Very baroque; hardest to learn	No	Business
Stata (and SPSS)	Easy statistical analysis	Less programmatic than R/Matlab/Py	No	Science (bio and social)

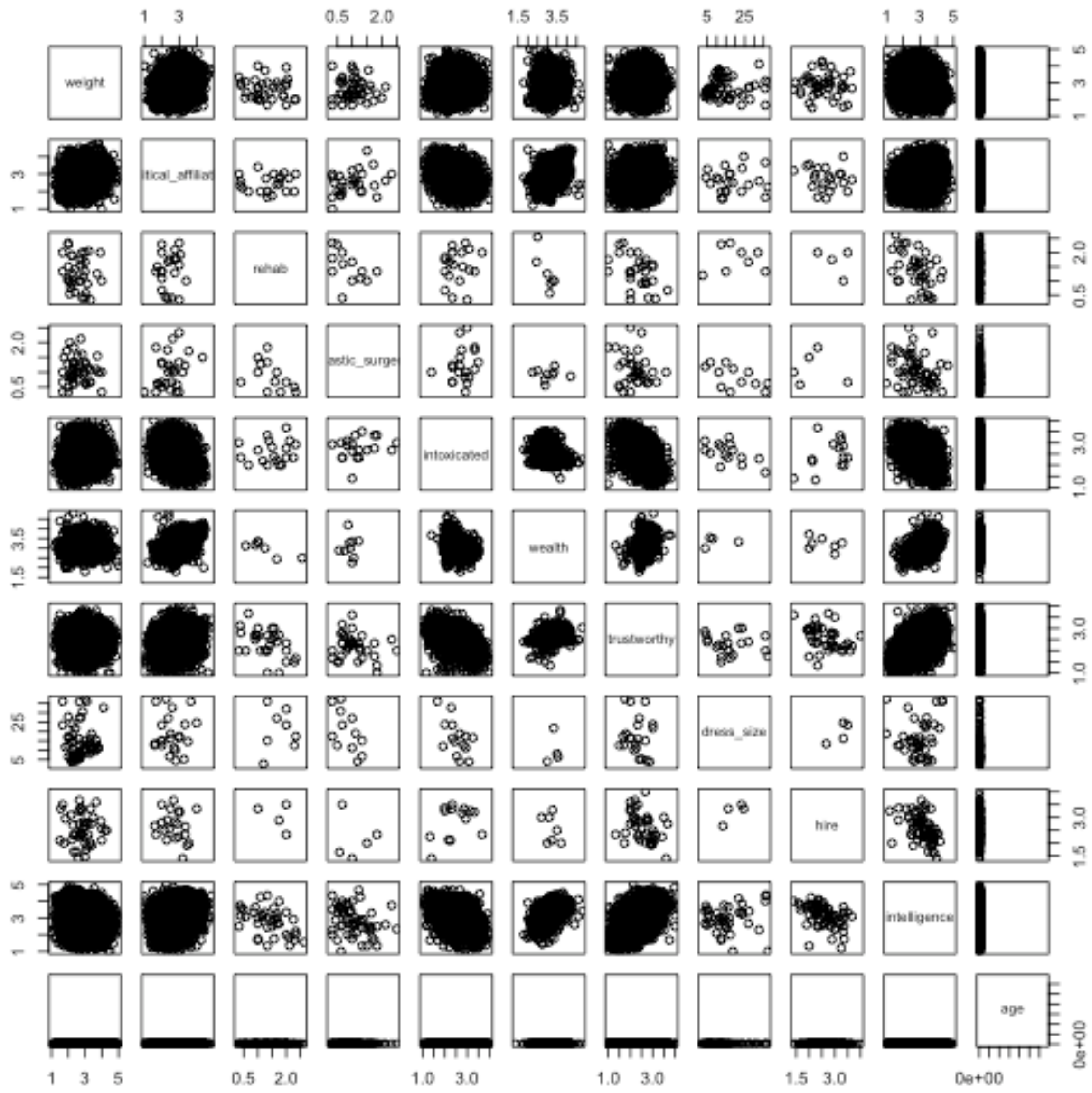
We like to use R. It's an open source statistical and visualization programming environment with a vibrant and growing development community. It's emerged as a de facto standard among statisticians. For exploratory data analysis, we prefer it to the other options because of its graphing libraries, convenient indexing notation, and an amazing array of statistically sophisticated, community maintained packages. You can read about it and download it at http://www.r-project.org; also see the references at the end of this chapter.

R provides many of excellent tools for looking at what's in the data. From its interactive interpreter:

Load the data and plot.	*> data = read.delim("http://data.doloreslabs.com/face_scores.tsv", sep="\t")* *> plot(data)*

Given a basic table of records, R's default plotting action is to give us a scatterplot matrix of every pair of variables. [Figure BADSCATS.] One thing that jumps out is that the age correlations look funny -- the right-most column and bottom-most row.

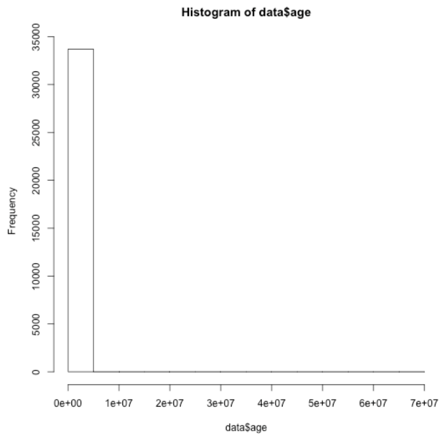
Figure BADSCATS: Initial scatterplot matrix of the face data.



We need to investigate. The first thing to do is look at the distribution of age values. [Figure BADHIST.]

> hist(data$age)

Figure BADHIST: Initial histogram of face age data.



This doesn't look right. The x-axis has been scaled all the way up to 70 million because of outliers. Let's look at the records with outlying age values.

Select records with age greater than 100.   *> data[which(data$age > 100),]*

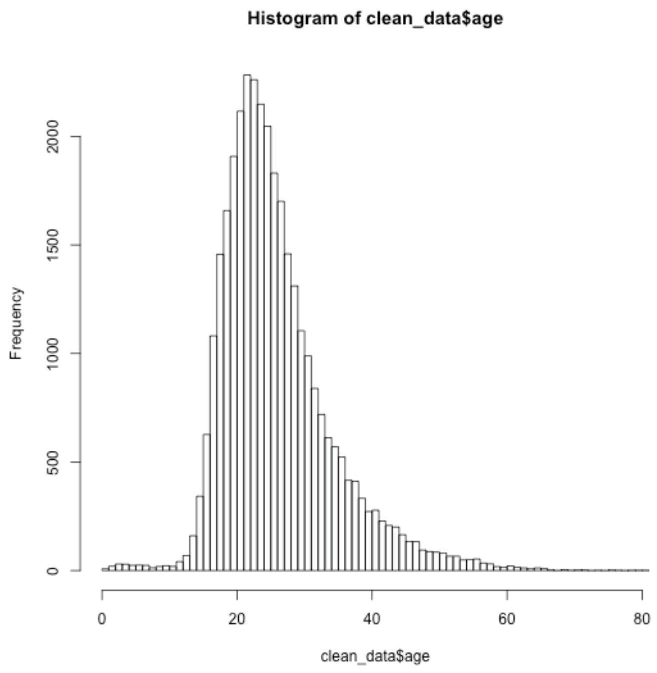
```
   id num_judgments           age male attractive intelligence
40623           150      402.3333 TRUE   2.416667           NA
57021           133    47882.3010 TRUE         NA           NA
66441           197 66666692.0000 TRUE         NA           NA
```

Earlier, we cleaned out the non-numeric age values, but we didn't check for absurdly high values. The easiest thing to do for now is to just remove these outliers. If you haven't used a data analysis language before, notice how R's rich subscripting notation makes basic exploration and cleaning easy and fun.

Subselect rows with age less than 100.   *> clean_data = data[which(data$age < 100),]*

We check the histogram again – Figure GOODHIST – and find out that most of our users are (or appear to be) between 18 and 30, which seems reasonable.

Figure GOODHIST: Histogram of cleaned face age data.



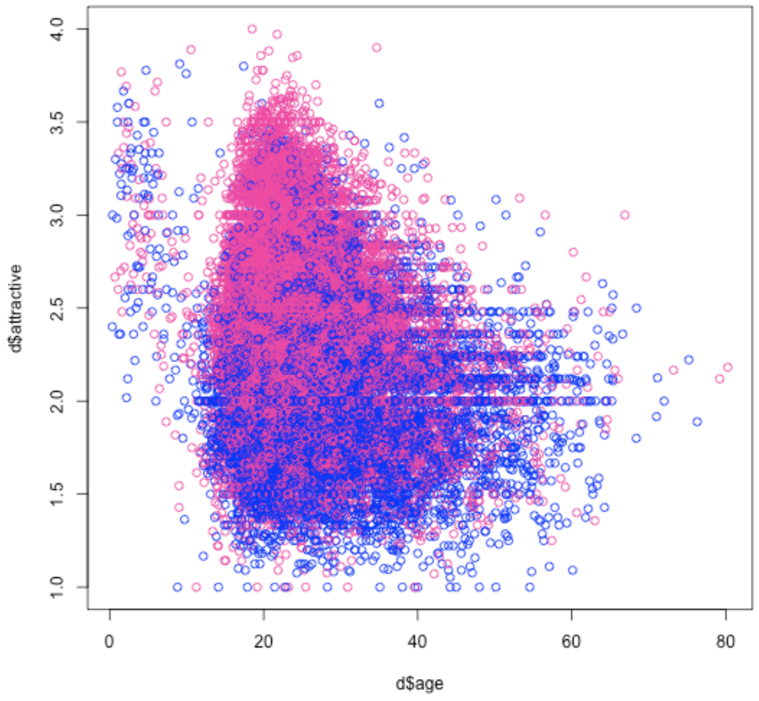
Age, attractiveness, and gender

We want to zoom in on interactions of some of the most interesting perceived attributes: age, gender, and attractiveness. Whenever we have a table with a few interesting columns, it's straightforward and often informative to throw it up as a scatterplot [Figure AAG_SP1].

Draw a scatterplot of age vs. attractiveness, using gender to define the points' colors.

```
> plot(d$age, d$attractive,
    col = ifelse(d$male, 'blue', 'deeppink'))
```

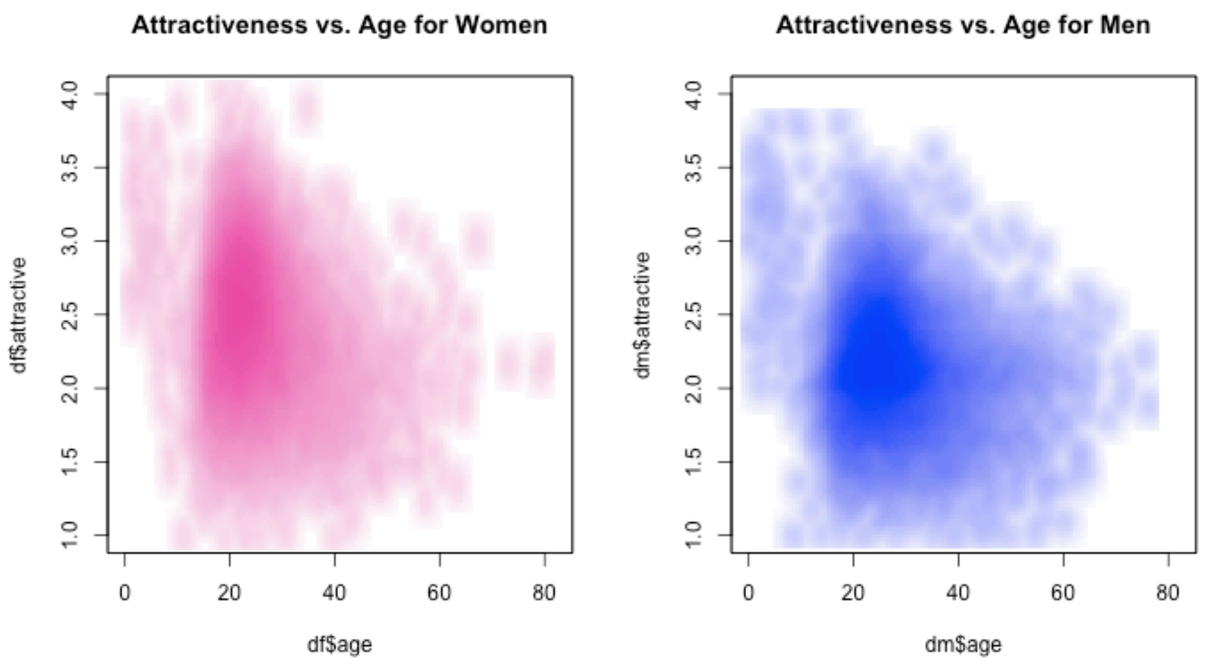
Figure AAG_SIMPLE: Scatterplot of attractiveness vs. age, colored by gender.



This plot is suggestive -- for example, women seem to be more attractive than men. But it's hard to tell anything for sure, since tens of thousands of points are being drawn over each other. When there is an overload of data, scatterplots can be misleading. One way to deal with this is to smooth the data, by plotting an estimated distribution rather than the points themselves. We use a standard technique called kernel density estimation.

Lay out side-by-side plots.	*> par(mfrow=c(1,2))*
For males and females,	*> dm = d[d$male,]; df = d[d$female,]*
draw smoothed plots, with a color gradient, and aligned axes.	*> smoothScatter(dfage, dfattractive, colramp = colorRampPalette(c("white", "deeppink")), ylim=c(0,4))*
	> smoothScatter(dmage, dmattractive, colramp = colorRampPalette(c("white", "blue")), ylim=c(0,4))

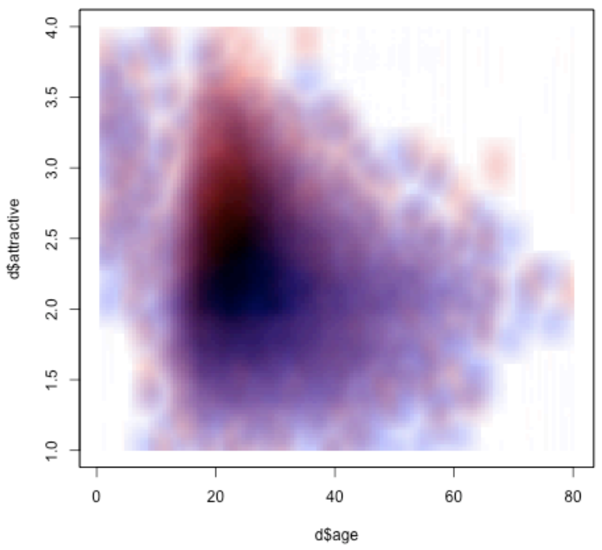
Figure SMOOTHCOMPARE: Smoothed scatterplots for attractiveness vs. age, one plot per gender.



We can even try putting them on the same plot [Figure SMOOTHCOMBINED].

```
> smoothScatterMult(d$age, d$attractive, d$male, blendFun=bl_burn, colramps = c(colorRampPalette(c("white", "red")), colorRampPalette(c("white", "blue")), colorRampPalette(c("white", "green"))), pch="", nrpoints=10000)
```

Figure SMOOTHCOMBINED: Smoothed scatterplots for attractiveness vs. age, colored by gender and overlaid on one plot.



These graphs show the full distribution of the data, but it's hard to see patterns. For example, how does age affect attractiveness? It's easier to see this by computing summary statistics and plotting them. [Figure AAG_BUCKETS_SIMPLE.]

For males	> dm = d[which(d$male),]
and females,	> df = d[which(d$female),]
average across faces	> male_avg_by_year = by(dm$attractive,
within bins	cut(dm$age, breaks=0:80), mean)
(one per year)	> female_avg_by_year = by(df$attractive,
then	cut(df$age, breaks=0:80), mean)
plot them	> plot(male_avg_by_year, col='blue')
all together.	> points(female_avg_by_year, col='deeppink')

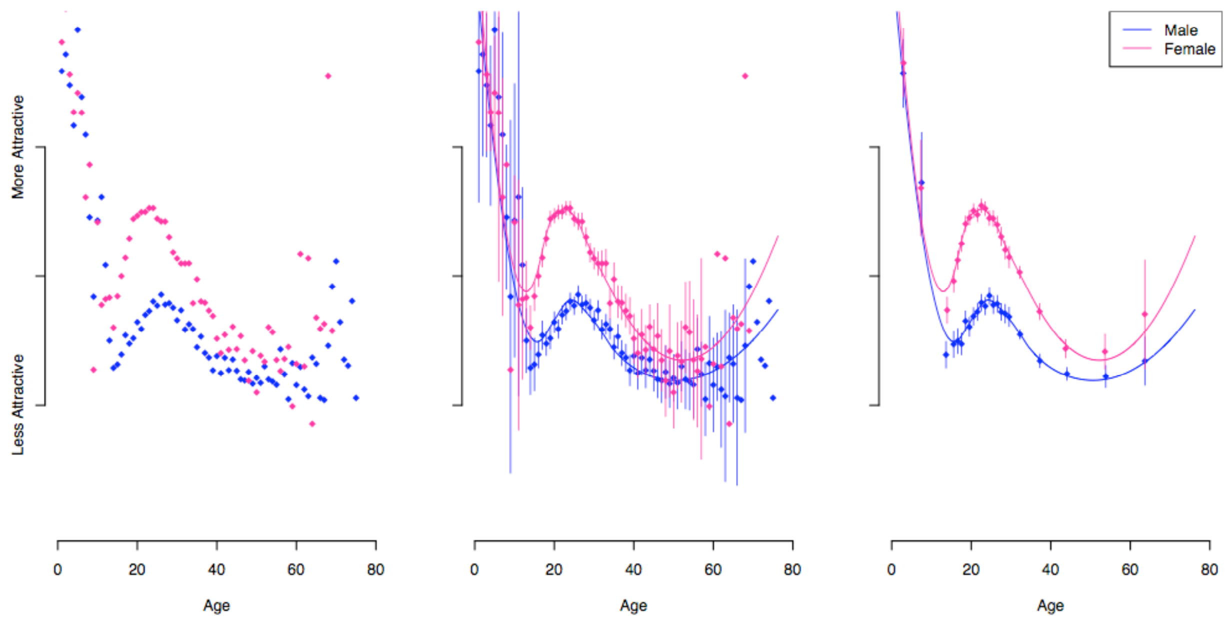
This graph starts to tell a story, but it's still a bit hard to read. Some of the points are averages from thousands of faces, while some of the more elderly points come from just a handful of observations. Therefore there's more noise on the right since the samples are smaller.

We'll add two new features to the plot [Figure AAG_BUCKETS_FANCY]. First, we compute 95% confidence intervals to make sure we're not fooling ourselves into seeing patterns from noise. Confidence intervals are a way to estimate a range of possible means with the limited data we have. Second, we'll fit a loess curve to help visualize aggregate patterns in this noisy sequential data. Ordinarily, we might fit a linear regression to the data, but this data isn't linear, and doesn't look like any function we know of. A loess function ("locally weighted regression") is a way to fit an arbitrary curve to data. It's basically a fancy moving average.

This graph still isn't perfect. There are a number of points around the edges with just one or two samples where it's impossible to compute confidence intervals. This is not surprising if you look back at that age histogram in Figure GOODHIST – people appearing over 50 make up only 1.7% of the data set. Furthermore, many intervals are so big that the data points they represent aren't that meaningful. So for the areas where we have fewer data points -- the very young and the old -- we use larger, 5- and 10-year buckets. This graph looks far less noisy [Figure AAG_BUCKETS_VAR].

Women are generally judged as more attractive than men across all ages except babies. Babies are found to be most attractive, but the attractiveness drops until around age 18 (perhaps users are uncomfortable judging adolescents as "attractive"?) after which it rises and peaks around age 27. After that, attractiveness drops until around age 50, at which point it seems to increase again. But it's hard to say for sure, since the data is very sparse among people perceived to be above 50.

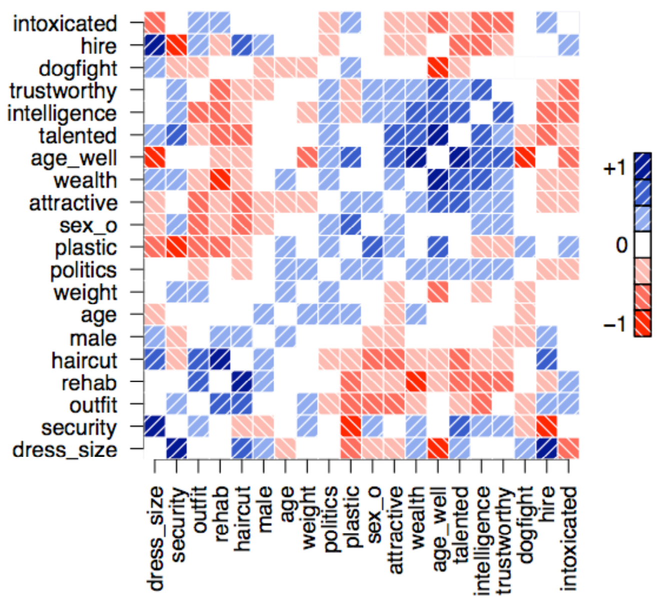
Figure AAG_BUCKETS_ALL: Three iterations of plotting attractiveness vs. age vs. gender. (SIMPLE) Ages averaged within buckets per age year. (FANCY) 95% confidence interval for each bucket, plus loess curves. (VAR) Larger buckets where the data is sparser.



Of course, among the twenty or so non-textual attributes, there are many more relationships to explore. We could make lots more of plots similar to the above; but could we view all interesting interactions at once? Let's stay with the approach of looking at pairwise interactions and make a variant of the pairs plot from earlier. Instead of trying to show a scatterplot in every panel, we instead show a single color indicating the overall correlation between the attributes. Blue is positive correlation, and red is a negative one.

First compute pairwise correlations,	*> cors = cor(d, use='pair')*
and order the attributes to try to	*> ord = order.hclust(cors)*
put similar attributes next to each other.	*> cors = cors[ord,ord]*
Plot the correlation matrix,	*> image(cors, col=col.corrgram(7))*
with axis labels.	*> axis(1, at=seq(0,1, length=nrow(cors)), labels=row.names(cors))*

Figure CORMAT: Pearson correlation matrix. Attribute pairs with blue squares are positively correlated; pairs with red squares are anti-correlated.



Text of questions

- dress_size: What is my dress size?
- security: If you were an airport security guard, would you search me?
- outfit: Do you like my outfit?
- rehab: Will I end up in rehab?
- haircut: Do you like my hairstyle?
- age: How old am I?
- weight: How much do I weigh?
- political_affiliation: What is my political affiliation? (Higher is more conservative)
- plastic_surgery: Have I had plastic surgery?
- sexual_orientation: What is my sexual orientation? (Higher is more gay)
- attractive: How attractive am I?
- wealth: How wealthy am I?
- age_well: Will/Have I age(d) well?
- talented: Am I talented?
- intelligence: How smart am I?
- trustworthy: How trustworthy am I?
- dogfight: Do you think I would win a fight with a medium sized dog?
- hire: Would you hire me?

intoxicated: How intoxicated am I?

This plot is rich with interesting correlations that could warrant further investigation.

- Women are judged as more intelligent than men.
- Women are judged more likely win a dogfight.
- Dress size is only weakly correlated with weight.
- Women are more likely to be hired as security guards.
- People who look like they have had plastic surgery are less likely to be hired as security guards.
- Trustworthiness, intelligence, talent, aging well, wealth, and conservativeness all correlate with each other. An "axis of responsibility"?

Looking at tags

In addition to all this ordinal and numeric data, we have a set of free-form tags that users were able enter about a person's picture. The tags range from descriptive ("freckles", "nosering") to crass ("takemetobed", "dirtypits") to friendly ("you.look.good.in.red") to advice ("cutyourhair", "avoidsun") to editorial ("awwdorable!!!!!", "EnoughUploadsNancy") to mean ("Thefatfriend") to nonsensical ("...", "plokmnjiuhbygvtfcrdxeszwaq"). In general, free text data is more complicated to process.

The first thing to do is examine the distribution of the tags. What's the most common tag? What are the least most common tags?

Load our tags	> *face_tags = read.delim("face_tags.tsv",sep=”\t”,as.is=T)*
then count	> *counts = table(face_tags$tag)*
and rank them.	> *sorted_counts = sort(counts, decreasing=T)*
Show the most common tags.	> *sorted_counts[1:20]*

```
    cute   pretty    happy     nice      fun    young
   81333    40954    36263    33221    30622    27900
   sweet friendly     cool    weird      hot      gay
   20362    14895    14709    12731    12662    12409
    Cute    funny    scary     sexy      old    goofy
   12132    11508    11445    11287    10958    10511
     emo      shy
   10292    10207
```

Show the least common tags. > *tail(sorted_count, 20)*

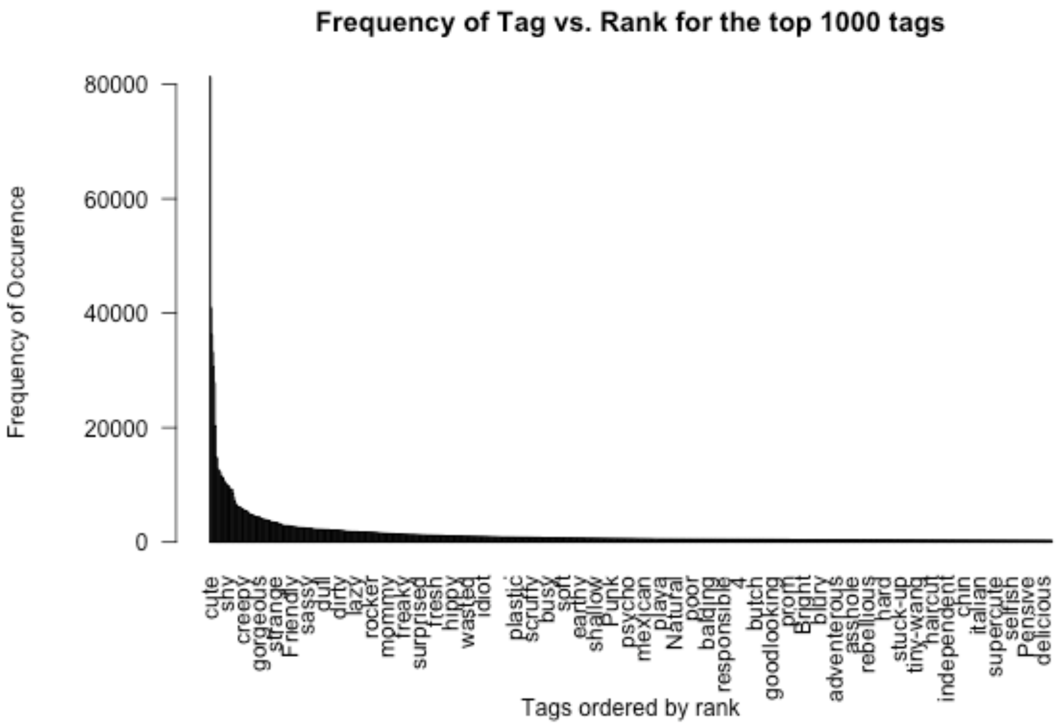
```
                 überdude            übersöt          ünsall
                        1                  1               1
ýour.nose.is.sexymamama!!                  我              浅
                        1                  1               1
                       良                  ？        ♥haiir!!
                        1                  1               1
                     白人               賢母            ⊠⊠;
                        1                  1               1
                     шдд             オタク        ロンリー
                        1                  1               1
                   ешкув              сшеет           херня
                        1                  1               1
             ダースベイダー          Красивая!
                        1                  1
```

Glancing at a few of the tags raises questions about normalization. Should "cute" and "Cute" be merged into the same tag? Should punctuation be dropped entirely? Should that funny looking full-width question mark for Asian languages be considered the same as the standard ASCII question mark? Clearly, it depends on the application. Whenever possible, our instinct is to err on the side of caution and leave the original data intact. This preserves information -- for example, the tags "hot" and "HOT!!!" certainly have different semantic content. It's always easier to carefully merge data when necessary for a specific visualization or analysis, rather than try to guess ahead of time what all the requirements are and be forced to undo earlier normalization decisions.

A basic plot of a tag distribution looks at frequency of a tag against its frequency rank. Typically, when counting words or other lexical items, we see a a quick drop-off from the most frequent words to less frequent words. In our data, there are 290,000 unique tags out of 2.4 million total. The top 1,000 unique tags have 1.4 million occurences -- more than half the total mass of tags. And just among those there's a sharp fall-off. From our table of common tags, we see that the most common tag, "cute", has 36,000 occurrences, but the second most common, "pretty", has just half of that. [Figure TAG_HIST.]

For the top 1000 tags,	*> s = sorted_counts[1:1000]*
draw a plot of their counts.	*> barplot(s)*

Figure TAG_HIST: Tag frequencies for top 1000 tags.



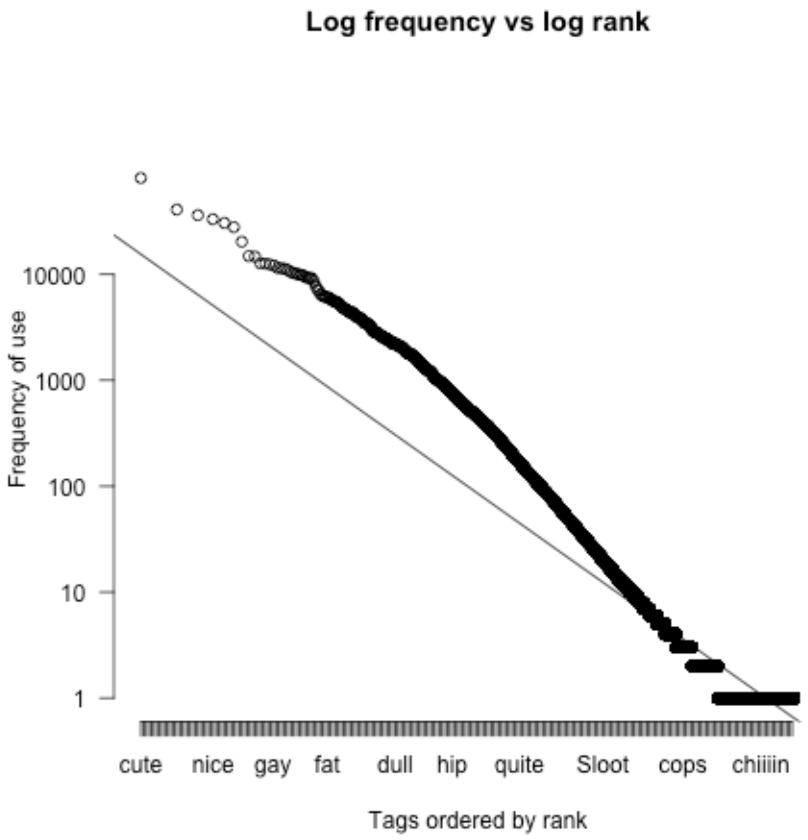
In 1935, the linguist George Zipf observed that word frequency distributions often follow a "power-law", where the frequency of the k-th word is proportional to *(1/k^s)*, where *s* is a constant. Unlike a Gaussian distribution, this distribution has infinite variance, which can make it somewhat unwieldy for certain statistical algorithms. Popular books such as "The Black Swan" and "The Long Tail" have made these distributions famous as "fat tail" or "long tail" distributions. Indeed, our data has quite a long tail: 220,000 words, or 76% of the vocabulary, occur only once.

We can check to see if we have a power-law distribution, by plotting our word frequencies in log space [Figure ZIPF].

Plot log ranks against log frequency.

```
> log_ranks = log(1:length(sorted_counts))
> plot(log_ranks, log(sorted_counts))
```

Figure ZIPF: Tags' log frequencies by log rank, with fitted line from the power law model.



A power law distribution should look linear in the log-log space.

Fit a model of log count against log rank and draw it on our Figure ZIPF.

```
> model = lm(log(sorted_counts) ~ log_ranks)
> abline(model)
```

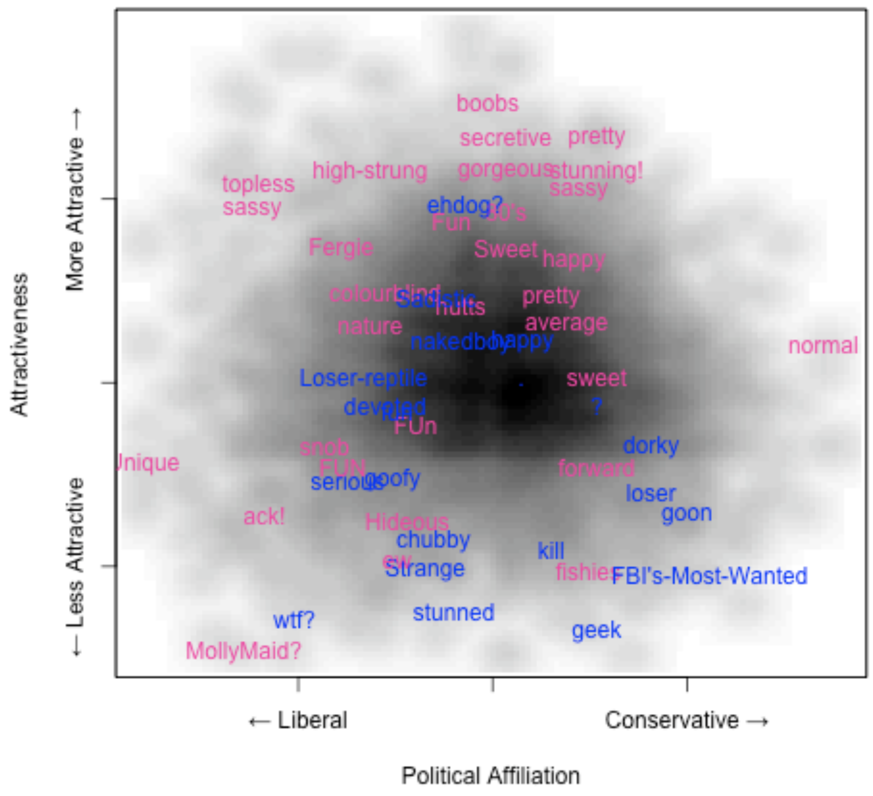
We find our tags' frequencies are fairly close to a *(1/$n^{0.80}$)* distribution. (If you don't think it looks like the best fit line, keep in mind that 76% of all the points are on that last bottom-right ledge of the data.)

If you do this log-log frequency plot on any sort of text -- newspapers, novels, web pages, etc. -- it looks similar.[1] Perhaps unsurprisingly, when FaceStat users write description tags, they're engaging in a linguistic behavior that has some fundamental similarities to other types of human communication.

[1] George Zipf, 1935, *The Psychobiology of Language*. See also http://en.wikipedia.org/wiki/Zipf's_law

How do the tags fit in with the rest of our data? A first pass is to randomly sample from them and overlay them on plots that we've already generated. [Figure TAG_SCAT.]

Figure TAG_SCAT: Tag sample plotted on attractiveness vs. age smoothed scatterplot.



Here the darkness of the plot shows the density in the overall distribution of Political Affiliation vs. Attractiveness. Words are randomly sampled from throughout the distribution. The blue words are tags for males, and the pink words indicate tags for females. This gives us a sense for whether or not the tags are corresponding to the variables in the plot. The data looks roughly reasonable: the tag "average" shows up in the middle of the graph, while someone tagged "topless" in the liberal/attractive quadrant and someone tagged "dorky" is in the conservative/unattractive quadrant. The graph can be regenerated multiple times with different random number seeds to look at distributions of tags throughout the data.

Which words are gendered?

Many social theorists have wondered to what extent gender is reflected in language. Our dataset lets us explore this at the word level: we can find which description tags are most characteristic of male or female faces. We could just count the words that occur most often for men, and the words that occur most often for women, but this mostly just gets words that are frequent everywhere. A better approach is to score tags by their ratio of occurrences between genders. That is, to determine how characteristic a tag *t* is for gender *g*, look at:

$$\frac{\text{no. of occurrences of tag } T \text{ for a face with gender } G}{\text{no. of occurrences of tag } T \text{ overall}}$$

This has a flaw: rare tags introduce noise. For example, any tag that appears just once automatically gets a perfect score of 1 for whichever gender it appeared with. (This is another example of error due to small sample sizes that we saw for sparse age buckets.) A simple way around this is to to use a frequency threshold -- we'll only look at tags that occur more than 100 times.

Calculating these scores -- in statistical terminology, they're maximum likelihood estimates of the conditional probabilities *Pr(G|T)* -- we get the following tables.

Words most characteristic of men:

```
                g      t     ratio
     daddy    122    122 1.0000000
  fatherly    115    115 1.0000000
   fratboy    177    177 1.0000000
    father    172    173 0.9942197
       dad    341    343 0.9941691
    douche    229    231 0.9913420
  Handsome    110    111 0.9909910
   scruffy    149    151 0.9867550
      bald    343    350 0.9800000
      jock    395    404 0.9777228
  handsome    510    524 0.9732824
      thug    141    145 0.9724138
      tool    255    264 0.9659091
    player    522    542 0.9630996
       Gay    307    319 0.9623824
      jerk    131    137 0.9562044
     gamer    103    108 0.9537037
       fag    148    156 0.9487179
      pimp    121    128 0.9453125
```

Words most characteristic of women:

```
                g      t     ratio
    Bubbly    118    118 1.0000000
       Mom    161    161 1.0000000
     busty    148    148 1.0000000
      milf    267    267 1.0000000
       mom   1088   1088 1.0000000
 motherly     396    396 1.0000000
partygirl     221    221 1.0000000
     mommy    307    308 0.9967532
    mother    358    360 0.9944444
     ditzy    144    145 0.9931034
   fjortis    113    114 0.9912281
      MILF    103    104 0.9903846
      Pretty    926    935 0.9903743
 cheerleader    159    161 0.9875776
       boobs    153    155 0.9870968
      makeup    143    145 0.9862069
      bitchy    284    288 0.9861111
      cougar    141    143 0.9860140
      slutty    538    546 0.9853480
        slut    509    517 0.9845261
```

It's perhaps surprising how extremely gendered words such as "handsome", "gamer", "Bubbly" and "slut" are. They appear with their gender almost ALL of the time.

Clustering

What are the typical types of people in our data? Clustering is a powerful statistical method to find this sort of pattern. A clustering algorithm splits data points into several characteristic classes by grouping together similar instances. There are many methods for clustering. One of the most popular and simple methods is called *k-means*. In k-means, each cluster has a center point, a "centroid." Several different centroids are found in the data and each data point is assigned to a centroid. The algorithm iteratively adjusts the clusters so that as many data points as possible are close to their assigned centroids.

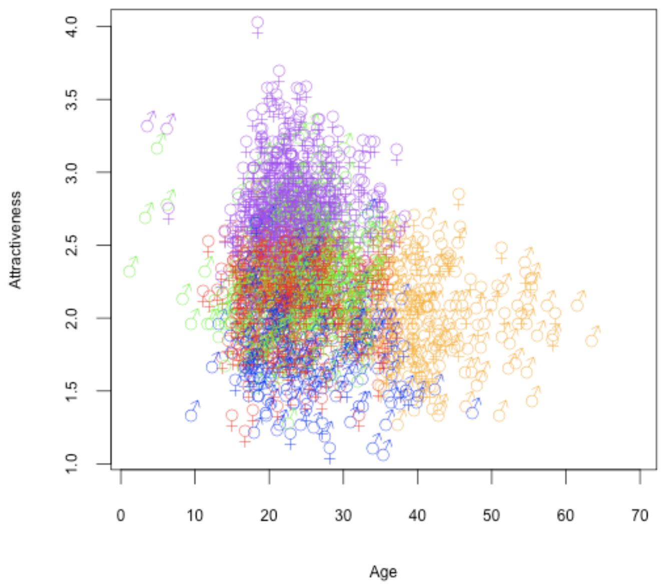
In our dataset, each face has about 20 numeric attributes. Thus faces are points in a 20-dimensional space. K-means will place faces into several different clusters within that space, trying to select clusters where faces are as similar to their cluster's center as possible.

One unfortunate aspect about k-means clustering is that you have to pick a fixed number of clusters, "k", up front. There isn't an obvious way to choose the number of clusters. The best thing to do is to try a few different numbers and see what patterns emerge. Here's one run of k-means we did that gave reasonable output.

Preprocess the data,
by changing missing values to the mean,
and unit-normalizing values,
which usually makes k-means work better.
Then run k-means for 5 clusters,
and plot our standby, attractiveness vs. age,
but color by
cluster assignment,
and have fun with unicode.

```
> norm_data = apply(d, 2, function(x) {
   x[is.na(x)] = mean(x, na.rm=TRUE)
   x = (x - mean(x)) / sd(x)
   x })
> clus = kmeans(norm_data, 5)
> plot(d$age, d$attractive,
   col = c("red", "purple", "blue", "orange",
     "green",”darkturquoise”)[clus$cluster],
   pch = ifelse(d$male, '\u2642', '\u2640'))
```

Figure KMEANS_AAG: Attractiveness vs. age, colored by cluster, showing a subsample of 2000 points.



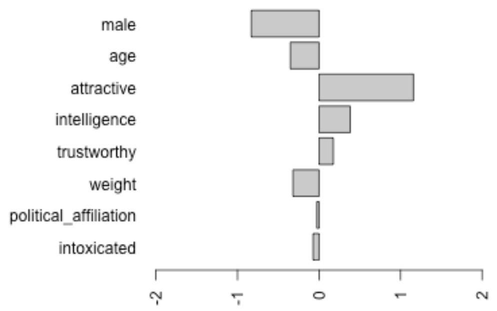
Points on the scatterplot are faces, with colors corresponding to the clusters they were assigned to. We're showing faces within the attractiveness vs. age space, like our earlier plots. A few clusters are already interpretable: the orange cluster corresponds to older people, while purple seems to be attractive young people, and so on.

This plot only shows two or three dimensions of the data, so does not adequately summarize the clustering algorithm, which compares faces in the full 20-dimensional space. That's why some clusters overlap above: for example, red and green seem to have fairly similar ranges of age and attractiveness. Those clusters must differ by other attributes.

Let's look at individual clusters in several ways. First, we show a cluster's attribute weights. This is the position of the cluster's centroid point; it can be thought of as the typical attributes for a face in that cluster. So if you looked at the average points per cluster in the above graph, that would give you the cluster weightings for age and attractiveness. (We'll show eight attributes; the rest are insignificant because of too many missing values.) Second, we show the top ten characteristic tags for faces in that cluster, ranked by conditional probability like in the gender analysis above.

Figure CLUSTER5.

Purple cluster centroid



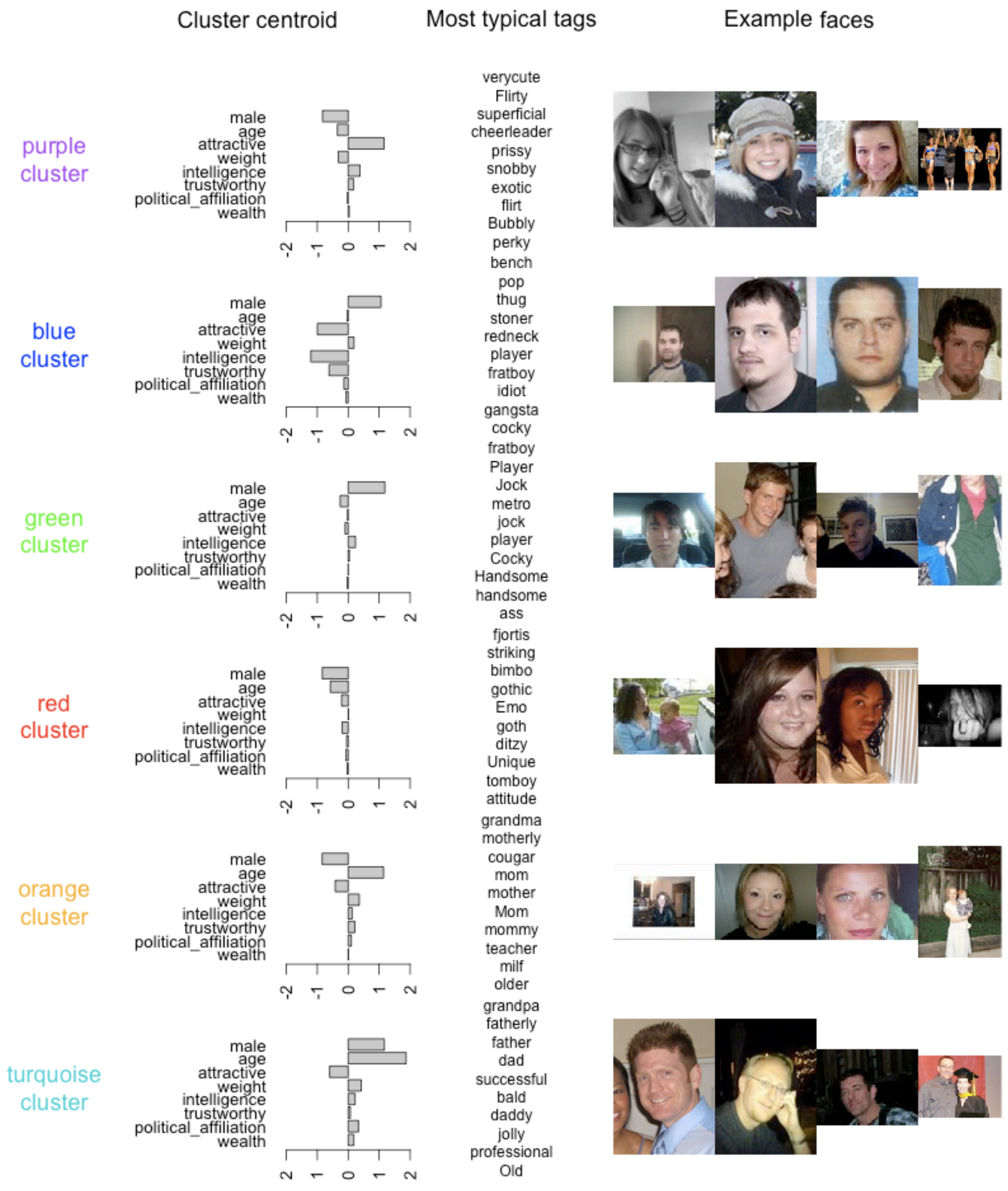
Purple cluster tags
c = # of occurrences of the tag in the purple cluster
t = # of occurrences of the tag overall

	c	t	ratio
verycute	226	525	0.430
striking	204	606	0.336
Flirty	113	439	0.257
superficial	106	437	0.242
cheerleader	202	837	0.241
prissy	126	561	0.224
snobby	121	545	0.222
flirt	170	785	0.216
exotic	498	2327	0.214
peppy	123	579	0.212

First off, here's the purple cluster in Figure CLUSTER2. This is a heavily female, highly attractive cluster. The tags are interesting. They uncannily resemble the attributes -- in fact, if you cover up the graph on the left, you probably could guess many of the attributes for the group. The tags paint a coherent and vivid picture -- even though our k-means algorithm completely ignored this information! This illustrates that tags have intuitive correlations to social attributes. (Perhaps this is not surprising.)

We've put all the clusters in the table CLUSTER_TABLE, along with the four most representative faces (meaning, ones closest to the centroid) per cluster.

Figure CLUSTER_TABLE: cluster centroids, tags, and exemplars.



Some of the clusters have straightforward interpretations and some are less clear.

- Purple cluster: young, attractive women.
- Blue cluster: unattractive, unintelligent men. ("Losers"?)
- Green cluster: other, more generic young men. Many of its tags are also highly likely for the blue cluster.
- Red cluster: other young women.
- Orange cluster: older women.
- Turquoise cluster: older men.

Clustering can be useful to find high dimensional patterns or groups in data which are hard to visualize in two dimensions. On the other hand, it's hard to validate whether clustering is telling you anything "real". There are many clustering algorithms and many parameters to tweak (such as that *k*) which can give different results. Was this exercise useful? Well, the clusters seem fairly coherent, and are quite suggestive of a number of patterns. It's interesting to see vivid sets of tags are associated with each. And k-means might provide an analogue with how our minds think of people -- from the centroids and tag sets we can imagine a prototypical person representing each cluster.

Conclusion

Our data indicates people hold some familiar stereotypes. Women are considered more attractive than men. Age has a stronger attractiveness effect for women than men. The space of social attributes falls along lines that feel familiar to us: jocks, fathers, attractive young women. But there are also some potential surprises -- babies are most attractive, conservatives look more intelligent, etc. We found examples of gendered words.

We're tempted to go on and on with suggestive findings, but the point of this chapter is not to come to any particular conclusion. Instead, we wanted to show some examples of the rich set of significant patterns contained in a large, messy dataset of human judgments. A more rigorous data collection process -- like carefully controlled lab experiments -- would never produce such a volume of data, but could be useful as follow-up experiments.

Every day we reveal more and more about ourselves through the things we buy, the websites we use, the queries we search for, the messages we send, and the places we go. Whether we like it or not, for the first time in human history all this data is being carefully saved. Setting aside the important privacy concerns, the value to social science is enormous. Through this mess of repurposed information, we will learn about ourselves in completely new ways.

Acknowledgments

Many thanks to all the people who gave feedback for early drafts of this chapter: Joanna Gubman, Sasha Goodman, Jeff Hammerbacher, Mike Love, Will Moffat, and Toby Segaran. FaceStat owes its existence to the hard and brilliant work of our colleague Chris Van Pelt.

References

We have uploaded a subset of the data, as well as notes and code to help replicated our analyses, at http://data.doloreslabs.com.

If you're interested in learning R, we recommend two websites.

1. Quick-R: http://statmethods.net. High-level overviews and topic guides. By Robert Kabacoff.
2. RSeek: http://rseek.org. A search engine for R documentation, packages, and mailing lists. By Sasha Goodman.

R's official website is http://www.r-project.org. If you are interested in how it compares to other data analysis packages, see the many comments on an early draft of Table COMP at http://anyall.org/blog/?p=421 (especially the amazing SAS war stories!)

Aside from R's core functionality, some of the add-on packages we used include: corrgram, flowCore, gclus, geneplotter, plyr, and pixmap.

Good overviews of clustering, loess, and other machine learning techniques are in *The Elements of Statistical Learning* by Trevor Hastie, Robert Tibshirani, and Jerome Friedman, 2008.

The section on tags barely touches the surface of statistical language analysis. For more, see *Foundations of Statistical Natural Language Processing* by Christopher Manning and Hinrich Schütze, 1999; and also *Speech and Language Processing* by Daniel Jurafsky and James H. Martin, 2008.

There are many better ways for estimating confidence intervals for the attractiveness vs. age analysis. One method is partial pooling; see pp. 252—258 of Andrew Gelman and Jennifer Hill's *Data Analysis Using Regression and Multilevel Models*, 2006.

What we do in this chapter is called "exploratory data analysis" – as opposed to ploddingly careful hypothesis testing that is usually taught in statistical methodology courses. EDA was strongly advocated by statistician John Tukey in his 1977 book of the same name.

Our startup, Dolores Labs, specializes in crowdsourcing: collecting human task data from large masses of people to solve practical problems in content moderation, information extraction, web search relevance, and other domains. We collect, look at, and automatically analyze lots of human judgment data. You can see follow-ups to this chapter, and analyses of other subjects like sex, colors, and ethics, at our blog: http://blog.doloreslabs.com.