How to Win Ratings and Influence Reviewers: Preferential Attachment in Rating Systems

Jao-ke Chin-Lee
SEAS
Harvard College

April 2015
Abstract

In this paper we introduce the concept of preferential attachment in the context of recommendation and rating systems. We present several models incorporating different qualities that may manifest in such systems, such as inherent bias, and examine the resulting degree distributions (i.e. ratings) as snapshots and through time. We then take preliminary steps towards testing real-world feasibility with the Yelp Academic Dataset.
0.1 Introduction

“Nobody goes there anymore, it’s too crowded.” –Yogi Berra

During the vote for Scottish independence, a series of polls were taken and distributed by the media, which, due to the drastic age divide, gave rise to the concern that younger citizens than ever before could vote in the referendum [6]. This controversy in turn which suggested that independence was the more popular outcome by a large scale. What passed, of course, was that a larger turnout against independence occurred, resulting in Scotland remaining in the United Kingdom.

The concerns raised by the polls themselves raise a counterfactual question: what if the polls had not been performed? Would the turnout have been different? And assuming the turnout would have been different, why: was the demographic taking the polls was not the demographic voting (i.e. there was selection bias on the poll-taking side); was the demographic being polled was not the demographic voting (i.e. there was selection bias on the poll-making side); did the older demographic, aghast at the potential outcome based on the predictions, turned out in much larger numbers actually to vote?

We focus on the latter question, namely, the case for feedback into a multiagent system. In a case like the vote for Scottish independence, or Yogi Berra’s canny observation—assuming that the predictions did in fact sufficiently influence voters to turn out in different numbers—the very act of forming a prediction or estimate of the true state, and disseminating it, affects the true state. In this mental model, the critical aspect is the dissemination itself: via polls in the former case, or simply by definition (presence of others, which is social, either communicated by mouth or by sight) in the latter.

0.1.1 Recommendation systems, voting, social influence, and their interactions

Previous work on determining the effect of social influence can be found in [5] and [1]. In [5], Muchnick et al. experimentally test the effect of up-votes, down-votes, and comments on a social news aggregation website. In their experimental design, users see the comment scores (up-votes and down-votes tied to comments) simultaneously with the comments themselves to mitigate viewing
bias, and there is no order imposed on the comments for showing. They found that positive influence, i.e. up-votes, accumulated, but down-votes did not, and appeared to be “corrected” by the crowd, suggesting a greater readiness to increase and respond to positivity. In [1], Aral combined the notions of voting and social influence by examining how influence travels through social networks, specifically, Facebook, ultimately to affect voting, by tracking the spread of a Facebook message “I voted” that users could volunteer to display on their page, and which was coupled with a statement encouraging voting as well as information on who of a user’s social network had already voted, versus a purely informational message. What Aral found was that those who had been treated via social means were more likely to report themselves as having voted, look for polling information, and vote, than those who had only seen the information message.

With this motivation, we proceed to consider several models for influence through recommendation systems, based on preferential attachment.

0.2 Preliminaries

We will use the following notation, redescribed in greater detail where used, but provided below for convenience.
number of possible ratings

time

index into set of possible rankings

weight of preferential attachment behavior when connecting to graph

weight of truncated normal behavior when connecting to graph

length of history remembered

cdf of truncated normal distribution centered around $\mu$ and evaluated at $x$

pdf of discretized truncated normal distribution centered around $\mu$ and evaluated at $x$

proportion of ratings of level $k$ at time $t$

proportion of ratings observed (at some point in time) of level $k$

error between observed proportions $\{\gamma_k\}$ and predicted, given model parameters (trailing arguments), at time $t$

### 0.3 Preferential Attachment Model

We assume a framework in which users input some discrete level to indicate quality: this allows us to model the rating scheme in terms of a graph, as per preferential attachment. Each definition introduces the model in the framework of graphs, but we will chiefly concern ourselves with degree distribution rather than the graphs themselves.

**Definition 1** (Simple preferential attachment rating). We define a simple preferential attachment rating graph $G^n_\alpha = (S, R, E)$ to be an undirected bipartite graph in which $n = |S|$ vertices represents some value, e.g. ratings, and the other $|R|$ vertices represent selections of one of those $n$ values. When a vertex $r$ joins set $R$, with probability $1 - \alpha$ we add edge $(r, s)$ where $s$ is drawn from $S$ uniformly, and with probability $\alpha$ we add edge $(r, s)$ where $s$ is drawn from $S$ proportionally to its degree.
Suppose we have a simple preferential attachment rating graph $G_n^\alpha$. Let the proportion of ratings of value $k$ at time $t$ be $\rho_t(k)$. Then we have

$$\rho_t(k) = \begin{cases} \frac{\rho_{t-1}(k) - 1}{t} & \text{with probability } \frac{1-\alpha}{n} + \alpha \rho_{t-1}(k) \\ \frac{\rho_{t-1}(k) - 1}{t} & \text{with probability } 1 - \frac{1-\alpha}{n} - \alpha \rho_{t-1}(k) \end{cases},$$

that is, we add one rating at each time step, incrementing $k$’s count with probability weighted $1 - \alpha$ on uniform probability, and weighted $\alpha$ on probability proportional to degree, which we must extract from $\rho_{t-1}(k)$, and leaving the count untouched otherwise.

We can rewrite this as

$$\rho_t(k) = \begin{cases} (1 - \frac{1}{t}) \rho_{t-1}(k) + \frac{1}{t} & \text{with probability } \frac{1-\alpha}{n} + \alpha \rho_{t-1}(k) \\ (1 - \frac{1}{t}) \rho_{t-1}(k) & \text{with probability } 1 - \frac{1-\alpha}{n} - \alpha \rho_{t-1}(k) \end{cases},$$

hence

$$\mathbb{E}[\rho_t(k)] = \left(\frac{1-\alpha}{n} + \alpha \rho_{t-1}(k)\right) \left[(1 - \frac{1}{t}) \rho_{t-1}(k) + \frac{1}{t}\right]$$

$$+ \left(1 - \frac{1-\alpha}{n} - \alpha \rho_{t-1}(k)\right) \left[(1 - \frac{1}{t}) \rho_{t-1}(k)\right]$$

$$= \left(1 - \frac{1}{t}\right) \rho_{t-1}(k) + \frac{1 - \alpha}{t} \frac{1}{n} + \frac{\alpha}{t} \rho_{t-1}(k)$$

$$= \left(1 - \frac{1-\alpha}{t}\right) \rho_{t-1}(k) + \frac{1 - \alpha}{t} \frac{1}{n}$$

If we consider this roughly by transforming everything to expectation (we now consider expected $\hat{\rho}_t(k)$), we can rewrite the equation in terms of the rate of proportion change.

$$\Delta \hat{\rho}_t(k) = \frac{1-\alpha}{t} \left[\frac{1}{n} - \hat{\rho}_t(k)\right]$$

where $\rho_1(k') = 1$ for the first $k$ selected, denoted $k'$, and $\rho_1(k \neq k') = 0$ for every other.
In the discrete case, this becomes a difference equation with solution, for \( t \geq 2, \)

\[
\hat{\rho}_t(k') = \prod_{s=2}^{t+1} \left( 1 - \frac{1 - \alpha}{s} \right) + \sum_{s'=2}^{t+1} \left( \frac{1 - \alpha}{s'n} \prod_{s=s'+1}^{t+1} \left( 1 - \frac{1 - \alpha}{s} \right) \right) \tag{2}
\]

\[
\hat{\rho}_t(k \neq k') = \sum_{s'=2}^{s+1} \left( \frac{1 - \alpha}{s'n} \prod_{s=s'+1}^{t+1} \left( 1 - \frac{1 - \alpha}{s} \right) \right) \tag{3}
\]

This is extremely unwieldy, however, so, inspired by the approaches towards modelling the time evolution of degree distribution in [4] and [2], we will also consider the continuous case, in which this becomes a separable differential equation with solution

\[
\hat{\rho}_t(k') = \frac{1}{n} + \left( 1 - \frac{1}{n} \right) t^{\alpha-1} \tag{4}
\]

\[
\hat{\rho}_t(k \neq k') = \frac{1}{n} - \frac{1}{n} t^{\alpha-1} \tag{5}
\]

**Proof.** For convenience, we let \( f(t) = \hat{\rho}_t(k) \): we can consider the functions of \( k \) homogeneously as the only difference lies in initial condition. Then

\[
f' = \frac{1 - \alpha}{t} \left[ \frac{1}{n} - f \right] \text{ by Eqn. 1}
\]

\[
\Rightarrow f' = \frac{1 - \alpha}{t} \left[ \frac{1}{n} - f \right]
\]

\[
\Rightarrow - \ln \left( \frac{1}{n} - f \right) = (1 - \alpha) \ln t + c, \text{ } c \text{ some constant}
\]

\[
\Rightarrow \frac{1}{n} - f = Ct^{\alpha-1}, \text{ } C \text{ some new constant}
\]

\[
\Rightarrow f = \frac{1}{n} - Ct^{\alpha-1}
\]

Now we recall that \( \rho_1(k') = 1 \), so for case \( k' \) we require \( 1 = 1/n - C \cdot 1 \), or \( -C = 1 - 1/n \). Similarly, \( \rho_1(k \neq k') = 0 \), so for case \( k \neq k' \) we require \( 0 = 1/n - C \cdot 1 \), or \( -C = -1/n \) (note that in the above derivation, whether the negation appears on the left-hand side or right-hand side depends on whether \( f < 1/n \) or \( f > 1/n \), because log is undefined on the negatives, and which depends on the initial condition; the result is unchanged, however). Substituting yields the desired result. \( \square \)
Given empirical proportions $\gamma_1, \ldots, \gamma_n$, we define the sum of squared errors

$$\varepsilon(t, \gamma_1, \ldots, \gamma_n, \alpha) = \left[\left(\frac{1}{n} + \left(1 - \frac{1}{n}\right) t^{\alpha-1} - \gamma_k\right)^2 + \sum_{k \neq k'} \left(\frac{1}{n} - \frac{1}{n} t^{\alpha-1} - \gamma_k\right)^2\right]$$

and determine the $\alpha$ minimizing such $\varepsilon$ to be

$$\alpha = 1 + \frac{1}{\ln t} \cdot \ln \left(\gamma_{k'} - \frac{1}{n-1} \sum_{k \neq k'} \gamma_k\right)$$

**Proof.** We differentiate with respect to $\alpha$, and solve for it equal to 0; because the sum of squares is convex, this guarantees a minimum.

\[
0 \equiv \frac{\partial \varepsilon}{\partial \alpha} = 2 \left[\frac{1}{n} + \left(1 - \frac{1}{n}\right) t^{\alpha-1} - \gamma_k\right] \cdot \left[\left(1 - \frac{1}{n}\right) t^{\alpha-1} \ln t\right] \\
+ 2 \sum_{k \neq k'} \left[\frac{1}{n} - \frac{1}{n} t^{\alpha-1} - \gamma_k\right] \cdot \left[-\frac{1}{n} t^{\alpha-1} \ln t\right] \\
= \left[\frac{1}{n} + \left(1 - \frac{1}{n}\right) x - \gamma_k\right] \cdot \left(1 - \frac{1}{n}\right) x \\
+ \sum_{k \neq k'} \left[\frac{1}{n} - \frac{1}{n} x - \gamma_k\right] \cdot \left(-\frac{1}{n} x\right) \text{ cancelling and setting } x \triangleq t^{\alpha-1} \\
= -\left[\frac{1}{n} + \left(1 - \frac{1}{n}\right) x - \gamma_k\right] \cdot \left(1 - \frac{1}{n}\right) + \sum_{k \neq k'} \left[\frac{1}{n} - \frac{1}{n} x - \gamma_k\right] \cdot \frac{1}{n} \text{ cancelling} \\
= -\frac{1}{n} \left(1 - \frac{1}{n}\right)^2 x + \left(1 - \frac{1}{n}\right) \gamma_{k'} + (n-1) \left(\frac{1}{n} - \frac{1}{n} x\right) \cdot \frac{1}{n} - \frac{1}{n} \sum_{k \neq k'} \gamma_k \\
\text{observing that there are } n-1 \text{ independent terms we can pull out} \\
= -\frac{1}{n} \left(1 - \frac{1}{n}\right) - \left(1 - \frac{1}{n}\right)^2 x + \left(1 - \frac{1}{n}\right) \gamma_{k'} + \frac{1}{n} \left(1 - \frac{1}{n}\right) \left(1 - x\right) - \frac{1}{n} \sum_{k \neq k'} \gamma_k \\
= -\frac{1}{n} \left(1 - \frac{1}{n}\right) x + \gamma_{k'} + \frac{1}{n} (1 - x) - \frac{1}{n-1} \sum_{k \neq k'} \gamma_k \text{ cancelling} \]
Hence

\[- \frac{1}{n} + \gamma_{k'} - \frac{1}{n-1} \sum_{k \neq k'} \gamma_k = \left( 1 - \frac{1}{n} \right) x - \frac{1}{n} (1-x) \]

\[ \Rightarrow - \frac{1}{n} + \gamma_{k'} - \frac{1}{n-1} \sum_{k \neq k'} \gamma_k = x - \frac{1}{n} \]

\[ \Rightarrow x = t^{\alpha-1} = \gamma_{k'} - \frac{1}{n-1} \sum_{k \neq k'} \gamma_k \]

\[ \Rightarrow \alpha = 1 + \frac{1}{\ln t} \cdot \ln \left( \gamma_{k'} - \frac{1}{n-1} \sum_{k \neq k'} \gamma_k \right) \], as desired.

Recall that these results assume \( \rho_1(k') = 1 \) for the first \( k \) selected for attachment and \( \rho_1(k \neq k') = 0 \) for every other: this simplifying condition captures starting the preferential attachment after a single initial attachment. We may wish to begin attaching preferentially only after having initialized our graph to some state, in which case we need only initialize \( \rho_1(k) \) appropriately.

**Definition 2** (Simple delayed preferential attachment rating). We define a simple delayed preferential attachment rating graph \( G_{n,\rho_1} = (S, R, E) \) to be an undirected bipartite graph in which \( n = |S| \) vertices represent some value, e.g. ratings, and the other \( |R| \) vertices represent selections of one of those \( n \) values. \( E \) is initialized such that the degree proportion of vertex \( k \in S \) is precisely \( \rho_1(k) \). When a vertex \( r \) joins set \( R \), with probability \( 1 - \alpha \) we add edge \((r, s)\) where \( s \) is drawn from \( S \) uniformly, and with probability \( \alpha \) we add edge \((r, s)\) where \( s \) is drawn from \( S \) proportionally to its degree.

Our transition functions remain the same, but our initial conditions are different, so our difference equation solution becomes, for \( t \geq 2 \),

\[
\hat{\rho}_t(k) = \prod_{s=2}^{t+1} \rho_1(k) \left( 1 - \frac{1-\alpha}{s} \right) + \sum_{s'=2}^{t+1} \left( \frac{1-\alpha}{s'} \prod_{s=s'+1}^{t+1} \left( 1 - \frac{1-\alpha}{s} \right) \right) \tag{8}
\]
with continuous approximation solution

\[ \hat{\rho}_t(k) = \frac{1}{n} + \left( \rho_1(k) - \frac{1}{n} \right) t^{\alpha-1} \]  

(9)

Accordingly, we update the error function in this more general case to be

\[ \varepsilon(t, \rho_1, \gamma_1, \ldots, \gamma_n, \alpha) = \sum_k \left[ \frac{1}{n} + \left( \rho_1(k) - \frac{1}{n} \right) t^{\alpha-1} - \gamma_k \right]^2 \]  

(10)

which is minimized by setting

\[ \alpha = 1 + \frac{1}{\ln t} \left[ \ln \sum_k \left( \rho_1(k) - \frac{1}{n} \right) \left( \gamma_k - \frac{1}{n} \right) - \ln \sum_k \left( \rho_1(k) - \frac{1}{n} \right)^2 \right] \]  

(11)

**Proof.** Similarly to before, we differentiate \( \varepsilon \) with respect to \( \alpha \), which yields

\[ 0 = \frac{\partial \varepsilon}{\partial \alpha} = 2 \sum_k \left[ \frac{1}{n} + \left( \rho_1(k) - \frac{1}{n} \right) t^{\alpha-1} - \gamma_k \right] \cdot \left[ \left( \rho_1(k) - \frac{1}{n} \right) t^{\alpha-1} \ln t \right] \]

\[ = \frac{\partial \varepsilon}{\partial \alpha} = \sum_k \left[ \frac{1}{n} + \left( \rho_1(k) - \frac{1}{n} \right) t^{\alpha-1} - \gamma_k \right] \cdot \left( \rho_1(k) - \frac{1}{n} \right) \text{ cancelling} \]

\[ = \sum_k \left( \rho_1(k) - \frac{1}{n} \right)^2 t^{\alpha-1} + \sum_k \left( \frac{1}{n} - \gamma_k \right) \left( \rho_1(k) - \frac{1}{n} \right) \]

Hence

\[ t^{\alpha-1} = \frac{\sum_k \left( \rho_1(k) - \frac{1}{n} \right) \left( \gamma_k - \frac{1}{n} \right)}{\sum_k \left( \rho_1(k) - \frac{1}{n} \right)^2} \]

\[ \Rightarrow \alpha = 1 + \frac{1}{\ln t} \left[ \ln \sum_k \left( \rho_1(k) - \frac{1}{n} \right) \left( \gamma_k - \frac{1}{n} \right) - \ln \sum_k \left( \rho_1(k) - \frac{1}{n} \right)^2 \right] \]

Note that with \( \rho_1(k') = 1, \rho_1(k \neq k') = 0 \), this indeed simplifies to Eqn. 7.

Preferential attachment can also be time-dependent: earlier ratings may have less of an influence than more recent ones, or vice versa. Below we consider the case where only the past \( h \) ratings exert influence.

**Definition 3** (Simple amnesic preferential attachment rating). We define a simple amnesic preferential attachment rating graph \( G_{\alpha,h}^n = (S,R,E) \) to be an undirected bipartite graph in which
Given such a simple forgetful preferential attachment rating graph $G_{\alpha,h}^n$, then, the proportion transitions are as follows:

$$\rho_t(k) = \begin{cases} 
(1 - \frac{1}{t}) \rho_{t-1}(k) + \frac{1}{t} & \text{with probability } \frac{1-\alpha}{n} + \alpha \left( \frac{(t-1)\rho_{t-1}(k)-(t-h)\rho_{t-h}(k)}{h} \right) \\
(1 - \frac{1}{t}) \rho_{t-1}(k) & \text{with probability } 1 - \frac{1-\alpha}{n} - \alpha \left( \frac{(t-1)\rho_{t-1}(k)-(t-h)\rho_{t-h}(k)}{h} \right)
\end{cases}$$

Although this is infeasible to solve analytically, we will return to this model in simulation.

In some cases, attachment may have an underlying bias. For instance, there may be an underlying true state or rating about which attachment is sampled; there may be an imposed rating to which attachment adds some error; each attachment itself may be sampled about some mean. We therefore also consider the additive effect of a truncated discrete normal component.

**Definition 4** (Discrete truncated normal distribution). We define $\varphi_\mu(x) \triangleq \Phi_\mu(x+0.5) - \Phi_\mu(x-0.5)$, where $\Phi_\mu(x)$ is the cumulative distribution function of a truncated normal with mean $x$ and variance 1, evaluated at $x$.

$\varphi_\mu(x)$ then represents the probability distribution function of a discretized truncated normal with mean $x$ and variance 1, evaluated at $x$.

**Definition 5** (Pseudonormal preferential attachment rating). We define a pseudonormal preferential attachment rating graph $G_{\alpha,\beta}^{n,\mu} = (S, R, E)$ to be an undirected bipartite graph in which $n = |S|$ vertices represents some value, e.g. ratings, and the other $|R|$ vertices represent selections of one of those $n$ values. When a vertex $r$ joins set $R$, with probability $1 - \alpha - \beta$ we add edge $(r, s)$ where $s$ is drawn from $S$ uniformly, with probability $\alpha$ we add edge $(r, s)$ where $s$ is drawn from $S$ proportionally to its degree, and with probability $\beta$ we add edge $(r, s)$ where $s$ is drawn from $S$ with probability $\varphi_\mu(s)$. 

9
μ is then some measure of bias towards a mean. This can be a “true state” type of mean, in which case this model places weight on raters “knowing” the true state of the rated item and behaving distributed about it. This can also be a “perceived state” type of mean, in which case this model places weight on raters aggregating data from the other raters and—not dissimilar to the case of pure preferential attachment—acting based on such perception, but, with some noise, in a normally distributed manner.

Note that in order for this definition to make sense, we assume that the sequence of possible ratings 1, . . . , n has intrinsic meaning: that they monotonically map some categorical and relative quality. In the context of ratings, for instance, they should indicate a categorical relative level or ranking, such as first, second, third, or poor, fair, good.

Suppose we have a pseudonormal preferential attachment rating graph \( G_{\alpha, \beta}^{n, \mu} \). Then we have

\[
\rho_t(k) = \begin{cases} 
(1 - \frac{1}{t}) \rho_{t-1}(k) + \frac{1}{t} & \text{with probability } \frac{1 - \alpha - \beta}{n} + \alpha \rho_{t-1}(k) + \beta \varphi(\mu)(k) \\
(1 - \frac{1}{t}) \rho_{t-1}(k) & \text{with probability } 1 - \frac{1 - \alpha - \beta}{n} - \alpha \rho_{t-1}(k) - \beta \varphi(\mu)(k)
\end{cases}
\]

so similarly to the simple case,

\[
\mathbb{E}[\rho_t(k)] = \left( \frac{1 - \alpha - \beta}{n} + \alpha \rho_{t-1}(k) + \beta \varphi(\mu)(k) \right) \left[ (1 - \frac{1}{t}) \rho_{t-1}(k) + \frac{1}{t} \right] \\
+ \left( 1 - \frac{1 - \alpha - \beta}{n} - \alpha \rho_{t-1}(k) - \beta \varphi(\mu)(k) \right) \left[ (1 - \frac{1}{t}) \rho_{t-1}(k) \right] \\
= \left( 1 - \frac{1}{t} \right) \rho_{t-1}(k) + \frac{1 - \alpha - \beta}{t} \rho_{t-1}(k) + \frac{\alpha}{t} \rho_{t-1}(k) + \frac{\beta}{t} \varphi(\mu)(k) \\
= \left( 1 - \frac{1 - \alpha}{t} \right) \rho_{t-1}(k) + \frac{1 - \alpha - \beta}{t} \rho_{t-1}(k) + \frac{\beta}{t} \varphi(\mu)(k)
\]

As before, we approximate it as

\[
\Delta \hat{\rho}_t(k) = \frac{1 - \alpha}{t} \left[ (1 - \frac{\beta}{1 - \alpha}) \frac{1}{n} + \frac{\beta}{1 - \alpha} \varphi(\mu)(k) - \hat{\rho}_t(k) \right]
\]

10
which yields discrete solution
\[
\hat{\rho}_t(k') = \prod_{s=2}^{t+1} \left( 1 - \frac{1 - \alpha}{s} \right) + \sum_{s' = 2}^{t+1} \left( \left[ 1 - \frac{\alpha - \beta}{s'} \cdot \varphi_\mu(k) \right] \cdot \prod_{s = s' + 1}^{t+1} \left( 1 - \frac{1 - \alpha}{s} \right) \right) 
\]

(12)

\[
\hat{\rho}_t(k \neq k') = \sum_{s' = 2}^{t+1} \left( \left[ 1 - \frac{\alpha - \beta}{s'} \cdot \varphi_\mu(k) \right] \cdot \prod_{s = s' + 1}^{t+1} \left( 1 - \frac{1 - \alpha}{s} \right) \right) 
\]

(13)

and continuous solution
\[
\hat{\rho}_t(k') = \left( 1 - \frac{\beta}{1 - \alpha} \right) \frac{1}{n} + \frac{\beta}{1 - \alpha} \varphi_\mu(k) + \left( 1 - \left( 1 - \frac{\beta}{1 - \alpha} \right) \frac{1}{n} - \frac{\beta}{1 - \alpha} \varphi_\mu(k) \right) t^{\alpha - 1} 
\]

(14)

\[
\hat{\rho}_t(k \neq k') = \left( 1 - \frac{\beta}{1 - \alpha} \right) \frac{1}{n} + \frac{\beta}{1 - \alpha} \varphi_\mu(k) - \left( \left( 1 - \frac{\beta}{1 - \alpha} \right) \frac{1}{n} + \frac{\beta}{1 - \alpha} \varphi_\mu(k) \right) t^{\alpha - 1} 
\]

(15)

**Definition 6** (Delayed pseudonormal preferential attachment rating). We define a delayed pseudonormal preferential attachment rating graph \( G_{\alpha, \beta, \rho_1}^{n, \mu} = (S, R, E) \) to be an undirected bipartite graph in which \( n = |S| \) vertices represents some value, e.g. ratings, and the other \(|R|\) vertices represent selections of one of those \( n \) values. \( E \) is initialized such that the degree proportion of vertex \( k \in S \) is precisely \( \rho_1(k) \). When a vertex \( r \) joins set \( R \), with probability \( 1 - \alpha - \beta \) we add edge \((r, s)\) where \( s \) is drawn from \( S \) uniformly, with probability \( \alpha \) we add edge \((r, s)\) where \( s \) is drawn from \( S \) proportionally to its degree, and with probability \( \beta \) we add edge \((r, s)\) where \( s \) is drawn from \( S \) with probability \( \varphi_\mu(s) \).

We accordingly generalize our difference and continuous approximation solutions with the initial state:
\[
\hat{\rho}_t(k) = \prod_{s=2}^{t+1} \rho_1(k) \left( 1 - \frac{1 - \alpha}{s} \right) + \sum_{s' = 2}^{t+1} \left( \frac{1 - \alpha}{s'} \cdot \prod_{s = s' + 1}^{t+1} \left( 1 - \frac{1 - \alpha}{s} \right) \right) 
\]

(16)

\[
\hat{\rho}_t(k) = \frac{1}{n} + \left( \rho_1(k) - \frac{1}{n} \right) t^{\alpha - 1}, 
\]

(17)
respectively, and sum of squared errors in the continuous case

\[ \varepsilon(t, \gamma_1, \ldots, \gamma_n, \alpha, \beta) = \sum_k \left[ \left( 1 - \frac{\beta}{1 - \alpha} \right) \frac{1}{n} + \frac{\beta}{1 - \alpha} \varphi_{\mu}(k) \right. \\
\left. + \left( \rho_1(k) - \left[ 1 - \frac{\beta}{1 - \alpha} \right] \frac{1}{n} - \frac{\beta}{1 - \alpha} \varphi_{\mu}(k) \right) t^{\alpha - 1} - \gamma_k \right]^2 \]  

With the additional component, the error function of two variables (\(\alpha\) and \(\beta\)) becomes infeasible to minimize over both analytically. We derive only the minimization for fixed \(\alpha\), as solving for \(\alpha\) minimizing sum of squared errors is infeasible due to the presence of \(\alpha\) in exponential terms as well as polynomial terms and squared exponential terms and combinations thereof, and find that

\[ \beta = (1 - \alpha) \left[ 1 - \frac{(t^{\alpha - 1} - 1) \sum_k \varphi_{\mu}(k)(\varphi_{\mu}(k) - \rho_1(k)) + \sum_k \varphi_{\mu}(k)(\gamma_k - \rho_1(k))}{(t^{\alpha - 1} - 1) \sum_k \left( \frac{1}{n} - \varphi_{\mu}(k) \right)^2} \right] \]  

Proof. As before, we step through the differentiation of \(\varepsilon\), noting its convexity, and also the knowledge we have of the distribution functions: that integrating over their support (the rating indices)
yields 1.

\[
0 := \frac{\partial \varepsilon}{\partial \beta} = \sum_k 2 \left[ \left( 1 - \frac{\beta}{1 - \alpha} \right) \frac{1}{n} + \frac{\beta}{1 - \alpha} \varphi_{\mu}(k) + \left( \rho_1(k) - \left[ 1 - \frac{\beta}{1 - \alpha} \right] \frac{1}{n} - \frac{\beta}{1 - \alpha} \varphi_{\mu}(k) \right) t^{\alpha - 1} - \gamma_k \right] \\
\cdot \left[ - \frac{1}{1 - \alpha} \cdot \frac{1}{n} + \frac{\varphi_{\mu}(k)}{1 - \alpha} + \left( \frac{1}{1 - \alpha} \cdot \frac{1}{n} - \frac{\varphi_{\mu}(k)}{1 - \alpha} \right) t^{\alpha - 1} \right]
\]

\[
= \sum_k \left[ (1 - \alpha - \beta) \frac{1}{n} + \beta \varphi_{\mu}(k) + \left( [1 - \alpha] \rho_1(k) - [1 - \alpha - \beta] \frac{1}{n} - \beta \varphi_{\mu}(k) \right) t^{\alpha - 1} - (1 - \alpha) \gamma_k \right] \\
\cdot \left[ (t^{\alpha - 1} - 1) \left( \frac{1}{n} - \varphi_{\mu}(k) \right) \right] \text{cancelling}
\]

\[
= \sum_k \left[ \beta \left( \frac{1}{n} - \varphi_{\mu}(k) \right) (t^{\alpha - 1} - 1) + (1 - \alpha) \left( (t^{\alpha - 1} - 1) \left[ \rho_1(k) - \frac{1}{n} \right] + (\rho_1(k) - \gamma_k) \right) \right] \\
\cdot \left[ (t^{\alpha - 1} - 1) \left( \frac{1}{n} - \varphi_{\mu}(k) \right) \right]
\]

\[
\beta = -(1 - \alpha) \sum_k \left[ \left( t^{\alpha - 1} - 1 \right) \left( \rho_1(k) - \frac{1}{n} \right) + (\rho_1(k) - \gamma_k) \right] \cdot \left( \frac{1}{n} - \varphi_{\mu}(k) \right)
\]

\[
= (1 - \alpha) \sum_k \left[ \left( t^{\alpha - 1} - 1 \right) \left( \frac{1}{n} - \varphi_{\mu}(k) \right) + \varphi_{\mu}(k) - \rho_1(k) \right] \cdot \left( \frac{1}{n} - \varphi_{\mu}(k) \right)
\]

\[
= (1 - \alpha) \left[ 1 - \frac{\left( t^{\alpha - 1} - 1 \right) \sum_k (\rho_1(k) - \varphi_{\mu}(k)) \left( \frac{1}{n} - \varphi_{\mu}(k) \right) + \sum_k (\rho_1(k) - \gamma_k) \left( \frac{1}{n} - \varphi_{\mu}(k) \right)}{\left( t^{\alpha - 1} - 1 \right) \sum_k \left( \frac{1}{n} - \varphi_{\mu}(k) \right) \left( \frac{1}{n} - \varphi_{\mu}(k) \right)} \right]
\]

\[
= (1 - \alpha) \left[ 1 - \frac{\left( t^{\alpha - 1} - 1 \right) \sum_k \varphi_{\mu}(k) (\varphi_{\mu}(k) - \rho_1(k)) + \sum_k \varphi_{\mu}(k) (\gamma_k - \rho_1(k))}{\left( t^{\alpha - 1} - 1 \right) \sum_k \left( \frac{1}{n} - \varphi_{\mu}(k) \right) \left( \frac{1}{n} - \varphi_{\mu}(k) \right)} \right]
\]

recalling that \( \sum_k \varphi_{\mu}(k) = \sum_k \gamma_k = \sum_k \rho_1(k) = 1 \)

\[ \square \]

Note that we can also scale the probabilities by each other for a multiplicative effect; however, as this requires renormalization and interdependence between the proportion functions, we introduce only the transitions and their probabilities and will demonstrate simulation results.

**Definition 7** (Scaling pseudonormal preferential attachment rating). We define a scaling pseudonormal preferential attachment rating graph \( G_{\alpha, \beta}^{n, \mu} = (S, R, E) \) to be an undirected bipartite graph in which \( n = |S| \) vertices represents some value, e.g. ratings, and the other \( |R| \) vertices represent selections of one of those \( n \) values. When a vertex \( r \) joins set \( R \), we scale the following according to a discretized truncated normal centered at \( \mu \) and with variance 1: with probability \( 1 - \alpha \) we add edge \( (r, s) \) where \( s \) is drawn from \( S \) uniformly, and with probability \( \alpha \) we add edge \( (r, s) \) where \( s \) is
drawn from $S$ proportionally to its degree.

Our proportions then grow as

$$
\rho_t(k) = \begin{cases} 
(1 - \frac{1}{t}) \rho_{t-1}(k) + \frac{1}{t} & \text{with probability } \frac{\frac{1-\alpha}{n} + \alpha\rho_{t-1}(k)}{\sum_i \frac{1-\alpha}{n} + \alpha\rho_{t-1}(i)} \\
(1 - \frac{1}{t}) \rho_{t-1}(k) & \text{with probability } 1 - \frac{\frac{1-\alpha}{n} + \alpha\rho_{t-1}(k)}{\sum_i \frac{1-\alpha}{n} + \alpha\rho_{t-1}(i)} 
\end{cases}
$$

0.4 Experimental Results

0.4.1 Model simulations

We present the results of simulation of each of these models, coded in Julia [3]. The pseudonormal used to scale or bias sets the mean $\mu = 2$, and the initial states either include the mean as $\mu = 2$, or begin with attachments to 1. Due to space considerations we indicate the color mapping here: red $\rightarrow$ 1, blue $\rightarrow$ 2, green $\rightarrow$ 3, black $\rightarrow$ 4, and yellow $\rightarrow$ 5.
Figure 1: Simple preferential attachment rating graph degrees, tracked over time. Left: initialized with one attachment (i.e. simple), right: initialized with pseudonormally distributed attachments (i.e. delayed).
Figure 2: Pseudonormal preferential attachment rating graph degrees, tracked over time. Initialized with one attachment (i.e. simple pseudornormal).
Figure 3: Con't: Pseudonormal preferential attachment rating graph degrees, tracked over time. Initialized with one attachment (i.e. simple pseudonormal).
Figure 4: Pseudonormal preferential attachment rating graph degrees, tracked over time. Initialized with pseudonormally distributed degrees (i.e. delayed pseudonormal).
Figure 5: Con't: Pseudonormal preferential attachment rating graph degrees, tracked over time. Initialized with pseudonormally distributed degrees (i.e. delayed pseudornormal).
Figure 6: Scaled pseudonormal preferential attachment rating graph degrees, tracked over time. Left: initialized with one attachment (i.e. simple), right: initialized with pseudonormally distributed attachments (i.e. delayed).
Figure 7: Simple amnesic preferential attachment rating graph degrees, tracked over time, with history length 2. Left: initialized with one attachment (i.e. simple), right: initialized with pseudonormally distributed attachments (i.e. delayed).
Figure 8: Pseudonormal preferential attachment rating graph degrees, tracked over time, with history length 2. Initialized with one attachment (i.e. simple pseudonormal).
Figure 9: Con't: Pseudonormal preferential attachment rating graph degrees, tracked over time, with history length 2. Initialized with one attachment (i.e. simple pseudonormal).
Figure 10: Pseudonormal preferential attachment rating graph degrees, tracked over time, with history length 2. Initialized with pseudonormally distributed degrees (i.e. delayed pseudornormal).
Figure 11: Con’t: Pseudonormal preferential attachment rating graph degrees, tracked over time, with history length 2. Initialized with pseudonormally distributed degrees (i.e. delayed pseudonormal).
Figure 12: Scaled pseudonormal preferential attachment rating graph degrees, tracked over time, with history length 2. Left: initialized with one attachment (i.e. simple), right: initialized with pseudonormally distributed attachments (i.e. delayed).
Figure 13: Simple preferential attachment rating graph degrees, tracked over time, with history length 5. Left: initialized with one attachment (i.e. simple), right: initialized with pseudonormally distributed attachments (i.e. delayed).
Figure 14: Pseudonormal preferential attachment rating graph degrees, tracked over time, with history length 5. Initialized with one attachment (i.e. simple pseudonormal).
Figure 15: Con’t: Pseudonormal preferential attachment rating graph degrees, tracked over time, with history length 5. Initialized with one attachment (i.e. simple pseudonormal).
Figure 16: Pseudonormal preferential attachment rating graph degrees, tracked over time, with history length 5. Initialized with pseudonormally distributed degrees (i.e. delayed pseudornormal).
Figure 17: Con’t: Pseudonormal preferential attachment rating graph degrees, tracked over time, with history length 5. Initialized with pseudonormally distributed degrees (i.e. delayed pseudonormal).
Figure 18: Scaled pseudonormal preferential attachment rating graph degrees, tracked over time, with history length 5. Left: initialized with one attachment (i.e. simple), right: initialized with pseudonormally distributed attachments (i.e. delayed).
We make several observations in simulation of these models:

• not many crossovers tend to occur in the simple preferential attachment case: preferential attachment is in some sense order preserving, in both the simple and simple delayed cases;

• introducing some (constant) bias in the form of a normal to each entering rating indeed results in a somewhat bimodal distribution when vying against preferential attachment;

• while the effect of history is difficult to describe analytically or even verbally, it appears to have a significant effect, especially in conjunction with pseudonormality.

0.4.2 The Yelp dataset

We consider the Yelp Academic Dataset [7], available on request from Yelp, which provides three tables of data for the 250 closest businesses to 30 universities:

1. data on the businesses, each of which includes locale information and categories,

2. data on the reviewers, each of which includes number of reviews, average rating, and some ratings of qualities of qualities (useful, funny, cool) of their reviews as provided by other reviewers, and

3. data on the reviews, each of which includes reviewer, date, an integral 1–5 rating, and some ratings of qualities (useful, funny, cool) as provided by other reviewers.

Not every business, however, is represented in the set of reviews, nor every reviewer in the set of reviews. A complete, cross-referenced picture is hence difficult to form. Moreover, it is difficult to form a sampling of the same scale as provided by the dataset, so while it is possible the dataset was constructed with certain properties, we also operate under the assumption that Yelp has not predoctored the academic dataset.
Figure 19: Simple preferential attachment rating graph degrees, over time. Left: initialized with one attachment (i.e. simple), right: initialized with pseudonormally distributed attachments (i.e. delayed).
Figure 20: Pseudonormal preferential attachment rating graph degrees, over time. Initialized with one attachment (i.e. simple pseudonormal).
Figure 21: Con’t: Pseudonormal preferential attachment rating graph degrees, over time. Initialized with one attachment (i.e. simple pseudonormal).
Figure 22: Pseudonormal preferential attachment rating graph degrees, over time. Initialized with pseudonormally distributed degrees (i.e. delayed pseudonormal).
Figure 23: Con’t: Pseudonormal preferential attachment rating graph degrees, over time. Initialized with pseudonormally distributed degrees (i.e. delayed pseudonormal).
Figure 24: Scaled pseudonormal preferential attachment rating graph degrees, over time. Left: initialized with one attachment (i.e. simple), right: initialized with pseudonormally distributed attachments (i.e. delayed).
Figure 25: Simple amnesic preferential attachment rating graph degrees, over time, with history length 2. Left: initialized with one attachment (i.e. simple), right: initialized with pseudonormally distributed attachments (i.e. delayed).
Figure 26: Pseudonormal preferential attachment rating graph degrees, over time, with history length 2. Initialized with one attachment (i.e. simple pseudornormal).
Figure 27: Con’t: Pseudonormal preferential attachment rating graph degrees, over time, with history length 2. Initialized with one attachment (i.e. simple pseudornormal).
Figure 28: Pseudonormal preferential attachment rating graph degrees, over time, with history length 2. Initialized with pseudonormally distributed degrees (i.e. delayed pseudornormal).
Figure 29: Con’t: Pseudonormal preferential attachment rating graph degrees, over time, with history length 2. Initialized with pseudonormally distributed degrees (i.e. delayed pseudornormal).
Figure 30: Scaled pseudonormal preferential attachment rating graph degrees, over time, with history length 2. Left: initialized with one attachment (i.e. simple), right: initialized with pseudonormally distributed attachments (i.e. delayed).
Figure 31: Simple preferential attachment rating graph degrees, over time, with history length 5. Left: initialized with one attachment (i.e. simple), right: initialized with pseudonormally distributed attachments (i.e. delayed).
Figure 32: Pseudonormal preferential attachment rating graph degrees, over time, with history length 5. Initialized with one attachment (i.e. simple pseudornormal).
Figure 33: Con’t: Pseudonormal preferential attachment rating graph degrees, over time, with history length 5. Initialized with one attachment (i.e. simple pseudonormal).
Figure 34: Pseudonormal preferential attachment rating graph degrees, over time, with history length 5. Initialized with pseudonormally distributed degrees (i.e. delayed pseudornormal).
Figure 35: Con’t: Pseudonormal preferential attachment rating graph degrees, over time, with history length 5. Initialized with pseudonormally distributed degrees (i.e. delayed pseudornormal).
Figure 36: Scaled pseudonormal preferential attachment rating graph degrees, over time, with history length 5. Left: initialized with one attachment (i.e. simple), right: initialized with pseudonormally distributed attachments (i.e. delayed).
When one wishes to review a business, one must navigate to the business’ page (where is prominently displayed the “overall rating” of a business, which is simply the average across all reviews, rounded to the nearest half-star). On this page, without scrolling, one can see location, pictures, “review highlights” (phrases from reviews determined to be characteristic of the business), hours of operation, and price range, as shown in Fig. 37. One then proceeds to the review form itself. Only members are allowed to post reviews, intended to help prevent the formation of phony reviews. A sidebar displays other recent reviews.

**Overview**

It is therefore not unexpected for Yelp reviewers to have been influenced by previous reviews at the time of writing their own reviews: not only is the overall average one of the first things a prospective reviewer sees, but a prospective reviewer is also heavily exposed to other reviews. We therefore focus our attention on only the social aspect of reviews and whether reviewers’ behaviors exhibit components of preferential attachment, rather than on attempting to model the system fully (the latter of which might include accounting for business-specific characteristics such as locale, hours of operation, or price).

If that is the case, then we would expect to see ratings to approach each other over all businesses over time. More precisely, we would expect any of several cases to arise:

- aggregating all businesses, the mean of the set of reviews over slices in time should stabilize;
- aggregating all businesses, the set of variances of the reviews over slices in time should decrease,
- aggregating all businesses, the mean of the set of reviews cumulatively over time should stabilize, and
- aggregating all businesses, the set of variance of the reviews cumulatively over time should decrease

where time might be quantized per rating (sub-day), day, month, etc. We will quantize on per-day, per-month, and per-quarter (three months) scales (the per-year scale generates insufficient data points in many cases).
(a) Business page. Note the rating (and number of ratings) placed directly under business name and “review highlights” placed under the map and images. Further below on the page (not shown) are “recommended reviews.” At the top of the page are also quicklinks to user-specific functions, including for social networking. At the side are also links to “People also viewed” for presumably related businesses.

(b) Reviewing page. Note sample reviews on right hand column—these are the “recommended reviews” not shown in the business page image.

Figure 37: Clips of Yelp pages associated with a sample business.
(a) Means of pooled set of businesses and quantized by day.
(b) Means of pooled set of businesses cumulatively and quantized by day.

(c) Mean of pooled set of business variances quantized by day.
(d) Mean of pooled set of business variances cumulatively and quantized by day.

Figure 38: Plots of pooled data by day.
(a) Means of pooled set of businesses and quantized by month.

(b) Means of pooled set of businesses cumulatively and quantized by month.

(c) Mean of pooled set of business variances quantized by month.

(d) Mean of pooled set of business variances cumulatively and quantized by month.

Figure 39: Plots of pooled data by month.
(a) Means of pooled set of businesses and quantized by quarter.

(b) Means of pooled set of businesses cumulatively and quantized by quarter.

(c) Mean of pooled set of business variances quantized by quarter.

(d) Mean of pooled set of business variances cumulatively and quantized by quarter.

Figure 40: Plots of pooled data by quarter.
As can be seen from the timeslice-wise mean plots in Figs. 38a, 39a, and 40a, there is no apparent funneling. In the cumulative mean plots in Figs. 38b, 39b, and 40b, although the the means do appear to be stabilizing, this may simply be an effect of the Law of Large Numbers. Moreover, the variance plots in Figs. 38c, 39c, and 40c, 38d, 39d, and 40d all show a steady upwards trend in variance, i.e. the ratings in businesses pooled with respect to time relative to each’s beginning is increasing in variance! This is hardly reassuring.

Pooling together businesses has a very rough granularity, however, so we next consider the businesses individually. Due to the great uncertainty in causality of considering cumulative means, and the infeasibility of looking through all the businesses by hand, we use a heuristic to prefilter whether some businesses have a definite upwards trend in variance per interval, and whether some businesses have a definite downwards trend. Our heuristic is to fit a simple linear regression to the variances against time:

$$\sigma_i^2 = \beta_0 + \beta_2 t_i$$

If our model fits with low p-value (we select significance level $\alpha < .25$) for $\beta_1$, then we should be able to reject the hypothesis that there is no upwards or downwards trend temporally. Note that we are unconcerned with the actual fit of the linear model, and only whether or not there is a monotonic trend.

After prefiltering thus, it becomes much easier to cull through businesses manually. Rather than all businesses lacking trend, it appears that some businesses have a very increase in variance over time, while others the reverse, as shown in Fig. 41. This suggests that some businesses at least have more of a potential to display preferential attachment than others.

Additionally, considering the final rating distributions of different businesses also reveals several different shapes, as shown in Fig. 42. While the approximately normal distribution is the most common, the other shapes appear very often as well, suggesting that there may be some intrinsic quality in businesses that correlates with normality or lack thereof, although that is beyond the scope of this paper. We merely note here that these shapes are not unlike those presented in the model simulations above.
Figure 41: Variances plotted against time (quantized over quarters). The number next to business name indicates the $p$-value of the slope.
Figure 42: Final degree distributions for several restaurants with representative distribution shapes, with fitted normal (mean about the assigned rating, which is simply sample mean rounded to the nearest half-star) to contrast.
0.5 Discussion and Further Research

We conjecture the role of preferential attachment in rating and recommendation systems, hypothesizing that the social nature of many rating and recommendation systems introduces such preferential attachment. We present several models to capture different variants of preferential attachment, particularly, different combinations, mostly linear, some multiplicative, of completely random attachment, preferential attachment based on different history lengths, and normal about some mean, and observe the different degree distributions that arise accordingly, both analytically in some simpler cases, and in simulation.

Experimentally, we would like to test these models on more recommendation systems, as well as the businesses within Yelp with finer granularity. As suggested earlier, it seems that there may be underlying characteristics of businesses that make them more prone to be rated with preferential attachment, versus according to personal bias: we wish to identify these businesses and why. Although the author makes no claims as to familiarity with different businesses, one observation was that restaurants such as Chipotle or Flat Patties, which serve more “general” or “all-purpose” food, were found not infrequently to have decreasing variance, while restaurants like Rialto or Miracle of Science Bar and Grill, in some sense more “niche” restaurants, did not. A sample hypothesis would thereby be that because reviewers who go to general purpose restaurants understand that they have a large range, be more willing to act based on the judgment of other reviewers, and to the contrary for reviewers who go to niche restaurants. Testing this would require examining the user data for personal biases, as well as simply each user’s personal rating distribution, depending what other reviews and ratings they may have made.

Acknowledgements

Many thanks to my advisor Prof. Yaron Singer for his guidance and encouragement, and to my parents and sister for their love and support, and to Keno Fischer for food and silverware.
Bibliography


Appendices
.1 Code

module pref_attach

using JSON
using DataFrames
using Dates
using Iterators
using Distributions

Str = Union(ASCIIString, UTF8String)

type PrefAttachRankings
  n::Int # number of rankings (e.g. 5)
  num_ranks::Array{Int,1} # array with number of each ranking
  history_len::Int # length of history to save
  i::Int # position within history
  history::Array{Int,1} # history of ranks
  alpha::Float64 # weight of preferential attachment
  beta::Float64 # weight of truncated normal bias
  mu::Float64 # mean of truncated normal bias

  PrefAttachRankings(n,
    num_ranks,
    history_len,
    alpha,beta) = new(n,
      num_ranks,
      history_len, 0,
      Int[],
      alpha, beta, 0)

  PrefAttachRankings(n,
    history_len,
    alpha,beta) = new(n,
      0*vec(Array(Int,n,1)),
      history_len, 0,
      Int[],
      alpha, beta, 0)

  PrefAttachRankings(n,
    alpha,beta) = new(n,
      0*vec(Array(Int,n,1)),
      -1, 0,
      Int[],
      alpha, beta, 0)

end

function attach(r::PrefAttachRankings, k::Int)
  @assert 1 <= k && k <= r.n
  r.num_ranks[k] += 1
  return r
end
if r.history_len > 0
    r.i += 1
    if length(r.history) < r.history_len
        push!(r.history, k)
    else
        if r.i >= r.history_len
            r.i = 1
        end
        r.history[r.i] = k
    end
end

r

function pdf_trunc(k, μ)
    cdf(Truncated(Normal(μ), .5, 5.5), k+.5) -
    cdf(Truncated(Normal(μ), .5, 5.5), k-.5)
end

comp_unif(r::PrefAttachRankings) = 1/r.n
comp_prop(r::PrefAttachRankings, k::Int) = r.num_ranks[k]/sum(r.num_ranks)
comp_hist(r::PrefAttachRankings, k::Int) = count(x->x==k, r.history)/length(r.history)
comp_norm(r::PrefAttachRankings, k::Int) = pdf_trunc(k, r.μ)

function simple(r::PrefAttachRankings, k::Int)
    dot([1-r.alpha, r.alpha], [comp_unif(r), comp_prop(r, k)])
end

function simple_hist(r::PrefAttachRankings, k::Int)
    dot([1-r.alpha, r.alpha], [comp_unif(r), comp_hist(r, k)])
end

function pseudonorm(r::PrefAttachRankings, k::Int)
    dot([1-r.alpha-r.β, r.alpha, r.β], [comp_unif(r),
        comp_prop(r, k),
        comp_norm(r, k)])
end

function pseudonorm_hist(r::PrefAttachRankings, k::Int)
    dot([1-r.alpha-r.β, r.alpha, r.β], [comp_unif(r),
        comp_hist(r, k),
        comp_norm(r, k)])
end

function scaled_pseudonorm(r::PrefAttachRankings, k::Int)
    tot = sum([simple(r, i) * pdf_trunc(i, r.μ) for i=1:r.n])
    simple(r, k)*pdf_trunc(k, r.μ)/tot
end
function scaled_pseudonorm_hist(r::PrefAttachRankings, k::Int)
    tot = sum([simple_hist(r, i) * pdf_trunc(i, r.mu) for i=1:r.n])
    simple_hist(r, k) * pdf_trunc(k, r.mu) / tot
end

function prob_attach(r::PrefAttachRankings, k::Int, prob_fn::Function)
    prob_fn(r, k)
end

function find_attach(r::PrefAttachRankings, prob_fn::Function)
    pdf = Float64[prob_attach(r, k, prob_fn) for k=1:r.n]
    p = rand()
    findfirst(x -> x > p, cumsum(pdf))
end

function simulate(r::PrefAttachRankings, prob_fn::Function, steps::Int)
    snapshots = Array(Int, steps+1, r.n)
    snapshots[1, :] = deepcopy(r.num_ranks[:])
    for i in 1:steps
        k = find_attach(r, prob_fn)
        attach(r, k)
        snapshots[i+1, :] = deepcopy(r.num_ranks[:])
    end
    snapshots
end

function sse(expected, actual)
    sum((expected - actual).^2)
end

module make_plots
    using DataFrames
    using Dates
    using HDF5, JLD
    using Distributions
    using Gadfly
    using GLM, RDatasets

    Str = Union(ASCIIString, UTF8String)
    include("pref_attach.jl")

    function load_data()
        dat_biz = load("yelp.jld", "dat_biz")
    end
end
dat_rev = load("yelp.jld", "dat_rev")
(dat_biz, dat_rev)
end

function get_good_biz_ids(dat_biz, dat_rev)
    map = Dict{Str, Int}()
    for i in 1:length(dat_rev[:, :Biz])
        if dat_rev[i, :Biz] in keys(map)
            map[dat_rev[i, :Biz]] += 1
        elseif dat_rev[i, :Biz] in dat_biz[:, :Id]
            map[dat_rev[i, :Biz]] = 1
        end
    end
    keys(filter!((k,v)->v > 100, map))
end

function reorg_reviews(good_biz_ids, dat_rev)
    maps = Dict{Str, DataFrame}()
    for id in good_biz_ids
        revs = dat_rev[dat_rev[:, :Biz] .== id, :]
        sort!(revs, cols=(:Date))
        num_days = Int[(d-revs[1,:Date]).value for d in revs[:, :Date]]
        maps[id] = DataFrame(NumPeriods=num_days, Stars=revs[:, :Stars],
                              Date=revs[:, :Date], User=revs[:, :User])
    end
    maps
end

function gather_by_day(maps)
    maps_by_day = Dict{Str, DataFrame}()
    for (id, revs) in maps
        nodup = unique(revs[:, :Date])
        _num_days = Array(Int, length(nodup))
        _stars = Array(Float64, length(nodup))
        _date = Array(Date, length(nodup))
        (i, j) = (1, 1)
        while j <= length(revs[:, :Date])
            d = revs[j, :Date]
            subset = revs[revs[:, :Date] .== d, :]
            _num_days[i] = subset[1, :NumPeriods]
            _stars[i] = mean(subset[:, :Stars])
            _date[i] = subset[1, :Date]
            j += length(subset[:, :Date])
        end
        maps_by_day[id] = DataFrame(NumPeriods=_num_days, Stars=_stars,
                                     Date=_date)
    end
end
function reorg_reviews_by_month(good_biz_ids, dat_rev)
    maps = Dict{Str, DataFrame}()
    for id in good_biz_ids
        revs = dat_rev[dat_rev[:,Biz] .== id, :]
        sort!(revs, cols=(:Date))
        dates = Date[firstdayofmonth(d) for d in revs[:,Date]]
        num_months= Int[(12*(Year(d)-Year(dates[1])).value + (Month(d) - Month(dates[1])).value) for d in dates]
        maps[id] = DataFrame(NumPeriods=num_months, Stars=revs[:,Stars], Date=dates, User=revs[:,User])
    end
    maps
end

function reorg_reviews_by_quarter(good_biz_ids, dat_rev)
    maps = Dict{Str, DataFrame}()
    for id in good_biz_ids
        revs = dat_rev[dat_rev[:,Biz] .== id, :]
        sort!(revs, cols=(:Date))
        dates = Date[firstdayofquarter(d) for d in revs[:,Date]]
        num_quarters = Int[int((12*(Year(d)-Year(dates[1])).value + (Month(d) - Month(dates[1])).value)/3)
        for d in dates]
        maps[id] = DataFrame(NumPeriods=num_quarters, Stars=revs[:,Stars], Date=dates, User=revs[:,User])
    end
    maps
end

function reorg_reviews_by_year(good_biz_ids, dat_rev)
    maps = Dict{Str, DataFrame}()
    for id in good_biz_ids
        revs = dat_rev[dat_rev[:,Biz] .== id, :]
        sort!(revs, cols=(:Date))
        dates = Date[Date(year(d)) for d in revs[:,Date]]
        num_days = Int[(Year(d) - Year(dates[1])).value for d in dates]
        maps[id] = DataFrame(NumPeriods=num_days, Stars=revs[:,Stars], Date=dates, User=revs[:,User])
    end
    maps
end

function compress!(maps)
    for (id, revs) in maps
        (i,j) = (1, 2)
    end
end
weight = ones(Int, length(revs[:, :Date]))
stars = Array(Array{Int,1}, length(revs[:, :Date]))
stars[i] = Int[revs[i, :Stars]]
while j <= length(revs[:, :Date])
    stars[j] = Int[]
    if revs[j, :Date] == revs[i, :Date]
        revs[i, :Stars] += revs[j, :Stars]
        push!(stars[i], revs[j, :Stars])
        weight[i] += 1
        revs[j, :Stars] = NA
    else
        i = j
        stars[i] = Int[revs[j, :Stars]]
    end
    j += 1
end
(revs[:,NumRevs], revs[:,StarSet]) = (weight, stars)
deletrows!(revs, find(isna(revs[:, :Stars])))

function get_mean_rev_time(maps)
    ndays = int(mean(Int[v[length(v[:,NumPeriods]), :NumPeriods] for (k,v) in maps]))-1
    all_vars = Array{Float64,1}[Float64[] for ind=1:(ndays+1)]
    for (k,v) in maps
        print(".")
        tmp = Array{Int,1}[Int[] for ind=1:(ndays+1)]
        # ratings of business, per day
        for ind in 1:length(v[:,NumPeriods])
            if v[ind,:NumPeriods] <= ndays
                num_by_day[v[ind,:NumPeriods]+1] += 1
                sum_by_day[v[ind,:NumPeriods]+1] += v[ind,:Stars]
                push!(all_vars[v[ind,:NumPeriods]+1], v[ind,:Stars])
            end
        end
        for ind in 1:length(v[:,NumPeriods])
            if v[ind,:NumPeriods] <= ndays && length(tmp[v[ind,:NumPeriods]+1]) > 1
                push!(all_vars[v[ind,:NumPeriods]+1], var(tmp[v[ind,:NumPeriods]+1]))
            end
        end
        vars = zeros(Float64,ndays+1)
    end
end
for (i,v) in enumerate(all_vars)
    if length(v) > 0
        vars[i] = mean(v)
    else
        vars[i] = 0
    end
end

(num_by_day, sum_by_day, vars)
end

function make_biz_plots(dat_biz, maps, good_biz_ids)
    for b in good_biz_ids
        mu_assigned = dat_biz[dat_biz[:Id] .== b, :Stars][1]
        biz_name = dat_biz[dat_biz[:Id] .== b, :Name][1]
        revs = maps[b]
        p = Gadfly.plot(
            layer(x -> pdf(Truncated(Normal(mu_assigned), 0.5, 5.5), x), 0.5, 5.5, Theme(default_color=color("red"))),
            layer(x=1:5, y=[count(x->x==i, revs[:, :Stars])/length(revs[:, :Stars]) for i=1:5], Geom.bar),
            Guide.title("$biz_name"))
        draw(PNG("biz_plots/biz_plot_$b.png", 8inch, 6inch), p)
    end
end

function get_pooled_means_and_vars(maps)
    ndays = int(mean(Int[v[length(v[:, :NumPeriods]), :NumPeriods] for (k,v) in maps])+.5)-1
    (num_by_day, sum_by_day) = (zeros(Float64, ndays+1), zeros(Float64, ndays+1))
    all_vars_per = Array{Float64,1}[Float64[] for ind=1:(ndays+1)]
    all_vars_cum = Array{Float64,1}[Float64[] for ind=1:(ndays+1)]
    for (k,v) in maps
        for ind in 1:length(v[:, :NumPeriods])
            if v[ind, :NumPeriods] <= ndays
                if length(v[ind, :StarSet]) > 1
                    push!(all_vars_per[v[ind, :NumPeriods]+1], var(v[ind, :StarSet]))
                    push!(all_vars_cum[v[ind, :NumPeriods]+1], var(vcat(v[1:ind, :StarSet],...)))
                end
            end
        end
    end
    vars_per = zeros(Float64, ndays+1)
    vars_cum = zeros(Float64, ndays+1)
for (i, (v1, v2)) in enumerate(zip(all_vars_per, all_vars_cum))
    vars_per[i] = length(v1) > 0 ? mean(v1) : 0
    vars_cum[i] = length(v2) > 0 ? mean(v2) : 0
end

(sum_cum, num_cum) = (Array(Int32, ndays+1), Array(Int32, ndays+1))
(sum_cum[1], num_cum[1]) = (sum_by_day[1], num_by_day[1])
for i in 2:(ndays+1)
    sum_cum[i] = sum_cum[i-1] + sum_by_day[i]
    num_cum[i] = num_cum[i-1] + num_by_day[i]
end

([x/n for (x,n) in zip(sum_by_day, num_by_day)],
 [x/n for (x,n) in zip(sum_cum, num_cum)],
 vars_per, vars_cum,
 num_by_day, sum_by_day)
end

function make_compressed(good_biz_ids, dat_rev)
    maps_day = reorg_reviews(good_biz_ids, dat_rev)
    compress!(maps_day)
    maps_month = reorg_reviews_by_month(good_biz_ids, dat_rev)
    compress!(maps_month)
    maps_quarter = reorg_reviews_by_quarter(good_biz_ids, dat_rev)
    compress!(maps_quarter)
    (maps_day, maps_month, maps_quarter)
end

function plot_pooled(maps_day, maps_month, maps_quarter)
    # by day
    (means_day, means_day_cum, vars_day, vars_day_cum, _, _) =
        get_pooled_means_and_vars(maps_day)
    p = plot(x=0:(length(means_day)-1), y=means_day,
             Guide.xlabel("Num Days"),
             Guide.ylabel("Mean"),
             Geom.point)
    draw(PNG("final_images/pooled_means_day.png", 8inch, 6inch), p)
    p = plot(x=0:(length(means_day_cum)-1), y=means_day_cum,
             Guide.xlabel("Num Days"),
             Guide.ylabel("Cumulative Mean"),
             Geom.point)
    draw(PNG("final_images/pooled_means_day_cum.png", 8inch, 6inch), p)
    p = plot(x=0:(length(vars_day)-1), y=vars_day,
             Guide.xlabel("Num Days"),
             Guide.ylabel("Average Variance"),
             Geom.point)
    draw(PNG("final_images/pooled_vars_day.png", 8inch, 6inch), p)
    p = plot(x=0:(length(vars_day_cum)-1), y=vars_day_cum,

70
Guidex.xlabel("Num Days"),
Guidex.ylabel("Average Cumulative Variance"),
Geom.point

draw(PNG("final_images/pooled_vars_day_cum.png", 8inch, 6inch), p)

# by month
(means_month, means_month_cum, vars_month, vars_month_cum, _, _) =
get_pooled_means_and_vars(maps_month)
p = plot(x=0:(length(means_month)-1), y=means_month,
    Guidex.xlabel("Num Months"),
    Guidex.ylabel("Mean"),
    Geom.point)
draw(PNG("final_images/pooled_means_month.png", 8inch, 6inch), p)
p = plot(x=0:(length(means_month_cum)-1), y=means_month_cum,
    Guidex.xlabel("Num Months"),
    Guidex.ylabel("Cumulative Mean"),
    Geom.point)
draw(PNG("final_images/pooled_means_month_cum.png", 8inch, 6inch), p)
p = plot(x=0:(length(vars_month)-1), y=vars_month,
    Guidex.xlabel("Num Months"),
    Guidex.ylabel("Average Variance"),
    Geom.point)
draw(PNG("final_images/pooled_vars_month.png", 8inch, 6inch), p)
p = plot(x=0:(length(vars_month_cum)-1), y=vars_month_cum,
    Guidex.xlabel("Num Months"),
    Guidex.ylabel("Average Cumulative Variance"),
    Geom.point)
draw(PNG("final_images/pooled_vars_month_cum.png", 8inch, 6inch), p)

# by year
(means_quarter, means_quarter_cum, vars_quarter, vars_quarter_cum, _, _) =
get_pooled_means_and_vars(maps_quarter)
p = plot(x=0:(length(means_quarter)-1), y=means_quarter,
    Guidex.xlabel("Num Quarters"),
    Guidex.ylabel("Mean"),
    Geom.point)
draw(PNG("final_images/pooled_means_quarter.png", 8inch, 6inch), p)
p = plot(x=0:(length(means_quarter_cum)-1), y=means_quarter_cum,
    Guidex.xlabel("Num Quarters"),
    Guidex.ylabel("Cumulative Mean"),
    Geom.point)
draw(PNG("final_images/pooled_means_quarter_cum.png", 8inch, 6inch), p)
p = plot(x=0:(length(vars_quarter)-1), y=vars_quarter,
    Guidex.xlabel("Num Quarters"),
    Guidex.ylabel("Average Variance"),
    Geom.point)
draw(PNG("final_images/pooled_vars_quarter.png", 8inch, 6inch), p)
p = plot(x=0:(length(vars_quarter_cum)-1), y=vars_quarter_cum,
Guide.xlabel("Num Quarters"),
Guide.ylabel("Average Cumulative Variance"),
Geom.point)
draw(PNG("final_images/pooled_vars_quarter_cum.png", 8inch, 6inch), p)
end

function get_good_and_bad_biz(maps, dat_biz, good_biz_ids, name_ident)
    (good, bad) = (Float64,Float64,Str)[], (Float64,Float64,Str)[]
    for (i, id) in enumerate(good_biz_ids)
        revs = maps[id]
        (_, _, vars, vars_cum, _, _) =
            make_plots.get_pooled_means_and_vars([id=>revs]);
        mask = vars .== 0
        if length(vars[!mask]) > 5
            xdat = (0:(length(vars)-1))[!mask]
            ydat = vars[!mask]
            correlation = cor(xdat, ydat)
            model = lm(Y ~ X, DataFrame(X=xdat, Y=ydat))
            (pval, slope) =
                (coeftable(model).mat[:,coeftable(model).pvalcol][2],
                 coef(model)[2])
            biz_name = dat_biz[dat_biz[:Id] .== id, :Name][1]
            p = plot(layer(x=xdat, y=ydat, Theme(default_color=color("blue")), Geom.point),
                    layer(x=xdat, y=dot(coef(model), [1, k]) for k in xdat,
                        Geom.line, Theme(default_color=color("red"))),
                    Guide.title("$biz_name: $pval"))
            if pval < .2
                if slope < 0
                    push!(good, (pval, slope, biz_name))
                    draw(PNG("scratch_images/good_\$name_ident\_$i.png", 8inch, 6inch), p)
                else
                    push!(bad, (pval, slope, biz_name))
                    draw(PNG("scratch_images/bad_\$name_ident\_$i.png", 8inch, 6inch), p)
                end
            end
        end
    end
end

function plot_simulations()
    alphas = [0, .2, .8, 1]
    betas = [0, .2, .8, 1]
    fns = [pref_attach.simple, pref_attach.simple_hist,
            pref_attach.pseudonorm, pref_attach.pseudonorm_hist,
            pref_attach.scaled_pseudonorm, pref_attach.scaled_pseudonorm_hist]
    init = Array{Int,1}[[2, 0, 0, 0, 0],
    72
[[floor(100*pref_attach.pdf_trunc(i, 2)) for i=1:4],
  100 - sum([floor(100*pref_attach.pdf_trunc(i, 2))
             for i=1:4])]]

hist_len = [0, 2, 5]
tmp = [[floor(5*pref_attach.pdf_trunc(i, 2)) for i=1:4], 5 - sum([floor(5*pref_attach.pdf_trunc(i, 2))
             for i=1:4])]

tmp2 = Int[]
for (i,v) in enumerate(tmp) for x in 1:v push!(tmp2, i) end end

history = Array{Int,1}[[], [1,1], shuffle!(tmp2)]
nsteps = 98
for (ai, a) in enumerate(alphas)
    for (h, hist) in zip(hist_len, history)
        if h == 0
            for (fname, f) in zip(["Simple", "Scaled_Pseudonormal"], [fns[i] for i in [1, 5]])
                for (si, state) in enumerate(init)
                    r = pref_attach.PrefAttachRankings(5, deepcopy(state), h, a, 0)
                    snapshots = pref_attach.simulate(r, f, nsteps)
                    d = plot(layer(x=1:(nsteps+1), y=snapshots[:,1],
                                Theme(default_color=color("red")), Geom.line),
                             layer(x=1:(nsteps+1), y=snapshots[:,2],
                                Theme(default_color=color("blue")), Geom.line),
                             layer(x=1:(nsteps+1), y=snapshots[:,3],
                                Theme(default_color=color("green")), Geom.line),
                             layer(x=1:(nsteps+1), y=snapshots[:,4],
                                Theme(default_color=color("black")), Geom.line),
                             layer(x=1:(nsteps+1), y=snapshots[:,5],
                                Theme(default_color=color("yellow")), Geom.line),
                             Guide.title("$fname: alpha=$a`))
                    draw(PNG("final_images/$fname\_ai\_si.png", 8inch, 6inch), d)
                    d = plot(x=0:5, y=[0, r.num_ranks],
                             Geom.bar, Guide.title("$fname (alpha=$a) Final Distribution`))
                    draw(PNG("final_images/$fname\_fin\_ai\_si.png", 8inch, 6inch), d)
            end
        end
    end
end

for (bi, b) in enumerate(betas)
    if a + b <= 1
        for (si, state) in enumerate(init)
            r = pref_attach.PrefAttachRankings(5, deepcopy(state), h, a, b)
            snapshots = pref_attach.simulate(r, fns[3], nsteps)
            d = plot(layer(x=1:(nsteps+1), y=snapshots[:,1],
                                Theme(default_color=color("red")), Geom.line),
                                 layer(x=1:(nsteps+1), y=snapshots[:,2],
                                Theme(default_color=color("blue")), Geom.line),
                                 layer(x=1:(nsteps+1), y=snapshots[:,3],
                                Theme(default_color=color("green")), Geom.line),
                                 layer(x=1:(nsteps+1), y=snapshots[:,4],
                                Theme(default_color=color("black")), Geom.line),
                                 layer(x=1:(nsteps+1), y=snapshots[:,5],
                                     73
Theme(default_color=color("yellow")), Geom.line),
Guide.title("Pseudonormal (alpha=$a, beta=$b)"))
draw(PNG("final_images/Pseudonormal\$_ai\$_bi\$_si.png",
8inch, 6inch), d)
d = plot(x=0:5, y=[0, r.num_ranks],
Geom.bar, Guide.title("Pseudonormal (alpha=$a, beta=$b) Final Distribution"))
draw(PNG("final_images/Pseudonormal\_fin\$_ai\$_bi\$_si.png", 8inch, 6inch), d)
end
end
end
else
for (fname, f) in zip(["Amnesic_Simple", "Amnesic_Scaled_Pseudonormal"], [fns[i] for i in [2, 6]])
for (si, state) in enumerate(init)
r = pref_attach.PrefAttachRankings(5, deepcopy(state), h, a, 0)
r.history = deepcopy(hist)
snapshots = pref_attach.simulate(r, f, nsteps)
d = plot(layer(x=1:(nsteps+1), y=snapshots[:,1],
  Theme(default_color=color("red")), Geom.line),
layer(x=1:(nsteps+1), y=snapshots[:,2],
  Theme(default_color=color("blue")), Geom.line),
layer(x=1:(nsteps+1), y=snapshots[:,3],
  Theme(default_color=color("green")), Geom.line),
layer(x=1:(nsteps+1), y=snapshots[:,4],
  Theme(default_color=color("black")), Geom.line),
layer(x=1:(nsteps+1), y=snapshots[:,5],
  Theme(default_color=color("yellow")), Geom.line),
Guide.title("$fname: alpha=$a"))
draw(PNG("final_images/$fname\$_h\$_ai\$_si.png", 8inch, 6inch), d)
d = plot(x=0:5, y=[0, r.num_ranks],
  Geom.bar, Guide.title("$fname (hist=$h, alpha=$a) Final Distribution"))
draw(PNG("final_images/$fname\_fin\$_h\$_ai\$_si.png", 8inch, 6inch), d)
end
for (bi, b) in enumerate(betas)
if a + b <= 1
for (si, state) in enumerate(init)
r = pref_attach.PrefAttachRankings(5, deepcopy(state), h, a, b)
r.history = deepcopy(hist)
snapshots = pref_attach.simulate(r, fns[4], nsteps)
d = plot(layer(x=0:(nsteps+1), y=[0, snapshots[:,1]],
  Theme(default_color=color("red")), Geom.line),
layer(x=1:(nsteps+1), y=snapshots[:,2],
  Theme(default_color=color("blue")), Geom.line),
layer(x=1:(nsteps+1), y=snapshots[:,3],
  Theme(default_color=color("green")), Geom.line),
layer(x=1:(nsteps+1), y=snapshots[:,4],
  Theme(default_color=color("black")), Geom.line),
layer(x=1:(nsteps+1), y=snapshots[:,5],
  Theme(default_color=color("yellow")), Geom.line),
Guide.title("$fname: alpha=$a"))
draw(PNG("final_images/$fname\$_h\$_ai\$_si.png", 8inch, 6inch), d)
d = plot(x=0:5, y=[0, r.num_ranks],
  Geom.bar, Guide.title("$fname (hist=$h, alpha=$a) Final Distribution"))
draw(PNG("final_images/$fname\_fin\$_h\$_ai\$_si.png", 8inch, 6inch), d)
end
for (bi, b) in enumerate(betas)
# Yelp module: responsible for Yelp data processing

## Module Yelp

```julia
using JSON, Dates
using DataFrames
using HDF5, JLD

function load_data()
    fname = "/home/jkcl/yelp_academic_dataset.json"
    dat_biz = DataFrame()
    dat_rev = DataFrame()
    dat_use = DataFrame()
    (b_id, b_name, b_stars, b_nrevs, b_city, b_cats, b_schools) = (Str[], Str[], Float64[], Int[], Str[], Any[], Any[])
    (r_biz, r_stars, r_dat, r_user, r_use, r_fun, r_cool) = (Str[], Int[], Date[], Str[], Int[], Int[], Int[])
    (u_id, u_nrevs, u_stars, u_use, u_fun, u_cool) = (Str[], Int[], Float64[], Int[], Int[], Int[])
    open(fname, "r") do f
        for line in eachline(f)
            l = JSON.parse(line)
            try
                if l["type"] == "business"
                    push!(b_id, l["business_id"])
                    push!(b_name, l["name"])
                    push!(b_stars, l["stars"])
                    push!(b_nrevs, l["review_count"])
                    push!(b_city, $(l["city"]), $(l["state"]))
                    push!(b_cats, l["categories"])
                else
```

```
push!(b_schools, l["schools"])
elseif l["type"] == "review"
push!(r_biz, l["business_id"])
push!(r_stars, l["stars"])
push!(r_dat, Date([int(s) for s in split(l["date"], ",")...]))
push!(r_user, l["user_id"])
push!(r_use, l["votes"]["useful"])
push!(r_fun, l["votes"]["funny"])
push!(r_cool, l["votes"]["cool"])
elseif l["type"] == "user"
push!(u_id, l["user_id"])
push!(u_nrevs, l["review_count"])
push!(u_stars, l["average_stars"])
push!(u_use, l["votes"]["useful"])
push!(u_fun, l["votes"]["funny"])
push!(u_cool, l["votes"]["cool"])
end
catch
for (k,v) in l
@show (k, typeof(v))
end
end
end
(dat_biz[:Id], dat_biz[:Name], dat_biz[:Stars], dat_biz[:NRevs],
 dat_biz[:City], dat_biz[:Categories], dat_biz[:Schools]) =
(b_id, b_name, b_stars, b_nrevs, b_city, b_cats, b_schools)
(dat_rev[:Biz], dat_rev[:Stars], dat_rev[:Date], dat_rev[:User],
 dat_rev[:Useful], dat_rev[:Funny], dat_rev[:Cool]) =
(r_biz, r_stars, r_dat, r_user, r_use, r_fun, r_cool)
(dat_use[:Id], dat_use[:NRevs], dat_use[:Stars], dat_use[:Useful],
 dat_use[:Funny], dat_use[:Cool]) =
(u_id, u_nrevs, u_stars, u_use, u_fun, u_cool)

(dat_biz, dat_rev, dat_use)