CHAPTER

1

Users: The *Who* of Social Media

Social media revolves around users, and their activities and interactions. Users create the content, communicate with each other, and ultimately keep the service alive and growing. This chapter looks at the typical user's behavior on social media services and the universal similarities you can see across the different services.

First, we focus on the most basic questions about the *overall activity* of those using the service: Are there some regularities in their aggregate statistics? If regularities exist in one service, can they be generalized to other systems? A few very basic conditions affecting usage give rise to measured activity distributions, and we quantify the differences among users in terms of overall activity with the help of observed regularities. Because activity distributions have a specific analytical form, we discuss why it's hard to take and interpret averages in actual social media systems in the presence of such distributions.

Throughout, we support our conclusions with data collected from Wikipedia and Twitter.

Measuring Variations in User Behavior in Wikipedia

One of the most important questions in terms of user activities is: How much do users contribute to, or use, the service? You can look at this question from many different points of view, but certainly one of the most straightforward ways to characterize users is to describe how frequently they come back and are present on the service. You can certainly expect that some users are more "active" than others—but how do you exactly quantify user activity in relation to the service?

User activity can be characterized in the most obvious manner by *how many times* a user performed a certain action such as leaving a comment, sharing a picture, creating or removing social network connections, and so on—in other words, using any facility that the service provides to its users. To determine this, the first thing to do is to define the time period for collecting the data needed to make the measurements.

Figure 1.1 shows two possible scenarios for choosing periods from which we can collect user activity data. In scenario (a), we chose more or less random, non-consecutive periods for the data collection. Although this choice may be valid under specific requirements, we generally prefer consecutive, closed-time ranges for data collection, like those that we can see in case (b). General user behavior may change over time (for instance new users might have different characteristics than older ones), so we prefer to sample user activity within as short a time range as possible. For this reason, case (b) is the natural choice, in which we select a continuous time interval and count the number of times a user has been active within this interval. This is the *frequency* of usage in the given time window.

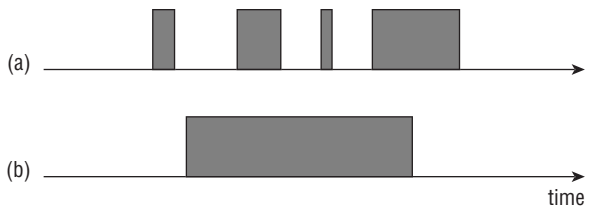

Figure 1.1: Possible choices for sampling time windows to measure aggregate user activity. In scenario (a), we pick non-consecutive time windows randomly. In (b), we choose a continuous time window between two given points in time.

The Diversity of User Activities

We can reasonably assume that users will differ in how likely they are to use a service: some will be very active, whereas others will use the service only once in a while. How large are these differences, and how can we characterize them? These are the questions you look at in this section.

This section uses the Wikipedia edit history logs. First, we look at how often Wikipedia editors contribute to articles: The question is how many times a given user makes a change to any Wikipedia article in each time period. A Wikipedia "editor" is anyone with a registered user name, and in the broader sense anyone who makes a smaller or larger change to any Wikipedia article. Luckily, the Wikimedia foundation makes the edit history of all articles and users available on its web site (`http://en.wikipedia.org/wiki/Wikipedia:Database_download#English-language_Wikipedia`). The dataset used in this chapter describes all revisions (edits) of the English-language Wikipedia. We chose the English Wikipedia because it is the oldest and most comprehensive among Wikipedias in various languages. (For statistics on the different Wikipedias, see `http://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia#Comparisons_with_other_Wikipedias`).

You can find the revisions metadata (without actual page content) in the file named `enwiki-*-stub-meta-history.xml.gz` for the latest dump date. At the beginning of 2018, the compressed file measured approximately 54GB. The data is in XML format, and for each page the file contains the full edit history with the name and ID of the user, and the timestamp of the edit. To measure users' frequencies of usage in a time window, it's only these fields that are of interest to us. You can download the latest Wikipedia data dump by running the shell script `src/chapter1/wikipedia/get_data.sh` from the book's online repository.

Generally, in data analysis you spend a significant amount of effort on just preprocessing and cleaning up data in a database or flat file. To illustrate the typical workflow, we'll walk you through this step with the downloaded Wikipedia file. Because XML is a rather verbose way to describe structured data, the first task is to transform this file into a format that you can easily work with later because you'll want to read through it multiple times as you experiment with the data. Transforming data files into a format that you feel comfortable with improves development and running times. To that end, in this example you first preprocess the downloaded file with the `src/chapter1/wikipedia/process_revisions_xml.py` Python script, which generates a flat text file with the edit records in rows and the user IDs and timestamps in tab-separated columns (this may take a long time, given the large size of the input file). Alternatively, we could have chosen to store the data in a relational (SQL) database, which in many cases is a preferable solution. However, for simplicity, we use a plain text file in this example.

PROCESSING LARGE XML FILES: DOM VERSUS SAX

For processing large XML files, you have a choice between the Document Object Model (DOM) or the Simple API for XML (SAX) approach. In both cases, there is a variety of high-quality libraries at your disposal in every major programming language, including Python, so the toolset available to you with either approach is essentially equivalent. The difference is, however, that the DOM model represents the XML file after reading it in its entirety as a *tree in memory*, mapping the XML nodes to nodes of the tree. On the other hand, SAX takes an event-driven approach: As the SAX parser reads the XML input, events are fired as callbacks to your main program, and you can decide what to do and how to process the nodes and data read. Obviously, for this example, only SAX parsing works, as you cannot reasonably hope to store this much data in RAM. (Nor should you want to, as you need only one pass through the file to distill the relevant information about the revisions, writing the fields of every revision out as you finish with a record.) In Python, the SAX library is in the package `xml.sax`, and you can see how XML nodes are handled in `process_revisions_xml.py` in the `startElement` and `endElement` methods that are called when an XML node is opened and closed, respectively.

The result of parsing the Wikipedia revision database with our script is a flat tab-separated text file as mentioned, with the following columns:

- Name of the page edited
- "Namespace" of the page (which is the "type" of the page, `https://en.wikipedia.org/wiki/Wikipedia:Namespace`)
- ID of the page
- Revision ID
- Timestamp of the revision
- ID of the editing user
- Editing user's account name
- IP address of the user (only for anonymous editors)

Although Wikipedia pages are categorized into *namespaces*, such as normal articles, user pages, and help pages, among others, we did not restrict ourselves to any specific namespace and considered all edits. With this distilled file, we can easily calculate how many edits any user made in a given time frame by simply iterating through the file and recording the user ID of a revision that falls within a specific time range. In Listing 1.1, we specify three date ranges (the first one, two, and three months of 2013, respectively) for counting the times a user made an edit within each of the date ranges. The number of times a user edited an article will then be written into the output file, where every line is for a user and three columns are for the number of edits by that user for the given date range, respectively.

Listing 1.1: This script takes all Wikipedia revisions and outputs the number of edits made by any user between the date periods defined in `DATE_RANGES`. (`user_edits_in_timeframes.py`)

```
'''
Count the number of times particular users made edit in the given time
frames.
'''

import gzip
from collections import defaultdict


INPUT_FILE = 'data/wikipedia/revisions.tsv.gz'
OUTPUT_FILE = 'data/wikipedia/user_edits_in_timeframes.tsv.gz'

DATE_RANGES = [('2013-01-01T00:00:00', '2013-02-01T00:00:00'),
               ('2013-01-01T00:00:00', '2013-03-01T00:00:00'),
               ('2013-01-01T00:00:00', '2013-04-01T00:00:00')]

# The number of times a user made a revision in a given date range.
user_frequencies = defaultdict(lambda: defaultdict(int))

user_names = dict()
with gzip.open(INPUT_FILE, 'r') as input_file:
    for line in input_file:
        title, namespace, page_id, rev_id, timestamp, user_id, \
        user_name, ip = line[:-1].split('\t')
        # We only keep registered users, and need to strip user ID 0
        # due to a logging bug (http://en.wikipedia.org/wiki/User:0).
        if user_id != '' and user_id != '0':
            for range_id in xrange(0, len(DATE_RANGES)):
                if timestamp >= DATE_RANGES[range_id][0] and \
                timestamp < DATE_RANGES[range_id][1]:
                    user_frequencies[user_id][range_id] += 1
                    user_names[user_id] = user_name

with gzip.open(OUTPUT_FILE, 'w') as output_file:
    for user_id in user_frequencies.iterkeys():
        output_file.write('\t'.join(
            [user_names[user_id]] + \
            [str(user_frequencies[user_id].get(range_id, 0)) \
             for range_id in xrange(0, len(DATE_RANGES))
             ]))
        output_file.write('\n')
```

The first step toward describing aggregate user activities is to calculate a histogram of the number of times a user made an edit within a date range.

Visualizing histograms is a common task carried out to understand the differences among users, and we'll frequently do this in this book. The R code snippet in Listing 1.2 reads in the output produced by the Python script in Listing 1.1 and plots the number of users with a given number of edits in the first time period (January 2013).

Listing 1.2: Read and plot a frequency histogram for the number of times a Wikipedia editor changed a page for January 2013. (`user_edits_in_timeframes.R`)

```
library(plyr)
library(ggplot2)

revs.in.periods = read.table(
        gzfile('data/wikipedia/user_edits_in_timeframes.tsv.gz'),
        sep='\t', col.names=c('account', 'range1', 'range2', 'range3'),
        comment.char='', quote='')

# Only users with > 0 edits in Jan 2013 are considered.
ggplot(subset(revs.in.periods, range1 > 0, select='range1'),
aes(range1)) +
        geom_histogram(binwidth=1, origin=-0.5) + xlim(0, 20) +
        xlab('Number of revisions made') + ylab('Number of users in
        period')
```

The result of this is shown in Figure 1.2, where you can see that as you consider a larger and larger number of revisions, the number of editors making that many edits quickly decreases. In fact, when you look at the distribution more in detail, you find that there were a handful of registered users (this may include so-called "bots", or robots, that are automated Wikipedia agents to carry out some bookkeeping task) that made tens of thousands of edits in just 1 month!

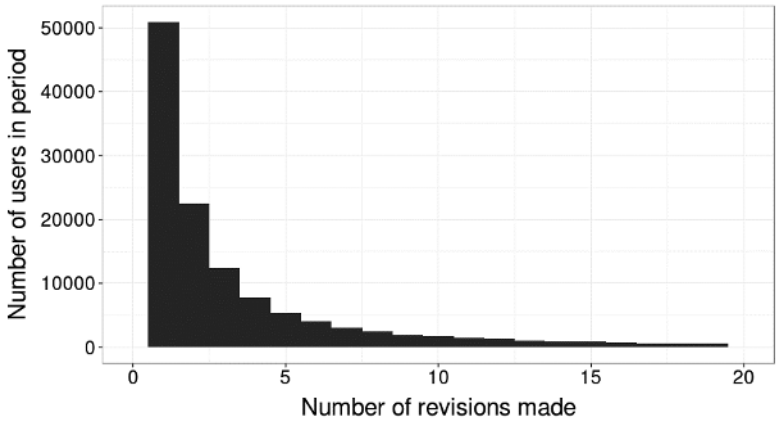

Figure 1.2: The number of editors who made a certain number of revisions to any Wikipedia article during January 2013. The horizontal axis has been truncated to show no more than 20 edits a month; however, the data shows that you can find users with tens of thousands of edits as well.

Nevertheless the number of such users is few; the average number of edits per user in January 2013 is approximately 30, and the median is 2. You can immediately see a big difference between the mean and the median: This is a sign of a highly *skewed* distribution, where a few of the high-end outliers can change the average, whereas the median will not be significantly affected by them. We discuss this in more detail a little later in this chapter.

You can consider the number of revisions made by a user as a random process because so many unknown factors determine a user's activity that it's impossible to account for all. Users may come and go, find something interesting and add some modifications to an article, or start a new article when they find that the topic of their interest does not exist yet. Indeed, when and how frequently users find the time to come back to the service varies as well, yet, as Figure 1.2 suggests, you can expect some regularities in this if you aggregate the behavior of many users: The histogram looks like a rather smooth function, so it wouldn't be surprising to find an explanation for why this is so.

Regarding user activity as a random process, then, you can also approximate the *probability density function* (PDF) of the process as just the normalization of the histogram displayed in Figure 1.2. This means that you need to divide the frequency counts for users by the total number of users, which of course is also the area under the histogram considered as a function. The probability that a user will have n edits is given by the following formula:

$$P(edits = n) = \frac{U(edits = n)}{\sum_{i=1}^{\infty} U(edits = i)}. \tag{1.1}$$

In this formula, P gives the likelihood that a randomly chosen user will have made n edits in the period under consideration. $U(edits = n)$ denotes the number of users who made n edits, so the denominator is just exactly the total number of users we have (because a user may not belong to two different activity buckets at the same time, and every user belongs to a bucket). This is the probability distribution function, yielding the likelihood of an event over the range of all possible events. (It is also sometimes called the *probability mass function* for discrete distributions, such as we have for the number of edits per time period). If we were to plot the probability distribution function, it would look just like Figure 1.2, with only the vertical axis rescaled because we have divided all values by the same constant, the total number of users.

Is there anything more we can say, though, about how many revisions we will expect to see from the users, in the future, or for any time period? The smooth decay of the frequencies we have seen from this limited example suggests that we may find some regularities for user behavior in a more general sense as well. After all, the whole purpose of this exercise is to learn from the past and to anticipate the user activity distribution in the future and detect any deviations from our expectations.

Now we'll see how the user activity distribution changes if we consider longer and longer periods for our observations. To this point in the example we've been discussing, we took the edits of all users who were active during 1 month. Now, we will extend this 1-month period to 2, and then to 3 months. We made provisions in Listing 1.1 for these measurements, where, to go through our distilled Wikipedia revisions file only once, we defined two other time periods aside from the 1 month we have been focusing on until now. These two additional date ranges, as the code example shows, are from the beginning of January until the end of February, and until the end of March 2013, respectively. To see how the user activity distributions relate to each other, we'll plot them together. It is, of course, expected that 2 or 3 months will naturally enable more users to participate, and this is exactly what you can see in Figure 1.3. Although the shapes of the functions for the number of users making a certain number of revisions are similar across the three different time ranges, a longer range allows for more users to make more edits.

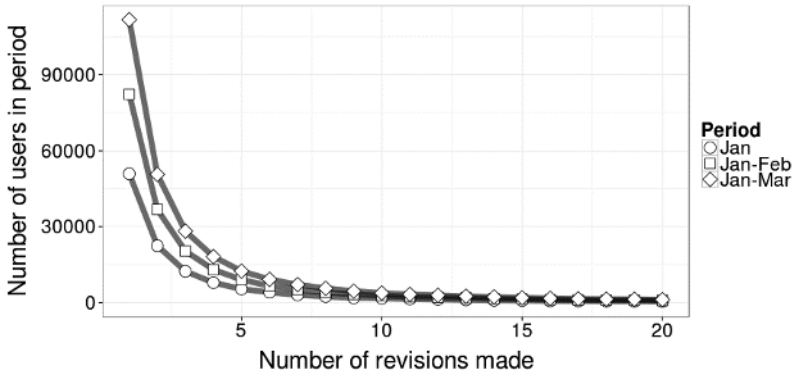

Figure 1.3: The number of revisions for three different time windows: for the first, the first 2, and first 3 months of 2013, respectively, as is also indicated by the figure's legend. The calculations were made the exact same way as for Figure 1.2.

Moving on from just absolute user counts, now look at the probability distribution functions for the number of revisions for all time periods: What is the likelihood that a given user will make a certain number of edits in the given period? For this we will calculate the probability that a randomly chosen user will make exactly r edits in period p. The corresponding equation follows the same formulation as Equation 1.1:

$$P_p(r) = \frac{U_p(r)}{\sum_{i=0}^{\infty} U_p(i)}. \tag{1.2}$$

Here and later in this section, $U_p(r)$ denotes the number of users making exactly r Wikipedia edits in the chosen period p (where p may be 1, 2, or 3 in our particular setup to denote the three date ranges of Figure 1.3, for instance). $p = 1$ for Jan 2013, $p = 2$ for Jan–Feb 2013, and $p = 3$ for Jan–Mar 2013. $P_p(r)$ is therefore the likelihood that a randomly chosen user has r revisions in period p.

The part of the R code that calculates $P_p(r)$ can be seen in Listing 1.3.

Listing 1.3: This R code snippet splits up `revs.in.periods.long` twice, first by time period, and then by revision count, and calculates the fraction of users in a time period who made a given number of edits, out of all the users in that time period. (`user_edits_in_timeframes.R`)

```
# Calculate the fraction of users separately for each date range who
# make a certain number of revisions, excluding all users who make zero
# edits in any of the time windows.

revs.in.periods.long = melt(revs.in.periods, 'account',
        variable.name='range', value.name='revisions')

normalized.revisions = ddply(subset(revs.in.periods.long,
                        revisions > 0),
            .(range), function(one.range) {
                  user.count = nrow(one.range)
                  ddply(one.range, .(revisions),
                            function(one.revision)
                                  data.frame(user.fraction=
                                  nrow(one.revision) / user.count)
                  )
            })
```

We plotted the results of this for the normalized user counts in Figure 1.4. One thing that you can immediately notice is that the three probability distribution functions are similar to each other. (Actually, to prove this visual similarity with numbers, look at the code snippet under Calculation 1.1 in the `user_edits_in_timeframes.R` source file. This takes the three probability points for every single revision count and calculates the relative mean squared errors from their respective averages: They're all in the 6–8% range for the revision counts between 1 and 10.) Now take a leap of faith, and assume that for every possible revision count r the user fractions are the same for all three periods:

$$\frac{U_1(r)}{\sum_{i=1}^{\infty} U_1(i)} = \frac{U_2(r)}{\sum_{i=1}^{\infty} U_2(i)} = \frac{U_3(r)}{\sum_{i=1}^{\infty} U_3(i)}. \tag{1.3}$$

This assumption will help us a lot going forward, as we'd like to explain why this regularity may arise.

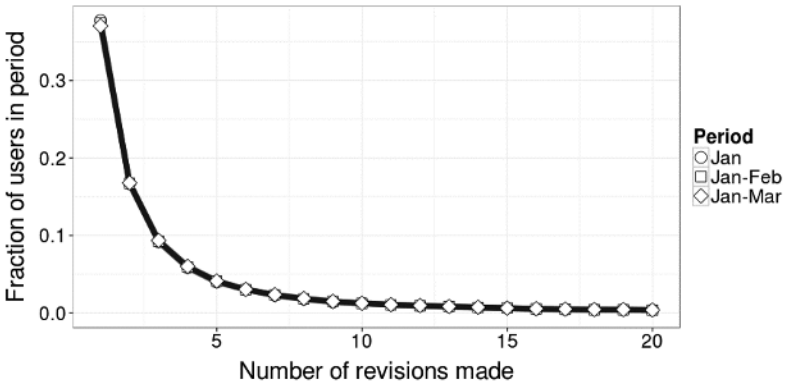

Figure 1.4: The probability that a user will make a given number of Wikipedia edits. Note that the functions for the three periods overlap to a large degree, and it is hard to see a difference for any but the first data point.

Because the denominators of Equation 1.3 are all constants (these are the total number of users making at least one edit in each of the periods), we also should express the detailed user counts $U_p(r)$ with each other in a simpler way by introducing the following ratios:

$$C_{21} = \frac{\sum_{i=1}^{\infty} U_2(i)}{\sum_{i=1}^{\infty} U_1(i)}$$

$$C_{31} = \frac{\sum_{i=1}^{\infty} U_3(i)}{\sum_{i=1}^{\infty} U_1(i)}. \tag{1.4}$$

Ostensibly, C_{21} is the total number of active users in Period 2, divided by the number of active users in Period 1 (and similarly for C_{31}). With these, we can express both $U_2(r)$ and $U_3(r)$ using $U_1(r)$, as follows:

$$U_2(r) = C_{21} U_1(r)$$
$$U_3(r) = C_{31} U_1(r). \tag{1.5}$$

Remember, though, that it was only our intuition that the normalized user counts should be equal at every r, and Equation 1.3 holds. To check this in a more direct way, we'll for a second relax our assumption that C_{p1} is independent of r because we can measure the $U_p(r) / U_1(r)$ fractions for every r, which we'll naturally call $C_{p1}(r)$. We can also check how C_{21} and C_{31} look in practice, and

whether, by measuring them directly, we can still see them being constant. The R code example in Listing 1.4 calculates these ratios for both Periods 2 and 3.

Listing 1.4: Calculate the ratio of the number of edits between users in Periods 2 and 3 to those in Period 1, with the same number of revisions. See the figure below for the results. (`user_edits_in_timeframes.R`)

```
# Count the number of users in each period with a given number of > 0
# revisions.
user.counts.long = ddply(subset(revs.in.periods.long, revisions > 0),
                   .(range, revisions), nrow)

# Reformat the results into a wide table where the number of revisions
# are the rows and in three columns we have the user counts for each of
# the ranges.
user.counts.wide = dcast(user.counts, revisions ~ range)

# Calculate the pairwise ratios between the user frequencies in each
# revision bucket, with respect to those in range 1.
ratios = within(user.counts.wide, {
                    ratio21 = range2 / range1
                    ratio31 = range3 / range1
             })
```

Figure 1.5 displays $C_{21}(r)$ and $C_{31}(r)$ as a function of the number of revisions made. What we can immediately notice is that to a large degree, (the revision dependent) $C_{21}(r)$ and $C_{31}(r)$ appear to be constant, independent of the number of revisions, r. Now we can ask: What if we can indeed model these ratios as constants, and what consequences does this fact have on our understanding of user activities?

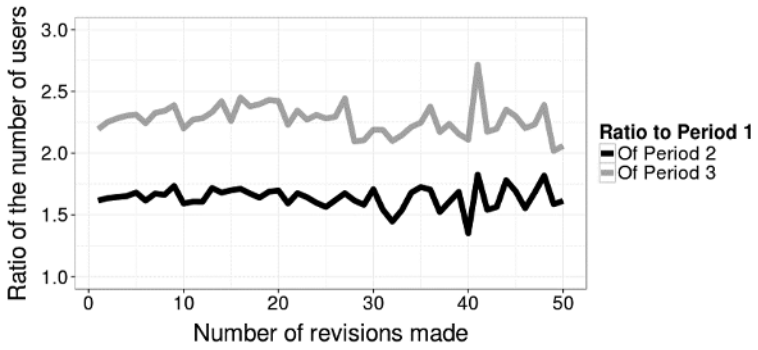

Figure 1.5: The number of active users were taken for the three periods we used before (Jan 2013 for Period 1, Jan–Feb 2013 for Period 2, and Jan–Mar 2013 for Period 3) for any given revision count. For this plot, we divided the number of editors with a given number of revisions in Period 2 with those in Period 1 with the same number of revisions, and plotted it in the dark line. Similarly, we also took the ratio of the user counts in Period 3 to those in Period 1 and plotted those with the lighter line.

First, let's estimate what the values of these constants are! Figure 1.5 suggests that the ratio between Periods 2 and 1 is approximately 1.6, and the ratio between Periods 3 and 1 is approximately 2.2–2.3. If we were thinking that these ratios are the same as the ratios between the lengths of the appropriate periods, assuming that a longer period gives rise to a proportionally larger number of active users, then we were wrong: If we divide the number of days in each period with each other, we will get (31 + 28) / 31 ≈ 1.9 and (31 + 28 + 31) / 31 ≈ 2.9, respectively. The differences between these pairs of numbers are large, so we cannot just assume that the longer periods we take, the more users we will see in the revision buckets, proportionately.

At any rate, what have we learned about user activity distributions up to now from Wikipedia's example? To summarize the main findings:

- A large diversity exists in the number of actions that users take in a given period. There are many users who have only a few actions, and the number of the active users decreases sharply as the number of actions we consider increases.
- If we take longer and longer time periods, we will naturally observe a larger number of actions from a larger number of users. However, the histograms for user activity counts appear to come from the same family of functions, as their functional forms are scaled with respect to each other.
- Considering the normalized probability distribution functions, no matter which period we are sampling users from, these functions will be the same. We have also seen that this is because the number of users with a certain number of actions is a constant multiple of a universal function that does not depend on the period. (This is what we expressed by the C_{p1} constants earlier.)

The Origin of the User Activity Distribution

Let's develop this last point further: Can we say something more about the user activity distributions, given that we found the regularities in the previous section? This section highlights some more measurements necessary to come to our conclusion, and also some analytical methods often useful in modeling the random nature of online user behavior. For this we can also make and verify one more assumption: If we observe *particular* users for longer, the number of times they are active will be also larger. This is an almost trivial assumption, as we can expect that the longer time ranges are available for the users, the more chances they will find to use the service. Also, we can reasonably think that their individual number of actions, on a large scale, will be proportional

to the length of the time window for the observation. This expresses our belief that individual users can be characterized by an average activity rate *specific to them* that describes how many times they use the service in a unit of time, and we assume that this rate stays more or less constant over time. (At least if we take long enough time windows. We may expect that there are times when the user acts in "bursts," which are short periods of time when they are more active than at other times. More on this in Chapter 4.) Note that this proportionality assumption does not necessarily mean that a time window that is, for example, twice as long as another will necessarily result in twice as many actions for a given user. Overall user activity may seasonally fluctuate in time (for instance, with a yearly periodicity), and there could be periods when large-scale user activity goes down, and some other periods when it goes up.

To see how the number of user edits changes when you change the length of the observation window, you can look at how many more or fewer edits users made in Periods 2 or 3, *given* that they made a certain number of edits in Period 1. There are obviously a lot of users who make a given number of edits in Period 1, and they all individually have possibly different numbers of revisions in the other periods. Therefore, you can take the *average* of the number of edits in the other periods for all users with that same exact number of edits in Period 1. In other words, if $\overline{r_2}(r_1)$ denotes the average number of edits in Period 2 for all users who had r_1 number of edits in Period 1, then, more formally,

$$\overline{r_2}(r_1) = \frac{\sum_{i=1}^{N} I(r_{i,1} = r_1) r_{i,2}}{\sum_{i=1}^{N} I(r_{i,1} = r_1)}, \tag{1.6}$$

where the sums go over all users (their number we denoted here by N); $r_{i,1}$ and $r_{i,2}$ stand for the number of revisions made by the ith user in Periods 1 and 2, respectively; and $I(r_{i,1} = r_1)$ is the *indicator function* being 1 when its argument is true, and 0 when it is false. The numerator of this equation is therefore the sum of the revisions in Period 2 for users who had r_1 revisions in Period 1, and the denominator is the number of such users.

When you look at the average number of edits made by users calculated in this way, as shown by the Figure 1.6, you can notice that the edits in the longer periods are well approximated by linear functions of the edits made in the shorter period. Also, remember that Periods 2 and 3 run from January through the end of February and March, respectively, so they overlap with Period 1 (which is January only), and thus the edits made in those two periods are never fewer than the edits in Period 1.

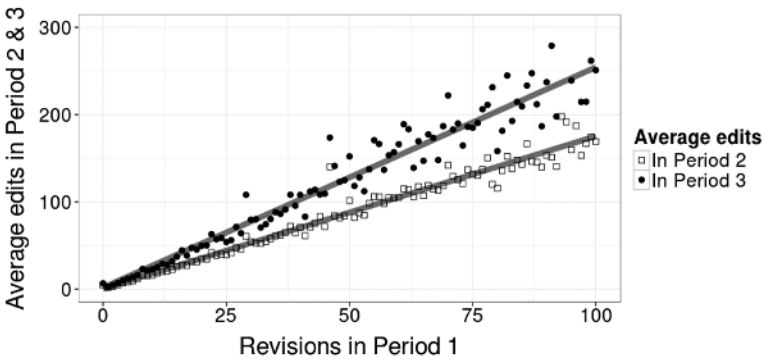

Figure 1.6: The average number of edits made by users in Periods 2 (and 3), given that they made a certain number of edits in Period 1. The average values seem to be in a linear relationship with the number of edits in Period 1, and the best fitting linear functions are shown with the straight lines.

According to the observations above, then, you can assume that a good model for the number of revisions in two different time frames, at least in the average sense, as shown by the equation below, is that

$$r_2 = R_{21} r_1 , \tag{1.7}$$

such that if the user made r_1 revisions in Period 1, he will make *on average* r_2 revisions in Period 2, which is linearly proportional to r_1 with the constant R_{21}.

To recap, bear in mind the following two facts, which we will develop further:

1. The edits made by a user in two time periods of different lengths are linearly proportional to each other (Figure 1.6).
2. The probability distribution function of the activities does not depend on the time frame we performed the measurements in, as you saw in the previous section (Figure 1.4).

In the next few paragraphs we elaborate on both facts, starting with the first. Remember: Our goal is to explain the origin of the distribution functions that describe user activities.

Consider the number of edits that a user makes in a given time period as a random variable, R. We'll denote the probability distribution function (which is, recall, approximated by the normalized number of users making a certain number of edits as in Figure 1.4) by $f_R(r)$. R here stands for the random variable, and r is its particular value. This is, again, the probability that a random user will have r revisions. Although we know that r is a discrete random variable, we can proceed with the underlying assumption that it is continuous. The reason is that at this point we don't want to derive exact results, but instead understand

the distributions that we have seen in a general sense that may be applicable to other systems as well.

If we multiply this continuous random variable by a constant c, its PDF $f_{cR}(r)$ will become, as expressed by its original PDF $f_R(r)$,

$$f_{cR}(r) = \frac{1}{|c|} f_R\left(\frac{r}{c}\right). \tag{1.8}$$

This is just a fact from probability theory for the resulting distribution function when we multiply the random variable by a constant. We should now consider how the PDF of the edit activity changes when we move from Period 1 to Period 2. Let's first write out the PDF for the random variable r, the number of revisions a user makes, because this is the PDF we're going to work with. If U_1 denotes the total number of users in Period 1 such that $U_1 = \sum_{i=1}^{\infty} U_1(i)$, then the PDF in Period 1 for instance is approximated by

$$f_R(r) = \frac{U_1(r)}{U_1}. \tag{1.9}$$

We know that according to our assumption (Equation 1.5) we need to multiply the number of revisions for every user by R_{21}, to arrive at the number of revisions r_2 they make in Period 2; our ultimate goal is to compare the product-distribution with the actual distribution that we can measure and see if they match up. Now when we multiply the random variable argument r of f_R by R_{21}, we should be able to deduce what the new PDF will be. Using Equation 1.8:

$$f_{R_{21}R}(r) = \frac{1}{R_{21}} \frac{U_1\left(\frac{r}{R_{21}}\right)}{U_1'}. \tag{1.10}$$

Notice that, more importantly, we did not simply write U_1 in the denominator of this equation. Instead, we have U_1' there. The reason we do this is that the normalization factor will also change when we multiply the random variable as in Equation 1.9: The proper sum for the normalization again runs through all integers from 1 to ∞ as

$$U_1' = \sum_{r=1}^{\infty} U_1\left(\frac{r}{R_{21}}\right). \tag{1.11}$$

We have a challenge now. The arguments of U_1 in the sum are not going to be integers, and also for $r < R_{21}$, the argument of U_1 is less than 1, which is not possible. This we can interpret, though, as referring to users who "did not show up" in Period 1 because their activity rate was so low that we could not observe even one revision from them. However, we can still think of them as having had a strictly positive, but less than 1, activity rate, which, after being multiplied by

R_{21}, became measurable in the longer Period 2. But because we could not count their numbers (given that they did not show up in Period 1!), how are we going to determine what U_1' has to be?

We can also have a different approach to calculating U_1': It must be the sum of all user counts over all possible activity buckets in Period 2, because it was exactly our purpose with multiplying the revision counts r_1 by R_{21} to arrive at the revision counts r_2 in Period 2. This sum, we know, is exactly U_2, the number of users who had any activity in Period 2; therefore $U_1' = U_2$. Seeing this we can finally complete the expression for the probability density function for the rescaled argument in Equation 1.10 as

$$f_{R_{21}R}(r) = \frac{1}{R_{21}} \frac{U_1\left(\frac{r}{R_{21}}\right)}{U_2}. \tag{1.12}$$

Why is it good, though, that we know this? We can use our other empirical observation now about the probability distribution functions being unchanged over observation periods, as described by Equation 1.3. What we recovered, then, when we multiplied the revision counts for all users in Period 1 by R_{21} (Equation 1.7), is the probability distribution of the number of revisions in Period 2. Combining this and Equation 1.12, we get

$$\frac{U_1(r)}{U_1} = \frac{U_2(r)}{U_2} = \frac{1}{R_{21}} \frac{U_1\left(\frac{r}{R_{21}}\right)}{U_2}. \tag{1.13}$$

Because we would like to say something about $U_1(r)$, let's just focus on the first and third terms in this equation. Slightly reorganizing the constants,

$$U_1\left(\frac{r}{R_{21}}\right) = \frac{U_2}{U_1} R_{21} U_1(r). \tag{1.14}$$

Remember though that our ultimate goal is to figure out why $U_1(r)$ (and $U_2(r)$, and $U_3(r)$) have the particular shape they have, which could help us understand several further properties of user behavior down the line. For this, let's simplify for a moment the notations in Equation 1.12 to see the problem more clearly. By pulling the constants together and using the simpler notation g, instead of $U_1(r)$ for the unknown function, we can say that what we're looking for is a function g that satisfies:

$$Ag(x) = g(Bx), \tag{1.15}$$

with A and B being constants. In other words, if we multiply the argument of the function by a given constant, we get back almost exactly the function value as we would otherwise have for the argument, except that this value is also

multiplied by another, different constant. This looks simple, but what kind of functions will satisfy this condition? We can use a theorem called *Euler's homogeneous function theorem* to find a solution for g. Euler's theorem states that if the following is true for a given constant γ and *any* other constant C:

$$g(Cx) = C^{\gamma} g(x), \tag{1.16}$$

then g should also satisfy

$$xg'(x) = \gamma g(x). \tag{1.17}$$

g', as usual, is the derivative of g with respect to x. It must be also true then that

$$\frac{g'(x)}{g(x)} = \frac{\gamma}{x}. \tag{1.18}$$

Because the left side is equal to $\left[\ln g(x) + a\right]'$, and the right side to $\left[\ln\left(kx^{\gamma}\right) + b\right]'$ with arbitrary, new constants k, a, and b (it's easy to see knowing the basic rules of derivation), we come to

$$\left[\ln g(x) + a\right]' = \left[\ln\left(kx^{\gamma}\right) + b\right]'$$

$$\ln g(x) + a = \ln\left(kx^{\gamma}\right) + b + c$$

$$g(x) = Kx^{\gamma}. \tag{1.19}$$

(We introduced K, which is just another constant, and could be expressed with k, a, b, and c.) This result is rather simple and elegant, and we used two facts only: (1) that the probability distribution functions for user activities over time are unchanged (Figure 1.4); and (2) that the number of user actions between two time frames changes proportionally, no matter how active the users were in the first place (Figure 1.6). With this we can go back and rename our variables again. Our final result is that the number of users U having a certain number of revisions, r, should follow a so-called *power law* (because we raise the variable in the formula to a constant power)

$$U(r) = Kr^{\gamma}, \tag{1.20}$$

with some constant γ and a scaling factor K. Seeing this, it's indeed surprising how compactly we can express our user activity distribution: Our hunch at the beginning of the chapter that the distribution could be some sort of a smooth function was indeed right. Now that we have a closed form for this, we can analyze some of its properties.

We can also check directly whether we can observe a power law for the user activities: Let's see if we can show that this is the case for Wikipedia editors as well! If our assumptions along the way were correct, we should find that the

number of editors making a certain number of revisions follows Equation 1.20 to a large degree. The easiest way to proceed with this is by noticing that if we take the logarithms of both sides of this equation, we get

$$\ln U(r) = \ln\left(Kr^{\gamma}\right) = \gamma \ln r + \ln K. \tag{1.21}$$

In other words, the logarithm of the number of users with r revisions, $U(r)$, is a linear function of the logarithm of the number of revisions, r. ln K is just a constant, and so is γ, so we can see how the linearity arises between ln U and ln r. We can now take the exact same data as we did for Figure 1.3, transform the variables both on the horizontal and vertical axes, and plot them against each other. The result is shown in Figure 1.7.

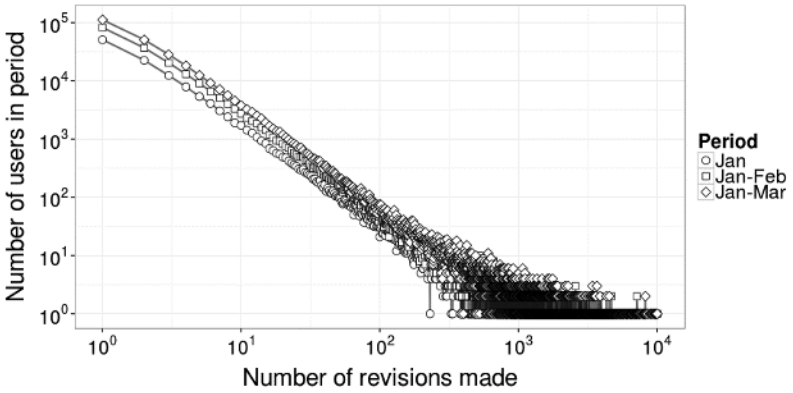

Figure 1.7: Similar to Figure 1.3, we show the number of users who made a certain number of revisions in the three time periods. However, in this figure, we rescaled both axes logarithmically, so we can now clearly observe the power law relationships.

We can notice a few things:

- Whereas the horizontal axis of Figure 1.3 shows a limited range only (it goes only up to 20 because otherwise we would not see anything interesting due to the fast decay of the power-law function), the logarithmic rescaling lets us see a much more detailed and complete picture of the relationships well beyond this restricted range. Although the distance between neighboring data points becomes shorter and shorter on paper as we go toward larger x values, it's exactly this gradual increase in density that allows us to also discern how the functions behave at the far ends of their ranges.
- The logarithmic axes also make comparisons between the three distributions easier. In this figure, we can clearly follow through the functions'

differences across the entire horizontal domain, and we can discern both large differences (on the order of tens of thousands of revisions, on the left side) and small differences (on the order of 1–10, right side) in the same plot. This is because the vertical, y axis is logarithmically rescaled as well.

- What we can observe is that the functions seem to remain "parallel" with each other. What this means is that the *ratios* of their y values at every point are constant: A $\ln y_1 - \ln y_2 = C$ constant difference can be expressed as $y_1 / y_2 = e^C = const$; therefore, a parallel shift on a logarithmic axis means a multiplication by a constant on the original scale. Alternatively, if we multiply a function by a constant, on a logarithmic y axis, it will appear as a shift upward or downward, depending on whether the multiplier constant was greater or less than 1, respectively.
- Now we can immediately see the linear relationships on the logarithmic scales between the log-transformed number of revisions and the log-transformed number of users, apparent from the fact that all three curves follow (approximate) straight lines on the log-log scales. Equation 1.21 already expressed this, and now we can see that it is indeed the case. In fact, if any function is a straight line on a double logarithmic plot, we can be assured that it's a power law of the form given by Equation 1.19.
- We can also observe a "fanning out" of the y values at the high end of the horizontal scale. This is a result of the higher variances at smaller sample sizes: You can see that at approximately 1,000 revisions per user, we find only a few users per revision bucket (fewer than 10). Although we still expect that a power-law model reasonably approximates the user counts, our observations, which can be considered a sample drawn with a small size from a random process, will have considerable variation around the expected mean (which is given by the power law). A remedy for this can be to use increasing bin sizes to pool several buckets into one, and average out the user counts, as we'll see in the "Logarithmic Binning" section of this chapter.
- In contrast to our usual experience with linear plots, moving one "unit" to the right or toward the top in the log-log plot means that we are stepping *decades* (in the case of base-10 logarithmic scales), or in other words we are multiplying our values by 10. So, as we move from the origin to the right, for example, we are moving on the x axis from 1, to 10, to 100, to 1,000, and so on, so making linearly displaced steps in the plot will result in exponential changes in the values on the axes they represent.

Furthermore, we can make one more prediction for the power-law exponent as a validation of our assumptions, realizing that Equation 1.14 for u_1 has the

same shape as Equation 1.16. Comparing these two equations, we can see that $C = 1/R_{21}$, and so what we need to determine is, n, the exponent of C, on the right side of Equation 1.16. But this we can calculate easily:

$$C^{\gamma} = \frac{U_2}{U_1} R_{21}$$

$$\gamma = \log_C \left(\frac{U_2}{U_1} R_{21} \right) = \frac{\log\left(\frac{U_2}{U_1} R_{21} \right)}{\log \frac{1}{R_{21}}} = -\frac{\log \frac{U_2}{U_1}}{\log R_{21}} - 1. \tag{1.22}$$

We know all the actual values in this expression for Periods 1 and 2: the measured total number of active users happens to be $U_1 = 134,804$ and $U_2 = 219,604$. R_{21} is the slope of the linear fit to the data points represented by the "square" symbols in Figure 1.6: $R_{21} = 1.75$. Substituting these for the exponent γ into Equation 1.22, we will get γ=–1.87. So, our expectation is that the number of users with a given number of revisions will go as

$$\frac{U_1(r)}{U_1} = \frac{U_2(r)}{U_2} = \frac{U_3(r)}{U_3} \propto r^{-1.87}. \tag{1.23}$$

(We neglected the normalizing constant, and this proportionality is indicated by the commonly used "∝" sign.) We obviously have the chance to check this statement for the exponent, using real data; we will do this momentarily, after inspecting some more properties of the power-law relationship found for user activities.

The Consequences of the Power Law

What are the further consequences of this law that we just found for the user activities? Let's first look at the *cumulative distribution function* of the probability distribution function that is shown in Figure 1.4. The cumulative distribution function gives the probability that the value of the random variable is *no greater* than a given threshold. In our specific case, it means that we are looking for the fraction of our users that make less than or equal to a certain number of revisions in the time period. Let's call this upper limit ρ.

$$CDF(\rho) = P(r \le \rho) = \frac{1}{U} \sum_{i=1}^{\rho} U(i). \tag{1.24}$$

Here, as before, $U(i)$ is the number of users making i edits, and U is the total number of users. To calculate this, we can use Listing 1.5. (For Period 1, as we

can see in the `subset` statement, however all periods have the same PDF so this should not matter.)

Listing 1.5: Approximate the cumulative distribution function of the user activities—the fraction of users who have no more than a certain number of edits in a given time period. (`user_edits_in_timeframes.R`)

```
rev.buckets = ddply(subset(revs.in.periods, range1 > 0,
                           select='range1'), .(range1), summarise,
            count=length(range1))
names(rev.buckets)[1] = 'revisions'

# Make sure we have an increasing ordering of the revision buckets.
rev.buckets = rev.buckets[order(rev.buckets$revisions),]
total.users = sum(rev.buckets$count)
rev.buckets = within(rev.buckets, {
                  cdf = cumsum(count) / total.users
            })
```

The result of this simple calculation is shown in Figure 1.8. Again, we rescaled the horizontal axis, the number of edits, logarithmically. We have every reason to do so, as we by now know that this value spans a vast range, from 1 to 10,000; if we used linear scale, we could not have discerned the sharp rise of the function at the beginning of the scale. This lets us observe surprising facts about our activity distribution: 40% of the active users have only one edit for that one month, and approximately 85% or so have at most 10! Apparently, most of our users are not that active compared to the most frequent editors, and only a small fraction makes a large number of edits.

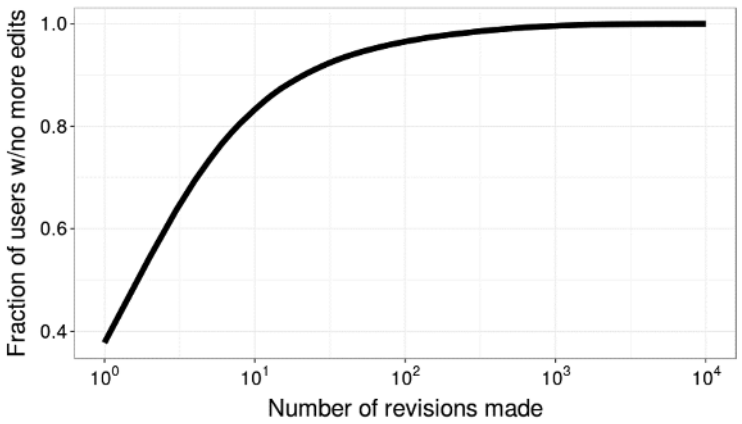

Figure 1.8: The cumulative distribution function of the number of users with a given number of edits. The CDF is the fraction of users with no greater than a specific number of edits. We rescaled the horizontal axis for better visibility.

We have now looked at what fraction of our users have made no more than a certain number of edits, or in other words what fraction of the users are making the least number of revisions. Can we also look at the inverse of the problem, the fraction of users who are making the *most* number of edits? This is certainly not much harder than what we just did, and this is called the *complementary cumulative distribution function* or the *tail distribution* of the activities:

$$CCDF(\rho) = P(r > \rho) = \frac{1}{U}\sum_{i=\rho+1}^{\infty} U(i) = 1 - CDF(\rho). \tag{1.25}$$

Similar to Listing 1.5, we can also calculate the tail distribution of the user activities as shown in Listing 1.6. Note that we use a trick to calculate this using the built-in `cumsum` function: We reversed the frequency vector twice to simulate a cumulative sum from the end to the beginning of the vector.

Listing 1.6: Calculating the tail distribution of the fraction of users with more than a given number of edits. (`user_edits_in_timeframes.R`)

```
rev.buckets = within(rev.buckets, {
            # We reverse the vector twice since 'cumsum' adds up
            # from the beginning, and discard the very first bucket
            # since the CCDF is defined as a strict "greater". Finally,
            # we append a 0.0 value for the last element since there are
            # no users with more than the maximum number of edits.
            ccdf = c(rev(cumsum(rev(tail(count, -1)))) / total.users,
                       0.0)
      })
```

Figure 1.9 displays the tail distribution as we calculated it on Wikipedia edits. Again, we can notice a few consequences of the long-tailed distribution of the revisions: Only 15% of the users have more than 10 revisions in a month, and this drops to below 5% for 100 revisions. To inspect how many users have a large number of edits, it would make sense to also rescale the vertical axis so that we can see the smaller fractions as well, as we did in Figure 1.10.

When we perform this rescaling, we can notice something interesting: The tail distribution seems to follow a power law as well, just like the original probability distribution did (Figure 1.7). Can we find a reason for why this should be so? For this let's first recall that our power-law PDF for the number of revisions r can be described by the following relationship:

$$P(r) = Ar^{\gamma}, \tag{1.26}$$

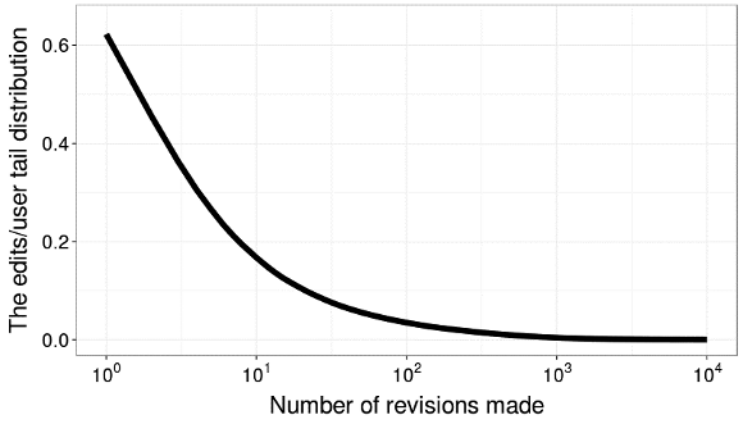

Figure 1.9: The tail distribution of the users' revisions: This shows what fraction of users had more revisions than a threshold. Comparing this with Figure 1.8, we can immediately see that the two functions add up to 1.

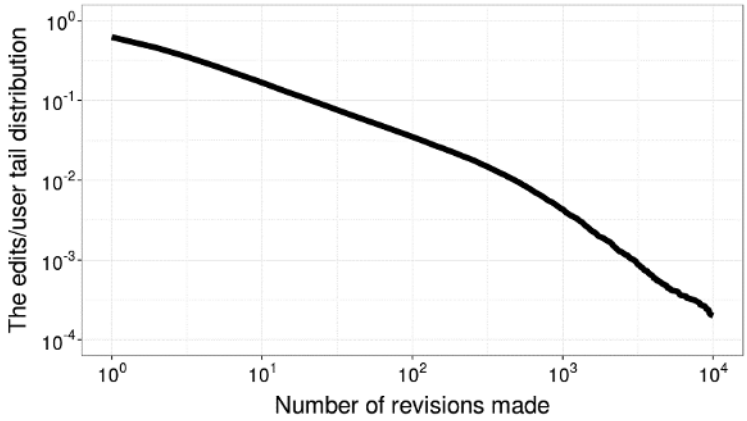

Figure 1.10: The tail distribution of the users' revisions again as in Figure 1.9, but this time on double-logarithmic axes. We can now see that, similar to the PDF, this tail distribution also follows a power law (or two power laws, given the slightly faster decay at the end as we can recognize by the steeper linear section of the plot starting at approximately $10^{2.5}$ revisions).

with an appropriate constant A that normalizes the area of the function to 1, and γ being the exponent of the power law. Then the tail distribution can be expressed as the tail sum over all values above the threshold: $CCDF(\rho) = \sum_{r>\rho}^{\infty} P(r)$.

Let's try to express this in a closed form by approximating this discrete sum with a continuous integral of the same function. This we can do assuming the function does not change *too* quickly because the integral is nothing else but an infinitely refined box covering of the integrand, and the discrete sum can be thought of as a rough covering of the continuous function. This trick is illustrated by Figure 1.11.

$$CCDF(r) = \sum_{i>r}^{\infty} P(i) \approx \int_r^{\infty} Az^{\gamma} dz = \frac{A}{\gamma+1}\left[z^{\gamma+1}\right]_r^{\infty} = \frac{A}{\gamma+1}(0 - r^{\gamma+1})$$
$$= -\frac{A}{\gamma+1} r^{\gamma+1}. \tag{1.27}$$

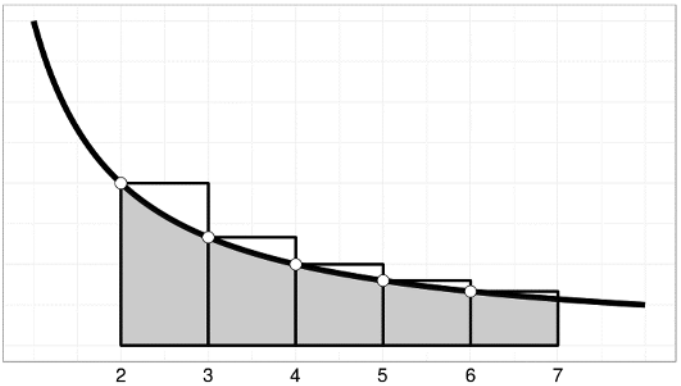

Figure 1.11: Imagine that we are summing up the values of a function indicated by the five white dots. This is the same as the sum of the areas of the white (unfilled) rectangles. However, we can approximate this area by taking the integral of the underlying continuous function as well (shaded by gray): $\sum_{i=2}^{6} f(i) \approx \int_{x=2}^{7} f(x)dx$. Although we cover a slightly smaller area as can be seen from the figure, the error we're making is negligible compared to the actual differences between our model and the actual social media system.

You may have noticed that there was one bold assumption in the previous calculation: that $\infty^{\gamma+1} = 0$. This is, however, true, as long as $\gamma < -1$ because in this case we raise ∞ (think of it as a very large number) to a negative power, which is the same as $1/\infty^{-(\gamma+1)}$. For this to converge to 0, $-(\gamma + 1) > 0$ is required, hence our stipulation that $\gamma < -1$. (Otherwise, if this is not the case, the theoretical power law would not have a finite area under it, and we could not have assumed the power law extends up to infinity. It must have been "truncated" so that we can have a finite area under the probability density function.)

To summarize Equation 1.27 in plain terms: If we can assume that our probability distribution function for user activities follows a power law with an exponent γ, then the tail distribution will also follow a power law, with an exponent $\gamma + 1$.

There is moreover one more thing we can read from Figure 1.10, which is that the tail end of the CCDF decays with a steeper exponent beyond approximately 2.5 decades ($10^{2.5} \approx 316$) than the earlier part of the distribution. This means that the original PDF of Figure 1.7 *also* drops faster at the end than earlier, but we could not really have seen this fact from that plot. This is immediately one benefit of transforming the PDF into a tail distribution: For a power law, we essentially "smoothed" out that function, as here we obviously do not observe the spreading out of the tail as happened for the original PDF due to the small sample sizes at the end.

However, we can discover a further non-obvious fact of the tail of the PDF. If we plot the tail distributions of Periods 2 and 3 separately, we notice that where this regime of faster decay starts is dependent on which period we are looking at: For this period, Period 1, it was $10^{2.5}$, whereas if we had taken Period 3, which is about three times longer than Period 1, we would have seen it begin later, at 10^3. (We leave it to the reader to check this.) The point of onset of this faster decay is sometimes called a *cutoff*, especially if it's even more pronounced than what we see in this example. This phenomenon is called the *finite size effect*, which arises because we can observe only a constrained snapshot, instead of an infinitely long period as it would be for the "ideal" system. However, as we noted, the threshold where the finite size starts to show up keeps shifting to higher activity values as we increase the length of the observation period.

The Long Tail in Human Activities

Over the last few pages we saw that the power-law distribution that describes users' activities reveals a large diversity among the users, showing that most of them are rather inactive, but there are a few who are immensely more active than the rest. Another question that comes up frequently for characterizing user contributions is the following: If we rank users by increasing activities, what percentage of actions comes from the most active users? Conversely, what percentage of contributions can be attributed to the least active users?

To answer these questions we'll order users by the number of revisions they have, and will look at what fraction of all edits have the most active 10% of users made. Obviously, we can look at any percentage of users, but for this example we've chosen 10%. What we need to do, therefore, is to order the revision counts of users from highest to lowest, so we will get the most active users in the beginning of the vector, and the least active ones at the end. We call the position of a user in this vector a user's *rank*, so the user ranked 1 is the user with the largest number of revisions, the user ranked 2 is the second most active user, and so on. Then we can calculate the cumulative sum over this vector, arriving at the number of edits users made up to a certain rank (Listing 1.7).

Listing 1.7: To calculate the number of edits that the most active users make, we first order the users by decreasing activity counts so that we can cumulatively sum up their number of revisions. (`user_edits_in_timeframes.R`)

```
range.considered = subset(revs.in.periods, range1 > 0, select='range1')
names(range.considered)[1] = 'revisions'
ordered.activities = range.considered[order(range.considered$revisions,
                decreasing=TRUE),]
total.revisions = sum(ordered.activities)
tail.fractions = data.frame(user.rank=(1 : length(ordered.activities)),
        fraction=cumsum(ordered.activities) / total.revisions)
```

The plot for the fraction of edits made by the most active users is shown in Figure 1.12: It's surprising that only a few of the users seem to be responsible for most of the edits. In fact, it looks like only 10 users make approximately 12% of all edits; and the top 100 users make approximately 29%. This is tremendous; of the 134.8k users active in this period, only 0.07% is making nearly one-third of all revisions!

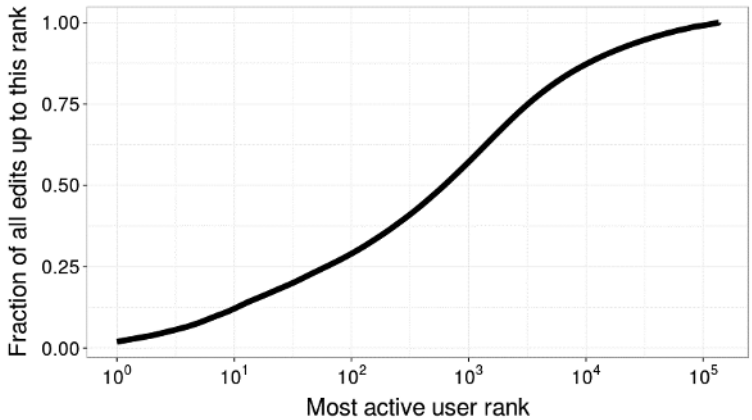

Figure 1.12: The fraction of all edits made by users up to a certain rank. We rescaled the axis for the user rank logarithmically.

At this point we should realize that it's humanly impossible to have as many edits in only one month as the top users have. For instance, Table 1.1 lists the heaviest "users" and their revision counts. It's apparent from this table that the users with the most edits are "bots" (an abbreviation for "robots"): Wikipedia agent programs that perform some automatic maintenance or fix on the encyclopedia pages. "CLueBot NG," the most active bot, is for instance a bot that tries to detect and prevent vandalism to pages, whereas "Addbot" is a bot that performs a diverse set of clean-up and maintenance tasks on pages. In the case of Wikipedia,

we may have an idea which accounts are bots, but this is not always possible in every social media service. (A list of registered bots can be found here: `http://en.wikipedia.org/wiki/Wikipedia:Bots/Status`.) In many cases, automated accounts may appear as legitimate users; on Twitter, for instance, it's often the case that Internet-connected appliances, news headline relays, or algorithmic accounts replying to or quoting other users are also present on the service as legitimate (and useful) accounts. At other times, these active accounts belong to spammers, in which case it's doubtful we can call the account useful.

Table 1.1: The Most Active Wikipedia Accounts in Our Period 1, January 2013

ACCOUNT NAME	PERIOD 1 EDITS	AVERAGE SECONDS BETWEEN TWO EDITS
ClueBot NG	80003	33
EmausBot	79357	34
Addbot	63136	42
BG19bot	47897	56
WP 1.0 bot	46910	57
Cydebot	46067	58
Yobot	39489	68
Makecat-bot	31228	86
ZéroBot	30351	88
AnomieBOT	30122	89
Xqbot	30111	89
BD2412	25710	104

The list in Table 1.1 was generated by the (UNIX) shell command: `zcat data/wikipedia/user_edits_in_timeframes.tsv.gz | sort -n -r -k 2 -t "$(printf '\t')" | head -12`. Often it saves time to run these "one-liners" in a terminal window.

There's usually no obvious way to know what accounts are automated, unless the service whose data we're analyzing provides a way to identify them. To see the effect of removing these bot accounts from Wikipedia, we can simply exclude all known accounts listed as bots from our activity data. Because even after this we found accounts that were apparently bots, we examined the top accounts by hand and also removed these from among the top 100 most active users (Listing 1.8).

Listing 1.8: This simple piece of code removes all known (and assumed) bot accounts from our data frame. (`user_edits_in_timeframes.R`)

```
bots = read.table(gzfile('wikipedia_robots.txt.gz'),
         col.names=c('account'))
revs.in.periods =
         revs.in.periods[!(revs.in.periods$account %in% bots$account), ]
```

We can also generate a plot similar to Figure 1.12 with the bot accounts removed; however, we will not repeat that in print because it looks similar to this figure. With the bots removed, the top 10 users have 6% of the edits, and the top 100 have 18% (cf. 12% and 29% before removing the bots, respectively). This is a significant change in the percentages, but the proportions are still huge for such a small number of users. Only a vanishingly small fraction of the users is responsible for a majority of all activities on the service. We can also describe this surprisingly unbalanced split more generally. What could we expect in other social media systems, for which the activity distribution is also a power law, similarly to Wikipedia?

Long Tails Everywhere: The 80/20 Rule (*p*/*q* Rule)

You have seen that there's a strong imbalance in the engagements of users: A few of them are very active, whereas the majority of them have relatively low engagement counts. Can we find a way to describe this observation quantitatively as well?

In 1906, the Italian economist Vilfredo Pareto recognized that (in his time) 80% of the land in Italy was owned by only 20% of the population: 20% seems to be a relatively low number, whereas 80% is high. Later, this observation was generalized and was called the *Pareto principle*. In the original incarnation of the Pareto principle it is posited that 80% of the consequences are attributable to 20% of the causes. (Note that the 80% and 20% in this phrasing don't have to add up to 100%, because they refer to different entities; it's just a slightly misleading "coincidence" that the principle was formulated in this way.) In a more general setting, you could say that a fraction p of the consequences are coming from a fraction q of the effects, where p is relatively close to 1, and q is relatively close to 0. We already know that applying it to the metrics we have considered for our social media services, this will certainly be the case. The question is now how large p and how small q will be for our typical cases.

What we would like to do, then, is find the relationship between the *fraction of the most active users* and the *fraction of activities they are responsible for*. The information we have available for each user, which should be enough for us, is how many activities they have for a given period. We already know that the number of users with r revisions follows a power law: $U(r) \propto r^{\gamma}$. The trick we will use,

as before, is to approximate the sums of discrete variables by integrals over the functions of their expected values. Although strictly speaking the discrete sums are not equal to the integral of the approximating function (a graphical illustration can be seen in Figure 1.11), the difference is small, especially for ranges of a function where it changes slowly (such as at the tail end of the power law function). Also, these calculations are always going to be a well-informed model of reality, so as long as we can find a reasonably good description of online user behavior, we're fine with making good numerical approximations such as this as well.

So to repeat: We will express the total number of edits made by the most active users. The most active users are those whose number of revisions are the highest; let's say they have more than R number of revisions. Because the number of users having r revisions is $U(r) = U_0 r^\gamma$ (as before with a normalizing constant U_0), and because we can approximate the discrete distribution by a continuous one and the sum of users by an integral over the continuous distribution function, the number of users having more than R revisions will be $N_u(R)$ as shown in Equation 1.28:

$$N_u(R) = \int_{r=R}^{\infty} U_0 r^\gamma \, dr = \frac{U_0}{\gamma+1}\left[r^{\gamma+1}\right]_R^\infty = -\frac{U_0}{\gamma+1} R^{\gamma+1}. \tag{1.28}$$

We had to assume again that when we evaluate the antiderivative at ∞, we get 0. The condition for this, again, is that $\gamma < -1$, which holds for our case, and this is what we also see in practice for social media services in general.

Now, what can we say about the total number of edits that these most active users make? For one given revision count, r, the number of edits made by all the users who have exactly this many edits is $rU(r)$. To calculate the total number of edits made by users with more than R revisions is simple, and similar to the calculation we just performed. However, we will not immediately try to integrate up to infinity, but only up to a maximum revision count, R_m. The reason for this will become clear in a moment. $N_a(R)$ will be the total number of activities (revisions) by these users:

$$N_a(R) = \int_{r=R}^{R_m} r \cdot U_0 r^\gamma \, dr = \frac{U_0}{\gamma+2}\left[r^{\gamma+1}\right]_R^{R_m} = \frac{U_0}{\gamma+2}\left(R_m^{\gamma+2} - R^{\gamma+2}\right). \tag{1.29}$$

We can see why we couldn't automatically assume that we can integrate up to infinity: having $\gamma \approx -1.8$ for Wikipedia, $\gamma + 2 > 0$, so substituting R_m as the upper limit would make $N_a(R)$ grow without bounds if we did not stop at some point with it. However, how can we have a good model assumption for what R_m's reasonable value is? Note that at this point we are slightly at the mercy of our assumptions, and instead of getting back exactly what we can measure in an

actual system (Wikipedia), what we would like to do is explain qualitatively *why* we are seeing such huge skews in the activities of the topmost users. Let's therefore do our best, and say that R_m is the revision count beyond which we do not expect to see more than *one* user altogether; at this point the power-law model would tell us that there is "fewer" than one user expected with more than so many edits, so we can conveniently stop counting. Therefore, when we plug R_m into $N_u(r)$, we expect to see 1 as the result:

$$N_u(R_m) = -\frac{U_0}{\gamma+1} R_m^{\gamma+1} = 1. \tag{1.30}$$

From this we can express R_m as

$$R_m = \left(-\frac{\gamma+1}{U_0}\right)^{\frac{1}{\gamma+1}}. \tag{1.31}$$

We are almost finished; we just need to put R_m back into Equation 1.29. Before we do that, let's revise our original goal: to express the number of edits made by the topmost users. This means that we need to get rid of R because this was just a "helper" parameter for us to express both N_u and N_a. We can then invert Equation 1.28 to get R:

$$R = \left(-\frac{\gamma+1}{U_0} N_u\right)^{\frac{1}{\gamma+1}} = \left(-\frac{\gamma+1}{U_0}\right)^{\frac{1}{\gamma+1}} N_u^{\frac{1}{\gamma+1}}. \tag{1.32}$$

This way we have now both R_m and R to complete Equation 1.29 so that we can get N_a as a function of N_u. In the end, what we'll get is the following:

$$N_a(N_u) = \frac{U_0}{\gamma+2}\left(-\frac{\gamma+1}{U_0}\right)^{\frac{1}{\gamma+1}}\left(1 - N_u^{\frac{\gamma+2}{\gamma+1}}\right). \tag{1.33}$$

This is the relationship we wanted to derive. We can see that the first terms are just a constant. This constant, according to this model, should be the *total number edits*, as setting $N_u = \infty$ guarantees that we get back all the activities. (Setting $N_u = \infty$ intuitively means that we're taking all the users.) $(\gamma + 2)/(\gamma + 1) < 0$, so raising a large number to a negative power will make it small. Therefore, for the sake of our discussion, we can simplify this expression to show only the salient term that depends on the variable N_u:

$$N_a(N_u) \propto 1 - N_u^{\frac{\gamma+2}{\gamma+1}}. \tag{1.34}$$

N_u is the exact same user rank that we used to plot Figure 1.12 because what we just calculated is the number of edits made altogether by the *most active* N_u users.

The reason we went through these lengthy calculations was to expose the reader to the oft-cited *long tail of human behavior*, where only a small fraction of the participants is responsible for most of the actions. To recap what you saw earlier in this chapter, we realize that this is the consequence of two simple factors in social media:

- That the user activity distribution is unchanged over time.
- That users make proportionally more edits by the same constant multiplier between two sampling windows, independently of how active they are.

We know that any time we observe similar regularities in any social media system (or for that matter, any other kind of a natural system whose statistical properties are similar to what we have seen in Wikipedia), the rules for the skewed activity distribution should apply.

What are the consequences of the relationship we discovered in Equation 1.34? To see this more clearly, we can plot this function for some choices of γ, as we did in Figure 1.13. One thing we can immediately see is that the function is rather sensitive to the value of γ. Slight changes in the exponent yield vastly different results, even when we consider the top 100 users or so only. γ, in reality, means the exponent that can be fitted to the *tail end* of the user–activity distribution: According to Figure 1.10, this exponent for Wikipedia is greater (by absolute value) at the end of the distribution than in the body. Remember, the tail distribution of Figure 1.10 is closely related to the original $U(r)$ distribution with a γ exponent: The slope of the locally fitted straight line to the function in Figure 1.10 yields γ + 1. So, as we include more and more users from among the most active ones, any small local change in the γ exponent will cause strong deviations from the pure model that assumed a constant for γ.

Figure 1.13 shows that none of the theoretical curves with a fixed γ is a great fit to the actual measurement (which for Wikipedia is shown by the line with a lighter shade). But it's expected that when we have so few users to fit a model to (we are talking about the top 100 or 1,000 most active users!), the changes in the "local value" (at a given user rank) of the γ exponent will be amplified by the $N_a(N_u)$ function, which is extremely sensitive to it. Actually, consider that just by removing the most active 10 (about 0.01% of all) users we could have erased *one-tenth* of all edit activities! It's hard to overstate this fact, and therefore it is understandable that at this minuscule scale any prediction will have large uncertainties. To put it in a different way, we are trying to predict how many edits only 10 or 100 users will have; although these are the most active users, any slight relative change in their individual production rates will show up in the total number of activities.

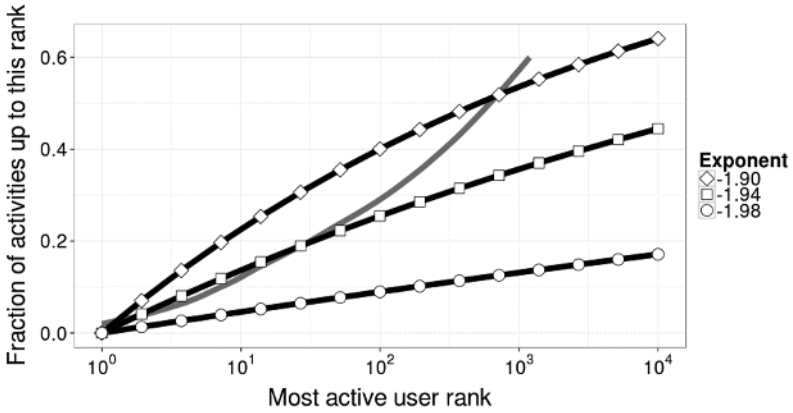

Figure 1.13: The expectation for the fraction of activities generated by the most active users, in a hypothetical system whose user activity distribution perfectly follows the $U(r) \propto r^{\gamma}$ law. The horizontal axis shows how many of the most active users we consider, and the vertical axis is the proportion of activities attributable to them. We show three examples with different γ exponents. The lighter unmarked line shows the real measurements for Wikipedia, which happens to be the initial part of the line in Figure 1.12.

The lesson we learned is far-reaching when we are designing or provisioning for a social media service. Due to this general characteristic that we can observe in systems affected by human behavior, it is always true that there will be vast differences in activity levels between individuals.

Online Behavior on Twitter

Do our conclusions hold more generally for other kinds of online social systems? What we have seen until now is strictly speaking only true for Wikipedia. However, you will learn shortly that, at least in terms of large-scale trends, you should expect to find similar behavior in most social media systems. It is nevertheless the users who generate activities on the online services, so it is the statistical properties of the underlying human behavior that you can measure with most of these metrics exhibited through the different services.

For this reason, let's turn to a different kind of social sharing service, Twitter. In Twitter, users can send out status updates of at most 280 characters in length, and other users, who "follow" the sender, will receive these short messages in their so-called timeline. The service provides an API (`https://developer.twitter.com/en/docs.html`) for third-party applications that can read and manipulate timelines and various Twitter objects on behalf of the user. Due to this easy extensibility and API access, we can also download example datasets that we can analyze for user activity.

Similarly, as the number of edits characterize the activity of Wikipedia users, we would like to measure the activity of Twitter users in the example in this section by the *number of Tweets* that they have sent in a given period of time. The API lets us download all the Tweets that a given user has sent in the recent past; therefore, if we have a list of valid IDs for users we can query the API in a loop to return all their most recent Tweets. Courtesy of Twitter, we have a list of *randomly chosen* user IDs belonging to normal users that we can continue working with in this chapter. (A normal account is one that has not been deleted by the user or has not been suspended for violating any terms of the service, such as spamming.)

Retrieving Tweets for Users

We need to iterate through the list of user IDs and ask the API for their most recent Tweets. We would like to work with 4 weeks of data, and because Twitter returns only a limited number of the most recent Tweets for each request to keep the response size under control (this count is 200), we may need to issue more than one request to retrieve all the Tweets for a user for the last 28 days. We also must watch out for *rate limiting*, which is the maximum number of requests in a service-specific time window. Rate limiting is employed by virtually every popular Web service's API to maintain a predictable quality of service level for everyone. Without this, we could reasonably expect that there would be a few API applications or users who would consume most of the bandwidth or server capacity. In fact, by now you might expect that the query activities of these apps could likely have a long tail, having learned about the presence of a strong skew in activity distributions from the previous sections, so a few of the most aggressive clients would dominate resource usage in the absence of such limits.

Since response throttling, retry fallbacks, and rate limiting are common notions in third-party API access patterns, we list the corresponding Python code in Listing 1.9 that implements these to download the Twitter data. This script fetches and records the Tweet IDs for the last 4 weeks for a predefined list of users. The longer we let the script run, the more users we will cover, and the more data we will have as well. This is a bare minimum example for how to connect to and download data from a Web service that provides API access and rate limiting. To simplify the OAuth authentication and response handling, we utilize the `tweepy` external library.

Listing 1.9: The Python code to consume the Twitter API to get the latest Tweets for a list of valid users. (`get_users_tweets.py`)

```
import sys, gzip, time, tweepyfrom datetime import datetime, timedelta

# The consumer and access keys & secrets for the Twitter application.
# See https://developer.twitter.com/en/docs/basics/authentication
```

```
# /overview/oauth
# on how to access these credentials.
CONSUMER_KEY = '<consumer key from the Twitter dev site>'
CONSUMER_SECRET = '<consumer secret from the Twitter dev site>'
ACCESS_KEY = '<access key from the Twitter dev site>'
ACCESS_SECRET = '<access secret from the Twitter dev site>'

# The maximum number of Tweets we can ask for in one request.
# See https://developer.twitter.com/en/docs/tweets/timelines
# /api-reference/get-statuses-user_timeline.html
MAX_ITEMS_PER_REQUEST = 200

# The file where we store a list of valid Twitter user IDs.
USER_LIST = 'data/twitter/user_handles_sample.gz'
# The result file
OUTPUT_FILE = 'data/twitter/tweets_per_user.tsv'

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_KEY, ACCESS_SECRET)
api = tweepy.API(auth)

# The start date and time of our data collection; 28 days before now.
start_day = datetime.utcnow() - timedelta(days=28)

user_list_file = gzip.open(USER_LIST, 'r')
output_file = open(OUTPUT_FILE, 'w')
for user_id in user_list_file:
    user_id = user_id.rstrip()
    # The ID of the earliest Tweet in the result batch.
    earliest_tweet_id = None
    while True:
        try:
            if earliest_tweet_id is None:
                # The first request for the user
                timeline = api.user_timeline(
                    id=user_id, include_rts=True,
                    count=MAX_ITEMS_PER_REQUEST)
            else:
                # There are possibly more recent Tweets than
                # MAX_ITEMS_PER_REQUEST.
                timeline = api.user_timeline(
                    id=user_id, include_rts=True,
                    count=MAX_ITEMS_PER_REQUEST,
                    max_id=earliest_tweet_id)
```

```
            except Exception as e:
                if e.response.status == 429:
                    # If we are rate limited, wait 60 seconds before
                    # retrying. See https://developer.twitter.com/en/docs
                    # /basics/response-codes.html
                    time.sleep(60)
                    continue
                else:
                    # In any other case do not retry to load user data.
                    # This may be changed to cover other error conditions.
                    print 'Could not access', user_id
                    break
            tweet_count = 0
            found_early_tweets = False
            for tweet in timeline:
                if tweet.created_at >= start_day:
                    output_file.write('\t'.join( \
                        [str(f) for f in [user_id, tweet.id,
                                          tweet.created_at]]))
                    output_file.write('\n')
                    output_file.flush()
                else:
                    found_early_tweets = True
                if earliest_tweet_id is None or \
                tweet.id < earliest_tweet_id:
                    earliest_tweet_id = tweet.id
                tweet_count += 1
            if tweet_count < MAX_ITEMS_PER_REQUEST or found_early_tweets:
                # Finished with this user's Tweets if no more to download
                # or we got back before start_day.
                break
    user_list_file.close()
    output_file.close()
```

When we have collected enough users, we can look at their activity distributions. We have retrieved the Tweet counts for the random set of users for a one-week (Period 1), a two-week (Period 2), and a three-week period (Period 3). Again, we chose overlapping periods starting on the same day, like we did for Wikipedia. Figure 1.14 shows the probability distribution functions for the number of users who tweeted a given number of times in the three periods.

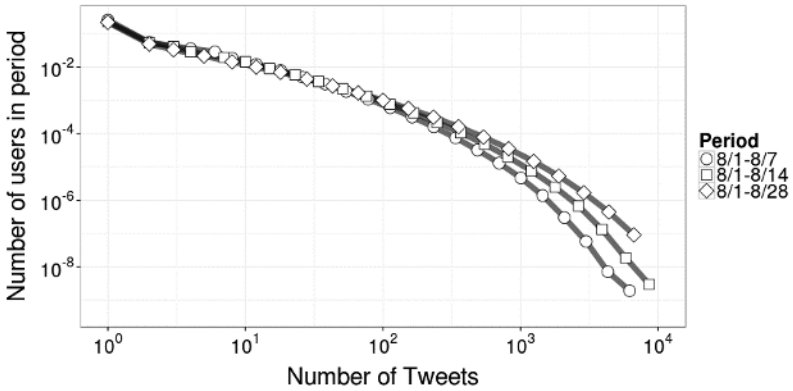

Figure 1.14: The probability distribution function for the number of users with a given number of Tweets sent, for a one-week, a two-week, and a three-week period, respectively.

Logarithmic Binning

Comparing Figure 1.14 to its Wikipedia counterpart in Figure 1.7, the first thing to notice is that the data points appear to be sparser, more spaced out, and positioned more equally from each other than in the previous plot. The reason for this is that in this case, to illustrate a common way of aggregating and smoothing a distribution that is being plotted on a logarithmically rescaled horizontal axis, we used *logarithmic binning*. Where previously our buckets were the naturally occurring integer activity counts (as in Figure 1.7), here we created buckets whose length is *not uniform* along the horizontal axis. However, if we carefully inspect any one of the three curves corresponding to a given period in Figure 1.14, we see that the data points (corresponding to buckets) are equally spaced from each other. What this means on the logarithmic scale is that their distance in the log-space is equal; therefore, on a linear scale the positions of the bucket boundaries are constant multiples of each other. In R, it's easy to create bucket boundaries that satisfy this condition: First, we create equidistant bins in log-space, and then transform them back to the natural scale with the exponential function (Listing 1.10).

Listing 1.10: Create bucket boundaries that we can use in `hist` to create histograms with increasing bin sizes. The range is defined by `from` and `to`, and `bucket.count` is the number of bins we want to create.

```
buckets = exp(seq(log(from), log(to), length.out=bucket.count + 1))
```

Figure 1.15 illustrates the relative sizes of these buckets on a linear scale. What this kind of binning means is that as we progress toward higher and higher activity counts, we will have buckets that are longer and longer, and therefore

able to capture an ever-increasing range of activities. However, we also know that if the distribution we are plotting is approximately a power law, we will have fewer and fewer users in the high activity ranges, so increasing the bucket sizes counteracts the diluting statistics, to the effect that we will also have a substantial number of data points in the buckets in the upper range. In fact, the challenge with *not* increasing the bucket sizes for Figure 1.7 was exactly that the tail ends of the distributions became noisy because in many cases we found only one or two users with a given number of edits (in contrast to the head part of the distribution, where we have a large number of users with just one or two revisions).

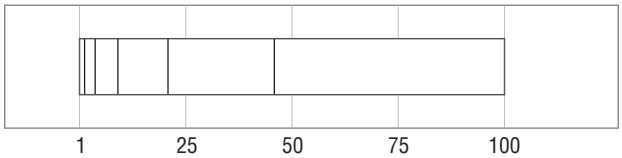

Figure 1.15: Logarithmic binning illustrated. In this example our original range is 1 … 100, and we split this range up into six buckets that increase exponentially in size: The length of every bucket is a constant multiple of the previous one. You can see that in the beginning the buckets are short, whereas their size is growing rapidly on this natural, linear scale.

However, we would like to calculate the probability that a randomly selected user will fall into a given bucket. Obviously, the larger buckets we take, the more chances we will have to capture users within this bucket. So to arrive at an approximation for the probability distribution of the underlying random process, we need to divide the number of users that we empirically count into one bucket by the *length* of the bucket (as the `hist` R command does this automatically when we consider the `density` field in the result).

User Activities on Twitter

Turning back to user behavior on Twitter, it's also apparent from Figure 1.14 that, in contrast to Wikipedia, for the three different time windows the probability distribution functions do not overlap, at least not on the higher end of Tweet scale. The consequence of this is that the relationships on log-log scales between the Tweet counts and the corresponding user counts are not linear, and the distributions *do not* exactly follow power laws. We have seen that the stability of the distribution functions over time was a requisite for us to see a power-law behavior—on Twitter, although our model *approximately* holds, there are also measurable deviations from it. In this case, as we take longer and longer observation periods, we are seeing more of the high activity users as well, and the high end of the activity distribution shifts up. On this note, the second

benefit of the logarithmic binning is that it lets us actually discern the differences among the high activity users: We can see in Figure 1.14 that even beyond 10^3 Tweets we can meaningfully explore the tails of the probability distributions and see their differences. In the case of linear binning, the smallest number of users we might get per bucket is one, and as such that is the lower bound for our estimated probability in a bucket (refer to Figure 1.7). For logarithmic binning the bucket sizes themselves grow, so we can estimate the likelihood of progressively smaller probability events as well.

Furthermore, we can check one more property of user activities for Twitter users: how many more Tweets they send if we increase the length of our observation period. Figure 1.16 is the counterpart of Figure 1.6, and it essentially implies what we have found previously: that as we consider another time window (Period 2 and 3, respectively), the average number of Tweets sent by a user who sends q number of Tweets in Period 1 is a constant multiple of q. (Again, this is *almost* true; if we take a closer look, we can see that the data does have some curvature, and it is not increasing strictly monotonically for the first few data points.)

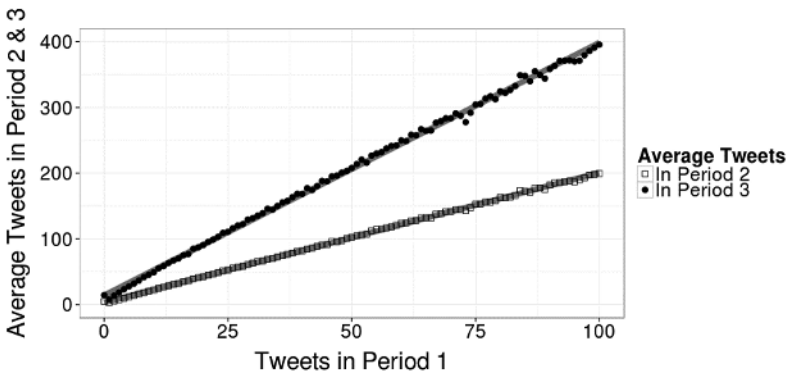

Figure 1.16: The average number of Tweets that Twitter users sent in Periods 2 and 3, as a function of the number of Tweets they sent in Period 1, respectively.

In summary, we have seen that our model and quantitative explanations that fit Wikipedia users well will not be unconditionally good descriptions for Twitter users with a high degree of accuracy; however, such idiosyncrasies are present in every kind of social media system. In fact, our point with the Twitter example was that while in general these model assumptions hold to a large degree, if we need more accurate descriptions, we need to refine the models. After all, certain product decisions or intentional limitations can very well change the user behavior that we can observe in the end. Think of for instance the upper bound on the number of social connections that certain social networks impose: In this case, obviously, we will not see anyone with

more than that many connections. However, the principles we have studied are general enough that notwithstanding these system-specific constraints, we will normally observe large variations among user activity levels that can be well described by the laws and regularities that we discovered in the previous sections of this chapter.

Summary

In this chapter, we have talked about the large degree of diversity that we can observe in the activities of users. Understanding that they can be characterized by general statistical laws across different types of social media systems suggests that these are the consequences of more universal human behavioral traits. Specifically, this chapter discussed the following:

- You saw that although most users use online social media rather infrequently, a few are very avid users. The counts of activities within a given time frame follow power-law distributions.
- When we consider the distributions across different time windows, they are similar to each other in the head of the distributions (characterized by the exponent of the power law). The cutoff where they diverge depends on the length of the time window. This is, of course, only true if there're no major changes to how users use the service over time (no major site redesigns, competing service, or rapidly increasing growth).
- The long-tail activity distributions give rise to surprising facts for the user metrics. We cannot say that there's an "average" or "typical" user behavior: The averages of the activity metrics are subject to large variances whenever we measure them, and we must be mindful about the strong effect on the means of the outliers in the distributions that are always present.
- In practice, a small fraction of the most active users can influence our averages strongly. If for some reason these users don't come back, we could see a significant drop in our metrics, even though the behavior of most of the users may stay unchanged, for example.
- Therefore, if our goal is concentrated on measuring total activities, we should focus on understanding how our most active users behave. Because there are relatively few of these, even going through them "by hand" can give insights.
- If we want to focus on describing how active user counts change for a time window, it's most useful to understand the behavior of the least active users. Namely, these are the users who are likely to be the most numerous in our social media service.

The focus of this chapter is to understand and describe the most important features of the user activity statistics. We did this by thinking of the users in isolation from each other, as if each individual acts independently of each other. This is certainly true to a large degree; however, the next chapter explores another defining function of social media systems: The networks that users create to express their interest in each other's activities.