Psychological Methods
2016, Vol. 21, No. 4, 493–506

1082-989X/16/$12.00 http://dx.doi.org/10.1037/met0000105

Mining Big Data to Extract Patterns and Predict Real-Life Outcomes

Michal Kosinski, Yilun Wang, Himabindu Lakkaraju, and Jure Leskovec
Stanford University

This article aims to introduce the reader to essential tools that can be used to obtain insights and build predictive models using large data sets. Recent user proliferation in the digital environment has led to the emergence of large samples containing a wealth of traces of human behaviors, communication, and social interactions. Such samples offer the opportunity to greatly improve our understanding of individuals, groups, and societies, but their analysis presents unique methodological challenges. In this tutorial, we discuss potential sources of such data and explain how to efficiently store them. Then, we introduce two methods that are often employed to extract patterns and reduce the dimensionality of large data sets: singular value decomposition and latent Dirichlet allocation. Finally, we demonstrate how to use dimensions or clusters extracted from data to build predictive models in a cross-validated way. The text is accompanied by examples of R code and a sample data set, allowing the reader to practice the methods discussed here. A companion website (http://dataminingtutorial.com) provides additional learning resources.

Keywords: computational social science, big data, digital footprints, R, personality

Human activities are increasingly mediated by digital products and services (Lambiotte & Kosinski, 2014). Individuals use social networking sites and messaging apps to communicate, make payments using online platforms and credit cards, and stream digital media. Additionally, they are virtually inseparable from wearable devices such as fitness trackers and smartphones. The growing immersion in digital environments and dependence on digital devices imply that people's behaviors, communication, geographical location, and even physiological states can be easily recorded, producing large samples of *digital footprints*. Such footprints include web browsing logs, records of transactions from online and offline marketplaces, photos and videos, global positioning system location logs, media playlists, voice and video call logs, language used in Tweets or e-mails, and much more.

The unprecedented availability of large samples of digital footprints, combined with computing power and modern statistical tools, offers great opportunities for social science (Lazer et al., 2009). Big data samples facilitate the discovery of patterns that might not be apparent in smaller samples, thereby helping to reduce sampling errors typical in social science studies. Additionally, many online environments provide access to digital footprints produced by diverse populations. This can address another major challenge in social science, namely its overreliance on samples that are small, composed of (largely female) students, and disproportionately WEIRD (i.e., Western, educated, industrialized, rich, and democratic; Henrich, Heine, & Norenzayan, 2010). Finally, the high statistical power typical of big data analyses offers an opportunity to address the replicability crisis in psychological sciences (Open Science Collaboration, 2015).

Recent research demonstrates that digital footprints can be successfully employed to study important psychological outcomes ranging from personality (Youyou, Kosinski, & Stillwell, 2015), language (Schwartz et al., 2013), and emotions (Kramer, Guillory, & Hancock, 2014) to cultural fit (Danescu-Niculescu-Mizil, West, Jurafsky, Leskovec, & Potts, 2013) and social networking (Ugander, Karrer, Backstrom, & Marlow, 2011). Unfortunately, an interest in studying digital footprints, as well as the necessary skills to do so, are relatively rare among social scientists. Consequently, such research is increasingly ceded to computer scientists and engineers, who often lack the theoretical background in social science

Michal Kosinski, Graduate School of Business, Stanford University; Yilun Wang, Himabindu Lakkaraju, and Jure Leskovec, Department of Computer Science, Stanford University.

We thank the editor and reviewers for their excellent advice. Also, we would like to thank Jose Hernandez-Orallo, Thore Graepel, Annalyn Ng, Isabelle Anne Abraham, Robert Wilson, Sam Gosling, Gabriela Harari, and Steffen Fohr for providing us with very useful feedback on the previous versions of this article. MyPersonality.org has generously provided the sample data set used in this article. This work was supported by a Robert Bosch Stanford Graduate Fellowship, a Google Faculty Research Award, the National Science Foundation, the Defense Advanced Research Projects Agency (DARPA), and the Stanford Center for the Study of Language and Information. This article is accompanied by a companion website (http://dataminingtutorial.com) containing the sample data set used here, examples of code, and additional resources. Methods used to preprocess the data and build prediction models are similar to those employed by Kosinski, Stillwell, and Graepel (2013) to predict a broad range of psychological traits and real-life outcomes from a large sample of digital footprints.

Conflict of interest statement: Michal Kosinski co-owns the rights to the myPersonality.org database.

Correspondence concerning this article should be addressed to Michal Kosinski, Graduate School of Business, Stanford University, 655 Knight Way, Stanford, CA 94305. E-mail: michalk@stanford.edu

and training in ethical standards pertaining to human subjects research (Buchanan, Aycock, Dexter, Dittrich, & Hvizdak, 2011; Hall & Flynn, 2001).

This article aims to address this issue by providing an accessible tutorial for social scientists seeking to benefit from the availability of big data sets. We focus on two complementary analytical approaches. First, we show how to employ cluster analysis and dimensionality reduction to extract patterns from large data sets. Second, we use the extracted patterns to build models aimed at predicting psychological outcomes. We argue that these methods can be used to study human psychology and behavior in the same manner in which similar techniques are used to extract insights from small data sets. Additionally, predictive models based on digital footprints can be used to develop diagnostic tools and psychometric measures useful for psychological research and practice (Kosinski, Stillwell, & Graepel, 2013).

The following sections describe the typical steps involved in studying big data sets: data preprocessing, dimensionality reduction, and construction of predictive models. We put particular emphasis on the issues rarely encountered when working with small data, such as the challenges related to computation time and memory requirements. To enable the readers to practice the methods discussed here, the text is accompanied by the examples of R code (R Core Team, 2015) and a sample data set. A companion website (http://dataminingtutorial.com) contains the sample data set, code used to generate the figures included in this article, additional examples of R code, and links to other useful resources. This text is aimed at readers who have a basic familiarity with R; if you have not previously used R, we recommend that you start by reading the official introduction to this powerful language for statistical programming (http://cran.r-project.org/doc/manuals/R-intro.pdf).

Big Data Sets of Digital Footprints

This section focuses on importing, storing, and preprocessing large samples of digital footprints. Many big data sets are freely available online or can be obtained from the companies and institutions collecting and storing them. A popular example, the myPersonality.org database[1] (Kosinski, Matz, Gosling, Popov, & Stillwell, 2015), stores the scores from dozens of psychological questionnaires as well as the Facebook profile data of over six million participants. The Stanford Network Analysis Project website[2] (Leskovec & Krevl, 2014) hosts a wide range of data sets, including social networks, Tweets, and product reviews. Additionally, journal articles are now often accompanied by a publicly available data set. A recent article by Eichstaedt et al. (2015), for instance, is supplemented by a data set of U.S. users' Tweets aggregated at the county level.[3] Moreover, many online platforms (e.g., Twitter) contain large amounts of publicly available data that can be easily recorded. The companion website provides a collection of links to other potentially interesting data sets. The following hands-on subsection introduces the sample data set used in this tutorial, which was obtained from the myPersonality.org database.

Hands-On: Sample Big Data Set

The sample data set used in this article contains psychodemographic profiles of n_u = 110,728 Facebook users and their Facebook Likes.[4] For simplicity and manageability, the sample is limited to U.S. users. The following three files can be downloaded from the companion website:

1. *users.csv*: contains psychodemographic user profiles. It has n_u = 110,728 rows (excluding the row holding column names) and nine columns: anonymized user ID, gender ("0" for male and "1" for female), age, political views ("0" for Democrat and "1" for Republican), and five scores on a 100-item-long International Personality Item Pool questionnaire measuring the five-factor (i.e., Openness, Conscientiousness, Extroversion, Agreeableness, and Neuroticism) model of personality (Goldberg et al., 2006).

2. *likes.csv*: contains anonymized IDs and names of n_L = 1,580,284 Facebook Likes. It has two columns: ID and name.

3. *users-likes.csv*: contains the associations between users and their Likes, stored as user–Like pairs. It has $n_{u\text{-}L}$ = 10,612,326 rows and two columns: user ID and Like ID. An existence of a user–Like pair implies that a given user had the corresponding Like on their profile.

To load these files into R, use the following code:

```
users <- read.csv("users.csv")
likes <- read.csv("likes.csv")
ul <- read.csv("users-likes.csv")
```

You can inspect the dimensions of the resulting R objects and their first and last rows by using the following commands:

```
dim(ul)
head(ul)
tail(ul)
```

Note that Facebook users can like a given object only once. Thus, all of the user–Like associations have the same strength. Consequently, *object ul* (an R representation of the *users-likes.csv* file), storing the information about users' Likes, contains only two columns: user ID and Like ID. In the context of other types of digital footprints, a third column describing the strength of the user–footprint relationship might be necessary. A data set describing language use, for example, may require a third column recording the total number of times that a given person used a given word.

Constructing a User–Footprint Matrix

Most types of digital footprints (such as web browsing logs, purchase records, playlists from online radios, or Facebook Likes) can be conveniently represented as a *user–footprint matrix*. A hypothetical example of a user–footprint matrix—the user–movie

[1] See http://mypersonality.org/.
[2] See http://snap.stanford.edu/data/.
[3] See https://osf.io/rt6w2/.
[4] Facebook users employ *Likes* to display positive associations with entities such as products, sports, musicians, books, restaurants, websites, photos, or friends' status updates. Likes represent a generic class of digital footprints, similar to web search queries, web browsing histories, or credit card purchases.

Matrix *X* representing the movie preferences of six users—is displayed in Figure 1. The rows of the user–footprint matrix represent users, columns represent digital footprints, and cells record the association between users and footprints. (We often refer to digital footprints as *variables* in this article.)

User–footprint matrices can also be used for storing language data (e.g., Tweets, e-mails, or Facebook status updates). In such matrices, columns represent words or *n*-grams,[5] and cell values represent the frequency of particular words or *n*-grams per each user.

In most cases, each individual user is associated with only a small fraction of all possible footprints. Even a keen buyer, for instance, will only purchase a small fraction of the products available in an online marketplace. Thus, user–footprint matrices are usually very *sparse* or, in other words, a great majority of cells have a value of zero. As user–footprint matrices are often also extremely large, it is more practical to store them in a *sparse format* retaining only nonzero values to minimize required computer memory. Sparse matrices can be constructed using the R library *Matrix* (Bates & Maechler, 2015) or the Python library *SciPy* (Jones, Oliphant, & Peterson, 2001). The following hands-on section demonstrates how to convert the data into a sparse user–footprint matrix in R.

Hands-On: Constructing a User–Like Matrix

In this hands-on section, we proceed to constructing a user–Like matrix, which we denote as *M*. For the sake of convenience, we want to match the location of the users and Likes, in the rows and columns of Matrix *M*, with their location in *users* and *likes* objects, respectively. In other words, the user stored in the *i*th row of the object *users* is being put in the *i*th row of Matrix *M*, and the Like stored in the *j*th row of the object *likes* is being put in the *j*th column of Matrix *M*. To achieve that, we first use the function `match`[6] to match the user and Like IDs in the *ul* object with the appropriate rows in the *users* and *likes* objects:

```
ul$user_row <- match(ul$userid,users$userid)
ul$like_row <- match(ul$likeid,likes$likeid)
```

Next, we use the pointers to rows in the *users* and *likes* objects to build a user–Like matrix using the `sparseMatrix` function from the *Matrix* library:[7]

	True Romance	Pretty Woman	Aliens	Star Wars	Due Date	Hangover
Noah	.	.	1	1	.	.
Emma	.	.	1	1	.	.
Mason	1	1	.	.	.	.
Sophia	1	1	.	.	.	.
William	.	1	.	.	1	1
James	.	.	.	.	1	1
Tom	1	1	1	1	.	.

Figure 1. A hypothetical user–movie Matrix *X*. Cells are set to one if a given user has a preference for a given movie; zeros are replaced with "." for clarity.

```
require(Matrix)
M <- sparseMatrix(i=ul$user_row, j=ul$like_row,
  x=1)
```

Parameters *i* and *j* of the `sparseMatrix` function indicate the locations (rows and columns, respectively) of the nonzero cells of the matrix. Parameter *x* indicates the values of the respective cells. As discussed before, Facebook users can issue each Like only once, and thus we set all the nonzero cells to $x = 1$.

Finally, the row names of the user–Like Matrix *M* are set to contain the IDs of the respective users, and *M*'s column names are set to contain the names of the respective Likes. We also display the dimensions of Matrix *M*:

```
rownames(M) <- users$userid
colnames(M) <- likes$name
dim(M)
```

As the objects *ul* and *likes* will not be necessary at the next stages of this analysis, they can be removed to free the computer memory:

```
rm(ul, likes)
```

The *Raw Matrix M* column in Table 1 presents descriptive statistics of the resulting user–Like Matrix *M*. It contains $n_u = 110{,}728$ users and $n_L = 1{,}580{,}284$ Likes. While the maximum possible number of user–Like pairs is about 174 billion ($n_u \times n_L = 110{,}728 \times 1{,}580{,}284$), the total number of such pairs present in this data set equals $n_{u\text{-}L} = 10{,}612{,}326$, or less than 0.006% of the maximum possible value. In other words, Matrix *M* is highly *sparse*. The benefit of storing Matrix *M* in a sparse format can be illustrated by the fact that it only occupies about 270 MB of memory, whereas saving it as a regular matrix would require 1.4 TB of memory.

Trimming the User–Footprint Matrix

Descriptive statistics provided in Table 1 highlight an issue that arises frequently when working with big data sets: a large number of footprints (i.e., variables) and users only appear a few times in the data set. Such data points are of little significance for model building and subsequent inferences; therefore, they can be removed to reduce the data set size and minimize analysis time (Schein, Popescul, Ungar, & Pennock, 2002).

There is no single, or simple, correct method to select the minimum frequency below which a given user or footprint should be removed, but the following rules can be used to reach a decision. First, remove single instances of users and footprints from the data, as they are not useful for extracting patterns. Second, consider the available hardware and computation time. Retaining too many data points may exponentially increase the required time and memory. Removing too many data points, on the other hand, may significantly reduce the amount of information available for the analyses. Thus, it is advisable to first conduct

[5] *N*-gram is a contiguous sequence of *n* words. For example, a tri-gram "to be or" can be extracted from the excerpt, "to be or not to be."

[6] Command `match(a,b)` finds the locations of the elements contained in *a* within *b*. For example, `match(c("y","h"), c("h","e","y"))` returns (3, 1), as letters *y* and *h* in the first object occupy the 3rd and the 1st positions, respectively, in the second object.

[7] Before using R library for the first time, you have to install it on your computer. Use the following command to install *Matrix* library: `install.packages("Matrix")`. Note that R code is case-sensitive.

Table 1
Descriptive Statistics of the User–Like Matrix M

Descriptive Statistic	Raw Matrix *M*	Trimmed Matrix *M*
# of users	110,728	19,724
# of unique Likes	1,580,284	8,523
# of user–Like pairs	10,612,326	3,817,840
Matrix density	0.006%	2.27%
Likes per User		
Mean	22	193
Median	96	106
Minimum	1	50
Maximum	7,973	2,487
Users per Like		
Mean	7	448
Median	1	290
Minimum	1	150
Maximum	19,998	8,445

Note: See text for details.

the planned analyses on several small, randomly selected subsamples of different sizes to approximate the relationship between the data size and the time and memory required for the analysis. This would not only inform the decision of how much data to retain, but also expedite the code writing by reducing the time required to test it.

Hands-On: Trimming the User–Like Matrix

Here, we demonstrate how to remove the least frequent data points from the user–Like Matrix *M* built in the previous hands-on section. Removing rare users and Likes from a matrix is relatively straightforward: one has to discard rows and columns that have fewer nonzero entries than chosen thresholds. However, as removing users can push some Likes below the threshold (and vice versa), this process has to be iterated repeatedly until all users and Likes in the matrix are above the corresponding thresholds.

We use relatively high thresholds of a minimum of 50 Likes per user and a minimum of 150 users per Like to reduce the time required for further analyses:[8]

```
repeat {
  i <- sum(dim(M))
  M <- M[rowSums(M) >= 50, colSums(M) >= 150]
  if (sum(dim(M)) == i) break
}
```

This code employs a `repeat` loop that runs until it is interrupted with the command `break`. Inside the loop, we first set *i* to contain the sum of dimensions of *M* (i.e., the total count of its rows and columns). Next, we retain only rows and columns containing at least as many elements as the preset thresholds of 50 and 150, respectively. Finally, we check if the size of *M* has changed. If it did, the loop is interrupted; otherwise, it continues.

Next, users deleted from *M* are removed from the *users* object:

```
users <- users[match(rownames(M),
   users$userid),]
```

The *Trimmed Matrix M* column in Table 1 presents descriptive statistics of the trimmed user–Like Matrix *M*. Its size has been significantly reduced—it now contains only n_u = 19,724 users and n_L = 8,523 Likes. In the next section, we turn our attention to extracting patterns from the trimmed user–Like Matrix *M*.

Extracting Patterns from Big Data Sets

This section focuses on extracting patterns from a user–footprint matrix via two methods representative of two broad families: (a) singular value decomposition (SVD; Golub & Reinsch, 1970), representing eigendecomposition-based methods, projecting a set of data points into a set of dimensions; and (b) latent Dirichlet allocation (LDA; Blei, Ng, & Jordan, 2003), representing cluster analytical approaches. The main advantage of LDA and other cluster analytical approaches is the ease of their interpretation. However, they also tend to be computationally expensive and work with only non-negative data. (Fortunately, negative data are rare in the context of digital footprints.) The main strengths of SVD and many other eigendecomposition-based methods are their simplicity and computational speed. As a result, they are often used when developing predictive models. In contrast to LDA, eigendecomposition-based methods can also be applied to data sets including negative data points.

Reducing the dimensionality of the data (or extracting clusters) has many advantages. First, in the context of big data sets, there are often more variables than users. In such cases, reducing dimensionality is essential, as most of the statistical analyses require that there are more (and preferably many more) users than variables. Second, even when there are more users than variables, further reducing their numbers reduces the risk of overfitting and may increase the statistical power of the results. Third, reducing dimensionality removes multicollinearity and redundancy in the data by grouping the correlated variables into a single dimension or cluster. Fourth, a small set of dimensions or clusters subsuming the data is easier to interpret than hundreds or thousands of separate variables. Finally, reducing dimensionality decreases the computation time and memory required for further analyses.

Selecting the Number of Dimensions or Clusters to Extract

One of the main considerations regarding data dimensionality reduction is selecting the right number (denoted by *k*) of dimensions or clusters to extract. Unfortunately, there is no single (or simple) correct way of doing so. Moreover, the desirable value of *k* depends on the intended application. If the goal is to gain insights from the data, a small number of dimensions or clusters might be easier to interpret and visualize. On the other hand, if the aim is to build predictive models, a larger number of dimensions or clusters will retain more information from the original matrix, thus enabling more accurate predictions. Set *k* too high, however, and the benefits of dimensionality reduction discussed earlier are lost, and the prediction accuracy may decrease. The following subsections discuss SVD and LDA in more detail and introduce a few simple methods of selecting the right value of *k*.

[8] Studies based on similar data may employ lower thresholds to retain more information. Kosinski et al. (2013), for example, used thresholds of a minimum of two Likes per user and a maximum of 20 users per Like.

Singular Value Decomposition

SVD is a popular dimensionality reduction technique widely employed in various contexts, spanning computational social sciences, machine learning, signal processing, natural language processing, and computer vision (Wall, Rechtsteiner, & Rocha, 2003). Social scientists are often more acquainted with a similar approach: principal component analysis (PCA). In fact, PCA can be considered a special case of SVD (Madsen, Hansen, & Winther, 2004): SVD performed on a centered matrix produces V and Σ, which are an equivalent of eigenvectors and square roots of eigenvalues produced by PCA. Unfortunately, PCA requires multiplying a matrix by its transpose, which is computationally inefficient for large matrices.

SVD represents a given matrix (of size m rows $\times$ n columns) as a product of three matrices: a Matrix U (of size $m \times k$) containing *left singular vectors*; a non-negative square diagonal Matrix Σ (of size k) containing *singular values*; and a Matrix V (of size $n \times k$) containing *right singular vectors*, where k is the number of dimensions that the researcher chose to extract. For simplicity, we will refer to the left and right singular values as *SVD dimensions*.

If $k \geq r$, where r is the rank[9] of the matrix, the product $U\Sigma V^T$[10] reproduces the original matrix *exactly*. When k is smaller (i.e., $k < r$), the product $U\Sigma V^T$ *approximates* the matrix. In other words, selecting a smaller k reduces the dimensionality of the matrix.

Selecting the *k*. A popular approach to selecting the right number k of SVD dimensions to extract is plotting the singular values against k. The optimum k lies at the "knee" of the resulting scree plot (Zhang, Marron, Shen, & Zhu, 2007). An alternative approach suggests retaining dimensions accounting for 70% of the variance in the original data (the variance explained by a given SVD dimension is proportional to the square of the corresponding singular value; Madsen et al., 2004). As it is often computationally expensive to search for an optimum k on the complete data set, a randomly selected subset of the data might be used instead.

Centering the data. It is a common practice to center the data (i.e., decrease the entries in matrix columns by column means) before conducting SVD to improve the interpretability of the SVD dimensions. This is because the first SVD dimension extracted from noncentered data is strongly correlated with the frequencies of the objects in rows and columns.[11] As the remaining dimensions have to be orthogonal to the first one, the resulting SVD dimensions may not represent the data well.

Centering, however, is often impossible in the context of large data sets, as it does not preserve the sparsity of the matrix. Centering converts most of the zeros (which are skipped in sparse matrices) into other values, thus greatly increasing the required memory. Fortunately, SVD dimensions based on noncentered data, although more difficult to interpret, still offer effective predictive performance. Furthermore, the interpretability of the dimensions extracted from both centered and noncentered data can be improved by applying *factor rotation* techniques.

Rotation. SVD aims at maximizing the variance accounted for by the first and subsequent orthogonal dimensions. Consequently, early SVD dimensions relate highly to many users and footprints. Additionally, many users and footprints relate to many SVD dimensions, making the SVD results difficult to interpret.

Factor rotation techniques can be used to simplify SVD dimensions and increase their interpretability by mapping the original multidimensional space into a new, rotated space. Rotation approaches can be orthogonal (i.e., producing uncorrelated dimensions) or oblique (i.e., allowing for correlations between rotated dimensions).

Varimax is one of the most popular orthogonal rotations. It minimizes both the number of dimensions related to each variable and the number of variables related to each dimension, thus improving the interpretability of the data. Other commonly used rotation techniques include quartimax, equimax, direct oblimin, and promax.

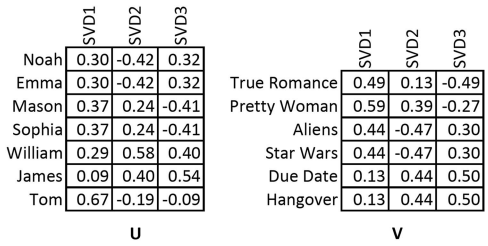
Computing SVD. Most programming languages provide off-the-shelf packages for computing SVD: the `svd` function of *Python*'s *SciPy* library (Jones et al., 2001), *eigen* library (Guennebaud & Jacob, 2010) in *C++*, *PROPACK* library (Larsen, 2005) for *Matlab*, and the *irlba* package (Baglama & Reichel, 2012) for R. For sparse or very large matrices, it is advisable to use a sparse variant of SVD available in most of the aforementioned packages. Factor rotation can be conducted using a variety of functions available in R (e.g., `varimax` or `promax`), *Matlab*, and other languages. For more details on rotation techniques, see Abdi (2003).

Example of SVD. Let us examine the results of the SVD analysis applied to user–movie Matrix X presented in Figure 1. A visual examination of Matrix X reveals several clear patterns. Looking columnwise, it can be seen that users who like *True Romance* also tend to like *Pretty Woman*. Similarly, users who like *Aliens* also like *Star Wars* and, finally, both comedies (*Due Date* and *The Hangover*) piqued the interest of the same set of users. Examining the rows of Matrix X reveals that all of the users, except Tom and William, have a preference for movies belonging to a single genre.

Figure 2 shows $k = 3$ SVD dimensions extracted from the noncentered Matrix X. The resulting Matrix U contains users' scores on the SVD dimensions, while Matrix V shows movies' scores on the SVD dimensions. As discussed before, the interpretation of the SVD results based on the noncentered data is not straightforward. Both users and movies score highly on all three dimensions. Also, the first unrotated SVD dimension (*SVD*1) relates substantially to the frequency of the users and movies: Popular movies (e.g., *Pretty Woman*) and users expressing most preferences (e.g., Tom) score highly on this dimension.

To improve SVD's interpretability, Matrix X could have been centered. Centering, however, is usually not feasible in the context of big data sets. Instead, the interpretability can be improved by *rotating* SVD dimensions. Figure 3 shows the result of varimax-rotating the SVD dimensions presented in Figure 2. The results are much clearer. The first rotated SVD dimension ($SVD_{rot}1$) could be described as a "romantic dimension." Both romantic movies (*True Romance* and *Pretty Woman*) and users liking romantic movies (Mason, Sophia, Tom, and William) score high on this dimension. Note that as William likes only one romantic movie, he scores lower on this dimension than users who like both romantic movies. The second rotated dimension ($SVD_{rot}2$) could be labeled as a "disliking sci-fi" dimension (i.e., the higher the score, the lower the preference for sci-fi movies). The third rotated SVD dimension

[9] The rank of a matrix captures the number of linearly independent rows or columns in nondegenerate matrices. See Strang (2016) for more details.
[10] V^T denotes the transposed matrix V.
[11] An object's frequency is, in this case, equivalent to the mean, because column means in a binary data set are proportional to frequencies.



	SVD1	SVD2	SVD3
Noah	0.30	-0.42	0.32
Emma	0.30	-0.42	0.32
Mason	0.37	0.24	-0.41
Sophia	0.37	0.24	-0.41
William	0.29	0.58	0.40
James	0.09	0.40	0.54
Tom	0.67	-0.19	-0.09

U

	SVD1	SVD2	SVD3
True Romance	0.49	0.13	-0.49
Pretty Woman	0.59	0.39	-0.27
Aliens	0.44	-0.47	0.30
Star Wars	0.44	-0.47	0.30
Due Date	0.13	0.44	0.50
Hangover	0.13	0.44	0.50

V

Figure 2. Users' (*U*) and movies' (*V*) scores on $k = 3$ singular value decomposition dimensions extracted from the user–movie Matrix *X*.

($SVD_{rot}3$) could be dubbed a "comedy dimension," as both comedy movies and both comedy fans (William and James) score high on it. Additionally, as one of the comedy fans (William) liked *Pretty Woman*, but not *True Romance*, the former has a low yet positive score on the comedy dimension, while the latter scores negatively on it.

Latent Dirichlet Allocation

We now turn our attention to LDA—a popular cluster analysis approach in the context of big data sets (Blei et al., 2003). While it is commonly used to study patterns in language, it can be readily applied to nontextual data as well, as long as the data is composed exclusively of positive integers (e.g., counts of words used by bloggers or counts of products purchased by consumers). LDA is one of the most readily interpretable dimensionality reduction techniques, as it produces probabilities unambiguously quantifying the associations between users, footprints, and underlying clusters.

LDA posits that each user and each footprint in the matrix (in the context of LDA, users are referred to as *documents*, and footprints as *words*) belong, with some probability, to a set of *k* clusters (referred to as *topics*). Applied to a matrix of size *m* rows × *n* columns, LDA produces matrix γ of size $m \times k$, describing the probabilities of each of the users belonging to each of the clusters; and matrix β of size $k \times n$, describing the probabilities of each of the footprints belonging to each of the clusters.

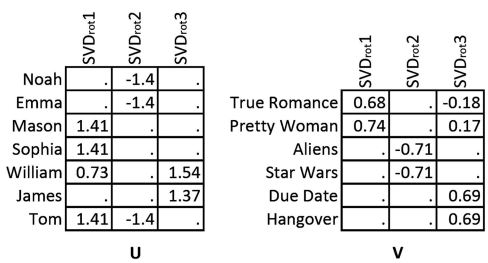
Selecting the *k*. As is the case with other methods, there is no single or simple way of selecting the correct number *k* of LDA clusters to be extracted. The commonly used approach employs the model's log-likelihood estimates, which can be plotted against the number of extracted clusters *k*. This requires producing several models for different values of *k*. Typically, the log-likelihood grows rapidly for lower ranges of *k*, flattens at higher *k* values, and may start decreasing once the number of clusters becomes very large. Selecting a *k* that marks the end of a rapid growth of log-likelihood values usually offers decent interpretability of the topics. Larger *k* values usually offer better predictive power. Alternatively, interpretability of the topics could be supported by the application of hierarchical LDA (Chang & Blei, 2010).

Dirichlet distribution parameters. Another important decision for LDA analysis is the selection of concentration parameters α and δ of the Dirichlet distribution. For symmetric Dirichlet distributions (used by most LDA implementations), high α values result in users belonging to many clusters. Using low α values results in users belonging to either one or a few clusters. Similarly, higher δ values result in clusters containing many footprints, and lower δ values produce more distinct clusters, each containing fewer footprints.

The common approach is to set $\alpha = 50/k$ and $\delta = 0.1$, or $\delta = 200/n$ (where *n* is the number of variables in *M*, and *k* the number of clusters to be extracted; Griffiths & Steyvers, 2004). Higher α and δ values are useful in the context of prediction, as they impose fewer constraints on the distribution and may enhance the amount of information retained in the LDA output. Lower values, on the other hand, produce more distinct and easily interpretable clusters, where each user and each footprint relate to very few clusters. There are several data-driven methods that can be used to fine-tune LDA parameters, but they are beyond the scope of this tutorial (see Asuncion, Welling, Smyth, & Teh, 2009 for a good introduction).

Computing LDA. Most programming languages provide off-the-shelf implementations of LDA: *topicmodels* package (Grün & Hornik, 2011) for R, *scikit-learn* package (Pedregosa et al., 2011) for *Python*, *GibbsLDA++* (Phan & Nguyen, 2007) for *C++*, and *topic modeling* toolbox (Steyvers, 2011) for *Matlab*.

Example of LDA. Let us extract $k = 3$ LDA clusters from user–movie Matrix *X* presented in Figure 1. Due to the unrealistically small size of *X*, we do not apply the recommended values of α and δ, and set them to $\alpha = 0.01$ and $\delta = 0.01$ instead. The results presented in Figure 4 are very clear and LDA clusters are easily interpretable. Associations between the movies and LDA clusters presented in matrix β indicate that *LDA*1 contains sci-fi movies, *LDA*2 contains romantic movies, and *LDA*3 contains comedies. Matrix γ is similarly clear. Note that William, who liked one of the romantic movies in addition to two comedy movies, belongs to the *romantic* (*LDA*2) and *comedy* (*LDA*3) clusters. Furthermore Tom, who liked both of the sci-fi movies and both of the romantic movies, belongs to both of the respective clusters (*LDA*1 and *LDA*2).



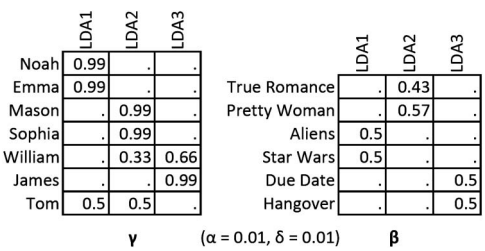
	$SVD_{rot}1$	$SVD_{rot}2$	$SVD_{rot}3$
Noah	.	-1.4	.
Emma	.	-1.4	.
Mason	1.41	.	.
Sophia	1.41	.	.
William	0.73	.	1.54
James	.	.	1.37
Tom	1.41	-1.4	.

U

	$SVD_{rot}1$	$SVD_{rot}2$	$SVD_{rot}3$
True Romance	0.68	.	-0.18
Pretty Woman	0.74	.	0.17
Aliens	.	-0.71	.
Star Wars	.	-0.71	.
Due Date	.	.	0.69
Hangover	.	.	0.69

V

Figure 3. Singular value decomposition dimensions from Figure 2 rotated using varimax rotation; zeros are replaced with "." for clarity.



	LDA1	LDA2	LDA3
Noah	0.99	.	.
Emma	0.99	.	.
Mason	.	0.99	.
Sophia	.	0.99	.
William	.	0.33	0.66
James	.	.	0.99
Tom	0.5	0.5	.

γ

(α = 0.01, δ = 0.01)

	LDA1	LDA2	LDA3
True Romance	.	0.43	.
Pretty Woman	.	0.57	.
Aliens	0.5	.	.
Star Wars	0.5	.	.
Due Date	.	.	0.5
Hangover	.	.	0.5

β

Figure 4. Latent Dirichlet allocation topics extracted from user–movie Matrix *X* using $\alpha = 0.01$ and $\delta = 0.01$; zeros are replaced with "." for clarity.

The importance of parameters α and δ for the interpretability of the LDA topics can be illustrated by rerunning the analysis using different α and δ values. Figure 5 presents the results of LDA analysis employing a higher value of α = 16.66 and a higher value of δ = 0.1. The resulting matrices γ and β are difficult to interpret. (Note, however, that the lower interpretability of these matrices does not necessarily imply that the accuracy of the predictive models based upon them would also be lower).

Hands-On: Reducing the Dimensionality of the User–Like Matrix Using SVD and LDA

In this section, we use SVD and LDA to extract patterns from the user–Like Matrix *M* constructed in the previous hands-on section. For the sake of simplicity, we select a relatively small value of $k = 5$, but encourage the readers to explore different values of *k*, as discussed in the Selecting the Number of Dimensions or Clusters to Extract section.

SVD. We use the *irlba* package (Baglama & Reichel, 2012) to compute (or rather *approximate*) SVD dimensions of the user–Like Matrix *M*. As SVD approximation is nondeterministic, we preset R's random number generator to a fixed value of *seed* = 68 to ascertain that the results presented here and computed by the readers are the same:

```
set.seed(seed = 68)
library(irlba)
Msvd <- irlba(M, nv = 5)
u <- Msvd$u
v <- Msvd$v
```

Parameter *nv* = 5 corresponds to *k*, or the number of the SVD dimensions to be extracted. Note that we do not center the data prior to conducting the SVD to maintain the sparse format of Matrix *M*. Centering could be achieved by using the following command: `M <- scale(M, scale = F)`.

The scree plot representing the singular values of the consecutive SVD dimensions (matrix Σ) can be displayed using the following command. As discussed before, this plot is helpful when selecting number *k* of dimensions to extract:

```
plot(Msvd$d)
```

Next, we use the `varimax` function to rotate the SVD dimensions. The following code produces V_{rot} and U_{rot}, the varimax-rotated equivalents of matrices *U* and *V*:

```
v_rot <- unclass(varimax(Msvd$v)$loadings)
u_rot <- as.matrix(M %*% v_rot)
```

	LDA1	LDA2	LDA3
Noah	0.34	0.34	0.32
Emma	0.34	0.34	0.32
Mason	0.34	0.34	0.32
Sophia	0.34	0.34	0.32
William	0.31	0.33	0.35
James	0.32	0.32	0.36
Tom	0.35	0.35	0.31

γ

(α = 16.66, δ = 0.1)

	LDA1	LDA2	LDA3
True Romance	0.47	0.01	0.02
Pretty Woman	0.02	0.54	0.02
Aliens	0.02	0.41	0.02
Star Wars	0.47	0.01	0.02
Due Date	0.02	0.01	0.46
Hangover	0.02	0.01	0.46

β

Figure 5. Latent Dirichlet allocation topics extracted from user–movie Matrix *X* using α = 16.66 and δ = 0.1.

LDA. We use the *topicmodels* package (Grün & Hornik, 2011) to extract LDA topics from the user–Like Matrix *M*. We set Dirichlet distribution parameters to α = 10 and δ = 0.1 (see the Dirichlet Distribution Parameters section for information on setting these parameters). As in the case of the SVD analysis, we preset R's random number generator to *seed* = 68 to ascertain that the results presented here and computed by the readers are the same:

```
library(topicmodels)
Mlda <- LDA(M, k = 5, control = list(alpha =
  10, delta = .1, seed = 68), method =
  "Gibbs")
gamma <- Mlda@gamma
beta <- exp(Mlda@beta)
```

As discussed before, the fit of the LDA model is expressed by its log-likelihood. The following code can be used to compute the log-likelihoods across different values of *k*. We use the `for` loop to cycle through increasing values of *i*, train the LDA model while setting the $k = i$, and then extract the model's log-likelihood and its degrees of freedom using the `logLik` function. The results are saved in the *lg* object. Consider widening the range of *k*s to be tested, but be aware that this code may take a very long time to run (depending on the speed of your computer):

```
lg <- list()
for (i in 2:5) {
  Mlda <- LDA(M, k = i, control =
  list(alpha = 10, delta = .1, seed = 68),
  method = "Gibbs")
  lg[[i]] <- logLik(Mlda)
}
plot(2:5, unlist(lg))
```

Interpreting Dimensions and Clusters

This section focuses on interpreting dimensions and clusters extracted from the user–footprint matrix. Such interpretations are certainly not trivial, yet may offer important insights. Several major psychological models and theories originated from analyses not unlike the one presented here. Lexical hypothesis, for instance, postulates that important interpersonal psychological differences become part of the language. This theory is a major foundation of much of the personality psychology, and was used to develop major personality frameworks, such as HEXACO or Big Five (Goldberg, 1992).[12] It is reasonable to expect that interpersonal psychological differences and other psychological phenomena might be also reflected in the patterns of digital footprints.

Several strategies can be employed to interpret the nature of patterns extracted from large data sets. First, one can explore the footprints most strongly associated with a given dimension or cluster. This approach was employed to interpret dimensions and clusters extracted from the user–movie Matrix *X* in the section Extracting Patterns From Big Data Sets. Second, one can examine the relationships between dimensions and clusters and some known properties of the users (e.g.,

[12] The six dimensions of HEXACO include Honesty-Humility (H), Emotionality (E), Extraversion (X), Agreeableness (A), Conscientiousness (C), and Openness to Experience (O).

demographic information, psychological traits, answers to individual questions on psychological questionnaires, etc.). Finally, the interpretation of the dimensions or clusters can be supported by exploring their hierarchical structure across different levels of *k*. An interesting approach suitable for SVD has been discussed by Goldberg (2006). An equivalent for LDA is provided by hierarchical LDA (Chang & Blei, 2010).

Hands-On: Interpreting Clusters and Dimensions

This hands-on section shows how to extract information helpful in interpreting clusters and dimensions. For simplicity, we use the $k = 5$ LDA clusters and $k = 5$ SVD dimensions extracted from Matrix *M* in the previous hands-on section, though we encourage the reader to experiment with different values of *k*.

LDA clusters. We start by computing the correlations between user scores on LDA clusters (stored in the R object *gamma*) and psychodemographic user traits. Note that the first column of the object *users* is excluded from the correlation because it contains the user IDs:

```
cor(gamma, users[,-1], use = "pairwise")
```

For ease of interpretation, the resulting correlation matrix is presented as a heat map in Figure 6.[13] (The companion website contains the code used to generate this heat map.) The pattern of correlations reveals that gender, age, political views, and personality trait of openness are most strongly correlated with Like clusters.

The membership in cluster *LDA*1 correlates negatively with age and positively with gender (which takes the value of 0 for males and 1 for females) and political views (where 0 denotes Democrats and 1 denotes Republicans), suggesting that this cluster is dominated by young, conservative females. Cluster *LDA*2 is similar, but does not correlate strongly with gender. Cluster *LDA*3 contains older, open-minded, and politically liberal males. *LDA*4 is weakly related to psychodemographic traits included in this sample, compared with other clusters. The last cluster, *LDA*5, is most strongly related to personality dimensions. It groups emotionally stable, agreeable, extroverted, conscientious, conservative (both in terms of personality and political views), old, and female users.

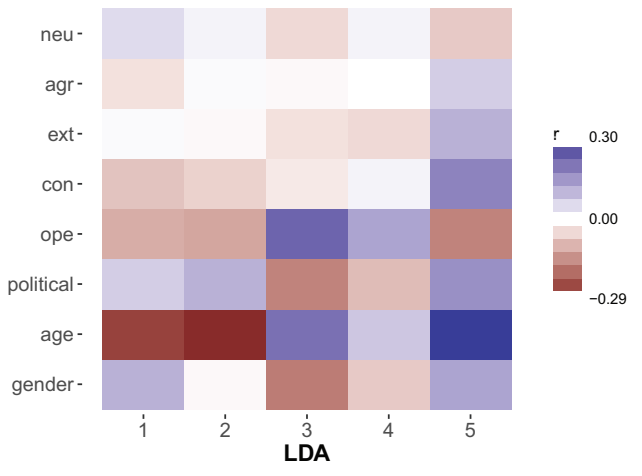


Figure 6. The heat map presenting correlations between users' membership in $k = 5$ latent Dirichlet allocation clusters and their psychodemographic traits.

Next, we investigate which Likes are most representative of these $k = 5$ LDA clusters. The strength of the relationship between Likes and clusters is stored in the R object *beta*. The following code can be used to extract the top 10 Likes most strongly associated with each of the clusters:

```
top <- list()
for (i in 1:5) {
  f <- order(beta[i,])
  temp <- tail(f, n = 10)
  top[[i]] <- colnames(M)[temp]
}
top
```

In this code, we first create an empty list *top* to store the results. Next, we start a `for` loop assigning consecutive values from 1 to 5 to *i*. Inside the loop, we use the `order` function to order the Likes in ascending order based on their scores on the *i*th LDA cluster. Next, we use the `tail` function to extract the indexes of the last 10 Likes (i.e., Likes with the highest LDA scores) and save their names as the *i*th element of list *top*.

Table 2 presents the top 10 Likes associated with each of the $k = 5$ LDA clusters extracted using the above code. An examination of these Likes provides additional insights into the character of the clusters and their members. Relatively young and female cluster *LDA*2 is best represented by the Likes containing humorous and juvenile statements, such as "I hate it when I'm taking a drink and all the ice attacks my face." *LDA*2 contains some of the most popular commercial Facebook Likes (e.g., *YouTube*, *Oreo*, *Skittles*, and *Reese's*), suggesting that this is a cluster grouping Likes advertised to Facebook users. The fact that young users are overrepresented in the *LDA*1 cluster suggests that this audience is either most inclined to interact with adverts and commercial content, most targeted by advertisements, or both. The Likes associated with cluster *LDA*3 (e.g., *Barack Obama* or *The Colbert Report*) confirm its more mature and liberal character. Clusters *LDA*4 and *LDA*5 seem to be related to music: the Former could be labeled a *rock music* cluster (e.g., *Pink Floyd* or *Metallica*), while the latter, containing more conservative and female users, seems focused on *pop* (e.g., *Taylor Swift* and *Lady Gaga*).

SVD dimensions. A similar approach can be employed in the context of SVD dimensions. The correlations between user scores on the varimax-rotated SVD dimensions (U_{rot}) and psychodemographic user traits could be obtained using the following code:

```
cor(u_rot, users[,-1], use = "pairwise")
```

The results can be found in Figure 7. As is the case with LDA clusters, gender, age, and the personality trait of openness are most strongly related to varimax-rotated SVD dimensions. Dimension $SVD_{rot}2$, for example, correlates negatively with age and positively with being female. Dimension $SVD_{rot}4$ is positively correlated with openness and age, and negatively with being Republican and being female.

[13] A heat map is a graphical representation of data in which the individual values contained in a matrix are represented using color scales.

As in the case of LDA clusters, one can also investigate the associations between Likes and varimax-rotated SVD dimensions. In the interest of space, we do not include these results here, but encourage the reader to produce them. Note that V_{rot} is an equivalent of β in LDA. Also, as SVD produces both negative and positive associations, it is convenient to separately consider footprints associated most positively and most negatively with a given SVD dimension.

Table 2
Likes Most Strongly Associated With the k = 5 Latent Dirichlet Allocation (LDA) Clusters

LDA Cluster	Top 10 Facebook Likes
LDA1	I'd rather do nothing at your house than at mine
	If you remember the L'Oreal kids FISH SHAPED SHAMPOO BOTTLE!
	Open fridge, nothing. Freezer? nothing. Might as well try the fridge again.
	Comebacks that make the whole room go OOOOOHHHHHH
	My level of maturity changes depending on who im with.
	I hate it when i'm taking a drink and all the ice attacks my face
	Telling inanimate objects to STAY when they look like they're going to fall
	I feel stupid when I say what? a thousand times because I can't hear
	i finally stop laughing . . . look back over at you and start all over again
	Ok, If we get caught here's the story . . .
LDA2	Reese's
	Disney Pixar
	Facebook
	Owl City
	Duck Tape
	Music
	YouTube
	Oreo
	Starburst
	Skittles
LDA3	Futurama
	The Onion
	Barack Obama
	The Lord of the Rings Trilogy (Official Page)
	Harry Potter
	Music
	The Daily Show
	The Colbert Report
	Queen
	The Beatles
LDA4	House
	Disturbed
	Music
	Pink Floyd
	Metallica
	The Beatles
	Red Hot Chili Peppers
	Nirvana
	Linkin Park
	Family Guy
LDA5	Usher
	Adam Sandler
	Taylor Swift
	Katy Perry
	Lady Gaga
	Victoria's Secret
	The Hangover
	Rihanna
	Eminem
	Lil Wayne

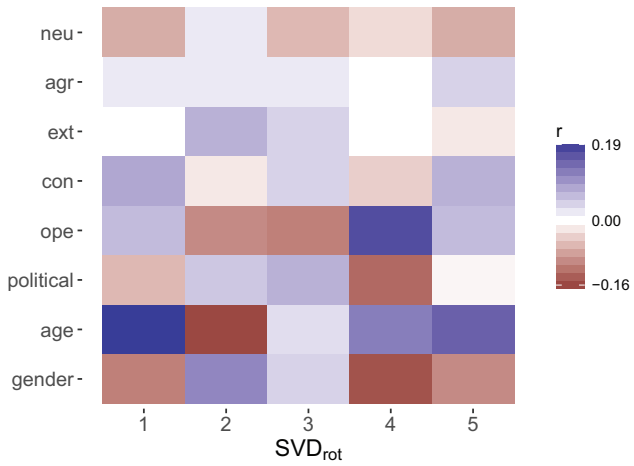


Figure 7. The heat map presenting correlations between $k = 5$ varimax-rotated singular value decomposition dimensions and users' psychodemographic traits.

The loop used previously to extract the Likes most representative of the LDA clusters could be adapted to extract the Likes with the 10 highest and 10 lowest scores on the SVD dimensions. To order the Likes according to their scores on the *i*th SVD dimension, use the following command:

```
f <- order(v_rot[,i])
```

To extract the indexes of the Likes with the extreme scores, use the `tail` and `head` functions providing the *n* last or first elements of an object, respectively:

```
colnames(M)[tail(f, n = 10)]
colnames(M)[head(f, n = 10)]
```

Predicting Real-Life Outcomes

This section focuses on building prediction models based on the dimensions and clusters extracted from the user–footprint matrix.[14] There is an abundance of methods that can be used to build prediction models based on large data sets, ranging from relatively sophisticated approaches, such as deep learning, neural networks, probabilistic graphical models, or support vector machines, to much simpler approaches, such as linear and logistic regressions. In practice, it is sensible to start with the simpler prediction methods. They are computationally faster and easier to implement, and can offer a good baseline for judging quality and for debugging more sophisticated approaches. Furthermore, in our experience, simple models, such as linear and logistic regressions, often offer accuracy similar to that of more sophisticated approaches—while being more readily interpretable. Finally, employing simpler

[14] Some prediction models, such as least absolute shrinkage and selection operator regression (Tibshirani, 2011), can be applied directly to the user–footprint matrix. However, as discussed before, reducing dimensionality of the data before building the model can benefit the quality and the accuracy of the results.

approaches reduces the risk of errors, maximizes methodological transparency, and facilitates the replicability of the results. In this work, we employ linear and logistic regressions, well known to most social scientists. We encourage the reader, however, to review more advanced methods; a good introduction can be found in James, Witten, Hastie, and Tibshirani (2013); Bishop (2006); and Bengio and Courville (2016).

Cross-Validation

Overfitting is one of the main risks associated with training predictive models. Any data set contains both signal (i.e., underlying effect) and noise (i.e., random error). Overfitting occurs when a model describes a random error instead of or beyond the underlying effect. One way of avoiding overfitting is to reduce the number of variables in the data, as discussed in the Extracting Patterns From Big Data Sets section. Another is to employ cross-validation (James et al., 2013). In the most basic cross-validation setting, the data set is split into two independent subsets: *training* and *test* subsets. The model is built on the training subset, and its accuracy is validated on the independent test subset.

To reduce the variability of the results, multiple rounds of cross-validation are usually performed, employing different partitions of the data. For example, *k*-fold cross-validation splits the data into k (usually $k = 10$) equal-sized subsets (referred to as *folds*). The model is built on a training subset composed of all-but-one $(k - 1)$ folds and validated on the excluded, testing subset. This process is repeated k times for each subset, and the accuracy is averaged across all trials.

Cross-validation can be easily conducted in R and other statistical languages using, for example, a `for` loop, but excellent designated libraries exist as well. Popular R libraries supporting cross-validation include *boot* (Canty & Ripley, 2016) and *caret* (Kuhn, 2015). For *Python* users, we recommend the cross-validation module in *scikit-learn* (Pedregosa et al., 2011).

Hands-On: Predicting Real-Life Outcomes With Facebook Likes

In this hands-on section, we demonstrate how to build a prediction model based on the SVD dimensions extracted from the user–Like Matrix *M*. We start by assigning a random number from 1 to 10 to each user in the sample as a way of splitting the users into 10 independent subsets (or *folds*) that will be used for cross-validation. The following code produces vector *folds* of a length equal to the number of users, composed of randomly selected numbers ranging from 1 to 10:

```
folds <- sample(1:10, size = nrow(users),
  replace = T)
```

Next, the users for whom the value of *folds* equals 1 are assigned to the test subset, and the remaining users are assigned to the training subset. The following code produces a logical vector[15] *test* that takes the value of *TRUE* when object *folds* equals 1 and *FALSE* otherwise:

```
test <- folds == 1
```

Logical vectors can be used to extract desired elements from R objects, such as other vectors or matrices. For example, the command `mean(users$age[test])` can be used to compute the average age of the users in the test subset. Moreover, the true/false elements of the logical vector can be easily reversed using the logical operator *not*, denoted by "!" Thus, instead of creating a separate logical vector indicating the membership in the training subset, one can simply use the reverse of *test* vector, or !*test*. In the following code, *test* and !*test* vectors are used to access test and training subsets, respectively.

In the next step, we extract $k = 50$ SVD dimensions from a training subset of Facebook Likes (i.e., *M*[!*test*,]). SVD scores of Likes are varimax-rotated and used to compute varimax-rotated user SVD scores for the entire sample. Consequently, the SVD scores for the users in the test subset are based on the analysis conducted solely on the training subset, preserving the independence of the results obtained from the training and test subsets:

```
Msvd <- irlba(M[!test,], nv = 50)
v_rot <- unclass(varimax(Msvd$v)$loadings)
u_rot <- as.data.frame(as.matrix(M %*%
  v_rot))
```

Next, we apply the function `glm` to build regression models predicting variables *ope* (openness) and *gender* from the user SVD scores in the training subset. By default, `glm` employs the linear regression model. The logistic regression model, suitable for dichotomous variables such as gender, can be obtained by specifying the parameter *family* = "*binomial*". Additionally, we indicate that the object *u_rot* is the source of the independent variables (*data* = *u_rot*), and stipulate that the model should be built on only the training subset (*subset* = !*test*). Finally, we indicate that the variable *ope* in the object *users* should be predicted using all independent variables ("." in "*users$ope* ~ ." stands for "use all independent variables"):

```
fit_o <- glm(users$ope~., data = u_rot,
  subset = !test)
fit_g <- glm(users$gender~., data = u_rot,
  subset = !test, family = "binomial")
```

Next, we estimate the predictions for the testing sample using the function `predict`. Predictions are produced using the models developed in a previous step (*fit_o* and *fit_g*) and varimax-rotated SVD scores for the users in the test subset (*u_rot*[*test*,]):

```
pred_o <- predict(fit_o, u_rot[test,])
pred_g <- predict(fit_g, u_rot[test,], type =
  "response")
```

Finally, we proceed to estimate the accuracy of the predictions for this particular cross-validation fold. The accuracy of the linear predictions (e.g., for openness) can be conveniently expressed as a Pearson product–moment correlation:

```
cor(users$ope[test], pred_o)
```

The resulting accuracy should be about $r \approx 0.43$.

In the case of dichotomous variables, such as gender, we suggest reporting prediction accuracy using the area under the receiver-operating characteristic curve coefficient (AUC), which can be computed using the ROCR library (Sing, Sander, Beerenwinkel, & Lengauer, 2005):

```
library(ROCR)
temp <- prediction(pred_g, users$gender
```

[15] Logical vector is a vector composed of *TRUE* and *FALSE* values.

```
  [test])
performance(temp,"auc")@y.values
```

In this example, accuracy should be about $AUC \approx 0.94$. AUC is an equivalent of the probability of correctly ranking two randomly selected participants, one from each class. An AUC = 0.94, for example, indicates that in 94% of randomly selected pairs containing a male and a female, the female will be ranked as more likely to be female. (Note that for dichotomous variables, the random guessing baseline corresponds to an AUC = 0.50.)

So far, we have estimated prediction performance based on all $k = 50$ SVD dimensions. Here, we will investigate the relationship between the cross-validated prediction accuracy and the number k of SVD dimensions. In the following snippet of code, we first define a set *ks* containing the values of k that we want to test. It includes all values from 2 to 10, 15, 20, 30, 40, and 50. Next, we create an empty list *rs* that is used to store the results. We then start a `for` loop that reruns the code surrounded by the curly brackets while changing the value of i to consecutive elements stored in *ks*. The code within the brackets is similar to the one used before, but only i first of the $k = 50$ SVD dimensions are used (*Msvd$v*[,1 : *i*]). The results are saved as a new element on the list *rs* (*rs*[[*as.character*(*k*)]]). Once again, we fix R's random number generator to make sure that our results match those obtained by the readers:

```
set.seed(seed = 68)
ks<-c(2:10,15,20,30,40,50)
rs <- list()
for (i in ks){
  v_rot <- unclass(varimax(Msvd$v[,
    1:i])$loadings)
    u_rot <- as.data.frame(as.matrix(M%*%
      v_rot))
    fit_o <- glm(users$ope~., data = u_rot,
      subset = !test)
    pred_o <- predict(fit_o, u_rot[test,])
    rs[[as.character(i)]] <- cor(users$ope
      [test], pred_o)
}
```

The results of this analysis are presented in Figure 8 (the code used to generate this figure can be obtained from the companion website). It is clear that the accuracy grows steeply with the number k of the SVD dimensions used in prediction, from $r = .09$ for $k = 2$ to $r = .44$ for $k = 50$. The results suggest that employing $k = 20$ SVD dimensions might be a good choice for building models predicting openness, as it offers accuracy that is close to what seems like the higher asymptote for this data. Another interesting observation relates to the sudden increases in the accuracy that can be observed when including the third and eighth SVD dimension in the model. It suggests that these dimensions contain information useful for predicting openness. A more detailed picture could be offered by exploring the coefficients of the regression model, that is, `coef(fit_o)`.

Finally, we compute the cross-validated predictions of openness for the entire sample. To this end, we build models and compute the predicted values using each of the 10 folds of data. First, we create a vector *pred_o* to store the prediction results for the entire sample. It is filled with missing values (NA) and

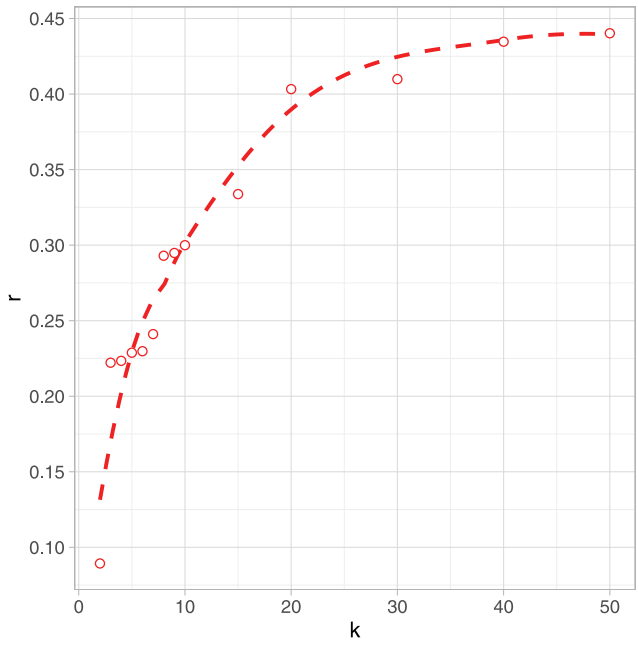


Figure 8. Relationship between the accuracy of predicting openness and the number of the varimax-rotated singular value decomposition dimensions used.

has as many elements as the number of users. Next, we use the `for` loop to rerun the code introduced before, while selecting consecutive cross-validation folds. Each time, the predicted openness values are saved as the elements of the *pred_o* vector. After the loop is complete, we compute the correlation between the actual and predicted openness scores for all users in the sample:

```
set.seed(seed = 68)
pred_o <- rep(NA, n = nrow(users))
for (i in 1:10){
  test <- folds == i
  Msvd <- irlba(M[!test,], nv = 50)
  v_rot <- unclass(varimax(Msvd$v)$loadings)
  u_rot <- as.data.frame(as.matrix(M %*%
    v_rot))
  fit_o <- glm(users$ope~., data = u_rot,
    subset = !test)
  pred_o[test] <- predict(fit_o, u_rot[test,])
  }
cor(users$ope, pred_o)
```

The prediction accuracy for openness, estimated on the entire sample, is $r = .44$.

We encourage the readers to use the examples above to develop prediction models based on the LDA cluster memberships. (Consult the companion website for the code required to conduct such analysis.) Note that users' LDA cluster scores are stored in the object *gamma* (which is an equivalent of the object *u_rot* used in the SVD analysis). The following example of code shows how to develop the LDA model on the training subset and then use it to estimate LDA cluster scores for all users using the function `posterior`. Try wrapping it within a `for` loop, similar to the one used for producing SVD-based predictions:

```
Mlda <- LDA(M[!test,], control =
  list(alpha = 1, delta = .1, seed = 68),
  k = 50, method = "Gibbs")
temp<-posterior(Mlda, M)
gamma <- as.data.frame(temp$topics)
```

Note that the LDA algorithm is significantly slower than SVD. Thus, both LDA and posterior functions may take a long time to run.

As an exercise, try to independently develop the models aimed at the remaining traits. You can compare your results with the ones presented in Table 3 and verify your code by comparing it with the one published on the companion website.

Conclusion and Discussion

This work aims at introducing the reader to crucial tools that can be used to study large data sets of digital footprints. Those inspired by this tutorial to further their knowledge in the area may be interested in the additional resources and exercises included in the companion website. We also recommend two great textbooks: Field, Miles, and Field's (2012) introduction to statistical analysis in R and James, Witten, Hastie, and Tibshirani's (2013) introduction to statistical learning in R.

Previous sections list multiple advantages of large digital footprint samples, such as their longitudinal character and the inclusion of individuals often underrepresented in traditional studies. Importantly, however, studying large samples of digital footprints may also present large risks related to privacy and scientific misconduct (Kosinski et al., 2015; Hall & Flynn, 2001; Barchard & Williams, 2008). This issue is, unfortunately, exacerbated by the lack of clear guidelines; the protocols related to designing large-scale online studies, storing data, and analyzing results are scarce and often contradictory (Solberg, 2010; Wilson, Gosling, & Graham, 2012). This situation is further complicated by the speed of technological progress and constant changes in digital platforms and environments. As a result, both researchers and ethical boards are prone to either over or underestimating the threats.

One major challenge relates to the vague boundary between public and private information. Massive amounts of data are publicly available and can be freely scraped from online platforms and environments. Some researchers argue that mining such public data is equivalent to conducting archival research, a method frequently employed in disciplines like history, art criticism, and literature, which rarely involve rules for the protection of human subjects (Bruckman, 2002; Herring, 1996). Others, however, point out that the border between public and private is not determined by accessibility, but by social norms and practices. Take, for instance, a small village, where people know most of the intimate details about each other. Despite the public knowledge of such details, people implicitly assume that certain intimate facts are personal and should not be discussed or, even less so, studied (Schultze & Mason, 2012).

Another challenge relates to participants' consent. Data donated by the participants often contain information related to or contributed by other people (e.g., comments on the participant's profile). Deciding if and to what extent such data could be used in research is not straightforward (Kosinski et al., 2015).

Finally, researchers should be extremely careful when working with data obtained from third parties. At a minimum, one should review the conditions of the consent obtained by a third party and consider the implications of the research for the users. A good illustration of risks related to such research is provided by an overwhelmingly negative public reception of the emotional contagion study conducted on Facebook (Kramer et al., 2014). The sheer availability of the data does not mean that its study is ethical or appropriate.

Table 3
Approximate Prediction Accuracy Based on k = 50 Singular Value Decomposition (SVD) Dimensions and k = 50 Latent Dirichlet Allocation (LDA) Clusters

Variable	SVD	LDA
Gender (AUC)	.94	.88
Political views (AUC)	.88	.84
Age	.61	.68
Openness	.44	.42
Conscientiousness	.26	.22
Extroversion	.30	.25
Agreeableness	.24	.18
Neuroticism	.29	.24

Note. AUC = area under the receiver-operating characteristic curve coefficient.

References

Abdi, H. (2003). Factor rotations in factor analyses. In M. Lewis-Beck, A. E. Bryman, & T. F. Liao (Eds.), *The SAGE encyclopedia of social science research methods* (pp. 792–795). Thousand Oaks, CA: SAGE.

Asuncion, A., Welling, M., Smyth, P., & Teh, Y. W. (2009). On smoothing and inference for topic models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 27–34). Arlington, VA: AUAI Press.

Baglama, J., & Reichel, L. (2012). irlba: Fast partial SVD by implicitly-restarted Lanczos bidiagonalization (R package Version 2.0.0) [Computer Software]. Retrieved from https://cran.r-project.org/package=irlba

Barchard, K. A., & Williams, J. (2008). Practical advice for conducting ethical online experiments and questionnaires for United States psychologists. *Behavior Research Methods, 40,* 1111–1128. http://dx.doi.org/10.3758/BRM.40.4.1111

Bates, D., & Maechler, M. (2015). Matrix: Sparse and dense matrix classes and methods (R package Version 1.2–6) [Computer Software]. Retrieved from https://cran.r-project.org/package=Matrix

Bengio, I. G. Y., & Courville, A. (2016). *Deep learning*. Manuscript in preparation. Retrieved from http://www.deeplearningbook.org/

Bishop, C. M. (2006). *Pattern recognition and machine learning*. New York, NY: Springer.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research, 3,* 993–1022. http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993

Bruckman, A. (2002). Studying the amateur artist: A perspective on disguising data collected in human subjects research on the Internet. *Ethics and Information Technology, 4,* 217–231. http://dx.doi.org/10.1023/A:1021316409277

Buchanan, E., Aycock, J., Dexter, S., Dittrich, D., & Hvizdak, E. (2011). Computer science security research and human subjects: Emerging considerations for research ethics boards. *Journal of Empirical Research on Human Research Ethics, 6,* 71–83. http://dx.doi.org/10.1525/jer.2011.6.2.71

Canty, A., & Ripley, B. D. (2016). boot: Bootstrap R (S-Plus) functions (R package Version 1.3–18) [Computer Software]. Retrieved from https://cran.r-project.org/package=boot

Chang, J., & Blei, D. M. (2010). Hierarchical relational models for document networks. *The Annals of Applied Statistics, 4,* 124–150. http://dx.doi.org/10.1214/09-AOAS309

Danescu-Niculescu-Mizil, C., West, R., Jurafsky, D., Leskovec, J., & Potts, C. (2013). No country for old members: User lifecycle and linguistic change in online communities. *Proceedings of the 22nd International Conference on World Wide Web* (pp. 307–318). New York, NY: ACM. http://dx.doi.org/10.1145/2488388.2488416

Eichstaedt, J. C., Schwartz, H. A., Kern, M. L., Park, G., Labarthe, D. R., Merchant, R. M., . . . Seligman, M. E. P. (2015). Psychological language on Twitter predicts county-level heart disease mortality. *Psychological Science, 26,* 159–169. http://dx.doi.org/10.1177/0956797614557867

Field, A., Miles, J., & Field, Z. (2012). *Discovering Statistics Using R*. Los Angeles, CA: Sage. http://dx.doi.org/10.1111/insr.12011_21

Goldberg, L. R. (1992). The development of markers for the Big-five factor structure. *Psychological Assessment, 4,* 26–42. http://dx.doi.org/10.1037/1040-3590.4.1.26

Goldberg, L. R. (2006). Doing it all bass-ackwards: The development of hierarchical factor structures from the top down. *Journal of Research in Personality, 40,* 347–358. http://dx.doi.org/10.1016/j.jrp.2006.01.001

Goldberg, L. R., Johnson, J. A., Eber, H. W., Hogan, R., Ashton, M. C., Cloninger, C. R., & Gough, H. G. (2006). The International Personality Item Pool and the future of public-domain personality measures. *Journal of Research in Personality, 40,* 84–96. http://dx.doi.org/10.1016/j.jrp.2005.08.007

Golub, G. H., & Reinsch, C. (1970). Singular value decomposition and least squares solutions. *Numerische Mathematik, 14,* 403–420. http://dx.doi.org/10.1007/BF02163027

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America, 101,* 5228–5235. http://dx.doi.org/10.1073/pnas.0307752101

Grün, B., & Hornik, K. (2011). topicmodels: An R package for fitting topic models. *Journal of Statistical Software, 40,* 1–30. http://dx.doi.org/10.18637/jss.v040.i13

Guennebaud, G., & Jacob, B. (2010). Eigen v3 [Computer Software]. Retrieved from http://eigen.tuxfamily.org

Hall, T., & Flynn, V. (2001). Ethical issues in software engineering research: A survey of current practice. *Empirical Software Engineering, 6,* 305–317. http://dx.doi.org/10.1023/A:1011922615502

Henrich, J., Heine, S. J., & Norenzayan, A. (2010). The weirdest people in the world? *Behavioral and Brain Sciences, 33,* 61–135. http://dx.doi.org/10.1017/S0140525X0999152X

Herring, S. (1996). Linguistic and critical analysis of computer-mediated communication: Some ethical and scholarly considerations. *The Information Society, 12,* 153–168. http://dx.doi.org/10.1080/911232343

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: with applications in R*. New York, NY: Springer.

Jones, E., Oliphant, T., & Peterson, P. (2001). SciPy: Open source scientific tools for Python [Computer software]. Retrieved from http://www.scipy.org/

Kosinski, M., Matz, S. C., Gosling, S. D., Popov, V., & Stillwell, D. (2015). Facebook as a research tool for the social sciences: Opportunities, challenges, ethical considerations, and practical guidelines. *American Psychologist, 70,* 543–556. http://dx.doi.org/10.1037/a0039210

Kosinski, M., Stillwell, D., & Graepel, T. (2013). Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences of the United States of America, 110,* 5802–5805. http://dx.doi.org/10.1073/pnas.1218772110

Kramer, A. D. I., Guillory, J. E., & Hancock, J. T. (2014). Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences of the United States of America, 111,* 8788–8790. http://dx.doi.org/10.1073/pnas.1320040111

Kuhn, M. (2015). caret: Classification and Regression Training (R package Version 6.0–68) [Computer Software]. Retrieved from https://cran.r-project.org/package=caret

Lambiotte, R., & Kosinski, M. (2014). Tracking the digital footprints of personality. *Proceedings of the Institute of Electrical and Electronics Engineers, 102,* 1934–1939. http://dx.doi.org/10.1109/JPROC.2014.2359054

Larsen, R. M. (2005). *PROPACK 2.1*. Retrieved from http://sun.stanford.edu/rmunk/PROPACK/

Lazer, D., Pentland, A., Adamic, L., Aral, S., Barabasi, A. L., Brewer, D., . . . Alstyne, M. V. (2009). Life in the network: The coming age of computational social science. *Science, 323,* 721–723. http://dx.doi.org/10.1126/science.1167742

Leskovec, J., & Krevl, A. (2014). *SNAP Datasets: Stanford Large Network Dataset Collection*. Retrieved from http://snap.stanford.edu/data

Madsen, R. E., Hansen, L. K., & Winther, O. (2004). *Singular value decomposition and principal component analysis*. Retrieved from http://www2.imm.dtu.dk/pubdb/p.php?4000

Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. *Science, 349,* aac4716. http://dx.doi.org/10.1126/science.aac4716

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12,* 2825–2830.

Phan, X.-H., & Nguyen, C.-T. (2007). *A C/C++ implementation of latent Dirichlet allocation*. Retrieved from http://gibbslda.sourceforge.net/

R Core Team. (2015). R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from http://www.R-project.org/

Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International Conference on Research and Development in Information Retrieval* (pp. 253–260). New York, NY: ACM. http://dx.doi.org/10.1145/564418.564421

Schultze, U., & Mason, R. O. (2012). Studying cyborgs: Re-examining Internet studies as human subjects research. *Journal of Information Technology, 27,* 301–312. http://dx.doi.org/10.1057/jit.2012.30

Schwartz, H. A., Eichstaedt, J. C., Kern, M. L., Dziurzynski, L., Ramones, S. M., Agrawal, M., . . . Ungar, L. H. (2013). Personality, gender, and age in the language of social media: The open-vocabulary approach. *PLOS ONE, 8,* e73791. http://dx.doi.org/10.1371/journal.pone.0073791

Sing, T., Sander, O., Beerenwinkel, N., & Lengauer, T. (2005). ROCR: Visualizing classifier performance in R. *Bioinformatics, 21,* 3940–3941. http://dx.doi.org/10.1093/bioinformatics/bti623

Solberg, L. (2010). Data mining on Facebook: A free space for researchers or an IRB nightmare? *Journal of Law, Technology & Policy, 2,* 311–433.

Steyvers, M. (2011). *Matlab topic modeling toolbox 1.4*. Retrieved from http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm

Strang, G. (2016). *Introduction to linear algebra, fifth edition*. Wellesley, MA: Wellesley-Cambridge Press.

Tibshirani, R. (2011). Regression shrinkage and selection via the lasso: A retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 73,* 273–282. http://dx.doi.org/10.1111/j.1467-9868.2011.00771.x

Ugander, J., Karrer, B., Backstrom, L., & Marlow, C. (2011). The anatomy of the Facebook social graph. *Computing Research Repository,* 1–17. Retrieved from http://arxiv.org/abs/1111.4503

Wall, M. E., Rechtsteiner, A., & Rocha, L. M. (2003). Singular value decomposition and principal component analysis. In D. Berrar, W. Dubitzky, & M. Granzow (Eds.), *A practical approach to microarray data analysis* (pp. 91–109). Norwell, MA: Kluwer.

Wilson, R. E., Gosling, S. D., & Graham, L. T. (2012). A review of Facebook research in the social sciences. *Perspectives on Psychological Science, 7,* 203–220.

Youyou, W., Kosinski, M., & Stillwell, D. J. (2015). Computer-based personality judgements are more accurate than those made by humans. *Proceedings of the National Academy of Sciences of the United States of America, 112,* 1036–1040.

Zhang, L., Marron, J., Shen, H., & Zhu, Z. (2007). Singular value decomposition and its visualization. *Journal of Computational and Graphical Statistics, 16,* 833–854.

Received June 9, 2015
Revision received July 10, 2016
Accepted July 12, 2016 ■