10-2021

Links do matter: Understanding the drivers of developer interactions in software ecosystems

Subhajit DATTA
Singapore Management University, subhajitd@smu.edu.sg

Amrita BHATTACHARJEE

Subhashis MAJUMDER

Citation

DATTA, Subhajit; BHATTACHARJEE, Amrita; and MAJUMDER, Subhashis. Links do matter: Understanding the drivers of developer interactions in software ecosystems. (2021). *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME): Luxembourg, September 27 - October 1: Proceedings*. 1-5.
Available at: https://ink.library.smu.edu.sg/sis_research/6579

Links *do* Matter: Understanding the Drivers of Developer Interactions in Software Ecosystems

1st Subhajit Datta
Singapore Management University
Singapore
subhajit.datta@acm.org

2nd Amrita Bhattacharjee
Arizona State University
Tempe, USA
abhatt43@asu.edu

3rd Subhashis Majumder
Heritage Institute of Technology
Kolkata, India
subhashis.majumder@heritageit.edu

***Abstract*—Studies of collaborating individuals engaged in collective enterprises usually focus on the individuals, rather than the links supporting their interaction. Accordingly, large scale software development ecosystems have also been examined primarily in terms of developer engagement. We posit that communication links between developers play a central role in the sustenance and effectiveness of such ecosystems. In this paper, we investigate whether and how developer attributes relate to the importance of the communication channels between them. We present a technique using 2nd order Markov models to extract features of interest of the links and apply the technique on data from a real-world project. Our statistical models – developed on records involving 900+ software developers, exchanging 20,000+ comments, across 500 units of work – offer surprising insights on factors associated with link importance, even after controlling for known effects. These results inform a deeper appreciation of the importance of links in large scale software development along with a number of practical implications.**

***Index Terms*—Software development, importance of links, developer influence, Markov models**

I. Introduction and Study Setting

In the two decades since the primacy of "individuals and interactions over processes and tools" was proclaimed in the *Manifesto for Agile Software Development* (https://agilemanifesto.org/), large scale software development has been recognized to be increasingly more *interactional* than *instructional*. In development ecosystems that have formally embraced the Agile way, as well as those that have not, interactions between developers in large and distributed teams play a critical role in the design, development, and delivery of complex software systems. When these ecosystems are studied to understand the key drivers towards quality and efficiency in the development processes, the focus is primarily on identifying the individuals who are important [1]. We posit that examining the *importance of interactions* between individuals can present a complementary view towards a deeper understanding of development ecosystems. The criticality of interactions in software development has long been recognized; the prognosis for delayed projects that *Brooks' Law* predicts, comes largely out of the interaction costs between developers [2].

Links or communication channels between developers serve as essential conduits for their interaction. There is enough evidence in the state of art and practice of large scale software development to suggest that some interactions are proverbially "more equal than others" [3]. However, factors that relate to the importance of communication channels supporting these interactions have not been examined to a notable extent in existing studies. With this perspective, we examine the following **research question** in this paper: *In a software development ecosystem, is the importance of a communication channel related to the levels of influence of the developers interacting over that channel?*

Addressing this research question offers **benefits for multiple stakeholders**. At the individual level, it can inform developers whether and how to nurture particular working relationships with their peers within and across teams. For project managers, communication channels are vital pathways of information dissemination and reception. A mechanism for understanding the factors that drive the importance of such channels can aid team governance. At the organizational level, communication channels entail set-up and maintenance costs involving physical infrastructure, as well as those arising from locational and temporal asynchrony between team members [4]. These costs are significant for large and distributed teams, with the number of possible links between developers growing quadratically, even as the team grows linearly in size. Thus, being able to decide whether a particular channel is important, based on the characteristics of the developers at its two ends, can be useful mechanism for optimal resource allocation and cost saving. In this paper, we report preliminary results from examining our research question using a technique involving 2nd order Markov networks. To the best of our knowledge, this is one of the first uses of such a technique in the study of software development ecosystems. Our research question is distilled into the following hypotheses, which we will empirically examine:

H_1: *Higher influence of the sender sending information over a communication link is related to higher link importance.* The corresponding null hypothesis is that there is no relationship between the level of influence of the sender and the importance of the link.

H_2: *Higher influence of the receiver receiving information over a communication link is related to higher link importance.* The corresponding null hypothesis is that there is no relationship between the level of influence of the receiver and the importance of the link.

In a typically large and distributed software development project, developers are embedded in a *network of interactions* with their peers. Networks can serve as useful abstractions for examining the characteristics of collaborative enterprises [5]. Accordingly, the first step in our approach is to define the *Review Comment Network* (RCN), with developers as vertices and the links representing some tangible aspect of their interaction. Using established metrics pertaining to the vertices of this network, we are able to quantify relevant developer attributes. Using a 2nd order Markov model, we next transform this "original network" to another network (the "M2 network" or M2(RCN)) whereby links in the original network become vertices in the M2 network. Due to the nature of this transformation, importance of RCN's links will be reflected in specific network metric(s) pertaining to the vertices of M2(RCN). Thus, examining properties of the original network and the corresponding M2 network allows us to extract characteristics of developers and the links between them.

We test our hypotheses on the Chromium dataset, collected and curated by Hamasaki et al. [6] for use by the research community. This dataset includes records of developers discussing code review activities in Chromium projects (http://www.chromium.org/), which are the open-source projects underpinning the Google Chrome browser and the Chrome operating system. Peer review of code is a critical activity in all large scale software systems. Developers actively interact during the review process, commenting on *review items* (hereinafter referred to as "reviews"). Reviews are code units relating to bug fixes or change requests; each review is owned by a developer. Developer interaction during the review process involves commenting on reviews owned by other developers, and having one's own reviews commented upon by others. Thus every comment has uniquely identifiable *sender(s)* and *receiver(s)* associated with it. The outcome of this developer interaction around a review is a decision on whether the related code units will be merged with main code base – or abandoned – leading to the closure of the review. Evidently, the importance of developer interaction around reviews makes this development ecosystem specially suited for examining our hypotheses. To gauge each developer's level of engagement in the review process, we queried the dataset to extract a set of developer attributes of interest to us. We also calculated a set of metrics from the RCN to reflect on each developer's position in the network vis-a-vis other developers. The importance of each link in RCN was measured by an appropriate network metric (as explained in the next section) of the vertex in M2(RCN) corresponding to that link. Using these parameters, regression models were developed to validate the hypotheses.

II. Methodology

Data pre-processing: As mentioned in the previous section, we use data from the Chromium projects as reported and made available by Hamasaki et al. [6]. We accessed 826398 comments spanning over 193135 review items from the dataset. The review items were ranked on the basis of how many comments they had attracted from the developers. We observed that the distribution of comments per review is highly skewed, with few reviews having many comments, and many reviews having few. Given the context of our study, we select the top 500 reviews with the highest number of comments which collectively includes 22424 comments.

Construction of networks: In the datasets. email ids are used to uniquely identify the developers. In some records, the email ids have special characters, or wrong format that might have been introduced due to parsing errors while populating the database shared by Hamasaki et al. [6]. We delete such records, along with the ones where one or more attributes have missing or "NA" values. As mentioned earlier, on the lines of our previous work, we generated a Review Comment Network (RCN) using the filtered dataset, whose vertices are developers, and two developers (d_i and d_j) are connected by a link directed from d_i to d_j if there is at least one instance of d_i sending a comment to d_j on a review [7]. If a comment is not a reply to another specific comment, we consider the link to be between the commenter and review owner. Our constructed RCN has vertices corresponding to 916 uniquely identifiable developers and 5552 links between these developers. We then proceed to create M2(RCN), the network based on 2nd order Markov model from the RCN, using the following protocol: For each link in the input network, we add a vertex to the M2 network, and for every directed 2-hop in the input network, we add a link (a directed edge) between the corresponding two vertices in the M2 network. This transformation results in a larger network which encodes the concept of a *2nd order* structure. We note that this construction results in the links of RCN being mapped to the vertices of M2(RCN). Thus, M2(RCN) has 5552 vertices (equal to the number of links in RCN).

Use of 2nd order Markov model: Simple network representations such as the RCN defined earlier implicitly assume the 1st order Markov property, according to which, the transition probability $P(i_t \rightarrow i_{t+1})$ of a random walker at the state denoted by vertex i_t to the state denoted by vertex i_{t+1} is given by,

$$P(i_t \rightarrow i_{t+1}) = \frac{W(i_t \rightarrow i_{t+1})}{\sum_j W(i_t \rightarrow j)}$$

where j denotes the neighbours of i_t and $W(i_t \rightarrow j)$ represents the weight of the edge from i_t to j. If all the edge weights are 1, this simply reduces to,

$$P(i_t \rightarrow i_{t+1}) = \frac{1}{OutDegree(i_t)}$$

This implies that the next state of the random walker depends only on the current state and is independent of the previous states, thus implying the *memoryless* nature of the 1st order Markov property. This is often too strict a constraint and such representations are not expressive enough to model complex real-world interactions such as the ones we are analysing, where developers discuss review items to reach a decision on their outcomes. Some real-world networks that have been shown to have higher-order dependencies are constructed from Web click-stream, shipping traffic, passenger traffic along flight pathways, and journal citations [8], [9]. Most of these networks have some notion of sequential flow in them and given the similar nature of information flow in RCN, we leverage the 2nd order Markov property in the construction of M2(RCN).

Model development: To validate our hypotheses we need to identify metrics that capture the level of a link's importance, and how influential the sender and receiver of information across that link are, in the context of RCN. Accordingly, we define our **dependent variable** as *LinkImportance*, calculated as the *pagerank* of the corresponding vertex in M2(RCN). Pagerank is a commonly used measure of vertex importance in a network [5]. As the links of RCN are transformed into the vertices of M2(RCN), the pagerank of M2(RCN)'s vertices indicate the importance of corresponding links in RCN. We recognize that there are other methods of quantifying strength of links in a network [10], [11]. However, as discussed, the use of the network based on 2nd order Markov model offers a more effective way of capturing link importance based on the *flow of information between developers*, as appropriate in our study setting. To measure how influential the individuals who send and receive information across a particular link of RCN are, we compute their *eigenvector centralities*. Eigenvector centrality is widely used to capture how influential a vertex is [5]. So, *SenderInfluence* and *ReceiverInfluence* serve as our **independent variables**. To isolate the relations between the independent variables and the dependent variable, we need to account for some of the peripheral factors that may affect the level of importance of communication channels in our network. These factors are represented as the **control variables** in our models. We consider the following categories of control variables as their relevance has been identified in existing studies [12]:

- *Developer activities in the development ecosystem:* We consider the number of reviews owned by a developer sending and receiving information over a link as *SenderWorkload* and *ReceiverWorkload* respectively; these serve as proxies for the amounts of work being handled by each developer. To have a sense of how quickly developers are completing the activities they are in charge of, we consider the *SenderTaskCompletionTime*, and *ReceiverTaskCompletionTime*. For a particular link, *SenderTaskCompletionTime* is calculated as the median of the elapsed times between the opening and closure of all reviews owned by the developer who is the source vertex for the link. For a particular link, *ReceiverTaskCompletionTime* is calculated as the median of the elapsed times between the opening and closure of all reviews owned by the developer who is the destination vertex for the link. Developer experience accrues out of the time spent working in a project. In our study setting, comments by developers on reviews serve as pointers to their involvement in the project. Thus the span of time between the first and last review comments by a developer can be a reliable proxy for the developer's experience in the project; the *SenderExperience* and *ReceiverExperience* variables are calculated accordingly.
- *Developer position in the interaction network:* The number of peers a particular developer is connected to, is given by the degree of the corresponding vertex in RCN, as reflected in the variables named *SenderConnections* and *ReceiverConnections*. Similarly, the variables *SenderBetweenness* and *ReceiverBetweenness* are given by the betweenness centralities of the corresponding vertices in RCN. The clustering coefficient of a vertex reflects the extent to which that vertex is clustered with other vertices in the network; the model variables *SenderClustering* and *ReceiverClustering* are given by the clustering coefficients of the corresponding vertices in RCN. The pageranks of the vertices representing the sender and the receiver at the ends of the links in RCN are used as the variables *SenderImportance* and *ReceiverImportance* respectively.

The formulas for calculating network metrics such as degree, betweenness, eigenvector centralities, pagerank, and clustering coefficients have been derived and explained by Newman [5]. As the dependent variable is *continuous* in nature, multiple linear regression is chosen as a modelling paradigm, rather than Poisson or negative binomial regressions which are considered if the dependent variable is a *count* of some parameter of interest [13]. Multiple linear regression rests on the assumptions of linearity, normality, and homoscedasticity of the residuals, as well as the absence of multicollinearity between the independent variables. To validate these assumptions, residual properties were examined using histograms, Q-Q plots and scatter plots of the standardized residuals, and the variance inflation factors were calculated to check for multicollinearity. To develop the most parsimonious models, we removed the following variables as they were found to be strongly correlated with other model variables, and their presence would have lead to issues related to multicollinearity: *SenderConnections*, *ReceiverConnections*, *SenderBetweenness*, *ReceiverBetweenness*, *SenderImportance*, and *ReceiverImportance*. The descriptive statistics of the variables finally included in the models is given in Table I. As the dependent variable has a strong positive skew, its fourth root is taken for inclusion in the model.

Table II shows results from the multiple linear regression models. The model with the dependent variable and the control variables is described on the left (Model I), and the model on the right shows the enhanced model that additionally includes the independent variables (Model II). As mentioned in the table caption, superscripts of the coefficients indicate the ranges of their respective p values; the p values are calculated using the *t-statistic* – the ratio of each coefficient to its standard error – and the *Student's t-distribution*. The lower portion of the table describes the overall model in terms of: N – the number of data points in the model; R^2 – the coefficient of determination, expressing the ratio of the regression sum of squares to the total sum of squares and it indicates the goodness-of-fit of the model; *df* – degrees of freedom; F –

TABLE I: Descriptive statistics of the model variables

Variable	Mean	Std Dev	Median
LinkImportance	7.221×10^{-5}	4.176×10^{-5}	6.48×10^{-5}
SenderClustering	0.466	0.218	0.43
ReceiverClustering	0.467	0.234	0.426
SenderWorkload	579.221	666.93	339
ReceiverWorkload	548.202	566.76	360
SndrTaskComplnTm	140.069	197.302	15.5
RcvrTaskComplnTm	168.457	207.796	62
SenderExperience	911.76	471.088	936
ReceiverExperience	970.092	443.151	1007
SenderInfluence	0.096	0.097	0.064
ReceiverInfluence	0.138	0.119	0.1

TABLE II: Regression models to understand the influences on *LinkImportance*. Note: Significance levels "****", "***", "**", "*", "-", denote corresponding p-value ≤ 0.001, ≤ 0.01, ≤ 0.05, ≤ 0.1, and ≥ 0.1 respectively.

	Model I	**Model II**
Intercept	0.091 ****	0.075 ****
	(8.229×10^{-4})	(7.556×10^{-4})
SenderWorkload	-3.46×10^{-6} ****	-2.927×10^{-6} ****
	(2.328×10^{-7})	(1.856×10^{-7})
ReceiverWorkload	-3.77×10^{-7} -	1.755×10^{-7} -
	(2.749×10^{-7})	(2.191×10^{-7})
SndrTaskComplnTm	2.845×10^{-6} ****	-1.829×10^{-6} ****
	(6.761×10^{-7})	(5.449×10^{-7})
RcvrTaskComplnTm	4.476×10^{-7} -	1.053×10^{-6} **
	(6.38×10^{-7})	(5.131×10^{-7})
SenderExperience	4.613×10^{-6} ****	3.495×10^{-6} ****
	(3.844×10^{-7})	(3.066×10^{-7})
ReceiverExperience	4.446×10^{-7} -	-2.457×10^{-7} -
	(4.043×10^{-7})	(3.237×10^{-7})
SenderClustering	-0.009 ****	0.013 ****
	(7.469×10^{-4})	(7.101×10^{-4})
ReceiverClustering	0.002 ***	0.003 ****
	(6.887×10^{-4})	(6.457×10^{-4})
SenderInfluence		0.074 ****
		(0.001)
ReceiverInfluence		0.004 ****
		(0.001)
Model parameters		
N	5552	5552
R^2	0.118	0.442
df	5543	5541
F	92.562	438.076
Sig level	****	****

Fisher F-statistic, the ratio of the variance in the data explained by the linear model divided by the variance unexplained by the model. The overall model's p value, calculated using the F-statistic and the F-distribution, indicates whether the overall model is statistically significant. If the $p \leq$ *level of significance* for a coefficient or the overall model, we conclude that the corresponding result is statistically significant, on the basis of null hypothesis significance testing.

III. Results and Discussion

We observe from Table II that both models (I and II) are statistically significant overall (p-value ≤ 0.001 in each case). Model I has a goodness-of-fit around 12% ($R^2 = 0.118$), which grows to around 44% for Model II ($R^2 = 0.442$). This implies that addition of the independent variables, over and above the control variables, leads to an increase in the goodness-of-fit by a factor of more than 2.5. The values of the F-statistic corresponding to Models I and II also indicate that the latter model with the independent variables is able to explain the variability in the data to a much larger extent.

Nature of our results: We observe from Model II in Table II that both *SenderInfluence* and *ReceiverInfluence* have statistically significant relations with *LinkImportance*. Higher levels of both of these variables relate to higher *LinkImportance*. Thus for both the hypotheses H_1 and H_2 introduced in Section I, we can reject the null hypothesis, in favour of the corresponding alternate hypothesis. Conventional wisdom leads us to expect the most important communication channels to be the ones bridging widest power gradients, with most influential developers sharing information with the least influential ones. *However, we see a counter-intuitive trend in our results: higher importance of links is related to higher levels of influence of both sender and receiver of information across such links. These relationships hold after controlling for the effects of workloads, task completion times, experiences, and clustering levels of senders and receivers.* Barring receiver workload and receiver experience, all other relations are statistically significant.

Implications: As we recall, RCN is constructed from instances of developers exchanging comments over code review items. As evident from some of the related work cited in the preceding sections, code review is a peer-driven process where each developer shares concerns and suggestions for improvement. Hierarchical seniority – and the prestige that comes with it – is not expected to influence the progression and outcome of the review process. We see a reflection of this dynamic in our results; higher importance of communication channels are seen to be related to higher influence of senders *as well as* receivers. This is indicative of interaction between *peers* who are communicating on a common platform, bereft of positional asymmetries.

Developer profiles: In Model II of Table II, we see evidence that senders with lower workload, quicker task completion, higher experience, and higher levels of clustering are associated with higher link importance (all effects being statistically significant). Such a profile typifies developers who are able to focus more intently on peer review activities, coupled with an elevated level of familiarity with the development ecosystem. Thus they are better placed to offer useful information and advice to their peers. On the other hand, higher levels of link importance are seen to be related to receivers who have higher workload, slower task completion, lower experience, and higher clustering (only the effects around workload and experience lack statistical significance). Developers with these characteristics are best positioned to utilize the shared wisdom of the peer review process.

Utility of our findings: Our results can inform team assembly and governance in large scale software development ecosystems. As communication channels between influential developers are seen to be of enhanced importance, managers can facilitate communication between such developers by colocating them and/or making it easy for them to exchange their views in other ways. Additionally, insights from our study can facilitate the design of features in collaborative development environments which enable smooth delivery of information to those developers who need it most. At organizational levels, the ability to identify important links on the basis of characteristics of individuals at the two ends of the links, can lead to effective processes towards employee retention and reward.

Threats to validity: We present results from an observational study rather than a controlled experiment; thus correlations in our statistical model do not necessarily imply causation. However, in any study-setting involving real-world software development ecosystems, controlled experiments are often infeasible to design and execute. Thus results from our correlational study can offer useful insights on the dynamics of developer interaction. **Construct validity** is concerned with ensuring that measurement errors in the variables of interest do not challenge the conclusions from a study. The model variables have been calculated from the available data or extracted from RCN or M2(RCN) using established network metrics. Thus, we do not see notable threats to construct validity from this aspect. But including other control variables such as code complexity may influence our results. **Internal validity** seeks to establish that systematic errors and biases do not invalidate the conclusions from a study. As we have used a dataset whose processes of collection and curation have been published in a peer-reviewed paper [6], and the dataset includes historical information, we do not expect mortality and maturation of the subjects to represent notable threats to internal validity. However, the extent to which code review information in the development ecosystem being studied was captured in the available dataset can be a threat to internal validity. As the paper presenting the dataset mentioned that the review process was captured in a review management system and the data was derived from it [6], we believe the threat to internal validity to be minimal. **External validity** of a study relates to the generalizability of its results. In this paper, we report results from a single study; while we believe the results are informative and useful, we do not claim them to be generalizable as yet. **Reliability** of a study comes from the reproducibility of its results. With access to the dataset, our results are fully reproducible and we share the code components used in our analysis (https://github.com/AmritaBh/markov-network). In our **future work** extending these preliminary results, we plan to address the threat to external validity by replicating the results from this study across other datasets. Additionally, our next objective is to analyse contents of the developer comments to help us better understand the flow of information between developers. This can enable a deeper examination of 2nd order Markov model dynamics in our study setting.

IV. Conclusion

In this paper, we present results from a study to understand how developer characteristics relate to importance of their communication channels in large software development ecosystems. Using data from a real-world project, we construct a developer interaction network and transform it using a 2nd order Markov model. The original network and the transformed network allow us to extract quantitative measures of the attributes of developers, as well as those of the communication links between them. Statistical models built using these attributes lead us to the counter-intuitive result that higher link importance is related to higher levels of influence of *both* the sender and the receiver of information across the links, even after controlling for the effects of workload, task completion times, experience, and levels of clustering of senders as well as receivers. Our results can inform various aspects of team assembly and project governance and they offer insights leading to wider investigations of link importance in large scale software development.

References

[1] K. Ehrlich and M. Cataldo, "All-for-one and one-for-all?: a multi-level analysis of communication patterns and individual performance in geographically distributed software development," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, ser. CSCW '12. New York, NY, USA: ACM, 2012, pp. 945–954.

[2] F. P. Brooks, *The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition*. Addison-Wesley, 1995.

[3] A. Schroter, J. Aranda, D. Damian, and I. Kwan, "To talk or not to talk: factors that influence communication around changesets," in *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, ser. CSCW '12. New York, NY, USA: ACM, 2012, pp. 1317–1326.

[4] P. Wagstrom and S. Datta, "Does latitude hurt while longitude kills? geographical and temporal separation in a large scale software development project," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 199–210.

[5] M. E. J. Newman, "The structure and function of complex networks," *cond-mat/0303516*, Mar. 2003, SIAM Review 45, 167-256 (2003).

[6] K. Hamasaki, R. G. Kula, N. Yoshida, A. E. C. Cruz, K. Fujiwara, and H. Iida, "Who does what during a code review? datasets of oss peer review repositories," in *Proceedings of the 10th Working Conference on Mining Software Repositories*, ser. MSR '13. Piscataway, NJ, USA: IEEE Press, 2013, pp. 49–52. [Online]. Available: http://dl.acm.org/citation.cfm?id=2487085.2487096

[7] S. Datta, D. Bhatt, M. Jain, P. Sarkar, and S. Sarkar, "The Importance of Being Isolated: An Empirical Study on Chromium Reviews," in *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, Oct. 2015, pp. 1–4.

[8] J. Xu, T. L. Wickramarathne, and N. V. Chawla, "Representing higher-order dependencies in networks," *Science advances*, vol. 2, no. 5, p. e1600028, 2016.

[9] M. Rosvall, A. V. Esquivel, A. Lancichinetti, J. D. West, and R. Lambiotte, "Memory in network flows and its effects on spreading dynamics and community detection," *Nature communications*, vol. 5, p. 4630, 2014.

[10] R. Xiang, J. Neville, and M. Rogati, "Modeling relationship strength in online social networks," in *Proceedings of the 19th international conference on World wide web*. ACM, Apr. 2010, pp. 981–990.

[11] E. Gilbert and K. Karahalios, "Predicting Tie Strength with Social Media," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '09. New York, NY, USA: ACM, 2009, pp. 211–220, event-place: Boston, MA, USA.

[12] S. Datta, "How does developer interaction relate to software quality? an examination of product development data," *Empirical Software Engineering*, vol. 23, no. 3, pp. 1153–1187, Jun. 2018.

[13] B. Tabachnick and L. Fidell, *Using Multivariate Statistics*. Boston: Pearson Education, 2007.