Tulane Economics Working Paper Series

Winmail3: An automated email package with an application to correspondence audit tests

Luca Fumarco
Tulane University
lfumarco@tulane.edu

S. Michael Gaddis
University of California, Los Angeles
mgaddis@soc.ucla.edu

Iain Snoddy
Analysis Group
iainsnoddy@gmail.com

Working Paper 2110
June 2021

Abstract

Correspondence audits are a popular method to examine discrimination in a causal framework. However, correspondence audits often require sending hundreds or thousands of emails to subjects. The Winmail3 package allows users to automatically send emails with Stata through PowerShell, which is open-source and cross-platform. Researchers can use this package to perform basic email tasks, such as contacting students or colleagues with standardized messages. Additionally, researchers can perform more complex tasks that entail sending randomized messages with multiple attachments from multiple accounts, tasks that are often necessary to conduct correspondence audit tests. This paper introduces the command and illustrates multiple examples of its application. We believe that researchers can apply this package to correspondence audit tests to save time and money.

Keywords: correspondence audit tests, field experiments, automation, PowerShell, email
JEL codes: C8, C93

Winmail3: An automated email package with an application to correspondence audit tests.

Fumarco, L.,[1] Gaddis, S. M.,[2] Snoddy, I.,[3]

[1] Tulane University, IZA, GLO; lfumarco@tulane.edu

[2] University of California, Los Angeles; mgaddis@soc.ucla.edu

[3] Analysis Group; iainsnoddy@gmail.com

Abstract. Correspondence audits are a popular method to examine discrimination in a causal framework. However, correspondence audits often require sending hundreds or thousands of emails to subjects. The Winmail3 package allows users to automatically send emails with Stata through PowerShell, which is open-source and cross-platform. Researchers can use this package to perform basic email tasks, such as contacting students or colleagues with standardized messages. Additionally, researchers can perform more complex tasks that entail sending randomized messages with multiple attachments from multiple accounts, tasks that are often necessary to conduct correspondence audit tests. This paper introduces the command and illustrates multiple examples of its application. We believe that researchers can apply this package to correspondence audit tests to save time and money.

JEL-Classification: C8, C93
Keywords: correspondence audit tests, field experiments, automation, PowerShell, email.

1 Correspondence Audit Tests

For nearly sixty years, researchers have used correspondence audit studies to covertly detect discrimination on the basis of race/ethnicity, gender, age, sexual orientation, gender identity, disability, and other characteristics (Button et al. 2020; Carlsson and Rooth 2007; Gaddis 2018a, 2018b; Cherry and Bendick 2018; Fumarco 2017; Neumark, Burn, and Button 2019; Pedulla 2016; Quadlin 2018). Audit studies are field experiments in which a researcher manipulates one or more characteristics (e.g., race or gender) to examine the effects of those characteristics on a variety of outcomes (e.g., replies to job applications or email responses from bureaucrats). Social scientists have increasingly used this research method in recent years for two reasons. First, audits allow researchers to make strong causal claims about discrimination and avoid social desirability bias that often plagues surveys (Gaddis 2018a; Gaddis and Ghoshal 2019; Pager and Quillian 2005). Second, audits can now detect discrimination in a variety of contexts due to online shifts in various applications processes and easy communication with a wide range of actors (see recent reviews and meta-analyses: Baert 2018; Gaddis 2018a; Gaddis and DiRago 2021; Oh and Yinger 2015; Quillian et al. 2017; Rich 2014; Zschirnt and Ruedin 2016).

In the age of ubiquitous internet correspondence, the shift to correspondence audits – as opposed to in-person audits – has eliminated some methodological concerns while introducing new ones (Gaddis 2017a, 2017b, 2018a, 2019a, 2019b, 2019c; Heckman and Siegelman 1993; Lahey and Beasley 2018; Larsen 2020; Vuolo, Uggen, and Lageson 2018; Zschirnt 2019). In correspondence audits, researchers contact research subjects (e.g., employers, landlords) via correspondence (e.g., emails, applications) posing as individuals who are interested in an opportunity or requesting information. However, the process of designing and collecting data for a correspondence audit is time-consuming and painstaking. Correspondence audits often require technological knowledge that goes beyond the average researcher's standard statistical and

coding skill-set. Researchers gain efficiency by mastering techniques – or borrowing and adapting code and packages from others – that reduce workload and automate much of the process of implementing a correspondence audit study and the ensuing data collection (Lahey and Beasley 2009, 2018). Some researchers have created solutions that work for specific types of studies (Lahey and Beasley 2009), created one-off custom solutions (Gaddis 2015; Gaddis and Ghoshal 2015, 2020), or suggested freelance hiring platforms (Crabtree 2018).

Researchers need simplified and standardized solutions for the different steps that comprise the data collection process of correspondence audits. Because there are many variations of correspondence audits across contexts and disciplines, standardizing an automated process of data collection is a difficult task. However, as Lahey and Beasley (2009) have shown, a semi-automated package or program can still save resources, even if it focuses on a narrow part of the design and data collection process, but still permits some customization. Researchers can benefit from a package that sends emails with a (previously) randomized text to ensure standardization, and to reduce the burden of data collection.

This article focuses on a Stata package designed to help researchers conserve resources by reducing the overall burden of data collection. We introduce Winmail3, which can automate sending emails with randomized components. Although others have developed codes to facilitate the mass email process in Python, SQL, and R (Chehras 2017; Crabtree 2018), ours is the first Stata program to do so. We believe this package will be useful to researchers conducting correspondence audits by simplifying and standardizing the data collection process. In the following sections of this article, we discuss the details of this command and provide examples to help the reader better understand the proper usage and full potential of Winmail3.

2 A Command for Standardizing and Automatically Sending Email: Winmail3

2.1 Syntax

Winmail3 uses Windows PowerShell to send emails.

This is the syntax of Winmail3:

winmail3 recipient@email.com [, s(subject) b(body) attone(attachment1) atttwo(attachment2) folder(folder name) html(email html) par(paragraphs) mailps(name_ps1_file) psloc(folder_name) smtpport(smtp port) smtpsserver(smtp server) from(name) sleep(time gap between emails) ufile(username_file) pfile(password_file) cc(cc_recipient) bcc(bcc_recipient) nossl]

2.2 Options

recipient@email.com is the recipient of the email. It has to be specified.

s(subject) declares the email subject. It has to be specified.

b(body) specifies the body of the email to be sent. The body should be input as a single string with blocks of text being parsed by "|". Each substring separated by "|" will be numbered and modified using html wrappers if specified. Text inputted as "line1 | line2 | line3" will be treated as 3 separate blocks of text and numbered "1, 2, 3". It has to be specified.

attone(attachment1) gives the location and name of one file to be included as an attachment to the email.

atttwo(attachment2) gives the location and name of one file to be included as an attachment to the email.

attone and *atttwo* do not stand as the order or priority of the files: you can send atttwo even if attone is empty. *attone* and *atttwo* allow you to send only one file each, but the location of these two files might different.

folder(folder_name) gives the folder location for all of the attachments you want to send at once. There is no limit to the amount of files that can be extracted from the folder. If you want all of the files in the folder, after the folder name you write "*.*"; assuming the folder is called "robe" then folder_name = "C:\Users\luca_\Desktop\robe*.*". If you want all of the files with a certain extension, the second * has to be substituted by the extension, e.g., "*.pdf"; assuming you the folder is called "robe" then folder_name = "C:\Users\luca_\Desktop\robe*.pdf"

html(email html) provides the html to be included in the email. The html provided modifies the text provided in body. As each block of text parsed by "|" in body is numbered, html should be provided as wrappers around these numbers. Users should input a string where html wrappers are placed around numbers, with each number representing a substring of body. For example, "1 2 3 4" makes all text in substrings 1 and 2 italic and text in substring 2 bold. If html is specified it must include a number for every substring. If there are 4 inputted substrings and html takes input " 1 2 3" then the fourth string will be missing from the email. You can create paragraphs using html or using par.

par(paragraphs) provides paragraph breaks between substrings parsed by "|" in body. The input to this option should take the format of a numbered list, for example par(1 3 5) creates a new paragraph following substrings 1, 3 and 5.

mailps(name_ps1_file) gives the name of the .ps1 file, default is "mailps.ps1".

psloc(folder_name) gives the folder location of where the .ps1 file will be saved, default location is working directory.

smtpport(smtp port) gives the smtp port number. By default it is set to that used by gmail.

smtpsserver(smtp server) gives the smtp server address which by default is the gmail smtp server.

from(name) is the name of the sender, default is username.

sleep(time gap between emails) is the time gap between multiple email, default is 3.000.

ufile(username_file) gives the location and filename of the user's email address saved as plain text in a .txt file; default location is working directory. The file extension should not be included. The file name has to be specified.

pfile(password_file) gives the location and filename of the user's password saved as a secure string in a .txt file; default location is working directory. The file extension should not be included. The file name has to be specified.

cc(cc_recipient) is the email address included as a cc to the email.

bcc(bcc_recipient) is the email address included as a bcc to the email.

nossl specifies that the Secure Sockets Layer (SSL) to establish a connection is not used.

3 Technical Details

PowerShell was originally a Windows component exclusively; however, on 18 August 2016 it was made open-source and cross-platform. If you do not have already PowerShell (e.g. if you have an Apple computer), you can download it here: https://github.com/PowerShell/PowerShell

Emails are sent using the Send-MailMessage cmdlet (information here: https://msdn.microsoft.com/en-us/powershell/reference/5.1/microsoft.powershell.utility/send-mailmessage).

Winmail3 sends email using SMTP (Simple Mail Transfer Protocol). By allowing for the inclusion of html the emails created can be quite versatile. User credentials are captured by the program, the recipients email address and features to be included in the email. A .ps1 script is

created and run using Windows PowerShell. This file is stored on the user's system in a specified location along with user's credentials.

Users must specify the location of their email address and password. By allowing for paragraphing and the inclusion of html, users are provided with substantial flexibility in the format of their email.

By default, the smtp settings are set for the use of gmail accounts. These settings can be modified to work for other email account types. For instance, to send an email using outlook.com the stmpserver should be set to "smtp-mail.outlook.com" and the stmpport to "587". Different email accounts may require users to modify settings prior to the use of smtp. For instance, gmail requires that users allow access to less secure apps before an email can be sent using smtp via PowerShell. The speed of email delivery may also differ across services.

Users who have not used Windows PowerShell before should note that by default it may not have sufficient administrative privileges to perform the operations in Winmail3. First time users should run "Set-ExecutionPolicy RemoteSigned" when running Windows PowerShell as administrator and follow the instructions on screen. This command specifies that scripts created on the current system and files with a digital signature may be run. This keeps PowerShell privileges quite strict but is sufficient for this program as the .ps1 script executed is created locally.

Owed to the configuration of Send-MailMessage cmdlet, some location and file names are restricted, and some options are mandatory. While the locations of *attone(attachment1)* and *atttwo(attachment2)* can contain blank spaces, the location of *folder(folder_name)* cannot. The location within parenthesis in *pfile*(*password_file)*, *ufile*(*username_file)*, *psloc(folder_name)*

cannot contain blank spaces. The following options are mandatory: *ufile(username_file)*, *pfile(password_file)*, and *s(subject)*.

You cannot specify *attone(attachment1)* and/or *atttwo(attachment2),* when you specify already *folder(folder_name).*

While the *b(body)* is not mandatory within Send-MailMessage cmdlet, here it is mandatory to prevent mistakes that would increase chances of being detected.

Winmail3 requires tknz be installed.

4 Examples

In this section, we discuss the basic usage of the command in some common cases.

4.1 One standard email

Let us assume the user wants to send only one simple email—without selecting the message components from a previously randomized dataset. Below, we go through a few basic versions.

4.1.1 Example 1: Sending one email with attachment and cc

With this example, the user creates two text files with the email provider account details and then sends a simple email.

Let us create txt files where we store the password and the username of the email:

```
clear all
cd "C:\User\yourcd"

file open myfile using "password.txt", write replace
file write myfile  "mypassword"
file close myfile
file open myfile using "username.txt", write replace
file write myfile  "your_email"
file close myfile
```

Now, the let us send a one liner email and provide the location and name of both password and username files. Let us further assume that your email provide is outlook, you want to attach a txt file and put in cc friend2.

Let us give Stata the command:

winmail3 friend@email.com, b(hello friend) s(hi) ufile(C:User\username) pfile(C:User\password) smtpport(587) smtpserver(smtp-mail.outlook.com) attone(C:\User\file.txt) cc(friend2@email.com)

This will create an email which looks like:

hello friend

4.1.2 Example 2: Sending an email with paragraphs

With this example, the user sends a three liner email. Let us assume that the user has already created the two txt files, with email account password and username, that is, *"password.txt" and "username.txt"*.

Create a new paragraph following the first and second substrings:

local body "hello friend, | How are you doing? | Best, Friend"
winmail3 friend@email.com, b(`body') s(hi) par(1 2) ufile(C:User\username) pfile(C:User\password) smtpport(587) smtpserver(smtp-mail.outlook.com)

This will create an email which looks like:

hello friend,
How are you doing?
Best, Friend

4.1.3 Example 3: Sending an email with html and bold words

With this example, the user sends a one liner email with some words in bold. Let us assume that the user has already created the two txt files, with email account password and username, that is, *"password.txt" and "username.txt"*.

local body "hello friend, | How are you doing?| Best, Friend"
winmail3 friend@email.com, b(`body') s(hi) html(1 2 3) ufile(C:User\username) pfile(C:User\password) smtpport(587) smtpserver(smtp-mail.outlook.com)

This will create an email which looks like:

hello friend, ***How are you doing?*** *Best, Friend*

Note that if we erroneously specified html(1 2) the output would be:

hello friend, How are you doing?

4.1.4 Example 4: Sending an email with html and paragraphs

With this example, the user sends a three liner email with html. Let us assume that the user has already created the two txt files, with email account password and username, that is, *"password.txt" and "username.txt"*. With html, you do not need to use *par()*.

The following uses of the Winmail3 command are equivalent:

```
local body "hello friend, | How are you doing?| Best, Friend"
winmail3 friend@email.com, b(`body') s(hi) html(1 2<br><br> 3) ufile(C:User\username) pfile(C:User\password) smtpport(587) smtpserver(smtp-mail.outlook.com)

winmail3 friend@email.com, b(`body') s(hi) par(2) ufile(C:User\username) pfile(C:User\password) smtpport(587) smtpserver(smtp-mail.outlook.com)
```

Both commands give the same result:

hello friend, How are you doing?
Best, Friend

4.1.5 Example 5: Many emails from randomized dataset

Let us assume the user wants to send many emails, over several days, with previously randomized components of the message—this is the usual setting of a correspondence audit test. Below, we create a one-observation dataset and show how to feed the email components to a loop for sending emails through Winmail3.

Let us assume that the user has already created the two txt files, with email account password and username, that is, *"password.txt" and "username.txt"*.

4.1.5.1 Create a dataset with one observation

This part of the example is to create the dataset with components of the message and email information.

```
clear all
cd "C:\User\yourcd"

set obs 1
```

Generate receiver email

```
cap drop emailfriend
gen emailfriend= "receiver@rossoneri.it"
```

Generate access email and its record

```
cap drop myemail username password
gen myemail= "myemail@nerazzurri.it"
gen username= "username"                    //note that username is the name of the file "username.txt" without extension
gen password= "password"                    //note that password is the name of the file "password.txt" without extension

cap drop server port
gen server = "smtp-mail.outlook.com" if myemail== "myemail@nerazzurri.it"
gen port = 587 if myemail== "myemail@nerazzurri.it"    //note that 587 is the real outlook.com port
```

Generate components of the message:

```
cap drop name surname salutation friend valediction
gen name= "Barbera "
gen surname= "DelMonferrato"
gen salutation= "Dear"
gen friend= " friend,"
gen valediction= "Ciao"

cap drop *_sent
gen first_sent= "Have you seen the new Stata command? Winmail3."
gen second_sent= "It allows you to send emails through Stata, with the help of PowerShell."

cap drop return subject
gen return= "|"
```

gen subject = "Great news"

Put the message together

cap drop body
egen body = concat(salutation friend return first_sent return second_sent return valediction return name surname)

Generate the date variable that tells us when a given email should be sent; we are going to loop over this variable when we send the emails.

cap drop date_send
gen date_send= "dd/mm/yyyy"

cap drop datestand
gen datestand = daily(date_send, "DMY") //Stata loops over the number of days since January 1, 1960

format datestand %td

save "C:\User\yourcd\originaldataset.dta", replace

save "C:\User\yourcd\workingdataset.dta", replace

4.1.5.2 Conduct the experiment

On each day you conduct the experiment, and thus have to send emails, you have to run the following do file.

clear all
cd "C:\User\yourcd"

use "C:\User\yourcd\ workingdataset.dta", clear

Generate today date

display "`c(current_date)'"
cap drop today
gen today=daily("`c(current_date)'", "DMY")
format today %td

cap drop identifier
gen identifier= _n if datestand==today //Stata gives an increasing number only to those emails for which the

planned date for the email delivery equals today

Tell Stata to create temporary information on when the email is actually sent:

```
cap drop date_sent
gen date_sent = “”
cap drop time_sent
gen time_sent = “”
```

Loop over identifier for which datestand==today:

```
set trace on
set more off
sum identifier if datestand==today
forval i= `r(min)'/`r(max)' {

cap drop `bodystr'
tempvar bodystr
gene `bodystr'=_n if identifier==`i' & datestand==today
summ `bodystr', meanonly
local index=r(mean)
local b=body[`index']

cap drop `serverstr'
tempvar serverstr
gene `serverstr'=_n if identifier==`i' & datestand==today
summ `serverstr', meanonly
local index=r(mean)
local s=server[`index']

cap drop `portstr'
tempvar portstr
gene `portstr'=_n if identifier==`i' & datestand==today
summ `portstr', meanonly
local index=r(mean)
local po=port[`index']

cap drop `userstr'
tempvar userstr
gene `userstr'=_n if identifier==`i' & datestand==today
summ `userstr', meanonly
local index=r(mean)
local u=username[`index']

cap drop `passstr'
tempvar passstr
```

```
gene `passstr'=_n if identifier==`i' & datestand==today
summ `passstr', meanonly
local index=r(mean)
local pa=password[`index']

cap drop `friendstr'
tempvar friendstr
gene `friendstr'=_n if identifier==`i' & datestand==today
summ `friendstr', meanonly
local index=r(mean)
local r=emailfriend[`index']

cap drop `subjectstr'
tempvar subjectstr
gene `subjectstr'=_n if identifier==`i' & datestand==today
summ `subjectstr', meanonly
local index=r(mean)
local sbj=subject[`index']

        winmail3 `r', ///
        b(`b') par(1 2 3 4)  s(`sbj')  ///
        smtpport(`po') smtpserver(`s') ///
        ufile(`u') ///
        pfile(`pa') ///
        folder("C:\User\yourdc\ApplicationPackage\*.*") ///        //this is the folder with all of the application material of your fictitious person (e.g. one cv and one cover letter in pdf format and your one professional picture in jpg)

        psloc("C:\User\yourcd\")

        replace date_sent="`c(current_date)'" if identifier==`i'
        replace time_sent="`c(current_time)' " if identifier==`i'

}
replace deliverydate = date_sent if deliverydate== ""
replace deliverytime = time_sent if deliverytime== ""

save "C:\User\yourcd\ workingdataset.dta", replace
```

5 Discussion

5.1 Feedback

You can find this package at https://sites.google.com/site/lucafumarco/stata-codespackages. If you want to report a bug or request a feature, please send an email to Luca Fumarco or Iain Snoddy.

5.2 Conclusion

The Winmail3 package allows users to automatically send emails with Stata through PowerShell, which is open-source and cross-platform. Using this package researchers can perform basic email tasks, such as contacting students or colleagues with standardized messages, or more complex ones, such as conducting correspondence audit tests. We believe that the specific application of Winmail3 to correspondence audit tests will help users save time and resources. Correspondence audit tests typically require sending hundreds or thousands of individual emails; with Winmail3 users can send these emails faster and for free, significantly slashing the usual experimental budget.

References

Baert, Stijn. 2018. “Hiring Discrimination: An Overview of (almost) All Correspondence Experiments since 2005.” Pp. 63–77 in *Audit Studies: Behind the Scenes with Theory, Method, and Nuance*, edited by S. M. Gaddis. Cham, Switzerland: Springer.

Button, Patrick, Eva Dils, Benjamin Harrell, Luca Fumarco, and David Schwegman. 2020. “Gender Identity, Race, and Ethnicity Discrimination in Access to Mental Health Care: Preliminary Evidence from a Multi-Wave Audit Field Experiment.” *NBER* Working Paper. Available at: https://www.nber.org/papers/w28164

Carlsson, Magnus, and Dan-Olof Rooth. 2007. “Evidence of Ethnic Discrimination in the Swedish Labor Market Using Experimental Data.” *Labour Economics*, 14(4):716-29.

Chehras, N. 2017. Automating correspondence study applications with python and SQL: Guide and code. Mimeo.

Cherry, Frances, and Marc Bendick. 2018. “Making It Count: Discrimination Auditing and the Activist Scholar Tradition.” Pp. 45–62 in *Audit Studies: Behind the Scenes with Theory, Method, and Nuance*, edited by S. M. Gaddis. Cham, Switzerland: Springer.

Crabtree, Charles. 2018. “An Introduction to Conducting Email Audit Studies.” Pp. 103–117 in *Audit Studies: Behind the Scenes with Theory, Method, and Nuance*, edited by S. M. Gaddis. Cham, Switzerland: Springer.

Fumarco, Luca. 2017. “Disability Discrimination in the Italian Rental Housing Market: A Field Experiment with Blind Tenants.” *Land Economics*, 93(4):567-84.

Gaddis, S. Michael. 2015. “Discrimination in the Credential Society: An Audit Study of Race and College Selectivity in the Labor Market.” *Social Forces* 93(4):1451–79.

Gaddis, S. Michael. 2017a. “How Black Are Lakisha and Jamal? Racial Perceptions from Names Used in Correspondence Audit Studies.” *Sociological Science* 4(19):469–89.

Gaddis, S. Michael. 2017b. "Racial/Ethnic Perceptions from Hispanic Names: Selecting Names to Test for Discrimination." *Socius* 3:1–11.

Gaddis, S. Michael. 2018a. "An Introduction to Audit Studies in the Social Sciences." Pp. 3–44 in *Audit Studies: Behind the Scenes with Theory, Method, and Nuance*, edited by S. M. Gaddis. Cham, Switzerland: Springer.

Gaddis, S. Michael. 2018b. *Audit Studies: Behind the Scenes with Theory, Method, and Nuance*. Cham, Switzerland: Springer.

Gaddis, S. Michael. 2019a. "Understanding the 'How' and 'Why' Aspects of Racial-Ethnic Discrimination: A Multimethod Approach to Audit Studies." *Sociology of Race and Ethnicity*, 5(4):443-55.

Gaddis, S. Michael. 2019b. "Signaling Class: An Experiment Examining Social Class Perceptions from Names Used in Correspondence Audit Studies." *SSRN* Working Paper. Available at https://papers.ssrn.com/abstract=3350739

Gaddis, S. Michael. 2019c. "Assessing Immigrant Generational Status from Names: Evidence for Experiments Examining Racial/Ethnic and Immigrant Discrimination." *SSRN* Working Paper. Available at https://papers.ssrn.com/abstract=3022217

Gaddis, S. Michael. 2021. "The Honeypot Audit: Reversing the Correspondence Audit Method." *SSRN* Working Paper. Available at https://papers.ssrn.com/abstract=3179222

Gaddis, S. Michael, and Nicholas DiRago. 2021. "Housing Audit Studies in the Social Sciences." *SSRN* Working Paper. Available at https://papers.ssrn.com/abstract=3796335

Gaddis, S. Michael, and Raj Ghoshal. 2015. "Arab American Housing Discrimination, Ethnic Competition, and the Contact Hypothesis." *Annals of the American Academy of Political and Social Science* 660(1):282–99.

Gaddis, S. Michael, and Raj Ghoshal. 2019. "Dynamic Racial Triangulation: Examining the Racial Order Using Two Experiments on Discrimination among Millennials." *SSRN* Working Paper. Available at https://papers.ssrn.com/abstract=3022208

Gaddis, S. Michael, and Raj Ghoshal. 2020. "Searching for a Roommate: A Correspondence Audit Examining Racial/Ethnic and Immigrant Discrimination among Millennials." *Socius: Sociological Research for a Dynamic World*.

Gaddis, S. Michael, and Edvard Nergård Larsen. 2021. "Auditing Audit Studies: The Effects of Name Perception and Selection on Social Science Measurement of Racial Discrimination." *SSRN* Working Paper. Available at https://papers.ssrn.com/abstract=3022207

Heckman, James J., and Peter Siegelman. 1993. "The Urban Institute Audit Studies: Their Methods and Findings." Pp. 187–258 in *Clear and Convincing Evidence: Measurement of Discrimination in America*, edited by M. Fix and R. J. Struyk. Washington, DC: The Urban Institute Press.

Lahey, Joanna N., and Ryan A. Beasley. 2009. Computerizing audit studies. *Journal of Economic Behavior & Organization, 70*(3), 508–514.

Lahey, Joanna N., and Ryan A. Beasley. 2018. "Technical Aspects of Correspondence Studies." Pp. 81–101 in *Audit Studies: Behind the Scenes with Theory, Method, and Nuance*, edited by S. M. Gaddis. Cham, Switzerland: Springer.

Larsen, Edvard N. 2020. "Induced Competition in Matched Correspondence Tests: Conceptual and Methodological Considerations. *Research in Social Stratification and Mobility*.

Neumark, David, Ian Burn, and Patrick Button. 2019. "Is It Harder for Older Workers to Find Jobs? New and Improved Evidence from a Field Experiment." *Journal of Political Economy*¸127(2):922-70.

Oh, Sun Jung, and John Yinger. 2015. "What Have We Learned from Paired Testing in Housing Markets?" *Cityscape* 17(3):15–60.

Pager, Devah, and Lincoln Quillian. 2005. "Walking the Talk? What Employers Say Versus What They Do." *American Sociological Review* 70(3):355–80.

Pedulla, David. 2016. "Penalized or Protected? Gender and the Consequences of Nonstandard and Mismatched Employment Histories." *American Sociological Review*, 81(2):262-89.

Quadlin, Natasha. 2018. "The Mark of a Woman's Record: Gender and Academic Performance in Hiring." *American Sociological Review*, 83(2):331-60.

Quillian, Lincoln, Devah Pager, Ole Hexel, and Arnfinn Midtbøen. 2017. "The Persistence of Racial Discrimination: A Meta-analysis of Field Experiments in Hiring over Time." *Proceedings of the National Academy of Sciences* 114(41):10870–5.

Rich, Judith. 2014. "What Do Field Experiments of Discrimination in Markets Tell Us? A Meta-analysis of Studies Conducted since 2000." *IZA* Working Paper. Available at https://www.iza.org/publications/dp/8584/what-do-field-experiments-of-discrimination-in-markets-tell-us-a-meta-analysis-of-studies-conducted-since-2000.

Vuolo, Mike, Christopher Uggen, and Sarah Lageson. 2018. "To Match or Not to Match? Statistical and Substantive Considerations in Audit Design and Analysis." Pp. 119–40 in *Audit Studies: Behind the Scenes with Theory, Method, and Nuance*, edited by S. M. Gaddis. Cham, Switzerland: Springer.

Zschirnt, Eva. 2019. "Research Ethics in Correspondence Testing: An Update." *Research Ethics*, 15(2):1-21.

Zschirnt, Eva, and Didier Ruedin. 2016. “Ethnic Discrimination in Hiring Decisions: A Meta-analysis of Correspondence Tests 1990–2015.” *Journal of Ethnic and Migration Studies* 42(7):1115–34.