

EPO-193

Extracting Email Domains and Geo-Processing IP Addresses in SAS®

Alec Zhixiao Lin, LoanDepot, Foothill Ranch, CA

ABSTRACT

Web data has become a very important source for analytics in the current era of social media and booming ecommerce. Email domain and IP address are two important attributes potentially useful for market sizing, detection of online statistical anomaly and fraud prevention. This paper introduces a few methods in SAS that extract email domains and process IP addresses to prepare data for subsequent analyses.

INTRODUCTION

Many companies in ecommerce look for innovative ways to know their customers better and to build a safe and secure environment for online transactions. Email address and IP address are two attributes that are commonly used for these purposes, as illustrated by the following examples:

- 1) Over 98% of emails are from such common domains as yahoo, gmail, hotmail, aol, junos, etc. If one observes a sudden rise by logins from one or several obscure email domains in a very short span of time, this could suggest some statistical anomaly and call for an investigation into what might have happened.
- 2) Accurately identifying the geographic locations of existing and prospective customers by IP address can help a business to know where their customers are. Combined with geo-level demographic data related to income, age and lifestyles, one can gain an understanding of customer base and additional business opportunities.
- 3) In ecommerce, a buyer's IP address matching to the city and/or zip code of the shipping address is usually a good measure for his/her authenticity. A sudden rise of online customers whose billing addresses are far from their IP addresses – for example, as triggered by a large number of logins via VPN – could trigger an alert for fraud prevention.

This paper introduces a few methods in SAS that will help to achieve the above business purposes.

EXTRACTING EMAIL DOMAINS

Let's use *John.Smith@emaildomain.com* as an example to illustrate the components of an email address:

- 1) A local part, usually the ID of an email holder, e.g., *John.Smith* in our example. This ID is usually case-insensitive.
- 2) An @ symbol. This is the most useful byte in extracting email domain.
- 3) Name of the email domain, such as *emaildomain.com* in our example. Domain name is case-insensitive.

The following are the steps we can take to extract an email domain.

- 1) Find the position of @ (11th in the example).
- 2) Truncate the email from @ on. In our example, we will be interested in *emaildomain.com*.
- 3) Find the position for the dot (.) between *emaildomain* and *com*.
- 4) Extract the email domain by specifying the location of the byte preceding and after it.

Suppose the email addresses are stored in the field *email* in a data set, we can use the following code to extract their domain names:

```
email_domain1=find(email, "@");
email_truncate=substr(email, email_domain1+1);
email_domain2=find(email_truncate, ".");
email_domain_name=lowercase(substr(email_truncate, 1, email_domain2-1));

drop email_domain1 email_domain2 email_truncate;
```

In the example, the domain name will be retained in the field `email_domain_name`. Users can use such procedures PROC FREQ to examine the login frequency of different email domains.

PROCESSING IP ADDRESSES

An IP directory can map IP addresses to county, city, zip code, etc. However, a directory that lists all IP addresses could be quite big in file size. Many vendors present their IP directory in the following manner:

ip_from	ip_to	country_short	country_name	region	city	latitude	longitude	zipcode
3521813760	3521814271	US	United States	California	Mission Viejo	33.60002	-117.672	92675

Table1- an example of IP directory

In the above example, *ip_from* is the beginning IP number and *ip_to* is the ending IP number, with the range suggesting the number of IP addresses contained in the row. The following example illustrates how to convert an IP address to an IP number.

IP address: 209.234.157.58

IP number: $209 \times 256^3 + 234 \times 256^2 + 157 \times 256 + 58 = 3521813818$

We present two methods in SAS to achieve the same purpose.

METHOD 1

This method takes three steps to complete.

Step1 - Expand the IP directory to all IP numbers

Suppose you have an IP directory *ipgeocoding_prep* with IP number named as *IP_number*, the following code can be used to obtain all IP addresses.

```
proc sort data=ipgeocoding_prep; by ip_from; run;
data ipgeocoding_prepared(drop=ip_number_temp);
set ipgeocoding_prep;
by ip_from;

ip_number_temp=ip_from;
do until(ip_number_temp=ip_to+1);
ip_number_temp+1;
ip_number=ip_number_temp-1;
output;
end;
run;
```

The code above will generate the following expanded IP directory that lists all IP_numbers in the column *ip_number*.

ip_from	ip_to	country_short	country_name	region	city	latitude	longitude	zipcode	ip_number
3521813760	3521814271	US	United States	California	Mission Viejo	33.60002	-117.672	92675	3521813760
3521813760	3521814271	US	United States	California	Mission Viejo	33.60002	-117.672	92675	3521813761
(omitting 507 IP numbers here.)									
3521813760	3521814271	US	United States	California	Mission Viejo	33.60002	-117.672	92675	3521814269
3521813760	3521814271	US	United States	California	Mission Viejo	33.60002	-117.672	92675	3521814270
3521813760	3521814271	US	United States	California	Mission Viejo	33.60002	-117.672	92675	3521814271

Table2 – IP directory expanded to contain all IP addresses

The field *ip_number* will be used as a linkage between the IP directory and a customer file.

Step2 - Convert IP addresses into IP numbers in the customer file

The following SAS code will convert IP address to IP number.

```
data customerfile2;
set customerfile;

ip_scan1=input(scan(IPAddress, 1), 4.);
ip_scan2=input(scan(IPAddress, 2), 4.);
ip_scan3=input(scan(IPAddress, 3), 4.);
ip_scan4=input(scan(IPAddress, 4), 4.);

ip2_number=256*256*256*ip_scan1+256*256*ip_scan2;
ip3_number=256*256*256*ip_scan1+256*256*ip_scan2+256*ip_scan3;
ip4_number=256*256*256*ip_scan1+256*256*ip_scan2+256*ip_scan3;
ip_number=256*256*256*ip_scan1+256*256*ip_scan2+256*ip_scan3+ip_scan4;

if ip2_number=. then delete;
drop IP_scan: ip2_number ip3_number ip4_number;
run;
```

Step 3 – Merge the outcome files from Step1 and Step2

The field *IP_number* that appears in both files will be used as the key for merging.

```
proc sort data=customerfile2; by ip_number; run;
proc sort data=ipgeocoding_prepared; by ip_number; run;

data ipgeocoding_completed;
merge customerfile2(in=s)
      ipgeocoding_prepared(in=t);
by ip_number;
if s;

if t then IP_geocoded=1; else IP_geocoded=0;
run;
```

Please note that we have also added a dummy field *IP_geocoded*. It can be used to examine the coverage of records that have been successfully geocoded.

METHOD 2

Even though the first method follows the full logic of conversion between IP numbers and IP addresses, the file size will usually increase by 300 times as all IP addresses need to be listed before merging¹. For conserving system resources, there is a shortcut that avoids expanding an IP directory to all possible IP addresses. Let's use the same example to illustrate.

ip_from associated with 209.234.157.58: $209 \times 256^3 + 234 \times 256^2 + 157 \times 256 = 3521813860$

Please note that we did not add the last two digits 58 in the calculation. The following is the SAS code used:

```
data customerfile2;
set customerfile;

ip_scan1=input(scan(IPAddress, 1), 4.);
ip_scan2=input(scan(IPAddress, 2), 4.);
ip_scan3=input(scan(IPAddress, 3), 4.);

ip_from=256*256*256*ip_scan1+256*256*ip_scan2+256*ip_scan3;
drop IP_scan:;
run;
```

¹ Most rows in a IP directory contain 256 IP numbers. Some could contain 512 IP numbers.

Readers might have noticed that we did not care to include `ip_scan4` as in Method 1. We can now just use `ip_from` to merge the IP directory and a customer file for geocoding.

```
proc sort data=customerfile2; by ip_from; run;
proc sort data=ipgeocoding_prep; by ip_from; run;

data ipgeocoding_completed;
merge customerfile2(in=s)
      ipgeocoding_prep(in=t);
by ip_from;
if s;

if t then IP_geocoded=1; else IP_geocoded=0;
run;
```

The two methods give identical results, but the second one is much more efficient.

CONCLUSION

This paper introduces the logics for identifying email domains and for processing IP addresses. We hope the SAS codes provided will help users to accomplish some of the related tasks.

ACKNOWLEDGEMENTS

I would like to thank the team in Loan Depot led by Shawn Benner for their continuous support for this endeavor.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at

Alec Zhixiao Lin

VP of Modeling

Loan Depot

26642 Towne Center Drive

Foothill Ranch, CA 92610

Email: alecindc@gmail.com

Web: www.linkedin.com/pub/alec-zhixiao-lin/25/708/261/

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® Indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.