ABSTRACT

“Should I change my starting lineup?” This is the dreaded question all 40 million fantasy football participants have asked at one time or another. In particular, a player’s injury report can make or break a fantasy football participant’s lineup for the week. The unlikely combination of fantasy football with DO loops can assist in alleviating this concern. Often overlooked due to their simplicity, DO loops are a vital key to your SAS® programming. When used correctly, they are a main driver for data or code productivity. In this paper, we discuss the use of DO loops with SAS features such as macro variables and text notifications. Each of these features plays a role in fulfilling your business needs at a faster and more efficient rate.

INTRODUCTION

It does not matter if you are Julius Peppers or J.J. Watt. All National Football League (NFL) players are prone to injury. Aaron Rodgers, Dalvin Cook, Odell Beckham Jr., and Zac Ertz are game-changing players who have been injured at some point in the 2017-2018 season. Reliance on these players (and others) is essential for fantasy football participants. How do you know who to start once an injury occurs?

“Are they still injured?” “What is their injury status?” “Will they be active before game time?” These are a few of the questions that fantasy football participants ask once a player on their team has been identified as injured. It is critical that you remove injured players from your starting lineup before game time; otherwise, you could end up with no points for that player and position. The steps I discuss in this paper will help lessen these anxieties.

In addition to creating a clever fantasy football team name, removing inactive players from your roster is essential for team success. Notice in Figure 1 how Sterling Shepard was inactive and provided Football Jeasus with 0 points. If Football Jeasus had replaced Shepard with an active player, the fantasy team could have won that week. Fantasy football applications might notify you of an injured player if you select...
that option, but there is no guarantee that the application will notify you prior to game time. Internet problems might also prevent you from receiving timely notification. I will address this type of challenge in this paper.

This paper will explore a way to give your fantasy football team an edge next season. After being provided with a list of NFL players, this SAS program will use a Google Custom Search API to retrieve the most recent and relevant injury update on your players. The Google Custom Search API can be used for SAS 9.4 M4 or higher, including SAS® VIYA®. The SAS program will then develop a text message from this information that contains a list of players you should review in your roster. You have the flexibility to schedule this text to be sent at an appropriate time in order to change your lineup.

**PREGAME**

Preparation work for athletes is essential for their game performance, and the same is true for this SAS code. Before getting to the heart of this paper, we need to make some key preparations. An API key and a Custom Search Engine (CSE) cx ID are required for the Google Custom Search to run. This custom search API returns the data in one of two formats: Atom or JSON. JSON is the default data format that is compatible with SAS programming. The documentation page for Google Custom Search (https://developers.google.com/custom-search/json-api/v1/overview) provides a link to request an API key that allows 100 free searches per day (see Figure 2). There is no need to restrict the key’s usage for this SAS program. Once you have retrieved an API key (see Figure 3), the next step is to retrieve a cx ID.

**Figure 2. API Key Request**

![API key](https://developers.google.com/custom-search/json-api/v1/overview)

**Figure 3. API Key Confirmation**

![You’re all set! You’re ready to start developing with Custom Search API](https://developers.google.com/custom-search/json-api/v1/overview)
RETRIEVE CX ID

For each requested search, the CSE returns metadata about the search performed, metadata about the custom search engine used for the search, and the search results. See https://developers.google.com/custom-search/json-api/v1/reference/cse for more information. You must identify a particular search engine to use in your request (by using the cx query parameter). Following these steps will enable you to retrieve the cx ID.

1. In a browser, navigate to CSE home.

2. Click Add below Edit Search Engine.

3. For this example, we will use Google for our search site. Type www.google.com in the Sites to Search field. Click Create. See Figure 4.

4. Select Control Panel to modify your search engine.

5. Under the Basics tab, specify the search engine name.

6. Specify any additional sites you would like to search under Sites to Search. Figure 5 shows an example of several sports websites that will be searched.

7. Select Advanced under the Sites to search section. Add any sites you would like to exclude (optional). See Figure 5.

When attempting to investigate injury reports of players, several results returned sites that contained NFL team rosters, which muddles our search results on injuries. Excluding these sites drastically decreased these occurrences in the results and improved the effectiveness of the search.

8. In the Details section, select Get code.

9. Retrieve the value for cx ID (Figure 6). Your cx ID will be consistent throughout all searches.

Figure 4. Create Search Engine
Figure 5. Specify Sites

Figure 6. Retrieve cx ID
GAME TIME

Once you have received your API key and cx ID, code development can begin. First, create a data set for your fantasy football team (excluding defense). Using a DATALINES step, create a table containing the first and last name of each of your fantasy football players. See Program 1.

Program 1. Roster Code

```r
data Roster;
  input First$ Last$;
cards;
  Dak Prescott
  Jerick McKinnon
  Alvin Kamara
  Alshon Jeffery
  Rishard Matthews
  Jimmy Graham
  DeMarco Murray
  Harrison Butker
  Devonta Freeman
  Marshawn Lynch
  Kelvin Benjamin
  Ed Dickson
  Alex Collins
  Marqise Lee
  Zac Ertz
;```

Next, create two macro variables. One should be based on the players’ names, and the other will be used in the Google Custom Search API (Program 2). Create the second macro by inserting a “+” between the player’s first and last name. See Output 1.

Program 2. Macro Creation Code

```r
/* Create Macro variable for roster */
data roster;
  set roster;
  search=catx("+",first,last);
  player=catx(" ",first,last);
  call symputx('num',_n_);
run;

proc sql;
  select search, player
  into :roster1-:roster&num, :player1-:player&num
  from roster;
quit;
```
Output 1. Macro Variables

<table>
<thead>
<tr>
<th>search</th>
<th>player</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dak+Prescott</td>
<td>Dak Prescott</td>
</tr>
<tr>
<td>Jorick+McKinnon</td>
<td>Jorick McKinnon</td>
</tr>
<tr>
<td>Alvin+Kamara</td>
<td>Alvin Kamara</td>
</tr>
<tr>
<td>Alshon+Jeffery</td>
<td>Alshon Jeffery</td>
</tr>
<tr>
<td>Rishard+Matthews</td>
<td>Rishard Matthews</td>
</tr>
<tr>
<td>Jimmy+Graham</td>
<td>Jimmy Graham</td>
</tr>
<tr>
<td>Demarco+Murray</td>
<td>Demarco Murray</td>
</tr>
<tr>
<td>Harrison+Butker</td>
<td>Harrison Butker</td>
</tr>
<tr>
<td>DeVonta+Freeman</td>
<td>DeVonta Freeman</td>
</tr>
<tr>
<td>Marshawn+Lynch</td>
<td>Marshawn Lynch</td>
</tr>
<tr>
<td>Kelvin+Benjamin</td>
<td>Kelvin Benjamin</td>
</tr>
<tr>
<td>Ed+Dickson</td>
<td>Ed Dickson</td>
</tr>
<tr>
<td>Alex+Collins</td>
<td>Alex Collins</td>
</tr>
<tr>
<td>Marquise+Lee</td>
<td>Marquise Lee</td>
</tr>
<tr>
<td>Zac+Ertz</td>
<td>Zac Ertz</td>
</tr>
</tbody>
</table>

CUSTOM GOOGLE API

At the heart of this paper is the application of the Google Custom Search API, which allows you to develop websites and applications to retrieve and display search results from Google Custom Search programmatically. We will use the SAS macro “injury” to conduct a Google Custom Search on every football player on the fantasy football team (Program 3). According to Custom Search JSON/Atom API, by sending an HTTP GET request to its Uniform Resource Identifier (URI), you pass in the details of a search request as query parameters. The format for the JSON/Atom Custom Search API URI is listed at https://developers.google.com/custom-search/json-api/v1/using_rest.
Program 3. Implementation of Google Custom Search API

```sas
filename Resp "%sysfunc(pathname(work))/FF.json";
%macro injury;
%do i = 1 %to &num;
proc http
   url="https://www.googleapis.com/customsearch/v1?q=&qroster&i+questionable&cx=cx_id&key=api_key&orTerms=injury&dateRestrict=d1"
   method="GET" out=resp;
run;
libname FF json fileref=resp;
/* Cleaning up results */
data Searchs&i;
   set FF.PageMap_metatags;
   format name $32.;
   name="&player&i";
run;
%if i>1 %then %do ;
   proc append base=Searchs1 data=Searchs&i out=combined force;
%end;
%mend;
%injury;
```

The order of parameters does not affect code efficiency. Additional parameters, which will impact the results of the search query, can be found [here](#). Below is a list of parameters used in this program. Each parameter is separated by “&” in the code, and the definitions are provided from the Google Custom Search CSE: list.

- **q** = the search expression
  - Each term will be separated by “+”.
  - The macro variables roster1-roster&num each contain the first and last name of every player separated by “+”. This program searches a combination of a player’s name and “questionable.” For example, when “&roster&i” evaluates to roster3 the search query is Alvin+Kamara+questionable.
- **cx** = the custom search engine ID
- **key** = API key
  - Has a max of 100 free searches per day
- **orTerms** (optional) = provides additional search terms to check for in a document, where each document in the search results must contain at least one of the additional search terms.
  - This program requires the document to contain the word “injury.”
- **dateRestrict** (optional) = Restricts results to URLs based on date. You can specify by number of days, weeks, months, or years.
  - This code restricts results to one day (d1) to retrieve the most recent status of a player’s injury. The format for days is d[number].

The next step creates a library for the search results. The library is called FF and converts the JSON files into SAS data sets. Several data sets are created from the custom search, but PageMap_metatags is most relevant for our task. It returns information about websites that contain information from the search query. Searchs&i will be created from the PageMap_metatags data set for every player. To keep track of which results belong to a player, a variable called “player” has been created. Once all Searchs&i data sets are created, they will be combined into one data set (Searchs1) through an additional DO loop. Once complete, all search results will be in one table and can be deciphered by player name (see Output 2). Be aware that you will see an error similar to Output 3 if the search creates no results, which would populate
no data. Also, the Pagemap_metatags data set is not always consistent. You may see an error similar to Output 4 if a variable no longer exists in the data set.

### Output 2. Search Results Example

<table>
<thead>
<tr>
<th>Title</th>
<th>URL</th>
<th>Source</th>
<th>Tag</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dak Prescott</td>
<td><a href="http://example.com">http://example.com</a></td>
<td>article</td>
<td>80</td>
<td>0.95</td>
</tr>
<tr>
<td>Dak Prescott</td>
<td><a href="http://example.com">http://example.com</a></td>
<td>article</td>
<td>80</td>
<td>0.95</td>
</tr>
<tr>
<td>Dak Prescott</td>
<td><a href="http://example.com">http://example.com</a></td>
<td>article</td>
<td>80</td>
<td>0.95</td>
</tr>
<tr>
<td>Dak Prescott</td>
<td><a href="http://example.com">http://example.com</a></td>
<td>article</td>
<td>80</td>
<td>0.95</td>
</tr>
</tbody>
</table>

### Output 3. Potential Error for No Results

**ERROR:** File FF.PAGEMAP_METATAGS_DATA does not exist.

### Output 4. Error Due to Inconsistent Variable

**ERROR:** Variable article_modified_time is not on file WORK.SEARCHS1

---

**CLEAN RESULTS**

A lot of noise in the Searches1 table needs to be cleaned up. The search query looks for the specified terms anywhere on a web page. It is important to filter out search terms that matched on an image. Figure 7 provides an example of a website where the only mention of Dak Prescott is in a caption to an image for a separate article. This is a false positive result since the article returned in the search does not mention Dak. The highlighted section in Program 4 removes this result in the data.

---

**Figure 7. Image Result**

![Image Result](image-url)
Program 4. Data Cleaning

/* Creating a macro for today's date. */
%let todaysDate = %sysfunc(today(), ddmmyyd8.);

data combine;
  set Searches1;
  format combine_title $50.;
  if twitter_card ne "summary_large_image"; /* Gets rid of search results for images*/
  where article_modified_time like "/&todaysDate/%;
  call symputx('num',_n_);
  macro=tranwrd(trim(compbl(og_title))," ","+");
  combine_title=og_title;
run;

data _null_; length cat $140.;
do until (last.og_type);
  set combine;
  by og_type notsorted;
  cat=catx(', ',cat,name);
end;
call symputx("Final",cat);
run;

As stated previously, the best results regarding a player’s injury are the most recent. Therefore, the
program filters out any pages that are not updated the same day the code runs by using the macro
todaysDate. This DATA step also creates a macro variable for the number of observations from the
cleaned data set, which will be used later. The last step in this DATA step creates a variable from the
title of the website "og_title". Links are in the Pagemap_metatags data set. However, they do not appear to
work properly. Therefore, another custom search will need to be conducted on these titles to retrieve
appropriate links for these sites. Running a Google Custom Search on the result titles is one option for
getting useable links. The variable "macro" is substituting “+” for every space in the og_title.

Table Combine is now filtered to include only relevant search results and contains a variable containing
players’ names. The last DATA step creates one observation for all remaining players that will be saved
in the final macro. This macro contains a list of players whose status might need to be reevaluated. The
list of players as well as a list of relevant search links will be sent to you in a text message, which I will
discuss later.

RETRIEVING WEBSITE LINKS

In this occurrence, the Google Custom Search API will only search on the title while the other parameters
will remain the same. The Items data set will be used in this section of the program since it contains
functional website links. Be aware that you will see an error similar to Output 5 if the search creates no
results.

Output 5. Potential Error for No Results

ERROR: File FF.ITEMS.DATA does not exist.

Once data is filtered to only observations that contain the relevant title, a macro variable will be created of
all the relevant links. Now there are two macro variables. One contains applicable player names, and the
other contains website links. Program 5 only creates a macro variable for a maximum of one link. If the
link length is longer than 140 characters, the macro variable will be created from the title of the web page,
which you can use to search manually. Due to length restrictions of text messages, if links are longer than
140 characters, they will be split into two text messages, which renders them impractical.
Program 5. Link Implementation

/* Getting links */
proc sql;
select macro, combine_title
    into :text1-.text&num, :title1-.title&num
from combine;
quit;

%macro link;
%do i = 1 %to &num;
proc http
url="https://www.googleapis.com/customsearch/v1?q=&text&i&cx=cx_id&key=api_key&orTerms=injury&dateRestrict=d1"
method="GET" out=resp;
run;

libname FF json fileref=resp;
/* Cleaning up results */
data Link&i;
set FF.items;
format name $32.;
where title like "&title&&i%";
run;
%end;
%if i>1 %then %do ;
proc append base=Link1 data=Link&i force;
%end;
%mend

%link;
proc sql;
create table link_final as
select distinct link, title
from Link1;
quit;

data _null_;
set link_final;
    if length(link) > 140 then
        call symput("Link", title); else call symputx("Link", link);
run;

POST GAME

As any athlete knows, post-game recovery is also important to player performance. Now it is time to create a text message for your results. Program 6 was developed using guidance from *How to Send a Text Message with SAS*. The code will generate a text message containing a list of players whose status you might need to change, and a corresponding link or title of a relevant website for further investigation. Program 6 was developed in SAS 9.4 and this method of text notification is currently not supported in VIYA.
Program 6. Text Program

/* Sending Text Message */

options emailhost='mailhost.fyi.sas.com' emailsys=smtp;

filename msg email to="9198675309@txt.att.net"
FROM = "Update Lineup ASAP <email address>"
subject="Check the following;"
;

data _null_;  
file msg;  
put " &link" ;  
run;

data _null_;  
file msg;  
put " &Final" ;  
run;

Your final result should be similar to Figure 8. Once you develop your code, schedule it to run in time for you to decide on your lineup based on the text message you receive. For example, you can schedule it to run on Sundays at 12:00 p.m., which will give you enough time to further evaluate your roster before a 1 p.m. EST game time. If your text message fails to deliver, you will receive an email notifying you. If the macro variables link and final appear in your text message, that suggests that all your players are active and no search results were applicable.

Figure 8. Text Results

FRM:Update Lineup ASAP    
SUBJ:Check the following:  
MSG:Fantasy Football Injury Updates: Aaron Rodgers, Leonard Fournette

Figure 9. Website Results from Week 15

Kelvin Benjamin injury update
UPDATE: Benjamin is officially ACTIVE.

Benjamin (knee) practiced all week, and Bills coach Sean McDermott said he "feels good" about Benjamin’s chances of playing Sunday against the Dolphins.

The weather looks to be fairly clear in Buffalo on Sunday, but a matchup against Kelvin Howard figures to further slow down Benjamin. He’s a borderline WR2 because of his red zone potential.

CONCLUSION

As you know, SAS programming can be used for any business need, but we sometimes overlook its usefulness for personal needs. This paper has shown that the simple combination of macros, DO loops, and text notifications can assist with enjoyable activities such as fantasy football. These SAS programming techniques paired with APIs can be modified for any personal or professional project. Good luck in the upcoming season.
REFERENCES


ACKNOWLEDGMENTS
Special thanks to Jane Eslinger, Michael Drutar, Olga Harkola, and John McCall for guidance and encouragement.

RECOMMENDED READING
• CSE:list
• Using SAS to access Google Analytics APIs
• How to send a text message with SAS

CONTACT INFORMATION
Your comments and questions are valued and encouraged. Contact the author at:

Alexandria McCall
100 SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
Alexandria.McCall@sas.com
http://www.sas.com

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 trademarks of their respective companies.