SAS 1874-2018

The DO's of Fantasy Football

Alexandria McCall, SAS Institute Inc.

ABSTRACT

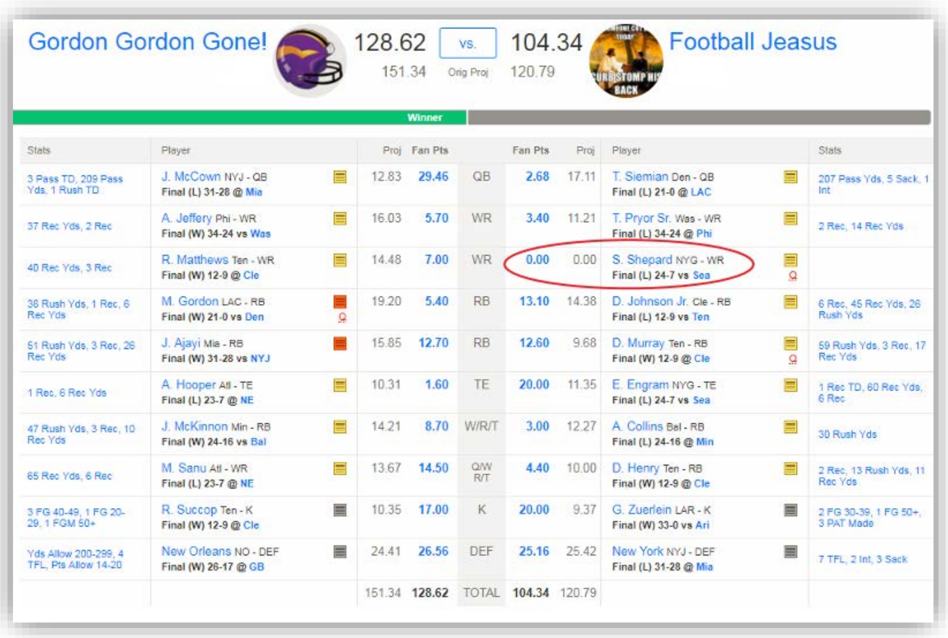
“Should I change my starting lineup?” This is the dreaded question all 40 million fantasy football participants have asked at one time or another. In particular, a player’s injury report can make or break a fantasy football participant’s lineup for the week. The unlikely combination of fantasy football 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.

Figure 1. Overlooked Lineup



Gordon Gordon Gone! 128.62 vs. 104.34 Football Jeasus

151.34 Orig Proj 120.79

Winner

Stats	Player	Proj	Fan Pts		Fan Pts	Proj	Player	Stats
3 Pass TD, 209 Pass Yds, 1 Rush TD	J. McCown NYJ - QB Final (L) 31-28 @ Mia	12.83	29.46	QB	2.68	17.11	T. Siemian Den - QB Final (L) 21-0 @ LAC	207 Pass Yds, 5 Sack, 1 Int
37 Rec Yds, 2 Rec	A. Jeffery Phi - WR Final (W) 34-24 vs Was	16.03	5.70	WR	3.40	11.21	T. Pryor Sr. Was - WR Final (L) 34-24 @ Phi	2 Rec, 14 Rec Yds
40 Rec Yds, 3 Rec	R. Matthews Ten - WR Final (W) 12-9 @ Cle	14.48	7.00	WR	0.00	0.00	S. Shepard NYG - WR Final (L) 24-7 vs Sea	
36 Rush Yds, 1 Rec, 6 Rec Yds	M. Gordon LAC - RB Final (W) 21-0 vs Den	19.20	5.40	RB	13.10	14.38	D. Johnson Jr. Cle - RB Final (L) 12-9 vs Ten	6 Rec, 45 Rec Yds, 26 Rush Yds
51 Rush Yds, 3 Rec, 26 Rec Yds	J. Ajayi Mia - RB Final (W) 31-28 vs NYJ	15.85	12.70	RB	12.60	9.68	D. Murray Ten - RB Final (W) 12-9 @ Cle	59 Rush Yds, 3 Rec, 17 Rec Yds
1 Rec, 6 Rec Yds	A. Hooper Atl - TE Final (L) 23-7 @ NE	10.31	1.60	TE	20.00	11.35	E. Engram NYG - TE Final (L) 24-7 vs Sea	1 Rec TD, 60 Rec Yds, 6 Rec
47 Rush Yds, 3 Rec, 10 Rec Yds	J. McKinnon Min - RB Final (W) 24-16 vs Bal	14.21	8.70	W/R/T	3.00	12.27	A. Collins Bal - RB Final (L) 24-16 @ Min	30 Rush Yds
65 Rec Yds, 6 Rec	M. Sanu Atl - WR Final (L) 23-7 @ NE	13.67	14.50	Q/W R/T	4.40	10.00	D. Henry Ten - RB Final (W) 12-9 @ Cle	2 Rec, 13 Rush Yds, 11 Rec Yds
3 FG 40-49, 1 FG 20-29, 1 FGM 50+	R. Succop Ten - K Final (W) 12-9 @ Cle	10.35	17.00	K	20.00	9.37	G. Zuerlein LAR - K Final (W) 33-0 vs Ari	2 FG 30-39, 1 FG 50+, 3 PAT Made
Yds Allow 200-299, 4 TFL, Pts Allow 14-20	New Orleans NO - DEF Final (W) 26-17 @ GB	24.41	26.56	DEF	25.16	25.42	New York NYJ - DEF Final (L) 31-28 @ Mia	7 TFL, 2 Int, 3 Sack
		151.34	128.62	TOTAL	104.34	120.79		

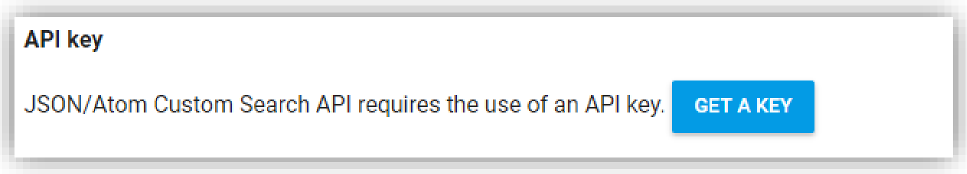
In addition to creating a clever fantasy football team name, removing inactive players from your roster is essential for team success. Notice in Figure 1 how Sterling Shepard was inactive and provided Football Jeasus with 0 points. If Football Jeasus had replaced Shepard with an active player, the fantasy 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.

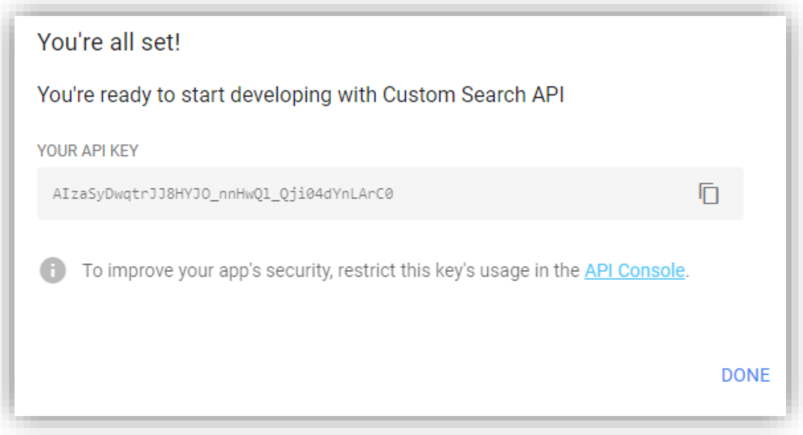
Figure 2. API Key Request



API key

JSON/Atom Custom Search API requires the use of an API key.

Figure 3. API Key Confirmation



You're all set!

You're ready to start developing with Custom Search API

YOUR API KEY

AIzaSyDwqtrJJ8HYJO_nnHwQ1_Qji04dYnLArC0

To improve your app's security, restrict this key's usage in the API Console.

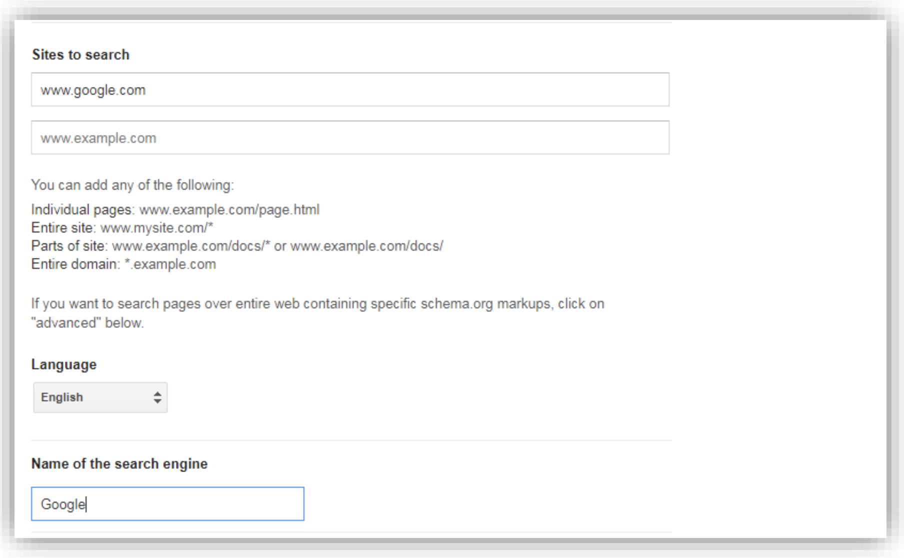
RETRIEVE CX ID

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



Sites to search

www.google.com

www.example.com

You can add any of the following:

Individual pages: www.example.com/page.html
Entire site: www.mysite.com/*
Parts of site: www.example.com/docs/* or www.example.com/docs/
Entire domain: *.example.com

If you want to search pages over entire web containing specific schema.org markups, click on "advanced" below.

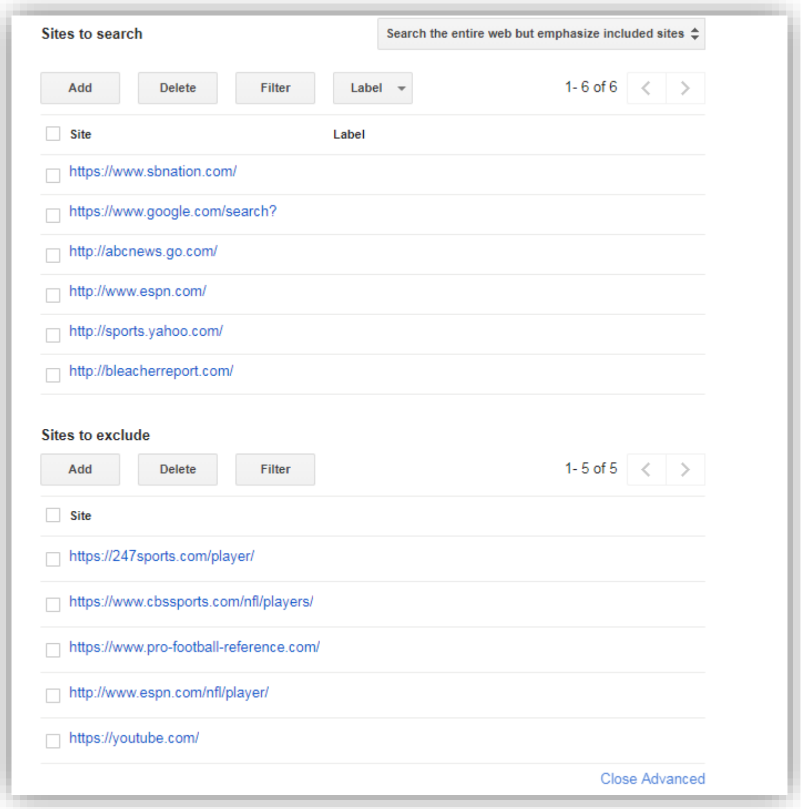
Language

English

Name of the search engine

Google

Figure 5. Specify Sites



Sites to search

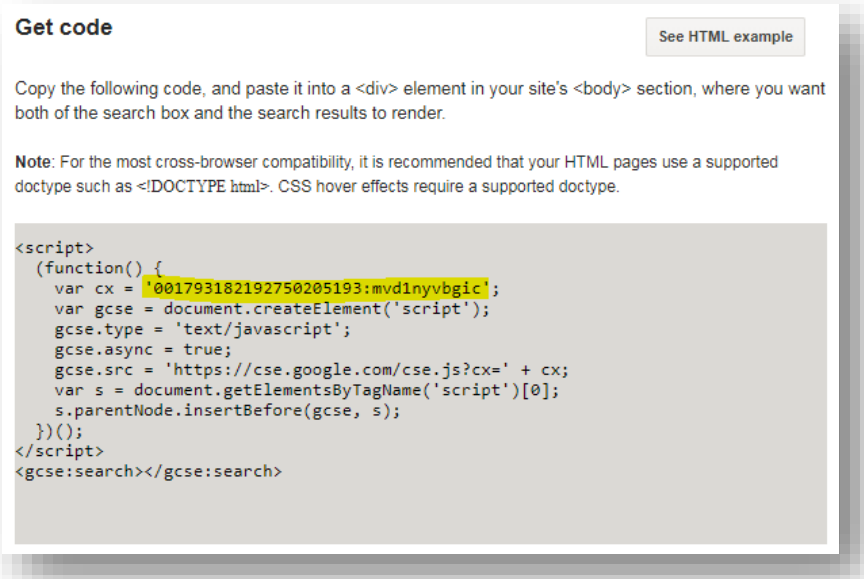
Search the entire web but emphasize included sites

□ Site	Label
□ https://www.sbnation.com/	
□ https://www.google.com/search?	
□ http://abcnews.go.com/	
□ http://www.espn.com/	
□ http://sports.yahoo.com/	
□ http://bleacherreport.com/	

Sites to exclude

□ Site
□ https://247sports.com/player/
□ https://www.cbssports.com/nfl/players/
□ https://www.pro-football-reference.com/
□ http://www.espn.com/nfl/player/
□ https://youtube.com/

Figure 6. Retrieve cx ID



Get code

Copy the following code, and paste it into a <div> element in your site's <body> section, where you want both of the search box and the search results to render.

Note: For the most cross-browser compatibility, it is recommended that your HTML pages use a supported doctype such as <!DOCTYPE html>. CSS hover effects require a supported doctype.

```
<script>
  (function() {
    var cx = '001793182192750205193:mvd1nyvbgic';
    var gcse = document.createElement('script');
    gcse.type = 'text/javascript';
    gcse.async = true;
    gcse.src = 'https://cse.google.com/cse.js?cx=' + cx;
    var s = document.getElementsByTagName('script')[0];
    s.parentNode.insertBefore(gcse, s);
  })();
</script>
<gcse:search></gcse:search>
```

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

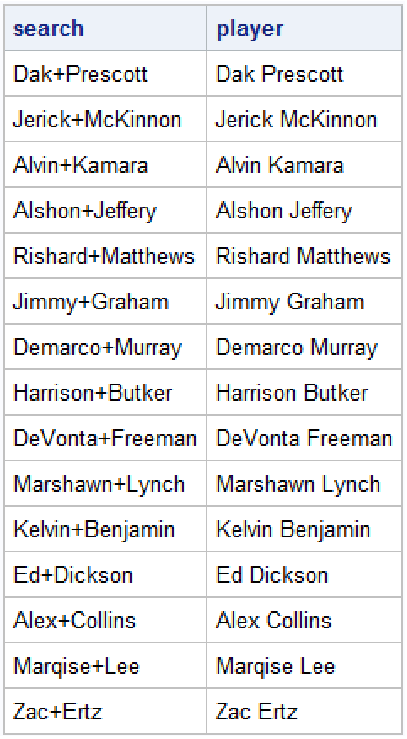
```
data Roster;
input First$ Last$;
cards;
Dak Prescott
Jerick McKinnon
Alvin Kamara
Alshon Jeffery
Rishard Matthews
Jimmy Graham
Demarco Murray
Harrison Butker
DeVonta Freeman
Marshawn Lynch
Kelvin Benjamin
Ed Dickson
Alex Collins
Marqise Lee
Zac Ertz
;
```

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

```
/* 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



search	player
Dak+Prescott	Dak Prescott
Jerick+McKinnon	Jerick McKinnon
Alvin+Kamara	Alvin Kamara
Alshon+Jeffery	Alshon Jeffery
Rishard+Matthews	Rishard Matthews
Jimmy+Graham	Jimmy Graham
Demarco+Murray	Demarco Murray
Harrison+Butker	Harrison Butker
DeVonta+Freeman	DeVonta Freeman
Marshawn+Lynch	Marshawn Lynch
Kelvin+Benjamin	Kelvin Benjamin
Ed+Dickson	Ed Dickson
Alex+Collins	Alex Collins
Marqise+Lee	Marqise Lee
Zac+Ertz	Zac Ertz

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

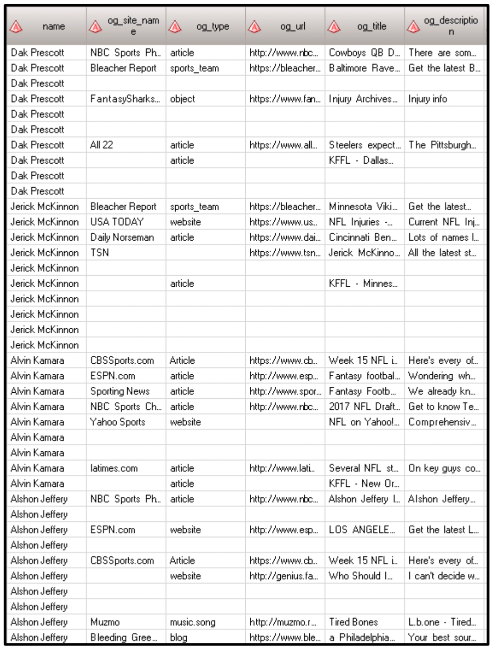
```
filename Resp "%sysfunc(pathname(work))/FF.json";
%macro injury;
%do i = 1 %to &num;
proc http
url="https://www.googleapis.com/customsearch/v1?q=&roster&&i+questionable&cx=cx_id&key=api_key&or
Terms=injury&dateRestrict=d1"
method="GET" out=resp;
run;

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;
%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-rosternum 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].

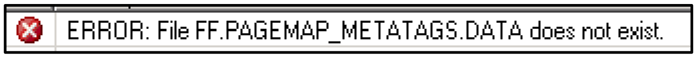
The next step creates a library for the search results. The library is called FF and converts the JSON files into SAS data sets. Several data sets are created from the custom search, but PageMap_metatags is most relevant for our task. It returns information about websites that contain information from 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



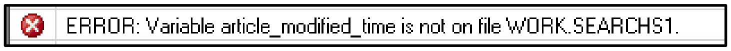
name	og_site_name	og_type	og_url	og_title	og_description
Dak Prescott	NBC Sports Ph...	article	http://www.nbc...	Cowboys QB D...	There are som...
Dak Prescott	Bleacher Report	sports_team	https://bleacher...	Baltimore Rave...	Get the latest B...
Dak Prescott					
Dak Prescott	FantasySharks...	object	https://www.fan...	Injury Archives...	Injury info
Dak Prescott					
Dak Prescott					
Dak Prescott	All 22	article	https://www.all...	Steelers expect...	The Pittsburgh...
Dak Prescott		article		KFFL - Dallas...	
Dak Prescott					
Dak Prescott					
Jerick McKinnon	Bleacher Report	sports_team	https://bleacher...	Minnesota Viki...	Get the latest...
Jerick McKinnon	USA TODAY	website	https://www.us...	NFL Injuries -...	Current NFL Inj...
Jerick McKinnon	Daily Norseman	article	https://www.dai...	Cincinnati Ben...	Lots of names l...
Jerick McKinnon	TSN		https://www.tsn...	Jerick McKinno...	All the latest st...
Jerick McKinnon					
Jerick McKinnon		article		KFFL - Minnes...	
Jerick McKinnon					
Jerick McKinnon					
Jerick McKinnon					
Jerick McKinnon					
Alvin Kamara	CBSSports.com	Article	https://www.cb...	Week 15 NFL i...	Here's every of...
Alvin Kamara	ESPN.com	article	http://www.esp...	Fantasy footbal...	Wondering wh...
Alvin Kamara	Sporting News	article	http://www.spor...	Fantasy Footb...	We already kn...
Alvin Kamara	NBC Sports Ch...	article	http://www.nbc...	2017 NFL Draft...	Get to know Te...
Alvin Kamara	Yahoo Sports	website		NFL on Yahoo!...	Comprehensiv...
Alvin Kamara					
Alvin Kamara					
Alvin Kamara	latimes.com	article	http://www.lati...	Several NFL st...	On key guys co...
Alvin Kamara		article		KFFL - New Or...	
Alshon Jeffery	NBC Sports Ph...	article	http://www.nbc...	Alshon Jeffery I...	Alshon Jeffery...
Alshon Jeffery					
Alshon Jeffery	ESPN.com	website	http://www.esp...	LOS ANGELE...	Get the latest L...
Alshon Jeffery					
Alshon Jeffery	CBSSports.com	Article	https://www.cb...	Week 15 NFL i...	Here's every of...
Alshon Jeffery		website	http://genius.fa...	Who Should I...	I can't decide w...
Alshon Jeffery					
Alshon Jeffery					
Alshon Jeffery	Muzmo	music.song	http://muzmo.r...	Tired Bones	L.b.one - Tired...
Alshon Jeffery	Bleeding Gree...	blog	https://www.ble...	a Philadelphia...	Your best sour...

Output 3. Potential Error for No Results



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

Output 4. Error Due to Inconsistent Variable

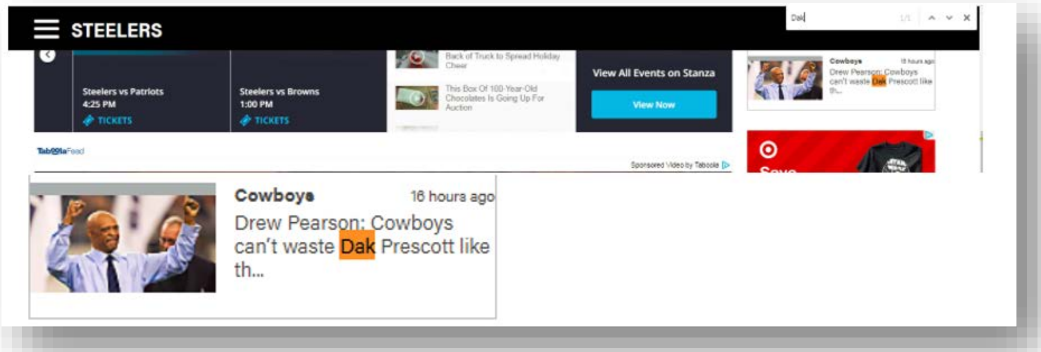


ERROR: Variable article_modified_time is not on file WORK.SEARCHS1.

CLEAN RESULTS

A lot of noise in the Searchs1 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



Program 4. Data Cleaning

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

data combine;
set Searchs1;
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;
```

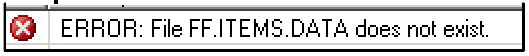
As stated previously, the best results regarding a player's injury are the most recent. Therefore, the program filters out any pages that are not updated the same day the code runs by using the macro todaysDate. This DATA step also creates a macro variable for the number of observations from the cleaned 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&da
teRestrict=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

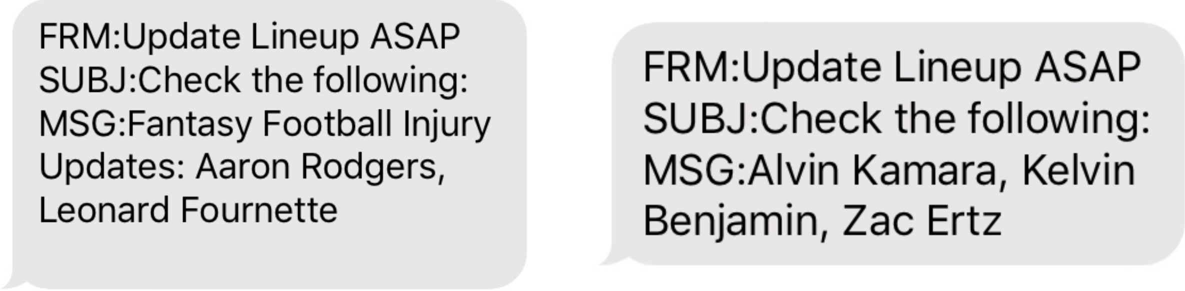
As any athlete knows, post-game recovery is also important to player performance. Now it is time to create a text message for your results. Program 6 was developed using guidance from *How to Send a Text Message with* SAS. The code will generate a text message containing a list of players whose status 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.

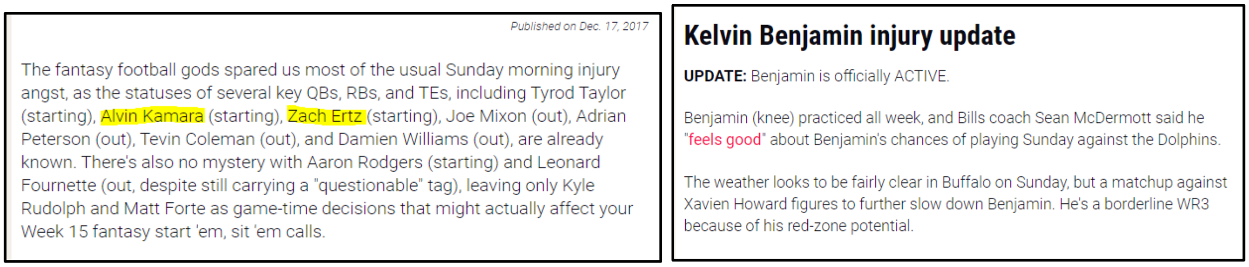
Figure 8. Text Results



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

FRM:Update Lineup ASAP
SUBJ:Check the following:
MSG:Alvin Kamara, Kelvin Benjamin, Zac Ertz

Figure 9. Website Results from Week 15



Published on Dec. 17, 2017

The fantasy football gods spared us most of the usual Sunday morning injury angst, as the statuses of several key QBs, RBs, and TEs, including Tyrod Taylor (starting), Alvin Kamara (starting), Zach Ertz (starting), Joe Mixon (out), Adrian Peterson (out), Tevin Coleman (out), and Damien Williams (out), are already known. There's also no mystery with Aaron Rodgers (starting) and Leonard Fournette (out, despite still carrying a "questionable" tag), leaving only Kyle Rudolph and Matt Forte as game-time decisions that might actually affect your Week 15 fantasy start 'em, sit 'em calls.

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 Xavien Howard figures to further slow down Benjamin. He's a borderline WR3 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

"CSE." Google Custom Search. Available at https://developers.google.com/custom-search/json-api/v1/reference/cse. Accessed January 11, 2018.

"Custom Search JSON/Atom API." Google Custom Search. Available at https://developers.google.com/custom-search/json-api/v1/overview. Accessed January 11, 2018.

"Getting a cx ID for custom search, Google API - Python." StackOverflow.. Available at https://stackoverflow.com/questions/6562125/getting-a-cx-id-for-custom-search-google-api-python. Accessed February 1, 2018.

Haikuo. "Concatenate multiple rows into a single value." SAS Communities. Available at https://communities.sas.com/t5/Base-SAS-Programming/Concatenate-multiple-rows-into-a-single-value/td-p/132882. Accessed January 11, 2018.

Hemedinger, Chris. "How to send a text message with SAS." SAS Blogs. https://blogs.sas.com/content/sasdummy/2017/04/14/using-sas-to-access-google-analytics-apis/.

Hemedinger, Chris. "Using SAS to access Google Analytics APIs." SAS Blogs. https://blogs.sas.com/content/sasdummy/2017/04/14/using-sas-to-access-google-analytics-apis/. Accessed January 11, 2018.

"XML API reference." Google Custom Search. Available at https://developers.google.com/custom-search/docs/xml_results. Accessed January 11, 2018.

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