

Paper 4668 -2020

Play More, Achieve More: Training the Next Generation of Data Analysts Through Strategic Gaming

Hillary Graham, Eli Lilly and Company

ABSTRACT

How did you learn to program? Most of us would answer this question by pointing to on-the-job experience. Documentation and instructional videos have ample value, but we all know that the best way to learn something is to jump in and do it. However, building a **comprehensive programming skill set can take years' worth of individual experiences**, and, with the ever-changing landscape of technical innovation, the expectation for analytical programmers to learn faster is increasing. Recent research indicates that gaming can be an effective way to learn new skills because games encourage an emotional investment that traditional training methods do not. This paper and presentation demonstrate a gaming solution based on SAS® for introducing a new data analyst to the most common tasks they will encounter on the job, including skills every analytical programmer should learn: simulating a data set by following a specification document, the basics of visualizing data, implementing common statistical modeling techniques, and using macros to streamline an analysis, all while being transported to the different 'worlds' of R and Spotfire along the way.

INTRODUCTION

Gamification, or the process of enhancing an activity with elements of gaming, has been an increasingly popular way to engage employees in on-the-job learning. This combination of education and entertainment has initially shown to be an effective way to both motivate personnel to acquire new skills and have them retain and apply such skills in their daily work (Nacke, 2015; Sailer and Homner, 2019). An effectively gamified process can keep the learner engaged and entertained simultaneously in a way that traditional training methods do not. There are countless approaches to turning training into a fun and engaging exercise. Commonly used gamification techniques include incorporating quests, levels, rewards/scoring, storylines, and/or risk elements to the learning process (Morschheuser 2017).

Today's newest data analysts are presented with a unique landscape when they enter the workforce. Programming has become partially automated or made more efficient with macros, meta-programs, and other time-saving tools in a way that traditional education **doesn't fully prepare for**. The expectation for these analysts is that they will become familiar with the available tools and then build on them further as they find more efficient ways to work. However, this means the initial learning curve is increasingly steeper while the trainings **for climbing it haven't necessarily improved. In order to bridge this gap and get** data analysts quickly up-to-speed, we propose a gamified training module with quest, storyline, and reward elements to introduce a new analyst to the most common tasks they may face on the job and the tools available to help master them.

TRAINING MODULE CREATION

The SAS® based training game developed for this presentation is structured in a way that the user must complete role-based tasks in order to collect a small snippet of a numerical code, which are then compiled at the conclusion of the training to solve a final puzzle. The

user assumes the role of a 'secret-agent' who is prompted by 'mission control' to uncover the location of a long-hidden piece of art. This narrative layout serves to engage the user with a descriptive scenario while each objective provided by mission control serves the ultimate purpose of training the user in specific tasks they will encounter in their day-to-day work. The complete list of objectives included in our game can be found below in table 1:

Training Objectives	
1. Familiarize the user with file structure used in typical workflow.	2. Teach the user how to compile a setup macro program to make downstream analyses more efficient.
3. Introduce the user to time-to-event analysis with PROC PHREG and PROC LIFETEST in SAS.	4. Have the user generate a common time-to-event analysis figure, the Kaplan-Meier plot. In addition, the user will learn how to run R code within the SAS interface and how to pass objects between the two software applications.
5. The user completes a data manipulation exercise within SAS using PROC TRANSPOSE and common DATA steps.	6. Introduce the user to the Spotfire software using R code within the interface to conduct analytical subgroup analysis in real time.

Table 1. Training Objectives

Each objective is introduced to the user with a PDF file with a narrative scenario outlining why the agent must complete the given task and instructions on how to progress to the next objective. For each of the tasks, the user is also supplied with a program script and/or dataset to get them started. In providing these resources, we try to strike a balance between the difficulty of the task and programmer autonomy. If the task is too simple or the provided script is too thorough, the user may not end up retaining the training objective. On the other hand, if the provided script is too sparse, the user may lose interest and motivation to finish the task. When a new procedure is introduced, we typically supply the link to the relevant SAS, R, or Spotfire documentation as an aid.

Once the user is confident in their solution to the given objective, they call a macro program stored in 'mission control' which will check several components of their completed program script. If all the tasks associated with the current objective are complete, they will be supplied with a new PDF file outlining the next objective in the mission along with a small reward (in the form of a numerical code) for completing the previous objective. In order to keep progressing in the mission, the user will need to document these reward codes in a tracking file so that they are readily available at the conclusion of the training module. **Following this process ensures that the user will simultaneously be trained in several 'meta-objectives'; objectives that are not associated with just one specific task, but are also important for the individual's on-the-job training:**

Meta - Objectives	
Introduce user to the typical workflow (raw data -> standardized data -> analysis).	Give the user a working understanding of which software is most appropriate for a given task.
Show the user how SAS, Spotfire, and R can be utilized together to make processes more efficient.	Familiarize the user with good documentation practices and tracking files.

Table 2. Training Meta-Objectives

The heart of this training module is the compiled macro program that checks the completeness of each objective. Using binary macro variables, we check that each portion of the given task has been completed and flip the macro variable from FALSE to TRUE if the **user's solution is viable. Once each component of the objective is complete, the macro executes to place the files pertaining to the next objective into the user's working directory.** A simplified example is shown in the code below:

```

* Check which operation the user is currently completing;
%if %sysfunc(fileexist(&missionloc/documentation/04_OP.pdf)) %then %do;

options mprint;
%global answer1 answer2;
%let obj1=F; %let obj2=F; %let obj3=F;

* Check if log file exists in user folders;
%if %sysfunc(fileexist(&log/operation04.log)) %then %let obj1=T;

* Check calculations are correct;
%if %str(&answer1)=%str(35) %then %let obj2=T;
%if %str(&answer2)=%str(235) %then %let obj3=T;

* Give next mission or mission failed document;
%if %str(&obj1. &obj2. &obj3.) = %str(T T T)%then %do;

    %if %sysfunc(fileexist(&missionloc/documentation/04_OP_Failed.pdf)) %then %do;
        %let rc=%sysfunc(filename(temp,&missionloc/documentation/04_OP_Failed.pdf));
        %let rc=%sysfunc(fdelete(&temp));
    %end;

* copy next operation files;
x cp      "/MissionControl/MissionPDFs/05_OP.pdf"
          "&missionloc./documentation/05_OP.pdf";

%end; %else %do;

    * If mission failed: copy hint document;
    x cp   "/MissionControl/MissionPDFs/04_OP_Failed.pdf"
          "&missionloc./documentation/04_OP_Failed.pdf";

%end;
%end;

```

Due to the unlimited applications of gamification, the following sections of this paper will describe two finite tasks rather than the entirety of the training game in order that the reader may use the ideas presented as inspiration rather than a step-by-step guide. In the following section we outline objective 4 from table 1, combining both SAS and R code using PROC IML to generate a Kaplan-Meier plot. The final section will focus on using R code within Spotfire to create an on-demand subgroup visualization tool that accepts SAS - generated datasets as input, described as objective 6 in table 1.

SAS + R: PROC IML

New and even experienced analysts may be familiar with both SAS and R programming, but they may not know how the two languages can work together to accomplish related tasks. The use of PROC IML within the SAS/IML software allows us to submit R code within the SAS interface. The advantage of utilizing SAS generally lies in its power and efficiency in running complex **analyses**, while R's **object-oriented** language provides a way to easily customize an analysis or figure. To leverage both capabilities, one application is to have SAS to run the analysis while submitting R code to output the corresponding figure. PROC IML allows data values, matrices, and vectors to be passed to R so we can accomplish both tasks (analysis and figure in our example) simultaneously in one program.

To get the user started we provide a SAS program script containing an outline of the statements required to complete the objective. This allows the user to seek help through SAS and R documentation in order to absorb the purpose of each statement as they complete the script. The user is also supplied with a simple time-to-event dataset as input to the given program, the snapshot below shows the first several rows of this dataset:

ID	METHOD	TTE	CNSR
5537478181	mthd1	15.51105	0
2415843310	mthd1	0.4307	0
4400679100	mthd2	5.0637	0
2664026167	mthd1	1.901	1
5945148088	mthd2	49.50765	0
2858643649	mthd2	48.1511	0
2037527888	mthd2	49.99985	0
8337381690	mthd2	0.8798	0
7995922736	mthd1	18.59075	0

Figure 1. Simulated Time-to-Event Dataset

A valid solution to this objective is included in the code section of the paper below. PROC IML uses a local connection to the R software to run the code supplied. In this case, we used SAS to run PROC PHREG and PROC LIFETEST and output numerical summaries to an RTF file, while R creates a corresponding Kaplan-Meier plot and annotates the numeric values calculated by PROC PHREG and PROC LIFETEST. A screenshot of the resulting figure can be found below:

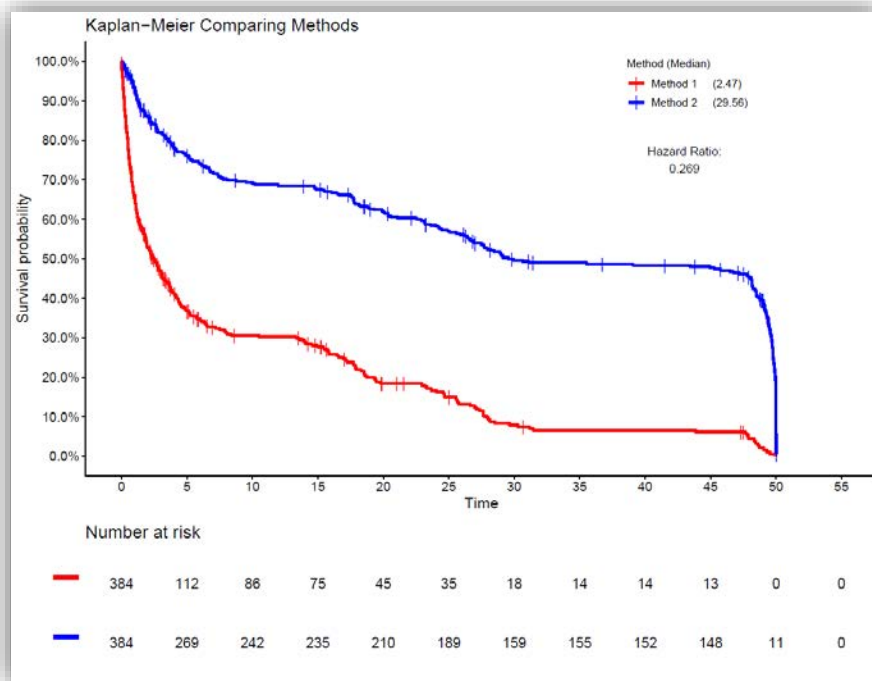


Figure 2. Kaplan-Meier Generated with R through PROC IML

Utilizing R within SAS has many advantages, although implementing this technique takes some initial setup. SAS users who would like to apply a similar method may find the troubleshooting guide below helpful:

Issue	Action	Discussion/Notes
Unknown if SAS/IML is licensed	PROC SETINIT; run;	Check the log to see if SAS/IML is a licensed part of your software package. If not, you will need to purchase the IML license.
Unknown if RLANG option is enabled	PROC OPTIONS option=RLANG; run;	Check the log to see if the option to enable R language within PROC IML is turned on.
RLANG option is disabled (log shows NORLANG in prior troubleshooting check)	Add -RLANG to the configuration file.	Locate the configuration file in your SASHome directory (often named sasv9.cfg). Once you add the -RLANG statement you should be able to access the capabilities described.
Can't find the configuration file and/or running on a SAS Grid environment	Contact the SAS administrator for your organization to petition for the RLANG option or install SAS locally to modify individually.	Without the RLANG option the user will be unable to run the final code. One of these two options must be completed.
Error 'No R installed' when running PROC IML	Make sure you have R installed locally on your device. Download R software if missing.	If R is installed in a non-standard location, SAS may have a hard time locating it. Edit the environment variables from the device control panel to direct SAS to the correct path.
R code doesn't execute as expected	Install all dependent packages in R directly with install.packages function.	If the R code submitted with PROC IML depends on one or more packages, you will need to ensure that all packages utilized have been downloaded before submitting the final code. Once all packages are installed, the code should execute as expected.

Table 3. Troubleshooting SAS+R PROC IML Capabilities

SPOTFIRE + R: TERR ENGINE

Spotfire is a very powerful data visualization tool. It allows the user to dynamically change data input through subgroup selection and toggle visualization types with a simple point-and-click interface. Because Spotfire is primarily a visualization tool, one of its biggest **limitations is that it doesn't support statistical analyses beyond a few basic tests. However,** through the TERR tools interface, a user can supply R code to a Spotfire template to supplement the desired visualizations.

For the training module we supply the user with a Spotfire template, an R script, a SAS dataset that will need to be merged with the data in the previous section, and a step-by-step guide on how to input both the data and R script into the template. A snapshot of the data input can be found below in figure 3 and the full accompanying R script can be found in the code section of the paper.





	 ID	 cov1	 cov2	 cov3
1	5537478181	Y	Level 1	0.5058753711
2	2415843310	N	Level 2	0.6592424398
3	4400679100	N	Level 1	0.4984222044
4	2664026167	N	Level 3	-0.741087269
5	5945148088	N	Level 2	-0.013199954
6	2858643649	Y	Level 2	0.0018911813
7	2037527888	N	Level 1	-0.779467582
8	8337381690	Y	Level 2	-0.248504574

Figure 3. Dataset Containing Covariates

In order to leverage the capabilities of R within Spotfire, the user must follow these steps:

1. Open the Spotfire template.
2. Load the provided data by selecting File -> Replace Data Table and importing the main dataset.
3. In the menu bar select Edit -> Data Function Properties -> Register New.
4. Copy and paste the provided R script into the "Script" text box.
5. Select a name for the data function in the "Name" text box and place relevant R packages in the "Packages" text box. (survival is the only package used in our example)
6. In the "Input Parameters" section, add any data columns needed by the R script as input. For our example we initialize the time, censor, and treatment variables as columns. While inputting the necessary columns make sure to select the "Active Filtering Scheme" option for limiting the data to enable us to refresh the analysis every time we pick a new subgroup.
7. In the "Output Parameters" section, add the variables or dataframes we want to pass back to Spotfire. In our example we provide three dataframes containing median tte, hazard ratio, and log-rank p-value respectively.
8. Click the "New Function" button in the main menu bar to embed the R script into the template. Close the Data Functions window.
9. In the main menu bar click Tools -> TERR Tools -> Package Management. Select a CRAN repository and click "Load".
10. Type packages needed (survival in this case) in the "Available Packages" text box and then "Install". You can then close the window.

The result of this exercise is a dynamic template to visualize a Kaplan-Meier for covariates contained in the dataset. Each time the user selects a new value the entire template will update including the corresponding median survival times, hazard ratio, and p-value. The user can then add additional supporting graphics and/or data values that can be used to discover further insights hidden in the data. A snapshot of the final template created for this training module can be found below in figure 4.



Figure 4. Spotfire Template Executing R Code

Utilizing R code in the background of a Spotfire template opens numerous possibilities for combining the on-demand power of Spotfire and the analytical strength of R. However, **there some helpful best practices that we've encountered as we developed this template:**

- Keep the R script as simple as possible. Spotfire only supports certain R packages and often throws errors that can be hard to decipher. Using base R commands when possible resolves many of these issues.
- Make sure filtering dependencies are thoroughly planned before beginning. Do you want your R-generated values to update every time you select values in another panel? Do you want the other panels to update based on the values supplied by R? These relationships should be explicit to avoid confusion.
- Make sure to specify how you want your R script to execute, locally or over a server connection. This can be done in the Edit -> Data Function Properties -> Edit Parameters window. The "Run Location" drop down menu can then be changed from local to server or vice versa.

CONCLUSION

In the two examples outlined in this paper, we guide the learner to complete the specific tasks of using SAS to execute a time-to-event analysis, run R within SAS to create the corresponding KM plot, and then uses Spotfire in conjunction with R to create a re-usable tool for subgroup analyses. These are all important tasks of our analysts to master, but the greater lesson we hope **to instill is knowledge of each of the three software and when it's** appropriate to combine their strengths. To this point, figure 5 below shows a diagram of SAS, R, and Spotfire and how they interact:

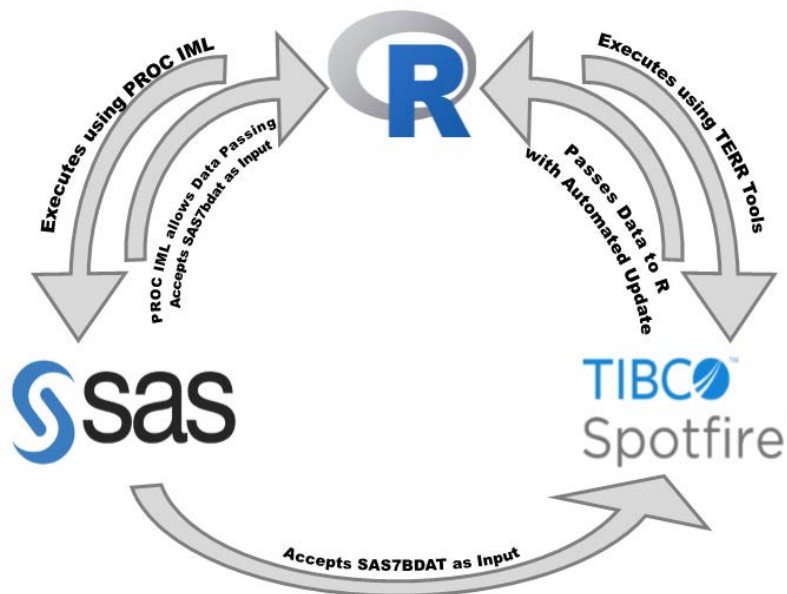


Figure 5. Software Interaction Possibilities

Figure 5 shows relationships between SAS, Spotfire, and R code and objects in addition to the strengths of each application. We see that R code can be used within both SAS and Spotfire, although in SAS we may be more inclined to use R for visualizations while in Spotfire, R is better for executing statistical analyses. SAS is most efficient for complex

analysis and data manipulation while Spotfire's strength is on-demand visualizations for data exploration. Both Spotfire and R will accept SAS generated data and objects as input and additionally, R will accept SAS data, vectors, and variables through PROC IML.

We find that these intricate relationships can be mastered through this gamification of learning approach. As the user completes the simple objectives for the invented scenarios, they acquire the necessary background knowledge to apply it to business needs.

REFERENCES

Sailer, M., Homner, L. The Gamification of Learning: a Meta-analysis. *Educ Psychol Rev* (2019). <https://doi.org/10.1007/s10648-019-09498-w>.

Nacke, Lennart. "The Use of Games and Play to Achieve Real-World Goals." *Gamification Research Network*, 24 June 2015, gamification-research.org/2015/06/the-use-of-games-and-play-to-achieve-real-world-goals/.

Morschheuser, Benedikt & Werder, Karl & Hamari, Juho & Abe, Julian. (2017). How to Gamify? A Method For Designing Gamification. 10.24251/HICSS.2017.155.

CODE

SAS + R: PROC IML Example Code:

```
PROC IMPORT datafile="&opdata\op_data_04.csv" out=dat04 dbms=csv replace;
  getnames=yes;
  guessingrows=4000;
  datarow=2;
run;
```

```
PROC LIFETEST data=dat04;
  time tte*cnsr(1) ;
  strata method;
  ods output Quartiles=quart;
run;
```

```
PROC PHREG data=dat04;
  class method(ref='mthd1');
  model tte * cnsr(1) = method / ties=exact;
  ods output ParameterEstimates=param;
run;
```

```
DATA dat1;
  length param $200;
  set quart;
  where Percent = 50;
  param = catx(' ', 'Median time-to-event for ', mthd);
  val = round(Estimate, .01);
  if mthd = 'mthd1' then call symputx("med_1", val, G);
  if mthd = 'mthd2' then call symputx("med_2", val, G);
  keep param val;
run;
```

```
DATA dat2;
  length param $200;
  set param;
  param = 'Hazard Ratio';
  val = round(HazardRatio, .001);
  call symputx("hr1", val, G);
```



```

fit1 <- survfit(Surv(time,status)~1,data=dat1,subset=(trt==1),conf.type="log-log")
fit0<-survfit(Surv(time,status)~1,data=dat1,subset=(trt==0),conf.type="log-log")
fit<-survfit(Surv(time,status)~trt)
t1=data.frame(probability=c(1,summary(fit1)$surv),month=c(0,summary(fit1)$time), uci =
c(1,summary(fit1)$upper), lci = c(1,summary(fit1)$lower), arm=arm1)
t0=data.frame(probability=c(1,summary(fit0)$surv),month=c(0,summary(fit0)$time), uci =
c(1,summary(fit0)$upper), lci = c(1,summary(fit0)$lower), arm=arm2)
t2=rbind(t1, t0)

ca1=aggregate(t2$month, list(t2$arm), tail, 1)
colnames(ca1) <- c("arm", "month")
ca2=aggregate(t2$probability, list(t2$arm), tail, 1)
colnames(ca2) <- c("arm", "probability")
ca3=merge(ca1,ca2)
tail(dat1, 1)
dat2 <- dat1[order(-trt, time),]
dat3=aggregate(dat2$time, list(dat2$trt), tail, 1)
colnames(dat3) <- c("trt", "time")
dat4=merge(dat3,dat2)
dat4$arm <- ifelse(dat4$trt==1,arm1,arm2)
dat5=merge(dat4,ca3)
dat6<-subset(dat5, status==0,select=c(-trt,-month,-status))
colnames(dat6)[2] <- "month"
dat6$uci = NA
dat6$lci = NA

t3=rbind(t2,dat6)
t4<-t3[order(t3$arm, -t3$probability,t3$month),]

# output median
fit_check<-survfit(Surv(time,status)~trt, data=dat1, conf.type="log-log")
fc=summary(fit_check)$table
fc0=data.frame(arm=arm2,records=fc[,"records"][1],events=fc[,"events"][1],median=fc[,"median"]
[1],lower_95=fc[,"0.95LCL"][1],upper_95=fc[,"0.95UCL"][1])
fc1=data.frame(arm=arm1,records=fc[,"records"][2],events=fc[,"events"][2],median=fc[,"median"]
[2],lower_95=fc[,"0.95LCL"][2],upper_95=fc[,"0.95UCL"][2])
fc2=rbind(fc1,fc0)

#log-rank p-value
fmtP3 <- function(x){
  if(x<0.001){ '<0.001'

```

```

} else{
  x1=ifelse(as.integer(x*10000) %% 10==5, ceiling(x*1000)/1000 ,round(x,3))
  sprintf("%.3f", round(x1 * 1000.)/1000.)
}
}
rho<-0
p1<-pchisq(survdiff(Surv(time,status)~trt , rho=rho)$chisq, df=1,lower.tail=FALSE)

# output hazard ratio
fit2=coxph
(Surv(time,status)~trt, method = 'exact')
hr1=summary(fit2, method = 'exact')$conf.int
hr2=data.frame(hazard_ratio=hr1[1],lower_95=hr1[3],upper_95=hr1[4],log_rank_P=fmtP3(p1))

```

CONTACT INFORMATION

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

Hillary Graham
 Eli Lilly and Company
graham_hillary_t@lilly.com