SOCIETY OF ACTUARIES
PREDICTIVE ANALYTICS AND FUTURISM SECTION

Predictive Analytics and Futurism

ISSUE 17 • APRIL 2018

Parallel Cloud Computing: Making Massive Actuarial Risk Analysis Possible

By Joe Long and Dan McCurley

Page 6

Predictive Analytics and Futurism

Issue Number 17 • April 2018

Published three times a year by the Predictive Analytics and Futurism Section of the Society of Actuaries.

475 N. Martingale Road, Suite 600
Schaumburg, Ill 60173-2226
Phone: 847.706.3500 Fax: 847.706.3599
www.soa.org

This newsletter is free to section members. Current issues are available on the SOA website (www.soa.org).

To join the section, SOA members and non-members can locate a membership form on the Predictive Analytics and Futurism Section webpage at http://www.soa.org/predictive-analytics-and-futurism/.

Publication Schedule
Publication Month: August, 2018
Articles Due: May 30, 2018

2018 SECTION LEADERSHIP

Better Tools—Less Dukkha

By Dave Snell

Many of us from the Western world are not familiar with the word "dukkha." Like many words of foreign origin, it does not have a one-word English translation. According to Wikipedia, it is the first of the Four Noble Truths of Buddhism. It is also found in scriptures of Hinduism, and it refers to "the fundamental unsatisfactoriness and painfulness of mundane life."[1]

What does this have to do with predictive analytics and futurism, and the associated techniques we embrace in this section? Perhaps this new era of artificial intelligence (AI) and machine learning will help liberate many of us from the dukkha of our current routines. Many of us spend boring hours commuting to and from work—sometimes in a paradoxical situation where as drivers we must patiently wait in long lines of traffic, yet we must be constantly vigilant to avoid accidents—often induced by the boredom of the waits. Autonomous cars may not only increase our safety, but also permit us to luxuriate in creative thought, having delegated the tedium of traffic mindfulness to our vehicles. We spend far too much time at work (and at home) on repetitive tasks that become mind-numbing rather than mind-expanding.

Some might argue that autonomous AI should never be trusted for life-critical decisions. I, for one, am ready and willing to delegate many of the processes and decisions of the day to AI, just as I delegate the life-critical tasks of breathing and digestion to my autonomous nervous system. In fact, I cannot imagine how tedious and stressful it would be to have to remember to breathe in and breathe out thousands of times per day; or to consciously have to tell my heart when to contract the left ventricle and send essential oxygenated blood to each of my cells.

Throughout human history, we have developed tools to do the "heavy lifting" for us: from shovels through backhoes. On the data assimilation, number-crunching and presentation side, we also are improving our tools, and this issue has several articles about new tools and techniques that can help you reduce your dukkha:

- Starting with Anders Larson in his "Chairperson's Corner," we are reminded of the importance of upgrading your tool set. Sure, there are sometimes temptations to treat every problem as a nail for your new hammer; but as Anders says, "just because everything isn't a nail, that doesn't mean there aren't nails out there that you've been hitting with a spoon." He talks about tools such as random forests that improved his analysis of the impact of multidimensional factors on health costs; and he even describes an upgrade to some of his Excel workbooks by using a function that computes vector products on a conditional basis.

- Next, "Parallel Cloud Computing: Making Massive Actuarial Risk Analysis Possible," by Joe Long and Dan McCurley, walks us through a cloud use case where they were able to cut a three-month machine learning exploration project down to just under four days using a mixture of open source tools and a cloud environment. That freed up a lot of time for them to digest the results, and run variations that would not have been feasible with a single processor approach to the project. Yes, they had to spend some time on the learning curve for parallelization; but it resulted in much faster throughput. Supposedly, Abraham Lincoln said, "Give me six hours to chop down a tree and I will spend the first four sharpening the axe." Lincoln would surely have viewed a 25-fold efficiency return as a wise investment.

- Moving on in our description of new tools, Michael Niemerg tells us about a novel technique. "The Forgery Game:

Generative Adversarial Networks" describes a generative adversarial network, or GAN. This is a very recent technique in artificial intelligence algorithms—introduced in 2014. A GAN is an unsupervised machine learning technique and can accomplish some interesting, and perhaps disturbing, outputs. Basically, models compete with each other and generate synthetic data. In one type of application, the result can be indistinguishable from a real photographic image (thus, the forgery game). This is leading-edge stuff; and as I am writing this issue introduction (Chinese New Year—新年快乐), I see an article about GANs in use to analyze molecular genetic mechanisms to create new synthetic drugs.[2]

- Not every tool has to be new, of course. Some are older ones that just have been underutilized. Ben Wolzenski led our "Blue Ocean" Delphi study back in 2009; and it predicted some nearly heretical ideas back then, such as pet insurance and custom-designed coverage developed online. Now, they have become important products with rapid growth. In "Why Consider a Delphi Study?" Ben describes advantages of this largely qualitative rather than quantitative approach to forecasting. It can provide value when other methods cannot, and can also serve as a second opinion for the other methods. He also details how our section has provided leadership in this technique in previous studies, and mentions another SOA Delphi study being launched now.

- I wrote the article "Hierarchical Clustering—A Recommendation From a Nonhierarchical Manager," where I describe a bottom-up, or agglomerative, technique that is more visually appealing to nonmathematicians than the more common k-means approach to clustering. Sometimes we overlook the fact that most senior managers are not actuaries or data scientists; and a tree-like visual that shows both the natural groupings you have discerned and the relative dissimilarity among the various groups, for even a multidimensional set of groupings, might be easier to understand, and thus more likely to be accepted.

- As the amount and types of data continue to increase, the complexity of models can be a limiting factor in their utility. Jeff Heaton, in his article, "Feature Importance in Supervised Training," addresses the issue of choosing which factors are the more important ones. Jeff takes us through model-specific feature ranking, model-agnostic feature ranking, and multivariate feature ranking. Removing unimportant features can increase both the speed and the accuracy of your models. This is especially important when you are employing feature engineering, which can benefit from feature importance evaluation to reduce the number of combinations involved in pair-wise multivariate considerations.

- It is nice that we have these new modeling tools available to us; but how do we share them with the folks who do not have, or even want to have, RStudio or a predictive analytics toolbox on their PCs? What if they want to get insights from your modeling efforts; but they do not wish to have to write R code to do that? In "Shiny: Another Step Forward in Data Democratization," Eileen Burns introduces us to a tool that addresses that concern. It's called Shiny, and the name is apt because it allows you to create an attractive and intuitive web application where non-programmers can try out your model and be creative and productive with it. Shiny can help you share your R apps with a larger base. The example she describes for us is a project where she put a web front end on the new *PAF Newsletter* Catalogue.

Eileen's article is also my segue to a new feature you should all be enjoying now—the newsletter index of all 195 articles from our section newsletters. These go back to September 2009, when the old Futurism Section became the Forecasting & Futurism Section (later Predictive Analytics and Futurism . . . as a result of a Delphi study). We know that most actuaries love Excel, and especially like to filter and sort and do lots of other data manipulations with it. On the newsletter webpage, you can now download your own copy of an Excel workbook with several columns for each article. If you want an actuarial perspective on agent-based modeling, neuroevolution of augmenting topologies (NEAT), hidden Markov models, genetic algorithms, or dozens of other topics, you can search, sort and filter it as much as you wish. Best of all, when you find the article you want to see, you can click on the hyperlink and go right to that issue! Thanks to Nick Hanewinckel, the PAF Section Council, section specialist Jessica Boyke and staff partner Beth Bernardi, we all have a handy new research tool.

Perhaps we can't completely escape dukkha; but the tools and techniques described in this issue ought to make it less mundane and less painful for you.

Enjoy! ■

Dave Snell, ASA, ACS, ARA, ChFC, CLU, FALU, FLMI, MAAA, MCP, teaches AI Machine Learning at Maryville University in St. Louis. He can be reached at *dave@ActuariesAndTechnology.com*.

ENDNOTES

1 Quoted from *https://en.wikipedia.org/wiki/Dukkha* (accessed Feb. 16, 2018).

2 *http://www.mauldineconomics.com/tech/tech-digest/right-to-try-our-best-shot-at-saving-healthcare*

Chairperson's Corner

By Anders Larson

I know that some of my fellow council members (and at least one former council member) will cringe when they see that I'm leading off the Chairperson's Corner with an anecdote about Microsoft Excel. But stick with me here. About five or six years ago, I realized that there was a formula that allowed you to do a conditional sum-product of two vectors. I was aware of the =sumproduct and =sumifs, but until then, I was unaware that the =sumproduct could be modified to add conditions. Mind = blown.

So what happened after that? I started noticing instances left and right where old workbooks could be improved with this "new" formula. A few years before that, I had a similar experience upon realizing the superiority of index-match functions to vlookups. I started cleaning up existing workbooks, but more importantly, I started thinking differently about setting up new workbooks. Of course, I didn't invent the conditional sum-product or the index-match. I just finally realized they existed, and all of a sudden I became a little bit better at my job.

I believe that actuaries can look at predictive analytics in much the same way. There are algorithms and techniques out there just waiting to be implemented into your existing work. Now, I realize that it's significantly more difficult to get comfortable with a support vector machine than a simple Excel formula, but the concept is the same. Once you start to see how a new approach can fit into one problem, it becomes that much easier to see how it can fit into countless others.

The obvious danger is that it is easy to start seeing everything as a nail once you have a cool new hammer to play with. In general, if a simpler model is just as effective as a more advanced approach, it's best to stick with the simpler approach. One of the key drawbacks I find with many machine learning algorithms is a lack of interpretability, particularly for those who don't work with them on a regular basis. In some cases, that's fine—I don't really care *how* my Amazon Alexa is able to understand speech, but a regulator may not be as willing to accept your estimates if they seem like they came from a black box.

But just because everything isn't a nail, that doesn't mean there aren't nails out there that you've been hitting with a spoon. Sure, the spoon will eventually drive the nail in there, but there's a better tool out there. In our July 2017 newsletter,[1] I wrote about a situation where we used a gradient boosting machine to predict primary care office visit utilization for individual patients. In the past, we might have attempted to predict primary care office visits using an existing risk score algorithm meant to predict health care costs. And while the existing risk score algorithm may have been useful, it was not really the best tool for this job. For instance, the sickest patients in a commercial population can have risk scores that are more than 100 times the population average, but very few patients will have even 10 times as many primary care visits as the population average.

Instead of thinking of each new algorithm as an all-purpose hammer, think of them as new tools to be added to your existing toolbox. Actuaries already have a wide array of traditional tools at their disposal, and those will continue to play an integral role in the future of actuarial science. But we can also improve our profession by incorporating new approaches into our work.

Here's another example from my own experience. I recently co-authored a paper[2] in which we identified the key drivers of gross savings for accountable care organizations (ACOs) participating in the Medicare Shared Savings Program (MSSP). We had more than 180 features about each ACO, many of which were highly correlated with each other. A few years ago, I likely would have approached this problem by limiting the data to a handful of reasonably independent features that I expected would be key drivers, and then running a simple linear regression. This would have still made for an interesting paper, but it likely would have been loaded with caveats that would have softened our conclusions. Instead, we used a random forest to estimate the relative importance of all 180+ features in predicting gross savings. This method allowed us to evaluate all the features together and let the machine identify which were most predictive. There were still caveats, of course—there is no silver bullet for a complex problem like this—but we felt the more rigorous statistical approach added credibility to our findings.

These predictive analytics tools are already out there. They've already been designed, built and tested for us. As actuaries, we just have to pay the small price of learning how to use them (and maybe some Amazon Web Services fees), and we can have them in our own toolbox. ■

Anders Larson, FSA, MAAA, is an actuary at Milliman in Indianapolis. He can be reached at *anders.larson@milliman.com*.

ENDNOTES

1 *https://www.soa.org/Library/Newsletters/Predictive-Analytics-and-Futurism/2017/june/2017-predictive-analytics-newsletter-issue-15.pdf*

2 *http://www.milliman.com/insight/2017/What-predictive-analytics-can-tell-us-about-key-drivers-of-MSSP-results/*

Parallel Cloud Computing: Making Massive Actuarial Risk Analysis Possible

By Joe Long and Dan McCurley

This article will walk through a cloud use case where we were able to cut a three-month machine learning exploration project[1] down to just under four days using a mixture of open source tools and the Microsoft Azure cloud. This translates to an approximate 25-fold reduction in serial compute time for such a task. We will give a short introduction to the cloud while sharing our experience of managing the pool of data-crunching machines that ran our analysis. In closing, we will discuss lessons learned and ways to improve the plan of attack, as well as touch on the importance of state management to aid in efficiency and the reproducibility of results when using the cloud.

SETTING THE STAGE FOR THE CLOUD

Machine learning is spreading quickly across many industries and is showing promising results for making better predictions and automating manual tasks. However, with increases in data size and the greater power of more complex algorithms, the computing resources it takes to crunch the numbers increase as well. Nowadays, it may take days or months to conduct an analysis on a single machine. There is a solution, though: Thanks to advances in cloud computing, the phrase "the sky's the limit" has a whole new meaning as we now have the ability to speed up time if the reward outweighs the cost of doing so.

In order to utilize the time-saving efficiencies of the cloud, a large computational process must be able to be broken down into independent tasks that can be run in parallel. Not every process fits this mold. Some processes rely on a series of sequential calculations, where each calculation is dependent on the ones that precede it. An example of such a process would be calculating a single sequence of time-dependent events, which would not be a good use case for the parallel compute capabilities of the cloud.

Machine learning, however, is full of many processes that can be broken down into independent tasks calculated in parallel, which can then be merged together after all independent calculations have been completed. A good example of this would be an ensemble method such as the random forest algorithm, which is used to develop a predictive model comprised of hundreds to thousands of independent decision trees that are averaged together to produce a single prediction. Another easily parallelizable example is the Monte Carlo simulation. These algorithms are prime candidates for the massive parallel computing abilities of the cloud. Almost all supervised learning algorithms use some kind of resampling technique (e.g., bootstrapping, cross-validation) to optimize the bias-variance trade-off for generalization. Most resampling techniques are embarrassingly parallel and can benefit greatly from cloud computing.

In our case, we used the cloud to help with a large machine learning exploration project, which was comprised of many calculations done in open source R. Our initial exploration started with a single heavy-duty, bare-metal machine that could handle traditional memory and compute intensive tasks. We quickly discovered that in order to run the full exploration analysis we mapped out, we would miss our deadline. Our initial estimate was that the full analysis—when run sequentially on our in-house machine—was expected to take 90 days of continuous computer run time. However, with some manual effort to break the analysis into semi-equal chunks, we estimated we could run it in Microsoft's Azure cloud and complete all of our calculations in less than a week. This approximately 25-fold reduction in serial compute time to run our analysis gave us more time to digest the results, giving us the ability to run further variants of our initial exploration plan. More variants can equal better value to the client.

THE MAGIC BEHIND THE CLOUD

"There is no cloud—it's just someone else's computer" is a common meme used to explain cloud services. While this phrase helps one understand the basic idea of the cloud, it does not fully recognize the great capabilities and flexibilities of the modern cloud infrastructure. The concept of the cloud dates back to the 1960s and is commonly attributed to J.C.R. Licklider and John McCarthy.[2] Joseph Licklider is credited for his core concept of a Galactic Network or "Network of Networks" and John McCarthy for theorizing utility computing. These ideas reached commercial viability in 2002 when Amazon Web Services (AWS) started providing web-based, pay-as-you-go services to companies to store data and run applications. Current major competitors to AWS include Microsoft Azure and Google Cloud.

All of these providers offer similar ways to access their resources. It is helpful to think of these resources in three main categories:

1. *Infrastructure as a service (IaaS)* creates a virtual data center in the cloud similar to what your company would have in an information technology (IT) climate-controlled room. It's easy to adopt but expensive to run.

2. The second way to access cloud resources is through *platform as a service (PaaS)*. In this method, the cloud provider takes care of storage and computation and provides a platform to do a focused type of work. If you want a database that is always available, but don't want to deal with any maintenance or tuning, this is an excellent solution.

3. Thirdly, *software as a service (SaaS)* allows a company to build a custom solution that can only exist in a cloud environment. Salesforce, Office 365 and G Suite are examples of SaaS.

Viewed in this context, our computing project was an example of an IaaS. But by the end of our exploration we had migrated much closer to a PaaS solution. The actual difference can get quite fuzzy.

THE LEARNING CURVE

Once we realized on-premise calculations would take too long, we turned to the task of determining how many (and what capacity) computers would be needed for a cloud solution. After a period of research on best approaches for parallelizing our process in the cloud, we estimated that 63 virtual machines (VMs) should be able to handle the work in a reasonable time frame. Each machine had eight cores and 56 gigabytes of RAM, giving us a total of over 500 cores and 3,500 gigabytes of RAM at our disposal. For this project, we chose to provision the machines with Windows as the operating system due to familiarity, but we note this costs about 50 percent more in license fees than an equivalent Linux VM. We wrote PowerShell scripts to automate cloning and administration of the machines. Later in this article we will describe a new tool that makes things much easier (and transitions this solution from pure IaaS to something closer to PaaS). At the time of our project, this setup had a sticker price of less than $2 per hour to run each virtual machine of this size in Azure.

Our first step was creating the initial VM and then installing R and all the R packages we would need to run our analysis. Once we had our initial VM configured, we created 62 clones of it using the Invoke-Parallel PowerShell script Warren Frame discussed in his "Invoke PowerShell on Azure VMs" article,[3] which had some other helpful pointers we used along the way.

Now we had 63 VMs available to process data but hit a roadblock. How do we launch our R scripts on the VMs in a coordinated way? For this, we ended up using another script by Warren (Invoke-AzureRmVmScript) to invoke commands remotely on the VMs. We wrapped these commands in the Invoke-Parallel script to kick off the R scripts simultaneously across the VMs. An additional script served the purpose of deallocating VMs after the R scripts finished running to measure progress and limit costs. Allocated VMs charge per minute and deallocated VMs carry no compute charges.

Once all the VMs completed their tasks we collected our data and analyzed our results. In the end we ran a total of 90 days' worth of parallel compute time across the VMs, with the longest VM running for a total of three-and-a-half days at a total cost of around $3,000. The equivalent cost of buying and setting up similar machines would have required weeks of setup and tens of thousands of dollars of hardware purchase for the same result. Of course, the cloud approach also required a fair amount of time spent crafting and debugging the PowerShell scripts, which adds significant soft costs in addition to the hard costs. Additionally, when using an IaaS solution over time there would also be the ongoing costs associated with keeping the VM image up-to-date with the latest security updates.

THINGS KEEP ON EVOLVING

After completing our first large run in the cloud, we found that Microsoft was working on an R package simultaneously that automated many of the tasks we had done in PowerShell. This R package is called doAzureParallel, leveraging an Azure service called Batch. The package allows a user to create a pool of VMs in the Azure Batch service with a few lines of R code and then register it as the parallel back end for the R *foreach* package. If you are already familiar with the R *foreach* package then making the transition to using doAzureParallel is done simply by running some code that creates the pool in Azure Batch. Any existing *foreach* code using the *%dopar%* function can then be used as is.

Azure Batch allows you to easily launch a pool of Linux VMs, which as we mentioned earlier is much more cost-effective than using a pool of Windows-based VMs. The auto scaling features of Azure Batch allow dynamically scaling up or down the number of VMs in a pool based on the demand of the tasks you are running. Another option is to use a mix of dedicated or low-priority VMs in a pool. Cloud providers make excess compute capacity available at steeply discounted rates with the caveat that these machines can be interrupted by those willing to pay at the higher rate. If this happens, the current task you are running gets canceled and reassigned on another low-priority machine. Therefore, it is recommended to only use the low-priority machines if you have short-running tasks or your calculation can progress despite multiple restart attempts.

One recently added feature of doAzureParallel worth noting is its ability to seamlessly run R inside a Docker container on the VMs within your pool. This is similar to how we cloned a

custom VM image in our initial IaaS approach. It allows use of a prespecified environment that keeps R versions and packages in sync, which ensures reproducibility of results. The added benefit with the doAzureParallel Docker container approach is that now you can rely on Azure Batch to create up-to-date VMs each time you run an analysis, ensuring that you have the latest security updates. By default, doAzureParallel uses the "rocker/tidyverse:latest" image that is developed and maintained as part of the rocker project.[4] However, you can also specify a custom Docker image, which allows you to lock in a version of R if you are concerned about duplicating results long term.

In our case, doAzureParallel has helped us move our initial IaaS approach to more of a PaaS approach. Now we can rely on doAzureParallel to maintain the administration work of creating pools of VMs with up-to-date security updates, which are running our prespecified environments. Using such solutions allows users to focus more on the analysis they are trying to conduct rather than spending the time managing the infrastructure it runs on.

LESSONS LEARNED AND RECOMMENDATIONS

Taking a look back at our journey in the cloud, we have some final recommendations for those looking to get the most out of these exciting new tools.

- If you plan on using the cloud for an analysis in R, check out the well-documented doAzureParallel package. Even if you don't plan on using R for analysis you might find some workflows that help with other languages as well.
- The tools cloud providers have are constantly evolving and iterating, and it is essential to be aware of what new tools are made available. For example, moving from the highly manual cloning of machines to Azure Batch for automated compute pool creation was revolutionary and much easier to use.
- We highly recommend the use of Docker containers or some other state management when conducting work in R or any other language if you need repeatable results over a long span of time.
- Finally, we recommend using Linux-based VMs over Windows if your task allows you to, as it can provide a welcome cost savings. Also investigate the use of low-priority VMs (or spot pricing in the AWS world) if your workflow supports short-running tasks.

Table 1 gives an estimate of potential cost reductions we could have achieved if we were to rerun our analysis applying these recommendations using the doAzureParallel package. For comparison, we have also estimated the cost of using AWS as the cloud provider. Note that these are estimated costs as of Jan. 23, 2018; pricing may vary in your region or the contract you have in place with Microsoft Azure or AWS.

Table 1
Potential Cost Reductions

VM Option	Total Compute Hours	Price Per Hour[1]		Total Cost	
		Azure[2]	AWS[3]	Azure	AWS
Windows OS	2,151	$1.17	$1.05	$2,516.67	$2,258.55
Linux OS	2,151	$0.78	$0.67	$1,677.78	$1,441.17
Linux OS with low priority[4]	2,151	$0.14	$0.07	$301.14	$150.57

1. Estimated prices from Microsoft Azure and AWS online pricing for VM compute charges only. Does not Include storage or data transfer prices, which can become meaningful if not managed efficiently.
2. Azure A10 VM with eight cores and 56 gigabytes of RAM in the North Central U.S. region.
3. AWS r.3.2xlarge VM with eight cores and 61 gigabytes of RAM in the U.S. East (Ohio) region.
4. Assumes tasks were run without the VMs being preempted.

As you can see, the cloud is more than just someone else's computer. It's an ecosystem of resources that can be leveraged to explore ideas and complete tasks that were once unfeasible to achieve with the local computing resources of the past. ■

Joe Long is an assistant actuary and data scientist at Milliman. He can be reached at *joe.long@milliman.com*.

Dan McCurley is the Cloud Solutions Architect at Milliman. He can be reached at *dan.mccurley@milliman.com*.

ENDNOTES

1 A research and development project conducted by the Milliman Advanced Risk Adjusters™ (MARA™) product group. See *http://www.millimanriskadjustment.com* for more information about MARA.

2 Mohamed, Arif. A History of Cloud Computing. *Computer Weekly.com*, March 2009, *http://www.computerweekly.com/feature/A-history-of-cloud-computing* (accessed Feb. 1, 2018).

3 F., Warren. Invoke PowerShell on Azure VMs. *Rambling Cookie Monster*, *http://ramblingcookiemonster.github.io/Invoke-AzureRmVmScript/* (accessed Feb. 1, 2018).

4 Tan, J.S. Scale Up Your Parallel R Workloads with Containers and doAzureParallel. *Revolutions*, Nov. 21, 2017, *http://blog.revolutionanalytics.com/2017/11/doazureparallel-containers.html* (accessed Feb. 1, 2018).

SOCIETY OF ACTUARIES®

2018 Predictive Analytics Symposium

September 20–21

Minneapolis, MN

Explore the world of big data and how it impacts the actuarial profession.

Register at Soa.org/PASymposium

The Forgery Game: Generative Adversarial Networks

By Michael Niemerg

Imagine a not-too-distant future. You open your mailbox to find a pretty ordinary-seeming catalog. You start to flip through it. Inside, you find pictures of beautiful, smiling people. You see perfectly manicured lawns and perfect bedrooms. The catch: None of this is real. These images weren't even created using computer graphics. All these images were created by a model—by a generative adversarial network (GAN).[1,2] Don't believe this is possible? There are already images of fake people that look eerily realistic[3] and ways to manipulate an image to turn that smile into a frown.[4]

What is a generative adversarial network? How does it create synthetic images of people and things that are nearly indistinguishable from real photos? The first thing we need to do is parse the moniker itself. The "generative" part of generative adversarial networks refers to what the model is doing: generating synthetic data. The "adversarial" refers to how it is trained—in an adversarial fashion between two competing models. The "networks" refer to the model form, which are neural networks (while there is no requirement that generative adversarial models *must* be neural networks, this is the primary focus of active research in the area).

TRAINING GANS

Let's dive a little more into how these models are trained. GANs are created via two competing networks: a generator that creates synthetic data and a discriminator whose job it is to distinguish the real data from the synthetic data. This adversarial connection is the whole key to the process. By putting the models in competition, the generator is forced to successively get better at creating data that looks real while the discriminator gets increasingly more adept at separating real data from synthetic data. A common analogy used to describe GANs is to think of the generator model as an artwork forger, trying to pass off forgeries as the real thing, while the discriminator plays the role of the curator trying to identify the real art and reject the forgeries. The forger gets continually better at generating the fake artwork but the curator also improves at spotting the real art apart from the forgeries.

GAN models are neural networks. While the relationship between the generator and the discriminator is unique, all the typical rules and structure of training neural networks apply to both. If the jargon of neural networks is foreign to you, simply remember that a neural network is a predictive model. It will take in some data, have parameters that will be fit by optimizing an objective, and ultimately produce output (the synthetic data for the generator, and the probability of data being real or synthetic for the discriminator).

Now let's get a little more precise on the algorithm for GANs.

GAN ALGORITHM

For each round of training:

- Generate random points from latent space (a good choice would be random numbers from a normal distribution) and create the synthetic data by feeding the random points into the generator.
- Combine this synthetic data with the real data.
- Train the discriminator to distinguish between these real and synthetic values.
- Update the generator to fool the discriminator:
  - Freeze the discriminator so that its weights do not change.

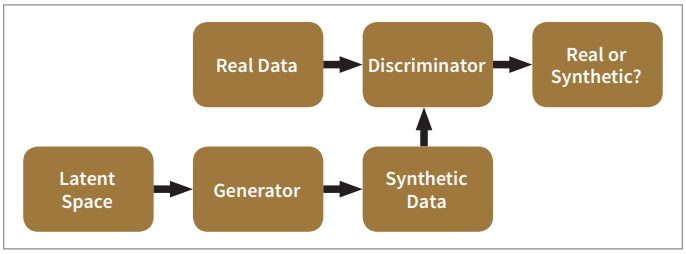
- Feed the generator random points from latent space as input.

- The generator will convert these random points to synthetic data.

- The frozen discriminator will then classify this synthetic data as "real" or "synthetic."

- Update the weights in the generator to alter how it creates its synthetic data so that it can more easily fool the discriminator.

Figure 1
A Representation of the GAN Model-Building Process



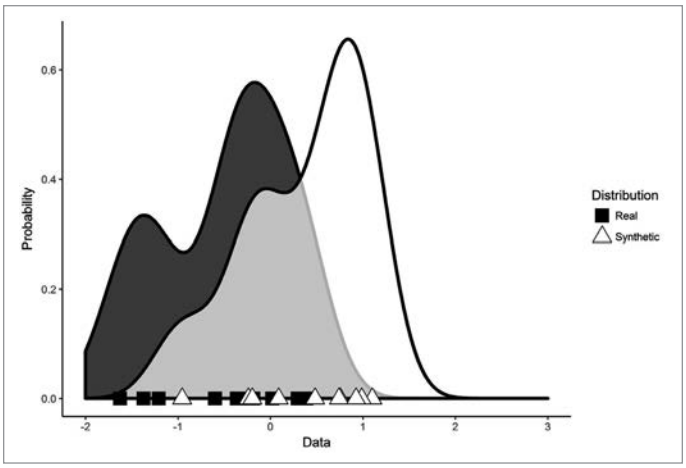
The last step above can seem a bit curious so let's look more closely at what is happening. In more precise terms, this step in the process is trying to minimize the difference between two vectors of numbers (with each entry in the vectors corresponding to an observation). Keeping in mind that the discriminator is being fed a series of synthetic observations, the first vector is the discriminator's prediction of whether each of these observations is real or synthetic. The second is simply a vector of targets that say that each observation is real. Because the generator is trying to fool the discriminator, it wants to get them as close as possible. However, while training with this objective, the generator is unable to manipulate the discriminator directly (in fact, being frozen, the discriminator doesn't change at all in this last step) but the generator is still able to indirectly alter the first vector (the discriminator's predictions) by altering its own weights so that its generated output becomes harder for the discriminator to distinguish from the actual data.

Ultimately, the generator is doing a good job when the discriminator can't tell the difference between synthetic data and real data (e.g., the predicted probability of either is 50 percent). The coolest part? Throughout this whole training process, the generator has no access to the real images! It learns to create them without ever having direct access to them.

Another way to think about what the model is doing is to think about our real sample data as coming from a high-dimensional, data-generating distribution. When training a GAN, our training set is really a sample of data points from this data-generating distribution. The GAN model uses this sample data to learn about the structure of the entire data-generating distribution so that it can learn how to approximate new samples from it.

For an illustrative example, see Figure 2. Our data set to build our GAN is a sample of points (black boxes) from the data-generating distribution (gray distribution). Our model learns an (imperfect) representation of that distribution (white distribution) from which we can draw samples (white triangles).

Figure 2
Data-Generating Distribution and GAN Approximation



CHALLENGES WITH TRAINING GANS

Currently, generative adversarial modeling is still an active area of research. There are several ways in which GANs can fail or in which training them can produce fickle results.

The most common is simply instability in training. For instance, training the model with the same parameters might work well in one training run only to produce poor results in another run without any changes to the model parameters other than different random number initializations.

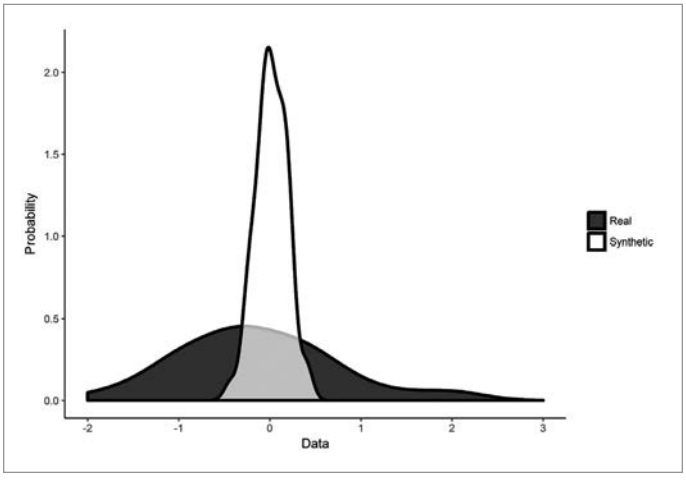
Another problem with GANs is that measuring the quality of the synthetic data can be difficult. While both the generator and the discriminator have a loss function, these loss functions are really only optimizing the competition against its adversary. In a regression problem, we know that higher R-squared is better

and, in a classification problem, that higher accuracy is better (*ceteris paribus*). If our task is generating realistic synthetic images, however, our real objective is independent of the nominal value of the loss function but is instead tied to how convincing the image is to a human. Because it is hard to come up with a good loss function for how different the synthetic picture of a bedroom is from a real bedroom, it can be hard to tell exactly when one GAN model performs better by simply checking metrics. We need to actually examine our sample output.

Another difficulty with training GANs is that they have a tendency to collapse into similar output for different input from the latent space. Part of the reason for this is that the GAN model can only look at each instance in isolation when determining whether a data point is real or synthetic. Why is this problematic? Well, imagine, for instance, that you wanted some synthetic data representing the rolls of a six-sided die. If I presented you with a 0 or a 7 you would easily recognize those data points as unrealistic. However, what if I presented you with a 4? That seems to be a very plausible die roll. What if I then generated for you a never-ending series of 4s as synthetic data? If you are constrained to only being able to look at one data point at a time to judge whether an instance looks real (i.e., we are memoryless like a GAN), you can't discriminate these obviously synthetic data points from real points. This is problematic. We need some way of relating observations to each other to tell the difference.

In Figure 3, we can see an example of a degenerative GAN. The GAN fails to learn a good representation of the true data-generating distribution, instead only learning to reproduce frequent values that lie near the mean of the data-generating distribution.

Figure 3
Data-Generating Distribution and GAN Approximation: Degenerative Example



Another challenge with GANs is one that faces all predictive models: They inherit the biases of the data used to train them. Say, for instance, we are training a model to generate images of bedrooms. Let's also suppose only a small percentage of bedrooms contain yellow bedsheets and that none of these bedrooms make it into our training set for the GAN model. What could likely happen is that our model will not learn to associate yellow bedsheets with bedrooms and our synthetic images will contain no yellow bedsheets even though they exist in the real world. Our model can only reconstruct the data-generating distribution to the extent that it is faithfully represented in our training data.

PRACTICAL TIPS AND ADVANCED ARCHITECTURES

Multiple techniques exist for aiding the training of GANs. Some techniques include: modifications to the loss function used in training, incorporating common neural network regularization techniques into the training phase, and adding some extra challenge to the discriminator by introducing noise to its input. Many of these techniques are incorporated into advancements to the original GAN algorithm.

A few of the advanced GAN algorithms are particularly noteworthy. Deep convolutional GANs (DCGANs)[4] improve upon GANs by offering refinements to the architecture of the neural networks used to train them. Wasserstein GANs[5] add several wrinkles to GANs, including using a loss function whose numerical value corresponds more closely with the true quality of the synthetic data. Furthermore, the idea of mini-batch discrimination[6] was created to counter the tendency of models to collapse to a narrow output range by adding distance information about other examples from within each training mini-batch to the discriminator.

Generally, research on GANs is proceeding at a rapid clip. In all likelihood, significant improvements have been made to GANs between when I wrote this article and the time it went to print.

DOES IT MATTER TO ACTUARIES?

Much of the work with GANs to date has been on synthetic image and audio generation but that is quickly changing. Will GANs ever make their way to the insurance or health care sectors? The future is still to be seen, but the potential is there as the quality of the algorithms mature and the use cases become more apparent.

One possible use for GANs could be to generate data synthetically to feed into other predictive models when training data is scarce. Various types of data set augmentation are already common practice when creating neural networks for image analysis. GANs could simply become another extension of this practice.

Another more creative use for GANs could be in the realm of data-sharing. Imagine being able to share the data needed to

build predictive models without sharing the data itself. Instead of training the predictive model with real data, one party could train a GAN on its data to "encrypt" it. The other party could then generate synthetic data from the GAN and use that synthetic data to actually train the ultimate predictive model. The only thing that needs to be shared is the neural network itself. In this way, data insight could be shared without actually sharing data.

These use cases are speculative at the moment but not unrealistic. It's still too early to tell whether GANs rise to prominence as another commonplace method in the modeler's toolbox or whether they remain a curiosity. ■

Michael Niemerg, FSA, MAAA, is an actuary at Milliman in Chicago. He can be reached at *michael.niemerg@milliman.com.*

ENDNOTES

1 Goodfellow, Ian J., et al. 2014. Generative Adversarial Nets. Advances in Neural Information Processing Systems, *https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf* (accessed Feb. 9, 2018).

2 Creswell, Antonia, et al. Generative Adversarial Networks: An Overview. Cornell University Library, Oct. 19, 2017, *https://arxiv.org/abs/1710.07035* (accessed Feb. 9, 2018).

3 Vincent, James. All of These Faces Are Fake Celebrities Spawned by AI. *The Verge*, Oct. 30, 2017, *https://www.theverge.com/2017/10/30/16569402/ai-generate-fake-faces-celebs-nvidia-gan* (accessed Feb. 9, 2018).

4 Radford, Alec., Luke Metz, and Soumith Chintala. Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks. Cornell University Library, Nov. 19, 2015, *https://arxiv.org/abs/1511.06434v2* (accessed Feb. 9, 2018).

5 Arjovsky, Martin, Soumith Chintala, and Léon Bottou. Wasserstein GAN. Cornell University Library, Jan. 26, 2017, *https://arxiv.org/abs/1701.07875* (accessed Feb. 9, 2018).

6 Salimans, Tim, et al. Improved Techniques for Training GANs. Cornell University Library, June 10, 2016, *https://arxiv.org/abs/1606.03498* (accessed Feb. 9, 2018).

Why Consider a Delphi Study?

By Ben Wolzenski

In the December 2017 *Predictive Analytics and Futurism Newsletter*, author and recent Predictive Analytics and Futurism (PAF) Section Council member Bryon Robidoux wrote about the TED talk, "The Human Insights Missing from Big Data," by Tricia Wang. I highly recommend that article, which also contains a link to access the TED talk. It provides a perfect preface to this article about an old futurism tool in the new world of predictive analytics: the Delphi study. Both articles support the idea of supplementing the results of a model with data from alternative sources to help validate the model. A more scientific way than relying on yourself or a co-worker for insight is to use a Delphi study.

Like predictive analytics, the Delphi method is used for forecasting. But there they diverge; instead of tools and data, the Delphi employs a panel of experts ("panelists") to address specific questions or issues. But unlike a roundtable discussion or a mere survey, the Delphi technique gathers responses from panelists anonymously, and sends all those separate responses (again, anonymously) to each panelist. The panelists are asked to reconsider and possibly refine their responses based on the information gleaned from the responses of all the others. These "rounds" of questions and answers are repeated until the respondents stop making material changes to their answers. The result may be a consensus, or convergence around two or more points of view.

The Delphi method is most useful when other forecasting techniques, especially those that use past data to estimate future outcomes, appear to have limited value. Or when the forecaster simply feels the need for a second opinion, derived by other means. The Delphi method has been around since the 1950s, but was almost unused by the actuarial profession until 2005, when the Society of Actuaries (SOA) published "A Study of the Use of the Delphi Method, A Futures Research Technique For Forecasting Selected U.S. Economic Variables and Determining Rationales for Judgments." That landmark study was as much (or more) about how to perform a Delphi study as it was about predicting economic variables in 2024 (and the rationales for those predictions).

Then, in 2009, the SOA published "Blue Ocean Strategies in Technology for Business Acquisition by the Life Insurance Industry." In three rounds of narrative questions and panelists' responses, a series of strategies were identified and refined. Here are two examples:

- Strategy #5: Your Way Insurance Company—"Prospects custom-design coverage online"
- Strategy #8: Holistic Insurance Company—"Risk 'agents' help mitigate all risks"

The next major Delphi study by the SOA was spearheaded by the Long Term Care Think Tank and published in 2014: "Land This Plane," with the goal of arriving at a consensus on solutions to the nation's long-term care financing challenges. There were widely different views about the roles of government and the insurance industry among the long-term care experts recruited to be panelists. Despite these differences, the final report identified a series of principles upon which there was general (albeit not unanimous) agreement.

And even as this article was written, the SOA has launched a second Delphi study regarding economic variables, with a focus on methods and assumptions for financial projection models. With an ever-greater world of data at our disposal, the comprehensive training of actuaries gives us an advantage in applying human insight—and the Delphi method can provide a means to derive value from that insight.

Ben Wolzenski, FSA, MAAA, is managing member at Actuarial Innovations, LLC in St. Louis. He can be reached at *bwolzenski@gmail.com*.

Hierarchical Clustering: A Recommendation From a Nonhierarchical Manager

By Dave Snell

Most of the people who know me well are aware that I'm not a big fan of hierarchical management. Back when I was VP over a fairly large area I used to value highly the direct reports who felt comfortable challenging my ideas; and the collaborative outcomes from our discussions were often far better than my original thoughts.

So, it might seem strange that my first choice on an article to describe clustering is about the benefits of hierarchical clustering as opposed to the more commonly used nonhierarchical techniques such as *k*-means clustering. Both categories are usually unsupervised machine learning techniques (techniques where you do not know the outcomes or labels ahead of time); but *k*-means clustering intuitively appeals to mathematicians because it is easy to conceptualize (but not visualize) in several dimensions.

In *k*-means clustering, you just pick a *k* (the desired number of clusters), assume *k* random points in your data as the initial centers of the clusters, assign each data point to one of the clusters based on their distances from those *k* centers, and then compute new centers for each cluster based on the distance metrics. Since the initial choices were random, it is likely they were wrong. At the next round of point assignments, some points are reassigned to another cluster based on closeness to the new centers you calculated. Again, the cluster centers are recalculated and the process continues until points stop changing from one cluster to another. This method is computationally efficient, easily accommodates several dimensions of factors, and, again, it appeals to mathematicians.

Unfortunately, it is not always the most appropriate clustering technique. As you can see in Figure 1, *k*-means can do a good job if the underlying data clusters are distinct (not overlapping), and the underlying clusters are somewhat spherical in nature and of similar density. If the data is donut-shaped, or follows a specific curve, or is radial in nature, as in Figures 2 and 3 (pg. 16), it does not give a good result.

Figure 1
Example Where *k*-Means (Where k=4) Works Well[1]

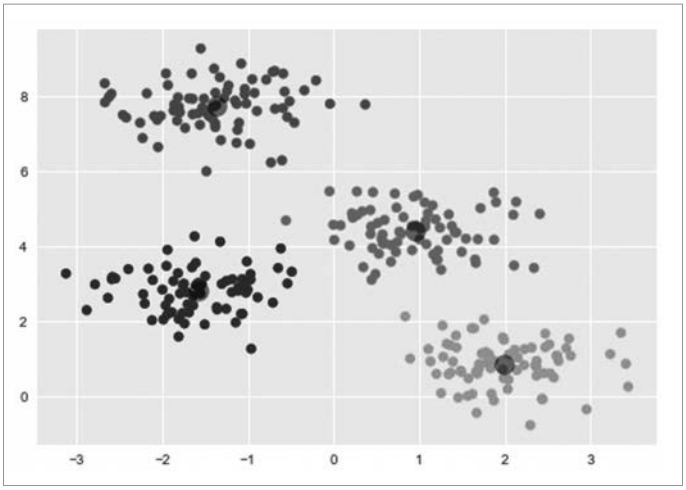
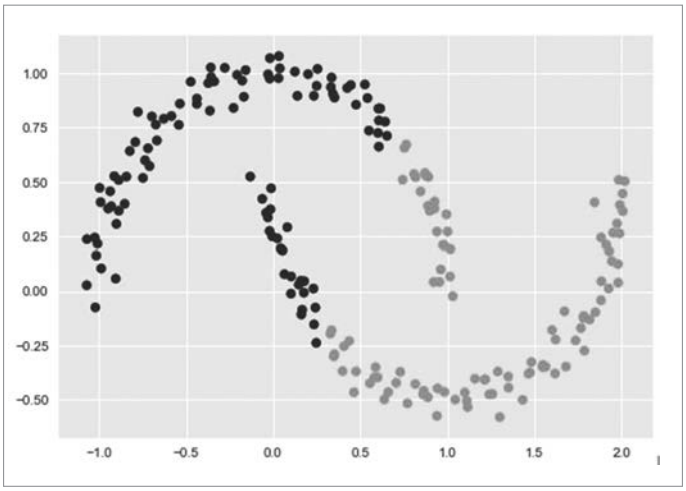
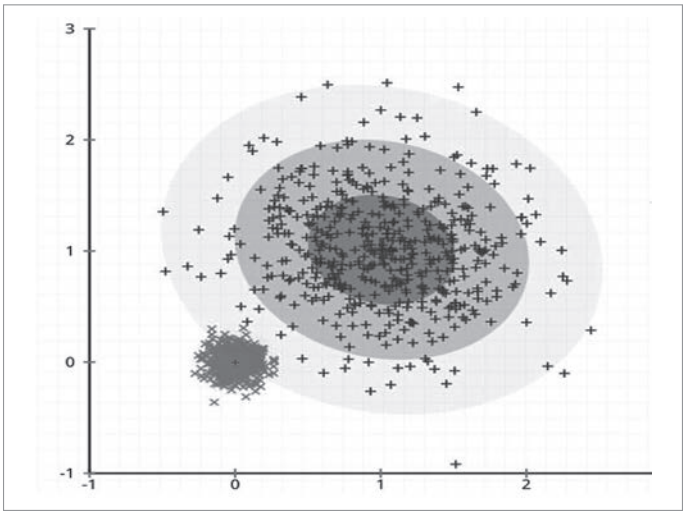


Figure 2
Example Data Where *k*-Means Does Not Work Well (consider an affinity method instead)



Beyond this, *k*-means clustering requires you to choose the number of clusters (*k*) ahead of time. If you are doing an exploratory analysis of a large set of data, you may not know the appropriate *k* ahead of time. Granted, you can try several different values of *k* and see where the sweet spots seem to be on an elbow curve; you can do a silhouette analysis; and you can measure the purity of each cluster; but these tests can introduce complexity rather than clarity.

Figure 3
Example Data Where *k*-Means Does Not Work Well (consider a Gaussian mixture model instead)



Most of all, though, the *k*-means approach is not as easy to explain to nonmathematicians, and once you get to higher dimensions, where scatter plots may not be appropriate, it lacks a visually intuitive presentation mechanism.

In cases of higher dimensionality,[2] such as four or more, you may wish to consider a hierarchical clustering approach. Even three-dimensional clusters can be very misleading when shown in two dimensions. A famous anamorphic creation by the artist Michael Murphy titled "Perceptual Shift" shows this vividly. Looking at it from the front, it appears to be a human eye; but from the side it is a cone of seemingly scattered balls.[3] The most recognizable pattern of stars in the northern hemisphere, the Big Dipper (actually part of the constellation Ursa Major) looks like a flattened ladle from Earth; but Mirza, the closest star of the seven, is 78 light years away from us while Dubhe, the farthest, is 123 light years away! Seen from another galaxy, this group of stars looks nothing like a dipper.

A hierarchical clustering approach starts with the assumption that every data point is its own cluster. Then, it computes the distance between each pair of clusters and starts grouping them accordingly.

In order for the algorithms to work, there are four distance rules we have to specify:

1. Distance cannot be negative: $d_{ij} > 0$ when $j \neq i$ (i.e., the distance from cluster i to a different cluster j is positive).
2. Distance from any cluster to itself is zero: $d_{ii} = 0$.
3. Distance is symmetric: $d_{ij} = d_{ji}$ (i.e., the distance from cluster i to cluster j is the same as the distance from cluster j to cluster i).
4. A triangular inequality holds: $d_{ij} + d_{jk} >= d_{ik}$.

Given these rules, we can choose any of a number of different metrics for "distance." Some common choices are shown in Figure 4.

Figure 4
Commonly Used Distance Metrics for Hierarchical Clustering[4]

Names	Formula
Euclidean distance	$\|a-b\|_2 = \sqrt{\sum_i (a_i - b_i)^2}$
Squared Euclidean distance	$\|a-b\|_2^2 = \sum_i (a_i - b_i)^2$
Manhattan distance	$\|a-b\|_1 = \sum_i \|a_i - b_i\|$
Maximum distance	$\|a-b\|_\infty = \max_i \|a_i - b_i\|$
Mahalanobis distance	$\sqrt{(a-b)^\top S^{-1}(a-b)}$ where *S* is the Convariance matrix

Figures 5 and 6 give an idea of what this process looks like visually. Initially, let's assume that we had only six data points. We start out assuming each is its own cluster. Alternatively, if you feel this is too trivial an example, we might wish to say that Figure 5 is the result of previous clustering of a large number of points already; and we are now down to six clusters.

We see in Figure 5 that clusters *b* and *c* are very close to each other, as are clusters *d* and *e*. This is reflected in Figure 6, as the number of clusters is reduced in Round 1 to four: clusters *a*, *bc*, *de* and *f*.

In the next round we note that cluster *de* is closer to cluster *f* than to any other cluster so they are combined into cluster *def*. Next, *def* is combined with cluster *bc* to obtain cluster *bcdef*. Finally, cluster *a* is combined with *bcdef* to form the single cluster *abcdef*. Usually, hierarchical clustering methods are also called agglomerative methods,[5] and you can see why here. Eventually, you end up with just one cluster.

At this point, you might be wondering where I am going with this discussion. Why is the lumping together of all the data into just one cluster of any use to us?

The usage comes into play via a special sort of tree diagram, called a dendrogram. A dendrogram of the clustering process we did for our example is shown in Figure 7. Note that this is a visual way of showing how the clusters are combined and also the relative dissimilarity between the clusters. The taller the height before two clusters are combined, the more dissimilar they are. We see that cluster *a* was most different from all of the other clusters, while *d* and *e* were relatively close.

Let's consider a more practical example of how hierarchical clustering can be useful.

Assume your daughter (or son or niece or nephew or friend) is a junior or senior in high school and wants to apply to a university with the intent of a double major—in actuarial science and data science. You want to help in this project, so you compile a list of 40 or so universities that offer both of these majors. The parameters for selection may include items such as student population, ratio of students to faculty, percentage of scholarships available, distance from home (far enough away for autonomy

Figure 5
Six Clusters Prior to Hierarchical Clustering

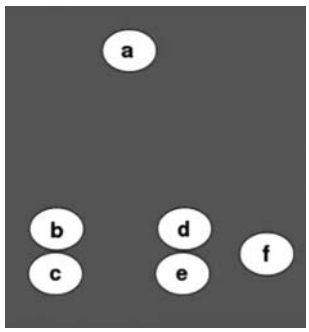
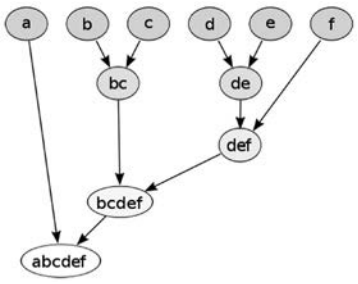


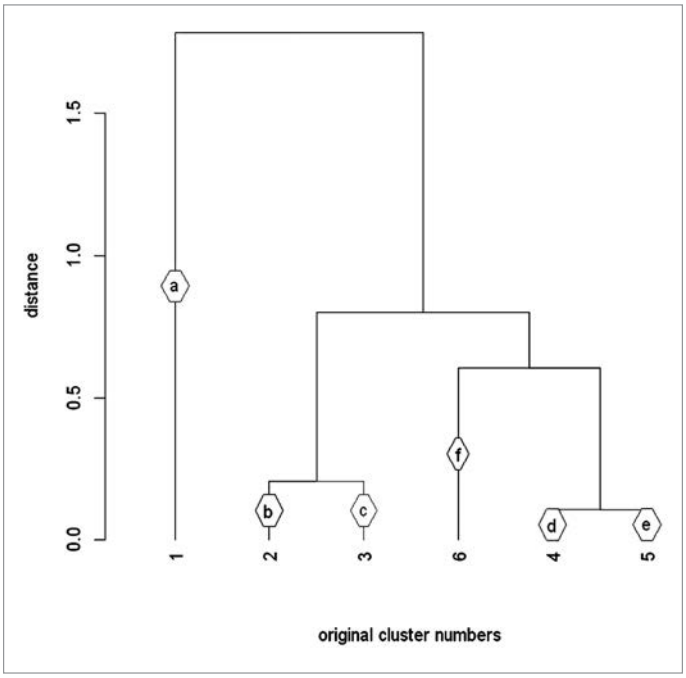
Figure 6
Traditional Representation of Hierarchical Clustering



Another benefit of hierarchical clustering is repeatability.

Unlike k-means clustering, which can result in different answers based on different starting values for the randomly chosen first set of center points, hierarchical clustering is repeatable. As long as your data has not changed, and you use the same distance metric, you will always get the same result.

Figure 7
Dendrogram of Six Clusters[6]



and close enough to bring laundry home), housing costs and tuition, number of Nobel Laureates teaching classes, median SAT and ACT scores of incoming students, median compensation of graduates after five years, athletic team performances, cultural opportunities, male-female student ratio, international student ratio, cafeteria selections, average temperature range, proximity to the ocean or the mountains, population of nearby city, Centers of Actuarial Excellence (CAE) status, data science rating and perhaps several other criteria.

You don't want to risk applying to only one university, since you can't predict how selective they may be. Perhaps the admissions officer at the interview will be impressed by her initiative and creativity to make an interview video while juggling on a skateboard to show multitasking ability. But what if the interviewer considers this an indicator of a frivolous nature? On the other hand, each application is expensive both in dollars and in the time spent visiting the campus and researching the overall school environment. It would be nice to be able to say with some confidence that a specific subset, or group within these 40 schools, is most similar to this student's interests and abilities. This can be an ideal problem for a hierarchical clustering solution. You have many dimensions and it is not obvious how to group the schools into logical clusters.

It will be necessary to convert the categorical factors, such as CAE status and cultural opportunities to numeric values—often via dummy variables. Then there is the issue that some of these numeric parameters have wide ranges relative to others. For example, the number of students might be just a few hundred, or many thousands. Expenses and distance from home may also have wide ranges. Compare those to the number of Nobel Laureates, where 0 to 5 might cover every one of the schools. In order to avoid having the wide-range items completely overshadow the importance of short-range ones, we would employ statistical techniques to standardize and normalize our values. One such technique might be to substitute each value x_i with $(x_i - x_{mean})/x_{standard\ deviation}$, which would work fine for a mix of all numeric parameters, but still tends to have higher weight than the categorical surrogates that range from 0 to 1. In a mixed parameter environment, it might be better to map x_i to $(x_i - x_{minimum})/(x_{maximum} - x_{minimum})$, thus ensuring all the items have the range 0 to 1.

Once you have your values normalized, both Python and R have packages that can do all the heavy-lifting work of creating the dendrogram for you. R, in particular, has a package *dendroextras* that allows you to label and color your clusters:

```
if (!is.element('dendroextras',
  installed.packages()[,1]))
  install.packages("dendroextras",
    repos='http://cran.us.r-project.org')
```

I don't have all those parameters available for my hypothetical problem, but I did find a ranking of world university rankings on Kaggle at *https://www.kaggle.com/mylesoneill/world-university-rankings* that I will use for a very quick demonstration of how to generate a dendrogram of the universities. In this demonstration, I'll keep it simple and use the built-in hierarchical clustering in R:

```
# file from Kaggle site in text
input <- read.csv('cwurData.csv')
tail(input)
```

that produces Figure 8.

Figure 8
Sample of Kaggle University Rankings (Kaggle dataset has 1,000 universities in this dataset)

	world_rank	institution	country	national_rank	quality_of_education	alumni_employment	quality_of_faculty	publications	influence	citations	broad_
2195	995	King Abdulaziz University	Saudi Arabia	4	367	449	218	595	430	645	
2196	996	University of the Algarve	Portugal	7	367	567	218	926	845	812	
2197	997	Alexandria University	Egypt	4	236	566	218	997	908	645	
2198	998	Federal University of Ceará	Brazil	18	367	549	218	830	823	812	
2199	999	University of A Coruña	Spain	40	367	567	218	886	974	812	
2200	1000	China Pharmaceutical University	China	83	367	567	218	861	991	812	

Now, we generate the dendrogram:

```
# just take the top 40 for this example
uniRatings <- input[1:40,c(2,1,4:10)]
# exclude university name and normalize
normalizedRatings <- scale(uniRatings[,2:9])
distance <- dist(normalizedRatings,
  method='euclidean')
clus <- hclust(distance, method='complete')
plot(clus,hang=-1) # display the dendogram
# cut the dendogram into 5 clusters
groups <- cutree(clus, k=5)
rect.hclust(clus, k=5, border='red')
# output is Figure 9
```

I then add a new column that denotes group number to the data frame:

```
uniRatings$group <- groups
uniRatings[1:8]
# Output is Figure 10
```

Figure 9
Top 40 World Universities in Five Clusters

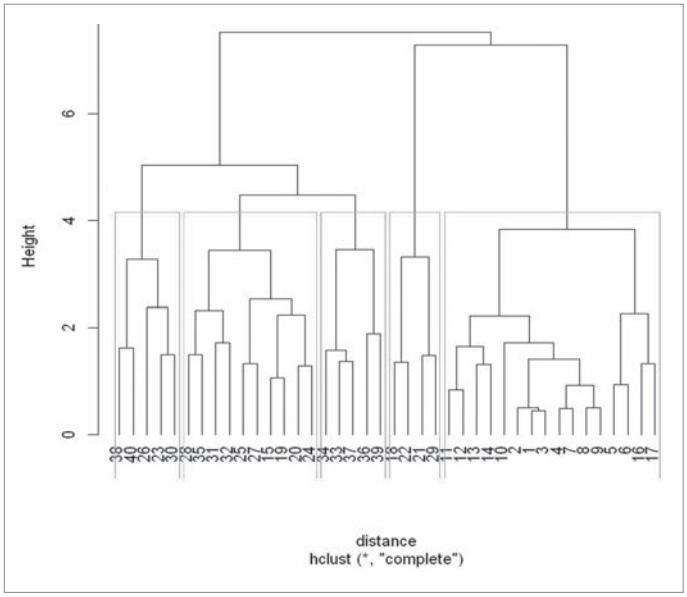


Figure 10
Section of Group 1 of the Top Universities

institution	world_rank	national_rank	quality_of_education	alumni_employment	quality_of_faculty	publications	influence	citations	group
Harvard University	1	1	7	9	1	1	1	1	1
Massachusetts Institute of Technology	2	2	9	17	3	12	4	4	1
Stanford University	3	3	17	11	5	4	2	2	1
University of Cambridge	4	1	10	24	4	16	16	11	1
California Institute of Technology	5	4	2	29	7	37	22	22	1
Princeton University	6	5	8	14	2	53	33	26	1
University of Oxford	7	2	13	28	9	15	13	19	1
Yale University	8	6	14	31	12	14	6	15	1

Our top group is probably no big surprise. The highest-rated universities are in the same group.

But later, we find some surprises, as the 10 universities shown in Figure 11 are all ranked very similarly (31 through 40), but they are not that much alike when you consider all of the parameters. In fact, University College London is more like Osaka University or University of Toronto than it is like Northwestern or Washington University in St. Louis. Of course, different criteria, such as my hypothetical ones, would group all these universities differently, but that is part of the beauty of hierarchical clustering: You get to decide what features are important, and the similarity grouping is based only upon them.

```
uniRatings[31:40,]
# output is Figure 11
```

In this article, I expressed my opinion that hierarchical clustering can provide advantages over *k*-means clustering when the number of dimensions, *n*, is too high for a scatter plot.[7] The dendrogram is a convenient way to show both the clusters and the relative dissimilarity between them. It also lets you choose a cut point (number of clusters) after construction of the dendrogram so you can see logical groupings by extent of dissimilarity before you do more calculations. I hope you find the examples using R useful. Python has very similar capabilities. Whichever programming language you prefer, I think it is worth investigating this underutilized technique for clustering. ■

Dave Snell, ASA, ACS, ARA, ChFC, CLU, FALU, FLMI, MAAA, MCP, teaches AI Machine Learning at Maryville University in St. Louis. He can be reached at *dave@ActuariesAndTechnology.com*.

ENDNOTES

1 Just because a scatter plot looks good in two dimensions does not mean it actually represents the data arrangement. See a detailed description of the anamorphic creation by Michael Murphy, "Perceptual Shift," at *https://mymodernmet.com/michael-murphy-perceptual-shift/*.

2 Although hierarchical clustering is good for *n* dimensions, where *n* is often > 3 and beyond those we can readily graph, it involves the computation and storage of an *n* by *n* matrix, which can be a strain on computing and storage resources.

3 *Supra*, note 1.

4 Figures 4, 5, and 6 are derived from Wikipedia. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled GNU Free Documentation License.

5 Actually, hierarchical clustering can be agglomerative (the usual case) where you start with *n* points and keep combining them until you have only one cluster; or they can be divisive, where you start with one cluster, then keep subdividing it.

6 Figure 7 was generated by the author using the R package *dendroextras*.

7 *Supra*, note 2.

Figure 11
Another Section of the Top University Rankings, Showing Varying Groupings

	institution	world_rank	national_rank	quality_of_education	alumni_employment	quality_of_faculty	publications	influence	citations	group
31	University College London	31	4	35	101	45	27	23	33	2
32	Osaka University	32	3	77	101	44	39	44	51	2
33	Northwestern University	33	23	101	32	101	24	25	20	5
34	University of Michigan, Ann Arbor	34	24	68	60	101	2	17	7	5
35	University of Toronto	35	1	101	101	34	7	14	18	2
36	University of North Carolina at Chapel Hill	36	25	101	86	56	31	29	31	5
37	Washington University in St. Louis	37	26	74	62	101	32	18	30	5
38	University of Utah	38	27	92	101	41	74	52	67	4
39	University of Washington - Seattle	39	28	101	101	40	5	7	5	5
40	University of California, Santa Barbara	40	29	101	101	28	68	72	36	4

SOCIETY OF ACTUARIES®

New Predictive Analytics Certificate

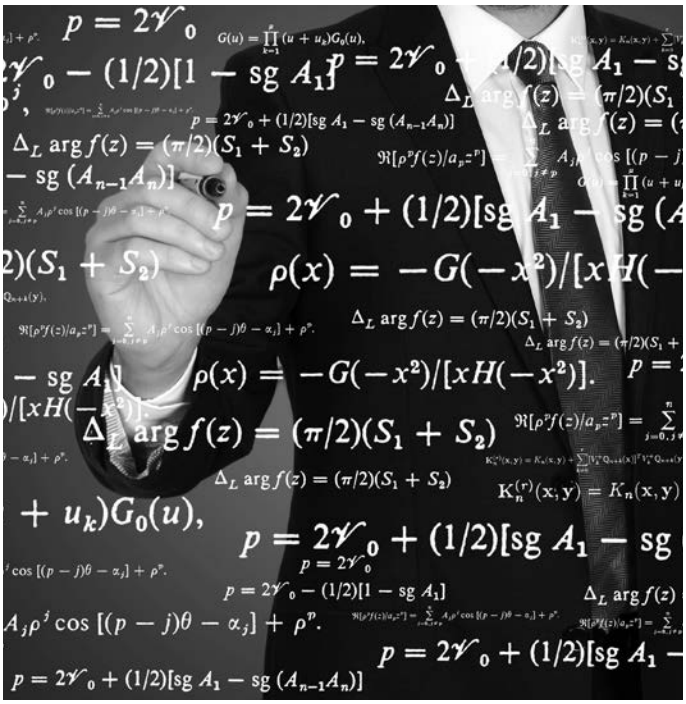
Prepare for Today's Data-Driven World

The SOA's Predictive Analytics Certificate is a new offering open to all credentialed actuaries. This five-month program is at the forefront of updated methods and knowledge with six e-learning modules and an in-person, two-day seminar with a project-based assessment.

Act Now. Space Is Limited. Visit *SOA.org/PACertificate*

Feature Importance in Supervised Training

By Jeff Heaton



Supervised learning is the class of machine learning where a model is trained to produce a specific result for a given input. These inputs and expected outputs form the training data for a model. Because the expected outputs are known, this type of training is referred to as supervised learning. If there are no expected outcomes, then the technique is referred to as unsupervised learning. The process of using these data is called training or fitting. Whether to use supervised or unsupervised learning depends upon the project goal. If the desire is to create a model that can be trained to produce some sort of output from input data, then you are using supervised training. The focus of this article is determining the importance of columns of your input data for supervised training.

In the domain of supervised learning, predictive models accept a feature vector and return a prediction. For example, a model might be asked to accept inputs that specify the face amount, annual premium, term, age of applicant, and other values to predict the likelihood of the policy being lapsed. These inputs are typically referred to as the feature vector or the x-values. The output from the model is typically referred to as the score, prediction or y-hat value. Some of the input features are more important to making an accurate prediction than others. For example, term length might be more important to predicting lapse than the face amount. There are a wide variety of techniques that can be used to measure the importance of the input features.

MODEL-SPECIFIC FEATURE RANKING

Depending on the type of model to be evaluated, there are a number of different ways to evaluate feature importance. These model-specific, feature-ranking techniques will change depending on what model you are using. For example, if you are dealing with a generalized linear model (GLM), the coefficients can provide an importance measure. Similarly, neural network feature importance can be gauged by examining the outbound weights from each of the input neurons.[1] Additionally, the importance of features in tree-based models, such as gradient boosting machines (GBMs), random forests, and classification and regression trees (CARTs) can be determined by evaluating the number and weighting of splits that the given feature was involved in.

Of course, these techniques are only valid for GLMs, neural networks and tree-based models. If you are making use of other model types, such as support vector machines (SVMs), k-nearest neighbors or any other, you will need to use a technique that is specifically designed for that model type. Furthermore, your importance will remain the same over time.

The importance of the model features is generated from the model parameters that were defined when the model was fit. It is not possible to see how important these features are with newer data sets that your model might need to score. Fitting a model and deploying it to production are only the first battles that a data scientist must face. It is important to ensure that your model remains relevant with new data sets and external conditions that might affect the validity of your model. Evaluating the importance of features for your trained model on new data sets can be an important piece of information in ensuring the continued robustness of your deployed model. Most model-specific, feature-ranking algorithms only analyze the model, and not the importance of features in entirely new data sets.

MODEL-AGNOSTIC FEATURE RANKING

Model-agnostic, feature-ranking algorithms consider the intrinsic characteristics of the data in evaluating the fitness of the feature subset. Model-agnostic, feature-ranking techniques do not require a learning algorithm and require fewer computing

resources. Rather, the model-agnostic algorithm makes use of an already trained model and a data set.

Correlation-coefficient feature importance is a very simple model-agnostic, univariate algorithm that calculates the absolute value of the correlation coefficient between each of a model's expected outputs. This value can be used to estimate the importance of each input feature to the model. The higher the correlation coefficient between an input (x) and the target (y), the greater a feature's importance. To calculate this coefficient, the first step is to calculate the covariance (C_{ij}) between the two features i and j. Usually, feature i will be the input feature currently being evaluated and j will be the target value. This is performed by the following equation:

$$C_{ij} = \sum_{i=i}^{n} \frac{(x_i - \bar{X})(y_i - \bar{Y})}{n-1}$$

The value n represents the number of rows in the training data. The value x represents each vector of predictors and y represents the expected value. The Pearson product-moment correlation coefficient is given by the following equation (which makes use of the previous equation):

$$R_{ij} = \frac{C_{ij}}{\sqrt{C_{ii} \cdot C_{jj}}}$$

The resulting value (R) gives the correlation between any of the inputs (i) and the target (j). The absolute value of R indicates how strongly correlated the input is to the target. Higher values are more strongly correlated. We provide a Python implementation of the correlation-coefficient, feature-importance-ranking algorithm that can be used with any Scikit-Learn model.[2]

The input perturbation algorithm[3] is a more complex agnostic, feature-importance algorithm that calculates the loss of a model when each of the input features to the neural network is perturbed by the algorithm. The idea is that when an important input is perturbed the neural network should have a considerable increase in error, that corresponds to the importance of that input. Because the inputs are being perturbed, rather than removed entirely, it is not necessary to train a new neural network for each evaluated feature. Rather, the feature is perturbed in the provided data set. The feature is perturbed in such a way that it provides little or no value to the neural network, yet the neural network retains an input neuron for that feature. No change is made to the neural network as each input is evaluated.

To effectively use feature-perturbation ranking it is necessary to evaluate the loss (E) of a model. If the model is regression, the following equation evaluates the loss between the expected output (y) and the model output ($\hat{y}$) over n data items:

$$E = \frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{n}$$

If there are multiple outputs, they are simply considered as additional y and $\hat{y}$ values. If the neural network is classification, then a multi-logloss evaluate is performed:

$$E = -\frac{1}{n} \sum_{i=1}^{n} (y_i \log(\hat{y}_i) \cdot (1 - y_i) \log(1 - \hat{y}_i))$$

To successfully perturb a feature for the input-perturbation, feature-importance algorithm two objectives must be met. First, the input feature must be perturbed to the point that it now provides little or no predictive power to the neural network. Secondly, the input feature must be perturbed in such a way that it does not have adverse effects on the neural network beyond the feature being perturbed. Both objectives are accomplished by shuffling, or perturbing, the column that is to be evaluated. By shuffling the column, the wrong input values will be presented for each of the expected targets. Secondly, the shuffle ensures that most statistical measures of the column remain the same, as the column will maintain the same distribution.

To effectively use feature-perturbation ranking it is necessary to evaluate the loss *(E)* of a model.

Feature importance is usually reported as a table that shows the name of each feature, its relative importance, and the error that the model reported when that feature was perturbed. For example, Table 1 might represent the importance of four features:

Table 1
Sample Feature Importance Ranking

Feature Name	Importance	Loss
D	1	5
B	0.6	3
A	0.4	2
C	0.1	0.5

The higher the loss, the more important a feature is. The perturbation effectively removes the feature from the prediction. Removing an important feature will result in a higher loss than

removing a less important feature. Each feature has an importance that is reported as the value of that feature's loss divided by the highest loss. Because of this, the most important feature will always have an importance of 1. The importance values will not sum to 1.0. Rather, the importance values show the relative importance of each feature to the most important feature. We provide a Python implementation of the perturbation-ranking algorithm that can be used with any Scikit-Learn model.[4]

MULTIVARIATE FEATURE RANKING

It is also possible to use the perturbation feature-ranking algorithm to evaluate multivariate features. It is possible that two features are more important together than they are separately. To evaluate this, a pair-wise feature importance could be generated for each of the possible pairs of features, similar to how a covariance matrix is often calculated to determine which feature pairs are strongly correlated to each other.

The generation of a pair-wise multivariate feature importance report is produced similarly to the univariate-perturbation, feature-ranking algorithm presented in the previous section. The primary difference is that two columns will be perturbed at a time, rather than a single column. To perform this, it will be necessary to loop over every combination of features taken two at a time. For example, 10 features result in 45 evaluations. This is because 10 items, taken two at a time, yield 45 combinations.

Visually, this can be thought of as a pair-wise matrix. The diagonal is discarded, because that would consider each feature with itself. Likewise, the upper or lower triangle of the matrix can be discarded because the pair-wise importance of feature-1 and feature-2 is the same as the pair-wise importance of feature-2 and feature-1. Considering triplets, quadruplets and higher multiples would considerably increase the amount of processing that would be necessary.

SUMMARY

Feature-importance ranking is a very important consideration for data science. It can be used to optimize your data set and remove unimportant features to improve the performance of your model. This decreases the computation time needed for your model and often increases the accuracy. Feature engineering also benefits greatly from feature importance evaluation. As additional features are engineered, they can be evaluated to see their relative importance to the model. When using feature importance in conjunction with feature engineering, it is important to remember that the perturbation-ranking algorithm will typically share the importance between two closely correlated features. Because engineered features are mathematical combinations and transformations of the original feature set, the engineered features are usually strongly correlated to the original feature set. Therefore, it is important to keep in mind that the engineered features are usually sharing importance with the original features from which they were constructed. ■

Jeff Heaton, Ph.D., is lead data scientist, Reinsurance Group of America, in Chesterfield, Mo. He can be reached at *JHeaton@rgare.com*.

ENDNOTES

1 Goh, Anthony T. C. 1995. Back-Propagation Neural Networks for Modeling Complex Systems. *Artificial Intelligence in Engineering* 9, no. 3:143–151.

2 Heaton, Jeff, Steven McElwee, and James Cannady. Early Stabilizing Feature Importance for TensorFlow Deep Neural Networks. May 2017. In *International Joint Conference on Neural Networks (IJCNN 2017)*. IEEE.

3 Olden, Julian D., Michael K. Joy, and Russell G. Death. 2004. An Accurate Comparison of Methods for Quantifying Variable Importance in Artificial Neural Networks Using Simulated Data. *Ecological Modelling* 178, no. 3-4:389–397.

4 *Supra*, note 2.

Shiny: Another Step Forward in Data Democratization

By Eileen S. Burns

RSTUDIO AND SHINY

Years ago, RStudio gave the R user community a better way to interact with R than the R graphical user interface (GUI). There were and are other GUIs available, but RStudio combines so many features that it's been a no-brainer for our team and countless others: project files, visual folder structures, seamless connection to CRAN repositories, easy update of R and package versions, integration with Markdown and Sweave, a customizable interface including plot and help features, history and environment maps, and seamless integration with Git and GitHub.

Through that easy-to-use interface, with a little writing of code, RStudio let us connect with our data in a transparent way—pull it in, analyze it, visualize it, fit models to it, make predictions, validate our models, save our scripts, and easily document which scripts were important, what they did, and what the results were.

In a nutshell, it let us be *real* data scientists, with an emphasis on enabling that gold standard: reproducible research.

With all that RStudio has let us streamline, it still requires that users know how to code in order to learn from data.

In comes Shiny, the next step in democratization of data science from the makers of RStudio.

What is Shiny? It is an R package described as "a web application framework for R." In short, Shiny lets those of us who know how to code create user-friendly interfaces to share with our friends who would rather not. It comes with some samples to get you started, and as you would expect for an R package, and anything from RStudio, it has a great user community for support and ideas.

OUR FIRST PAF SECTION WEB APPLICATION

Rather than simply tell you how great Shiny is (I promise I'm not being paid by RStudio for this), I wanted to show you. I'm using as a guinea pig the newly minted *Predictive Analytics and Futurism (PAF) Newsletter* Catalogue. The catalogue is here *https://www.soa.org/sections/pred-analytics-futurism/index-of-paf-articles.xlsx*. I confess I did some non-reproducible editing of the list in order to make it more fun to play with, so this will not update seamlessly as new articles are added to the index. I'll save that for a future iteration. I have also supplemented the catalogue with some metadata pulled for more recent articles by the Society of Actuaries (SOA) staff.

I created a Shiny app to visualize what our section members have been writing about for the past nine years. I built in three features—a histogram to display frequencies by author, newsletter edition, topic, etc.; a word cloud to visualize the relative frequency of various keywords; and a table to allow sorting, filtering to a single author or edition, and searching by text string.

Then, to give my application its first important use, I used the features to make sure I wasn't writing a repeat of a prior *PAF Newsletter* article.

Using the word cloud (Figure 1), I got a handle on what the biggest topics have been. Visualization tools certainly haven't made a huge splash to date.

Using the search function on the data table (Figures 2 and 3), I can see there have been no mentions of Shiny or Git (with this meaning anyway). There are only four mentions of "visual" and eight of "language." I've put it on my to-do list to go back to those four articles that mention visuals!

Figure 1
Word Cloud of *PAF* Article Keywords

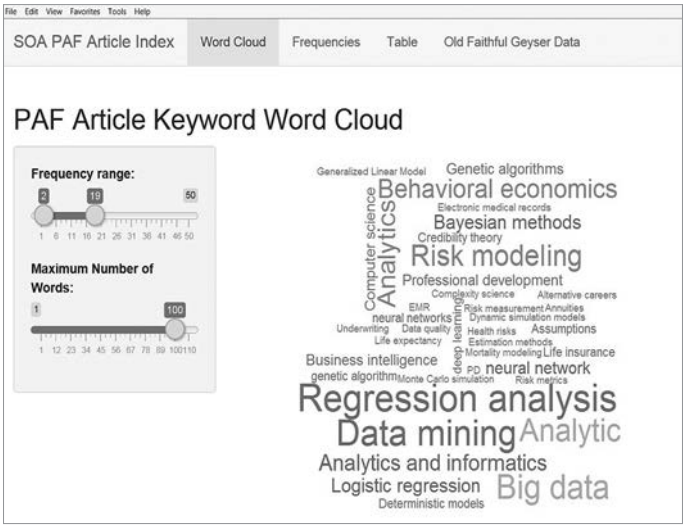
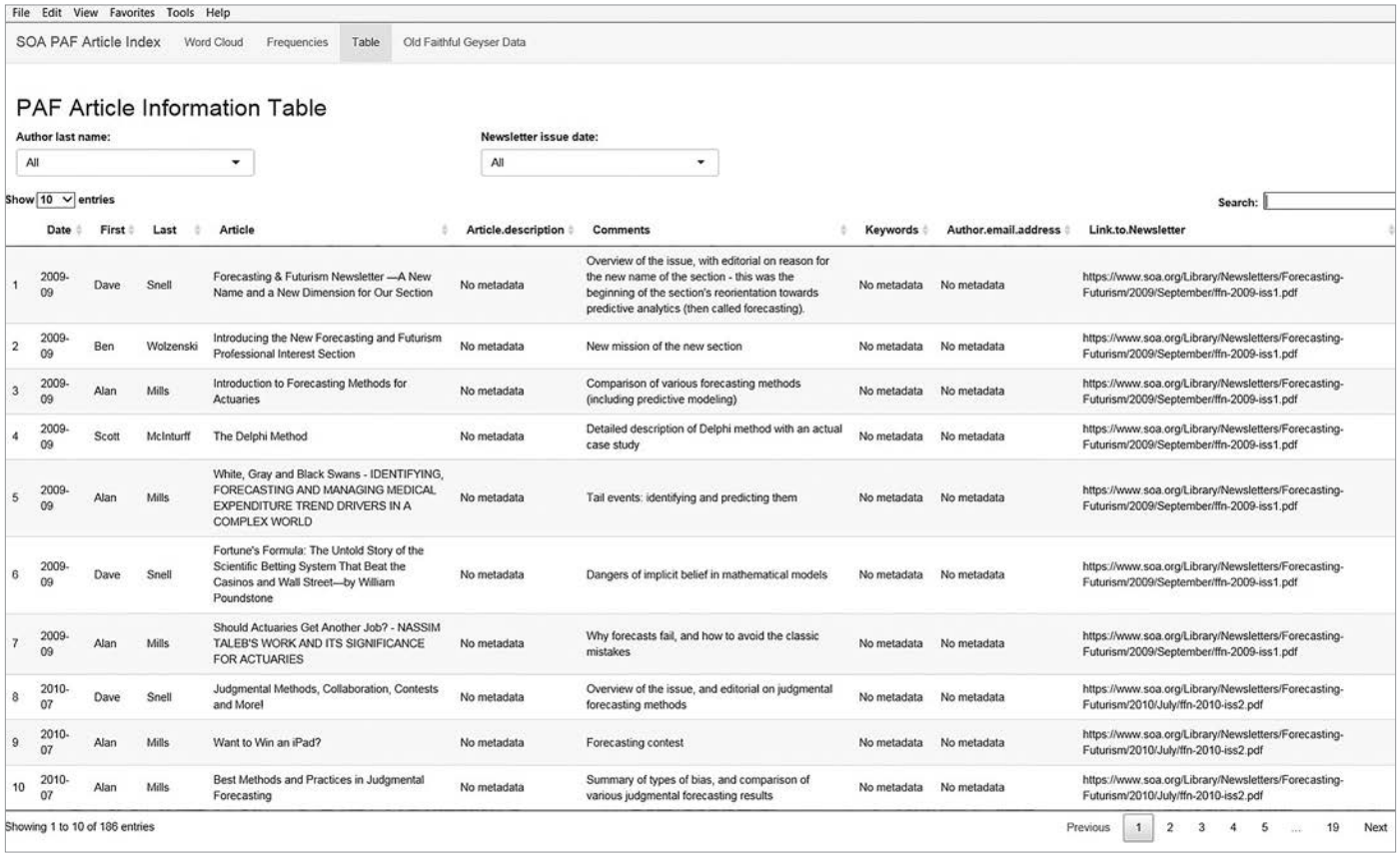


Figure 2
Table of *PAF* Articles



File Edit View Favorites Tools Help

SOA PAF Article Index | Word Cloud | Frequencies | Table | Old Faithful Geyser Data

PAF Article Information Table

Author last name: All

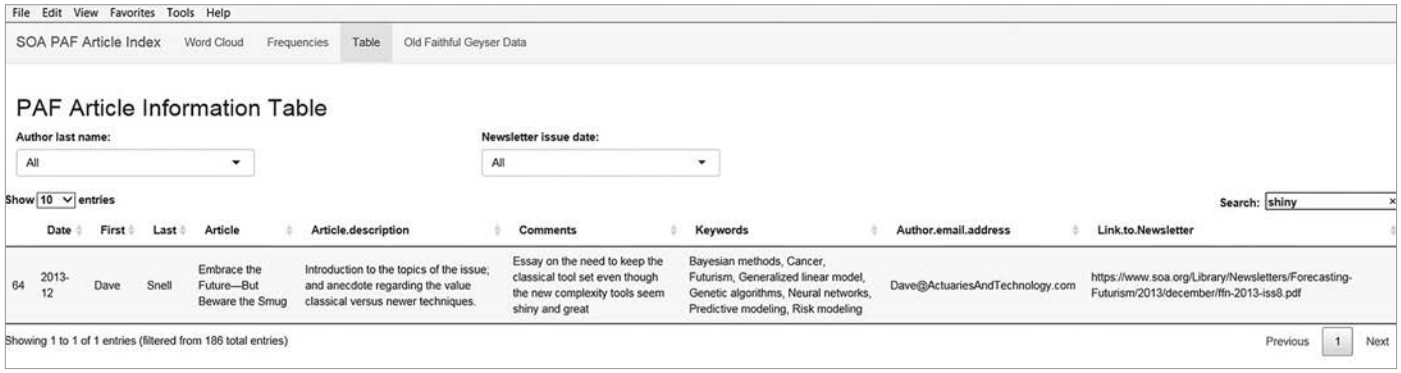
Newsletter issue date: All

Show 10 entries | Search:

	Date	First	Last	Article	Article.description	Comments	Keywords	Author.email.address	Link.to.Newsletter
1	2009-09	Dave	Snell	Forecasting & Futurism Newsletter —A New Name and a New Dimension for Our Section	No metadata	Overview of the issue, with editorial on reason for the new name of the section - this was the beginning of the section's reorientation towards predictive analytics (then called forecasting).	No metadata	No metadata	https://www.soa.org/Library/Newsletters/Forecasting-Futurism/2009/September/ffn-2009-iss1.pdf
2	2009-09	Ben	Wolzenski	Introducing the New Forecasting and Futurism Professional Interest Section	No metadata	New mission of the new section	No metadata	No metadata	https://www.soa.org/Library/Newsletters/Forecasting-Futurism/2009/September/ffn-2009-iss1.pdf
3	2009-09	Alan	Mills	Introduction to Forecasting Methods for Actuaries	No metadata	Comparison of various forecasting methods (including predictive modeling)	No metadata	No metadata	https://www.soa.org/Library/Newsletters/Forecasting-Futurism/2009/September/ffn-2009-iss1.pdf
4	2009-09	Scott	McInturff	The Delphi Method	No metadata	Detailed description of Delphi method with an actual case study	No metadata	No metadata	https://www.soa.org/Library/Newsletters/Forecasting-Futurism/2009/September/ffn-2009-iss1.pdf
5	2009-09	Alan	Mills	White, Gray and Black Swans - IDENTIFYING, FORECASTING AND MANAGING MEDICAL EXPENDITURE TREND DRIVERS IN A COMPLEX WORLD	No metadata	Tail events: identifying and predicting them	No metadata	No metadata	https://www.soa.org/Library/Newsletters/Forecasting-Futurism/2009/September/ffn-2009-iss1.pdf
6	2009-09	Dave	Snell	Fortune's Formula: The Untold Story of the Scientific Betting System That Beat the Casinos and Wall Street—by William Poundstone	No metadata	Dangers of implicit belief in mathematical models	No metadata	No metadata	https://www.soa.org/Library/Newsletters/Forecasting-Futurism/2009/September/ffn-2009-iss1.pdf
7	2009-09	Alan	Mills	Should Actuaries Get Another Job? - NASSIM TALEB'S WORK AND ITS SIGNIFICANCE FOR ACTUARIES	No metadata	Why forecasts fail, and how to avoid the classic mistakes	No metadata	No metadata	https://www.soa.org/Library/Newsletters/Forecasting-Futurism/2009/September/ffn-2009-iss1.pdf
8	2010-07	Dave	Snell	Judgmental Methods, Collaboration, Contests and More!	No metadata	Overview of the issue, and editorial on judgmental forecasting methods	No metadata	No metadata	https://www.soa.org/Library/Newsletters/Forecasting-Futurism/2010/July/ffn-2010-iss2.pdf
9	2010-07	Alan	Mills	Want to Win an iPad?	No metadata	Forecasting contest	No metadata	No metadata	https://www.soa.org/Library/Newsletters/Forecasting-Futurism/2010/July/ffn-2010-iss2.pdf
10	2010-07	Alan	Mills	Best Methods and Practices in Judgmental Forecasting	No metadata	Summary of types of bias, and comparison of various judgmental forecasting results	No metadata	No metadata	https://www.soa.org/Library/Newsletters/Forecasting-Futurism/2010/July/ffn-2010-iss2.pdf

Showing 1 to 10 of 186 entries | Previous 1 2 3 4 5 ... 19 Next

Figure 3
Table of *PAF* Articles Filtered to Show One Article with "Shiny" Mentioned in Comments



File Edit View Favorites Tools Help

SOA PAF Article Index | Word Cloud | Frequencies | Table | Old Faithful Geyser Data

PAF Article Information Table

Author last name: All

Newsletter issue date: All

Show 10 entries | Search: shiny

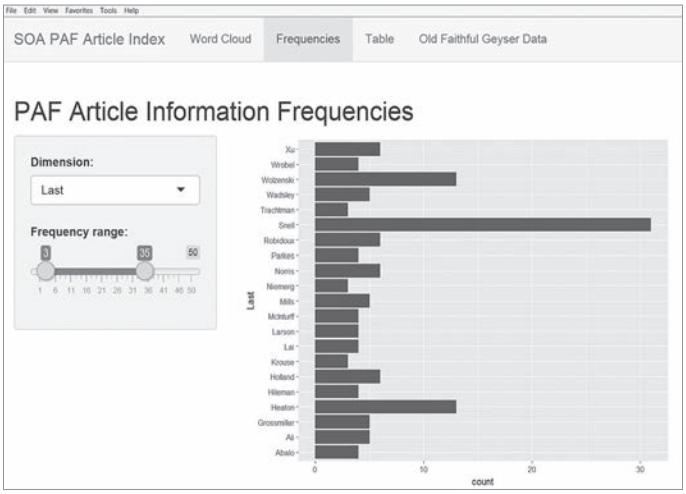
	Date	First	Last	Article	Article.description	Comments	Keywords	Author.email.address	Link.to.Newsletter
64	2013-12	Dave	Snell	Embrace the Future—But Beware the Smug	Introduction to the topics of the issue; and anecdote regarding the value classical versus newer techniques.	Essay on the need to keep the classical tool set even though the new complexity tools seem shiny and great	Bayesian methods, Cancer, Futurism, Generalized linear model, Genetic algorithms, Neural networks, Predictive modeling, Risk modeling	Dave@ActuariesAndTechnology.com	https://www.soa.org/Library/Newsletters/Forecasting-Futurism/2013/december/ffn-2013-iss8.pdf

Showing 1 to 1 of 1 entries (filtered from 186 total entries) | Previous 1 Next

The bar plot (Figure 4, pg. 27) confirmed that Dave Snell is by far our most prolific author; the table can help me discover if he has a particular focus outside of his contributions as our editor.

Could I have found all of these answers in Excel? Absolutely. Ultimately, that's where the information came from. I love Excel, but going forward, I won't be using it to get those answers. With this app I'll be able to more quickly visualize what's been written about, when, by whom, and what's ready for some more attention. I can share the app with my team so they can brainstorm what they'd like to add into the *PAF* dialogue. If I publish the app to an internal server, I can share it with more senior folks who don't

Figure 4
Bar Plot Displaying Distribution of Authorship



know the first thing about R, and they can use it to make suggestions. All of us can use it to identify past articles on topics relevant to our jobs. We've democratized the data on the *PAF Newsletter*.

HOW TO USE THIS APP AT HOME

With just a few steps you can be up and running in Shiny:

1. Install RStudio if you haven't already.
2. Install the Shiny package in RStudio.
3. Create a new project under File -> new Project, and select Shiny Web Application.
    a. You may or may not choose to create a Git repository.
4. Click Run App.

What you'll see is a simple interactive application based on Old Faithful geyser data.

If you want to go a few steps further and run *this* Shiny app, you can find it on Milliman's public GitHub account here: *https://github.com/milliman/SOA_PAF_Section_Newsletter_Catalogue*. The repository contains six key files (plus the standard README.md, LICENSE.txt, and .gitignore):

1. **Keywords.csv.** A list of the keywords referenced in the metadata for the more recent articles
2. **PAFCatalogueComplete.csv.** An augmented table based on the PAF catalogue referenced above
3. **loaddata.R.** An R script that loads the keywords and the article catalogue
4. **server.R.** Code for doing analysis and returning a figure or table
5. **ui.R.** Code for structuring the user interface
6. **SOA_PAF_Section_Newsletter_Catalogue.Rproj.** The R project file that holds it all together

You'll notice this app contains the same Old Faithful geyser feature as the default Shiny app. I kept it in the app to show how easily you can switch to a layout that has a navigation bar to flip between multiple features.

NEXT STEPS

While newsletter data makes for a useful jumping-off point, there are clearly more compelling applications for actuaries with access to large data sets and related business questions. It helps to start with a question and an idea for what data visualizations will help you answer it, but you don't have to come up with all of the ideas yourself.

Shiny.rstudio.com/gallery is a good place to go to see what other users are doing with Shiny. It pointed me to the word cloud as a good option for immediately seeing frequently addressed topics. It can give you some great ideas for graphs, maps, tables, dynamic input options and layouts.

If you get really into it and want to share your work, *Shinyapps.io* is there for you. For a richer experience, I recommend engaging with the broader R user community. I recently attended an R user group meeting in Seattle dedicated solely to sharing web applications built with Shiny. My team has been building Shiny apps for years, and I still came away with new ideas.

There is nearly no end to how sophisticated you can go, or how many data-based insights you can offer those using your applications. ■

Eileen Burns, FSA, MAAA, is a consulting actuary with Milliman. She can be contacted at *eileen.burns@milliman.com*.

SOCIETY OF ACTUARIES®

475 N. Martingale Road, Suite 600
Schaumburg, Illinois 60173
p: 847.706.3500 f: 847.706.3599
w: www.soa.org

NONPROFIT
ORGANIZATION
U.S. POSTAGE
PAID
SAINT JOSEPH, MI
PERMIT NO. 263