Andrew Redd: Thank you. I am really happy to be here. I’m talking about diagnosing models and code in R, so long-running models, big models, big data stuff. Of the questions that I get as a help desk R expert, this is one big category is, "Why isn't my code running; why is it taking so long; what can I do? Help, I was running a model for two days and then it got killed by VINCI, what can I do?"

So, this is going to be a discussion about how I would go about diagnosing a model. I don't claim to know everything--and I certainly won't be going over everything in here. This is kind of where our statistics and a little bit of the artistry of statistics intersect. So, data science is somewhat science and there's somewhat of an art to it, and this is where we're going to be talking about some of the artistry that goes beyond just the raw statistics, what we think happens theoretically doesn't necessarily happen when we put code into the program.

So, a couple of notes about my code: if you are not familiar with tidyverse, I apologize, you should familiarize yourself with tidyverse. My code is almost always in tidyverse style, which means that I am assuming that you have dplyr installed, which is for all of the data manipulations--filter, selecting variables, manipulations such as mutate or transmute. For map reduce operations and magrittr for the pipe; if you're not familiar with the pipe, the pipe is a simple convention that says, "Pass the result of the left-hand side into the right-hand side; so, pass into that." And so, it makes it so that your code is more readable and you can chain operations together in a way that is much more human-readable than endlessly-nested parentheses. If you haven't familiarized yourself with tidyverse, I highly recommend that.

Other than that, if there are other packages that I am using that are not typically loaded with tidyverse, I will try to use the double-colon scoping operator to indicate that I am using a function from a specific package.

Okay. So, let's start out with a quick question. I’d like to know what your experience is with r. So, I need to know are you an expert and you could be giving this presentation yourself or are you a practitioner, which would mean R is what you usually use for your analysis? You're comfortable with most functions of r, but would not consider yourself an expert. A journeyman of comfortable with most things in, but maybe you use R and SAS or and theta or R and theta, or are you kind of just getting your feet wet with R and you kind of know a couple things, maybe how to do graphics, but you're not really comfortable with much of the other stuff; or are you a complete noob, to use gamer language, and you know basically what R is, but know nothing more beyond that.

Moderator: Alright. Well, that poll has been open since you introduced it and answers are streaming in. It looks like things are leveling off, so I’m just going to give people just a few more seconds before I close the poll.

Andrew Redd: Okay. While people are finishing their answers, I will go over an example. This is a recent project that I have been working on, and so it's kind of a work in progress, but I was instructed to go through because there were a lot of things that I found that will be helpful for people to see how--and it is a real-world example of a big model that is complicated, that has issues with it. So, we are looking at Choice data from--or data from the Choice Act, and how veterans choose different hospitals. And we're specifically looking, in this model, as modeling death after discharge from hospitals; we have a little over a million records which recorded almost 22,000 deaths; we're trying to fit this with a general linear mixed-effects model. So, we have random effects, we have fixed effects; it's complicated, we have to fit it with restricted maximum likelihood, so there's going to be iterations, so it's going to take a long time.

Okay. So, the results are...?

Moderator: Yes, only 5 percent said that they are experts; 32 percent, which is the largest number, consider themselves practitioners; 16 percent consider themselves journeymen; 27 percent consider themselves learners; and 16 percent consider themselves noobs.

Andrew Redd: Alright. I think I’ve got a good crowd here. Hopefully, what I can present is going to be useful for people. There are some things that I will mention that I won't actually show, but I would encourage you to go and look at those topics if you are not familiar with them, especially for the learner group there, because there are some very useful things that I will mention but I don't have time to cover in this presentation.

So, let's go to a second poll--and this is the last poll, I’m not going to just be doing polls every other slide. But I would like to know what a typical data set is to you. In the VA, we have wonderful--especially in VINCI, we have access to enormous amounts of data; and so, we could be doing billions of records. I’d like to know what kind of sample size or what kind of a data set is the typical sample size for you: 10s, 100s, 1000s, tens of thousands? I typically work in the millions--hundreds of thousands to the millions is most of what the work I do for actual studies is, but I have done everything from single digits all the way through... I don't think I’ve ever got billions, but hopefully, someday, I will get a data set that large that I can analyze, and that will pose some interesting challenges other than the things that I will go over today. Most of what I will be doing today will be millions or below, probably in the tens of thousands or hundreds of thousands is where we will mostly be spending our time.

Moderator: I just closed the poll and I can read the results off if you'd like.

Andrew Redd: Yes.

Moderator: Nobody answered A, less than ten; only 5 percent answered B, 10s; 16 percent answered C for thousands; another 16 percent answered D for tens of thousands; 21 percent answered E for hundreds of thousands; 24 percent; and zero for billions and more. So, it's petty steady across tens of thousands, hundreds, and millions.

Andrew Redd: So, it looks like we've got a good crowd here; so, mostly between tens of thousands and millions. So, that’s exactly the crowd that this kind of presentation is geared for, so that's good.

Okay. So, the first thing, when you are asking the question of, "Why is my code running so long?" The first question is I want to know how long it takes. The simplest answer is to use the system.time function, and what this simply does is it takes a time measurement and runs the code, does another time measurement, takes the difference between those two, and gives you the answer. This is the kind of code that we're going to be running, so we're saving the system time results to run time and inside of that call there's the model fit. We're taking our data, I am sampling it down to 100,000 to make that a reasonable amount. So, we know that a million is going to take longer than we have. So, we want to sample it down to something that is reasonable.

Now, in statistics, it makes sense: if you're going to sample a population the results are should generalize to the whole population from the sample to the whole population. The same thing here, it's we are writing a model and if there are problems with the model in the sample of 100,000, then there's going to be problems in the model with a million. So, don't think that you have to always look at your whole data set; sampling down to bite-size chunks is totally reasonable and appropriate to do. We are doing glmer; this is usually pronounced, "glmer", GLMER, Generalized Linear Mixed Effects Random model; I am not showing you the formula because it's long, but there are several variables in it.

So, what comes out when you use system.time? You have a user, a system, and then elapsed time. But the first column in that is the user time; this is how much time the actual computations took; system time is the time of system operations such as disk input, output, memory, swaps, that kind of stuff; and then elapsed time it the total clock time--the wall time--that it should have taken. That doesn't always reflect the actual wall time if there's process-swapping and stuff; but essentially, if you're running it on your computer, that's how long passed on the wall.

There are two other values on here that are not present and they're not usually shown, but they will be shown if, for some reason, you are using a function that will spawn child processes. So, parallel processes and stuff like that will be counted, can be counted in the system time; and so you would see numbers in there; but for simple sequential operations which is what we normally run in R, you're going to see just these three numbers. I will talk a little bit about parallel processing in the end because that's one of the ways that we get around this kind of long code operation.

So, a useful trick for this is actually to use the [lubridate] package. And so, I’m taking run time, I’m dropping the class of proc time off of it, and then I’m saying this is actually a duration in seconds; and what I get out is an easy conversion to hours, minutes seconds, or whatever. And so, we see that we're having 31 minutes of the four hours in memory; that's a little bit concerning.

If you have a lot of system input/output, then you're doing something like you are writing to disk all the time, or you're reading from disk all the time, or you're writing or reading repeatedly to a database. Those are the kind of things that can really kill a project, really kill runtime. Another thing to do is if you're or memory, if there's a lot of memory allocation, you'll see that system time just explode; and if you're taking a lot of system time, you're doing something wrong; it should be on all of the execution time.

This looks fairly reasonable--I’m a little concerned about the system time, but not overly concerned. There's a better way to get what your program is doing and that is profiling. And I find that a lot of people, even those who consider themselves R experts, don't realize about profiling. The other things that would be great to use are debugging and traceback functions; especially when you have code that is throwing errors, you want to use that debug function, and/or trace functions--the trace functions where you can inject code into a function--to get some useful information, but that's beyond the scope of what I’m going to cover today.

I think that profiling is totally underrated and is useful both for figuring out where any code is, so it's useful for figuring out why any code runs so long, not just necessarily models, but it will give us a little bit of insight into models. But this is kind of one of my first things to check to say, "Okay, is this doing what I think it's supposed to be doing?"

So, if you are running R studio--which if you are not running in R through R studio, I may question your sanity, but that’s between you and R, some are purists which I won't resent--but I highly recommend using R studio; R studio is on VINCI; there is a menu specifically for profiling. You'll go to profile and you'll go to either Start Profiling or Profile Selected Lines. This is my go-to is Profile with Selected Lines; you highlight a section of code, usually, the code that's actually giving you problems, and you say "Profile this."

Now, what profiling does is it runs code. You know it's running when you see in the console, this clock icon next to the Interrupt or kind of the soft button, or the Stop Profiling button. And you'll know that it's running, you'll let it run for a while--in this case, I let it run for like four hours and got the results. Once it's done, it will display a report. Now, this report is--what it's doing in the background is every like two-hundredths of a second, it's taking a snapshot of the code and saying, "Where is this going? What functions is it running? What's the stack? So, what function called which function, which called which function?", those kinds of things.

Then you get a report. The first page of that report is called a flame graph; the flame graph shows the function calls as horizontal bars. So, the first thing that we called is system time; then we are in a pipe, and then we are in glmer, so that's what we're actually wanting to pay attention to is everything that's above glmer. So, the first calls are on the bottom, and the deeper you go, the higher up you get; and the width of the bar is the time that was spent in that function.

So, we see glmer calls optimizeGlmer, called updraft, called Derivative 12--deriv12--which calls some anonymous function that then does a whole bunch of stuff inside of that. We can't--or we probably shouldn't be messing with anything that comes in from the optimize--and anything below glmer, we shouldn't really be messing with but it's good to know what's happening. And we're spending a lot of time here in computing those derivatives, which gives me information as a statistician to know where I should be looking to figure out problems in the model. Problems in the model will come from deficiencies in the model, correlations in the variables, or boundary problems where it's estimating right next to a boundary of zero or something like that.

The next page is the data table. So, you'll see two tabs; one will be the flame graph, one would be the data table. The data table takes that flame graph and summarizes it across all of them, and you'll see how much time was spent in each of those functions and how much memory was used for those functions. When you're coming into memory problems, this is what you want to look at; if you're coming into execution time, you'll want to pay attention to the time column. You can see that there are some pulling functions or some functions that are nested inside a list that are doing some work here. But this function--evaluating this function repeatedly is where we are getting stuck. So, if we can optimize that--or optimize how to get the derivative--it'll be better.

So, in this case, we get a degenerate Hessian or a not-full-rank, which means there are extraneous variables in the formula. This can happen when you have variables that are essentially the same and/or giving the same information and it's been presented in different ways. So, like if you included a variable and a transformed version of that variable so like you rescaled that variable, those would be considered redundancies in the matrix.

To check this, let's look at a couple of problems. So, we're just going to take our model fit, pull out the model frame which is the data that was actually used to fit the model, and summarize that; there's a lot of other variables that are listed here that I have abbreviated to show a couple of problem cases. So, one of the things that we have in here is a Station ID, STA6A. We also have in here an ED indicator where almost all of them have ED departments. We have bed\_count, we have academic affiliation where almost all of them are academically-affiliated; we have rurality that is almost never true. So, these are some problems that have some sparsity issues, particularly ED and academic affiliation are going to be identifiable from the station id; and so, these are actually problems that create redundancies in our matrix that are giving us problems, being problems fitting the data.

One thing I like to do is I like to look at the correlation matrix. I examine the correlation matrix reveal where there are deficiencies, so highly correlated variables should be of concern. This is one way I do it: I take the model fit, pull out the model matrix which is that model frame that has been converted into a matrix of 1s and 0s, so all the factor variables have been separated out into individual columns. Then I’m just looking at the correlation, throwing up an image of that so the darker places are going to be higher; lighter places are going to be lower. This diagonal line is our--it's the diagonal of the matrix--and I’m only showing part of it to show some of these, we only are worried about one side of the matrix or the other. And you can see there's a couple of problem variables here that are very highly correlated; I pointed them out with the purple arrows.

So, I want to actually enumerate what those highest correlations are. There's not a really great way to do this; there's a lot of different ways to do it and none of them are really--there's nothing really built-in, this is the way that I do it. So, I just take the correlation matrix, convert it to a data frame, then I reshape it. So, I pivot it longer, and then I just filter out the ones that I don't want to see, I don't want duplicates, and then I arrange it by the descending values.

We can see that our rurality variable is highly correlated with our drive time; so, this is drive time from a patient's address to the facility that they visited their; their race and ethnicity, those are highly correlated which should be concerning; the age and our [Venmo Raven] Elixhauser score highly correlated--which we kind of expect and these things. So, these are some of the concerns.

So, what we're going to do--another way to look at this is variance inflation factors, and this is something that many people may not be familiar with, is a variance inflation factor is a way to measure if a variable--if there's correlation in the design matrix of a model, then the variance of the estimates in that model could be really, really high. And the way to measure this is to look at the generalized variance inflation factor, and then divide by or raise it to the power of one over the two times the degrees of freedom. The bigger that number, the bigger concern that we have. Here we see that our emergency department estimation, which we've already said is redundant in the matrix, is causing a lot of problems, and that is a ridiculously high number; anything really over like four or five, as a rule of thumb, is bad in this inflation factor. So, both these ED and bed count are problematic; and the reason they're problematic, as we already discovered, was that we've included stations that identify those variables.

So, let me back up here. The reason that these cause problems when we're fitting especially when we're doing something like a generalized linear mix effects model, is that when you have multiple parameters that represent the same thing, you can just trans--you can raise one and shrink the other, or shrink one and raise the other, and then you can just transfer between those variables; and you end up never converging the model. That's one of the reasons why you're worried about correlation is because you can just--they're just trade-offs that you keep bouncing back and forth between and the model number converges because you keep changing more than should be. But what it looks like in the geographic space is you're just sitting on a line or a plane that's on top of a plateau of maximum likelihood, and you're just moving around in that plane and you don't ever settle on a position because they're all equally the same.

Okay. One other problem--and this may be specific to R, I’m not sure. I know that some machine learning tools will automatically handle this, and that is to standardize variables. Fitting algorithms, particularly the one with generalized linear models, does not automatically standardize variables--and it does not like that; it does not like variables that are orders of magnitude different than each other, because when they compute that correlation matrix and compute the eigenvalues, this one is just going to be enormous. Typically, it will actually let you know this if it comes in across a situation like and say that you should standardize the variables; but it's a good idea to do it anyway. So, one of the things that we're going to fix is we're going to fix our age.

Age, we're going to standardize it from a scale from like 20 to like 90, to a scale of zero to 1. And to show the difference that this makes is just a model with age and an intercept, drops from 83 seconds down to 15 seconds to converge with everything else being the same; just by changing the scale of that variable, the model was able to converge much better. Now, I know that it's coming in the future that you will be able to automatically standardize your variables--or that will be an option in the controls; as of right now, that's implemented in the version that is on VINCI. I don't know if it's been fixed on [the program], but it is a good idea to standardize your variables beforehand.

Other things that you may want to want to do, other transformations. Some of the other transformations that I performed in fitting this model that I didn't show in the slides. I took some blog post forms, so I took a variable that was from like 0 to 300, and scaled it down 0 to 6. And I took some percentages that were 1 to 100 and scaled those down to between 0 and 1; and those all help the fitting algorithm work better.

So, another thing that a lot of people aren't familiar with is marginal model plots. This actually comes from the CIR mode--and I apologize, this is PowerPoint, the "cir" in that "cir::mmps" should not be capitalized; it's a categorical and applied regression package. The idea here is that in a marginal model plot, there are two lines: one is estimated--one is a smooth estimated directly from the data; the other a smooth estimated from the fitted values; if the model is specified correctly, the two lines should be very close. In this case, they're really not; so, we can say that there is some level of misclassification here--and I think this is likely because of a log-transform, which is what I did to fix this variable. And so, we see that it's a heavy-tailed--if we log-transform it, it can fix this.

So, we've kind of gone from gone now from long-running models to diagnosing like, "Why is my model missing?" And missed specifications and model will also make it longer--I mean, if you've got the right model, it'll just go right to it.

Okay. So, summarizing what we have discussed. The first thing that I didn't really mention, but I think should be discussed--or, at least, thought out--is to evaluate if it's worthwhile. Your time as a person, as a researcher in the VA is expensive; computational time is cheap and if you've got time to wait, oftentimes, waiting is a decent solution. So, first thing, evaluate if it's worthwhile. Spending a week of work to save a few hours of computation time is very rarely worthwhile.

With that caveat, if you've got to figure out what the problem is, so say it's not converging, figure out why. Is it spending too much time in the memory; is it spending too much time extracting things from the data set? So, run a profiling file to figure out what's going on--and that works with all code, not just models. Use traceback, use debug, figure out where the errors are coming from; another helpful tool is to sequentially add or delete variables from a model, specifically for the runtime because there are usually specific variables that are causing problems with your model. Figure out if the model is correct or if adjustments such as transformations, such as scaling variables will improve the run time or the fit. Then if all else fails, then you throw more money at it.

One thing that I do not cover, but I think that I should cover here is that in R on VINCI--well, not in R, but in VINCI, we have access to the SAS grid; and SAS can, itself, run R code. So, one of the ways that people can avoid getting through the issues of jobs being killed and shut down is to run your code in batch on the SAS grid. If you do need instructions for that, there are instructions on VINCIPedia; I am not an expert on that portion of it--Mark [Esnow] who is the SAS expert for VINCI is far more familiar with running things on SAS grid. So, I won't cover his area of expertise, but let people know that that is one of the possibilities of of being able to run longer-running code is to run it through SAS so that it goes onto the SAS grid; so, you're on a separate machine than what is usually running our code, and that will allow you to run longer jobs or run it in a dedicated space.

So, that being said, even on the SAS grid, you may want to run things in parallel. So, either you're running--maybe you're running multiple models or you are running multiple models on subsets of data, there are two big paradigms for parallelism in R.  Yeah, I will go through this and then I’ll answer your questions if there are any questions. If you have questions, though, please put them in the Q&A and I’m happy to answer questions. So, I’m going to talk a little bit about parallelism and then we will go to questions.

So, parallel. So, by throwing more money, I mean computational resources: the two major approaches are 4-H which is explicit parallelism, and then there is a new approach that's actually a couple of years old now, but it's called futures which is an implicit parallelization. So, let's talk about 4-H. So, 4-H, you have to set up a cluster of workers; this doesn't work very well in VINCI because we have two servers and that doesn't--you're not setting up a whole network of things. But what you can do with 4-H on VINCI is to spawn multiple R processes, and then you have special functions that will delegate computing each iteration of a function or of a loop on each of the different nodes or different processes.

The main package that you're going to want to look at for this is 4-H, it was made by Revolution Analytics before that was bought by Microsoft. But the newer approach is Future or Futures, which is supported by our studio and their development. This is a different way of thinking about parallelism; basically, it is achieved through registering a plan or a strategy; and this really works well because you can set multiple levels--again, here, this is not totally applicable to VINCI, but we hope--I hope to get some things working with like the fast grid and stuff that maybe we'll improve this sequential would be a strategy that is the default, so you're just running things one after another, after another.

Transparent--I’m not sure how that differs from sequential, but it is for some reason. Multi-session is multiple R sessions on the host computer; this is what works typically with Windows and, as such, would work on VINCI. Multi-core is, for processes, it will not work on Windows but should work on SAS grid because I understand the subscript is Unix-based; or if you have a development machine on VINCI that has Linux or UNIX on it that would run. Multi-process is kind of the defaults, if you're wanting to run things in parallel. Basically, it says multi-core, if you can do multi-core; if not, do multi-sessions. So, that will work whether you're on Windows, or UNIX, or whatever, and it'll just kind of choose the best-case scenario.

Cluster is a heterogeneous cluster of machines, so what would be traditionally called a Beowulf cluster, so you can have a bunch of machines. One really nice thing here is that execute--the remote is one way to say, "I have code on this machine, I want to run it on that machine," which will be helpful outside of VINCI, you're going to want to run everything on VINCI inside of the server, or your development machine, or on SAS grid, the node that's assigned to that process.

So, I am happy to take questions; There are more topics here, and so this was a really quick overview of how I went to go about diagnosing a model; there is a lot more to look at. Basically, you're looking for inefficiencies; and those inefficiencies can pop up from some pretty weird places--this is some of the weird places that they pop up from. There is one question in the Q&A, but it looks like they have answered it themselves, but I’ll go ahead and repeat it: "Is there anything special about 4-H compared to mclappy?" I’ve used mclapply often in my previous work; perhaps, it doesn't work in VINCI. I’m not super familiar with mclapply; I believe that mclapply is a much older version than 4-H or Fortunes.

"Are there tutorials for using 4-H, Futures, VINCI R-Studio, and/or R on the SAS grid?" That is a very good question. So, for the first two: 4-H, Futures on VINCI with R-Studio, not yet. They are coming, though. I am working on a project called R Academy. Because COVID was what it was, this has been delayed, but we are getting things going again and that is one of the advanced topics that is on the syllabus that we will be preparing some tutorials for VINCI, that you will be able to find on VINCIPedia or on SAS grid. Yes, there is a tutorial for that, you can find it on VINCIPedia, you can find it on--it is in the SAS portion of VINCIPedia. I do not have the link handy for that.

Let's talk about the next question. "Any tips on reading a data file? For example, as SAS file into R that is larger than memory?" SAS files are not friendly to R, to say the least; they're SAS files. SAS does not like people playing with their stuff. So, R can read SAS files, but it must read those files into memory. What I would recommend, if you have large data files that is larger than memory, is leveraging the databases that are on R. Very rarely will you actually have a data set that can't fit in memory that you're going to be working with on R--and that is if you if you're needing out-of-memory computations for R, that's out of the scope of what I’m talking about today.

But I would recommend taking that SAS data file, putting it onto the server--the Microsoft SQL server that is assigned to your project, and then using dplyr and dplyr to query that data directly from R. So, use dplyr to create a pointer to the table that you can then filter, that gets filtered and mutations on, that gets translated into SQL, and so those data management and data manipulation processes happen on the SQL server rather than in R in memory, because the SQL server is built for that kind of problem, to do those kinds of operations on data, and manage data, filter data, the database servers are great for that.

R is good at that, but as far as implementation and speed, you're going to get more leverage by using the databases on VINCI than you would be trying to figure it out in memory--which is just better, it's the right tool for the job. Always use the right tool for the job is a point; and in that case, the database servers are the right tools. So, when you have really big data, database servers are your friend.

"Is there a tutorial on using dplyr on VINCI, setting up credential importing and exporting data from and to SQL server?" Yes, there is. Actually, the last cyber seminar that I presented goes over many of those details; there will be more detailed tutorials on VINCIPedia eventually, those are coming. I would refer you to my previous cyber seminar where I talk about using databases with R. There is a package that I talked about in detail that is still coming; we are working on getting approvals to put that package into that, but there is red tape and COVID gets in the way, and things like that.

But it is basically, you establish a database connection, and you point all your tables there. So, R does work very well with the Microsoft SQL servers that are connected to VINCI; it uses integrated authentication, so there's no passwords or usernames that have to be remembered or put in code. Heaven forbid. And then when you're using dplyr, you define all your operations and when you need to pull down the data, you just say the operation is collect; so, collect the data and it comes down into memory. But a lot of things can be done and are pushed off onto the server.

So, I don't think that I have missed any questions. If there are any others--if you're having issues with any of those connections, I refer you to the help desk where I can help you on a one-on-one basis to connect with your data, to help figure out what's wrong with your model; sometimes, it's just another pair of eyes that you need, but there's that--well, I'm not sure what whose razor it is or whatever, but when you're running your code, it's not going to work. But if I come and look over your shoulder, it's going to work just fine; so, sometimes, you just need someone to look over your shoulder.

Moderator: Dr. Redd, you haven't missed any questions. But one person sent something into the chat and I think it was in reference to you--in the chat when it comes to me, in reference to you may be struggling to find a word or a term, and this person just wrote "SAS files in R using Haven." I’m not sure if that's helpful or not.

Andrew Redd: SAS files in R using Haven. So, Haven is the package that is used for reading other statistical languages' files; so, Haven is the package that you're going to want to use to read SAS files; the question that I was answering was specifically if the file is larger than memory. If you are just reading a SAS file, the Haven function, readSAS, is fantastic. It also includes functions for SPSS, stata, the read\_dta, I believe... I think there are other formats in there, but Haven is the package that you would use for reading foreign language--so, other statistical languages data files. You also should look at readExcel to read Excel files, and the readR package that is very nice for .CSV flat files, it's just a format where it does some nice inferences about what the column types should be.

Moderator: This person also just sent in, "Or dsread."

Andrew Redd: I’m not familiar with that one, but I will have to look it up.

Moderator: One question came in while you were talking about that chat that I alerted you to, and they're asking how to contact the help desk. Is that vinci@va.gov?

Andrew Redd: Yep. vinci@va.gov, it goes to the help desk. If you put "R", like the R question in the subject line, it will get filtered to me, I believe, automatically if there is--if you just type it into the question, then someone may read it and then send it on to me. But yeah, vinci@va.gov is the help desk for VINCI; questions can go there, if you if you mention that you're using R, there's a good chance that it will come to me unless it's not really an R question, like it's a Windows question or something like that, and I may then bounce it off to other people.

Moderator: Okay. So, vinci@va.gov and put "R" in the subject line.

Andrew Redd: Yep.

Moderator: Well, we don't have any questions currently. It's possible that somebody will come up with one momentarily who's thinking about it now. But in the meantime, do you have any closing comments you'd like to make before we close?

Andrew Redd: Sure. I love to work with people, I love to work on other problems. I hope that this was helpful for people; like I said at the beginning, there is an artistry to some of this that, sometimes, comes with experience--and I don't claim to be the most experienced person in this, but I can certainly help out where I can. And if I haven't covered something that you find helpful when diagnosing problems with models, I would love to hear about it, either through the help desk or just send it to me directly, andrew.redd@va.gov--andrew.redd@va.gov.

So, there is another question that came in. You said, "There is an R series that you were teaching, I could register through VINCIPedia?" That's a great question. It's not actually a course per se, like a lecture series; however, some of the present, I will record some presentations. This is going to be a series of tutorials of using R on VINCI and just using R in general. So, the first one that should be up shortly, is just how to get it onto VINCI; what do you deal with when you're dealing R on VINCI? Just the ins and outs of getting started. Then we'll talk about syntax and things like that are general to R.

Eventually, we will get up some tutorials on things that are VINCI-specific like using databases, using those SQL servers, using R on SAS grid, which is already on the SAS portion of VINCIPedia, but we will get a reflected thing for our users on that side as well. But it's all coming, it's going to be great stuff. Just look on the VINCIPedia; it's called the R Academy, and there is a page up now, it's a single page that has the syllabus of the topics that we are going to go through, and those should start getting populated within the week--and probably one or two a week for the next little while.

Moderator: It looks like there is another question about guesstimating how long a model will take to run before running it? But to the answer to the question about recording, you'll get an email in two days.

Andrew Redd: Okay. Is there a way to guesstimate how long a model will take before running it? Unfortunately, not really. You start small before you run, you scale up to your whole data, I find the best; and you can usually--after a few runs of a couple small sets, you can get an idea of how long it takes with that set of data. Unfortunately, the scale up is not always linear; hopefully, it's less than linear, so adding a whole lot more data doesn't add a whole lot more computation time; but it entirely depends on how that algorithm is working.

One thing that I didn't go into detail on, but I should point out, way back up here in our first one. In this glmer call, there is a control option; that glmer control creates a list of parameters that affect the fit of the model--and you can adjust things such as tolerance how many iterations you want to allow to go through or what were the different criteria is that you want to do, and adjusting those will affect how it is set. And sometimes, one optimizer will work a lot better than another optimizer; and the details of when one works over another is far beyond the scope of what I wanted to go on to today.

But reading the help files, that's one of the first things I teach people: when you learn R, learn to read the help files; if you're not familiar with the help files, read the help files, learn how to read the help files. Glmer and lme4 is extensively documented; the team that runs, that maintains lme4 is very good, and they've done extensive documentation on those things. So, read the help files on those, and adjusting those parameters can make a model converge faster, or slower, or you can make sure that it's more precise, you can also do things like start specifying starting parameters, things like that. All of those will affect how well it fits.

But the more data you put at it, the longer it's going to take; and unfortunately, there's nothing that we can do that will change that pack. If we could get instant answers with infinitely large data sets, we would be out of jobs. So, the more data you throw at it, the longer it's going to take; you can do estimation of how long each iteration will take with different-sized data sets, and then extrapolate; that's not always necessarily accurate, but it's better than nothing.

Moderator: Well, we don't have any more questions and it's just about time to wrap up. And you have made closing comments; so without any further ado, if it's okay with you, I’ll go ahead and close the webinar.

Andrew Redd: Alright. Thank you, everyone.