Oral History of John Chambers

Interviewed by:
John Mashey

Recorded August 22, 2022

CHM Reference number: 2022.0126

Mashey: Okay. So I'm John Mashey, and I'm interviewing John Chambers, who's the progenitor of the S and R programming languages, which are in some sense a standard for statistical analyses. So John, let's go start at the beginning. Talk about where you grew up and how you got into this.

Chambers: Yeah, let's do that. Before I do that though, let me just kind of put a little bit more background to what you said, John. What I'm going to talk about today is a long, quite long, history that leads up to the R software, and right now I think it's fair to say that R is the standard go-to software for data-intensive science in a whole range of scientific applications, and I have a personal theory of why that's true, and I'm going to-- that will fit in very nicely with our discussion. But the theory is kind of quite simple. One of the things that distinguishes that software is that it derives from the story I'm going to tell you and it's the product of people who were all basically researchers in what nowadays we would call data science, or my preferred term is data intensive science, and so not professionally computer scientists, although you'll see we get referred to as computer scientists occasionally, and that colored our ideas of what we needed to do, what mattered, and in particular, enunciate three principles, which I think go behind R. And so the nature of the system, the nature of the software, is very much tied to this background, and I think that's one reason why it relates well to a lot of people who are doing analysis. But now. So yeah, let's start with the history.

Mashey: Start at the beginning. <laughs>

Chambers: Let's start with the history. I was born and raised in Toronto, Canada, as a couple of the other people in my story were, and the history from the perspective of what we're talking about kind of starts-- not kind of. It definitely starts in the year 1960. In 1960, I was a second-year undergraduate in a program called Mathematics, Physics and Chemistry at the University of Toronto, an excellent program, by the way, and I kind of got into it because science was big in those days and I kind of liked science but I liked other things as well, and so I was drifting along. It was fun. I loved learning, and two events in that year really changed that kind of vague feeling into something quite specific. The first one, and perhaps the most interesting one, is a lecture that I went to, or actually, more correctly, it's-- it was a film of a lecture. The lecture was given by Donald Ivey. Donald Ivey was a professor in physics at the University of Toronto, and here's Donald Ivey. He was an interesting guy, in fact. He did a lot of quite remarkable things. One of them was that he had a very early program on Canadian television popularizing physics for the general audience. But the lecture that I went to had nothing directly to do with physics. It was, in fact, an introduction to a programming language. It was an introduction to FORTRAN. Now, this was 1960. So FORTRAN was quite new; couple years old. Ivey had learned about it. I don't-- I have no idea how. Sadly I never got to meet him personally, but he's one of my heroes. But he was convinced that this language and other software like it was going to revolutionize scientific computing. Now, I was already a little familiar with computers, because I had played around with a clunky old machine, the IBM 650, in a summer job I'd had the year before in the insurance company, and the 650 you programmed by writing assembler code, which was-- I could understand how you did it but it was not fun. But Ivey's concept was that this notion of being able to translate a formula, FORTRAN, translate a formula that looked more or less like scientific notation or mathematics, into something that would actually run on the computer. As soon as I understood that during the lecture, I was bowled over. I was just knocked flat by

it. I walked out of that not only convinced that he was exactly right but that whatever I was going to do in the future was going to have something to do with this idea, and so that was the first event of that year. The same year, the second event, which was not quite as spectacular, was an introductory course in probability and statistics given by Daniel DeLury, who was the-- at that time the Chair of the Mathematics Department at the University of Toronto, and he was principally a mathematician but he also dabbled in statistics, and very much to his credit, in an introductory course for mathematicians, which could've just been a bunch of formulas and their derivations, he included some examples. One I think had to do with German V-1 attacks on London in the Second World War and the fact that you had a sort of random spray of where the bombs landed, but in spite of that complicated pattern you could make some inferences about what was happening and those would be interesting and useful. This idea actually kind of blew my mind too, this notion. You know, I'd been floating around one kind of statistics or another. I couldn't quite get into chemistry. Too much memorization. Physics, maybe. But this idea that there were some principles that applied broadly in science that allowed you to learn from data. I liked it. I liked it. So that kind of set the framework for my later activities. So that was the second-year undergraduate. We jump couple years ahead to the next event I want to mention, which is I was now about to graduate and head off to Harvard for a PhD, Wilmington said would I mind coming in to his office. He wanted to have a chat. He needed to have a chat with me. Well, fine. He was a very nice guy, and he said-- what he said basically was this. "John, you've done really well and we really have high hopes for your research career, but there's a serious problem that we need to talk about that may get in the way." He said, "I've noticed--" he could hardly avoid noticing, "--that you spend a lot of time doing computer programming. Punching cards and running programs on..." The U of T was very good about letting students run programs at that time, and that was certainly true. He said, "You really should be aware that computer programming and academic research don't mix."

Mashey: <laughs>

Chambers: "The time you take out for programming is time you take away from production research, and it's definitely going to inhibit your career."

Mashey: <laughs>

Chambers: So he said, you know, "You should think about this." Well, <laughs> now, it's easy... So, you know, we're looking back 40-plus-- 60-plus-- 60 years, roughly. It's easy to smile at Ralph's comments, but in fact, he was spot on. I know several examples of good friends from later on whose academic careers were either blighted or they just gave them up and went into industry for basically this problem. They were spending time working-- worrying about software and that was not producing articles in the appropriate journals and so forth, so... So there was a problem and I didn't face up to it at the time. I, "M'yeh, m'yeh."

Mashey: <laughs>

Chambers: And of course, I didn't stop programming. No way. So anyway, the next event. We now go to Harvard in the fall of 1963, and my thesis advisor-to-be there, who was also very kindly looking into my

<laughs> well-being, who incidentally was Arthur Dempster from Toronto. <laughs> The second Toronto person here. Art had a suggestion for me. Now, art had been a student at-- got his PhD at Princeton, and he was a student of John Tukey. Now, if anyone can be nominated and has been nominated as the father of data science, of data-intensive science, it's absolutely John Tukey, and the picture I have of John Tukey here is one I love. It's maybe a little blurry but the great thing about it is he's smiling, and John had a wonderful sense of humor and smiled all the time, and he's wearing a black polo shirt.

Mashey: <laughs>

Chambers: Now, in addition to being a professor in the Mathematics Department at Princeton, John was half-time at Bell Labs. He was, in fact, had a fairly high sounding anyway, executive position at Bell Labs, and so we got to know him very well. Now, in addition to Art having done his PhD with John, Art spent a year at Bell Labs. Now, I never actually asked him this, but my feeling was it didn't suit his style-- John. Art. Art was a-- was. Is. He's still alive. Bless him. Was wonderful guy, but his research style was basically to shut his office door and think. Now, you can do that at Bell Labs, but it's not optimal. At any rate. Whatever. Whatever the story. In spite of what his experiences were there, he realized that I was into this computing and other crazy stuff like that, and Bell Labs was obviously a place, and so he suggested that I apply for a summer internship at Bell Labs.

Mashey: Wow. And what year was that then?

Chambers: This would've been the spring of 1964.

Mashey: Oh, yeah, okay.

Chambers: Okay. Now, as it turned out-- so I did, and <laughs> after a wonderful "interview," quote, unquote, with couple of the people from Bell Labs over Chianti and pasta during a raging snowstorm in Harvard Square, I ended up there. It turned out-- none of us could've known this at the time-- it was a fateful time for me to go there, and I was blown away, of course. But more than that, at this point, Bell Labs was-- Bell Labs Research was in the process of being involved in the Multics project. Now, you've heard of Multics from some of your other interviewees. Multics was a very advanced look at computer operating systems, particularly in terms of interactive access and multiprocessing, all that kind of good stuff. It was a joint research project of MIT, Bell Labs and the General Electric Corporation, which was supplying the hardware. So not only <laughs> did the folks at Bell Labs not share Professor Wilmington's view of computer research, they were actually in the midst of thinking about how they could adapt to the new Multics operating system and environment to do better computing for data analysis. This is-- I think one has to realize, this was-- we're talking about the mid-1960s here, so in terms of the attitude in general and statistics and scientific research overall, this was an amazingly forward-looking, would say visionary point of view, and in my opinion, it was the product of several people, but of one person in particular, and that's Martin Wilk. Martin Wilk is another Canadian. Not from Toronto. And he had a very remarkable career. I call him sort of the unsung hero of data-intensive science because he had some ideas, along with John Tukey at the very early stage. John Tukey quotes him directly in a couple of places in Tukey's path breaking article of 1962, "The Future of Data Analysis." If I were to say to a statistician in Canada

that Martin Wilk is an unsung hero, they might not have the faintest idea what I could be talking about, because, in fact, Martin ended up being the chief statistician of Canada during the 1980s. But at the time we're talking about, he was, in fact, the head of the Data Analysis Research Department at Bell Labs, and I-- and as I say, I regard Martin as having been one of the really visionary people. As I say, he was very much involved in thinking about data analysis and a broader view of how statistics and science should function together.

Mashey: How long was he at Bell Labs?

Chambers: He was at Bell Labs for not quite a decade. <laughs> His career is a <laughs> really amazing one. He went from Bell Labs to the planning area of AT&T, where they plan research and stuff like that. Very different. Okay. And fell out with them, and Martin was a wonderful guy, but he was not one who suffered fools easily, and so he either loved you or didn't. He loved me and I reciprocated. But I think he fell out with AT&T. Then he went back to Canada, did a couple of things, ended up at Statistics Canada, which is the more general statistics department rather than the Census Bureau and NIH as we have around-- the standard systems we have here. It's pretty much clicked together in Canada, and he rose to be the highest non-political person in that thing over the next couple of decades. So was amazing stuff. So basically that's the summer of 1964. So definitely I was keen on Bell Labs, and Martin was very keen on me, particularly because the issue is now to plan for the computer future. So I went back to Harvard but I now had a consulting agreement with Bell Labs to come back there. So the next interesting event-- I guess I don't have any video for this-- the next interesting event comes in the-- in February of 1965. A bunch of people just casually met at Bell Labs, including John Tukey, Martin Wilk and four or so regular MTS, which is, in Bell Labs, the term for what's called-- well, you-- level of a principal investigator. Typically PhD. And I was invited, as this now second-year graduate student from Harvard.

<laughter>

Chambers: To come in and participate in full. You know, fully participate in this planning exercise for what Bell Labs was going to do for the software. I mention that because I think it gives a flavor of why Bell Labs was kind of a special place, and Martin said, you know, "You're really keen on this stuff. I think you know something about it, so come and pitch in," right, and we did, and so the end result of that was a plan for something which the acronym was BLISS, which I think stood for Bell Labs Interactive Statistical System. Because it was designed for the Multics operating system it was thought of, I think, as an extension to the PL/1 language, and for the next summer that I was there and for some time after that I worked on it. It was, you know, this is amazing for the mid-'1960s, but if we now ask what eventually came of this, the answer is, "Absolutely nothing."

<laughter>

Chambers: And the reason for "absolutely nothing," as you know, John, is that Bell Labs dropped out of the Multics project, and the reasons for that are interesting but too much of a byway for today's conversation. But basically you can say it was a lesson for research our management not to trust in the

expected delivery date for a piece of software. So I had gone meanwhile-- I got my PhD in 1966. I'd gone to London to spend a year with David Cox at Imperial College.

Mashey: Yeah. My wife did her PhD at Imperial, so...

Chambers: Ah.

Mashey: That was an interesting connection. There's a funny--

Chambers: Okay.

Mashey: A bunch of funny connections here. Yes.

Chambers: Yes. Well, David Cox is my other great hero, along with John Tukey, but in a totally different area. Not-- <laughs> I won't get into David Cox. He's just-- he's one of the greatest statisticians of the 20th Century and just died few months ago at the age of 97. But anyway, so at the end of that year I came back to Bell Labs a little reluctantly. Not because of any technical reason, but my background in Canada had been slightly left-wing and involved in politics fairly actively as an undergraduate, and so I had to talk myself into a career with the largest corporation in the world.

<laughter>

Chambers: But I did. <laughs> And so this was now after Bell Labs had dropped out of Multics, and this had a number of results, as you know, leading, amongst other things, to the UNIX operating system. For data analysis research it was somewhat of a disaster, however. The research management there had agreed to send our nice IBM computer away to a new branch of Bell Labs near Chicago. We had a piece of hardware from General Electric, which was intended for Multics, which had, to be blunt about it, a vastly inferior operating system and a bunch of bugs, and that was it. <laughs> That was where we were. So the next--

Mashey: So that was GCOS? Is that-- yeah.

Chambers: It was CGOS, yes.

Mashey: Okay.

Chambers: But eventually would be sold to Honeywell. It becomes Honeywell. At any rate, that meant that the next few months were kind of a rescue operation in getting software back into a state of being able to do things, and it's easy enough to say that <laughs> this was just bad news, period, but actually, I think, in a way, it's part of the story, because over the next decade or so-- we're not talking about the mid-1960s. So for about the next 5 years to 10 years, we were very much occupied with building a library of what in the day we would've called algorithms, which for us meant FORTRAN callable subroutines, but the point about them was that they were designed to do the best possible job for some computing task,

and that was part of our understanding, which is obvious to us, that we had to have the right answers. The best answers we could get, and we wanted to get them as quickly as possible and be able to use as sizable amounts of data as possible. So that was-- that led to a focus on methods, on the methods involved, and now having been diverted from-- for the moment from languages, I spent the next decade or so playing around with all sorts of areas, numerical and your algebra optimization, random number generation, graphics, all kinds of things like that, and got to be quite, I will say, fairly familiar with the state of the art in all of them and became convinced that it was important for computing for statistics that people used the best methods. This was not entirely an easy sell in those days. I recall one eminent statistician at the time saying to me after a dinner conversation, "Well, I'll tell you, John, the truth is, I just want the answer. I don't care how it's computed."

<laughter>

Mashey: So calibrate the sort of state of the art and what computing facilities you were using and what were Bell Labs researchers just doing? Were they just writing FORTRAN code? I mean, what were they doing?

Chambers: Yes. Basically... So now, we have to think little bit about Bell Labs locations. Most of Research was at Murray Hill.

Mashey: Yes.

Chambers: And the bulk of the Murray Hill researchers and developers-- there were quite a few developers there too-- used the central GCOS Murray Hill operating system, and that, at this point, meant you had to write-- well, you could write assembler code, but you had to basically write things in FORTRAN or FORTRAN callable, so what we ended up with was a rather extensive library of FORTRAN callable subroutines. As I say, I got quite into this. It got kind of not forgotten but I'd pushed to the back of my mind the dreams <laughs> of the BLISS project and all that kind of stuff, and was kind of interesting, and so early in the 1970s was John Tukey was about to write his books on exploratory data analysis.

Mashey: Yeah, I still have that book. Yes.

<laughter>

Chambers: Was reading quite a few books, and so I decided, "It sounds like fun. Let's write a book or two or three," okay. So I had a project of writing a-- eventually, or initially, I thought, a series of books on the computational methods for data analysis. It ended up being one book and it didn't come out until 1977, mostly because it took four or five years to find a publisher who was willing to publish it.

Mashey: <laughs>

Chambers: But it did eventually, and so this is that book. It's the first of the books I wrote. I want you to notice, the cover is blue. That will be relevant. But it basically, the contents of it, in fact, if you look at the

chapter headings, numerical, and linear algebra, optimization, random number generation, graphics, correspond very much to the kinds of things that we had available in our computing environment, and to some extent, I managed to talk some of the other places like Carnegie Mellon University, for example, who were doing active work with statistical software to follow in this general pattern. But, you know, it was a lot of work, and the truth was that for the statisticians at Bell Labs, or the other people who were doing data analysis, and who weren't fans of computing, particularly, it was not optimal. In order to run an analysis you basically had to set up a FORTRAN main program, arranged to get the data from whatever, magnetic tapes or wherever it was, do all the computations correctly, get the results out, do something with the results, put the results somewhere. So in effect what this meant was that serious data analysis had to involve a team and the team included a person who at other places would've been called a programmer, that were not-- emphatically not called programmers at Bell Labs, but their job was to do the grubby part of the computing. Or as the younger people came along they, the PIs would do the grubby part themselves to save time. But it wasn't fun. It wasn't fun. But that's where we were in the mid-1970s. So the next relevant event was the arrival at Bell Labs of a young man called Rick Becker. So there's a pretty long story there, which I think I'll also <laughs> omit, because it's a little bit tangential. But anyway, Rick came in. Rick had been-- had had jobs previously at the National Bureau of Economic Research in Cambridge, Massachusetts, in which, amongst other things, he'd worked on an interactive system called Dazzle, which did some fairly specialized computing, mainly for time series analysis, but it did it interactively in quite a nice way, and so Rick-- so Rick came to us in one of these husband-wife deals because his wife Brenda Baker was a excellent computer scientist, so-- and we strongly wanted to hire Rick. So Rick came to us and for the first year or so he and I worked together we hit it off immediately. So here's a picture of Rick and me a few years later. We'll come back to the context of the picture, but Rick's sitting there at the computer operating system. The first thing that we did was a collection of software for we call it computer graphics. It would now be more termed as data visualization. It was a FORTRAN subroutine package but quite integrated with a model for how graphics was done. The model, incidentally, is the same one as exists now in what's called base graphics in R. But that was that, so that was fine, and we talked about that and people used it, and then finally, Rick's patience ran out and he said to me, to put it bluntly, he said, "The way that people do computing here is shit."

<laughter>

Chambers: Said, "Look at these control cards you have to write. It's junk. This is impossible. This is unbearable." <laughs> Not to be put up with. Well, of course, that's what-- and I knew about his experience in NBER. Well, of course, he didn't have any trouble convincing me. <laughs> It just sort of reawakened my frustration. So we decided we'd think about, "Can we do something about it?" all right. So early 1976-- late 1975, early 1976, we just sat down and played around. We just did some stuff. Rick had some ideas. I had some ideas. We used various pieces of software that already existed, and, for example, some-- a preprocessor for FORTRAN from the computer science people that made FORTRAN little bit more like a reasonable programming language and things like that. So now we come to, in fact, the birthday of S. S has, rather unusually, for most kinds of software, a very specific birth date. It's the 5th of May, 1976. The 5th of May, 1976, Rick and I presented the ideas we had so far to a group of our colleagues at Bell Labs, and that. And that is, in fact, this piece of paper. This is-- well, when it was made into a Vugraph it was, the first graphics of my presentation at the May 5th, 1976 meeting. You'll see

that following the usual engineer rules at Bell Labs I put my initials and the date on piece of paper so that we could use it later on for patent purposes. It was never used for patent purposes, but this was basically our ideas, and I'm going to go into this in a little bit of detail, because it was, well, was fun, and it has some positions in terms of what I call my three principles of computing for data-intensive science. But let me just walk us through quickly what it describes. You notice that the top part says "Algorithm interface," and this is our concept, our concept was that we would provide a mechanism for an interactive interface to algorithms, namely Fortran and subroutines, that would be useable, natural, and effective for computations in an interactive mode. All the stuff that was missing from the standard interface at this point. So the idea was there would be a piece of software, what we called the interface routine, that would be-- would get created from the software we were going to write, and would then be able to do computations in an extended version of Fortran, which would allow it to access specific information, like Fortran arrays, and scalar numbers, and all that kind of stuff, and pass those down to Fortran subroutines. From the results that the subroutines would then return, it would construct the value for the call to this function. So this is the function, this square bracket. This square object will turn into a function in the interactive system that we were designing, which had no name at this point. So that was the concept, and we had done-- we already had code that did this, at least in an approximate sort of way. So that was the first principle. Now, this is obviously a bit primitive, but it catches two key principles of the three that I want to talk about. They are that everything that gets computed, R, is a function call, and that people think of their analysis and the computations they're doing in terms of function calls which take arguments representing data and other kinds of interesting things and produce a result which has some meaning for the analysis. The value of the function is then the result of that function call. So everything in R is a function call, okay? But the other principle that it invokes is that it's not assumed that you're going to sit down with this language and write everything, and get everything pre-programmed in R. In fact, it's assumed that there are interfaces from the interactive environment. In this stage, of course, it was always in interface to Fortran, but it'll quite soon become interfaces to multiple languages, and multiple other ways of getting computations done, and that it's those interfaces that ensure that you make use of the best algorithms available. So that's why it's called an algorithm interface. So this is still a principle, that's my second principle, that computations in R are always based on this concept of an interface to the actual computations, the deep computations. So, now, that's the top part of the diagram. The bottom part of the diagram, so we haven't said yet what it is that this interface is getting, and what it is that it's returning in terms of the actual computing going on, and what that is is a data structure. The bottom of this diagram is the data structure as it existed at that time, and it's simplicity itself in one way. It's a hierarchical object and it has named components, optionally, and those components have a type, and a length, and a pointer to something, okay? The pointer may be a pointer to a Fortran array, for example, but it can also be a pointer to another one of these structures. So in other words, this is a recursive hierarchical thing. So, that's it, and everything that is used in the computations, everything that's produced in the computations has this single structure. So that's my third principle in R, everything in R is an object, and R has its own, obviously, semantics for what objects are like, which is not totally unlike this, but <laughs> obviously adapted a lot. So those are the three principles, and the interesting thing I find now as I go back to 1976 is that although we were far away from thinking of these as principles at the time, they're hidden there in the design. Again, to me, that says that this reflects what we knew about what we needed to do, that we needed for computations. But anyway, that was that.

Mashey: Great, so this was in, I think, Henry Pollak Center, is that right?

Chambers: Yes.

Mashey: Yes.

Chambers: At this point, Henry Pollak was the director of that center. Martin Wilk was no longer the department head at this point. He had left and someone else had taken over, and a very fine person, but I won't talk to you <laughs> too much about him because in my opinion, he was not a visionary. That wasn't relevant. In fact, it's interesting to note that there were five or six of us in this meeting, no management at all. Again, it's kind of typical of our lives at the time that these two regular guys had no qualms about going away thinking about inventing something brand new, with no project, with no prior authorization, or anything like that.

Mashey: Well, that was in research. <laughs>

Chambers: Yes, indeed oh yes, quite so. It was quite in research.

Mashey: Yeah, so those are some other areas--

Chambers: Yes--

Mashey: -it was not quite that way

Chambers: -when I say Bell Labs, I mean Bell Labs research, of course. <laughs> But that was the way it was, and we didn't think about it. We'd tell our management at some appropriate <laughs> point. But anyway, so that got us going, so that started us off, and the colleagues that were with us in the meeting liked the idea, generally speaking, and contributed ideas, contributed software, and so over the next year or so, this came to be an actual useable system, locally, and--

Mashey: When did the name S--

Chambers: Yeah, <laughs> you're talking-- that's the next-- thank you for that, that's the next lead in. At this point, as people started to use it, and people nagged us to write some documentation and things like that, we realized that it would be necessary to give this thing a name, alright? So here's my story, people may dispute it, but this is my story. We asked our colleagues for suggestions for names, and a number of people had suggestions. Rick and I collected them together and looked at them, and we didn't like any of them. But we noticed there were two things that came to our mind, one was that all of them contained the letter S, all of the names. So the other thing that occurred to us was that the guys upstairs in computer science, on the fourth floor above our offices, had been playing around with some software and stuff, and one of them, Dennis Ritchie, had written this language which-- whose name was C, just the letter C. "Hey, that sounded cool." So we decided we would just use the letter S. <laughter> So that was it, for all--

Mashey: I'm glad to hear that. I always had a suspicion that had something to do with it, so...

Chambers: <laughs> Yes, and for a long time we had trouble convincing the people who wrote articles and stuff about it not to put quotes around S, because it definitely didn't-- anyway, so, that's the story. So, over the next year-and-a-half or so, it kind of gelled to something fairly specific and people started using it, not only--

Mashey: This is on GCOS?

Chambers: This is all on GCOS. Thank you, John, that's the next relevant point. <laughs> Not only our research colleagues but a number of the groups in development who were co-located at Murray Hill, and so also could use the Murray Hill computer, started to use it. But that was very important because we got some very valuable feedback from the advanced development people, who said "This is all very well, but <laughs> if I do this, it sits there for the next half-hour," and things like that. So it was wonderful, and we really enjoyed that. It gradually came to be somewhat more official, and still, within Bell Labs, and always having to be used at Murray Hill. But it really did become a piece of software that Bell Labs recognized, AT&T recognized internally. In fact, that picture that I showed before, if I go back to it, that picture of Rick and I is from an article in the Bell Labs News, the in-house newspaper, which talks about this statistical software, Sleuth, <laughs> notice the S, that these guys have produced, that is being used in various places around the labs. So there's Rick and me presumably running S, and in my hand is, in fact, a book. That's the first version of that book, and I have the second version of the book with me, and I rather enjoyed presenting this book today. Because this is a book that I'm sure virtually no one who watches this video will ever have seen, and I probably own one of the very few copies of it that still exist.

Mashey: Ah, yes, it's--

Chambers: <overlapping conversation> It's called "S: A Language and System for Data Analysis", and it describes-- it's book-length, it describes the use of the system as it was within AT&T at this point. It was now still just an internal product. Okay, so, that was that.

Mashey: And mostly usage was from Murray Hill folks?

Chambers: Well, the usage had to be through the Murray Hill machine, but fortunately the Murray Hill machine was time shared. So we did have users at other locations, but they had to go in through a computer terminal, and that kind of an interface. Yes, that is exactly the point. So Rick and I thought this was nice, and at this point, we're now, oh, approaching 1980 or so. Unix had begun a process of being licensed for use outside Bell Labs, and Rick and I thought that was a great idea. We had no hesitation about wanting to conquer the rest of the world. But there was a problem, a very serious problem, which was that we had written S software to run on the GCOS operating system, which was not the world's best, and had some peculiarities which Rick had cleverly exploited in order to be able to swap in and swap out the various functions, the various interface routines, and the code that they used. So we thought about how to make that work on another operating system and it was a daunting prospect. But bless their hearts, the Unix people had already also been thinking about this issue of portability, and so there was, at

this time, a new version of Unix written, which pushed the idea, wonderful idea, of a relatively portable operating system written in a high-level language, C, with a few key machine-specific operations, or primitives within the operating system, which allowed people to program it for a new--

Mashey: Research Edition Seven

Chambers: -<overlapping conversation> target.

Mashey: -yes.

Chambers: Exactly. Not only that, it was being tested and written on a VAX 32 machine. The original machine the Unix was on was a 16-bit machine, which <laughs> was not too suitable for data analysis. But the VAX was a fine machine for data analysis, and so voila, our problem had been solved. We would define portability to be a Unix implementation of S that could then be shipped off to anybody who had Unix. So we then proceeded to write a new version, technically version two of S, which was pretty much the same from the user's perspective as the first version. A couple of new things have been added to specialize language for writing those interface routines, and some other things like that, but the key point was that it was built on Unix. So, in particular, now the interface which had previously been solely to Fortran was now a three-way interface to Fortran, to C, and to the Unix shell. So this broadened the sort of things that people could use. So, that was that. Now we come to the mid 19-- well, we come to the early 1980s. We're now in the process of licensing S, and, indeed, that version of S was licensed, it went out, moderate number of places got it. Those who were more actively involved in computing for statistics were keen: Carnegie Mellon, Toronto, a few others, Princeton. But in fact, it didn't have any terrific traction primarily because in order to write new functions for S, you had to write one of these interface routines. We tried to make that straight forward, but it was a programming job, it was not something that the person who was just using the system interactively would be happy doing. So this was a block and it limited expansion of things. Now, I'm going to go a little off stream here, which-- well, I guess I will. <laughs> So at this point, there's a non-technical problem which arises. As I mentioned, the management had changed in the data analysis research area, and it was not against what we were doing but it was not hugely enthusiastic about it either. <laughs> I will quote one of my superiors at the time saying to me "You know, John, S is really neat, but serious data analysis will always have to be done in Fortran." Now, I don't know whether I had the smarts to give the right answer to that, which is fine. Any serious data analysis that people want to do in Fortran and spend the time programming, we'll grab it. Okay, anyway, <laughs> whatever. So, along came <laughs> a real character, a real cowboy, who was the director of a non-research center involved in business planning for AT&T, who had an unusual department in his center. It was funded by research money, and it was called the advanced software department. Its charter was a little vague, but basically, what it was intended or hoped to do was to use research type ideas to evince some clever new tools for the big programming problem that AT&T was facing at the time, which was to convert the switching systems for telephone calls into computer code. That is too far afield to get into, but that was what this department was meant to do, but fortunately they had the good sense to realize that they shouldn't tell us how to do it. So we realized this was an impossible task and <laughs> that we were not going to solve this problem. Anyways, so, <laughs> yes, get me back to the story here. So Jim Downs, the director of the thing, came into my office one day, and sat down and said "Do you

know, John, I don't think your management appreciates you." Now, this was a very clever first line > to use-- okay, because I kind of had <laughs> the same thought. So anyway, Jim talked me into taking over as head of this advanced software department. Anyway, Jim was a great talker. And I did, and they were a wonderful group, I loved them. But this was quite a serious blow in terms of our story, because it meant that Rick and I were now in very separate parts of the organization, and while we were still interested in getting S out there and all that, we didn't have the same kind of down-the-hall interactions that had made it possible to do the work on S. So the whole thing could kind of ground to a halt at this point, that it did not <laughs> is due to another totally non-technical-- well, non-technical from our point of view event, which took place at this time, which was that in the early 1980s, AT&T agreed to settle a anti-trust suit which had been placed by the federal government, accusing AT&T of being a virtual monopoly. As part of that settlement, it agreed to divest itself of all of the local telephone operating companies, which would then become a number of regional separate companies totally unconnected to AT&T. In return, AT&T would now become a competitive standard corporation, right? So this was certainly a huge event. Not viewed terribly positively by a lot of people in Bell Labs research because <laughs> there's one famous quote in the corridor, in the physics area said, "Well, no more transistors." <laughter> In fact, the change was not as dramatic as that, but that's another story, too.

Mashey: Yeah, I remember this well, that's what caused me to leave and come to Silicon Valley, yes. <laughs>

Chambers: Yes, you were not the only one. <laughs> However, for our story, the relevant fact is that the Henry Pollak and the then head of the data analysis research department both decided to go to a new research development arm which would be associated with the spun off telephone operating companies, okay? So that meant that it was necessary to have a new director of the center, with Ron Graham, and a new head of the data analysis research department. So I got this phone call in my office a couple months later, and it was Ron, I knew Ron, and Ron said "Hi, John. How'd you like to come back to research as head of the data analysis research department?" <laughter> So something like after a careful consideration of 10 seconds or so, I said "Yeah, that's okay." <laughs> So--

Mashey: So did you learn to juggle?

Chambers: <laughter> No. Ron had a number of talents <laughter> that I didn't try to emulate. <laughter> But no, Ron and I got along very well, very well. Anyway, so, but the point of all that for our story is that I came back to the area, to the research department and Rick and I now, again, had offices next door to each other, and so we sat down and got together. I had been doing some work because I insisted with Jim that I was still going to do research. I'd been doing some work on a different but very much related project which I called the coordinated programming environment, very much influenced by ideas in functional programming that were quite current at the time. Rick had been doing some work on the innards of S and the computations, and so we decided after talking around for a while to pool our ideas and produce a new version of S, which would be quite different from-- well, quite different in terms of how the user programmed it. It would now be a functional programming language built on top of the same sort of interfaces to algorithms that existed before, and with a unified data structure which was now more explicit, and more available at the language level. So we started working on that and we brought in

a third partner, Alan Wilks, into the project, and that produced, eventually, what we called the new S language and another book. Sure you know this also has a blue cover. So this is a book from the three of us, and it describes the new language in considerable detail, including not a semantic model in the formal sense, but a chapter on how the language works, which explains the innards of how function evaluations takes place, and things like that. So that went out there. That was 1988 that went out, and a different group of us then started to work on a different-- on what ended up being called “Statistical Models in S”. This was what it says, it was 10 authors, <laughs> I was one of the editors and an author. It was an interesting experience in cooperation which worked pretty well, and it described S software for all the-- a lot of the standard kinds of models and a few non-standard kinds, linear, non-linear, generalized linear models, all kinds of things like that, with separate chapters for each one. Also a few additions to the new S language, in particular, a simple, rather heuristic approach to classes, to classes of objects in the language. So that came out. So the blue covered book got to be called the blue book, and this, which you notice has some blue but is mostly white, got to be called the white book. The natural question is why, why did these two books get nicknames whereas the other books didn’t? Well, the sequence of events is that the new version of S, particularly with the statistical models built in, took off, it became quite popular. It was still a licensed proprietary software from AT&T, but academic institutions and other research organizations could get a research license almost for free, basically, for little more than the cost of reproducing the tape. There was also a provision for third-party commercial licenses, as there was for Unix. In fact, all of this was modeled on Unix’s licensing scheme and a number of companies took up the challenge to produce commercial versions of extensions to S. One of the most particularly successful in its product was called S+. So between the research you--

Mashey: And who did that?

Chambers: Well, <laughs> it has an interesting history, which is-- I believe in qualifying my memory to remember it all. It began with a couple of people from University of Washington, but it had a long history. It was bought out, and bought out, and bought out by different groups, and it actually still exists. It's now-- oh dear, I've forgotten who the corporate owner is now. But it’s now <inaudible 00:53:30>. <laughs> For our story, the relevant point is that lots of people started to use S in various-- particularly still, I think, emphatically particularly still in university environments and--

Mashey: So this is 1988--

Chambers: So now--

Mashey: -‘89?

Chambers: -we’re in the 1990s.

Mashey: The early ‘90s, yeah.

Chambers: 1992 is the white book, so this goes through the 1990s. I wrote another book, which I won’t talk about today, on S that came out in 1998. So that’s the continuing story. However--

Mashey: Because I'm looking at-- the timing of the takeoff is interesting.

Chambers: Yes, it is interesting. I think it reflects a number of things. Of course, personally, I believe it reflects this philosophy that I've been talking about, which I think fit in well with what people wanted to do. It also reflects, I think, very much increased involvement of people in doing what we'll now call data-intensive science, or data science. Because it was now much easier to do, there was much better hardware, you could handle larger datasets. It was just part of this Moore's Law. Everything was operative here and it was just part of the flow. It was, for us, very rewarding, and this is when the ACM Award was given to S.

Mashey: But, I mean, it's interesting because it to some extent follows the same path that Unix did sort of a decade earlier?

Chambers: Yes. I think the distinction-- and, again, this gets really back to my first point. The distinction here was that this was software written by data analysis researchers for people doing data analysis. So it's natural that Unix was picked up by computer science people, and then so that happened at that point. It wasn't really until the effects of that started to take place, namely the availability of software to wider groups of people, that it was more likely that people would use S. So, now, the next event, however, takes place in 1996, and it's the appearance in one of-- the Journal of Computational and Graphical Statistics, with an article by Ross Ihaka and Rob Gentleman called "R", colon, "A Language and a-- Language and System for Data Analysis and Graphics." As Ross was prone to say, R had a certain resemblance to S. <laughter> But, of course, the crucial-- well, from Ross's point of view, the crucial difference was that he was using some ideas, particular ideas related to LISP style computing, which were not part of S. But from the point of view of the readers of that article, the key point was that R was open-source software, freely available to anybody. So what happened then, which now becomes less kind of-- much less my personal story and kind of a story of what happened, <laughs> was that a group of people, an international group of people, generally in statistics but very much into computing in one way or another came together and said "Look, Ross and Robert, what we want is not something that vaguely looks like S, we want a free version of S," okay? I'm not sure actually, and I never got a chance to really iron this out with Ross, I'm not sure that was quite what Ross had in mind. I think Ross had in mind more moving towards a LISP model for computation. But <laughs> it was free software and the people who were willing to work on it spoke. So over the next two years, the R core group, as it came to be called, worked on a version, a free version of S. And that's why these two books are relevant. Because what did they have to work with? Obviously, they were not allowed to work with the software. That was verboten. What they did have though was a fairly detailed description of the language, and of the software that they used for most of the interest interesting stuff that it did in these two books. And so their task, which is basically to use the books as references and produce open source versions of the same thing, or equivalent things that worked slightly differently, and that's what they did. And on February 29th, the year 2000 on a lovely, lovely date, the first version of R was released. It's a lovely date because it was the first instance of the use of the third correction to our current model for the calendar. Anyway, so that was R. R took off and the rest is basically history. R has just gone-- we managed to get, I would guess users of maybe pushing hundreds, thousands. That may be a little optimistic, not too much, but R has millions of users. And not only that it has millions of users in all of these different areas. And kind of to summarize all

of this very important in that, things that are very characteristic of R and the wonderful people who work with it. I do some work myself. But the R package and the institution like this thing called C-RAN or CRAN, which holds 20,000 or so R packages for everything you could imagine.

Mashey: Yeah, I've looked at that. Yes.

Chambers: And all of this is really just part of why this has taken off and the way it's taken off. I think it's wonderful. I love it. The truth is for the last five years, my own-- finishing up my own side of this, I've not really been involved very much. I'm still on the board of the R Foundation, but my last five years have been-- well, for the last 10 plus years I've been at Stanford after retiring. And for the last five years, I've been officially senior advisor to Stanford for data science which is the unit of Stanford doing a variety of things trying to promote data science. The aspect of it that I'm involved in particularly is we have a program called Data Science Scholars which selects and funds a number of PhD students from all over the university who are doing research, which is related to in my terms, data intensive science. We help to fund their research. We have them get together and work together as a community, and we have them take on projects to enhance data science at Stanford and elsewhere. And that's what takes up most of my time these days. I love it. I'm just ecstatic about it. That's basically today.

Mashey: This raises a few other questions, just for fun.

Chambers: Yes. Yes.

Mashey: Some of which are going to be pretty random, right?

Chambers: Great. I love random questions. You'll get random answers.

Mashey: Okay. Building Five at Murray Hill.

Chambers: That was where I went when I went out of research. That was Jim Downs' building.

Mashey: Yes. Okay. But as I recall, they got a lot of data from around the telephone cos.

Chambers: They sure did. And that was one of the things that-- and so then-- so they were S users.

Mashey: Yes.

Chambers: So that's how Jim found out about me in the first place. Yeah. But their main job was basically to do business planning for AT&T.

Mashey: Yeah, I actually use them. You know, of course, I coined the phrase "big data" a long time ago. And I ended up doing a presentation where I said, you know, Bell Labs did big data a long time ago.

Chambers: Yes.

Mashey: …Building Five.

Chambers: Building Five did big data. We did big data, too and research before I was in Building Five. For example, one of the very early examples we had of people using S had to do with data on microwave transmission of telephone signals. A couple of the researchers in our group had set up one of the very early automated data recording devices which kept track of how well these signals were being transmitted through microwave. And of course, the issue was that rainfall and other kinds of disturbances would interfere with the quality, but that was big data. It was a whole reel of magnetic tape. So this would have been late 1960s, early 1970s. That was big data.

Mashey: Yeah. Yeah. Well, I often use images from here at the museum saying, you know, big data is nothing new, just the term you got was fairly new. But I go back to the Hollerith machine of 1890 because I have an image from the museum here, you know, because there was big data for its time.

Chambers: The definition of big data is if you have to worry about how big it is.

Mashey: Yes. Yeah.

Chambers: In fact, we got complaints because S was-- S could handle reasonably large stuff. But, of course, it all-- it did all its computations in memory and especially on the GCOS operating system that was not-- that was a limitation. And so we would get grief, particularly from our users in development about I tried to load this 500,000 observations to my machine and it sat there for the next hour. And so they would say to Rick and me, “Do something about it.” And Rick's answer, was always, yeah, Moore's law will solve that.

Mashey: Moore’s law. So looking back are there things that in retrospect you wish you would have done differently at various stages? That's always-- I always ask that.

Chambers: Let's restrict this first of all to technical things rather than another.

Mashey: Yeah, technical. I mean technically.

Chambers: Otherwise the list is too long.

Mashey: Yeah.

Chambers: I don't know. Do I wish I hadn't gone out of research for those two years? To some extent, yeah, but I don't know.

Mashey: So for instance, I would ask in retrospect the assignment being less than minus. Sorry, I had to ask that one.

Chambers: Yes, very good for you John. There's a story behind that. When we were doing S initially the facilities for timeshare computing existed on our operating system, but they were not the greatest. And the hardware available for doing it was not the greatest. In particular, the hardware that you could use from home was not very extensive, but it mainly consisted of a terminal slightly larger than my suitcase called the Execuport [And one of our principals-- well it wasn't the principal one of our pet peeves was using the equal sign for an assignment in Fortran because that implied a reciprocity between the left side and the right side, which was incorrect. So we wanted a non-symmetric symbol. Now, it happened that the good old Execuport had the property that if you use the underline key it printed as an arrow. So we made the underline key the assignment operation. But, of course, on other terminals the underline key was not an ideal one. So our users bitched at us and said, get us another assignment. We liked the arrow, so we produced the thing that look like an arrow. There.

Mashey: Okay. Well, there is some history there.

Chambers: I'm not-- this is not necessarily theoretical justification. It's an explanation.

Mashey: Yeah, it's interesting. I mean, you know, some of the quirks and oddities that have happened by strange coincidences like this, I like to do that history.

Chambers: Yeah, it's lovely. And I really enjoy sharing some of these moments of things like the where did the name come from, for example.

Mashey: Yeah. How about other design decisions early? Are there things that you wish you had known that we're coming that would have simplified things, or that you had to redo?

Chambers: Probably. It would have been nice if we had connected with the Unix people earlier than we did. That wasn't entirely our fault. They were not, in general-- they were a little bit cliquey, some of them more than others. Dennis Ritchie was a wonderful person.

Mashey: Well, I knew all these folks, of course.

Chambers: Ken Thompson was a little bit more of a character. Rick got to know them quite well because Rick was much more into the nitty-gritty of operating systems. I was coming from the numerical analysis side of things more. Yeah, so I think if we'd been more on board with Unix-- so then the other point was, of course, we got permission to use the Unix machine early on. Yes. That reminds me. And so I got-- I played with it. I managed to bring the system down after five minutes.

Mashey: When was that?

Chambers: Oh gosh, it's got to be early…

Mashey: Very early. The 11/20 time?

Chambers: It would be after we first had the SIDS.

Mashey: Okay. All right, so that's a little early.

Chambers: It's mid-70, sometimes.

Mashey: Okay. PDP 11/45.

Chambers: Yes, it was PDP 11/45. Yes, definitely.

Mashey: But yeah, one could bring those systems down. I mean we had at Piscataway even the programmer's workbench because we had to do a lot of hardening because we were always running-- we always running bigger systems and more users and everything.

Chambers: Oh yeah, the other story is we're getting onto interesting stories. So my wife was at Bell Labs. She was in the physics research area and did computing and programming for them long before we met. And one of the things that she got to do was to look at Unix as a possible environment for the physics people doing their own programming. But the first thing she discovered was that there was a serious bug in the C compiler routines that did trigonometry-- did sine calculations. And so Bee [ph?] felt if you couldn't do sine calculations…

Mashey: Yeah, forget it.

Chambers: …which possibly not anyway.

Mashey: Well, I mean…

Chambers: But, you know, it wasn't that we didn't appreciate the ideas. I mean we did very much.

Mashey: But certainly scientific computation was not a priority early on.

Chambers: No, no. And nor was it reasonable to expect it was. And then, of course, you know, the story behind the PDP-11 why that was the hardware they were using?

Mashey: Yes, yes, they wanted the PDP-10, but didn't have the budget.

Chambers: The machine floating around, they'll be using it. Yeah, you guys can have it. You're not doing anything useful anymore anyways. Yeah. So that's one. I always wished we brought more people into the program earlier, gotten more points of view-- but I don't know. It was a pretty happy time.

Mashey: Yeah. I think that is another example of something that gestated for a while in a kind of friendly environment in the sense of, you know, there was a big internal market for it.

Chambers: Yeah. And, you know, I have to keep-- I should say part of what's coloring, some of the things I'm saying today, I just finished writing a couple months ago a perspective piece for Statistical Science for a special issue that Cara and I edited on the pandemic, and citizen responses to the pandemic. So the opinion piece-- I was just invited to write an opinion piece on anything I wanted. So I actually wrote a piece which basically behind the verbiage says science can't solve the problems that we're facing these days. We have to make policy decisions and stick to them to solve those problems. When and if those decisions are made, though, science is going to be critical for making the right moves in response to the decisions. And it has to be a new kind of science. It has to be very broad, very interdisciplinary, very motivated and very, very much oriented towards finding the right answers rather than publishing papers. So, it's got some conflicts with the current academic research environment. And what I suggest is that it needs a kind of structure and a kind of inducement for the talented young people to be involved, that doesn't exist. And it would be a simplification but not too much to say that what I'm suggesting is we need four or five versions of Bell Labs research around the world, and people like my scholars to be the personnel for it. So that I think-- and that's what's coloring my statement. I think key to all of this was that there was a certain empathy with this kind of attitude towards, again, John Tukey. Let's go back to John Tukey. One of my favorite-- probably the favorite of all my quotes of John Tukey is that the fundamental justification for data analysis is science. Not mathematics. And that was very much John's perspective. And, you know, nothing against mathematics I'm a mathematician by training, but it's this relationship between the analysis and the underlying science. And that means not only do you do-- need to do the analysis, you need to understand the data. What's the property of the data? To what extent can you trust it? What does it mean? How does it relate to what you wanted to ask? And then in terms of the results it's not simply that you have the results, how can you make those results meaningful to the right people? And so, this is the triad of data analysis communication which is critical for the future.

Mashey: Well, I run into this, of course, because I'm close with the climate science community and I used to build super computers for them. And I'd visit NCAR and GDFL with other places and they have exactly these issues.

Chambers: It's absolutely that.

Mashey: So talk some about bad data or missing data. And, of course, some ideas of that were built into S pretty long ago.

Chambers: Yeah. I'm tempted to say so it's-- excuse me, a left field bit here. There was a famous lady, a British lady who taught, how to raise a dog, how to properly raise a dog, how to properly train a dog. And she was a real fierce British lady, Tweedy lady. Her favorite saying was, "There are no bad dogs. Only bad owners." And I'm tempted to say there's no bad data, only bad understanding of the data. Now, the understanding may lead you to say that the process that produced this data is-- well, in extreme cases fraudulent, but not very often, but perhaps not suited to the questions I want to ask or before I can use this data, I have to understand more clearly how it was collected? Why this part of the data differs from that part of the data? Again, climate science being a prime example of that. And so it's really just a never ending part of data intensive science is to be forever examining the data, examining the process that produced the data, asking yourself, if you could improve that process and folding that into the workflow. It

just can't be-- less so now than decades ago. Statisticians tended to have this idea that somebody else worked the data and when it was clean, gave it to me. That was never true. And people like Martin Wilk would have disabused them of that idea back in the 1960s. But it's-- but emphatically now understanding that whole flow must be part of the science that's involved in doing the analysis just can't be-- it can't be separated. It's not. And which is not to say anything against lovely tools for helping to do it.

Mashey: Yeah. I just wondered because I had managed a project at the labs around 1979 that involved putting together multiple data streams from different parts of the telco. And it turned out, of course, there would be errors in ticket numbers and things like this.

Chambers: And what this field meant to New Jersey Bell wasn't the same as it would have meant to Pacific Bell and Co.

Mashey: No, that wasn't actually a problem.

Mashey: Well, my wife worked in Common Languages, so she ran into that. right? But the issue was there were multiple data streams. Some were coming from Bell Labs created software. Some were coming from local payroll systems. And you had to match pieces up, okay, and they missed things. And they were wrong and there were errors and all this stuff. Right? And it turned out nowhere in the requirements was there anything for cleaning that data. And we kind of had to add it, but fortunately, most of the software is written in AWK so it was pretty easy to fix.

Chambers: Yeah, well, as I mentioned we put out the special issue on the data science response to the pandemic. And as you can imagine particularly for American data this is the crucial issue. The crucial issue is that, you know, data is collected either at the state level or at the county level. It's kind of different administrative organizations and being able to use that to build models that are national level is, of course, quite a challenge. And there were some really superb, I think, worked on-- principally at Carnegie Mellon by a number of stations and others to deal with this problem.

Mashey: Well, certainly the whole biomedical turf, I'm an advisor up at UC San Francisco, and the Center for Tobacco Control and Research and Education. And I've been in meetings where we look at forest plots like crazy of the different studies done on things like vaping. Okay. Right. And, you know, it's interesting to see the outliers and the ranges and if they can try to figure out why people are getting such different answers.

Chambers: Yeah. I think it's important to understand, you know, this whole period has just been revolutionary and just amazing. And I think it's important understand what's possible nowadays that was impossible not too long ago. And data is one of the areas which this is true. And, for example, in ecology and in the study of distributions of species and things like that there was just-- there's been a very-- a couple of very interesting articles by some people in the field about just the revolution in ecology. And so, in fact, the one sentence I like is, from one of these reviews, it says that it turns out that the-- I forget which specialty, one of the specialties, but let's say it's a bug collector, a scientific bug collector. The bug

collector of today doesn't go out with a net and the magnifying glass. He goes out with a laptop and a copy of R.

Mashey: Yeah. Okay.

Chambers: So the point is that there are these huge, huge portals that have been opened in so many fields of science from astronomy to biology to medicine through miniaturization and through the automatic collection, and the kinds of data, genetic data, for example, which didn't exist until recently. And part of the challenge of this is to integrate those with more conventional sources of data which is one of the things I think a lot of people are working on. A lot of our scholars are doing things that are related to this in various ways.

Mashey: How many of them are there?

Chambers: We have-- it's a two-year program. We support them part-time-- half time for two years.

Mashey: Is this a postdoc kind of thing?

Chambers: No. We have postdocs, too. We have postdoc program, but this program is for PhD students. Normally they'll be beyond their second year because they'll have picked a research area and be starting to work on it. So we have two-- a cohort each year. And so we just chose the cohort for this year and that's 16 people. So it's going to be 30ish people for two years.

Mashey: That's quite a big program.

Chambers: From my view, it's an almost ideal size because it's still small enough that these guys and gals, we got plenty of gals really get to know each other and get to be friends. Get to have a good time together. Get to really share stuff. And when we have meetings and discussions of topics, debate topics, it's very Lively. You get much beyond that size and it gets a little more difficult to have this kind of interaction. We'll see. It's-- I love it. It just keeps going on.

Mashey: Well, you're lucky to manage that kind of program course. Of course, there is this tradition of senior Bell Labs New Jersey folks retiring to the West Coast here.

Chambers: Yes. My distinction for what is that I'm called a senior advisor, and I'm actually not paid.

Mashey: Yes. Okay.

Chambers: So I don't run the program. There's a wonderfully able guy who runs the program, but nevertheless it is great.

Mashey: That's good. So is there anything we haven't talked about that we should?

Chambers: We've covered-- really-- well, I guess I could say a little bit more about the current situation with R.

Mashey: Yeah, that's worth.

Chambers: Yeah. And I should have emphasized that I'm just on the-- I'm still involved, but I'm just on the fringes of things. It's an interesting area, an interesting situation in that to some extent R is still owned and evolved by this group called R Core. The definition of R Core is basically that you have write permission on the master copy of the software. It's still going strong. A lot of the people are still in it from the beginning and they're responsible for the official version of R. But, of course, R is open source software so anybody can do whatever they want with it. A lot of the impetus nowadays comes from commercial company called R Studio. And various pieces of software that they have, particularly environment and the style of computing, which is collectively, termed the tidyverse. And it's the largely, not entirely, largely the work of somebody named Hadley Wickham, a young guy I've known since he was a graduate student. And, of course, there's a little bit of tension between the commercial and the academic people. But I think, in my opinion, my personal opinion, is that it's a very productive interface. I know quite well JJ Allaire who's the head of R Studio and he's a very conscientious CEO.

Mashey: Is it fair to compare this with sort of the relationship between Linux and Red Hat?

Chambers: I don't know enough about that.

Mashey: Okay. Well, I mean in the sense that Linux was completely open source. And then Red Hat, sort of said we're going to commercialize it.

Chambers: Commercialize Linux.

Mashey: Which it got bought by IBM.

Chambers: Yeah. I don't know because I don't know the people involved there.

Mashey: It sounds like…

Chambers: Technically it's the same. R was open source software to say it explicitly. You can do anything you like with it. The only thing you can't do legally with it is to build software on top of it and charge for licensing of that software. That's the no- no. So R Studio is quite careful about that. And they're quite happy with the open source situation. They they're good-- in my opinion, they're very good about making the very general style of software built on R of theirs open source. And so there's been a number of takeoffs from people in various research areas using their software to do new things like this, a thing called the tidy models, which is built on the tidyverse that kind of stuff. So, I think it's-- I think so. You know, people are people and some…

Mashey: You know, it's just interesting, you know, from a computer history viewpoint to look at some of the different industry models that have worked or social structure models because you think of Unix first being a Bell Labs internal thing, okay, right. And then, you know, percolating out to the universities and eventually took commercial stuff. And lots of people making commercial versions in one flavor or another, but then getting-- you know, having Linux come along and being a big open source type thing.

Chambers: To just clone.

Mashey: Yeah. But then still there being commercial things built around it or on top of it to fit the needs of other people.

Chambers: Yes. Well, one thing from a history point of view to keep in mind is that until 1980 AT&T was abiding by consent decree from the 1950s, which said it would not engage in computing.

Mashey: Yes. I remember the Unix license that basically said here it is, don't call us if there's bugs.

Chambers: Right. That flavor-- the decision that we <inaudible 01:26:50> <?> first, and then we reached with the legal department at Bell labs to license our software. The concept was that it was not going to be a revenue owner for AT&T except to the extent that they had a commercial license which was not hugely onerous. But basically it was just being done to share the software and to let people work on it. Now that all changed in principle when AT&T divested itself from the operating companies. And so AT&T made various efforts to get into computing, as you probably, know during the 1980s, which were almost uniformly a disaster. But that's another-- but that was unrelated to Unix, or S.

Mashey: Okay, if nothing else I think we're done.

Chambers: I think that’s it.

Mashey: Thanks a lot.

Chambers: Very enjoyable.

END OF THE INTERVIEW

