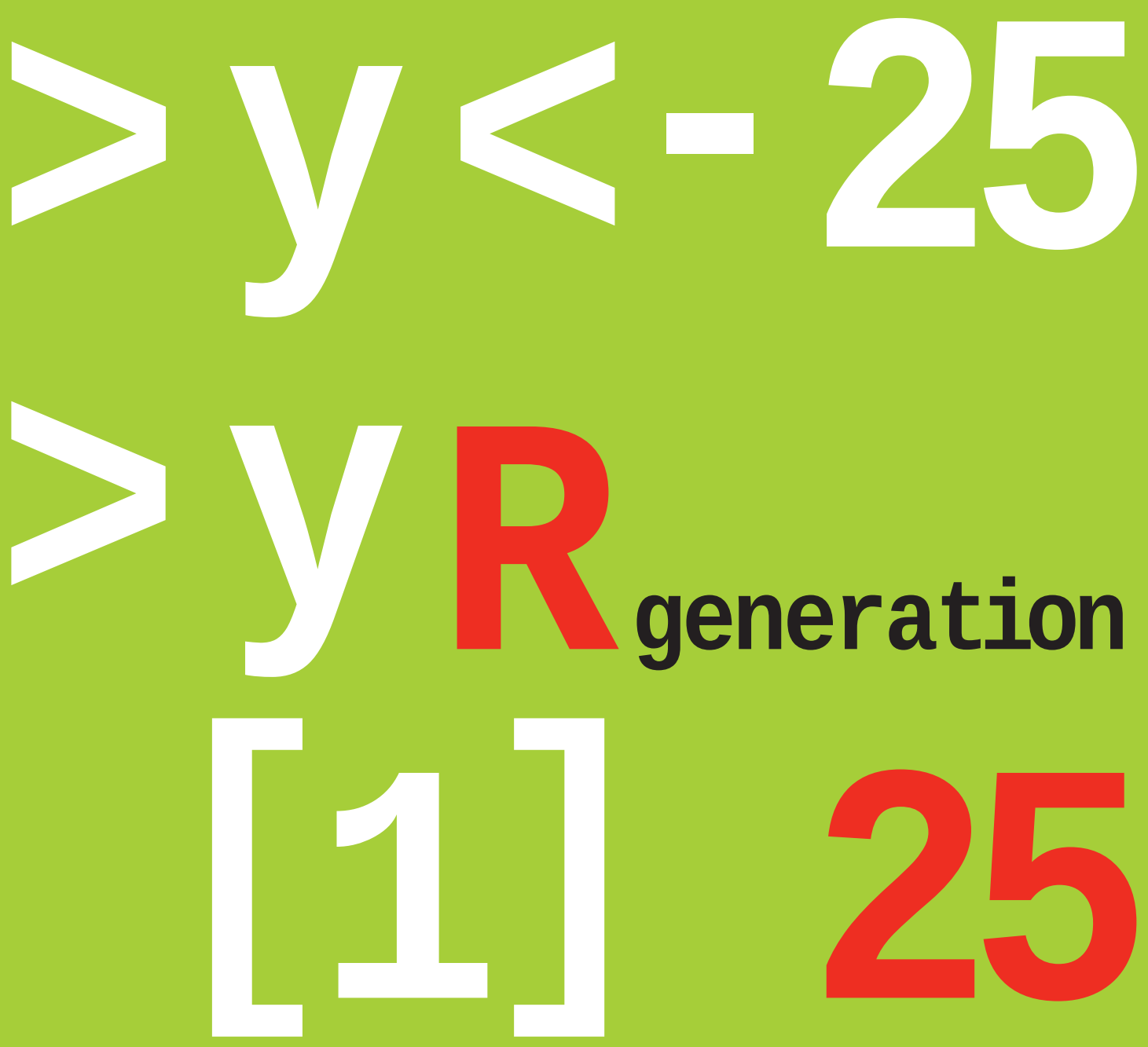

> y <- 25
> y
R generation
[1] 25

The story of a statistical programming language that became a subcultural phenomenon. By **Nick Thieme**

Beyond the age of 5, very few people would profess to have a favourite letter. But if you have ever been to a statistics or data science conference, you may have seen more than a few grown adults wearing badges or stickers with the phrase "I love R!".

To these proud badge-wearers, R is much more than the eighteenth letter of the modern English alphabet. The R they love is a programming language that provides a robust environment for tabulating, analysing and visualising data, one powered by a community of millions of users collaborating in ways large and small to make statistical computing more effective and efficient for all.

This community solidified into a vibrant culture consisting of meet-ups, hackathons, data jams, and data visualisation competitions. Consider that not only is there a Tumblr blog – accidental-art.tumblr.com – for when R visualisations go wrong in interesting ways, but that "aRt" from the blog recently featured at an art show in South Korea.

August 2018 is the 25th anniversary of the creation of R, this lingua franca of the statistics and data science communities, and here we tell the story of its birth, growth and development. The story begins quite unexpectedly, with a chance meeting between two statistics professors – Ross Ihaka, now retired from the University of Auckland, and Robert Gentleman, vice president of computational biology at 23andMe. As Gentleman explains: "There was no real intention to build anything other than a toy to play around with ideas."

It did not stay that way for long. R would bring the philosophy of collaboration in science to the distribution of code, democratising statistical computing. From an office in New Zealand, it scattered across the world into the hard drives of students and professors, Google data scientists, biologists, and more. It helped diversify data science intellectually – and it did all this without charging its congregation a dime.

R history

The history of R is one of good fortune and good choices. In 1992, Gentleman – then a professor at the University of Waterloo in Canada – travelled 8600 miles to the University of Auckland to lecture for three months. One day, he found himself needing a manual for a particularly tricky piece of software and Ihaka – still a professor of statistics in those days – was the only one in the department who had a copy. In time, both realised they shared an interest in what Ihaka calls "playing academic fun and games" with statistical computing languages.

Nick Thieme is a freelance reporter, with writings appearing in *Significance*, *Slate*, *BuzzFeed* and *Undark*. He is also a researcher with New America's Open Technology Institute, investigating fair internet usage and net neutrality.

Each had questions about programming languages they wanted to answer. In particular, both Ihaka and Gentleman shared a common knowledge of the language called "Scheme", and both found the language useful in a variety of ways. Scheme, however, was unwieldy to type and lacked desired functionality. Again, convenience brought good fortune. Each was familiar with another language, called "S", and S provided the kind of syntax they wanted. With no blend of the two languages commercially available, Gentleman suggested building something themselves.

Around that time, the University of Auckland needed a programming language to use in its undergraduate statistics courses as the school's current tool had reached the end of its useful life. There was one major caveat: the program needed to run on Macintosh. According to Gentleman, the Department of Statistics took inventory and decided "that thing Ross and Robert are working on", which happened to run on Macintosh, was better than their current language. The professors called it R, as both a nod to S and in reference to their forenames.

Ihaka and Gentleman kept the project secret from the wider community until August 1993, when an email to the S-news mailing list drew it into the public eye. A Canadian professor had a familiar problem: he needed a Macintosh version of S. Ihaka decided it was time to let R see the light of day, telling the list what the two professors had developed, while asking that people not "bug us about it" because it was not quite ready. Soon after, a usable version of R appeared on StatLib, an online system for distributing statistical software and data.

Though the R we have today is free software, in the mid-1990s Ihaka and Gentleman were seriously considering turning R into a commercial product, going so far as to buy a book on forming a business and asking people familiar with the field what it would take to sell the software. But ultimately the idea of selling R struck them as more trouble than it was worth. "We thought we'd sell five copies to our friends and that would be it," says Ihaka. Input from Dr Martin Mächler, an ETH Zurich statistician who had found R on StatLib, also helped push R in the direction of free software. Mächler was involved in the open-source software community and believed that everyone, regardless of income, should have access to it.

Ihaka and Gentleman agreed with the idea of making R free software – meaning that people would be free to use, change, and distribute it as they like. In 1995, the duo made R's source code available under a free software licence. Mächler joined Gentleman and Ihaka as one of the primary developers of R, taking bug submissions from the public and implementing

improvements users drew from the source code. At first, the changes added much needed bare necessities – such as avoiding complete crashes from loading memory, and ensuring calculations were correct. Soon, though, users were adding completely new functionality, making R faster, easier to use, and able to handle more data.

R community

As the language improved, more users joined – and more users meant less room for bugs to hide. As fixes and functions poured in, the names of the submitters began to look familiar. Usual suspects submitted so frequently that Ihaka and Gentleman gave them the ability to edit R's source code directly because it was easier than managing all the changes themselves. By mid-1997, 11 people – including Ihaka, Gentleman, Mächler, Peter Dalgaard, Kurt Hornik, Friedrich Leisch, and Thomas Lumley – had the keys to R's source code. The group fashioned themselves the "R Core" team.

"The users were the developers in those days," Ihaka says, and as more of them joined the community, they needed a place to show off what they had done and download contributions they found useful. In March 1997, Hornik and Leisch, of the Vienna University of Economics and Business, made a Herculean contribution to the R Project by building the Comprehensive R Archive Network (CRAN). This network made the essential information and files of R available for download in one place. Most importantly, CRAN meant users could browse packages – R's version of code libraries – and download the ones they needed.

CRAN makes R shine. Most of the functionality of R is contained in the packages stored in CRAN, which can be loaded and used when needed. This makes R more versatile than other statistical software. Closed-source software, such as SAS and SPSS, can only be updated by their official developers, whereas R has a community churning out updates all the time.

But why use packages at all? Base R, as the vanilla version of R is called, provides the ability to load, export and transform data, to access probability distributions and mathematical functions, to run linear models on data, and to visualise results in visually pleasing ways. Most importantly, it provides a programming syntax to build larger tools by tying these individual functions together. If you need to use an optimisation algorithm, or want to interface with a website, you can always write enough R code to enable those functionalities. But writing code that enables you to solve your ultimate problem is different from actively attempting to solve that problem, so the time spent writing code is not directly productive. That is why packages are useful. Instead of taking the time to rewrite hundreds or thousands of lines of code, users can download the work of others.

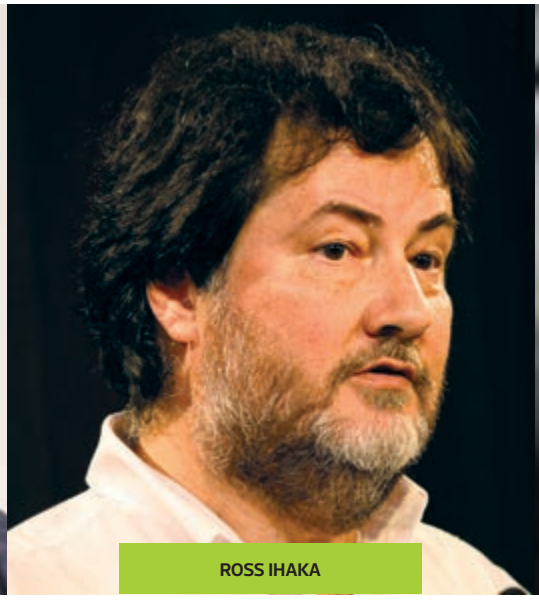
The output of this community of user-developers has grown phenomenally over the past quarter-century. But at the turn of the millennium, the language was not yet the darling it is today. As Lumley, of the University of Auckland and the R Core team, puts it: "I probably got tenure despite working on R, not because of it." The birth of data science changed all that.

In 2000, the R Project released R version 1.0.0, the first version they felt was ready for public usage. The following year, several influential statisticians published papers about data science, and 2003 saw the first academic journal dedicated to this growing field. For those people now identifying as data scientists, R, CRAN and the wider community provided the means to explore and familiarise themselves with statistical tools and techniques. In turn, those data scientists added packages to CRAN to help R manage important data types and models from fields as diverse as ecology, linguistics, bioinformatics and network science.

Gradually, knowledge of R became a near-prerequisite for working in data science jobs, and as the milieu of R users grew to include anyone with data needs, the original community

BELOW The original developers: R's creators Robert Gentleman and Ross Ihaka, and "R Core" team member Peter Dalgaard.


ROBERT GENTLEMAN


ROSS IHAKA


PETER DALGAARD

began to mix with the emerging data science community, the life science community, and others. As a result, the idiosyncrasies of the R community – its need to understand every piece of a software, including the technical nitty-gritty, and its desire to build software as much as use it – started to mutate. This new, expanded user base was less interested in the mechanics of R than in what R allowed them to do, and the less this new group of users thought about writing R code the more they could think about their own interesting problems. This new community found its champion in Hadley Wickham.

Wickham needs as much introduction to the R community as Kanye West does to *Rolling Stone* readers. He is currently chief data scientist at RStudio, but he is famous for his basically ubiquitous packages including *dplyr* and *ggplot2*. Clear writing is clear thinking, and Wickham sees his packages as a way for users to write "the notation of data science that makes hard problems seem straightforward". His packages allow someone reading a block of code to comprehend the ideas baked into it on first inspection; these packages also get the time-consuming task of programming out of the data worker's way. Users find his contributions near-essential. Downloads for *dplyr* and *ggplot2*, plus *purrr* and *devtools*, totalled about 1.5 million in July.

The collection of packages championed by Wickham's work is typically called the "tidyverse", and Wickham hopes they make it easier for people to join the R community and give them reasons to stick around once they are in. The "little frictions" inherent in any kind of data analysis "are real and it's worth trying to reduce them", he says.

For example, a package called *Reshape* recasts messy data into a canonical shape that other tidyverse packages are equipped to deal with. *Dplyr* provides simple methods for organising analyses and applying more complex transformations to data. Meanwhile, Wickham's famous *ggplot2* package provides a "grammar of graphics" to help users build visualisations. Most of the capabilities in the tidyverse are also available in base R, but the tidyverse makes them simpler to use and provides a more intuitive, more readable syntax (see "Welcome to the tidyverse", page 19).

Dr Julia Silge, a data scientist at Stack Overflow and author of the package *tidytext*, which applies tidyverse principles to natural language processing, is an example of tidyverse talent. "If it were not for *dplyr* and *ggplot2*, I probably would be a Python developer," she says, referring to the other primary programming language used by data scientists. Tidyverse principles allow Silge and other developers to "piggyback on the mature set of packages that already exist", leaving them to focus on the specific problems they want to solve.

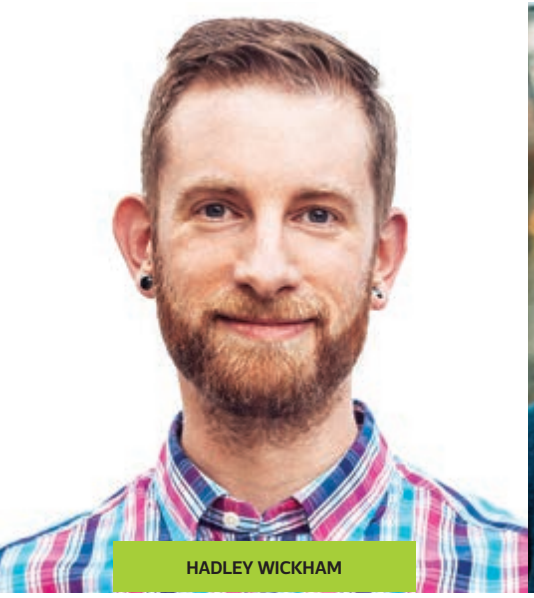
This "piggybacking" that tidyverse enables is less a break with the philosophy of base R and more a natural extension of it. The magic insight of R's creators was that by facilitating the creation and distribution of packages containing functions that make sense and are easy to use, the learning curve for non-experts could be infinitely flattened out.

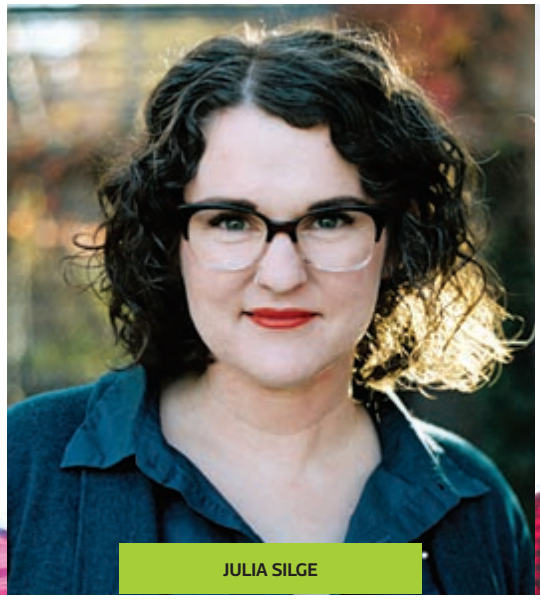
As a programming language, R has always been usable by non-computer scientists; the tidyverse further lowered barriers to entry. So while programming in Python, Java and C remains dominated by computing specialists, R has added users in the life sciences and humanities – and these new users have very different demographics.

R culture

Like most coding and software communities, R is male-dominated. But over the past several years, Gabriela de Queiroz and thousands of others have sought to change that. The catalyst for action came in 2012, when de Queiroz relocated from Brazil to San Francisco to attend California State University for her master's in statistics. To get involved in the data science community, she attended data science meet-ups almost daily. But while she would show up to the meetings, she noticed she was not contributing to conversations or asking the questions

BELOW The new developers: RStudio's Hadley Wickham, *tidytext* creator Julia Silge, and R-Ladies founder Gabriela de Queiroz.


HADLEY WICKHAM


JULIA SILGE


GABRIELA DE QUEIROZ

that came to mind. Instead, she "was there learning but not interacting or taking full advantage because I didn't see myself represented in the audience".

De Queiroz decided she wanted a space to learn about statistics and data science where reservation was replaced by an atmosphere of mutual support. On 1 October 2012, she founded the first chapter of R-Ladies in San Francisco as a place where women and others could feel comfortable asking questions and making suggestions. R-Ladies meet-ups were soon held in Taipei, Minneapolis–St Paul, and London. Today there are chapters in 122 cities and 28 000 members around the world.

The R-Ladies mission is "to achieve proportionate representation by encouraging, inspiring, and empowering people of genders currently underrepresented in the R community" (bit.ly/2t5MfnZ). But quantifying the precise degree of that underrepresentation is difficult. There are no reliable figures on the active R community as a whole, and until a formal survey is conducted we can only impute how many users identify as a gender other than male. For instance, a 2016 analysis of the first names of CRAN package authors (bit.ly/2t4yc2c) estimated the proportion of female package authors to be somewhere between 11% and 15%. The R Foundation Taskforce has also collected demographic data from its useR! series of conferences; it found that while gender equity had improved, sometimes tremendously, the conference was starting from a very low baseline. For example, the percentage of invited talks given by women increased from zero in 2004 to 34% in 2016.

As de Queiroz says: "It's so well known how diverse the community is not, historically. It has to change." R-Ladies and others are working to achieve this through a number of means, including publishing lists of women statisticians who can be invited to speak at conferences, encouraging more women to submit papers, and providing meetings and spaces to support and develop new R users.

These collective efforts seem to be having an effect. Many of those interviewed for this article mentioned how much more welcoming R's community and culture had become in recent years. But it was Silge who made this point clearest by comparing it to her time as a graduate student in astrophysics. Back then, she says, "I had my fair share of experiences of asking a question and having a response from a technical community be dismissive, and when I was making this transition to data science, I was mentally girding myself for tough times." Silge joined the R community in 2015 and since that time, she says, "my experience … has been almost entirely the opposite".

This change in culture is nothing but positive for R. The more welcoming it is, the more diverse the community becomes, resulting in a new "massive pool of talent that was previously untapped", says Wickham.

R future

We have now caught up with R's story so far, but it is by no means the end of the tale. What might the future have in store?

Lumley was unsure if another computing language would be coming to bury R anytime soon, but he felt that any successor would have to absorb CRAN and its stockpile of code. Gentleman agreed, saying: "There are really good algorithms in R, and no one should be reimplementing those."

The future of CRAN is a popular topic for speculation as the network is starting to creak under the weight of its own success. The archive now holds more than 12 000 packages and is growing near-exponentially. From January to May 2018, a median of 21 packages were added or updated per day. From the perspective of the R Core team, there simply are not enough hours in the day to review and fully comment on every submission. But users understandably want transparency and personalised feedback when their packages are rejected, so this causes friction.

With CRAN growing unabated and – in Peter Dalgaard's words – "the original Core team approaching pensionable age", the maintenance of R and CRAN will at some point need to change. R still works well when pulling packages from websites like GitHub, but downloading packages internally, through CRAN, is what made R great, and this should surely remain part of R's future. The obvious solution – and one the community seems willing to try – is to increase the number of people who can review and comment on packages.

In terms of adding new functionality to R, both Silge and Gentleman called for more seamless communication between R and other languages. Silge mentioned using Apache Arrow, a new method for cross-language development, to quickly move data from R to Python to Spark to Tensorflow, for example. Interestingly, in April Wickham started dedicating time to Ursa Labs, a project designed to create a run-time environment for cross-language data science that is built on Apache Arrow. Probably related, both Silge and Wickham mentioned exploring how to represent tree-type data in tidy formats. Someone looking for R developments in the future should expect to find tidyverse packages for relational and hierarchical data.

Ultimately, of course, the future of R will be determined by its community – the people who, over the last quarter of a century, have donated years of their lives to tweaking the source code, crafting clever packages and helping new users get started. These donations of time and effort did not come with the promise of future monetary rewards. They were made by people who wanted to see R flourish as a programming language, and because its community meant something to them.

R is free, open-source software that was created for fun, reared by committee, and developed by the masses. That such a software could survive and flourish for 25 years is a credit to its quality, to its creators, and to its users. "People in the past would have said you couldn't do something like this," says Lumley. "Now it's clear that you can." ■

The pick of R packages

To show more of what R can do, we asked several of our interviewees to write about their favourite R packages. To read their submissions, head to significancemagazine.com/594. And if you are an R user, please tell us about your favourite R package in the comments.

Welcome to the tidyverse

Here's an example of how tidyverse packages and principles streamline the data analysis pipeline, using a "messy" data set of video game sales and reviews.

Data input

Tidy data has three principles: (1) each variable forms a column; (2) each observation forms a row; and (3) each type of observational unit forms a table. We see in Table 1 that the first rule is violated twice. We have variable levels masquerading as variable titles because "Critic" and "User" are labels of an unseen variable, "Rater". We also have two variables (the rating of a game and the number of reviews) crammed into each of the Critic and User columns. Another issue is that critic scores are out of 100, and user scores out of 10. When combining these variables, we need to readjust the scales.

We can fix these issues using functions in *dplyr*, a package designed expressly for the purpose of formatting and transforming data (see Table 2). By mutating columns into the desired statistics and reshaping data into useable formats, *dplyr* can be used in a preprocessing step that feeds directly into the *ggplot* package for visualisation.

Graphical analysis

The graphics below were made using the *ggplot2* package in R and an expanded version of the data set shown in Tables 1 and 2. Figure 1 compares sales in millions of units for the top 10 highest-selling games, while Figure 2 plots user scores against global sales for a wider range of titles. Although each of the plots looks different, the R code is largely identical. This is thanks to Wickham's "grammar of graphics", which was introduced with *ggplot2*.

The basic idea is this. All plots are comprised of three components: data, a mapping of that data to visual elements, and a geometric shape that represents the mapped data. These components, along with concepts like scales, statistical transformations, and coordinate systems, make up the "grammar of graphics".

What that means from a practical perspective is that once users learn how *ggplot* works, they can create unique plots by changing only a little bit of code, often only part of a function name. For instance, in our examples, the bar plot uses "geom_bar()", plus "fill = Publisher" for colour, while the scatter plot uses "geom_point()" – those are the major coding differences between the two graphics.

TABLE 1 "Messy" data on video game sales and review scores. Sales data from VGChartz; review data from Metacritic. Data set sourced from Rush Kirubi, via Kaggle (bit.ly/2KNZgM9).

Name	Publisher	Global sales (million units)	Critic	User
Wii Sports	Nintendo	82.53	Score = 76, Count = 51	Score = 8, Count = 322
Mario Kart Wii	Nintendo	35.52	Score = 82, Count = 73	Score = 8.3, Count = 709
Wii Sports Resort	Nintendo	32.77	Score = 80, Count = 73	Score = 8, Count = 192
New Super Mario Bros.	Nintendo	29.80	Score = 89, Count = 65	Score = 8.5, Count = 431
Wii Play	Nintendo	28.92	Score = 58, Count = 41	Score = 6.6, Count = 129

TABLE 2 "Tidy" data on video game sales and review scores.

Name	Publisher	Global sales (million units)	Rater	Score
Wii Sports	Nintendo	82.53	Critic	76
Wii Sports	Nintendo	82.53	User	80
Mario Kart Wii	Nintendo	35.52	Critic	82
Mario Kart Wii	Nintendo	35.52	User	83
Wii Sports Resort	Nintendo	32.77	Critic	80
Wii Sports Resort	Nintendo	32.77	User	80
New Super Mario Bros.	Nintendo	29.80	Critic	89
New Super Mario Bros.	Nintendo	29.80	User	85
Wii Play	Nintendo	28.92	Critic	58
Wii Play	Nintendo	28.92	User	66

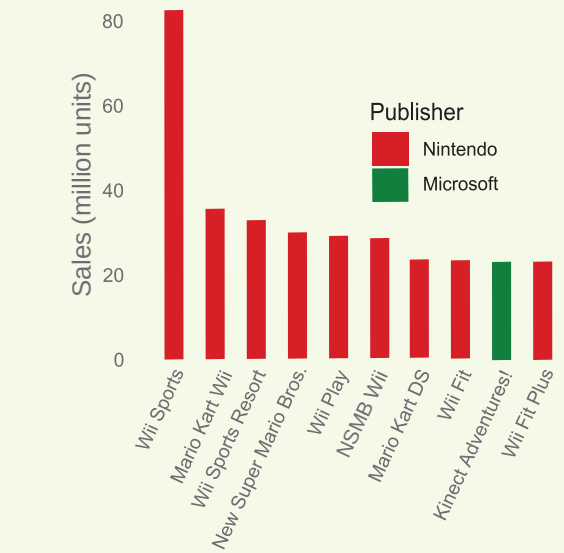


FIGURE 1 Global sales (in millions of units) for the 10 highest selling video game titles.

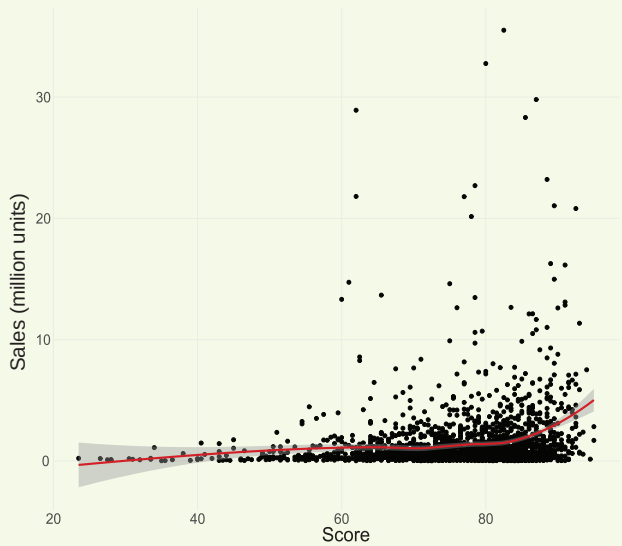


FIGURE 2 Global video game sales (in millions of units) versus user scores.