DEVELOPER ECONOMICS

STATE OF THE DEVELOPER NATION Q1 2017

The latest trends
from a survey of 21,200+ developers

http://vmob.me/DE1Q17

vision mobile

PUBLISHED MARCH 2017

About VisionMobile ™

VisionMobile™ is the leading analyst company in the app economy and the developer ecosystem. Our surveys and app analytics track the changing landscape of mobile, IoT, desktop, and cloud developers.

Developer Economics is the most global app developer research & engagement program reaching 16,500+ respondents from 145 countries around the world. Our research program tracks developer experiences across platforms, revenues, apps, tools, APIs, segments and regions.

Our mantra: We help the world understand developers – and developers understand the world.

VisionMobile Ltd.
90 Long Acre, Covent Garden,
London WC2E 9RZ
+44 845 003 8742

https://www.visionmobile.com/blog
Follow us on twitter: @visionmobile

Terms of re-use

Contents

Also by VisionMobile

ABOUT THE AUTHORS

Mark Wilcox
Senior Business Analyst

Mark has over 13 years of experience in mobile software across a variety of roles. He received his first computer at 4 years of age and wrote his first program at 7. He will probably always be a developer at heart. A growing interest in economics and business models has led him to work with VisionMobile as a Business Analyst, whilst still keeping his development skills up to date on development projects.

You can reach Mark at:
mark@visionmobile.com
@__MarkW

Stijn Schuermans
Senior Business Analyst

Stijn has been the lead Internet of Things researcher in the VisionMobile team since 2012. He has authored over 20 reports and research notes on mobile and the Internet of Things. He focuses on understanding how technology becomes value-creating innovation, how business models affect market dynamics, and the consequences of this for corporate strategy.

Stijn has a Master's degree in engineering and an MBA. He has over 10 years' experience as an engineer, product manager, strategist and business analyst.

You can reach Stijn at:
stijn@visionmobile.com
@stijnschuermans

Christina Voskoglou
Director of Research and Operations

Christina is responsible for all VisionMobile's research products and heads the analyst and operations teams. With more than 18 years of experience in data mining, BI and CRM design, she leads research planning and methodology, survey design, data analysis, insights generation and research commercialization. Christina is also behind VisionMobile's outcome-based developer segmentation model and is the leading VM researcher in Machine Learning and Data Science.

You can reach Christina at:
christina@visionmobile.com
@ChristinaVoskog

Alexandre Sobolevski
Software Engineer at Plotly

Alexandre is a software engineer at Plotly and an engineering graduate from McGill University. He previously worked at Siemens developing analytical and data visualisation tools. He is co-author of the Plotly Database Connector, enjoys developing tools for data scientists, and loves to pack a backpack and go on a trek.

You can reach Alexandre at:
@AlexandreSobo

TABLE OF CONTENTS

About Developer Economics

Welcome to the State of the Developer Nation Q1 2017 report, based on the 12th Developer Economics global survey wave. Produced by VisionMobile, Developer Economics is the leading research programme on mobile, desktop, IoT, cloud, augmented and virtual reality, and machine learning developers as well as data scientists, tracking the developer experience across platforms, revenues, apps, languages, tools, APIs, segments and regions.

The 12th Developer Economics global survey wave ran in November and December 2016 and reached more than 21,200 developers in over 160 countries. This research report delves into the key developer trends for 2017. We discuss the adoption of technologies and programming languages across AR/VR, web, IoT, data science, and cloud. We also look at financial opportunities for developers.

The report focuses on six major themes – each with its own visualisation, showing how the data lends insight into the developer community.

1. For the first time in the history of Developer Economics, we asked developers about their salaries. This allows us to explore what drives higher wages, and how developers can find the most lucrative opportunity.
2. In Augmented and Virtual Reality, a productive future seems further away than one might hope. Meanwhile, a look at the programming languages that AR/VR developers use reveals how technology vendors are jostling for position to capture this attractive market.
3. Such technology battles can last a long time. In web development, The Angular and React Javascript frameworks, built by two of the most powerful companies on earth, are fighting for world (or should we say 'web'?) domination. The current winner might surprise you!
4. Talking of technology battles: Amazon Web Services is now making so much money that they can't spend their profits fast enough, while being in a price war with their public cloud competitors. We take a look at how they're all faring.
5. Over in the Internet of Things arena, the developer tooling industry has yet to find a stable footing: IoT 'platforms' are highly fragmented, underused, and by and large fail to connect developers with an addressable market.
6. Data science forums are buzzing with the same question over and over again: what's the best language for machine learning? Being data scientists ourselves, we couldn't help but run a few models to see which are the most important factors that are correlated to language selection.

We hope you'll enjoy this report and find the insights useful! If you have any questions or comments, or are looking for additional data, you can get in touch with Sofia Aliferi, Marketing Manager for VisionMobile at sofia@visionmobile.com. You can download this free report at www.DeveloperEconomics.com/go

— Mark, Stijn, Christina, Alexandre, Christos, Matos, Alex, Eitan, Andreas, Emilia, Dimitris, Kosmas, David, Vanessa, Sarah, Nick, Andrea, Sofia, Sam, Andrea W, Michael and Chris at VisionMobile.

Thank you

We'd like to thank everyone who helped us reach 21,200+ respondents for our survey, and create this report. Our Research Partners – Intel and Microsoft, our Media Partners, who are too many to name here. Also, a special thanks to VisionMobile affiliates including Caven Cade Mitchell, and Peter Cooper for their support.

PARTNERS

VisionMobile is very lucky to be supported by the entire developer community: from the largest Internet and software companies to the smallest local Meetups. Partnering with these organisations, big and small, ensures that there is a representative sample across all developers so that something valuable is delivered back to the community. This is a list of the top contributing communities that helped us reach 21,200 responses in the State of the Developer Nation Q1 2017.

DIAMOND

amazon

Amazon Appstore powers game and app publishing for Android and Amazon Fire devices, giving developers instant access to millions of customers across 236 countries and territories worldwide. Committed to keeping publishing easy for Android-based apps and games, the Amazon Appstore o ers an array of services making it a complete end-to-end solution for developers from building and testing through marketing and monetisation.
https://developer.amazon.com/

outsystems

Thousands of developers worldwide trust OutSystems, the number one low-code platform for rapid application development. Engineers with an obsessive attention to detail crafted every aspect of the platform to help organisations build enterprise apps faster. It is the only solution that combines vis ual development with advanced mobile capabilities, enabling the delivery of entire application portfolios that easily integrate with existing systems.
https://www.outsystems.com/

sitepoint

SitePoint is a hub for web developers to share their passion for building incredible Internet things. Founded by Mark Harbottle and Matt Mickiewicz in 1999, SitePoint is for web professionals, by web profes- sionals: developers, designers, programmers, product creators and entrepreneurs alike. Since 1999, we have published more than 20,000 books and articles that help more than 10 million readers per month become better web developers.
https://www.sitepoint.com/

51CTO

Founded in 2005, 51CTO.com has established itself into a high quality comprehensive IT service platform in China o ering IT information, online and o ine IT training, HR services and more. By the end of 2016, 51CTO.com had accumulated 14 million IT professional users in China from all di erent fields both technologically and geographi- cally, and has been well accepted as a leading professional IT platform in China.
http://www.51cto.com/

ubuntu

Canonical is the company behind Ubuntu, the leading OS for cloud operations. Most public cloud workloads use Ubuntu, as do most new smart gateways, switches, self-driving cars and advanced robots. Canonical provides enterprise support and services for commercial users of Ubuntu. Established in 2004, Canonical is a privately held company.

https://www.ubuntu.com/

Microsoft

Microsoft is the leading platform and productivity company for the mobile-first, cloud-first world, and its mission is to empower every person and every organisation on the planet to achieve more.

https://www.microsoft.com

intel

At Intel we innovate at the boundaries of technology to make amazing experiences possible for business and society, and for every person on Earth.

https://software.intel.com

PLATINUM

THE LINUX FOUNDATION, ANDROID WEEKLY, INMOBI, DZone SOFTWARE, SellMyApp

GOLD

APPTRACTOR, REVMOB, ptc, ARDUINO, Google, mozilla

SILVER

EasyAR, BELATRIX, 开源中国 oschina.net, I PROGRAMMER, chupamobile, cooperpress, council

BRONZE

LibHunt, LEAPUP, CSDN, redhat, codeplace, FICO, Xamarin, IMK, NLAST, JS GREECE, Monaca

KEY INSIGHTS

- Developers who work in areas with a higher technical complexity or in very young sectors - and therefore with higher barriers to entry and ultimately fewer developers doing it - generally earn more. In Western Europe, for example, the median backend developer earns 12% more than the median web developer; a machine learning developer makes 28% more. Web and mobile development are the most commoditised.
- We're still a long way off a global market for developers. The median earnings of web developers in Western Europe are half of those of their North American counterparts; web developers in other regions earn half again. This opens up arbitrage opportunities for developers willing to work remotely.
- C# is the most popular primary programming language amongst Augmented and Virtual Reality developers, preferred by 30% of them. This is followed by C/C++ (16%) and Java (15%). Interestingly, professionals are more likely to use C# or C++ in comparison to hobbyists.
- Almost 90% of AR/VR developers would be considered juniors by other industries' standards, having less than 2 years experience. The industry consists of many newcomers who are inexperienced in the field - they will not be deeply invested in any tools, technologies or platforms, so any vendor has the potential to establish market leadership with the right product.
- 48% of web developers are currently using a third-party library or framework other than jQuery as their primary way of doing front-end web development. Angular and React account for 30%, leaving all the others fighting for the remaining 18%. Indeed front-end web development is such a fragmented space that no other single library or framework accounts for more than 2% of primary usage.
- Facebook's React appears to dominate Google's Angular in online discussion and open source activity. However, not only is Angular 2.x the primary framework for about as many developers as React (10% vs 9% globally), but Angular 1.x is still the most popular overall by a slim margin (11% use it as their primary framework). In total those using one or the other version of Angular number more than double those using React.
- Amazon Web Services (AWS) is the most popular primary cloud hosting at every company size. For the smallest companies (1-5 employees) where Amazon has just a 15% share, they face very credible competition from Microsoft (12%), Google (11%), and Digital Ocean (10%). However, when we look at larger companies, Amazon's share grows to 26-27% at every size, Microsoft stays in the 11-13% range, while Google fades along with Digital Ocean. Google has just a 5% share of companies with more than 5,000 employees, and Digital Ocean just 4% at the same size.

- AWS is also the most popular primary cloud host with developers regardless of targeted audience, although strongest with backend developers who target large enterprises, of whom 29% are primarily using AWS. Microsoft shows greater strength equally with developers who target large enterprises, and those who target small to medium businesses (14% each). They are weaker with those targeting consumers (11%) or professionals (9%). Google shows the opposite pattern being strongest with developers who target consumers (10%) but only half as popular with those who target large enterprises or internal employees.
- Despite the proliferation of IoT platforms and other tools, the IoT tool market is still underdeveloped and heavily fragmented. IoT developers use comparatively fewer tools than their colleagues in other software sectors. 11% of IoT developers don't use any of the tools in our list.
- Professional IoT developers use more tools than amateurs, but they are IoT solution components, not platforms that provide developers with addressable markets. 38% of IoT pros use IoT Cloud Platforms, compared to 24% of amateurs, but for developer ecosystem plays like Smart Home, smartwatch, and voice platforms this gap doesn't exist.
- Our data suggests that Python is now recognised as the native language of machine learning, while R is in most cases a complementary language. Python leads the pack of machine learning languages, both in terms of usage (57%) and prioritisation (33%). R comes fourth in usage (31%) and is prioritised by only 5% of developers.
- There is no such thing as a 'best language for machine learning' and it all depends on what you want to build, where you're coming from and why you got involved in machine learning. Developers building algorithms for customer support management, fraud detection or network security in large organisations use Java more than others. In areas that are less enterprise-focused, such as natural language processing (NLP) and sentiment analysis, developers opt for Python. Python is prioritised the most by those for whom data science is the first profession or field of study (38%).

1 THE MOST LUCRATIVE OPPORTUNITY

As a software developer, what is the most lucrative opportunity you could be working on? This is a very relevant question to ask. Software skills are generally scarce and good developers are highly coveted. Furthermore, developers are mobile, in the sense that the nature of their trade allows them to work from remote locations quite easily and marketplaces for their services are well established. So which project should you pick?

There are many reasons why someone might prefer one job over another, but let's be honest: developers deserve to get paid well, given their important position in the global value chain. For the first time in 12 editions, we asked developers in our survey how much they earn in salaries or contractor fees. The results are in and from the data we learn several insights that can help developers improve their paycheck, and conversely, provide opportunities for organisations to find talent.

Scarcity of skills drives up the price for developer services.

First, there are enormous differences in how much developers in each region and software sector earn. The best earning developers in our survey - those in the top ten percent - often earn tens and sometimes hundreds of times as much as the least well-off, i.e. the bottom decile. Part of this gap is location driven. We'll come back to that shortly. This said, we can only conclude that a developer's skill, knowledge, and reputation do matter. Investing in them will pay off.

Talking of skills, developers who work in areas with a higher technical complexity - and therefore higher barriers to entry and ultimately fewer developers doing it - generally earn more. Developers that work on cloud computing and other backend services report higher salaries than those working on front-end web apps. Machine learning specialists make even more than the backend folks. In Western Europe, for example, the median web developer has a yearly gross salary of $35,400 USD, the median backend developer earns $39,500 and a machine learning developer makes $45,200. This relationship is seen across regions and also at higher wage levels. Web and mobile development are the most commoditised; there is a fairly low barrier to start making simple apps or websites, and these tasks are relatively easily outsourced to other regions.

Scarcity of skills drives up the price for developer services. This is also true for new, emerging areas of development, like Augmented and Virtual Reality, or the Internet of Things, but only at the top end of the scale. The best developers in emerging areas earn top dollar, while the bottom half of the developer population makes less than their counterparts in more established sectors. Let's compare Augmented Reality (AR) with backend developers in North America as an example. The median wage for an AR developer in that region is $71,000 USD, a good bit less than the $79,200 that the median backend developer makes. At the top end, however, AR development is more lucrative. At the 75th percentile, the AR developer is paid $132,300 and the backend developer $122,800. At the very top (90th percentile), the difference is even more pronounced: $219,000 for AR, $169,000 for backend. The reason for this wide range of salaries is that markets like AR/VR or IoT are still commercially underdeveloped. Companies that are early adopters pay large sums for skilled developers, who are scarce. At the same time, less experienced developers are attracted by the hype. Their compensation suffers both from a lack of relevant skill and from a lack of companies that are hiring in the early market.

Again this pattern repeats across regions. The exception is South Asia. The outsourcing model that drives software development in that region seems to be built on maintaining legacy code and developers there are less involved in emerging innovations (a conclusion that's also supported by our developer population sizing research).

We started this chapter by saying that developers can market their services location-independently if they choose to. However, it's clear from the data that we're still a long way off a global market for developers. The median web developer in North America for instance earns $73,600 USD per year. A Western European web developer earns half of that - $35,400 USD - although recent exchange rate shenanigans due to Brexit and the Euro-crisis will have affected that comparison. Web developers in other regions earn again half of that: between $11,700 in South Asia and $20,800 in Eastern Europe. Not just the region of the world you live in matters, but also the country and even the city you call home.

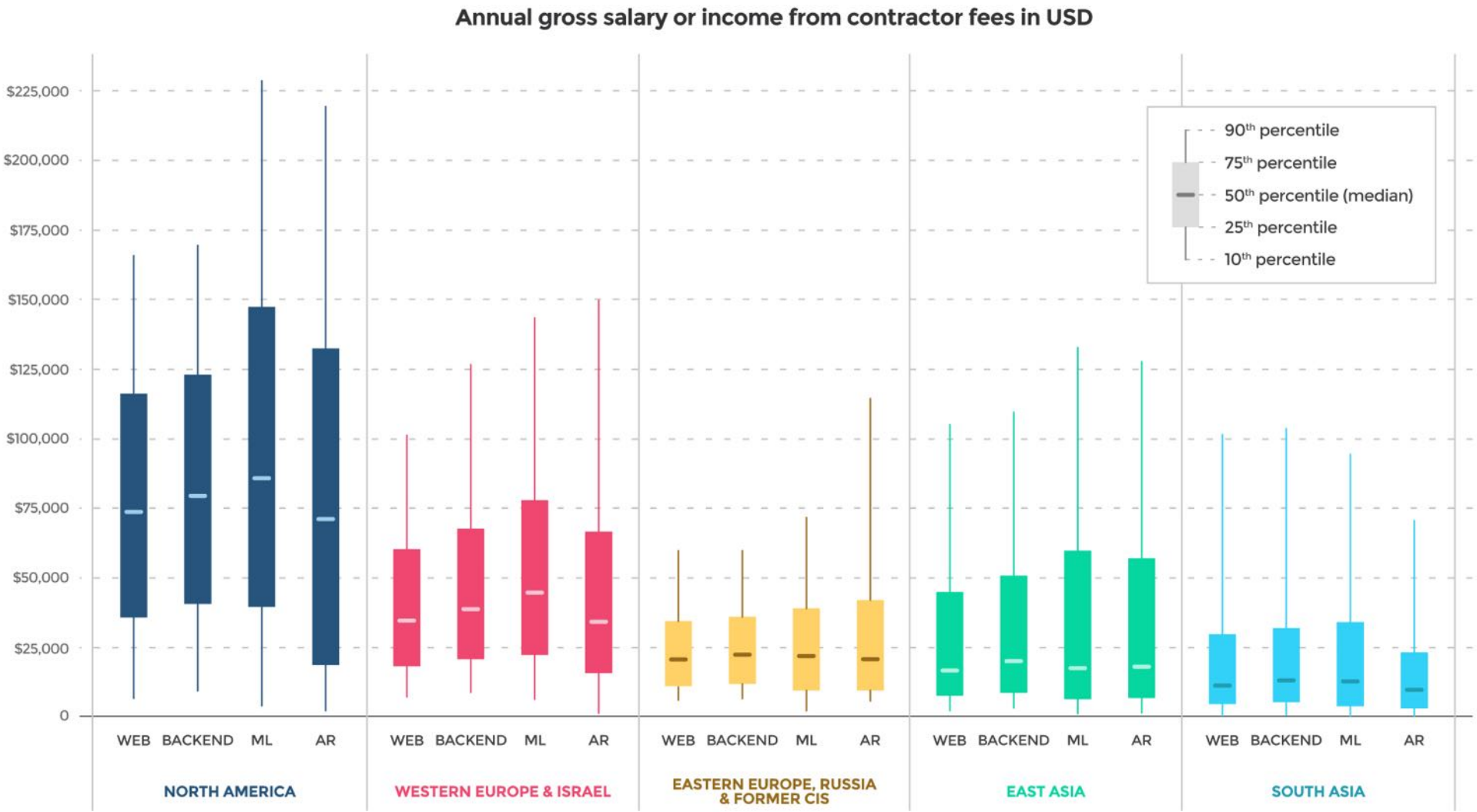
> *We're still a long way off a global market for developers.*

This opens up opportunities for organisations who will accept remote workers. You can hire a top 10% Eastern European backend developer for less money than the median North American wage in that sector. For developers, it means that brushing up your English skills and looking for opportunities beyond your backyard can be very interesting indeed. Developers who take that leap and seek opportunities that pay to international standards are in the minority. This explains why top wages in emerging regions (Asia, the Middle East, Africa) are so exuberantly high compared to local standards. A Western developer in the top decile earns about three times as much as the median wage in his sector and region. In the emerging world, top wages are seven to ten times the median. The best developers in those regions work for multinationals or sell their services on international marketplaces, while most stay employed locally, at much lower remuneration levels.

So what's a developer to do if you want to move up in the world, financially? Invest in your skills. Do difficult work. Improve your English. Look for opportunities internationally. Go for it. You deserve it!

LEVEL OF DIFFICULTY AND NOVELTY DRIVE UP TOP WAGES, BUT NOT BOTTOM ONES

Distribution of developer salaries by region and development area (n = 15,432)



ONLY DEVELOPERS WHO ARE PROFESSIONALLY INVOLVED IN EACH SECTOR ARE SHOWN.

Web: Web apps
Backend: Backend services (including cloud computing)
ML: Machine learning and data science
AR: Augmented Reality

vision mobile

Source: State of the Developer Nation Q1 2017 | http://vmob.me/DE1Q17 |

2 AUGMENTED & VIRTUAL REALITY: A HOBBYIST PLAYGROUND

Interacting, sharing, accessing. In his recent book "The Inevitable", Kevin Kelly - the founding executive editor of Wired magazine - counts twelve such technological imperatives. These imperatives guide the process of how new technologies should be turned into applications that are as useful as possible to our society. Built on the three imperatives mentioned, Augmented Reality (AR) and Virtual Reality (VR) are prime candidates to create the next mass markets of engaging and exciting tools.

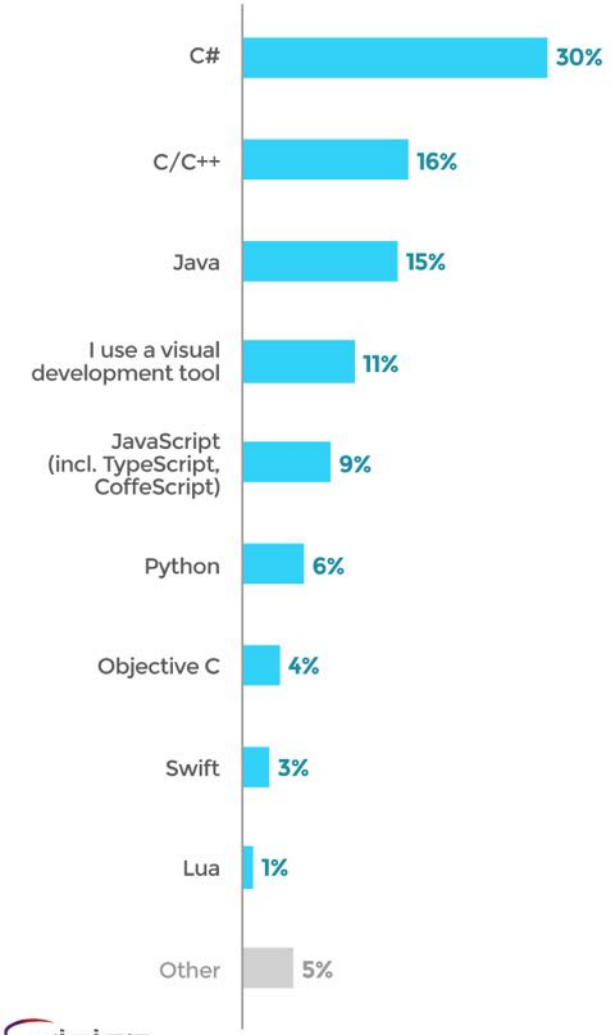
In this chapter we take a look at developers in the AR and VR industry in order to understand where it is headed. First, we look at how professional and hobbyist AR/VR developers differ in experience, primary programming languages and age. Second, we explore the various application categories in which projects compete for the pole position within the next billion-dollar technology market.

A large fraction of the global developer population have some involvement in AR or VR. In fact, of the 21,200+ developers we surveyed, 31% said they were involved in AR or VR projects either as a hobbyist or professional. Both AR and VR markets are not very mature. Of those involved in AR or VR projects, hobbyists are a large majority. In fact, 78% say they are AR hobbyists and 80% VR hobbyists. In comparison, 26% and 24% (in AR and VR respectively) claim to be professionals. There is a little overlap here as some professional developers work on projects in the same field as a hobby too. To look at this another way, 73% of all developers working both AR and VR identify themselves as hobbyists in both areas, while only 21% are professionals in both. Key challenges for AR or VR products to reach the mass market remain on the hardware side. For AR, it is creating hardware that is more socially acceptable than looking down at your mobile phone, or wearing the current range of headsets and glasses. For VR, it's affordable hardware that enables users to move around freely in their environment. Improved hardware could reduce the current usage friction, adoption barriers and create a broader addressable market for professionals.

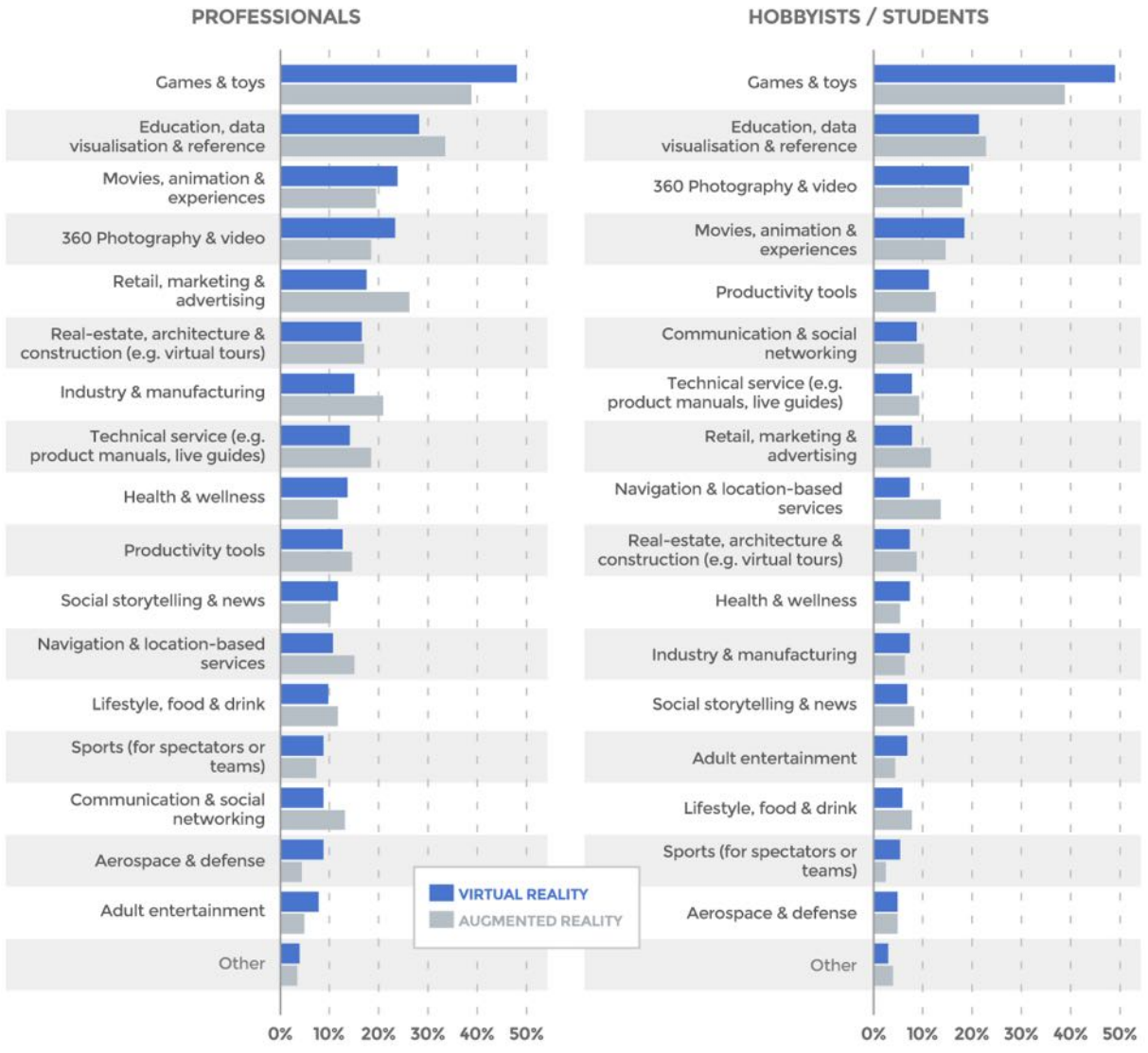
C# LEADS AS A PRIORITY LANGUAGE FOR AR/VR DEVELOPMENT

% of Augmented / Virtual Reality (n = 1,838)

Programming languages prioritised for AR/VR development



Augmented and Virtual Reality application areas targeted by professionals and hobbyists



vision mobile

Source: State of the Developer Nation Q1 2017 | http://vmob.me/DE1Q17 |

Identifying the right programming language for a given project is an excellent platform to spur a debate between team members. We looked closely into the differences between the AR and VR developers with respect to their preferred programming language. Firstly we found that of more than 1,800 developers who told us about their AR/VR development work, C# is the most popular primary programming language (30%) followed by C/C++ (16%) and Java (15%). Interestingly, professionals are more likely to use C# or C++ in comparison to hobbyists. The .NET framework has established itself as an excellent platform for the development of very secure and reliable web applications but its use in the Unity game engine has propelled it into a leading role in the AR/VR industry. C++ has a similar boost from the Unreal engine, with both of the engines supporting most major AR and VR hardware platforms.

> *Nine out of ten AR/VR developers would be considered juniors by other industries' standards (less than two years of experience).*

As mentioned earlier, the majority of AR/VR developers are hobbyists. Clearly there's a big difference between hobbyists who have been interested in the field for a long time and can't or don't want to make it their full time profession, and those who are newcomers working in other fields who are simply experimenting at this stage. Indeed, a large segment of our respondents fall into the latter category, having less than one year (49%) or between 1 and 2 years (41%) of experience in AR or VR. That is, nine out of ten AR/VR developers would be considered juniors by other industries' standards (less than two years of experience). The industry consists of many newcomers who are inexperienced in the field - they will not be deeply invested in any tools, technologies or platforms, so any vendor has the potential to establish market leadership with the right product.

If we look at the languages developers are using versus their experience, we see that JavaScript and Python users are the most inexperienced (51% and 50% respectively have less than one year of experience). It will be worth observing how the effervescent JavaScript community will find their niche in the AR/VR industry, now that several tools such as WebVR are starting to see the light of day. By contrast the currently dominant languages in AR and VR, C# and C/C++, are more likely to be used by experienced developers (68% each with at least one year of experience). We also see that most of the small but important segment of the AR/VR developer population who are professionals primarily use C# and C/C++ languages and have more than two year's experience.

Regardless of language or experience, developers don't seem to lack ideas for AR or VR applications. This is perhaps not surprising when anyone in need of inspiration can simply consult the many plots of the British TV series Black Mirror. Realistic and near-future AR or VR applications galore. From augmented reality horror games that learn from your reactions and maximise your senses' response, to a virtual reality realm for the terminally ill who are happily reliving their younger days once more. Although the possible ways to integrate AR and VR into our lives are numerous in theory, in practice developers are most often found creating games at this point. Games and Toys is by far the most common application category for VR projects (49% for both professionals and hobbyists). Education (23%), 360 Photography (21%), and Movies (21%) are the only other categories to surpass the 20% mark. Games and Toys was also the leading category for AR projects although it leads Education by a much smaller margin. However, a significant difference exists between hobbyists' and professionals' application categories. Both hobbyists and professionals embrace the

gaming industry almost equally, but hobbyists show less enthusiasm for categories other than Games and Toys and tackle fewer different categories on average.

Developing games has the advantage of having the clearest path to monetise AR and VR technologies at the moment, other than building apps on a contract basis. Moving forward, we see professionals exploring other categories to be the best route to AR and VR finding profitable markets. These developers are likely to create a lot of the value that future hardware platforms capture. Those hardware platforms are not here yet though, so in the near future it seems AR and VR will mainly impact the gaming industry. The types of applications that will affect our society in a more fundamental way - such as advanced productivity or communication tools - are further away than we hope.

3 ANGULAR VS REACT: BATTLE FOR THE FUTURE OF FRONT-END WEB DEVELOPMENT?

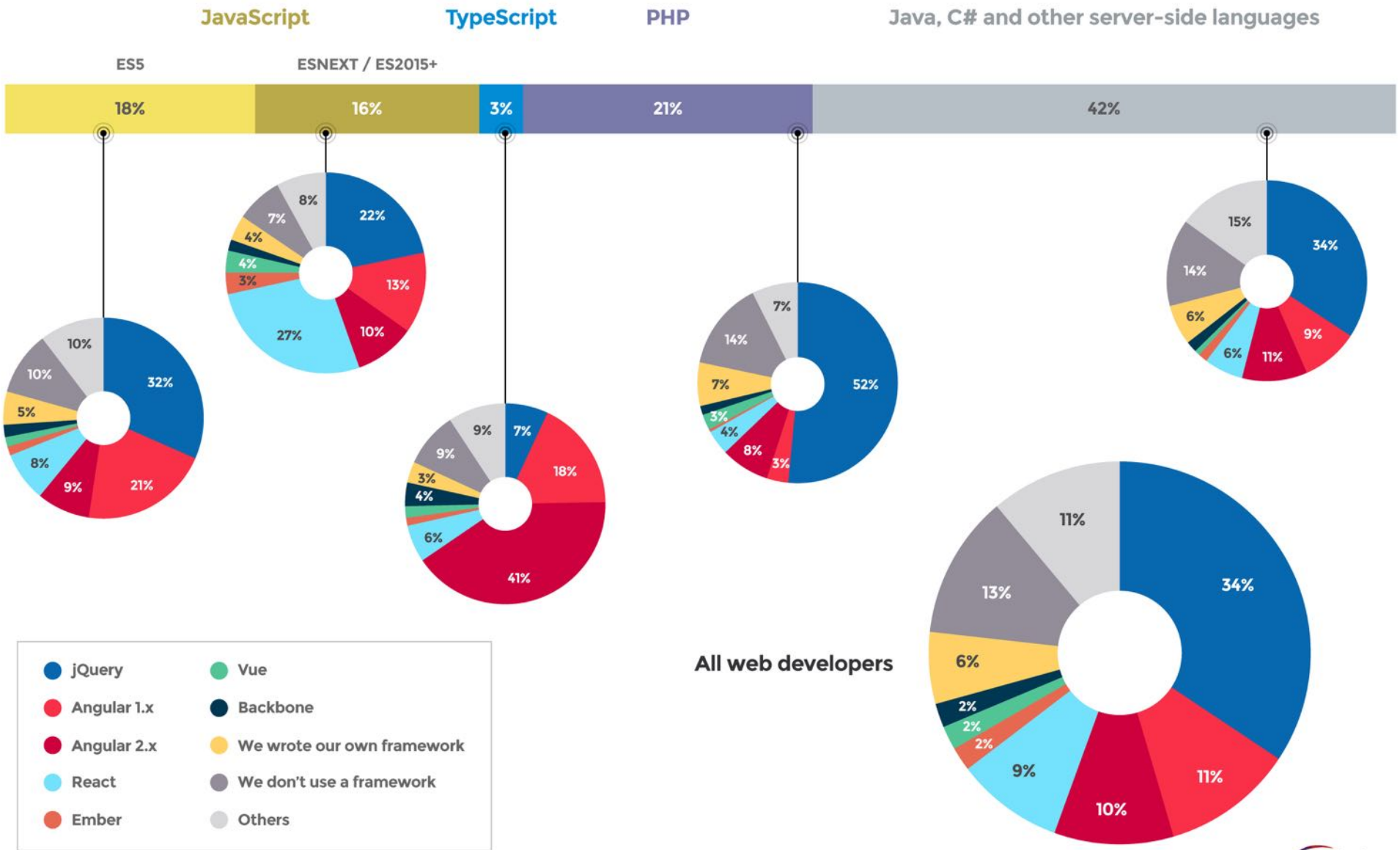
Google and Facebook are two of the world's most powerful companies and each has created a framework for building web apps. Angular and React respectively appear to be in a battle for the future of the web, with the active online debate and adoption for large consumer-facing apps seeming to lean quite strongly in React's favour at present. Are they collectively taking over the front-end? Is React really leading? Our data from a broad cross-section of nearly 6,000 web developers may surprise you.

Although traditional, largely static, web pages still have an important place, mobile is now the dominant computing paradigm and mobile users have come to expect the interactivity of native apps. To attempt to match a native app experience, a web app cannot be entirely rendered on the server side, the page has to be changed dynamically on the client. The more extensive the changes the greater the need for a better abstraction than the DOM (Document Object Model) to manage the complexity. This has driven ever growing usage of third-party JavaScript libraries and frameworks.

Historically jQuery was the first library to get really popular, enabling easier manipulation of the DOM on the client side. It's still the most popular today, as the primary front-end library for 34% of web developers. However, manually manipulating the DOM turns out to be extremely complex and error-prone when it's happening extensively, so frameworks that provide a better abstraction are increasingly important. Overall just 12% of web developers don't use any kind of framework and another 6% have written their own. That leaves 48% of web developers currently using a third-party framework other than jQuery as their primary way of doing front-end web development. Of those, Angular and React account for 30% of all usage, leaving all the others far behind. Indeed front-end web development is such a fragmented space that no other single library or framework accounts for more than 2% of primary usage. So React and Angular certainly lead other frameworks, although only around half of all web developers have fully embraced any single page application framework so far.

21% OF WEB DEVELOPERS PRIORITISE ANGULAR VERSUS JUST 9% FOR REACT

% of web developers using each JavaScript library or framework, overall and split by primary language (n=5,883)



AngularJS (Angular 1.x) was the first single page app framework to get the stamp of approval from an internet giant, when Google started to back the open-source side project of one of their employees publicly. Google's backing gave many large enterprises the confidence to adopt, and with broader adoption came a flourishing ecosystem of components and tools. As this was happening, React was built internally at Facebook and deployed on the Facebook newsfeed in 2011 and then Instagram's web app in 2012. Yet React wasn't released as open source until 2013, by which time Angular had an enormous lead in both adoption and ecosystem. Then in late 2014 Google appeared to stumble previewing Angular 2.0, which was going to be incompatible with Angular 1.x and use a new language. Reaction from the developer community was not good. By mid-2015 Google had agreed to work with Microsoft so that TypeScript became the official language for Angular 2.0, while the 1.x series had a promise of continued support, and a migration path between versions was created. This discontinuity for the Angular community seemed like a gift to the already rapidly growing React.

> *In total those using one or the other version of Angular number more than double those using React.*

Although Angular still had many vocal fans, anyone following the broader front-end web developer community online would have to assume that React was taking Angular's crown. At the time of writing React has passed Angular 1.x in terms of stars on their respective GitHub projects, with around 61,500 to 55,000. Angular 2.x trails both of these by far with 21,500. In the independent State of JavaScript survey run in late 2016, React came out way ahead of both versions of Angular in usage, interest, and retention. However, our own survey, which reaches out across many different developer communities does not reflect this result overall at all. Not only is Angular 2.x the primary framework for about as many developers as React (10% vs 9% globally), but Angular 1.x is still the most popular overall by a slim margin (11% use it as their primary framework). In total those using one or the other version of Angular number more than double those using React.

In order to see how reality in the market could be so different from the online buzz and even a large community survey, it's interesting to look at the breakdown of JavaScript library and framework usage by primary programming language. If we only look at the users of the latest versions of JavaScript - those who like to stay at the forefront and are more likely to be found debating framework choices on the internet - we see React is the primary framework for 27% of them. So amongst those who have made the switch to ESNext (i.e. the 2015 version of the JavaScript standard or later), who then use tools to convert their code to the JavaScript that's widely supported in browsers (known as ES5, introduced back in 2009), more are using React than both versions of Angular combined. However, this is the only group of developers for which React beats either version of Angular alone. These forward-looking JavaScript users are less than half of those primarily using JavaScript, and just 16% of all web developers (who almost all use some JavaScript). A further 18% of web developers are still primarily using ES5. More of these are currently still using Angular 1.x (21%) as their primary framework than Angular 2.x (9%) and React (8%) combined. These developers are getting on with what they know and are productive doing. They may be following the new standards and frameworks but most of them don't see enough benefit in switching yet. Another 3% of all web developers are primarily using TypeScript, which could be seen as the most advanced version of JavaScript currently available. However, some web developers understandably don't want to adopt anything not yet in the standards, others don't want to use the optional static types, and a significant minority still avoid anything from Microsoft. Given that Angular 2.x has adopted TypeScript it's not surprising to find 41% of those

primarily using the language have adopted the framework. There are another 18% currently still using Angular 1.x that will most likely migrate to Angular 2.x.

After some flavour of JavaScript, the most popular language for web developers is PHP, with 21% still considering it their primary language. Given the focus on rendering pages server-side in most of the popular PHP content management systems, it's not too surprising to find less interest in single page app frameworks in general amongst these developers, with 52% still using jQuery as their primary library. Interestingly only 3% of PHP developers are primarily using Angular 1.x, with 8% on Angular 2.x, and just 4% for React. In fact almost as many PHP developers don't use any library or framework for the front-end (14%) as use React plus either Angular version. Developers primarily using server-side languages other than JavaScript/Node.js or PHP (totalling 42% of all web developers) are significantly less likely to be using jQuery than PHP developers but they are also significantly less interested in Angular and React than the JavaScript developers (26% vs 38%). When they do primarily use one of these front-end frameworks, far more choose Angular (20%) than React (6%), and more of the Angular users are on version 2.x (11%) than version 1.x (9%). Considering all of those who are server-side developers not using Node.js, which is 63% of the web developer population, Angular is significantly preferred to React at this point, probably because it is complete framework, rather than forcing the developer to make lots of other library and tooling choices as they currently have to with React.

There are many alternative futures that could be inferred from this data. The simplest story would be that framework preferences won't move much for the different groups. Server-side developers will continue to have relatively little interest in the front-end frameworks and ES5 developers will stick to Angular 1.x when they eventually transition to ESNext or TypeScript. This doesn't fit the current trend of increased JavaScript usage across the web, front-end and server. It also ignores the fact that Google will be migrating to Angular 2.x internally and developers will not want to be left without support one day. We could also imagine that as developers start using ESNext or TypeScript their framework preferences shift accordingly. Both React and Angular gain greater share, with React growing faster than Angular. There's probably some truth in this, but it's too focused on the front-end developers. Server-side developers who aren't using Node.js are less likely to find React attractive without a much simpler learning curve for the ecosystem. Then again, the most popular PHP framework is still WordPress, and the company behind WordPress has chosen React as the new front-end framework for WordPress.com - many PHP developers may follow them. Facebook has significant momentum with React, but Angular is likely to remain the most popular for smaller projects and internal apps. What we can predict is that despite the inevitable churn on the front-end, both frameworks have successfully built a critical mass of developers creating valuable ecosystems, and both are set for significant growth in the years ahead. We'd be surprised if the 30% of web developers using either Angular or React didn't become 40% in the next 2 years.

4 CHALLENGERS CHASING THE CLOUD PRIZE

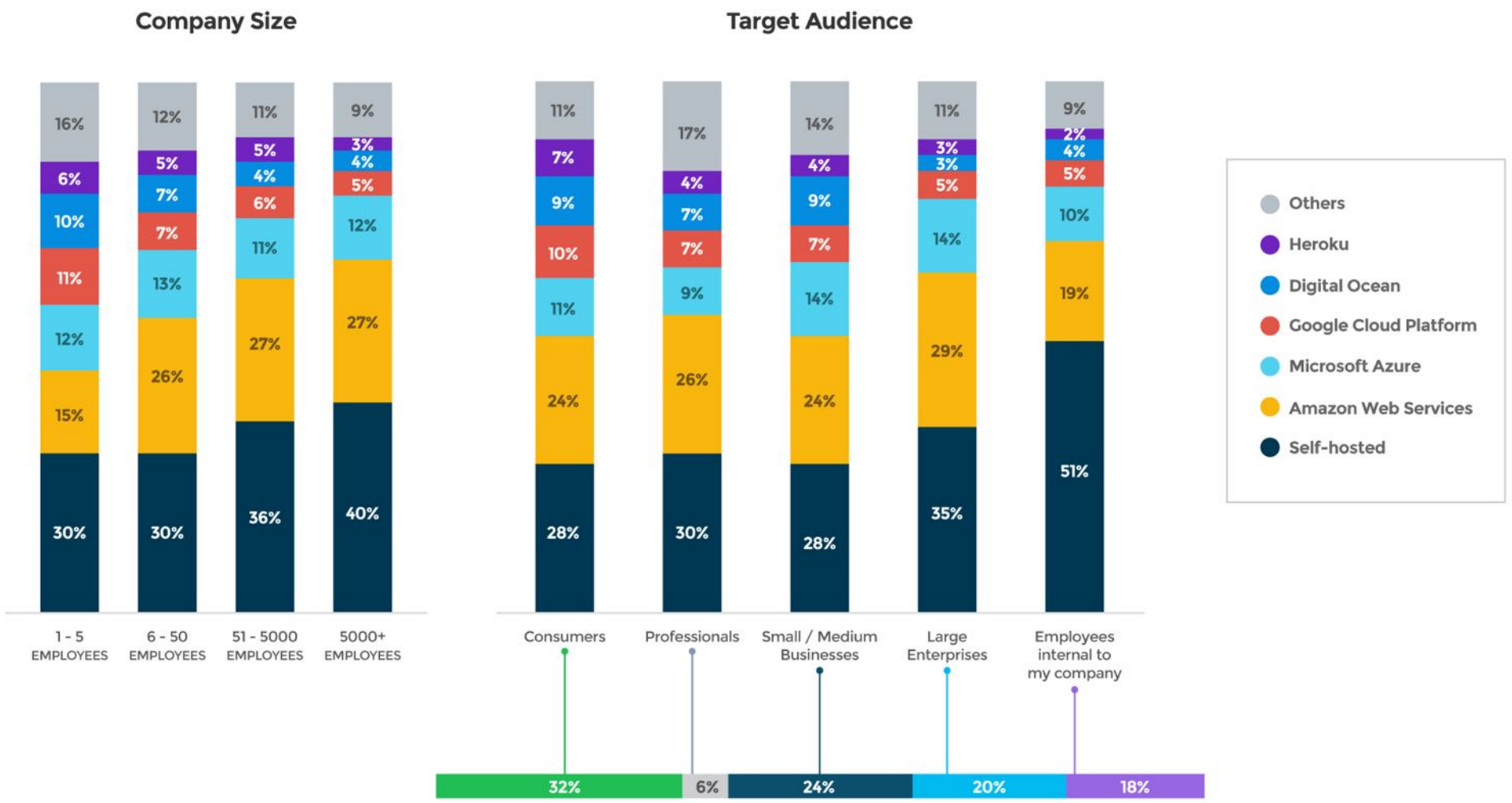
Amazon has a long history of re-investing profits from their existing businesses into expanding operations and creating new businesses. The result was that Amazon expanded to become one of the world's largest companies, while not reporting any significant profit in over two decades. Jeff Bezos has been very clear all along that he intends to continue reinvesting profits, and yet last year something changed. Amazon reported profits way outside its usual range. The business that is clearly driving this is Amazon Web Services (AWS). Their creation of the public cloud market is now paying off to such a degree they can't spend the profits fast enough. For several years now others have recognised the size of the prize that Amazon is chasing and raced to catch up so they could compete for a share of it.

Perhaps the most impressive thing about the level of profit that AWS is generating is that it's in the midst of a price war with these challengers. Using the data from more than 5,000 backend developers who took our global survey, we can see how successful the challengers have been so far. More importantly, we can take a look at how much further this market has to go, and how the various competitors are likely to do in the future.

An important thing to understand about the public cloud market is that vendors don't care too much about how many individual developers they attract. A solo developer with a niche Software-as-a-Service application running on a single server is never going to be the most profitable customer. A wide range of niche needs satisfied doesn't help a cloud platform in the way it helps a mobile or desktop platform. End users don't buy anything from the cloud hosting provider. Some cloud hosting businesses take this to the extreme of only really targeting large enterprise users. These can be profitable businesses but they miss out on the software startups that grow large serving those enterprises, or going directly to consumers. You can imagine that Netflix has an extremely large bill from Amazon every month and, for example, Snap revealed at its recent IPO that it spends $400 million a year on Google's cloud services and has signed another deal to spend $1 billion over the next five years on AWS as well.

60% OF LARGE COMPANY DEVELOPERS USE PUBLIC CLOUD, 51% OF INTERNAL APPS STILL SELF-HOSTED

% of cloud developers primarily using each cloud host, split by company size (n=5,383) and target audience (n=4,894)



A global network of data centres and fibre costs an incredible amount to build and run. The software required to utilise the hardware as efficiently as possible while ensuring great availability, performance, and scalability for hosted applications is also very expensive to build and maintain. The greater the scale a cloud hosting provider operates at, the better they can make the services for everyone, and the wider the range of services they can offer. A broad market appeal is almost certainly required to win big.

Amazon leads the market by far in terms of total developer numbers, number of services offered, and of course revenues. There are really only two other companies that can mount a serious challenge to them, both already giants in technology that can afford to fund the eye-watering capital expenditure required whilst participating in a price war. Microsoft Azure is the widely recognised second place cloud platform, with excellent growth but also high expectations as a major part of Microsoft's future business. Microsoft's advantage is that they already have relationships with, and software in use at, almost all of those large enterprise customers that everyone wants to win cloud business from. Google Cloud Platform is a slightly distant third place, very strong in both hardware and software infrastructure thanks to the scale of their own consumer internet services, but lacking experience working with the large customers that almost everyone wants to win.

If we look at the primary cloud hosting providers for developers who took our global survey, split by company size, we can see this reflected. Amazon leads at every company size although, for the smallest companies (1-5 employees) where Amazon has just a 15% share, they face very credible competition from Microsoft (12%), Google (11%), and Digital Ocean (10%). However, when we look at larger companies, Amazon's share grows to 26-27% at every size, Microsoft stays in the 11-13% range, while Google fades along with Digital Ocean. Google has just a 5% share of the companies with more than 5,000 employees, and Digital Ocean just 4% at the same size. We expect that Google will be able to grow their share at the larger company end of the market as they are making significant investments in the sales and support teams required, as well as adding some features suited to customers with legacy systems to migrate. Digital Ocean on the other hand is primarily a low-cost competitor with a simplified offering, which is unlikely to satisfy the needs of many large enterprises.

> *Developers building apps for their internal employees may only make up 18% of the market but 51% of them are still self-hosting and they skew towards larger companies. The revenue opportunity is almost certainly larger with this latter group.*

When developers assess the credibility of a cloud host, their decision often depends on their target audience. Amazon is most popular with developers regardless of targeted audience, although strongest with backend developers who target large enterprises, where 29% of backend developers are primarily using AWS. Microsoft shows greater strength equally with developers who target large enterprises, and those who target small to medium businesses (14% each). They are weaker with those targeting consumers (11%) or professionals (9%). Google shows the opposite pattern being strongest with developers who target consumers (10%) but only half as popular with those that target large enterprises or internal employees. This fits with the announcements from their respective conferences, where Amazon and Microsoft can show off a long list of global giants, whilst Google points to the creators of Pokémon Go (Niantic, who were spun out from Google in the first place), with real giants like HSBC only doing "critical proof of concept projects" for their internal apps.

Of course there are still plenty of opportunities to win business from developers targeting consumers, 32% of backend developers have this target audience and 28% of those are still self-hosting their backend. However, consumer-focused developers are heavily skewed to the smaller and less profitable end of the spectrum. Only a relatively small number of consumer internet companies make it big, and the ones that make it really big are quite likely to migrate to their own data centre eventually. Developers building apps for their internal employees may only make up 18% of the market but 51% of them are still self-hosting and they skew towards larger companies. The revenue opportunity is almost certainly larger with this latter group.

Taking the share of primary cloud host across both company sizes and target audiences into account, we predict that AWS will continue to outpace both Microsoft Azure and Google Cloud Platform, growing faster in absolute terms than both of them put together. We also expect Microsoft's cloud business to grow faster (again in absolute terms) than Google's in the short term, as building trust with large enterprises and a reputation for serving them well is going to take several years, however good their technology. That said, Google is increasingly well positioned to win business from the startups that target enterprises. As companies rush to enable mobile workflows some of these startups could grow spectacularly fast, giving Google a decent shot at second place in the cloud hosting market in the longer term.

5 THE EMERGING IOT TOOL SECTOR

IoT platforms were on the cusp of reaching the peak of inflated expectations in Gartner's Hype Cycle from August 2016. Not surprisingly - there are literally hundreds of them, and counting. Also, the word 'platform' is used for anything, from network infrastructure to hardware components to cloud services. In the end, IoT owes its boom in popularity to more and better tools becoming available for developers. In this chapter, we shed some light on the types of tools that IoT developers are actually using.

In our research we've long argued that the Internet of Things is lacking the kind of platforms that connect developers to an addressable market, rather than just implementing technology. Arguably, SmartThings, Android Wear, Amazon Alexa, and their peers already do that, be it with an as yet limited installed base of users. Our data indicates that IoT cloud platforms are still squarely in the 'just implementing technology' realm. A missed opportunity for those vendors, or is this just what the market wants?

Despite the proliferation of IoT platforms and other tools, the IoT tool market is still underdeveloped and heavily fragmented. We asked IoT developers to select technologies they use out of a list of 15 categories. On average, IoT developers use 2.9 types of tools on that list, or one in five out of the list; professionals slightly more at 3.5 tool types. That's comparatively fewer than developers in other sectors like cloud, mobile, or web, where developers use a quarter to a third of the tools listed. Part of the reason is fragmentation: not every tool is comprehensive enough to be relevant to a large number of developers. In part, the low tool usage is due to underdevelopment of the tool market. 11% of IoT developers don't use any of the tools in our list, compared to 6% of web developers and 3% of mobile developers, who we presented with similar sized lists. Either way, we expect to see a good bit of consolidation and development before we can call this a mature tooling market.

Professional IoT developers use more tools than amateurs, as we said, but they tend to use specific types of tools more often.

> *Despite the proliferation of IoT platforms and other tools, the IoT tool market is still underdeveloped and heavily fragmented.*

The biggest differences are seen in categories like software deployment tools, IoT cloud platforms, embedded operating systems, machine learning platforms, gateway middleware, beacons, message brokers, or fog computing. What all these technologies have in common is that they are components of a complete IoT solution, i.e. technologies that an engineer would integrate under the hood to implement a valuable product or project. Fog or edge computing - championed by Cisco - is notable by its absence: a mere 4% of IoT developers are working with this technology. It may be too early for this technology, or the need for it might not be as big as pundits proclaim. Time will tell.

> *Professional IoT developers use more tools than amateurs, but they are IoT solution components, not platforms that provide developers with addressable markets.*

The gap between professional and amateur use is virtually non-existent in hardware platforms such as single-board computers like the Raspberry Pi or prototyping boards like the Arduino or Intel Edison. These microprocessors and computers have become so cheap and accessible (i.e. easy to use) that anyone with a minimal technical background can play around with them and put them to productive use. Even wearables toolkits and middleware show signs of this level of accessibility.

We also don't see the amateur-pro gap in high-level, integrating platforms: Smart Home platforms like HomeKit or SmartThings, smartwatch platforms like WatchOS or Android Wear, or voice platforms like Amazon Alexa. These are all areas (IoT verticals) that are easy to get into, easy to imagine (and design) a solution that scratches your own itch, and therefore highly popular among hobbyists, as we've highlighted in other reports. Attractiveness to hobbyists aside, these comprehensive types platforms lower the barrier for people to start building meaningful solutions quickly, whereas the component technologies from above are still more the domain of specialists. Even health & wellness data platforms like Google Fit or HealthKit - arguably a more specific, advanced domain - have only a small difference in usage between professionals and amateurs.

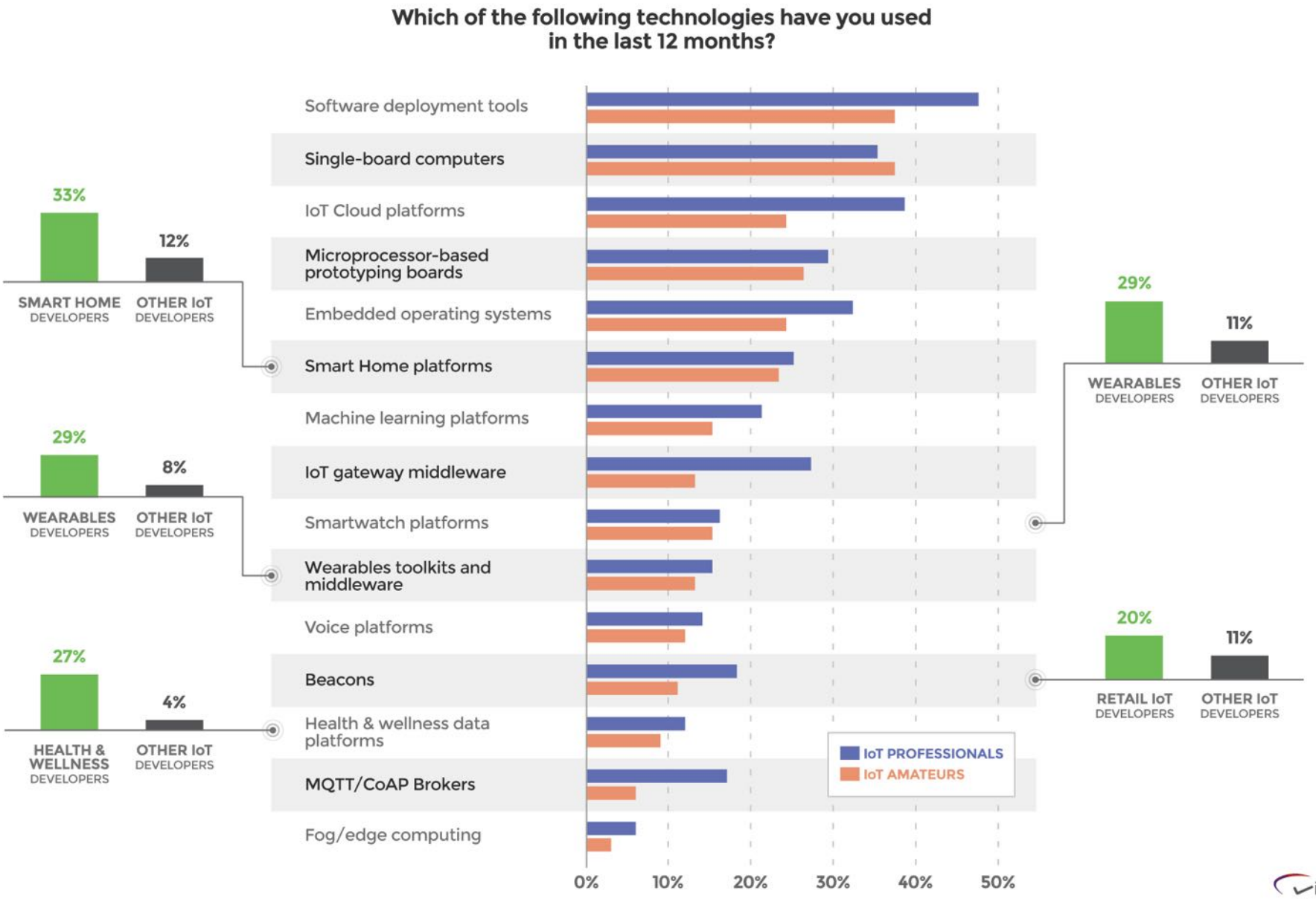
Some of the technologies in the list are specific to certain verticals: wearables toolkits are for wearables developers, Smart Home platforms for Smart Home developers, and so on. Or are they? 12% of developers who use Smart Home platforms are not currently targeting or planning to target that vertical, for example. That is a reasonably big number, even though the usage gap with Smart Home developers is indeed clear. Some of these technologies might be fairly generic, and might even be 'misused' for unrelated projects. In some cases like smartwatch platforms, developers might work on a smartwatch app as part of a broader IoT solution, without self-identifying necessarily as 'wearable developers'.

Location beacons are an interesting case. Their most marketed use cases were in retail and hospitality applications. However, only 20% of retail IoT developers use beacons; a good bit less than the 27% to 33% in-vertical usage we see for other vertical-specific technologies. Furthermore, the gap between in-vertical and out-of-vertical usage is only 9 percentage points, i.e. half that of the other technologies discussed here. We take this as a sign that beacons may be overhyped, perhaps technologically, but more likely in terms of how valuable the use cases are to customers. In our previous State of the Nation report (Q3 2016), we noted that retail was the sector within IoT with the fastest attrition of developers, possibly due to a sense of disillusionment and kickback from the hype. The data on technology use in the retail vertical seems to support that hypothesis.

We opened this chapter with Gartner's claim that we're at the peak of inflated expectations when it comes to IoT platforms. Our IoT research over the past few years says that we've already passed it, with stalled population growth and high churn among developers, heading full-speed towards the trough of disillusionment. The key reason is that the technology is still too immature, very few platforms are finding product-market fit, and thus the majority of consumer-focused developers lack a platform that gives them a viable market. Of course the core technology marches on, with some mostly consumer-focused tools finding uses outside their original intended market. The potential remains enormous. However, it's going to get worse before it gets better, with a lot of consolidation among the many existing technology platforms.

COMPONENTS ARE FOR IOT PROS; HARDWARE AND APP PLATFORMS FOR EVERYONE

% of IoT developers by involvement and vertical (n = 3,845)



Source: State of the Developer Nation Q1 2017 | http://vmob.me/DE1Q17 |

6 THE MACHINE LEARNING LANGUAGES WAR

Q&A sites and data science forums are buzzing with the same questions over and over again: I'm new in data science, what language should I learn? What's the best language for machine learning? There's an abundance of articles attempting to answer these questions, either based on personal experience or on job offer data. There's so much more activity in machine learning than job offers in the West can describe, however, and peer opinions are of course very valuable but often conflicting and as such may confuse the novices.

We turned instead to our hard data from 2,000+ data scientists and machine learning developers who responded to our latest survey about which languages they use and what projects they're working on - along with many other interesting things about their machine learning activities and training. Then, being data scientists ourselves, we couldn't help but run a few models to see which are the most important factors that are correlated to language selection. We compared the top-5 languages and the results prove that there is no simple answer to the "which language?" question. It depends on what you're trying to build, what your background is and why you got involved in machine learning in the first place.

First, let's look at the overall popularity of machine learning languages. Python leads the pack, with 57% of data scientists and machine learning developers using it and 33% prioritising it for development. Little wonder, given all the evolution in the deep learning Python frameworks over the past 2 years, including the release of TensorFlow and a wide selection of other libraries. Python is often compared to R, but they are nowhere near comparable in terms of popularity: R comes fourth in overall usage (31%) and fifth in prioritisation (5%). R is in fact the language with the lowest prioritisation-to-usage ratio among the five, with only 17% of developers who use it prioritising it. This means that in most cases R is a complementary language, not a first choice. The same ratio for Python is at 58%, the highest by far among the five languages, a clear indication that the usage trends of Python are the exact opposite to those of R. Not only is Python the most widely used language, it is also the primary choice for the majority of its users. C/C++ is a distant second to Python, both in usage

(44%) and prioritisation (19%). Java follows C/C++ very closely, while JavaScript comes fifth in usage, although with a slightly better prioritisation performance than R (7%). We asked our respondents about other languages used in machine learning, including the usual suspects of Julia, Scala, Ruby, Octave, MATLAB and SAS, but they all fall below the 5% mark of prioritisation and below 26% of usage. We therefore focused our attention on the top-5 languages.

> *R is in fact the language with the lowest prioritisation-to-usage ratio among the five, with only 17% of developers who use it prioritising it. This means that in most cases R is a complementary language, not a first choice.*

Our data reveals that the most decisive factor when selecting a language for machine learning is the type of project you'll be working on - your application area. In our survey we asked developers about 17 different application areas while also providing our respondents with the opportunity to tell us that they're still exploring options, not actively working on any area. Here we present the top and bottom three areas per language: the ones where developers prioritise each language the most and the least.

Machine learning scientists working on sentiment analysis prioritise Python (44%) and R (11%) more and JavaScript (2%) and Java (15%) less than developers working on other areas. In contrast, Java is prioritised more by those working on network security / cyber attacks and fraud detection, the two areas where Python is the least prioritised. Network security and fraud detection algorithms are built or consumed mostly in large organisations - and especially in financial institutions - where Java is a favourite of most internal development teams. In areas that are less enterprise-focused, such as natural language processing (NLP) and sentiment analysis, developers opt for Python which offers an easier and faster way to build highly performing algorithms, due to the extensive collection of specialised libraries that come with it.

Artificial Intelligence (AI) in games (29%) and robot locomotion (27%) are the two areas where C/C++ is favoured the most, given the level of control, high performance and efficiency required. Here a lower level programming language such as C/C++ that comes with highly sophisticated AI libraries is a natural choice, while R, designed for statistical analysis and visualisations, is deemed mostly irrelevant. AI in games (3%) and robot locomotion(1%) are the two areas where R is prioritised the least, followed by speech recognition where the case is similar.

Other than in sentiment analysis, R is also relatively highly prioritised - as compared to other application areas - in bioengineering and bioinformatics (11%), an area where both Java and JavaScript are not favoured. Given the long-standing use of R in biomedical statistics, both inside and outside academia, it's no surprise that it's one of the areas where it's used the most. Finally, our data shows that developers new to data science and machine learning who are still exploring options prioritise JavaScript more than others (11%) and Java less than others (13%). These are in many cases developers who are experimenting with machine learning through the use of a 3rd-party machine learning API in a web application.

> *Python is prioritised the most by those for whom data science is the first profession or field of study (38%). This indicates that Python has by now become an integral part of data science - it has evolved into the native language of data scientists.*

Second to the application area, the professional background is also pivotal in selecting a machine learning language: the developers prioritising the top-five languages more than others

come from five different backgrounds. Python is prioritised the most by those for whom data science is the first profession or field of study (38%). This indicates that Python has now become an integral part of data science - it has evolved into the native language of data scientists. The same can not be said for R, which is mostly prioritised by data analysts and statisticians (14%), as the language was initially created for them, replacing S.

Front-end web developers extend their use of JavaScript to machine learning, 16% prioritising it for that purpose, while staying clear of the cumbersome C/C++ (8%). At the exact opposite stand embedded computing hardware / electronics engineers who go for C/C++ more than others, while avoiding JavaScript, Java and R more than others. Given their investment in mastering C/C++ in their engineering life, it would make no sense to settle for a language that would compromise their level of control over their application. Embedded computing hardware engineers are also the most likely to be working on near-the-hardware machine learning projects, such as IoT edge analytics projects, where hardware may force their language selection. Our data confirms that their involvement is significantly above average in industrial maintenance, image classification and robot locomotion projects among others.

For Java, it's the front-end desktop application developers who prioritise it more than others (21%), which is also inline with its use mostly in enterprise-focused applications as noted earlier. Enterprise developers tend to use Java in all projects, including machine learning. The company directive in this case is also evident from the third factor that is strongly correlated to language prioritisation - the reason to get into machine learning. Java is prioritised the most (27%) by developers who got into machine learning because their boss or company asked them to. It is the least preferred (14%) by those who got into the field just because they were curious to see what all the fuss was about - Java is not a language that you normally learn just for fun! It is Python that the curious prioritise more than others (38%), another indication that Python is recognised as the main language that one needs to experiment with to find out what machine learning is all about.

It seems that some universities teaching data science courses still need to catch up with this notion though. Developers who say that they got into machine learning because data science is/was part of their university degree are the least likely to prioritise Python (26%) and the most likely to prioritise R (7%) as compared to others. There is evidently still a favourable bias towards R within statistics circles in academia - where it was born - but as data science and machine learning gravitate more towards computing, the trend is fading away. Those with university training in data science may favour it more than others, but in absolute terms it's still only a small fraction of that group too that will go for R first.

C/C++ is prioritised more by those who want to enhance their existing apps/projects with machine learning (20%) and less by those who hope to build new highly competitive apps based on machine learning (14%). This pattern points again to C/C++ being mostly used in engineering projects and IoT or AR/VR apps, most likely already written in C/C++, to which ML-supported functionality is being added. When building a new app from scratch - especially one using NLP for chatbots - there's no particular reason to use C/C++, while there are plenty of reasons to opt for languages that offer highly-specialised libraries, such as Python. These languages can more quickly and easily yield highly-performing algorithms that may offer a competitive advantage in new ML-centric apps.

Finally, contractors who got into machine learning to increase their chances of securing highly-profitable projects prioritise JavaScript more than others (8%). These are probably JavaScript developers building web applications to which they are adding a machine learning API. An example would be

visualising the results of a machine learning algorithm on a web-based dashboard.

> *C/C++ is prioritised more by those who want to enhance their existing apps/projects with machine learning (20%) and less by those who hope to build new highly competitive apps based on machine learning (14%).*

Our data shows that popularity is not a good yardstick to use when selecting a programming language for machine learning and data science. There is no such thing as a 'best language for machine learning' and it all depends on what you want to build, where you're coming from and why you got involved in machine learning. In most cases developers port the language they were already using into machine learning, especially if they are to use it in projects adjacent to their previous work - such as engineering projects for C/C++ developers or web visualisations for JavaScript developers. If your first ever contact with programming is through machine learning, then your peers in our survey point to Python as the best option, given its wealth of libraries and ease of use. If, on the other hand, you're dreaming of a job in an enterprise environment, be prepared to use Java. Whatever the case, these are exciting times for machine learning and the journey is guaranteed to be a mind-blowing one, irrespective of the language you opt for. Enjoy the ride!

MACHINE LEARNING LANGUAGES SHOOT OUT — HOW TO PICK THE RIGHT LANGUAGE

% of machine learning developers / data scientists who use or prioritise each language (n = 2,022)

Source: State of the Developer Nation Q1 2017 | http://vmob.me/DE1Q17 |

Are developers satisfied with your developer program investments?

The Developer Program Benchmarking is a research service that tracks how developers perceive developer program investments – from sample code to hackathons - and helps you stay ahead of the competition. This is not just an analyst assessment of developer programs, it's the real developer experience.

BOOK A DEMO TODAY

http://vmob.me/DPBdemo

Book a demo now and find out how you can answer important questions like the following:

- ✓ What are the strengths and weaknesses of your competitors' developer programs?
- ✓ What support are developers looking for from tech companies?
- ✓ Does your developer program have the right priorities and is it meeting developer needs?
- ✓ What are the gaps that you need to address in order to improve your developer program?

METHODOLOGY

Developer Economics 12th edition reached an impressive 21,200+ respondents from 162 countries around the world. As such, the Developer Economics series continues to be the most global independent research on mobile, desktop, IoT, cloud, AR/VR and machine learning developers and data scientists combined ever conducted. The report is based on a large-scale online developer survey designed, produced and carried out by VisionMobile over a period of six weeks between November and December 2016.

Respondents to the online survey came from over 160 countries, including major app, machine learning and IoT development hotspots such as the US, China, India, Israel, UK and Russia and stretching all the way to Kenya, Brazil and Jordan. The geographic reach of this survey is truly reflective of the global scale of the developer economy. The online survey was translated into eight languages (Portuguese, Spanish, French, Russian, Korean, Japanese, Chinese (Simplified), Vietnamese) and promoted by 69 leading community and media partners within the software development industry.

To eliminate the effect of regional sampling biases, we weighted the regional distribution across eight regions by a factor that was determined by the regional distribution and growth trends identified in our App Economy research. Each of the separate branches: mobile, desktop, IoT, cloud, augmented and virtual reality, and data science and machine learning were weighted independently and then combined.

To minimise other important sampling biases across our outreach channels, we weighted the responses to derive a representative distribution for platforms, segments and types of IoT project. Where we had sufficient historical data we used a Random Forest to calculate appropriate weights, excluding from our models the channels of our research partners to eliminate sampling bias due to respondents recruited via these channels. We derived a weighted platform distribution based on platform data from independent, representative channels. Again, this was performed separately for each of mobile, IoT, desktop and cloud, augmented and virtual reality, and data science and machine learning.

For more information on our methodology please visit https://www.visionmobile.com/methodology.

distilling market noise into market sense