

عشر قواعد بسيطة لكتابة ورقة علمية عن برمجية بحثية

جوزيف د. رومانو، جيسون هـ. مور



الملخص:

يتناول هذا المقال «أوراق البرمجيات»، أي الأوراق العلمية المحكّمة التي تصف برمجيات علمية جديدة، ويبيّن أنها تختلف جوهرياً عن المقالات البحثية التقليدية وتتطلب عناية خاصة لزيادة أثرها وفاعليتها في نقل المعلومات، ويميّز المؤلفان بين نوعين منها: الأوراق المستقلة التي تصف البرمجية وحدها، والمقالات البحثية التي يكون تطوير البرمجية أحد مكوناتها الحاسمة. ثم يعرضان عشر قواعد، منها: الاطلاع على أدلة كتابة الشيفرة الجيدة، واختيار أوعية النشر وأنواع المقالات الملائمة، والنشر للمستخدمين لا للمطورين، ووضع خطة طويلة الأمد لإدارة البرمجية، والوقاية من تعطل الروابط بالمعرفات الدائمة، والفصل بين التوثيق ونتائج البحث، ومواكبة أدوات التطوير الحديثة، والحفاظ على الاتساق بين الموارد، والتخطيط لأوراق المتابعة، وإعطاء الأولوية للظهور والإتاحة. ويخلصان إلى أن اتباع هذه القواعد يجعل أوراق البرمجيات سهلة الاستخدام ورسنية علمياً ومقاومة لتغيرات التقنية.

الكلمات المفتاحية: البرمجيات العلمية، أوراق البرمجيات، خطة إدارة البرمجيات، تآكل الروابط، المعرفات الدائمة

مقدمة:

أصبحت البرمجيات العلمية في العقود الأخيرة سمة حاسمة في جميع مسارات العمل البحثي تقريباً [1]. ولذلك يتحمل الباحثون الحاسوبيون والمعلوماتيون مسؤولية إصدار البرمجيات ونشرها بالقدر نفسه من الصرامة العلمية الذي يتّبع مع البروتوكولات البحثية ومجموعات البيانات والدراسات التجريبية الأخرى التي تُطرح في المجتمع العلمي. ولعل كتابة ورقة محكّمة (ونشرها) عن برمجية علمية مطوّرة حديثاً من أفضل السبل لتحقيق ذلك؛ إذ يمكن لـ«أوراق البرمجيات» أن تصل إلى عدد هائل من المستخدمين المحتملين (بل تؤدي دور الإعلان عن البرمجية)، وهي طريقة رائعة لإثبات أن البرمجية تصمد أمام التمهيص العلمي، وتتيح للمستخدمين إعادة استخدام البرمجية والاستشهاد بها بسهولة في بحوثهم.

غير أن أوراق البرمجيات تختلف اختلافاً جوهرياً عن المقالات البحثية «التقليدية» الأخرى. فعملية تصميم البرمجيات وتنفيذها تختلف عن تصميم التجارب المخبرية أو الدراسات السريرية أو تحليلات البيانات الخام وإجرائها [2]. وثمة أيضاً اختلافات في «المنتج النهائي» للبحث: فدراسات البرمجيات، بطبيعة الحال، تُثمر برمجية يُعاد استخدامها مباشرة، في حين تقدّم أنماط الدراسة الأخرى بروتوكولات جديدة أو نتائج محددة أو أسئلة أو فرضيات للمتابعة.

وثمة أساساً نوعان من أوراق البرمجيات: (1) أوراق مستقلة تصف البرمجية وحدها، وتكون عادةً بصيغة أقصر من المقال المكتوب عن دراسة بحثية تقليدية؛ و(2) مقال (أكثر تقليدية) يصف سؤالاً بحثياً أصيلاً يتضمن تطوير برمجية جديدة بوصفه أحد مكوناته الحاسمة. ومن أمثلة النوع الأول الأوراق الأصلية التي وصفت (Biopython) [3] و(scikit-learn) [4] و(SAMtools) [5]. ومن أمثلة النوع الثاني الأوراق التي قدّمت تحليل إثراء مجموعات الجينات و(Gene Set Enrichment Analysis) [6]، وأدوات خريطة الاتصال (Connectivity Map) [7]، و(VIPER) [8]. ومع أن هذين الخيارين يُنتجان أسلوبين مختلفين جداً من الأوراق، فإن القواعد البسيطة العشر المعروضة أدناه تنطبق إلى حد كبير على كليهما.

القاعدة 1: اقرأ أوراق «القواعد البسيطة العشر» الأخرى عن كتابة الشيفرة:

لكي تكون لديك ورقة برمجيات جيدة، تحتاج أولاً إلى برمجية جيدة. وتنطبق هنا جميع القواعد الأخرى لكتابة برمجيات علمية رائعة، ولا سيما تلك التي تناولتها مقالات أخرى من سلسلة «القواعد البسيطة العشر». فكل برمجية علمية مؤثرة ينبغي أن تسعى إلى أن تكون متينة [9]، وموثقة جيداً [10]، وسهلة الاستخدام [11]، ومُدارة في نظام للتحكم في الإصدارات (version control) [12]. ومزايا جعل برمجيتك مفتوحة المصدر (بشروط ترخيص شفافة) واستضافتها في مستودعات عامة معترف بها على نطاق واسع معروفة جيداً [13]، وينبغي اتباع هذه الممارسة بانتظام ما لم يكن هناك سبب قاهر يحول دون ذلك. وينبغي أن تستخدم التقييمات وحالات الاستخدام والأمثلة التوضيحية بيانات عالية الجودة، ويُفضّل أن تكون موصوفة جيداً من قبل [14,15].

القاعدة 2: اعرف أنسب أوعية النشر وأنواع المقالات المقدّمة:

كثيراً ما تخصص الدوريات وأعمال المؤتمرات التي تركز على المجالات الحاسوبية من البحث أنواعاً من المقالات مكرسة تحديداً لوصف البرمجيات أو قواعد البيانات الجديدة، ويمكن أن تكون هذه أوعية رائعة لنشر المعلومات بسرعة وإيجاز، ولجمهور يُفترض فيه مستوى معين من الكفاءة التقنية. ومن الجيد أن تفكر مبكراً وأن تعيد النظر كثيراً عند اختيار دورية أو مؤتمر بعينه. واحرص على إيلاء عناية خاصة لأي متطلبات غير معتادة تفرضها الدوريات؛ فبعضها يشترط أن تستخدم التقييمات بيانات حقيقية (أي غير مُصطنعة) [16]، وبعضها لديه متطلبات خاصة بالإبلاغ أو بإيداع البيانات أو البرمجيات [17]، وعليك دائماً أن تنظر فيما إذا كانت برمجيتك (وأسلوب الورقة الذي تريده) تتوافق مع بيان رسالة الدورية أو الاجتماع أو أهدافه. ويمكن أن تكون النقاشات مع المرشدين والمتعاونين والزملاء الآخرين مفيدة جداً في هذا السياق؛ فنجاحاتهم وإخفاقاتهم السابقة قد تُجَبِّك في النهاية تقديم ورقتك إلى دورية غير ملائمة (وكل ما يصاحب ذلك من متاعب). ومن أمثلة أنواع المقالات والدوريات الشائعة لأوراق البرمجيات: دورية (Bioinformatics) (نوع المقال: «(original papers)» أو «(application notes)»)، ودورية «(Database issue)» و«(Web Server issue)» السنويان في دورية (Nucleic Acids Research)، و«(software articles)» في (PLOS Computational Biology).

القاعدة 3: انشر للمستخدمين، لا للمطورين:

رغم القاعدة 2، ينبغي لك دائمًا أن تنظر في التقديم إلى أوعية غير حاسوبية. فنحن الباحثين الحاسوبيين كثيرًا ما نعمل في مجالات شديدة التداخل بين التخصصات، ونكتب برمجيات تجعل البحث في مجالات أخرى أسهل وأكفأ وأمتن علميًا. وكثيرًا ما يكون العلماء العاملون في هذه المجالات مهتمين جدًا بمعرفة الأدوات البرمجية الجديدة التي ستساعدهم يوميًا، لكنهم قد لا يتصفحون كثيرًا الدوريات أو أعمال المؤتمرات ذات التركيز الحاسوبي. وخاصةً إذا كان المقصود أن تكون البرمجية في متناول علماء المختبرات أو غيرهم من أصحاب المصلحة غير الحاسوبيين، فإن وصف برمجيتك في دورية متخصصة في المجال طريقة ممتازة للوصول إلى جمهور أوسع. وفضلًا عن ذلك، فإن ورقة تصف أداة برمجية مبتكرة جديدة في إحدى هذه الدوريات لديها فرصة كبيرة للتمييز مقارنةً بالمقالات الأخرى، ولا سيما عندما يكون المجال قد أبرز الحاجة إلى مقاربات برمجية جديدة لتحديات قائمة منذ زمن طويل.

غير أن ثمة أمورًا ينبغي مراعاتها، خاصةً عند النشر في دورية أو مؤتمر غير حاسوبي. فإن كانت أوراق البرمجيات غير شائعة في مجالك، فقد لا يوجد نوع نشر أو مقال ملائم تمامًا، وقد تحتاج إلى الإبداع في طريقة تنظيم عرضك للبرمجية وتقييمك لأدائها. وعلى وجه التحديد، فُكر في كيفية معالجة برمجيتك لقيد معين أو سؤال بحثي يهم المجال، وقدم مثالًا يثبت أنها تستطيع ذلك. ويمكن أن يصبح هذا محورًا رئيسًا للورقة، أو أن يكون واحدًا من عدة «دراسات حالة» أقصر تستعرض وظائف مفيدة. فُكر في القصة التي تريد أن ترويها، وفيما سيجده جمهورك المستهدف أكثر فائدة. وبالمثل، قد لا يكون المحكمون معتادين على كيفية تقييم أوراق البرمجيات ونقدها. وعند الشك، لا خير أبدًا في التواصل مع محرر الدورية أو أحد أعضاء لجنة البرنامج لطلب التوجيه؛ بل قد يستطيعون توجيه المقال إلى مجموعة من المحكمين يعرفون أن لديهم الخبرة التقنية اللازمة. وإذا طلب وعاء النشر من المؤلفين اقتراح محكمين، فينبغي أن تكون قادرًا على اقتراح مجموعة مماثلة بنفسك. وينبغي أيضًا أن تضع قراءك في حسابك: فإذا كان معظم جمهورك المقصود من ذوي الخبرة الحاسوبية المحدودة، فعليك أن تقلل بفاعلية من الرطانة المتخصصة والتفاصيل التقنية. ويمكن إضافة هذه التفاصيل بوصفها بيانات تكميلية، أو نشرها منفصلةً بوصفها مقالًا غير تقليدي (مثلًا عبر (Zenodo) أو (F1000) أو ما شابههما)، أو حتى نقلها بالكامل إلى التوثيق الإلكتروني (انظر القاعدة 6).

والأمثلة التوضيحية الجيدة على أوراق برمجيات نُشرت في دوريات غير حاسوبية كثيرة. فكثير من أوراق البرمجيات القديمة نُشرت في دوريات متخصصة في المجالات، لأن معظم المجالات البينية التي أدت لاحقًا إلى ظهور الدوريات ذات التركيز الحاسوبي كانت لا تزال ناشئة. ويمكن ملاحظة ذلك، على سبيل المثال، في علم تطور السلالات الحاسوبي (computational phylogenetics) وعلم التفرع الحيوي (cladistics)، وهو مجال بدأ منذ سبعينيات القرن العشرين [18,19]. والأمثلة الأحدث على أوراق برمجيات عالية الأثر في دوريات متخصصة كثيرة أيضًا، مثل الأوراق التي قُدمت (PLINK) [20] و(Circos) [21].

القاعدة 4: ضع خطة طويلة الأمد لإدارة البرمجية:

في الأوساط الأكاديمية تتغير الانتماءات المؤسسية ومصادر التمويل والبنى التحتية التقنية كثيرًا. ولذلك يتحمل الباحثون قدرًا من المسؤولية عن إبقاء منتجات بحثنا متاحة لبقية المجتمع العلمي عندما تتغير الأمور. وعندما تُصدر برمجية جديدة أو مجموعة من الشيفرات، ينبغي أن تضع إرشادات تساعد على ضمان استمرارها، وإلا فإن أوراقك،

وأوراق الآخرين التي تعتمد على البرمجية، ستتأثر سلبًا. وتشكل هذه الإرشادات ما يمكن عدّه «خطة لإدارة البرمجية» (software management plan) [22,23]. ولإعداد هذه الخطة، قد يكون من المفيد أن تطرح على نفسك وعلى المؤلفين المشاركين الأسئلة التالية:

- من المسؤول عن صيانة البرمجية مستقبلاً إذا تغيرت الانتماءات المؤسسية؟ المؤلف الأول للورقة، أم الباحث الرئيس (principal investigator - PI) في المختبر، أم شخص آخر؟
 - ما تكلفة (إن وجدت) إبقاء البرمجية وأي موارد مرتبطة بها - كقواعد البيانات ذات الصلة، وتطبيقات الويب، وواجهات برمجة التطبيقات (APIs)، وغيرها - متاحة على الإنترنت؟ وما مصدر التمويل؟ وكيف ستمول إذا نفذ هذا المصدر؟
 - من يملك الملكية الفكرية (IP) التي تقوم عليها البرمجية؟ كثيرًا ما تكون المؤسسة أو الشركة التي يعمل فيها الباحث الرئيس للورقة، لكنها قد تكون جهة مختلفة، وقد يؤثر ذلك في كيفية صيانة البرمجية على المدى الطويل. وفضلاً عن ذلك، من الضروري معرفة كيف ستؤثر الجهة المالكة للحقوق الفكرية في الترخيص [24]. وبوجه عام، من الممارسات الجيدة اعتماد أكثر التراخيص تساهلاً مما لا يخالف سياسات الملكية أو سياسات الاستخدام أو الخصوصية.
 - هل ستقدّم تحديثات وإصلاحات للأخطاء؟ وإذا كانت التحديثات كبيرة، فهل ستُنشر أوراق متابعة (انظر القاعدة 9)؟ وهل ثمة أنشطة صيانة دورية لازمة، وإن وجدت فمن سيؤديها؟
 - ماذا سيحدث للبرمجية إذا لم تعد البيانات أو الموارد الأخرى التي تعتمد عليها متاحة؟
 - متى وكيف ستؤرشف البرمجية؟ تجعل مستودعات الشيفرة الإلكترونية (مثل GitHub) ذلك سهلاً، وتتيح أدوات مثل (Zenodo) و(FigShare) ربط معرفات (DOI) دائمة بأرشيفات محددة (انظر القاعدة 5).
- وبوجه عام، لا تُعرض خطط إدارة البرمجيات في متن أوراق البرمجيات نفسها، لكن تصوراً لكيفية التعامل مع دورة حياة البرمجية - إلى جانب السياسات والاستراتيجيات العامة لصيانتها - كثيرًا ما يُدرج في مستودعات الشيفرة الإلكترونية، كما في أدلة «(Contributing)» أو في ملف (README) الخاص بالبرمجية (مثل [25-27]). والنصائح والأدلة العامة عن إعداد خطط إدارة البرمجيات متوافرة بكثرة على الإنترنت [22,23].

القاعدة 5: احفظ من «تعطّل الروابط»:

مع تقادم الأوراق، يشيع للأسف أن تتعطل الروابط التسمية تعطلًا دائمًا؛ إذ يكون المورد الذي تشير إليه قد نُقل، أو أُزيل من الإنترنت، أو تأثر بمشكلة أخرى تتعلق بالإنترنت. ويُعرف ذلك بـ«تعطّل الروابط» (link rot)، ولا يقتصر على المقالات الأكاديمية؛ فقد يصيب المدونات ومنشورات وسائل التواصل الاجتماعي وصفحات الويب وغيرها من الموارد الرقمية [28]. غير أنه منتشر بوجه خاص في المقالات العلمية؛ فقد وجدت دراسة أجراها هينيسي (Hennessy) وغي (Ge) عام 2013 أن العمر الوسيط لمحدد موقع المورد الموحد (uniform resource locator - URL) يبلغ 9.3 سنوات، وأن بعضها أقصر من ذلك بكثير [29]. وبينما يمكن تحرير المدونات ووثائق (README) والشيفرة المصدرية لتشير إلى روابط جديدة، فإن نصوص الأوراق المحكّمة ثابتة؛ فما لم تُصدر تصحيحًا أو استدراكًا، فإن عنوان (URL) الذي تستخدمه وقت النشر هو العنوان الذي سيبقى في تلك الورقة إلى الأبد.

ويمكن اتخاذ عدة خطوات سهلة نسبياً لمنع تعطل الروابط في الأوراق التي تتناول البرمجيات العلمية. فالانتماءات المؤسسية وبنية المواقع الإلكترونية قد تتغير (كما ذكر في القاعدة 4)، ولذلك من الأفضل استضافة تطبيقات الويب وواجهات برمجة التطبيقات وأوصاف البرمجيات وأمثلة الشيفرة وغيرها من الموارد الرقمية إما على نطاق مخصص، أو على موقع مختبر مستضاف استضافة مستقلة، أو على مورد استضافة ويب مجاني (مثل GitHub Pages). ولكن تعرّف على طريقة تعامل المضيف مع الروابط الدائمة، فعلى سبيل المثال، قد تتعطل الروابط في مواقع (GitHub Pages) إذا نُقل المستودع إلى مالك جديد. وعندما يلزم نقل الموارد الرقمية إلى عنوان (URL) جديد، ينبغي أن تسعى إلى إعداد إعادة توجيه من الموقع القديم إلى الموقع الجديد، وهو ما يمكن ترتيبه عادةً مع مديري خوادم الويب. وينبغي أيضاً إعداد إصدارات دائمة مرقّمة من البرمجية، وتخصيص معرفات (DOI) منفصلة تشير إلى إصدار البرمجية الحالي وقت نشر الورقة. و (Zenodo) (لإصدارات البرمجيات الموسومة في (GitHub)) و (FigShare) (ملفات البيانات والنصوص البرمجية وغيرها من الموارد الرقمية) خدمتان مجانيان تتبعان المواد البحثية المؤرشفة أرشفة دائمة وتخصصان لها معرفات (DOI) «تحل» مشكلة تعطل الروابط أساساً عند استخدامها بفاعلية. كما أن وجود نظام موثّق جيّداً لتخصيص عناوين (URL) ذات معنى للموارد المفردة يمكن أن يساعد على تشخيص السبب إذا تعطلت الروابط فعلاً. فعلى سبيل المثال، من المرجح أن يكون <http://<domain>/protein/> BRCA1 أفضل بكثير من <http://<domain>/540/65df7.php?id=18427>، من منظور قابلية الاستخدام ومن منظور الصيانة على السواء.

القاعدة 6: ميّز بوضوح بين توثيق الشيفرة ونتائج البحث:

كلما كان المقصود من البرمجية أن يُعاد استخدامها، كان التوثيق عالي الجودة أمراً حاسماً. غير أن الأوراق المحكّمة ليست، على الأرجح، المكان الذي ينبغي أن يُعرض فيه التوثيق. فينبغي أن تصف الورقة البرمجية (بما في ذلك عملية التصميم والتفاصيل التقنية والابتكارات الخوارزمية) وأي تحليلات مصاحبة. وكلما أدرجت شيفرة في الورقة، فإنك تلتزم بدعم بنية تلك الشيفرة ودلالاتها. ولأنها ستظل مرئية للمستخدمين العلميين إلى الأبد، فإن التغييرات التي تُعطل الشيفرة الواردة في الأمثلة ستؤدي على الأرجح إلى الارتباك، وقد تقضي إلى نفور المستخدمين. فإذا كان من المهم بوجه خاص عرض أمثلة الاستخدام أو غيرها من التعليمات عن كيفية استخدام البرمجية، وكانت تشغل أكثر من بضع جمل قليلة، فينبغي إما (1) نقلها إلى ملحق أو وثيقة مواد تكميلية؛ أو (2) وضعها في توثيق الشيفرة. وإذا كانت لديك صفحات توثيق مخصصة على الإنترنت، فمن الأفكار الرائعة أن تقدم رابطاً إلى تلك الصفحات في متن الورقة. ولضمان الاتساق، ينبغي أن يخضع التوثيق أيضاً للتحكم في الإصدارات، وأن يشير الرابط في الورقة إلى إصدار التوثيق الساري وقت الكتابة. وللإشارة، يمكن أيضاً وضع عينات المدخلات والمخرجات وأمثلة الشيفرة التي تدعم النتائج المعروضة في الورقة في نظام التحكم في إصدارات البرمجية، بل دمجها في مجموعة اختبارات البرمجية بوصفها اختبارات قبول [30].

القاعدة 7: واكب الأدوات الحديثة وأفضل ممارسات كتابة الشيفرة:

كثير من الخيارات التي تتخذها في تطوير برمجيتك نفسها يمكن أن يكون لها أثر عميق في طول عمر الورقة التي تصفها وفي أثرها العلمي. فإذا كانت البرمجية تلبي حاجة مهمة غير ملّاة، لكنها صعبة التثبيت، ومكتوبة بلغة برمجة متقدمة، وملئية بالأخطاء، فمن المرجح ألا تحظى باستخدام واسع. وبالمثل، ستكون الورقة معرضة بدرجة كبيرة لأن تُهمل، هذا إن اجتازت التحكيم العلمي أصلاً. ولحسن الحظ، يمكن لقدر صغير نسبياً من التخطيط المسبق في المراحل المبكرة من التطوير أن يساعد على تجنب هذه المشكلة بعينها. ونجد أن الإرشادات التالية مفيدة هنا:

- استخدم لغة برمجة تحظى بصيانة جيدة وتعمل على معظم الأنظمة الحديثة.
- انشر برمجيتك في فهرس حزم واحد على الأقل بحيث يستطيع المستخدمون تثبيتها بأمر واحد. ويمكن لاستخدام خدمة للتكامل المستمر (continuous integration - CI) أتمتة هذه العملية وتقليل احتمال الخطأ البشري [31].
- وبالمثل، وزّع برمجيتك، إن أمكن، بوصفها شيفرة مصدرة غير محرّمة وبوصفها إصداراً محرّماً أو مترجماً إلى شيفرة تنفيذية معاً.
- وكما ذكر سابقاً، قدّم توثيقاً مفصلاً وتعليمات للاستخدام.
- وإن أمكن، وقرّ للمستخدمين سبلاً للإسهام في التطوير المستقبلي، ولا سيما في إصلاح الأخطاء والميزات المطلوبة.

والقيام بذلك مهم أيضاً لأسباب عملية أكثر جوهرية: فمن الطرق الجيدة لتشجيع الاستخدام الواسع للبرمجية جعلها سهلة التثبيت والاستخدام ومنحها مظهرًا جديدًا وحديثًا. ومع أن هذا لا يتصل مباشرة بالجودة العلمية للبرمجية أو للورقة، فإن نشر البحث وأدوات البحث جزء مهم من العملية العلمية، وينبغي أن يحظى دائماً بتفكير خاص.

القاعدة 8: حافظ على الاتساق بين الشيفرة والتوثيق والمواقع الإلكترونية والأوراق:

تحتاج أي ورقة برمجيات، بطبيعتها ذاتها، إلى إدارة الحالات إلى منظومة من الموارد الرقمية التي تصف البرمجية (والحالات فيما بينها)، بما في ذلك المواقع الإلكترونية، ومستودعات الشيفرة المصدرة، والتوثيق، وأمثلة الشيفرة، ومقالات المدونات، وأنواع الوسائط الأخرى، وكلها تشير إلى البرمجية نفسها. فاحرص على الحفاظ على الاتساق عبر هذه المنظومة. استخدم التهجئة وعلامات الترقيم والأحرف الكبيرة نفسها في أي أسماء تبتكرها لبرمجيتك. وإذا صممت شعاراً للبرمجية، فاستخدمه في مواضع متعددة. وأكثر من الروابط بين المواقع والموارد المختلفة كي يجد القراء ما يبحثون عنه بسرعة وسهولة. ومن الحيل السهلة لضمان اتساق الإصدارات تضمين أرقام الإصدارات مباشرة في عناوين (URL)، حيثما كان ذلك ملائماً. فعلى سبيل المثال، قد تُعطى صفحات التوثيق عنوان (URL) مثل (<http://<domain>/version1/doc>). وإذا كانت البرمجية جزءاً من مجموعة بحثية أكبر أنتجت برمجيات أخرى، فقد يكون من الجيد وضع نظام للتسمية يبيّن العلاقة بينها، مع ضمان إمكانية الإشارة إلى كل منها بسهولة ودون لبس. ولا تُقحم المختصرات في تسميتك أيضاً؛ اجعل المختصرات بسيطة أو تجنبها تماماً.

القاعدة 9: خطط لأوراق المتابعة وحدث البرمجية تبعًا لذلك:

في أغلب الأحيان لا ينتهي تطوير البرمجية بعد إصدارها الرئيس الأول؛ بل يضيف المطورون ميزات جديدة ويستجيبون للأخطاء أو لمشكلات الأداء الأخرى. وينطبق هذا بالتأكيد على البرمجيات العلمية أيضًا، وهو ما ينبع إلى حد كبير من أن البحث الجيد يكون عادةً تكراريًا [13]، ويُجرى على مراحل إما مخطط لها منذ البداية وإما موجّهة بنجاحات الخطوات السابقة وإخفاقاتها. وكتابة عدة أوراق في أثناء ذلك أكثر من مجرد حيلة لتضخيم مقاييس الاستشهاد أو تعزيز السيرة الذاتية؛ إنها دليل على الالتزام الصارم بالعملية العلمية، وتتيح لك نشر النتائج الجديدة في المجتمع بسرعة.

غير أن من المهم أيضًا أن تعرف متى يكون من الملائم كتابة ورقة متابعة. فأمور مثل إصلاح الأخطاء أو التحسينات الطفيفة في قابلية الاستخدام أنسب لتدوينات المدونات، أو ملاحظات إصدار النسخ، أو سلاسل النقاش في لوحات الرسائل أو متبوعات المشكلات. ناقش ما إذا كان التحديث الجديد يمثل تقدمًا علميًا جديدًا. وما إذا كان هذا التقدم يلبي حاجة يواجهها مستخدموك حاليًا. وبوجه عام، نسعى إلى نشر ورقة جديدة لكل ميزة رئيسية جديدة ترتبط بسؤال بحثي محدد قائم. فعلى سبيل المثال، من المشروعات الجارية في مجموعتنا البحثية تطوير أداة تحسين خطوط المعالجة القائمة على الأشجار (Tree-based Pipeline Optimization Tool - TPOT)، وهي أداة للتعليم الآلي المؤتمت تستخدم البرمجة الجينية للعثور تلقائيًا على خطوط معالجة للتعليم الآلي تؤدي أداءً جيدًا على مجموعة بيانات معينة [32]. وإضافةً إلى المنشور الأصلي الذي وصف (TPOT)، كتبنا أوراق متابعة لعدة إضافات رئيسية إلى البرمجية، منها طرق جديدة لتحديد قوالب خطوط المعالجة [33]، ودعم التعلم العميق [34].

القاعدة 10: أعط الأولوية للظهور والإتاحة:

ثمة سيناريو مُحبط يتكرر كثيرًا عند إجراء البحث الحاسوبي: تعثر على ورقة تصف برمجية مجانية متاحة للعموم تحل تمامًا مشكلة تعاني منها منذ عدة أسابيع. الورقة قديمة بعض الشيء، لكن الأساليب تبدو أنيقة ومتينة. غير أنك بعد أن تعثر أخيرًا على نسخة من البرمجية، تجد أنه لا سبيل إلى تشغيلها على أي نظام تشغيل حديث. فتقضي بضع ساعات في محاولة العثور على توثيقها (غير الموجود على ما يبدو)، ثم تستسلم في النهاية، وتقرر إما أن تشغيل البرمجية مستحيل، أو أن تنفيذها بنفسك سيستغرق ببساطة وقتًا أقل. ولا يمكن تجنب مشكلات كهذه تمامًا؛ فالبرمجيات تشيخ، ولغات البرمجة تفقد شعبيتها في نهاية المطاف، والاعتماديات تتغير بطرق لا تستطيع بوصفك مستخدمًا التحكم فيها تحكمًا كاملًا. لكنك بوصفك مطورًا تستطيع اتخاذ خطوات لإطالة عمر برمجيتك بفاعلية، ويمكن تطبيق بعض هذه الخطوات مباشرة في الورقة التي تصف البرمجية.

أولًا، التكرار لا يضر. فإذا كان لديك موقع إلكتروني تعريفى رئيس للبرمجية، فأدرج رابطًا إليه في الورقة، وكذلك في المستودع المصدري، وفي التوثيق، وعلى مواقع المختبر والمؤسسة. واستفد من وسائل التواصل الاجتماعي للترويج لعملك، وشجّع المؤلفين المشاركين على فعل الشيء نفسه [35]. ومن المقاييس الشائعة لتحديد الأثر الاجتماعي للمقال مؤشر الاهتمام (Altmetric Attention Score) [36]، الذي يرصد مظاهر الاهتمام، مثل الإشارات في وسائل التواصل الاجتماعي، والتغطية الإخبارية، والحضور في منشورات العلوم المبسطة. وتتيح لك «أيقونات الشارات» (badge icons) (المعروفة أحيانًا بـ«الدروع» shields)) المستخدمة في المواقع الإلكترونية ومستودعات الشيفرة وفهارس الحزم تقديم روابط غنية إلى أجزاء مختلفة من منظومة برمجيتك (بما في ذلك

الورقة نفسها عبر معرفّات (DOI)، وهي روابط ديناميكية وكثيفة المعلومات وجذابة بصريًا. وأخيرًا، يمكن تحسين كل من ورقة برمجيتك والوسائط المرتبطة بالبرمجية لمحركات البحث، وهو ما قد يزيد أثرها العلمي زيادة كبيرة [37].

وكثير من هذه الخطوات ينطبق في الاتجاه المعاكس أيضًا. فبمجرد نشر ورقتك (وقبل ذلك، إن أصدرت نسخة أولية (preprint))، ينبغي أن تشير مواقعك الإلكترونية ومستودعات الشيفرة لديك إلى الورقة، باستخدام معرفّ (DOI) الخاص بها حيثما أمكن. ومن المفيد أيضًا أن توجه المستخدمين صراحةً إلى كيفية الاستشهاد بعملك، وأن تقدم استشهادات منسقة مسبقًا بعدة أساليب شائعة أو ملفات (BibTeX/EndNote) أو غيرها يمكن تصديرها مباشرة إلى برمجيات إدارة الاستشهادات.

الخاتمة:

تُعد أوراق البرمجيات مكونًا مهمًا من مكونات منظومة البحث العلمي، إذ تفيد المستخدمين في مجالات كثيرة وبمستويات متفاوتة جدًا من الخبرة الحاسوبية. وفضلًا عن ذلك، توفر المنشورات الأكاديمية (والاستشهادات بها) حاليًا الوسيلة الرئيسة التي يحظى بها مطورو البرمجيات العلمية والقائمون على صيانتها بالتقدير لعملهم (ولحسن الحظ، تجري حاليًا جهود لتغيير ذلك؛ فعلى سبيل المثال، يبين المرجعان [38,39] كيف يمكن استخدام الإسهامات البرمجية بوصفها أعمالًا علمية قابلة للاستشهاد بها مباشرة). وسيساعد اتباع هذه القواعد البسيطة العشر على ضمان أن تكون أوراق برمجياتك سهلة الاستخدام، ورسنية علميًا، ومقاومة للتغيرات المستقبلية في التقنية.

المراجع:

1. Humphreys P. Extending ourselves: Computational science, empiricism, and scientific method. Oxford University Press; 2004.
2. Brito JJ, Li J, Moore JH, Greene CS, Nogoy NA, Garmire LX, et al. Recommendations to enhance rigor and reproducibility in biomedical research. GigaScience. 2020;9(6):giaa056. pmid:32479592
3. Chapman B, Chang J. Biopython: Python tools for computational biology. ACM Sigbio Newsletter. 2000;20(2):15-19.
4. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. J Mach Learn Res. 2011;12:2825-2830.
5. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, et al. The sequence alignment/map format and SAMtools. Bioinformatics. 2009;25(16):2078-2079. pmid:19505943
6. Mootha VK, Lindgren CM, Eriksson KF, Subramanian A, Sihag S, Lehar J, et al. PGC-1 α -responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes. Nat Genet. 2003;34(3):267-273. pmid:12808457

7. Lamb J, Crawford ED, Peck D, Modell JW, Blat IC, Wrobel MJ, et al. The Connectivity Map: using gene-expression signatures to connect small molecules, genes, and disease. *Science*. 2006;313(5795):1929-1935. pmid:17008526
8. Alvarez MJ, Shen Y, Giorgi FM, Lachmann A, Ding BB, Ye BH, et al. Functional characterization of somatic mutations in cancer using network-based inference of protein activity. *Nat Genet*. 2016;48(8):838-847. pmid:27322546
9. Taschuk M, Wilson G. Ten simple rules for making research software more robust; 2017.
10. Lee BD. Ten simple rules for documenting scientific software; 2018.
11. List M, Ebert P, Albrecht F. Ten simple rules for developing usable software in computational biology; 2017. pmid:28056032
12. Perez-Riverol Y, Gatto L, Wang R, Sachsenberg T, Uszkoreit J, Leprevost FdV, et al. Ten simple rules for taking advantage of Git and GitHub; 2016.
13. Prlić A, Procter JB. Ten simple rules for the open development of scientific software. *PLoS Comput Biol*. 2012;8(12):e1002802. pmid:23236269
14. Rule A, Birmingham A, Zuniga C, Altintas I, Huang SC, Knight R, et al. Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks. 2019.
15. Helmy M, Crits-Christoph A, Bader GD. Ten simple rules for developing public biological databases; 2016.
16. Scope Guidelines—Bioinformatics—Oxford Academics; 2020. Available from: https://academic.oup.com/bioinformatics/pages/scope_guidelines.
17. Submitting experimental metadata—Submission Guidelines—Scientific Data; 2020. Available from: <https://www.nature.com/sdata/publish/submission-guidelines#metadata>.
18. Felsenstein J. Phylogenetic inference package (PHYLIP), version 3.2. *Cladistics*. 1989;5:164-166.
19. Farris J. Hennig86: a PC-DOS program for phylogenetic analysis. *Cladistics*. 1989;5(2):163.
20. Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MA, Bender D, et al. PLINK: a tool set for whole-genome association and population-based linkage analyses. *Am J Hum Genet*. 2007;81(3):559-575. pmid:17701901
21. Krzywinski M, Schein J, Birol I, Connors J, Gascoyne R, Horsman D, et al. Circos: an information aesthetic for comparative genomics. *Genome Res*. 2009;19(9):1639-1645. pmid:19541911

22. Writing and using a software management plan. Available from: <https://www.software.ac.uk/resources/guides/software-management-plans>.
23. Kuzak M, Crouch S. Developing an ELIXIR Software Management plan for use in Life Science Project; 2020. Available from: <https://elixir-europe.org/events/webinar-software-management-plans>.
24. Morin A, Urban J, Sliz P. A quick guide to software licensing for the scientist-programmer. PLoS Comput Biol. 2012;8(7):e1002598. pmid:22844236
25. Contribute to NumPy—NumPy; 2020. Available from: <https://numpy.org/contribute/>.
26. Contribution guide—Penn Machine Learning Benchmarks; 2020. Available from: <https://epistasislab.github.io/penn-ml-benchmarks/contributing.html>.
27. Developer's Guide—Scikit-Learn documentation; 2020. Available from: <https://scikit-learn.org/stable/developers/index.html>.
28. Markwell J, Brooks DW. "Link rot" limits the usefulness of web-based educational materials in biochemistry and molecular biology. Biochem Mol Biol Educ. 2003;31(1):69-72.
29. Hennessey J, Ge SX. A cross disciplinary study of link decay and the effectiveness of mitigation techniques. In: BMC bioinformatics. vol. 14. Springer; 2013. p. S5.
30. Haugset B, Hanssen GK. Automated acceptance testing: A literature review and an industrial case study. In: Agile 2008 Conference. IEEE; 2008. p. 27-38.
31. Hilton M, Tunnell T, Huang K, Marinov D, Dig D. Usage, costs, and benefits of continuous integration in open-source projects. In: 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE; 2016. p. 426-437.
32. Olson RS, Moore JH. TPOT: A tree-based pipeline optimization tool for automating machine learning. In: Automated machine learning. Springer, Cham; 2019. p. 151-160.
33. Le TT, Fu W, Moore JH. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. Bioinformatics. 2020;36(1):250-256. pmid:31165141
34. Romano JD, Le TT, Fu W, Moore JH. Is deep learning necessary for simple classification tasks? arXiv preprint arXiv:200606730. 2020.
35. Cheplygina V, Hermans F, Albers C, Bielczyk N, Smeets I. Ten simple rules for getting started on Twitter as a scientist; 2020.
36. Trueger NS, Thoma B, Hsu CH, Sullivan D, Peters L, Lin M. The altmetric score: a new measure for article-level dissemination and impact. Ann Emerg Med. 2015;66(5):549-553. pmid:26004769

37. Beel J, Gipp B, Wilde E. Academic Search Engine Optimization (aseo) Optimizing Scholarly Literature for Google Scholar & Co. J Sch Publ. 2010;41(2):176-190.
38. Park H, Wolfram D. Research software citation in the Data Citation Index: Current practices and implications for research software sharing and reuse. J Informet. 2019;13(2):574-582.
39. Druskat S. Software and Dependencies in Research Citation Graphs. Computing in Science & Engineering. 2019;22(2):8-21.