

عشر قواعد لإدارة المشروعات البحثية والتعاون باستخدام (Git) و (GitHub)

ياسيت بيريز-ريفيرو، لوران غاتو، روي وانغ، تيمو زاكسنبرغ، يوليان أوشكوريت، فيليبي دا فيغا ليريفوست، كريستيان فوفيزان، توبياس تيرنيت، ستيفن ج. إغلين، دانيال س. كاتس، توم ج. بولارد، ألكسندر كونوفالوف، روبرت م. فلايت، كاي بلين، خوان أنطونيو فيثكاينو



المخلص:

يقدم هذا المقال، الموجه أساساً إلى الباحثين في المعلوماتية الحيوية وفي مجالات بحثية أخرى، عشر توصيات عملية للاستفادة من منصة (GitHub) القائمة على نظام التحكم في الإصدارات (Git) في إدارة المشروعات البحثية وتميزها. وتتبع القواعد مسار التطوير المعتاد: تعلم أساسيات (Git) لتتبع الشفرة والوثائق، وتنظيم الصلاحيات للأفراد والفرق والمنظمات، والتعاون عبر الفروع والنسخ المتفرعة وطلبات السحب، ووسم الإصدارات وفق ترقيم الإصدارات الدلالي، وأتمتة الاختبار والتكامل المستمر وقياس تغطية الاختبارات وتوليد التوثيق، واستخدام البلاغات لمناقشة الأخطاء والتحسينات علناً. كما يتناول جعل الشفرة قابلة للاستشهاد بمعرفات (DOI) عبر خدمات الأرشفة، واختيار ترخيص معتمد للمصدر المفتوح، والترويج للمشروعات بصفحات (GitHub) وأدوات التواصل، ومتابعة المستخدمين والمشروعات الأخرى. ويعرض المقال جدولين بأثلة لمستودعات ومواد تدريبية، ويخلص إلى أن العائد يستحق الجهد المبذول في التعلم.

الكلمات المفتاحية: نظم التحكم في الإصدارات، (Git) و (GitHub)، البرمجيات البحثية، التكامل المستمر، الاستشهاد بالبرمجيات

مقدمة:

المعلوماتية الحيوية تخصص واسع، من القواسم المشتركة فيه الحاجة إلى إنتاج برمجيات و/أو استخدامها لتطبيقها على البيانات البيولوجية في سياقات مختلفة. ولتمكين قابلية تكرار الادعاءات العلمية وتتبعها وضمانها، من الضروري أن تُتاح للعموم الورقة العلمية المنشورة ومجموعات البيانات المقابلة لها وتحليل البيانات [1,2]. وينبغي أن تكون جميع البرمجيات المستخدمة في التحليل إما موثقة توثيقاً دقيقاً (كما في حالة البرمجيات التجارية مثلاً)، أو - وهو الأفضل - متاحة علناً ويمكن للآخرين الوصول إليها مباشرة [3,4]. ويسهل انتشار البرمجيات والشفرة

المصدرية المتاحة علنًا، وما يصاحبه من تطوير تعاوني، وجود عدد من خدمات مستودعات الشفرة، مثل (SourceForge) و (Bitbucket) و (GitLab) و (GitHub) وغيرها. وهذه الموارد ضرورية أيضًا لمشروعات البرمجيات التعاونية، لأنها تتيح تنظيم المهام البرمجية وتقاسمها بين مساهمين متباعدين. ونقدم هنا الميزات الرئيسة لمنصة (GitHub)، وهي منصة شائعة على الويب توفر بيئة مجانية ومتكاملة لاستضافة الشفرة المصدرية والتوثيق ومحتوى الويب المتعلق بالمشروعات مفتوحة المصدر. ويقدم (GitHub) أيضًا خططًا مدفوعة للمستودعات الخاصة للأفراد والشركات، فضلًا عن خطط مجانية تشمل مستودعات خاصة للاستخدام البحثي والتعليمي.

المستودعات العامة والخاصة:

مستودعات (GitHub) مرئية للجميع مجانًا افتراضيًا. وتقرر مشروعات كثيرة أن تشارك عملها علنًا وبصورة مفتوحة منذ بداية المشروع، لاكتساب الظهور والإفادة من إسهامات المجتمع مبكرًا. وتفضل مجموعات أخرى أن تعمل على المشروعات بصورة خاصة إلى أن تصبح مستعدة لمشاركة عملها. وتضمن المستودعات الخاصة إخفاء العمل، لكنها تقصر التعاون أيضًا على المستخدمين الذين يُمنحون حق الوصول إلى المستودع. ويمكن بعد ذلك جعل هذه المستودعات عامة في مرحلة لاحقة، مثل مرحلة تقديم مقالات الدوريات المقابلة لها أو قبولها أو نشرها. وفي بعض الحالات، حين يكون التعاون مقصودًا به أن يبقى خاصًا حصريًا، قد لا تُتاح بعض المستودعات للعموم أبدًا.

يعتمد (GitHub) في جوهره على نظام التحكم في الإصدارات (version control system) المعروف ومفتوح المصدر (Git)، الذي صممه في الأصل لينوس تورفالدس (Linus Torvalds) لتطوير نواة لينكس، ويطوّره ويصونه الآن مجتمع (Git). ومن أسباب نجاح (GitHub) أنه يقدم أكثر من مجرد خدمة بسيطة لاستضافة الشفرة المصدرية [5,6]؛ إذ يوفر للمطورين والباحثين بيئة ديناميكية وتعاونية، يُشار إليها كثيرًا بمنصة البرمجة الاجتماعية (social coding platform)، تدعم المراجعة من الأقران والتعليق والنقاش [7]. وقد وجدت جهود متنوعة، تتراوح بين المشروعات الفردية ومشروعات المعلوماتية الحيوية الكبيرة ومستودعات المختبرات، فضلًا عن أشكال التعاون العالمية، أن (GitHub) مكان مثير لتبادل الشفرات والأفكار وللتعاون (انظر الجدول 1).

الجدول 1. أمثلة على مستودعات للمعلوماتية الحيوية تتبع ممارسات جيدة في استخدام (GitHub):

يتضمن الجدول اسم المستودع، ونوع المثال (تتبع البلاغات، وبنية الفروع، واختبارات الوحدات)، ورابط المثال. وتبدأ جميع الروابط بالبادئة.

اسم المستودع	النوع	الرابط
(Adam)	مشروع مجتمعي، نسخ متفرعة متعددة	رابط المصدر
[18] (BioPython)	مشروع مجتمعي، مساهمون متعددون	رابط المصدر
(Computational Proteomics Unit)	مستودع مختبر	رابط المصدر
[19] (Galaxy Project)	مشروع مجتمعي، مستودع للمعلوماتية الحيوية	رابط المصدر
(GitHub Paper)	مخطوطة، نقاش البلاغات، مشروع مجتمعي	رابط المصدر
[20] (MSnbase)	مستودع مشروع فردي	رابط المصدر

اسم المستودع	النوع	الرابط
(OpenMS) [21]	مستودع للمعلوماتية الحيوية، نقاش البلاغات، الفروع	رابط المصدر
(PRIDE Inspector Toolsuite) [22]	منظمة مشروع، مشروعات متعددة	رابط المصدر
(Retinal wave data repository) [23]	مشروع فردي، مخطوطة، بيانات ثنائية منظمة	رابط المصدر
(SAMtools) [24]	مستودع للمعلوماتية الحيوية، منظمة مشروع	رابط المصدر
(rOpenSci)	مشروع مجتمعي، نقاش البلاغات	رابط المصدر
(The Global Alliance For Genomics and Health)	مشروع مجتمعي	رابط المصدر

تنطبق بعض التوصيات الواردة أدناه على خدمات استضافة المستودعات عمومًا، غير أن هدفنا الرئيس هو إبراز ميزات محددة في (GitHub). ونقدّم مجموعة من التوصيات نعتقد أنها ستساعد القارئ على الاستفادة الكاملة من ميزات (GitHub) في إدارة المشروعات والترويج لها في المعلوماتية الحيوية وفي مجالات بحثية أخرى كثيرة. ورُتبت التوصيات بحيث تعكس عملية تطوير نموذجية: تعلّم أساسيات (Git) و (GitHub)، والتعاون، واستخدام الفروع وطلبات السحب، وتسمية النسخ المحفوظة من الشفرة ووسمها، وتتبع أخطاء المشروع وتحسيناته باستخدام البلاغات، ونشر النتائج النهائية.

القاعدة 1: استخدم (GitHub) لتتبع مشروعاتك:

الأساس الذي يقوم عليه (GitHub) هو نظام التحكم في الإصدارات الموزّع (Git). فكل تغيير، من تصحيح خطأ مطبعي إلى إعادة تصميم كاملة للبرمجية، يُتَبَّع ويُعرَّف تعريفًا فريدًا. ومع أن لنظام (Git) مجموعة معقدة من الأوامر ويمكن استخدامه في عمليات معقدة نوعًا ما، فإن تعلّم تطبيق أساسياته لا يتطلب سوى عدد قليل من المفاهيم والأوامر الجديدة، وسيوفر أساسًا متينًا لتتبع الشفرة والمحتوى المرتبط بها في المشروعات البحثية بكفاءة. وتتوافر أدلة تعليمية كثيرة، تمهيدية ومفصلة (انظر الجدول 2 أدناه لبعض الأمثلة). ونوصي على وجه الخصوص بدليل «مقدمة سريعة إلى التحكم في الإصدارات باستخدام (Git) و (GitHub)» (A Quick Introduction to Version Control with Git and GitHub) لبليشاك (Blischak) وزملائه [5].

الجدول 2. مقررات ودروس تعليمية وورش عمل على الإنترنت حول (GitHub) و (Git) للعلماء:

اسم المادة	الرابط
(git help) و (git help -a)	وثيقة تُنَبِّت مع (Git)
(Karl Broman's Git/GitHub Guide) (دليل (Git/GitHub) لكارل برومان)	رابط المصدر
(Version Control with Git)	رابط المصدر
(Introduction to Git)	رابط المصدر
(Github Training)	رابط المصدر
(Github Guides)	رابط المصدر

اسم المادة	الرابط
(Good Resources for Learning Git and GitHub)	رابط المصدر
(Software Carpentry): (Version Control with Git)	رابط المصدر

باختصار، تؤدي تهيئة مستودع (محلي) ((repository)، ويختصر كثيرًا إلى (repo)) إلى تحديد مجلد بوصفه مجلدًا سيُتبع. ويمكن إضافة محتواه كله أو أجزاء منه صراحةً إلى قائمة الملفات المتتبع.

```
cd project ## move into directory to be tracked
git init ## initialize local repository
## add individual files such as project description, reports, source code
git add README project.md code.R
git commit -m "initial commit" ## saves the current local snapshot
```

بعد ذلك، يُسجل كل تغيير في الملفات المتتبع، متى أُودع (committed)، بوصفه مراجعة (revision) أو لقطة (snapshot) جديدة تعرف التغييرات في جميع الملفات المعدلة تعريفًا فريدًا. و(Git) فعال وكفاء على نحو لافت في أرشفة التاريخ الكامل للمشروع، وذلك بطرق منها تخزين الفروق بين الملفات فقط.

وإلى جانب النسخ المحلية من المستودع، من اليسير إنشاء مستودعات بعيدة على (GitHub) (تسمى (origin)، وفرعها الافتراضي (master) - انظر أدناه) باستخدام واجهة الويب، ثم مزامنة المستودعات المحلية والبعيدة.

```
git push origin master ## push local changes to the remote repository
git pull origin master ## pull remote changes into the local repository
```

واتباعًا لنصيحة توني روسيني (Tony Rossini) في عام 2005: «أودع مبكرًا، وأودع كثيرًا، وأودع في مستودع نستطيع منه بسهولة التراجع عن أخطائك»، يمكن للمرء أن ينظم عمله في تغييرات تدريجية صغيرة. ويمكن في أي وقت الرجوع إلى إصدار سابق. وفي المشروعات الأكبر، يستطيع مستخدمون متعددون العمل على المستودع البعيد نفسه، مع تسجيل جميع الإسهامات وإمكانية استعادتها ونسبتها إلى أصحابها.

ويتتبع المستخدمون عادةً داخل مستودعاتهم الشفرة المصدرية والملفات النصية والصور وملفات البيانات الصغيرة، ولا يتتبعون عمومًا الملفات المشتقة مثل سجلات البناء أو الملفات الثنائية المترجمة (اقرأ الإطار 2 لترى كيفية التعامل مع الملفات الثنائية الكبيرة في (GitHub)). ومع أن أغلب مستودعات (GitHub) تُستخدم لتطوير البرمجيات، يمكن للمستخدمين أيضًا حفظ وثائق نصية مثل تقارير التحليل والمخطوطات.

الإطار 2: التعامل مع الملفات الثنائية الكبيرة:

ليس استخدام (GitHub) أو أي نظام مشابه للإصدارات والتتبع بديلًا عن الإدارة الجيدة للمشروع، بل هو امتداد لها، وتحسين للإدارة الجيدة للمشروعات والملفات (انظر مثلًا [9]). ومن الاعتبارات العملية عند استخدام (GitHub)، على سبيل المثال، التعامل مع الملفات الثنائية الكبيرة. فالملفات الثنائية، مثل الصور ومقاطع الفيديو والملفات التنفيذية أو كثير من البيانات الخام المستخدمة في المعلوماتية الحيوية، تُخزن في (Git) كيانًا واحدًا كبيرًا. ونتيجة لذلك، يؤدي كل تغيير، مهما صغر، إلى نسخة جديدة كاملة من الملف في المستودع، مما ينتج زيادات كبيرة في الحجم، ويحول دون البحث ومقارنة محتوى الملفات عبر المراجعات. ويوفر (Git) وحدة لتخزين الملفات الكبيرة (Large File

(Storage, LFS) تستبدل بهذه الملفات الكبيرة مؤشرات، في حين يمكن تخزين الملف الثنائي الكبير عن بُعد، مما ينتج مستودعات أصغر وأسرع. ويدعم (GitHub) أيضًا (Git LFS)، وإن كان ذلك ضمن حصة محددة من المساحة أو مقابل رسوم، للحفاظ على سير العمل المعتاد في (GitHub) (الملف (S1)، القسم 1).

وبفضل التصميم الموزع، يُعد كل مستودع (Git) محلي محدّث نسخة تاريخية كاملة ومطابقة لكل ما أُودع - من تغييرات الملفات وسجلات رسائل الإيداع وغيرها. وتعمل هذه النسخ أيضًا بوصفها نسخًا احتياطية مستقلة موجودة على جهاز التخزين لدى كل مستخدم. ولهذا يمكن عدّ (Git) قادرًا على تحمل الأعطال (fault-tolerant)، وهي ميزة يتفوق بها على نظم التحكم في الإصدارات المركزية. فإذا تعدّر الوصول إلى خادم (GitHub) البعيد، يمكن أن يستمر التعاون والعمل بين المستخدمين، على خلاف البدائل المركزية.

وتوفر واجهة الويب التي يقدمها (GitHub) أدوات سهلة الاستخدام لأداء كثير من العمليات الأساسية، ومدخلًا ميسرًا إلى مجموعة أغنى وأعقد من الوظائف. وتتوافر كذلك برامج عميلة متنوعة ذات واجهات مستخدم رسومية لإدارة مستودعات (Git) و(GitHub). وتتكامل محررات وبيئات تطوير كثيرة - مثل محرر (RStudio) الشائع للغة البرمجة (R) [8] - تكاملًا مباشرًا مع إدارة إصدارات الشفرة باستخدام (Git) و(GitHub). وفضلًا عن ذلك، يقدم (GitHub) للمستودعات البعيدة لنظام (Git) سماته الخاصة التي سَنُوصف في القواعد التالية.

القاعدة 2: (GitHub) للمستخدمين الأفراد والفرق والمنظمات:

المشروعات العامة على (GitHub) مرئية للجميع، لكن صلاحية الكتابة، أي القدرة على تعديل محتوى المستودع مباشرة، يجب أن تُمنح صراحةً. وبوصفك مالكًا لمستودع، يمكنك منح هذا الحق لمستخدمين آخرين على (GitHub). وإلى جانب امتلاك المستخدمين للمستودعات، يمكن أيضًا إنشاء المستودعات وإدارتها بوصفها جزءًا من فرق ومنظمات.

ويستطيع مديرو المشروعات هيكلية المشروعات لإدارة الصلاحيات على مستويات مختلفة: المستخدمين والفرق والمنظمات. والمستخدمون هم العنصر المركزي في (GitHub) كما في أي شبكة اجتماعية أخرى. فكل مستخدم ملف تعريف ييسر مشروعاته وأنشطته على (GitHub)، ويمكن اختياريًا ملؤه بمعلومات شخصية تشمل الاسم وعنوان البريد الإلكتروني والصورة وصفحة الويب. وللبقاء على اطلاع على نشاط المستخدمين الآخرين، يمكن متابعة حساباتهم (انظر أيضًا القاعدة 10). ويمكن تحقيق التعاون بمجرد إضافة متعاون (Collaborator) موثوق، ومن ثم منحه صلاحية الكتابة.

غير أن التطوير في المشروعات الكبيرة يجري عادةً على أيدي فرق من الأشخاص داخل منظمة أكبر. والمنظمات في (GitHub) طريقة ممتازة لإدارة صلاحيات الوصول القائمة على الفرق في المشروعات المفردة للمعاهد ومختبرات البحث والمشروعات الكبيرة مفتوحة المصدر التي تحتاج إلى مالكيين ومديرين متعددين. ونوصيك، بوصفك باحثًا فردًا، بأن تجعل ملفك التعريفي مرئيًا للمستخدمين الآخرين، وأن تعرض جميع المشروعات والمنظمات التي تعمل فيها.

القاعدة 3: تطوير الميزات الجديدة والتعاون عليها: التفرع (Branching) والاستنساخ المتفرع (Forking):

يستطيع أي شخص لديه حساب على (GitHub) أن ينشئ نسخة متفرعة (fork) من أي مستودع يملك حق الوصول إليه. وسينشئ ذلك نسخة كاملة من محتوى المستودع، مع الاحتفاظ برابط إلى الإصدار الأصلي «المنبع» (upstream). ويمكن للمرء بعد ذلك أن يبدأ العمل على قاعدة الشفرة نفسها في نسخته المتفرعة تحت اسم المستخدم الخاص به. وفي الممارسة العملية، كثيرًا ما يعمل المطورون بالتزامن على ميزة أو تحسين واحد أو أكثر. ولإبقاء إبداعات الميزات المختلفة منفصلة منطقيًا، تُستخدم عادةً فروع متميزة. ولاحقًا، حين يكتمل التطوير ويُتحقق من أنه يعمل (أي لا يفشل أي من الاختبارات، انظر القاعدة 5)، يمكن دمج الميزات الجديدة مرة أخرى في خط التطوير أو في فرع (master). وفضلاً عن ذلك، يمكن دائماً سحب فرع (master) المحدث حالياً إلى فرع الميزة لمواءمة الميزة مع التغييرات في فرع (master).

وعند تطوير سمات مختلفة بالتوازي، ثمة خطر إجراء تغييرات غير متوافقة في فروع أو نسخ متفرعة مختلفة؛ ويقال عندئذ إنها أصبحت غير متزامنة. والفروع ليست إلا انحرافات قصيرة الأمد عن (master). فإذا أُجريت السحب كثيرًا، فستبقى نسختك من المستودع محدثة، وستحتاج لك فرصة دمج شفرتك المعدلة مع إسهامات الآخرين، ومن الأفضل أن يتم ذلك دون أن تضطر إلى معالجة التعارضات يدويًا لإعادة الفروع إلى التزامن.

القاعدة 4: تسمية الفروع والإبداعات: الوسوم والإصدارات الدلالية:

يمكن استخدام الوسوم (tags) لتسمية الإصدارات أثناء عملية التطوير. وينبغي أن يتبع ترقيم الإصدارات ممارسة «ترقيم الإصدارات الدلالي» (semantic versioning)، بالصيغة (X.Y.Z)، حيث (X) هو الإصدار الرئيس، و(Y) الإصدار الفرعي، و(Z) إصدار الترقيع (patch) للنسخة الصادرة، مع إمكانية تضمين معلومات وصفية إضافية، كما هو موضح في. ويوفر مخطط ترقيم الإصدارات الدلالي هذا للمستخدمين أرقام إصدارات متسقة توثق مدى الإصدارات الجديدة (إصلاحات للأخطاء أو وظائف جديدة) وتوافقها مع الإصدارات السابقة. وتتيح التسمية الصحيحة للمطورين والمستخدمين استعادة الإصدارات الأقدم بسهولة، أو مقارنتها، أو مجرد استخدامها لإعادة إنتاج النتائج الموصوفة في المنشورات (انظر القاعدة 8). ويساعد هذا النهج أيضًا على تحديد استراتيجية متسقة لنشر البرمجيات.

القاعدة 5: دع (GitHub) يؤدّ بعض المهام عنك: التكامل:

القاعدة الأولى في تطوير البرمجيات أن تكون الشفرة جاهزة للاستخدام في أقرب وقت ممكن [10]، وأن تظل كذلك أثناء التطوير، وأن تكون موثقة ومختبرة جيدًا. وفي عام 2005، حدد مارتن فاولر (Martin Fowler) المبادئ الأساسية للتكامل المستمر (continuous integration) في تطوير البرمجيات [11]. وقد أصبحت هذه المبادئ المرجع الرئيس لأفضل الممارسات في التكامل المستمر، إذ توفر الإطار اللازم لنشر البرمجيات، وبصورة ما البيانات أيضًا. وإلى جانب التحقق من التنفيذ الخالي من الأخطاء، يهدف الاختبار المخصص للشفرة إلى الكشف عن الأخطاء المحتملة التي قد تُدخلها ميزات جديدة أو تغييرات في الشفرة أو في الاعتماديات، وكذلك الكشف عن النتائج الخاطئة، المعروفة غالبًا بالأخطاء المنطقية، التي تنتج فيها الشفرة المصدرية نتيجة مختلفة عما كان مقصودًا. ويوفر التكامل المستمر وسيلة لتشغيل سلسلة من الاختبارات آليًا ومنهجيًا للتحقق من سلامة الشفرة وأدائها، وهي مهمة يمكن أتمتها من خلال (GitHub).

ويقدم (GitHub) مجموعة من الخطافات (hooks) - وهي نصوص برمجية تُنفَّذ آليًا - تعمل بعد كل عملية دفع (push) إلى المستودع، مما يسهّل اتباع المبادئ الأساسية للتكامل المستمر. وتتيح خطافات الويب في (GitHub) للمنصات الخارجية الوصول إلى مستودع (GitHub) والتفاعل معه، ومن ثم أتمتة مهام المعالجة اللاحقة. ويمكن تحقيق التكامل المستمر باستخدام (Travis CI)، وهي منصة مستضافة للتكامل المستمر مجانية لجميع المشروعات مفتوحة المصدر. وتبني (Travis CI) الشفرة المصدرية وتختبرها باستخدام عدد وافر من الخيارات، مثل المنصات المختلفة وإصدارات المفسّرات المختلفة (الملف (S1)، القسم 2). وفضلًا عن ذلك، تقدم إشعارات تتيح لفريقك وللمساهمين معرفة ما إذا كانت التغييرات الجديدة تعمل، ومنع إدخال الأخطاء في الشفرة (عند دمج طلبات السحب مثلًا)، مما يجعل المستودع جاهزًا للاستخدام دائمًا.

القاعدة 6: دع (GitHub) يؤدّ مزيدًا من المهام عنك: الأتمتة:

يمكن أن يُدمج في مشروعك البرمجي أكثر من مجرد ترجمة الشفرة واختبارها؛ إذ يمكن استخدام خطافات (GitHub) لأتمتة مهام عديدة تساعد على تحسين الجودة الإجمالية لمشروعك. ومن المكملات المهمة لاجتياز الاختبارات بنجاح إثبات أن الاختبارات تغطي قاعدة الشفرة القائمة تغطية كافية. ولهذا الغرض يوصى بدمج خدمة (Codecov). فهذه الخدمة تبين مقدار ما يُنفَّذ من قاعدة الشفرة وأي أسطر الشفرة تُنفَّذ في أثناء اختبارات شيفرتك. فمشروع (Bioconductor)، على سبيل المثال، يوصي بشدة بأن تطبّق الحزم اختبار الوحدات (unit testing) (الملف (S1)، القسم 2) لدعم المطورين في تطوير حزمهم وصيانتها، ويختبر منهجيًا تغطية جميع حزمه. وقد يفكر المرء أيضًا في توليد التوثيق عند تعديل الشفرة أو التوثيق (الملف (S1)، القسم 3). ويعني ذلك أن توفر مشروعاتك توثيقًا شاملاً حتى يتمكن الآخرون من فهمها والإسهام فيها. وبالنسبة إلى شيفرات (Python) أو (C/C++)، يمكن توليد التوثيق آليًا باستخدام (sphinx) ثم دمجها في (GitHub) باستخدام «(Read the Docs)». وستنشئ جميع هذه المنصات تقارير وشارات ((badges)، تسمى أحيانًا ((shields)) يمكن تضمينها في صفحة مشروعك على (GitHub)، مما يساعد على إثبات أن المحتوى عالي الجودة وحسن الصيانة.

القاعدة 7: استخدم (GitHub) لمناقشة البلاغات ومعالجتها وإغلاقها علنًا وتعاونيًا:

بلاغات (GitHub) (issues) طريقة ممتازة لتتبع الأخطاء والمهام وطلبات الميزات والتحسينات. فبينما يُقصد بمتبوعات البلاغات التقليدية أن تُستخدم أساسًا لمتبوعات الأخطاء، تتبع متبوعات البلاغات في (GitHub)، على العكس من ذلك، فلسفة مختلفة: فلكل متبوع قسمه الخاص في كل مستودع، ويمكن استخدامه لتتبع الأخطاء والأفكار الجديدة والتحسينات باستخدام نظام وسم قوي. والهدف الرئيس من البلاغات في (GitHub) تعزيز التعاون وتوفير السياق باستخدام الحالات المتبادلة.

ولا يتطلب فتح بلاغ ملء استمارات طويلة؛ بل يتطلب عنوانًا فقط، ويُفضّل أن يُرفق به وصف قصير على الأقل. وتتميز البلاغات بتنسيق واضح جدًا، وتوفر مساحة لتعليقات اختيارية تتيح لأي شخص لديه حساب على (GitHub) تقديم ملاحظاته. فعلى سبيل المثال، إذا احتاج المطور إلى مزيد من المعلومات ليتمكن من إعادة إنتاج خطأ ما، فبوسعه أن يطلبها ببساطة في تعليق.

ومن العناصر الإضافية للبلاغات: (1) تسميات مرمّزة بالألوان تساعد على تصنيف البلاغات وتصنيفها، (2) والمعالم (milestones)، (3) ومكّلف واحد مسؤول عن العمل على البلاغ. وتساعد هذه العناصر المطورين على تصفية المهام وترتيب أولوياتها، وتحوّل متبوع البلاغات إلى أداة تخطيط لمشروعهم.

ويمكن أيضًا لمديري المستودعات إنشاء قوالب للبلاغات ولطلبات السحب (انظر القاعدة 3) لتخصيص المعلومات التي ينبغي تضمينها حين يفتح المساهمون بلاغات وتوحيدها. ومن ثم فإن بلاغات (GitHub) دينامية، وتخضع العوائق أمام المستخدمين للإبلاغ عن الأخطاء وطلب الميزات. ويساعد متتبع البلاغات المنظم والموسوم جيدًا المساهمين والمستخدمين الجدد على فهم المشروع فهمًا أعمق. ومثال ذلك أن أحد البلاغات في مستودع (OpenMS) أتاح التفاعل بين ثمانية مطورين واجتذب أكثر من مئة تعليق. ويستطيع المساهمون إضافة أشكال وتعليقات وإحالات إلى بلاغات وطلبات سحب أخرى في المستودع، فضلًا عن إحالات مباشرة إلى الشفرة.

ومن الأمثلة الأخرى على البلاغات وتطبيقاتها العامة والواسعة أننا استخدمنا، كما استخدم آخرون، بلاغات (GitHub) لمناقشة التغييرات في المخطوطات والتعليق عليها ومعالجة تعليقات المحكمين.

القاعدة 8: اجعل شفرتك قابلة للاستشهاد بها بسهولة، واستشهد بالشفرة المصدرية!

من الممارسات البحثية الجيدة ضمان معرفات دائمة وغير ملتبسة للمواد القابلة للاستشهاد بها، مثل المقالات أو مجموعات البيانات أو الكيانات البيولوجية كالبروتينات والجينات والمستقلبات (انظر أيضًا الإطار 3). وقد استُخدمت معرفات الكائنات الرقمية (Digital Object Identifiers, DOIs) سنوات عديدة معرفات فريدة وغير ملتبسة لتمكين الاستشهاد بالمنشورات العلمية. وفي الآونة الأخيرة، بدأ اتجاه إلى إصدار معرفات (DOI) لأنواع أخرى من المنتجات العلمية، مثل مجموعات البيانات [12] والمواد التدريبية (انظر مثالًا [13]). ومن الدوافع الرئيسة لذلك بناء إطار يمنح العلماء تقديرًا أوسع لعملهم [14,15]، مع دعم طرق أوضح وأدوم للاستشهاد به وتتبعه في الوقت نفسه. ومما يساعد على دفع هذا التغيير هيئات التمويل، مثل المعاهد الوطنية للصحة (National Institutes of Health, NIH) والمؤسسة الوطنية للعلوم (National Science Foundation, NSF) في الولايات المتحدة، ومجالس البحث (Research Councils) في المملكة المتحدة، التي تزداد إدراكًا لأهمية المنتجات البحثية، مثل مجموعات البيانات المتاحة للعموم والبرمجيات.

الإطار 3: ملفات الترخيص والتعريف والاستشهاد:

ينبغي، من الناحية المثالية، أن يضم كل مستودع الملفات الثلاثة التالية. الملف الأول، والأهم على الأرجح، في المستودع هو ملف (LICENCE) (انظر أيضًا القاعدة 8)، الذي يحدد بوضوح الأذونات والقيود المرتبطة بالشفرة والملفات الأخرى في مستودعك. والملف المهم الثاني هو ملف (README)، الذي يقدم مثالًا وصفًا قصيرًا للمشروع، ودليلاً للبدء السريع، ومعلومات عن كيفية الإسهام، وقائمة بالمهام المطلوب إنجازها (TODO)، وروابط إلى توثيق إضافي. وتُكتب ملفات (README) هذه عادةً بلغة (markdown)، وهي لغة ترميز بسيطة تُعرض منسقة آليًا على (GitHub). وأخيرًا، فإن إضافة ملف (CITATION) إلى المستودع تُعلم مستخدميك بكيفية الاستشهاد بمشروعك ونسبة الفضل إليه.

ومن المشكلات الشائعة في البرمجيات أنها تتطور عادةً بسرعة مختلفة عن النصوص المنشورة في الإنتاج العلمي. بل من الشائع أن نجد برمجيات ذات سمات ووظائف جديدة لم تُوصف في الورقة الأصلية. ويتكامل (GitHub) الآن مع خدمات الأرشفة مثل (Zenodo) و(Figshare)، مما يتيح إسناد معرفات (DOI) إلى مستودعات الشفرة. والإجراء يسير نسبيًا، إذ لا يتطلب سوى توفير البيانات الوصفية وسلسلة من الخطوات الإدارية. وافترضًا، ينشئ (Zenodo)

أرشيفًا للمستودع في كل مرة يُنشأ فيها إصدار جديد في (GitHub)، مما يضمن أن تظل الشفرة المستشهد بها محدّثة. وبعد إسناد معرف (DOI)، يمكن إضافته إلى مصادر معلومات الإنتاج العلمي، مثل (Europe PubMed Central) [16].

وكما ذكر في المقدمة، ينبغي تمكين قابلية إعادة إنتاج الادعاءات العلمية بتوفير البرمجيات ومجموعات البيانات والعملية المفصّلة إلى نتائج قابلة للتفسير التي استُخدمت في دراسة بعينها. وينبغي للمنشورات، قدر الإمكان، أن تبرز أن الشفرة متاحة مجانًا، في (GitHub) مثلًا، إلى جانب أي مخرجات أخرى ذات صلة ربما أُودعت. وفي تجربتنا، يزيد هذا الانفتاح زيادة كبيرة من فرص قبول البحث للنشر. فمحررو الدوريات والمحكمون تتاح لهم فرصة إعادة إنتاج النتائج أثناء عملية مراجعة المخطوطة، مما يزيد الثقة بالنتائج المعروضة. وفضلاً عن ذلك، بعد نشر البحث، يمكن لأعضاء آخرين في المجتمع العلمي إعادة إنتاج عملك، مما قد يزيد الاستشهادات ويعزز فرص مزيد من النقاش والتعاون.

ولا يجعل توافر مستودع عام يحتوي على الشفرة المصدرية البرمجية مفتوحة المصدر في حد ذاته. فينبغي لك استخدام ترخيص معتمد من مبادرة المصدر المفتوح (Open Source Initiative, OSI) يحدد كيف يمكن استخدام البرمجية وتعديلها ومشاركتها بحرية. ويُفضّل استخدام التراخيص الشائعة مثل تلك المدرجة في. ولاحظ أن ملف (LICENSE) في المستودع ينبغي أن يكون ملفًا نصيًا عاديًا يتضمن نص ترخيص معتمد من مبادرة المصدر المفتوح، لا مجرد إحالة إلى الترخيص.

القاعدة 9: روج لمشروعاتك وناقشها: صفحة الويب وأكثر:

الطريقة التقليدية للترويج للبرمجيات العلمية هي نشر بحث مرتبط بها في الإنتاج العلمي المحكّم، وإن كان ذلك، كما أشار بكهاير (Buckheir) ودونوهو (Donoho)، مجرد دعاية [17]. ويمكن لخطوات إضافية أن تعزز ظهور المنظمة. فعلى سبيل المثال، صفحات (GitHub) (GitHub Pages) مواقع ويب بسيطة يستضيفها (GitHub) مجانًا. ويستطيع المستخدمون إنشاء واستضافة مواقع مدونات، وصفحات مساعدة، وأدلة استخدام، ودروس تعليمية، ومواقع ويب مرتبطة بمشروعات محددة. وتأتي (Pages) مع مولّد قوي للمواقع الثابتة يسمى (Jekyll)، يمكن دمجه مع أطر عمل أخرى مثل (Bootstrap) أو منصات مثل (Disqus) لدعم التعليقات وإدارتها.

وفضلاً عن ذلك، دُمجت منصات عديدة للتواصل الآن مع (GitHub)، مثل (Gitter) و(Slack). وتتيح نظم التواصل الآن لمجتمع المستخدمين والمطورين والمتعاونين في المشروع تبادل الأفكار والمسائل والإبلاغ عن الأخطاء أو الحصول على الدعم. فعلى سبيل المثال، (Gitter) أداة دردشة قائمة على (GitHub) تتيح للمطورين والمستخدمين مشاركة جوانب من عملهم. ويرث (Gitter) شبكة المجموعات الاجتماعية العاملة حول مستودعات (GitHub) ومنظّماته وبلاغاته. ويعتمد على الهويات داخل (GitHub)، فينشئ غرف دردشة شبيهة بدردشة الإنترنت المتناقلة (Internet Relay Chat, IRC) للمشروعات العامة والخاصة. وداخل دردشة (Gitter)، يستطيع الأعضاء الإحالة إلى البلاغات والتعليقات وطلبات السحب. ويدعم (GitHub) أيضًا صفحات الويكي (وهي نفسها مستودعات خاضعة للتحكم في الإصدارات) لكل مستودع، يستطيع فيها المستخدمون إنشاء صفحات للتوثيق أو الأمثلة أو الدعم العام وتحريرها.

ومن الخدمات المختلفة (Gist)، التي تمثل طريقة فريدة لمشاركة مقتطفات الشفرة، أو الملفات المفردة، أو أجزاء من الملفات، أو التطبيقات الكاملة. ويمكن إنشاء (Gists) بطريقتين مختلفتين: (Gists) عامة يمكن تصفحها والبحث فيها عبر (Discover)، و (Gists) سرية مخفية عن محركات البحث. ومن الميزات الرئيسة لـ (Gist) إمكانية تضمين مقتطفات الشفرة في تطبيقات أخرى، مما يتيح للمستخدمين تضمين (Gists) في أي حقل نصي يدعم (JavaScript).

القاعدة 10: استخدم (GitHub) لتكون اجتماعيًا: تابع وراقب:

كما يتابع الباحثون التطورات في مجالاتهم، يمكن للمبرمجين العلميين أن يتابعوا المشروعات المتاحة للعموم التي قد تفيد أبحاثهم. ويتيح (GitHub) هذه الوظيفة بمتابعة مستخدم (GitHub) الآخرين (انظر أيضًا القاعدة 2) أو مراقبة نشاط المشروعات، وهي ميزة شائعة في كثير من منصات التواصل الاجتماعي. فاستفد منها قدر الإمكان!

الاستنتاجات:

إذا كنت تعمل في البحث العلمي ولم تستخدم (Git) و (GitHub) من قبل، فإننا نوصيك باستكشاف إمكاناته في أقرب وقت ممكن. وكما هو الحال مع أدوات كثيرة، ثمة منحنى تعلم ينتظرك، لكن عددًا من الميزات الأساسية والقوية في آن معًا في متناول المبتدئ أيضًا، ويمكن تطبيقها على حالات استخدام كثيرة ومختلفة [6]. ونتوقع أن يكون العائد جديرًا بجهدك. وختامًا، نود أن نوصي ببعض الأمثلة على مستودعات المعلوماتية الحيوية في (GitHub) (الجدول 1)، وبعض المواد التدريبية المفيدة، ومنها ورش العمل والمقررات على الإنترنت والمخطوطات (الجدول 2).

المعلومات الداعمة:

الملف (S1). معلومات تكميلية تتضمن ثلاثة أقسام: تخزين الملفات الكبيرة في (Git) (LFS)، ومستويات اختبار الشفرة المصدرية والتكامل المستمر، وتوثيق الشفرة المصدرية.

(PDF)

المراجع:

1. Goodman A, Pepe A, Blocker AW, Borgman CL, Cranmer K, Crosas M, et al. Ten simple rules for the care and feeding of scientific data. PLoS Comput Biol. 2014;10(4):e1003542. pmid:24763340
2. Perez-Riverol Y, Alpi E, Wang R, Hermjakob H, Vizcaíno JA. Making proteomics data accessible and reusable: current state of proteomics databases and repositories. Proteomics. 2015;15(5-6):930-49. pmid:25158685
3. Osborne JM, Bernabeu MO, Bruna M, Calderhead B, Cooper J, Dalchau N, et al. Ten simple rules for effective computational research. PLoS Comput Biol. 2014;10(3):e1003506. pmid:24675742

4. Vihinen M. No more hidden solutions in bioinformatics. *Nature*. 2015;521(7552):261. pmid:25993922
5. Blischak J, Davenport E, Wilson G. A Quick Introduction to Version Control with Git and GitHub. *PLoS Comput Biol*. 2016;12(1):e1004668. pmid:26785377
6. Ram K. Git can facilitate greater reproducibility and increased transparency in science. *Source code for biology and medicine*. 2013;8(1):1.
7. Dabbish L, Stuart C, Tsay J, Herbsleb J. Social Coding in GitHub: Transparency and Collaboration in an Open Software Repository. In: *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work. CSCW '12*. New York, NY, USA: ACM; 2012. p. 1277-1286. Available from: <http://doi.acm.org/10.1145/2145204.2145396>.
8. R Core Team. R: A Language and Environment for Statistical Computing; 2016. Available from: <https://www.R-project.org/>.
9. Noble WS. A quick guide to organizing computational biology projects. *PLoS Comput Biol*. 2009;5(7):e1000424. pmid:19649301
10. Leprevost FV, Barbosa VC, Francisco EL, Perez-Riverol Y, Carvalho PC. On best practices in the development of bioinformatics software. *Front Genet*. 2014;5:199. pmid:25071829
11. Fowler M. Continuous Integration; 2006. <http://www.martinfowler.com/articles/continuousIntegration.html>.
12. Vizcaíno JA, Deutsch EW, Wang R, Csordas A, Reisinger F, Ríos D, et al. ProteomeXchange provides globally coordinated proteomics data submission and dissemination. *Nat Biotechnol*. 2014;32(3):223-6. pmid:24727771
13. Ahmadi A, Aiello-Lammens M, Ainsley J, Allen J, Alsheikh-Hussain A, Banaszkiewicz P, et al. *Software Carpentry: Programming with R*; 2015. <http://dx.doi.org/10.5281/zenodo.27353>.
14. Credit where credit is overdue. *Nat Biotechnol*. 2009;27(7):579. pmid:19587644
15. FORCE11 Software Citation Working Group. <https://www.force11.org/group/software-citation-working-group>.
16. Europe PMC Consortium. Europe PMC: a full-text literature database for the life sciences and platform for innovation. *Nucleic Acids Res*. 2015;43(Database issue):D1042-8. pmid: 25378340
17. Buckheit J, Donoho D. *WaveLab and Reproducible Research*. Springer-Verlag; 1995. p. 55-81.

18. Cock PJ, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*. 2009;25(11):1422-3. pmid:19304878
19. Goecks J, Nekrutenko A, Taylor J, Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol*. 2010;11(8):R86. pmid:20738864
20. Gatto L, Lilley KS. MSnbase-an R/Bioconductor package for isobaric tagged mass spectrometry data visualization, processing and quantitation. *Bioinformatics*. 2012;28(2):288-289. pmid:22113085
21. Sturm M, Bertsch A, Gröpl C, Hildebrandt A, Hussong R, Lange E, et al. OpenMS—an open-source software framework for mass spectrometry. *BMC bioinformatics*. 2008;9(1):163.
22. Perez-Riverol Y, Xu QW, Wang R, Uszkoreit J, Griss J, Sanchez A, et al. PRIDE Inspector Toolsuite: Moving Toward a Universal Visualization Tool for Proteomics Data Standard Formats and Quality Assessment of ProteomeXchange Datasets. *Molecular & Cellular Proteomics*. 2016;15(1):305-317.
23. Eglén SJ, Weeks M, Jessop M, Simonotto J, Jackson T, Sernagor E. A data repository and analysis framework for spontaneous neural activity recordings in developing retina. *Gigascience*. 2014;3(1):3. pmid:24666584
24. 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):2078-9. pmid:19505943