Supplementary material

- Supplementary Table 1: Metrics generation algorithms and weights applied to the FAIRsoftmetrics generation of the 2,000 tools.
- Supplementary Table 2: FAIRsoft metrics and weights.
- Supplementary Table 3: Main statistics of the 2,000 tools FAIRsoft scores when only bio.tools metadata was used and metadata from other sources was added.
- Supplementary Table 4: mean score of the high-level metrics.

Supplementary Table 1: **Metrics generation algorithms and weights applied to the *FAIRsoft* metrics generation of the 2,000 tools**. Some metrics weigth='NO', which means thay have not been measured. The column 'type it applies to' can be 'all', 'web' or 'non-web' and it makes reference to the type of software it applies too. Those metrics labeled 'all' apply to all types. Those labelled by 'web' only apply to web services, while those labelled 'non-web' apply to all the types that are not web services. Sometimes, a metric applies to all instances but have different weights depending on the type, so the type label appears as 'web/non-web' and the weight as 'w1/w2'. In these cases 'w1' and 'w2' are the weights assigned to 'web' and 'non-web' instances respectively. If an instance is a web service, only metrics labelled 'all' and 'web' will be applied to it, being the total weight 1. If an instance is not a web service, metrics labelled 'all' and 'non-web' will be applied to it, being the total weight 1.

Metric Identifier	Metric Name	Domain (*)	What is being measured?	Why should we measure it?	How do we measure it?	Weight (type it applies to / weight of each metric of type X.x / weight of the whole X)		
Findability								
F1	Identity uniqueness	canonical	Whether the software has a proper, unique and persistent identifier.	The uniqueness of an identifier is a necessary condition to unambiguously refer that resource, and that resource alone.				
F1.1	Uniqueness of name	canonical	Whether the software has a unique name to identify it.	The name is the commonly used as the main identifier of a software. Each tool should have a unique name to avoid ambiguities. Different versions of the same software should share a name, but if substantial modifications in the algorithm are done, the identifier should change for the new piece of software.	A name is valid.	all	0.8	0.4
F1.2	Identifiability of version	canonical	Whether there is a scheme to uniquely and properly identify the software version.	A version scheme is necessary to refer to an specific release of a software and keep track of the incrementally different versions of the software.	A version of the form X.X is considered valid.	all	0.2	
F2	Existence of Metadata	Depends on the overlap of F2.1 and F2.2 - they shouldn't be contradictory. Different instances may have different level of detail	Whether software is described with rich metadata, including scientific applicability.	Metadata makes finding through search engines and diciding if a tool is of interest possible.				
F2.1	Structured Metadata	instance	Metadata is adjusted to specific metdata formats	Specific formats are more machine readable, which increases its findability by search engines	At least a source of structured metadata is considered valid.	all	1	0.2
F2.2	Standarized Metadata	instance	Metadata is described using accepted ontologies	The same piece of information about a software can be stated in many equivalent forms. each tool being described with different terminology, with non specified meanings, makes metadata very hard to interpret. Automatic processing is also harder. When searching for a software with certain features, the lack of a consusuate terminology makes the process of searching slow and difficult.	NO	all	0	
F3	Searchability	Depends on the available information for F3.1, F3.2 and F3.3, we can provide specific information for versions and/or instances; or canonicalally	How software can be found	There are multitude of mechanisms for scientists looking to find specific software				
F3.1	Searchability in registries	canonical, versions and/or instances	Whether software is included in the main software registries.	Software registries are the main resource scientists use when seaching for software.	At least one software registry among the instance sources is considered valid.	all	1 whichever: 0.7	0.4
F3.2	Searchabiliy in software repositories	canonical, versions and/or instances	Wheter software ca be found in any of the major software repositories e.g. GitHub, GitLab, SourceForge,	Software repositories can be an additional resource used by scientists when looking for software	An associated software repository is considered valid.	all	2 whichever: 0.85	
F3.3	Searchability in literature	canonical, versions and/or instances	Whether software can be found in specialized literatue services e.g. EuropePMC, PubMed, Journals Site, bioArxiv	Specialized literature is a good reference to find software, especially to discover new software	At least one associated publication is considered valid.	all	all: 1	
Accessibility								
A1	Existance of downloadable, buildable or accesible working version	canonical, versions and/or instances	Whether it is possible to access/download/build a working version of the tool.	Being able to access the software as a user is the main aspect of Accessibility				
A1.1	Existance of API or web	instances	Whether it is possible to access a working version of the tool through and API or web.	Remote access to the tool requires only internet access from the user, increasing the usability of the resource.	An url is considered valid	web	0.6	0.7
A1.2	Existance of downloadable and buildable software working version	versions and/or instances	Whether it is possible to access/download/build a working version of the tool including the generation of a software container	Many users want to be able to install and run the software where and as they wish. Other times, it is imperative to install the resource loacally to be able to use it (modules, for instance). A downloadable and buildable version greatly increases the freedom of use and usability of software.	At least one download link is considered valid.	non-web	0.5	
A1.3	Existance of installation instructions	versions and/or instances	Whether there is a set of instructions and other necessary information the user can follow to build the software	A guideline to install the software might be absolutely necessary to successfully build a software. In all cases, it greatly increases the probability of successfull installation.	A link explicitly stated as installation instruction, instructions in web if Bioconductor package, availability through Galaxy ToolShed and an available manual are all considered valid.	non-web	0.2	
A1.4	Existence of test data	versions	Whether test data is available	Test data allows the user to make sure the program works as expected and serves as an example of woring data the user can look at when preaparing its own data.	At least one piece of test data is considered valid .	web/non-web	0.4/0.1	
A1.5	Existence of software source code	versions	Whether software source code is available	Source code can be compiled to work in any operatating system, and allows the user to solve installation and running issues. For intance, dependencies that may not be specified in the documentation may be clear from the source code. If the source code is available, the solution to these issues can even bea modification in the code.	A link explicitly stated as source code is considered valid.	non-web	0.2	
A2	Software history trackability	canonical	Whether there is available code and metadata even when the software is no longer in use	To match software provenance with analyzed data provenance				
A2.1	Metadata of previous versions at software repositories	versions	Whether there is available metadata of previous versions	Even if the code/funtionality is missing, the existance of metadata for a given version of a software, used in a published research the user is interested in, for instance, enables them to be certain about the past existance of a software and relevant details about it.	NO	all	0	0
A2.2	Existence of accesible previous versions of the software	versions	Whether there are available previous versions	To be able to reproduce analyses with the original software versions	NO	all	0	
A3	Restricted access	canonical, versions and/or instances						
A3.1	Registration compulsory	instances	Whether software can be used without registration	Registration is a barrier for accessibility, since users may not want to identify themselves.	NO	all	0	0.3
A3.2	Availability of version for free OS	versions	Whether the software can be used in a free operative system	Non-free operating systems are an economical barrier for software accessibility.	Linux among the compatible Operating Systems is considered valid.	non-web	0.25	
A3.3	Availability for several OS	versions	Whether there are versions of the software for several operative systems	Having to run a software in a different operatying system than the one used by a user in their research implies making a big effort. Sometimes, only one operating system is available in a research (if a cluster or supercomputer is	At least two operating systems are available is considered valid.	non-web	0.25	
A3.4	Availability on free e-Infrastructures	versions	Whether the software can be used in a free e-infrastructure	Everyting a user needs to run a software provided by a e-infrastructure is an internet browser, freely available for all operating systems, requiring no building or installation in the susers system.	A galaxy public server or vre link are considered valid	non-web	0.25	
A3.5	Availability on several e-Infrastructures	versions	Whether the software can be used in several e-infrastructure	Users usually use e-infrastrucures to build workflows. The e-infrastrucure they choose to use, thus, depends on several softwares been availale in the same platform and other factors. The more e-infrastructures a software is available in, the greater the likelyness a user will be able to use it.	At least two galaxy public server or vre links are considered valid	non-web	0.25	
Interoperability								
I1	Documentation on Input/output datatypes and formats	versions		Research software usually use data given by other programs as an input. In the same manner, its output will be process by another program. For a software to be interoperable, the transition of data from one program to another should be as smooth and less error prone as possible.				
I1.1	Usage of standard data formats	versions	Whether the input and output datatypes are formally specified and related to accepted ontologies	Data format transformations to meet software often unique input format requirements are a source of errors and take time and resources. The usage of standard formats by software minimizes the transformatios data is subjected to in the course of research. In addition, collective efforts can me made to create good quality tools for the management of data avoiding ad hoc scripting.	At least one input or output data format that is specified in the EDAM format ontology is considered valid.	all	0.5	0.6
I1.2	Usage of standard API framework	versions	Whether APIs (Rest, libraries) are documented in a standard framework (OpenAPI, CWL, ...)	The usage of standard frameworks greatly eases their usage	NO	web	0.3	
I1.3	Verificability of data formats	versions	Whether input/output data are specified using verifiable schemas (e.g. XDS, Json schema, ...)	Verifiability allows users to automatically make sure their data will be readable by other resources supporting that same format.	Standard formats (I1.1) as well as JSON are considered valid.	non-web	0.3	
I1.4	Flexibility of data format supported	versions	Whether the software allows to choose among various input/output data formats, or provide the necessary tools to convert other common formats into the supported ones.	Data format transformation to meet software often unique input format requirements are a source of errors and take time and resources. The capacity of a softwaere to support various formats avoids the need for data transformation by the user.	At least two input or output data formats are valid.	all	0.2	
I1.5	Generation of provenance information	versions	Whether the software provides provenance information according to accepted standards (PROV)	Ideally, FAIR software should produce FAIR Data, of which provenance is a very important part.	NO	all	0	
I2	Workflow compatibility	canonical	Whether software can be deployed in a format to be included in pipelines	Research software is usually used as part of a workflows.				
I2.1	Existance of API/library version	instances	Whether the software has API /library versions to be included in users' pipelines	It is common for users to want to use a piece of software as a part of a pipeline. In order to do so, the software must be accessible in the proper form, such as modules that can be loaded into other code or an API to connect.	Type being either library or API is considered valid	all	0.5	0.1
I2.2	E-infrastructure compatibility	instances	Whether the software can be deployed in e-infrastructures (e. g. Galaxy)	E-infraestructures are used by many to build pipelines and process their data. The impossibility to deploy a software in them prevents it from being used by these users.	Being one of the metadata sources the GalaxyShed is considered valid.	all	0.5	
I3	Dependencies availability	versions	Whether there is proper documentation on the software's dependencies as well as mechanisms to obtain them	Dependencies of a software are absolutely necessary to run a software.				
I3.1	Dependencies statement	instances	Whether the software includes details about dependencies	A dependencies statement allows the user to know what additional software they must install in order to get a software running	At least one dependency stated is valid	all	0.33	0.3
I3.2	Dependencies are provided	instances	Whether the software includes its dependencies or mechanisms to access them	Even if dependencies are stated, the proccess of downloading and installing them can be incredibly hard, except the dependencies are directly provided wih software or the mechanisms to obtain them are provided.	If Bioconductor, Bioconda or Galaxy among the sources is valid.	all	0.33	

I3.3	Availability through dependencies aware systems	instances	Whether the software is distributed via a dependencies aware system	Even if dependencies are stated, the proccess of downloading and installing them can be incredibly hard. Dependencies aware systems like 'bioconda' are the easiest and most straightforward way to obtain and install all the dependencies a software needs	If Bioconductor, Bioconda or Galaxy among the sources is valid. (Understimation, needs improvement)	all	0.33	
(Re)Usability								
R1	Existence of usage documentation	canonical, versions and/or instances	Whether software provides adequate usage documentation	Usage documentation such as tutorial, use guides, examples of user cases and ' how to' s allow users to make a proper and effective use of the software.				
R1.1	Existence of usage guides	instances	Whether software user guides are provided	Guides include the information a user must know in orderr to know how to make proper use of a software.	Any documentaion except news, license and terms of use is considered valid.	all	0.7	0.3
R1.2	Existence of usage examples	instances	Whether examples of use cases are provided	Examples provide a demostration of how to use a software and provide the specificity the guides lack. They complement the information in guides for undestanding the use and applicabilityu of a software.	NO	all	0.3	
R2	Existence of License	canonical, versions and/or instances	Whether a clear and accessible usage license is provided	The lack of License and specific Terms of use and installation can prevent researchers from industry to make use of a software				
R2.1	Existence of terms of use	instances	Whether Terms of Use are stated	Any condition for the usage of a software must be know prior to its usage.	Terms of use or license are considered valid	web	1	0.3
R2.2	Existence of conditions of use	instances	Whther conditions of installation and usage are stated	Any condition for the usage of a software must be know prior to its usage.	Conditions of use or license are considered valid	non-web	1	
R3	Existence of contribution policy	versions	Whether a contributors policy exists	The lack of a policy or guide stating how to contribute to a software prevents external developers willing to contribute to it.				
R3.1	Contributors policy specification	versions	Whether a document stating the contributors policy exists	The lack of a policy or guide stating how to contribute to a software prevents external developers willing to contribute to it.	NO	all	0	0.2
R3.2	Existence of credit	versions and instances	Whether credit for contributions is provided	The credits for contributions should be stated, as contributors hold the copyright.	Authors are considered valid	all	1	
R4	Provenance availability	canonical	Whether provenance is available	Provenace allows to undestand a software and the applications it could have had				
R4.1	Usage of version control	versions	Whether the software follows a version-control system	Version control ensures the changes in software are recorded and searchable.	A repository either in githu or sourceForge is considered valid	all	0.7	0.2
R4.2	Existence of release policy	versions	Whether the software follows a defined and documented release policy	Without a release policy, users cannot know the differences between versions of a software.	NO	all	0.2	
R4.3	Metadata of previous versions at software repositories	canonical	Whether there is available metadata of previous versions	Even if the code/funtionality is missing, the existance of metadata for a given version of a software, used in a published research the user is interested in, for instance, enables them to be certain about the past existance of a software and relevant details about it.	NO	all	0.1	

(*) Level at which the metric can be applied: canonical (canonical), version or instance.

Supplementary Table 2: *FAIRsoft* metrics and weights. The column 'type it applies to' can be 'all', 'web' or 'non-web' and it makes reference to the type of software it applies too. Those metrics labeled 'all' apply to all types. Those labelled by 'web' only apply to web services, while those labelled 'non-web' apply to all the types that are not web services. Sometimes, a metric applies to all instances but have different weights depending on the type, so the type label appears as 'web/non-web' and the weight as 'w1/w2'. In these cases 'w1' and 'w2' are the weights assigned to 'web' and 'non-web' instances respectively. If an instance is a web service, only metrics labelled 'all' and 'web' will be applied to it, being the total weight 1. If an instance is not a web service, only metrics labelled 'all' and 'non-web' will be applied to it, being the total weight 1.

Metric Identifier	Metric Name	Domain (*)	What is being measured?	Why should we measure it?	Weight (type it applies to / weight of each metric of type X.x / weight of the whole X)		
Findability							
F1	Identity uniqueness	canonical	Whether the software has a proper, unique and persistent identifier.	The uniqueness of an identifier is a necessary condition to unambiguously refer that resource, and that resource alone.			
F1.1	Uniqueness of name	canonical	Whether the software has a unique name to identify it.	The name is the commonly used as the main identifier of a software. Each tool should have a unique name to avoid ambiguities. Different versions of the same software should share a name, but if substantial modifications in the algorithm are done, the identifier should change for the new piece of software.	all	0.8	0.4
F1.2	Identifiability of version	canonical	Whether there is a scheme to uniquely and properly identify the software version.	A version scheme is necessary to refer to an specific release of a software and keep track of the incrementally different versions of the software.	all	0.2	
F2	Existence of Metadata	Depends on the overlap of F2.1 and F2.2 - they shouldn't be contradictory. Different instances may have different level of detail	Whether software is described with rich metadata, including scientific applicability.	Metadata makes finding through search engines and diciding if a tool is of interest possible.			
F2.1	Structured Metadata	Instance	Metadata is described using accepted ontologies	Specific formats are more machine readable, which increases its findability by search engines	all	0.8	0.2
F2.2	Standarized Metadata	instance	Metadata is adjusted to specific metadata formats	The same piece of information about a software can be stated in many equivalent forms. each tool being described with different terminology, with non specified meanings, makes metadata very hard to interpret. Automatic processing is also harder. When searching for a software with certain features, the lack of a consusuate terminology makes the process of searching slow and difficult.	all	0.2	
F3	Searchability	Depends on the available information for F3.1, F3.2 and F3. 3, we can provide specific information for versions and/or instances; or canonicalally	How software can be found	There are multitude of mechanisms for scientists looking to find specific software			
F3.1	Searchability in registries	canonical, versions and/or instances	Whether software is included in the main software registries.	Software registries are the main resource scientists use when seaching for software.	all	1 whichever: 0.7	0.4
F3.2	Searchabiliy in software repositories	canonical, versions and/or instances	Wheter software ca be found in any of the major software repositories e.g. GitHub, GitLab, SourceForge	Software repositories can be an additional resource used by scientists when looking for software	all	2 whichever: 0.85	
F3.3	Searchability in literature	canonical, versions and/or instances	Whether software can be found in specialized literatue services e.g. EuropePMC, PubMed, Journals Site, bioArxiv	Specialized literature is a good reference to find software, especially to discover new software	all	all: 1	
Accessibility							
A1	Existance of downloadable, buildable or accesible working version.	canonical, versions and/or instances	Whether it is possible to access/download/build a working version of the tool.	Being able to access the software as a user is the main aspect of Accessibility			
A1.1	Existance of API or web	instances	Whether it is possible to access a working version of the tool through and API or web.	Remote access to the tool requires only internet access from the user, increasing the usability of the resource.	web	0.6	0.6
A1.2	Existance of downloadable and buildable software working version	versions and/or instances	Whether it is possible to access/download/build a working version of the tool including the generation of a software container	Many users want to be able to install and run the software where and as they wish. Other times, it is imperative to install the resource loacally to be able to use it (modules, for instance). A downloadable and buildable version greatly increases the freedom of use and usability of software.	non-web	0.5	
A1.3	Existance of installation instructions	versions and/or instances	Whether there is a set of instructions and other necessary information the user can follow to build the software	A guideline to install the software might be absolutely necessary to successfully build a software. In all cases, it greatly increases the probability of successfull installation.	non-web	0.2	
A1.4	Existence of test data	versions	Whether test data is available	Test data allows the user to make sure the program works as expected and serves as an example of woring data the user can look at when	web/non-web	0.4/0.1	
A1.5	Existence of software source code	versions	Whether software source is available	Source code can be compiled to work in any operatating system, and allows the user to solve installation and running issues. For intance,	non-web	0.2	
A2	Software history trackability	canonical	Whether there is available code and metadata even when the software is no longer in use	To match software provenance with analyzed data provenance			
A2.1	Metadata of previous versions at software repositories	versions	Whether there is available metadata of previous versions	Even if the code/funtionality is missing, the existance of metadata for a given version of a software, used in a published research the user is interested in, for instance, enables them to be certain about the past existance of a software and relevant details about it.	all	0.66	0.15
A2.2	Existence of accesible previous versions of the software	versions	Whether there are available previous versions	To be able to reproduce analyses with the original software versions	all	0.33	
A3	Restricted access	canonical, versions and/or instances					
A3.1	Registration compulsory	instances	Whether software can be used without registration	Registration is a barrier for accessibility, since users may not want to identify themselves.	all	0.2	0.25
A3.2	Availability of version for free OS	versions	Whether the software can be used in a free operative system	Non-free operating systems are an economical barrier for software accessibility.	non-web	0.2	
A3.3	Availability for several OS	versions	Whether there are versions of the software for several operative systems	Having to run a software in a different operatying system than the one used by a user in their research implies making a big effort. Sometimes, only one operating system is available in a research (if a cluster or supercomputer is needed, for instance). The more supported	non-web	0.2	
A3.4	Availability on free e-Infrastructures	versions	Whether the software can be used in a free e-infrastructure	Everyting a user needs to run a software provided by a e-infrastructure is an internet browser, freely available for all operating systems, requiring no building or installation in the susers system.	non-web	0.2	
A3.5	Availability on several e-Infrastructures	versions	Whether the software can be used in several e-infrastructure	Users usually use e-infrastructures to build workflows. The e-infrastrucure they choose to use, thus, depends on several softwares been availale in the same platform and other factors. The more e-infrastructures a software is available in, the greater the likelyness a user will be able to use it.	non-web	0.2	
Interoperability							
I1	Documentation on Input/output datatypes and formats	versions		Research software usually use data given by other programs as an input. In the same manner, its output will be process by another program. For a software to be interoperable, the transition of data from one program to another should be as smooth and less error prone as possible.			
I1.1	Usage of standard data formats	versions	Whether the input and output datatypes are formally specified and related to accepted ontologies	Data format transformations to meet software often unique input format requirements are a source of errors and take time and resources. The usage of standard formats by software minimizes the transformatios data is subjected to in the course of research. In addition, collective efforts can me made to create good quality tools for the management of data avoiding ad hoc scripting.	all	0.5	0.6
I1.2	Usage of standard API framework	versions	Whether APIs (Rest, libraries) are documented in a standard framework (OpenAPI, CWL, ...)	The usage of standard frameworks greatly eases their usage	web	0.3	
I1.3	Verificability of data formats	versions	Whether input/output data are specified using verifiable schemas (e.g. XDS, Json schema, ...)	Verifiability allows users to automatically make sure their data will be readable by other resources supporting that same format.	non-web	0.3	

I1.4	Flexibility of data format supported	versions	Whether the software allows to choose among various input/output data formats, or provide the necessary tools to convert other common formats into the supported ones.	Data format transformation to meet software often unique input format requirements are a source of errors and take time and resources. The capacity of a softweare to support various formats avoids the need for data transformation by the user.	all	0.1	
I1.5	Generation of provenance information	versions	Whether the software provides provenance information according to accepted standards (PROV)	Ideally, FAIR software should produce FAIR Data, of which provenance is a very important part.	all	0.1	
I2	Workflow compatibility	canonical	Whether software can be deployed in a format to be included in pipelines	Research software is usually used as part of a workflows.			
I2.1	Existance of API/library version	instances	Whether the software has API /library versions to be included in users' pipelines	It is common for users to want to use a piece of software as a part of a pipeline. In order to do so, the software must be accessible in the proper form, such as modules that can be loaded into other code or an API to connect.	all	0.5	0.1
I2.2	E-infrastructure compatibility	instances	Whether the software can be deployed in e-infrastructures (e. g. Galaxy)	E-infraestructures are used by many to build pipelines and process their data. The impossibility to deploy a software in them prevents it from being used by these users.	all	0.5	
I3	Dependencies availability	versions	Whether there is proper documentation on the software's dependencies as well as mechanisms to obtain them	Dependencies of a software are absolutely necessary to run a software.			
I3.1	Dependencies statement	instances	Whether the software includes details about dependencies	A dependencies statement allows the user to know what additional software they must install in order to get a software running	all	0.33	0.3
I3.2	Dependencies are provided	instances	Whether the software includes its dependencies or mechanisms to access them	Even if dependencies are stated, the proccess of downloading and installing them can be incredibly hard, except the dependencies are directly provided with software or the mechanisms to obtain them are provided.	all	0.33	
I3.3	Availability through dependencies aware systems	instances	Whether the software is distributed via a dependencies aware system	Even if dependencies are stated, the proccess of downloading and installing them can be incredibly hard. Dependencies aware systems like 'bioconda' are the easiest and most straightforward way to obtain	all	0.33	
(Re)Usability							
R1	Existence of usage documentation	canonical, versions and/or instances	Whether software provides adequate usage documentation	Usage documentation such as tutorial, use guides, examples of user cases and ' how to' s allow users to make a proper and effective use of the software.			
R1.1	Existence of usage guides	instances	Whether software user guides are provided	Guides include the information a user must know in orderr to know how to make proper use of a software.	all	0.7	0.3
R1.2	Existence of usage examples	instances	Whether examples of use cases are provided	Examples provide a demostration of how to use a software and provide the specificity the guides lack. They complement the information in guides for undestanding the use	all	0.3	
R2	Existence of License	canonical, versions and/or instances	Whether a clear and accessible usage license is provided	The lack of License and specific Terms of use and installation can prevent researchers from industry to make use of a software			
R2.1	Existence of terms of use	instances	Whether Terms of Use are stated	Any condition for the usage of a software must be know prior to its usage.	web	1	0.3
R2.2	Existence of conditions of use	instances	Whther conditions of installation and usage are stated	Any condition for the usage of a software must be know prior to its usage.	non-web	1	
R3	Existence of contribution policy	versions	Whether a contributors policy exists	The lack of a policy or guide stating how to contribute to a software prevents external developers willing to contribute to it.			
R3.1	Contributors policy specification	versions	Whether a document stating the contributors policy exists	The lack of a policy or guide stating how to contribute to a software prevents external developers willing to contribute to it.	all	0.5	0.2
R3.2	Existence of credit	versions and instances	Whether credit for contributions is provided	The credits for contributions should be stated, as contributors hold the copyright.	all	0.5	
R4	Provenance availability	canonical	Whether provenance is available	Provenace allows to undestand a software and the applications it could have had			
R4.1	Usage of version control	versions	Whther the software follows a version-control system	Version control ensures the changes in software are recorded and searchable.	all	0.7	0.2
R4.2	Existence of release policy	versions	Whether the software follows a defined and documented release policy	Without a release policy, users cannot know the differences between versions of a software.	all	0.2	
R4.3	Metadata of previous versions at software repositories	canonical	Whether there is available metadata of previous versions	Even if the code/funtionality is missing, the existance of metadata for a given version of a software, used in a published research the user is interested in, for instance, enables them to be certain about the past existance of a software and relevant details about it.	all	0.1	

(*) Level at which the metric can be applied: canonical (canonical), version or instance.

Supplementary Table 3. Main statistics of the 2,000 tools *FAIRsoft* scores when only bio.tools metadata was used and metadata from other sources was added.

	Findability					
Condition	1st quart	Median	3th quart	Mean	Max	Min
Only bio.tools metadata	0.86	0.86	0.94	0.89	1.00	0.80
Enriched metadata	0.86	0.86	0.94	0.89	1.00	0.80
	Accessibility					
Condition	1st quart	Median	3th quart	Mean	Max	Min
Only bio.tools metadata	0.12	0.12	0.26	0.21	0.81	0.0
Enriched metadata	0.12	0.12	0.35	0.23	0.88	0.0
	Interoperability					
Condition	1st quart	Median	3th quart	Mean	Max	Min
Only bio.tools metadata	0.0	0.0	0.0	0.04	0.42	0.0
Enriched metadata	0.0	0.0	0.2	0.10	0.97	0.0
	(Re)usability					
Condition	1st quart	Median	3th quart	Mean	Max	Min
Only bio.tools metadata	0.2	0.20	0.5	0.27	1.0	0.0
Enriched metadata	0.2	0.20	0.5	0.27	1.0	0.0

Supplemetary Table 4: mean score of the high-level metrics of the 3393 instances.
Scores are normalized so the maximum score for each is 1.

Metric	Mean score
F1	0,91
F2	1,00
F3	0,83
A1	0,35
A2	0,00
A3	0,33
I1	0,06
I2	0,01
I3	0,25
R1	0,07
R2	0,56
R3	0,59
R4	0,03