https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

Bitfields encode what data knows about itself

Steffen Ehrmann[1, 2, 3*]

[1]Institute of Biology, Leipzig University, Leipzig, Germany
[2]German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, Leipzig, Germany
[3]resonantia research gUG (haftungsbeschränkt), Leipzig, Germany

Correspondence: Steffen Ehrmann (steffen.ehrmann@posteo.de)

Abstract. Every observation a scientist produces carries implicit computational context, from uncertainty estimates and distribution parameters to quality flags and algorithmic decisions. Yet, in most cases, none of this travels when data crosses workflow boundaries. What arrives is typically a mean value. Everything else remains trapped in supplementary materials or institutional memory. Bitfields offer a direct solution. They are compact binary encodings that pack observation-level computational context into standard integer arrays, travel inside existing data formats, and decode on the receiving end without external dependencies. This is demonstrated through three linked workflows spanning livestock density modeling, ecological carrying capacity assessment, and socioeconomic intervention planning, where each project decodes its predecessor's bitfield and enriches it with new information. Statistical distributions, uncertainty metrics, and processing decisions accumulate across project boundaries rather than fragmenting at each handoff. The bitfield R package and a community-driven protocol repository provide the practical tooling. Because encoding integrates directly into analytical code, adoption requires no separate documentation step. Computational context is captured as a byproduct of the analysis itself.

1 Introduction

Understanding the earth system requires integrating knowledge across scientific workflows, disciplines, and institutions (Pörtner et al., 2023). To this end, we have deployed sensor networks globally and in orbit, recording a vast array of signals (Bush et al., 2017; Cavender-Bares et al., 2022) and deriving an even greater number of metrics (Otto et al., 2015; Magagna et al., 2021). The resulting data underpins hypotheses about the interactions between ecological processes and human activities. Yet when data moves between workflows, the computational context that shaped it is routinely stripped away, leaving downstream analyses blind to the uncertainties, assumptions, and quality assessments that produced the values they inherit.

These connections become especially critical in computational modeling workflows, where coupling different model frameworks often requires manual data transfer between computing environments (Malik and Schaeffer, 2024; Dietrich et al., 2019; Hollmann et al., 2013). Whether in Integrated Assessment Models, Earth System Models, or ecological simulations, this limits the preservation of contextual information needed for comprehensive analysis. Generating actionable knowledge about complex earth system challenges depends on efficient information flow between research teams, modeling groups, and scientific communities, and thus on FAIR computational workflows (Goble et al., 2020).

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

While an exact definition of knowledge may be unattainable, it fundamentally involves subjectively interpreted information (Zins, 2007). Knowledge can be formalized into structured information by reducing its subjective nature. The foundations for such formalization trace back to Leibniz, who proposed representing simple ideas as unique symbols in an "Alphabet of human thought" (Leibniz, 1681), and to Frege, whose Begriffschrift (Frege, 1879) introduced predicate logic as a formal language for reasoning and mathematics. This line of thought persists in modern metadata management through the semantic triple, where a subject, predicate, and object (e.g., "Frege [S] read [P] Leibniz [O]") transform knowledge into structured, machine-readable information. The framework finds practical application in projects such as Wikidata (Vrandečić and Krötzsch, 2014), demonstrating the continued relevance of these foundational concepts.

The knowledge produced by the scientific enterprise is still predominantly shared through papers containing tables, statistical metrics, and natural language text rather than formalized knowledge structures (Barton et al., 2022). This approach emerged in an era of paper-based research and modest data volumes, creating an academic incentive structure that has failed to evolve alongside today's digital, data-intensive landscape. The consequences are tangible. When a researcher downloads, for instance, a cropland extent output (Potapov et al., 2022), what typically travels is a single raster of mean values, and in rare cases this is accompanied by a standard deviation layer. Distribution parameters, model agreement metrics, and quality flags that shaped the original analysis remain trapped in supplementary files, methods sections, or the producing team's institutional memory. This information loss compounds across workflow boundaries, creating blind spots precisely where synthesis is most needed.

Moreover, implementing sophisticated frameworks such as Dublin Core (Weibel, 1997), RDF, W3C PROV (Missier et al., 2013), or FAIR principles (Wilkinson et al., 2016) requires specialized expertise that many research teams cannot afford when faced with funding constraints and publication deadlines (Rüegg et al., 2014; Sarikhani and Wendelborn, 2018). Rather than solving metadata isolation, these well-intentioned frameworks have inadvertently amplified it by creating a two-tiered system. Well-funded projects with dedicated metadata specialists can navigate complex standards and achieve greater connectivity, while resource-constrained teams face even higher barriers to sharing methodological decisions, quality assessments, uncertainty metrics (Barton et al., 2022), and computational provenance (Gil et al., 2016). This disparity becomes self-reinforcing as institutional demands for metadata documentation strengthen. Interdisciplinary teams, already stretched between maintaining disciplinary credibility and pursuing novel integration, face mounting disincentives to document comprehensively. Academia effectively treats thorough documentation as an invisible burden rather than a valuable contribution, requiring significant up-front investment that yields little career advancement despite its immense value to the scientific community (Goring et al., 2014).

Improved metadata integration across scientific projects presents a significant opportunity to transition from isolated research units toward interconnected knowledge systems, unlocking tremendous value from currently siloed information (Ma et al., 2014; Barton et al., 2022; Pörtner et al., 2023). To address this, two specialized categories of metadata are defined that, to the best knowledge, have not been explicitly formalized before. *Meta-analytic data* encompasses quantitative information about analytical processes and outcomes, such as uncertainty estimates, quality assessments, confidence intervals, and distribution parameters. Its defining feature is that it can be transmitted between distinct research workflows to enable integrated statistical analysis. *Meta-algorithmic data* refers to procedural parameters and decision trees that document code logics, including model

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

configurations, processing thresholds, and methodological choices that can flow between workflows to enable algorithmic interoperability. The distinction lies not in these data types' existence, which have long been present in research, but in their formalization specifically for cross-workflow transmission and reuse. Unlike conventional summary statistics that become detached from their context once published, meta-analytic and meta-algorithmic data maintain their computational value when properly encoded, enabling formalized knowledge sharing across disciplinary boundaries (Edwards et al., 2011). Synthesis projects in particular often process data from numerous distinct sources, so their metadata must be recordable and reusable in an observation-specific or spatially explicit (per pixel) fashion. Current metadata and data provenance systems have not focused on this level of granularity (Ma et al., 2014; Sarikhani and Wendelborn, 2018).

The bit array or bitfield directly addresses efficient information flow between distinct workflows. A bitfield is a sequence of bits where each bit or group of bits encodes specific information in a highly compacted form. This approach has been successfully employed in remote sensing applications like the Moderate Resolution Imaging Spectroradiometer (MODIS) data products to store quality flags (Roy et al., 2002), but its potential for broader application in scientific workflows remains largely unexplored, likely due to the lack of data structures for storing such information and the complexity of processing and reusing it across project boundaries. In this paper, we will:

1. Explain the theoretical basis and present the `bitfield` R-package (Ehrmann, 2026a) to handle meta-analytic and meta-algorithmic data,
2. Illustrate which novel possibilities this framework presents for (meta) data exchange in scientific workflows, and
3. Discuss pathways for adoption and integration into existing scientific practices.

By encoding these kinds of metadata at a granular level and facilitating its transmission between distinct workflows, bitfields break down the boundaries that currently fragment scientific knowledge production by making computational context transmissible between independent workflows.

2 Methods

2.1 What are bitfields?

Any information in computer systems is stored as a sequence of bits, and if that is deliberately used as a data structure, we can call it bit array or bitfield. One can think of a bit as a switch representing off (0) and on (1) states. A sequence of two bits can store four states (00, 01, 10 and 11), and a sequence of n bits up to 2^n states. These bit sequences typically encode letters or words and integers or numbers, but could also map to specific information beyond that. We can, for example, capture the outcome of (simple) tests returning boolean values (yes/no), a small set of cases (integers along the cases) or even numeric values with a limited precision as bits. Several bits characterising various aspects of a dataset can be combined into a sequence of bits, which can then be transformed to integer representation that is easily used in any downstream application.

Bitfields are most prominently used in MODIS data products, where quality flags document a wide range of metadata that allow a user to assess the data quality and high-level provenance at a pixel level. Various tools exist to decode these quality

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

flags, with the `i.modis.qc` function in GRASS GIS, various built-in functions in python and the `luna` package (Hijmans and Ghosh, 2023) in R, as some of the more modern ones. However, these tools focus exclusively on *decoding* existing bitfields. None provide functionality for data producers to *encode* new bitfields accompanying their own workflows or models.

2.2 Bitfields as data structure

A byte is made up of eight bits and thus allows to store $2^8 = 256$ states, or unsigned integers from 0 to 255. These eight bits could, for instance, store the outcome of eight tests that result in a yes/no response, where the sequence 01001010 would encode the situation where tests two, five and seven resulted in 'yes' (encoded as 1) and all others in 'no' (encoded as 0) (Fig. 1A). To convert this sequence of bits to an integer, one would summarise the element wise multiplication between the bit sequence (X of length n, where each $X_i \in \{0,1\}$ with $i = 0,...,n-1$) and the vector $[2^{n-1}..2^0]$:

$$\sum_{i=0}^{n-1} X_i \cdot 2^{n-1-i} \tag{1}$$

For $X = 01001010_2$

$$0 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 64 + 8 + 2 = 74_{10} \tag{2}$$

or in scientific notation $7.4 \cdot 10^1$. An important side note here is that bit values can also have a radix point. To convert those, the sequence of exponents is continued into the negative integers. For $X = 1.1010_2$

$$1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} + 0 \cdot 2^{-4} = 1 + 0.5 + 0.125 = 1.625_{10} \tag{3}$$

Apart from the above-mentioned eight tests, there are, however, various other situations that could be encoded by this bitfield. For example, the first three positions could encode integer values from 0 to 7 representing one of eight cases ($2^3 = 8$), and where the stored value here is case three ($010_2 = 2_{10}$), while the remaining five positions are yes/no outcomes, as before (Fig. 1B). This means, we need additional information about what exactly is encoded by the bitfield, a *registry* documenting how to interpret the bits is thus an important tool to allow re-use of the recorded bitfields.

2.3 Handling floating-point numbers

Real numbers cannot be stored directly in binary systems. Instead, they are encoded using three components (Fig. 1C), a *sign bit* (positive/negative), *significand bits* (the digits), and *exponent bits* (radix point position). The "floating" position of this radix point gives these numbers their name.

A "bias" value ($2^{n-1} - 1$ for n exponent bits) that is subtracted from the exponent enables encoding of both very small and very large numbers using unsigned integers. With 3 exponent bits, the bias of 3 allows exponents from -3 to $+4$, representing values from 0.001 to 10000.

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

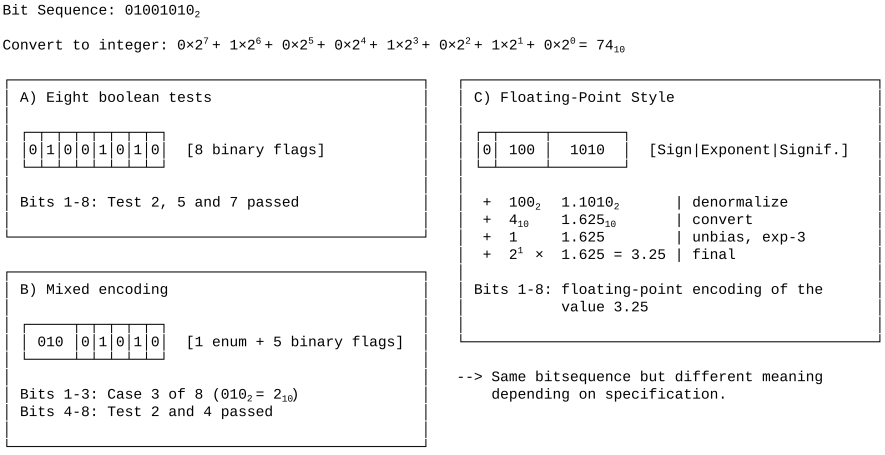

Figure 1. Various possible bitfield meanings are determined by the respective registry. The registry documents whether the bit sequence encodes (A) single binary flags, (B) a mix of different encodings such as enumerations (from 0 through $N_{cases-1}$) and binary flags or even (C) floating-point values. Panel (C) shows the process of decoding a floating-point bit sequence. Denormalization refers to the action of adding a 1 in front of the significand. This is a technique commonly used in floating-point encoding. It can be used, because in binary notation, values can only be expressed either with a 0 or a 1. If the value is not 1, it will have to be 0, and since the bit sequence is read from right to left, any 0 at the left end doesn't change the encoded value, and can thus be ignored. Based on this assumption, the 1 that will always be at the left end of the bit sequence can thus be omitted and added when decoding, save a bit position for more precision or range.

This creates a fundamental trade-off with limited bit count. More exponent bits extend the range, while more significand bits increase precision (Fig. 2). The IEEE 754 standard (IEEE Computer Society, 2008) specifies fixed allocations (e.g., so-called single-precision with 1/8/23 bits). However, bitfields can use flexible combinations tailored to specific applications, reducing storage requirements while maintaining adequate precision for the data at hand.

2.4 Package description

To describe the functionality of the `bitfield` R-package, consider first the following example data that contains various issues. It has invalid (`259`) or improbable (`0`) coordinate values, special values (`NA`, `NaN` or `Inf`), mislabelled values (`honey`) and variable flags (`*r`).

```
library(bitfield)
bf_tbl
#> # A tibble: 9 x 5
#>       x     y commodity yield year
#>   <dbl> <dbl> <fct>     <dbl> <chr>
#> 1  25.3  59.5 soybean   11.2  2021
#> 2  27.9  58.1 maize     12.0  <NA>
#> 3  27.8  57.8 soybean   13.2  2021r
#> 4  27    59.2 <NA>       4.43 2021
```

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

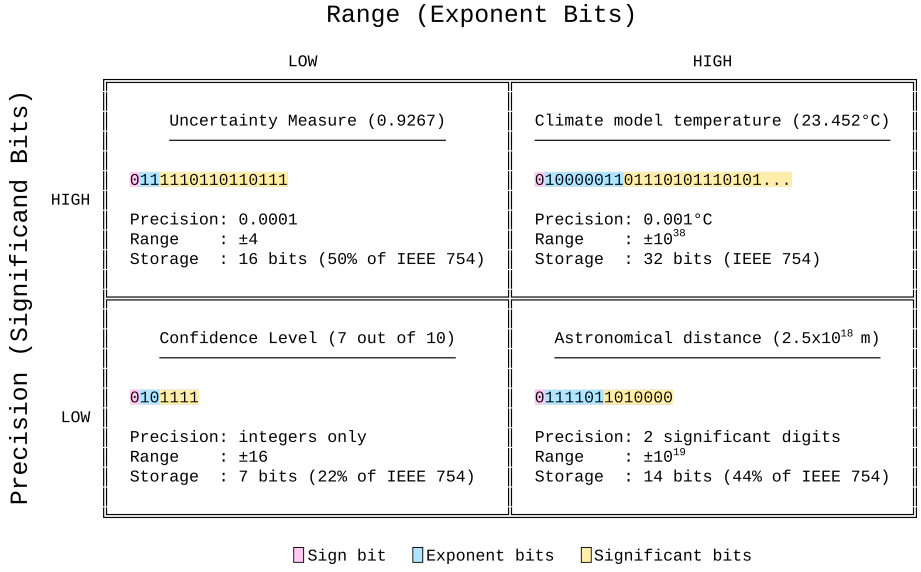

Figure 2. Precision-Range trade-off matrix for floating-point values. Values can have low and high range and low or high precision. Depending on the use case, it is thus possible to specify an encoding that is maximally flexible and minimal in size.

```
#> 5 259   Inf   honey     13.0  2021
#> 6  27.3  59.1 maize      8.55 2021
#> 7  26.1  58.4 soybean   11.3  2021
#> 8  26.5 NaN   maize     10.6  2021
#> 9   0     0   soybean    9.01 2021
```

Understanding the logic of the `bitfield` R-package is straightforward. The registry of a bitfield needs to be set up with `bf_registry()` and can then be grown along a workflow because the functions are designed in a modular way. The function `bf_map()` applies so-called bit-flag protocols to the data, successively building a registry. A bit-flag protocol is a standardized instruction to test for a particular attribute, such as the existence of NA-values or the number of characters of variable values. The data resulting from a test are called flags and can take several forms. *Boolean* values (0 and 1), *enumerations* (categories treated as integers from 0 through $N_{cases-1}$, taking up $log_2 N_{cases}$ bits), *(signed) integers*, or *numeric* values in floating-point logic taking up as many bits as required by range and precision. The function `bf_analyze()` helps setting up the correct bit combination for the encodings (*Appendix A*). After defining all mappings, the function `bf_flag()` can be used to compute and check the flag values by applying the protocol to the data. This function is internally called by `bf_encode()`, which uses the registry to transform the mappings to flags, those to binary representation, and then chains them into a sequence (called *bitfield*) that is transformed to an integer value.

```
reg <- bf_registry(
    name = "yield_QA",
    description = "quality assessment in a table of yield data.",
```

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026


```
    template = bf_tbl)

# tests for longitude availability
reg <- bf_map(
    protocol = "na",                                # the protocol to build the bit flag
    data = bf_tbl,                                  # specify where to determine flags
    x = y,                                          # ... and which variable to test
    registry = reg)                                 # provide the registry to update

# test which case an observation is part of
reg <- bf_map(protocol = "case", data = bf_tbl, registry = reg, na.val = 4,
              yield >= 11, yield < 11 & yield > 9, yield < 9 & commodity == "maize")

# test the length (number of characters) of values
reg <- bf_map(protocol = "nChar", data = bf_tbl, registry = reg, x = y)

# store a simplified (e.g. rounded) numeric value
reg <- bf_map(protocol = "numeric", data = bf_tbl, registry = reg, x = yield, format = "half")

# doublecheck the case-flags
bf_flag(registry = reg, flag = names(reg@flags)[2])
#> [1] 1 1 1 4 1 3 1 2 2

# finally, encode the flags as integer values
(field <- bf_encode(registry = reg))
#> # A tibble: 9 x 1
#>    bf_int1
#>      <int>
#> 1  1591704
#> 2  1591806
#> 3  1591965
#> 4  4736110
#> 5  1460863
#> 6  3688518
#> 7  1591715
#> 8 10897746
#> 9  2246785
```

Whenever the information contained in the bitfield-integer shall be used in a downstream application, the function `bf_decode()` unpacks and transforms it back into the original values. This returns a named list (or multi-layer raster for spatial data) containing all decoded values, which can be re-used algorithmically. This also means that a once finalised bitfield, or parts thereof, can easily be reused and extended in an entirely independent downstream application.

```
decoded <- bf_decode(x = field, registry = reg, verbose = FALSE)
```

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

```
# access underlying values
names(decoded)
#> [1] "na_y"                 "case_yield-commodity" "nChar_y"
#> [4] "numeric_yield"

decoded$numeric_yield
#> [1] 11.187500 11.984375 13.226562  4.429688 12.992188  8.546875 11.273438
#> [8] 10.640625  9.007812
```

One of the more interesting use cases is in applying this to raster data. This can be done intuitively by simply supplying a raster object in the place of a table. The code logic automatically recognizes the data type and handles it internally. The user merely needs to handle saving the information in the respective format.

```
library(terra)

# define an example dataset
bf_rst <- rast(nrows = 3, ncols = 3, vals = bf_tbl$commodity, names = "commodity")
bf_rst$yield <- rast(nrows = 3, ncols = 3, vals = bf_tbl$yield)

# build the registry
reg <- bf_registry(name = "reg_raster", description = "bitfield for rasters.", template = bf_rst)

reg <- bf_map(protocol = "na", data = bf_rst, registry = reg, x = commodity)
reg <- bf_map(protocol = "range", data = bf_rst, registry = reg, x = yield, min = 5, max = 11)
reg <- bf_map(protocol = "category", data = bf_rst, registry = reg, x = commodity, na.val = 0)

# encode as bitfield (returns a SpatRaster for raster templates)
field <- bf_encode(registry = reg)

# to store the bitfield, make use of terra-native functions for maximum flexibility
# writeRaster(x = field, filename = "modelBitfield.tif",
#             datatype = "INT1U",                    # determines how many bytes are used
#             gdal = c("..."),                       # additional GDAL storage options
#             ...)

# decode (gridded) bitfield somewhere downstream (returns multi-layer SpatRaster)
decoded <- bf_decode(x = field, registry = reg)
#> # A tibble: 3 x 3
#>     pos name               desc
#>   <int> <chr>              <chr>
#> 1     1 na_commodity       'commodity' contains NA-values.
#> 2     2 range_yield        The 'yield' values are between '5' and '11'.
#> 3     3 category_commodity 'commodity' is encoded as categorical enumeration.
```

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

To make full use of datacite compliant export functionality, author and project metadata have to be provided. The bitfield package comes with the `project()` function that has been designed as an imitation of the native `person()` function, both of which allow to specify a range of typical metadata. Those are stored in the registry and can be exported with the `bf_export()` function. Available formats currently are `xml`, `json`, `yaml` and the R native `rds` format. The registry itself has currently no export function as the bitfield package only exists for R currently, and would be stored, for example, with `saveRDS()`.

```
author <- person(person1, role = c("cre", "aut"))

project <- bitfield::project(
    title = "something with yield",
    people = c(person(person1, role = "aut"),
               person(person2, role = c("aut", "cre"))),
    publisher = "example publisher",
    type = "Dataset",
    identifier = "10.5281/zenodo.1234567",
    description = "description of this project.",
    subject = c("crops", "yield"),
    license = "MIT")

reg <- bf_registry(
    name = "yield_registry",
    description = "the registry to my modelling pipeline",
    author = author,
    project = project)

# build registry and bitfield
reg <- bf_map(...)

# eventually, after storing the bitfield object, export the metadata, for example as xml file
bf_export(registry = reg, format = "xml", file = "modelMetadata.xml")
```

3 Discussion

Bitfields encode observation-level computational context into standard integer arrays and make it transmissible across workflow boundaries. This addresses a level of granularity that existing metadata frameworks have not targeted: not the dataset, not the file, but the individual observation or pixel. The three-project meta workflow demonstrates the practical consequence. Statistical distributions, uncertainty metrics, and processing decisions accumulate across independent research teams rather than fragmenting at each handoff.

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

3.1 A Journey Through Connected Workflows

To illustrate the full potential of bitfields, consider how they enable the seamless flow of information across three distinct virtual research projects. The following meta workflow demonstrates how sophisticated metadata can grow and interact across workflows, creating an emergent knowledge base greater than any single project could achieve alone. Figure 3 provides a visual summary of this meta workflow.

The complete bit allocation specifications for all three project implementations, including detailed parameter descriptions, bit counts, value encodings, and technical notes are provided in Appendix A. These specifications demonstrate how theoretical bitfield concepts translate to practical encoding decisions in real-world research applications. The project methods and outputs have been simulated in reproducible scripts that can be found at the public repository https://github.com/bitfloat/examples (Ehrmann, 2026b). The methods used there should not be re-used as-is, since they were designed to generate illustrative patterns rather than represent proper parameter spaces, despite attempting to approximate realistic workflows.

3.1.1 Earth Observation & Modeling

The meta workflow begins with a research team focused on livestock density modeling (Table A1). Traditional data sharing would reduce their complex model outputs to a single mean density value per pixel, discarding uncertainty information accumulated through months of modeling work. Instead, they design a compact bitfield encoding that preserves a complete statistical fingerprint, including not just median density and standard deviation but also the higher moments (skewness, kurtosis) and distribution type needed to reconstruct full probability distributions downstream. The encoding uses integer representation for bounded quantities like density and confidence, and floating-point encoding for open-ended variables like standard deviation (see Appendix B for the rationale behind this choice). (github.com/bitfloat/examples/project_1.R)

The idea is straightforward. Ecological thresholds are often crossed not by average conditions but by extremes. A pixel with moderate mean livestock density but high variance and positive skew presents different management challenges than one with the same mean but low variance. By encoding confidence levels and uncertainty sources alongside model selection and agreement metrics, they create outputs that communicate not just "what we found" but "how confident we are and why."

3.1.2 Ecological Overshoot Analyses

A second team downloads this bitfield to assess sustainable carrying capacity (Table A2). The key insight is what becomes possible. They can reconstruct full pixel-wise probability distributions without replicating the computationally intensive modeling work. This enables risk assessment based on tail probabilities rather than simple comparisons of mean values.

Their 24-bit encoding captures carrying capacity, exceedance risk and magnitude, and resource limitations (type and severity). The preserved statistical fingerprint from upstream flows directly into their calculations, creating integrated knowledge where ecological insights accumulate rather than fragment across project boundaries. (github.com/bitfloat/examples/project_2.R)

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

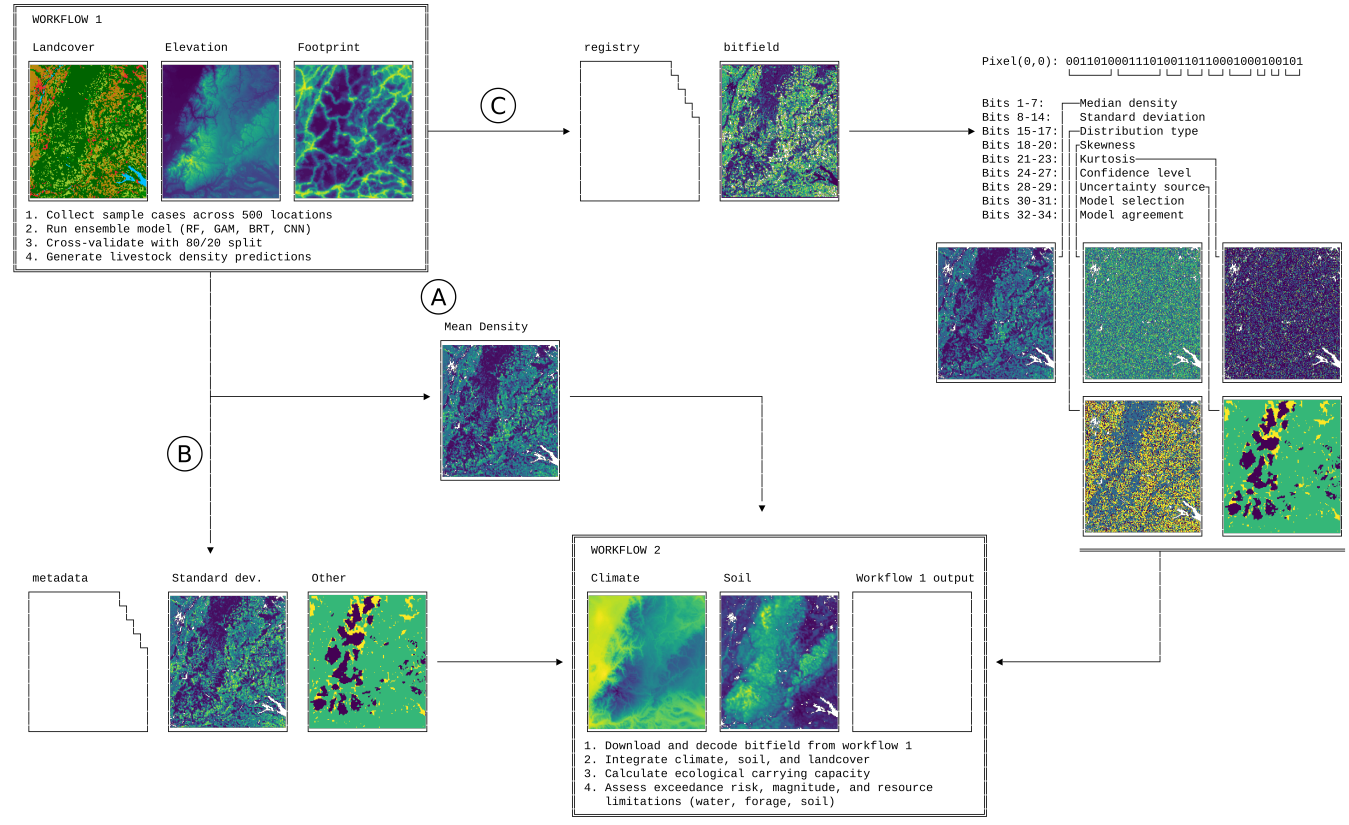

Figure 3. Simplified bitfield workflow. (A) Often, only a single data product is provided as research output that can be used in downstream applications. (B) Increasingly, also metadata, standard deviation and other (quality) layers are shared. However, this is often impractical because it leads to enormous data amounts and cumbersome handling. (C) In the bitfield approach, only a registry and the bitfield are shared (possible together with the 'main' average output). This registry allows decoding of the bitfield so that possibly a vast amount of data can be transmitted in a single integer layer, enhancing downstream applications.

3.1.3 Socioeconomic Analysis & Intervention Planning

The final team leverages both previous bitfields for food security planning (Table A3). They identify regions where high production variance coincides with carrying capacity exceedance and specific resource limitations, patterns that remain invisible when these variables exist in separate datasets.

Their encoding addresses a fundamental challenge, namely that economic signals often diverge from ecological realities. They quantify this through market distortion metrics, GDP adjustments for hidden ecological costs, and an Economic-Ecological Misalignment Index that combines pressure, constraints, and market failures. Crucially, they add temporal context through system trajectory classifications (stable, degrading, approaching threshold), transforming static assessments into dynamic ones.

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

The practical output is intervention prioritization, identifying where limited resources would yield maximum impact given not just current state but trajectory, and where interventions could simultaneously benefit multiple sectors (water, ecosystems, economy, climate). This final bitfield functions as a decision support system built on the accumulated knowledge of three independent research teams. (github.com/bitfloat/examples/project_3.R)

At the end of this chain, a single integer layer and its registry contain enough information for a policy analyst to identify regions where livestock density variance is high, the probability distribution is right-skewed, carrying capacity is exceeded with high confidence, water is the limiting resource, and the system is degrading, without access to any of the original models or their authors. This query requires precisely the observation-level context that conventional data products discard.

3.2 Multi-variable Encoding and Emergent Patterns

The meta workflow demonstrates bitfields accumulating information across workflows, but an equally powerful application lies in encoding multiple variables within a single observation. The ESA CCI Land Cover product already recognizes this need through its mosaic categories, single classes like "Tree cover (>15%) with shrub and herbaceous cover" that encode combinations of land cover types. However, these remain discrete categories with fixed thresholds that cannot capture continuous proportions or be flexibly recombined for different analytical needs.

Bitfields could encode the same compositional information as continuous variables, including proportional tree cover, shrub cover, herbaceous cover, water presence, and impervious surface, all within a single integer per pixel. This enables something discrete classification cannot, namely the detection of emergent patterns that exist only in the relationships between variables. A pixel with 40% vegetation, 30% water, and 30% built-up tells a fundamentally different story than three separate maps showing moderate values. When decoded together, spatial patterns emerge that reveal landscape configurations invisible in any single-variable map, such as wetland-urban interfaces, agricultural mosaics, or transitional zones that defy the boundaries of predefined mosaic categories.

The approach extends naturally across multiple dimensions. The thematic dimension captures composition (what exists where), the temporal dimension records trajectories and rates of change, and the causal dimension documents intervention-outcome relationships. The system trajectory classifications in the socio-economic bitfield exemplify how temporal dynamics can travel alongside static assessments.

This integrated encoding proves particularly valuable for bridging natural and social science data (Otto et al., 2015) and tracing how specific parameter combinations influence outcomes (*cf* Runge et al., 2019). Rather than treating variables as independent layers to be combined post-hoc, bitfields preserve their joint identity throughout analytical pipelines. Pattern recognition thus happens at the observation level, not as a post-hoc spatial analysis step.

3.3 Integration and Adoption

The bitfield approach complements rather than replaces established metadata systems (Gil et al., 2016). The `bf_map()` function automatically records W3C PROV-aligned (Missier et al., 2013) provenance relationships (data source, algorithm, researcher) without requiring additional documentation effort, addressing a key adoption barrier (Rüegg et al., 2014; Sarikhani

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

and Wendelborn, 2018). The `bf_export()` function generates DataCite-compatible metadata (DataCite Metadata Working Group, 2024) in multiple formats (JSON, XML, YAML), creating a multi-level architecture that preserves context at the workflow, bitflag, and observation levels.

Beyond interoperability with existing standards, the `bitfield` package enables community-driven standardization via a shared GitHub repository, accessible through the `bf_standards()` function (https://github.com/bitfloat/standards) (Ehrmann, 2026c). This repository acts as both archive and collaborative platform where encoding schemes evolve based on practical utility (Howison and Crowston, 2014). Researchers can immediately import existing encodings or contribute new ones, potentially increasing citation and recognition (Piwowar and Vision, 2013). Built-in versioning ensures reproducibility while allowing encodings to evolve. Unlike traditional standards requiring extensive committee work, this approach enables immediate utility while building consensus through demonstrated value (Rüegg et al., 2014; Stodden et al., 2018; Ram, 2013), aligning with calls for flexible, domain-specific metadata solutions that acknowledge researcher resource constraints (Barton et al., 2022).

The cumulative effect is that metadata can accumulate and enrich across workflows rather than remaining trapped in disconnected research silos. What makes this knowledge integration remarkable is that it emerges not from centralized coordination, but from standardized encoding and sharing of rich contextual information across independent research teams working in different disciplines (Edwards et al., 2011). The result is a compressed record of the computational decisions, transformations, and quality assessments that shaped each observation (Gil et al., 2016), and this record travels with the data itself rather than requiring separate infrastructure to maintain or query (Voinov and Shugart, 2013).

This works in practice because the approach integrates with existing scientific constraints rather than demanding their transformation. By embedding metadata capture directly into analytical workflows rather than requiring separate documentation steps, bitfields reduce the invisible burden that traditional metadata approaches impose (Goring et al., 2014). The protocol-based system substantially lowers the expertise threshold: where traditional approaches demand specialized knowledge of complex frameworks like Dublin Core or W3C PROV, bitfields use familiar function calls that integrate naturally into existing analytical code. The approach also benefits from network effects, as each additional workflow using compatible encodings increases the potential analytical combinations available to all participants. Well-documented datasets become more attractive for synthesis work, reducing the context reconstruction burden that typically limits large-scale comparative studies, and increased citation rates (Piwowar and Vision, 2013) and collaboration opportunities transform comprehensive documentation from an altruistic contribution into a strategic investment in professional advancement. The modular architecture further supports incremental adoption, allowing research teams to begin with simple flag protocols before progressing to sophisticated encodings (Edwards et al., 2011; Goble et al., 2020).

3.4 Future Directions

Several extensions of the bitfield concept merit exploration. First, when values are extracted from gridded data, the spatial neighborhood is lost. A pixel classified as “forest” could sit in the core of a large contiguous patch or at the edge of a small fragment, and that distinction determines ecological function at least as much as the land cover class itself (Tobler, 1970). Approaches such as Morphological Spatial Pattern Analysis (Vogt and Riitters, 2017) derive neighborhood-dependent classi-

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

fications (core habitat, edge, bridge, branch), but their outputs face the same extraction problem as any other raster-derived value. Encoding spatial summary statistics, such as local variability, gradient direction, autocorrelation, or structural role, into bitfields would allow neighborhood context to travel with extracted data rather than being discarded at the point of extraction.

Second, most computational pipelines traverse a finite set of decision nodes, each with a small number of realistic options. A bitfield could encode the full path through this decision graph as a sequence of small enumerations, creating a compact analytical provenance record per observation. Complementary to such designed encodings, learned representations from dimensionality reduction or other machine learning methods could be quantized into dedicated bit ranges, with the producing algorithm documenting each dimension's semantic content in the registry. This would create a hybrid of designed and discovered metadata within a single encoding.

These directions share a common theme, extending bitfields from encoding static observation attributes toward capturing relational, operational, and latent context that currently exists only implicitly in the data's structure.

Third, the current encoding system offers two precision profiles, uniform (integer encoding) and logarithmic (floating-point encoding). In practice, some variables may benefit from non-uniform precision that concentrates resolution in a specific value range rather than near zero. While the floating-point bias parameter shifts the representable window, it does not change the fundamental logarithmic precision structure. A piecewise encoding where the value range is split into segments, each with its own encoding parameters, could provide arbitrary precision profiles but adds design complexity. Whether real-world use cases demand this flexibility, or whether the integer/float dichotomy covers most needs sufficiently, is an open question that community experience with the framework will help answer (see Appendix B for a detailed discussion of encoding trade-offs).

4 Conclusions

The bitfield approach represents more than a technical advance in metadata management. It embodies a shift in how we conceptualize scientific information exchange. Like Leibniz's "Alphabet of human thought", bitfields offer a fundamental symbolic system for expressing complex concepts through the combination of simple, atomic elements. By contributing to this shared vocabulary, scientists can transmit rich contextual information across workflow boundaries using a formalized language that machines efficiently process and that preserves the semantics humans need for interpretation.

Unlike comprehensive metadata frameworks that require significant additional effort, bitfields can be incrementally adopted within existing analytical workflows, creating immediate value while laying groundwork for more integrated knowledge systems.

Effective scientific synthesis depends on communication between research teams, workflows, and institutions. Bitfields lower the barriers to this communication by embedding metadata capture into the analytical process itself, so that computational context accumulates across project boundaries rather than fragmenting at each one. The approach works within existing resource constraints and incentive structures, requiring neither dedicated metadata specialists nor adoption of external standards. Whether bitfields can shift the default from discarding computational context to preserving it will depend on community adoption, but the infrastructure to do so is now in place.

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

Code availability. https://github.com/bitfloat/bitfield, https://github.com/bitfloat/examples, https://github.com/bitfloat/standards

Appendix A: Detailed Bitfield Implementation Examples

When encoding any values it may at first not be obvious which encoding parameters to use. For this purpose, the `bf_analyze()` function helps find the most suitable parameters and to explore trade-offs between range, coverage and precision in floating-point encodings. For example, analyzing standard deviation values (range 0-15, requiring 1 decimal place precision).

```
library(bitfield)
set.seed(42)
x <- runif(1000, 0.1, 10)
bf_analyze(x, range = c(0, 15), decimals = 1)
```

```
## Float Analysis
## ==============
##
##   Observations      1000
##   NA values         0
##   Range             [0.102365, 9.98506]
##   Levels            -
##   Sign required     no
##   Bits required     select from the table below
##   Suggested na.val  automatic
##   Target range      [0.102365, 15]
##   Decimals          1
##
## Exp  Sig  Total  Underflow  Overflow  Changed     Min Res     Max Res        RMSE     Max Err
## ---  ---  -----  ---------  --------  -------  ----------  ----------  ----------  ----------
##   2    1      3      4.2%    19.8%    95.8%      0.2500      2.0000      2.3762      6.0000
##   2    2      4      4.2%    19.8%    91.9%      0.1250      1.0000      1.4195      4.0000
##   3    1      4      0.7%     0.0%    94.0%      0.0625      4.0000      0.9339      2.0000
##   2    3      5      4.2%    19.8%    84.5%      0.0625      0.5000      0.9514      3.0000
##   3    2      5      0.7%     0.0%    89.6%      0.0312      2.0000      0.6466      2.0000
##   4    1      5      0.0%     0.0%    93.3%      0.0312      4.0000      0.9338      2.0000
##   2    4      6      4.2%    19.8%    70.6%      0.0312      0.2500      0.7265      2.5000
##   3    3      6      0.7%     0.0%    80.7%      0.0156      1.0000      0.3072      1.0000
##   4    2      6      0.0%     0.0%    88.9%      0.0156      2.0000      0.6465      2.0000
##   2    5      7      4.2%    19.8%    56.1%      0.0156      0.1250      0.5992      2.2000
##   3    4      7      0.7%     0.0%    65.3%      0.0078      0.5000      0.1626      0.5000
##   4    3      7      0.0%     0.0%    80.0%      0.0078      1.0000      0.3071      1.0000
##   2    6      8      4.2%    19.8%    40.7%      0.0078      0.0625      0.5578      2.1000
##   3    5      8      0.7%     0.0%    48.7%      0.0039      0.2500      0.0895      0.3000
##   4    4      8      0.0%     0.0%    64.6%      0.0039      0.5000      0.1624      0.5000
##   3    6      9      0.7%     0.0%    29.2%      0.0020      0.1250      0.0559      0.2000
```

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026


```
##   4    5      9      0.0%     0.0%    48.0%      0.0020      0.2500      0.0891      0.3000
##   3    7     10      0.7%     0.0%    16.5%   9.77e-04      0.0625      0.0406      0.1000
##   4    6     10      0.0%     0.0%    28.5%   9.77e-04      0.1250      0.0553      0.2000
##   3    8     11      0.7%     0.0%     9.0%   4.88e-04      0.0312      0.0300      0.1000
##   4    7     11      0.0%     0.0%    15.8%   4.88e-04      0.0625      0.0397      0.1000
##   3    9     12      0.7%     0.0%     5.1%   2.44e-04      0.0156      0.0226      0.1000
##   4    8     12      0.0%     0.0%     8.3%   2.44e-04      0.0312      0.0288      0.1000
##   3   10     13      0.7%     0.0%     3.0%   1.22e-04      0.0078      0.0173      0.1000
##   4    9     13      0.0%     0.0%     4.4%   1.22e-04      0.0156      0.0210      0.1000
##   3   11     14      0.7%     0.0%     1.8%   6.10e-05      0.0039      0.0134      0.1000
##   4   10     14      0.0%     0.0%     2.3%   6.10e-05      0.0078      0.0152      0.1000
##   4   11     15      0.0%     0.0%     1.1%   3.05e-05      0.0039      0.0105      0.1000
##   3   13     16      0.7%     0.0%     0.8%   1.53e-05   9.77e-04      0.0089      0.1000
##   4   12     16      0.0%     0.0%     0.4%   1.53e-05      0.0020      0.0063      0.1000
##
## Usage:
##   bf_map(protocol = "numeric", ...,
##          fields = list(exponent = <exp>, significand = <sig>))
```

The output table shows exponent/significand combinations that lie on the Pareto front for each total bit count, meaning no other configuration with the same number of bits is strictly better across all quality metrics simultaneously. This often results in multiple rows per bit count, each representing a different trade-off between range and precision. The key columns are *Underflow* (percentage of values too small to represent, becoming zero), *Overflow* (percentage too large, clamped to maximum), *Changed* (percentage differing after round-trip encoding at the specified decimal precision), *Min/Max Res* (step size at smallest and largest representable values, as floating-point precision varies with magnitude), *RMSE* (root mean squared error of the round-trip encoding) and *Max Err* (maximum absolute error across all values).

For example, at 10 total bits, two Pareto-optimal configurations appear, exp=3/sig=7 and exp=4/sig=6. The exp=3 variant achieves lower RMSE (0.041 vs 0.055) and finer Max Res (0.0625 vs 0.125) thanks to the extra significand bit, but incurs 0.7% underflow because 3 exponent bits with bias 3 yield a minimum representable value of $2^{-3} = 0.125$, while the data extends down to approximately 0.1. The exp=4 variant eliminates underflow entirely because 4 exponent bits with bias 7 can represent values as small as $2^{-7} \approx 0.008$, at the cost of coarser precision. Neither configuration strictly dominates the other, so both are reported. This pattern repeats at other bit counts. Configurations with fewer exponent bits offer higher precision but narrower representable range, while those with more exponent bits cover the full range at lower resolution. The choice depends on whether avoiding underflow or maximizing precision matters more for the application at hand. A full list of all evaluated combinations is available via \texttt{View(bf_analyze(...)$results)}.

The following tables show the bitfield specification examples of the meta workflow outlined in the main text. *Theory* explains the logic applied to the respective bit flag, *Type* indicates the encoding type (int = integer, float = floating-point, enum = enumeration, bitmap = sequence of independent boolean flags), and *Value Encoding* shows exactly which values the bit flags

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

are going to have. The rationale for choosing between integer and floating-point encoding for specific variables is discussed in Appendix B.

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

Table A1. Livestock Density Modelling bitfield specifications.

Parameter	Theory	Type	Bits	Value Encoding
Median livestock density	Central tendency of livestock units per area, less affected by outliers than mean.	int	5	0-31 mapped to 0–3.1 n ha^{-1}
Standard deviation	Variability in density estimates. Higher values indicate greater uncertainty.	float	7	exp=4, sig=3
Distribution type	Distribution family for reconstructing full probability distributions from limited parameters.	enum	3	0…7: normal, lognormal, beta, gamma, weibull, poisson, binomial, other
Skewness	Distribution asymmetry. Critical for distinguishing otherwise similar distributions.	enum	4	0…7: highly neg., mod. neg., slightly neg., symmetric, slightly pos., mod. pos., highly pos., other; 8=NA
Kurtosis	Tail weight and peak sharpness. Essential for assessing probability of extreme events.	enum	4	0…7: very light, light, slightly light, mesokurtic, slightly heavy, mod. heavy, heavy, other; 8=NA
Confidence level	Degree of certainty in model outputs, essential for decision-making.	int	5	0-31 mapped to 0–100 %
Uncertainty source	Primary source of prediction uncertainty derived from environmental characteristics.	enum	2	0…3: data coverage, topography, heterogeneity, edge effects
Model selection	Algorithm used for density prediction.	enum	3	0…3: RF, BRT, GAM, CNN; 4=NA
Model Agreement Index	Consistency between predictions from RF, BRT, GAM and CNN at each location. Reveals uncertainty structure invisible in ensemble means.	enum	4	0…7: severe/high/moderate/mild disagreement, fair/good/strong/near-perfect agreement; 8=NA

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

Table B1. Ecological overshoot analysis bitfield specifications.

Parameter	Theory	Type	Bits	Value Encoding
Ecological carrying capacity	Maximum sustainable livestock density given ecological constraints. Foundational concept in sustainable resource management theory.	float	7	exp=3, sig=4
Risk of exceeding capacity	Probability that actual livestock density exceeds ecological carrying capacity. Calculated by integrating the probability density function beyond the carrying capacity threshold, leveraging higher moment data (skewness, kurtosis) from the upstream bitfield. Critical for identifying areas at risk despite seemingly sustainable mean values.	float	6	exp=4, sig=2
Carrying capacity exceedance	Ratio of current density to sustainable density. Values >1 indicate ecological overshoot and potential system degradation. Complements probabilistic risk assessment by providing magnitude of exceedance.	float	6	exp=3, sig=3
Resource limitation – type	Primary limiting factor based on Liebig's Law of the Minimum.	enum	2	0…3: none, water, forage, soil
Resource limitation – magnitude	Severity of resource deficit as demand divided by supply. Small values indicate small deviation of supply from demand and vice versa.	int	3	0-7

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

Table C1. Socioeconomic Analysis & Intervention Planning bitfield specifications.

Parameter	Theory	Type	Bits	Value Encoding
Market distortion type	Economic mechanism causing market failure. Critical for understanding policy intervention points.	enum	3	0…7: none, subsidy, tax, regulation, externality, monopoly, information, public good
Market distortion magnitude	Ordinal severity of deviation from efficient market outcomes.	enum	3	0…7: none, minimal, low, moderate, substantial, high, severe, extreme
GDP adjustment estimates	Estimated correction to conventional GDP accounting if ecosystem degradation were included. Based on natural capital accounting frameworks.	int	6	0-63 mapped to 0–100 %
Economic-Ecological Misalignment Index	Composite measure of disconnect between economic signals and ecological realities. Higher values indicate greater misalignment between economic incentives and sustainable practices.	int	8	0-255
System trajectory	Dynamic state of social-ecological system. Based on resilience theory in complex adaptive systems.	enum	3	0…7: stable, gradual improvement, rapid improvement, cyclic, gradual degradation, rapid degradation, approaching threshold, post-collapse
Intervention priorities	Weighted combination of ecological, economic, and temporal indicators to identify areas where interventions would yield maximum benefit. Considers both current state and trajectory.	int	5	0-31, higher values indicate greater urgency
Cross-sectoral synergy potential	Degree to which livestock interventions could simultaneously advance other development objectives. Based on policy coherence frameworks from sustainable development literature.	bitmap	4	bit 0 = water security, bit 1 = ecosystem conservation, bit 2 = economic development, bit 3 = climate resilience

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

Appendix D: Encoding Theory and Design Choices

At the bit level, all bitfield encodings store unsigned integers. The two fundamental encoding strategies differ in how these integers are *interpreted*, either as linearly scaled values (integer encoding) or as structured floating-point representations with exponent and significand fields (floating-point encoding). Categorical encoding, which maps discrete classes to indices $0, 1, 2, \ldots, k-1$, is a constrained special case of integer encoding where the integers are simply used as class labels without arithmetic interpretation, always starting from zero with no gaps. The choice between integer and floating-point encoding for continuous variables involves trade-offs that merit careful consideration.

D1 Integer Encoding and Fixed-Point Equivalence

Integer encoding maps a bounded value range linearly onto the available bit states. For n bits encoding a range $[a, b]$, the resolution is $(b-a)/(2^n-1)$, distributed uniformly across the entire range. For example, encoding livestock density from 0 to 3.1 $\mathrm{n\,ha^{-1}}$ in 5 bits yields $2^5 = 32$ states with a uniform resolution of $0.1\,\mathrm{n\,ha^{-1}}$.

This is mathematically equivalent to fixed-point encoding, where the integer value is implicitly scaled by a constant factor to represent fractional quantities. Whether the user specifies '5 bits for range 0–3.1' (integer with range mapping) or '5 bits with scaling factor $3.1/31 = 0.1$' (fixed-point), the encoded bit patterns and achievable precision are identical. The range-based interface is arguably more intuitive for domain scientists, which is why the `bitfield` package exposes integer encoding through range parameters rather than explicit fixed-point notation.

Integer encoding is the natural choice for variables with well-understood, bounded ranges where uniform precision is desirable, such as proportions (0–1), percentages (0–100), indices with defined bounds, or any quantity where the measurement uncertainty is roughly constant across the range. The number of allocated bits directly determines the precision, making the trade-off transparent.

D2 Floating-Point Encoding

Floating-point encoding divides the available bits into exponent and significand (mantissa) fields, optionally preceded by a sign bit. The exponent determines the magnitude (power of 2), while the significand provides precision within each magnitude interval. This produces a logarithmic precision profile where resolution is finest near zero and coarsens with increasing magnitude.

Specifically, for e exponent bits and s significand bits, the bias $\beta = 2^{e-1} - 1$ determines the representable range. Within each power-of-2 interval $[2^k, 2^{k+1})$, there are exactly 2^s uniformly spaced values. The step size at magnitude m is approximately $m/2^s$, meaning relative precision is constant while absolute precision varies.

This encoding is appropriate for variables spanning multiple orders of magnitude, where small values require fine resolution and large values tolerate coarser steps. The bias parameter shifts the representable window along the magnitude axis but does not alter the logarithmic precision structure because this is an inherent property of encoding values along an exponentially increasing scale.

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

D3 Choosing Between Integer and Floating-Point

The livestock density workflow (Table A1) illustrates both strategies applied to related variables.

Median livestock density uses integer encoding because the range (0–$3.1\,\mathrm{n\,ha^{-1}}$) is well-bounded and management decisions require comparable precision across the entire range. A density of 2.5 is no less important to resolve than a density of 0.5. Floating-point encoding would waste bits providing sub-0.001 resolution near zero while giving coarser steps in the upper range where management thresholds apply.

Standard deviation, by contrast, uses floating-point encoding because it can span several orders of magnitude depending on the underlying data. An SD of 0.3 near a low-density area and an SD of 8 near a high-density area both need to be represented meaningfully. The logarithmic precision profile matches this requirement, providing fine resolution for small SDs (where the difference between 0.1 and 0.5 matters for distribution reconstruction) and coarser resolution for large SDs (where the difference between 7 and 8 is less consequential).

Confidence level uses integer encoding because it represents a bounded proportion (0–1) where uniform resolution is appropriate because the difference between $60\,\%$ and $65\,\%$ confidence is as meaningful as the difference between $10\,\%$ and $15\,\%$.

As a general guideline, variables with bounded, well-understood ranges and roughly constant measurement uncertainty benefit from integer encoding. Variables with open-ended or multi-order-of-magnitude ranges, where relative rather than absolute precision matters, benefit from floating-point encoding.

D4 Practical Considerations

Unit choice as an encoding strategy. The number of bits required for integer encoding depends directly on the value range, which in turn depends on the chosen unit. Livestock density expressed in animals per km^2 might range from 0 to 290, requiring 9 bits for sub-unit resolution, while the same quantity in animals per ha ranges from 0 to 3.1, needing only 5 bits for a resolution of $0.1\,\mathrm{n\,ha^{-1}}$ ($\sim 3\,\%$ of the maximum). Choosing a unit that compresses the value range without losing domain meaning is a simple but effective strategy for reducing bit budget.

Storage of wide bitfields. Bitfields up to 32 bits fit naturally in a single unsigned integer raster layer. For wider bitfields, multiple integer layers can be used (tracked automatically by the registry). Storing a bitfield as a 64-bit floating-point value is technically possible but introduces a subtle limitation. The 64-bit IEEE 754 double format provides only 53 bits of integer precision (52 significand bits plus one implicit leading bit). Bitfields exceeding 53 bits would silently lose their least significant bits, a particularly insidious form of data corruption. For this reason, wide bitfields should use 64-bit integer storage formats or be split across multiple 32-bit integer layers.

Competing interests. The author declares no competing interests.

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

Acknowledgements. I thank Carsten Meyer, Kimberly Thompson, Daniel Bahrdt, and Julián Equihua for their contributions during the early stages of this work. The R package implementation, reproducible examples, and portions of the manuscript text were developed with assistance from Claude (Anthropic), a large language model. This included code architecture, debugging, code review, and drafting of technical descriptions. All scientific concepts, analytical decisions, and interpretations are my own. *Financial support:* This work was supported in part by the DFG grant FZT-118 and in part by the Land & Carbon Lab from the Bezos Earth Fund.

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

References

Barton, C. M., Lee, A., Janssen, M. A., van der Leeuw, S., Tucker, G. E., Porter, C., Greenberg, J., Swantek, L., Frank, K., Chen, M., and Jagers, H. R. A.: How to make models more useful, Proceedings of the National Academy of Sciences, 119, e2202112119, https://doi.org/10.1073/pnas.2202112119, 2022.

Bush, A., Sollmann, R., Wilting, A., Bohmann, K., Cole, B., Balzter, H., Martius, C., Zlinszky, A., Calvignac-Spencer, S., Cobbold, C. A., Dawson, T. P., Emerson, B. C., Ferrier, S., Gilbert, M. T. P., Herold, M., Jones, L., Leendertz, F. H., Matthews, L., Millington, J. D. A., Olson, J. R., Ovaskainen, O., Raffaelli, D., Reeve, R., Rödel, M.-O., Rodgers, T. W., Snape, S., Visseren-Hamakers, I., Vogler, A. P., White, P. C. L., Wooster, M. J., and Yu, D. W.: Connecting Earth observation to high-throughput biodiversity data, Nature Ecology & Evolution, 1, 0176, https://doi.org/10.1038/s41559-017-0176, 2017.

Cavender-Bares, J., Schneider, F. D., Santos, M. J., Armstrong, A., Carnaval, A., Dahlin, K. M., Fatoyinbo, L., Hurtt, G. C., Schimel, D., Townsend, P. A., Ustin, S. L., Wang, Z., and Wilson, A. M.: Integrating remote sensing with ecology and evolution to advance biodiversity conservation, Nature Ecology & Evolution, 6, 506–519, https://doi.org/10.1038/s41559-022-01702-5, 2022.

DataCite Metadata Working Group: DataCite Metadata Schema Documentation for the Publication and Citation of Research Data and Other Research Outputs v4.5, https://doi.org/10.14454/G8E5-6293, 2024.

Dietrich, J. P., Bodirsky, B. L., Humpenöder, F., Weindl, I., Stevanović, M., Karstens, K., Kreidenweis, U., Wang, X., Mishra, A., Klein, D., Ambrósio, G., Lotze-Campen, H., and Popp, A.: MAgPIE 4 - a modular open-source framework for modeling global land systems, Geoscientific Model Development, 12, 1299–1317, https://doi.org/10.5194/gmd-12-1299-2019, 2019.

Edwards, P. N., Mayernik, M. S., Batcheller, A. L., Bowker, G. C., and Borgman, C. L.: Science friction: Data, metadata, and collaboration, Social Studies of Science, 41, 667–690, https://doi.org/10.1177/0306312711413314, pMID: 22164720, 2011.

Ehrmann, S.: bitfloat/bitfield: publication ready (EGU earth observation), https://doi.org/10.5281/zenodo.18601920, 2026a.

Ehrmann, S.: bitfloat/examples: publication ready (EGU earth observation), https://doi.org/10.5281/zenodo.18602257, 2026b.

Ehrmann, S.: bitfloat/standards: publication ready (EGU earth observation), https://doi.org/10.5281/zenodo.18602313, 2026c.

Frege, G.: Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens, Halle, S.: Louis Nebert, 1879.

Gil, Y., David, C. H., Demir, I., Essawy, B. T., Fulweiler, R. W., Goodall, J. L., Karlstrom, L., Lee, H., Mills, H. J., Oh, J.-H., Pierce, S. A., Pope, A., Tzeng, M. W., Villamizar, S. R., and Yu, X.: Toward the Geoscience Paper of the Future: Best practices for documenting and sharing research from data to software to provenance, Earth and Space Science, 3, 388–415, https://doi.org/10.1002/2015EA000136, 2016.

Goble, C., Cohen-Boulakia, S., Soiland-Reyes, S., Garijo, D., Gil, Y., Crusoe, M. R., Peters, K., and Schober, D.: FAIR Computational Workflows, Data Intelligence, 2, 108–121, https://doi.org/10.1162/dint_a_00033, 2020.

Goring, S. J., Weathers, K. C., Dodds, W. K., Soranno, P. A., Sweet, L. C., Cheruvelil, K. S., Kominoski, J. S., Rüegg, J., Thorn, A. M., and Utz, R. M.: Improving the culture of interdisciplinary collaboration in ecology by expanding measures of success, Frontiers in Ecology and the Environment, 12, 39–47, https://doi.org/10.1890/120370, 2014.

Hijmans, R. J. and Ghosh, A.: luna: Tools for Satellite Remote Sensing (Earth Observation) Data Processing, r package version 0.3-6, 2023.

Hollmann, R., Merchant, C. J., Saunders, R., Downy, C., Buchwitz, M., Cazenave, A., Chuvieco, E., Defourny, P., de Leeuw, G., Forsberg, R., Holzer-Popp, T., Paul, F., Sandber Sørensen, S., Sathyendranath, S., van Roozendael, M., and Wagner, W.: The ESA Climate Change Initiative: Satellite Data Records for Essential Climate Variables, Bulletin of the American Meteorological Society, 94, 1541–1552, https://doi.org/10.1175/BAMS-D-11-00254.1, 2013.

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

Howison, J. and Crowston, K.: Collaboration Through Open Superposition: A Theory of the Open Source Way, MIS Quarterly, 38, 29–50, https://www.jstor.org/stable/26554867, 2014.

IEEE Computer Society: IEEE Standard for Floating-Point Arithmetic, Tech. Rep. IEEE Std 754-2008, New York, NY, USA, https://doi.org/10.1109/IEEESTD.2008.4610935, 2008.

Leibniz, G. W.: De Alphabeto cogitationum humanorum, in: Akademie, vol. 6, pp. 270 – 273, 1681.

Ma, X., Fox, P., Tilmes, C., Jacobs, K., and Waple, A.: Capturing provenance of global change information, Nature Climate Change, 4, 409–413, https://doi.org/10.1038/nclimate2141, 2014.

Magagna, B., Rosati, I., Stoica, M., Schindler, S., Moncoiffe, G., Devaraju, A., Peterseil, J., and Huber, R.: The I-ADOPT Interoperability Framework for FAIRer data descriptions of biodiversity, https://doi.org/10.48550/arXiv.2107.06547, 2021.

Malik, A. and Schaeffer, R.: Integrated assessment modelling and input-output analysis, Economic Systems Research, 36, 501–507, https://doi.org/10.1080/09535314.2024.2408660, 2024.

Missier, P., Belhajjame, K., and Cheney, J.: The W3C PROV family of specifications for modelling provenance metadata, in: Proceedings of the 16th International Conference on Extending Database Technology, EDBT '13, p. 773–776, Association for Computing Machinery, New York, NY, USA, ISBN 9781450315975, https://doi.org/10.1145/2452376.2452478, 2013.

Otto, I. M., Biewald, A., Coumou, D., Feulner, G., Köhler, C., Nocke, T., Blok, A., Gröber, A., Selchow, S., Tyfield, D., Volkmer, I., Schellnhuber, H. J., and Beck, U.: Socio-economic data for global environmental change research, Nature Climate Change, 5, 503–506, https://doi.org/10.1038/nclimate2593, 2015.

Piwowar, H. A. and Vision, T. J.: Data reuse and the open data citation advantage, PeerJ, 1, e175, https://doi.org/10.7717/peerj.175, 2013.

Potapov, P., Turubanova, S., Hansen, M. C., Tyukavina, A., Zalles, V., Khan, A., Song, X.-P., Pickens, A., Shen, Q., and Cortez, J.: Global maps of cropland extent and change show accelerated cropland expansion in the twenty-first century, Nature Food, 3, 19–28, https://doi.org/10.1038/s43016-021-00429-z, 2022.

Pörtner, H.-O., Scholes, R. J., Arneth, A., Barnes, D. K. A., Burrows, M. T., Diamond, S. E., Duarte, C. M., Kiessling, W., Leadley, P., Managi, S., McElwee, P., Midgley, G., Ngo, H. T., Obura, D., Pascual, U., Sankaran, M., Shin, Y. J., and Val, A. L.: Overcoming the coupled climate and biodiversity crises and their societal impacts, Science, 380, eabl4881, https://doi.org/10.1126/science.abl4881, 2023.

Ram, K.: Git can facilitate greater reproducibility and increased transparency in science, Source Code for Biology and Medicine, 8, 7, https://doi.org/10.1186/1751-0473-8-7, 2013.

Roy, D. P., Borak, J. S., Devadiga, S., Wolfe, R. E., Zheng, M., and Descloitres, J.: The MODIS Land product quality assessment approach, Remote Sensing of Environment, 83, 62–76, https://doi.org/10.1016/S0034-4257(02)00087-1, the Moderate Resolution Imaging Spectro-radiometer (MODIS): a new generation of Land Surface Monitoring, 2002.

Runge, J., Bathiany, S., Bollt, E., Camps-Valls, G., Coumou, D., Deyle, E., Glymour, C., Kretschmer, M., Mahecha, M. D., Muñoz-Marí, J., van Nes, E. H., Peters, J., Quax, R., Reichstein, M., Scheffer, M., Schölkopf, B., Spirtes, P., Sugihara, G., Sun, J., Zhang, K., and Zscheis-chler, J.: Inferring causation from time series in Earth system sciences, Nature Communications, 10, 2553, https://doi.org/10.1038/s41467-019-10105-3, 2019.

Rüegg, J., Gries, C., Bond-Lamberty, B., Bowen, G. J., Felzer, B. S., McIntyre, N. E., Soranno, P. A., Vanderbilt, K. L., and Weathers, K. C.: Completing the data life cycle: using information management in macrosystems ecology research, Frontiers in Ecology and the Environment, 12, 24–30, https://doi.org/10.1890/120375, 2014.

Sarikhani, M. and Wendelborn, A.: Mechanisms for provenance collection in scientific workflow systems, Computing, 100, 439–472, https://doi.org/10.1007/s00607-017-0578-1, 2018.

https://doi.org/10.5194/egusphere-2026-2041
Preprint. Discussion started: 10 June 2026

Stodden, V., Seiler, J., and Ma, Z.: An empirical analysis of journal policy effectiveness for computational reproducibility, Proceedings of the National Academy of Sciences, 115, 2584–2589, https://doi.org/10.1073/pnas.1708290115, 2018.

Tobler, W. R.: A Computer Movie Simulating Urban Growth in the Detroit Region, Economic Geography, 46, 234–240, https://doi.org/10.2307/143141, 1970.

Vogt, P. and Riitters, K.: GuidosToolbox: universal digital image object analysis, European Journal of Remote Sensing, 50, 352–361, https://doi.org/10.1080/22797254.2017.1330650, 2017.

Voinov, A. and Shugart, H. H.: 'Integronsters', integral and integrated modeling, Environmental Modelling & Software, 39, 149–158, https://doi.org/10.1016/j.envsoft.2012.05.014, 2013.

Vrandečić, D. and Krötzsch, M.: Wikidata, Communications of the ACM, 57, 78–85, https://doi.org/10.1145/2629489, 2014.

Weibel, S.: The Dublin Core: A Simple Content Description Model for Electronic Resources, Bulletin of the American Society for Information Science and Technology, 24, 9–11, https://doi.org/10.1002/bult.70, 1997.

Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., Gonzalez-Beltran, A., Gray, A. J. G., Groth, P., Goble, C., Grethe, J. S., Heringa, J., 't Hoen, P. A. C., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S. J., Martone, M. E., Mons, A., Packer, A. L., Persson, B., Rocca-Serra, P., Roos, M., van Schaik, R., Sansone, S.-A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M. A., Thompson, M., van der Lei, J., van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J., and Mons, B.: The FAIR Guiding Principles for scientific data management and stewardship, Scientific Data, 3, 160 018, https://doi.org/10.1038/sdata.2016.18, 2016.

Zins, C.: Conceptual approaches for defining data, information, and knowledge, Journal of the American Society for Information Science and Technology, 58, 479–493, https://doi.org/10.1002/asi.20508, 2007.