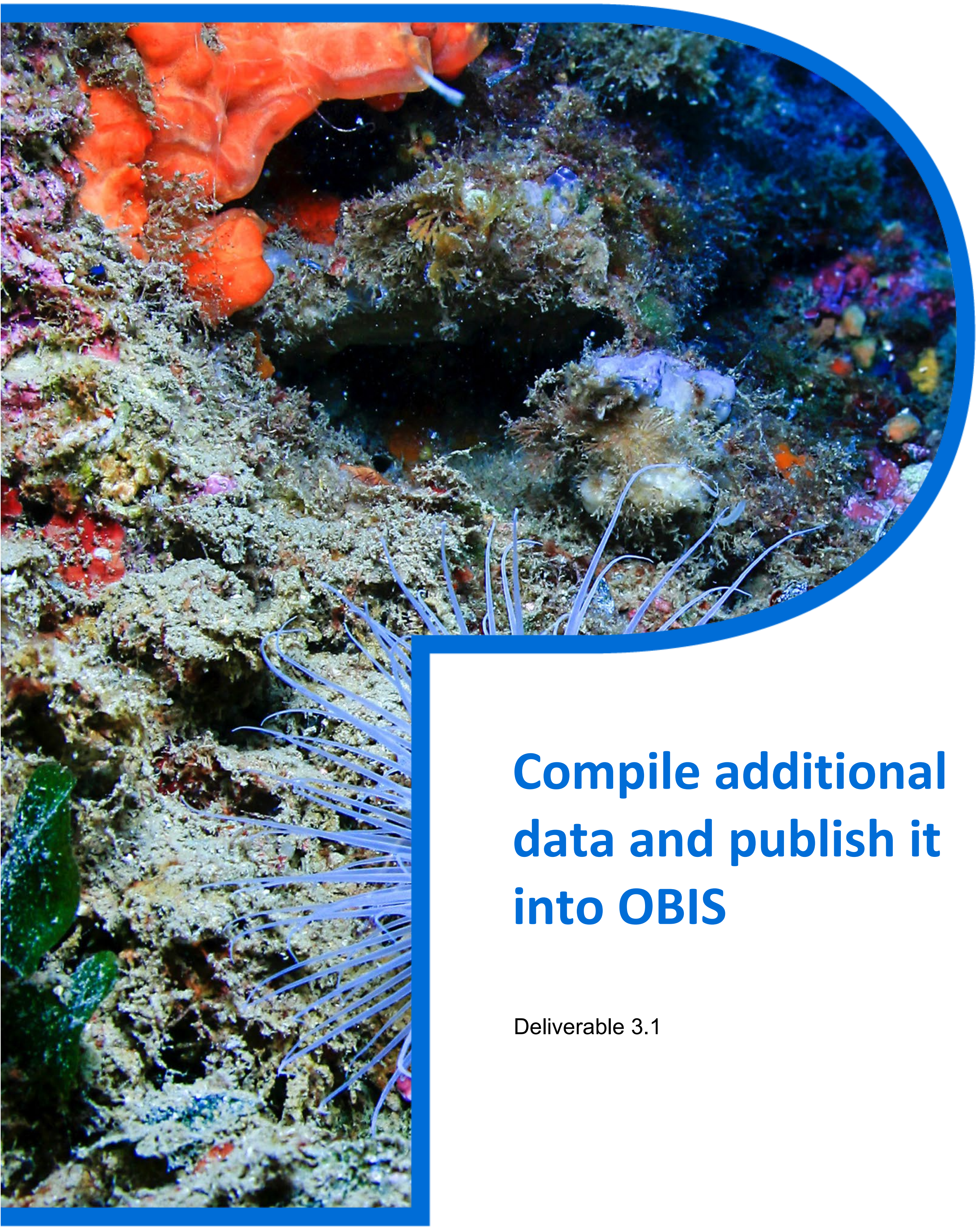
Co-funded by the European Union

Grant Agreement no.
101059988

UK Research and Innovation

Grant numbers
10067934 JNCC
10069750 SAMS



Compile additional data and publish it into OBIS

Deliverable 3.1

Authors |

Silas C. Principe, Pieter Provoost, Ward Appeltans, Sunni Heikes-Knapton, Anna M. Addamo, Mark John Costello

Publisher |

MPA Europe

How to Cite |

Principe, S.C., Provoost, P., Appeltans, W., Heikes-Knapton, S., Addamo, A.M., Costello, M.J. 2023. *MPA Europe Report Deliverable 3.1* Compile additional data and publish it into OBIS. 38 pp. DOI:10.5281/zenodo.8096273

Supported by/In partnership with

Co-funded by the European Union

UK Research and Innovation

***MPA Europe** is co-funded by the European Union under the Horizon Europe programme (Grant Agreement: 101059988). UK participants in Horizon Europe Project MPA Europe are supported by UKRI grant numbers 10067934 JNCC and 10069750 SAMS. Views and opinions expressed are those of the authors only and do not necessarily reflect that of the European Union or UK Research and Innovation. Neither the European Union nor the granting authority can be held responsible for them.*

DOCUMENT TRACKING DETAILS

PROJECT INFORMATION	
Project acronym	MPA Europe
Project title	Marine Protected Areas Europe
Instrument	HORIZON-CL6-2021-BIODIV-01-12
Project number	101059988
Starting date	1st January 2023
Duration	40 months
Project Coordinator	Mark John Costello

DELIVERABLE INFORMATION	
Deliverable Title	Compile additional data and publish it into OBIS
Deliverable Number	D3.1
Work Package	WP3
WP Co-Leader(s)	Ward Appeltans
Deliverable Beneficiary	UNESCO IOC
Authors	Silas C. Principe, Pieter Provoost, Ward Appeltans, Sunni Heikes-Knapton, Anna M. Addamo, Mark John Costello
Due date	30.06.2023
Submission date	30.06.2023
Type of deliverable	Data
Dissemination level	Public
How to cite	Principe, S.C., Provoost, P., Appeltans, W., Heikes-Knapton, S., Addamo, A.M., Costello, M.J. 2023. *MPA Europe Report Deliverable 3.1.* Compile additional data and publish it into OBIS. 38 pp. DOI:10.5281/zenodo.8096273

VERSION	Date	Description
V0.1	23.06.2023	First draft
V0.2	27.06.2023	Second draft
V0.3	29.06.2023	Third draft
V1.0	30.06.2023	Submitted version

Table of Contents

List of acronyms and abbreviations

Acronym/Abbreviation	Meaning/Full text
OBIS	Ocean Biodiversity Information System
IODE	International Oceanographic Data and Information Exchange
IOC	Intergovernmental Oceanographic Commission
GBIF	Global Biodiversity Information Facility
WoRMS	World Register of Marine Species
MPA	Marine Protected Area
MSP	Maritime Spatial Planning
NGO	Non-Governmental Organization
EEZ	Exclusive Economic Zone
IHO	International Hydrographic Organization
BioTIME	Global database of assemblage time series

List of figures

List of tables

The MPA Europe project

MPA Europe is a project co-funded by the European Union (EU) that will map a network of locations representative of biodiversity in European seas (Atlantic Exclusive Economic Zones of the EU and its neighbours, and all the Mediterranean, Baltic, and Black seas) (see Figure 1).

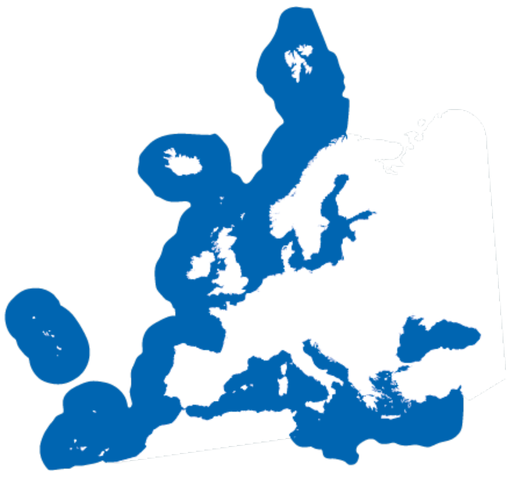


Figure 1: MPA Europe study area

This network will indicate where to prioritise placement of Marine Protected Areas (MPA) for biodiversity protection and how Maritime Spatial Planning (MSP) can maximise blue carbon benefits. By using a holistic set of biodiversity measures, from species to ecosystems (including habitats), and environmental data including carbon storage, water currents, and climate change velocity models, it will be possible to quantify and map both the present and future connectivity within the proposed MPA network. The multiple scenarios provided will support wider and improved MSP by policy makers, industry, and non-governmental organizations (NGOs). The major scientific areas of the project and its outcomes are summarized in the infographic presented in Figure 2.

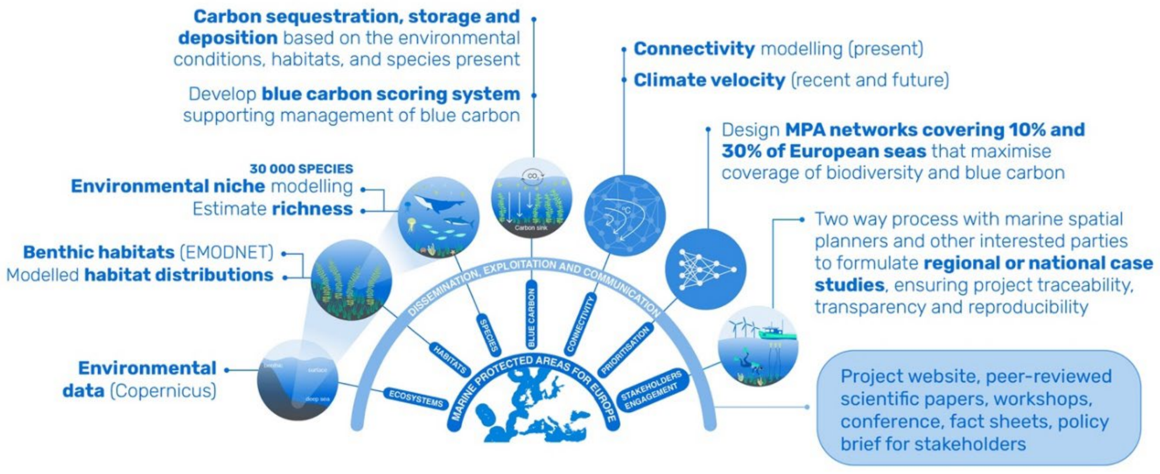

Carbon sequestration, storage and deposition based on the environmental conditions, habitats, and species present
Develop blue carbon scoring system supporting management of blue carbon
Connectivity modelling (present)
Climate velocity (recent and future)
Design MPA networks covering 10% and 30% of European seas that maximise coverage of biodiversity and blue carbon
30 000 SPECIES
Environmental niche modelling
Estimate richness
Benthic habitats (EMODNET)
Modelled habitat distributions
Two way process with marine spatial planners and other interested parties to formulate regional or national case studies, ensuring project traceability, transparency and reproducibility
Environmental data (Copernicus)
surface
deep sea
Carbon sink
DISSEMINATION, EXPLOITATION AND COMMUNICATION
ECOSYSTEMS
HABITATS
SPECIES
BLUE CARBON
CONNECTIVITY
PRIORITISATION
STAKEHOLDERS ENGAGEMENT
MARINE PROTECTED AREAS FOR EUROPE
Project website, peer-reviewed scientific papers, workshops, conference, fact sheets, policy brief for stakeholders

Figure 2. MPA Europe project scientific areas

The inter-disciplinary approach of MPA Europe is supported by a diversity of partners. These include Nord University (Norway), the Intergovernmental Oceanographic Commission of UNESCO (Belgium), the Scottish Association for Marine Science (Scotland, UK), the Centre of Marine Sciences (Portugal), the Ocean University of China (China), Science Crunchers (Portugal), the University of the Ryukyus (Japan), Aarhus University (Denmark), and the Joint Nature Conservation Committee (England, UK).. The partners include experts in marine biology, taxonomy, ecology, oceanography, biogeochemistry, and biogeography, MPA network design, benthic habitat mapping, carbon dynamics, species distribution, climate modelling and MSP compose the MPA Europe team.

1. OBIS contribution to MPA Europe

OBIS (Ocean Biodiversity Information System), an integral component of the International Oceanographic Data and Information Exchange (IODE) programme of the Intergovernmental Oceanographic Commission (IOC) of UNESCO, is responsible for the **Working Package 3 (WP3)** and will contribute with three pieces of information which will be supplied to the area prioritization process: (i) species distribution models (SDMs) of at least half of the European marine species; (ii) biodiversity metrics for European seas; and (iii) habitat maps considering habitat forming species. In all cases, models will be created using OBIS data and will include predictions for future scenarios according to the Coupled Model Intercomparison Project 6.

The first deliverable of the WP3 is the gathering of additional data on marine species occurrences (and its publishing on OBIS) for use in the habitat range modelling.

2. Deliverable 3.1 Bringing new data to OBIS

Goal of Deliverable 3.1

Compile additional datasets of marine species occurrences for inclusion on OBIS and prepare workflows for future efficiencies to access marine biodiversity data from the Global Biodiversity Information Facility (GBIF) and other sources.

Summary of products

- Standard study area shapefile
- List of species occurring on the study area according to OBIS and GBIF
- Project structure for D3.2 and further steps
- Quality control procedure for D3.2 and further steps
- Code for streamlined download of data from both OBIS and GBIF, under the same structure
- Compilation of potential datasets of interest for the project that are available on GBIF, but not on OBIS
- Compilation of potential datasets available in other sources (in process)
- R package for standardized and documented ingestion of datasets onto OBIS
- Publishing/ingestion of datasets onto OBIS (in process)

Establishing the study area

Our initial task was to establish a consistent shape for the study area that could be utilized by all teams working on different work packages. To define the boundaries of the final shape, we referred to the study area depicted in the MPA Europe. Using a product developed by the Flanders Marine

Institute[1] that intersects the Exclusive Economic Zones (EEZ) and the International Hydrographic Organization (IHO) seas map, we determined the boundaries.

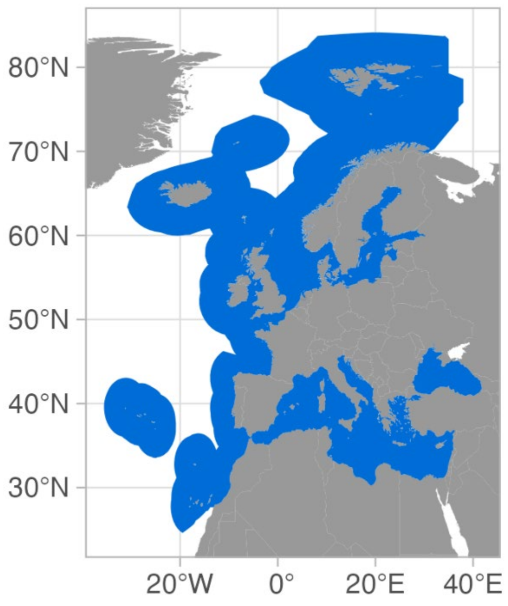


Figure 3. The final standard study area

The whole process of creating the study area, along with the produced shapefiles, is available at the GitHub repository **mpaeu_studyarea** (https://github.com/iobis/mpaeu_studyarea), in Zenodo community "MPA Europe Project" (https://zenodo.org/communities/mpaeurope/), and in the project website (https://mpa-europe.eu/).

Identification of species

Once the study area was defined, our next step was to compile a list of species recorded in the region. We obtained the list of species from both OBIS and GBIF (Table 1). From OBIS, we included only marine or brackish species. Furthermore, we filtered out records from certain kingdoms (Archaea, Bacteria, Fungi, or Protozoa) and records of extinct and non-extant (fossil) species, resulting in a total of **22159** unique species.

From GBIF, we initially gathered all the species records occurring in the study area. Subsequently, we used the World Register of Marine Species (WoRMS) to identify the marine species among them. Finally, we excluded records from the aforementioned kingdoms and extinct/non-extant species, resulting in **22782** unique species.

[1] Available in https://www.marineregions.org/

Table 1. The number of unique species in each dataset (OBIS and GBIF), the number of shared species (species that occur in both datasets), and the total number of species considering both datasets.

Unique OBIS	Unique GBIF	Shared species	Total
6211	6834	15948	28993

This process is documented on the GitHub repository **mpaeu_sdm** (https://github.com/iobis/mpaeu_sdm) and on the Appendix (see page 20 for more details), in Zenodo community "MPA Europe Project" (https://zenodo.org/communities/mpaeurope/), and in the project website (https://mpa-europe.eu/).

Download of data from OBIS and GBIF

Establishing a project structure

Prior to obtaining the occurrence data from OBIS and GBIF, we established a well-defined project structure that would:

1. Be easy to document and comprehend;
2. Facilitate version tracking;
3. Retain data from distinct working stages (i.e., raw, flagged, processed);
4. Allow for quick access;
5. Minimize the overall data size.

With these objectives in mind, we decided to utilize a hive format, employing parquet files[2]. Further documentation is available on the **"MPA Europe docs"** (https://github.com/iobis/mpaeu_docs, with live version on https://iobis.github.io/mpaeu_docs/), in Zenodo community "MPA Europe Project" (https://zenodo.org/communities/mpaeurope/), and in the project website (https://mpa-europe.eu/).

Quality control of data

An essential step in fitting habitat range models is ensuring the availability of data of sufficient quality. Rigorously curated data guarantees that the habitat maps are based on the accurate distribution of species and accurately reflect their environmental requirements. However, effectively controlling the quality of the extensive data involved in the MPA Europe project is a challenging task. To address this, we have developed methods to automatically apply three types of quality control to all data obtained from OBIS and GBIF:

1. Occurrence in seawater: we check if the occurrence point is not on land;
2. Geographical outliers: we identify spatial outliers by an innovative method that considers the presence of barriers when calculating the distance between points. Typically, geographical

[2] More details on https://parquet.apache.org/

outliers are determined based on the Cartesian distance between points. However, for marine species (and even terrestrial ones), barriers play a crucial role in dispersal. For instance, if we consider a species on opposite sides of the Panama strait (Atlantic and Pacific), a straight line between the two points would indicate a short distance. Nevertheless, by accounting for the barrier, we recognize that the animal (or its larvae) would need to travel a much longer distance to reach the other side of the strait (Figure 4);

3. Environmental outliers: we assess if the environmental characteristics of the occurrence point deviate significantly from the overall conditions of species occurrence. This helps identify outliers that may be influenced by unusual environmental conditions and ensures the reliability of the data used in habitat range modeling.

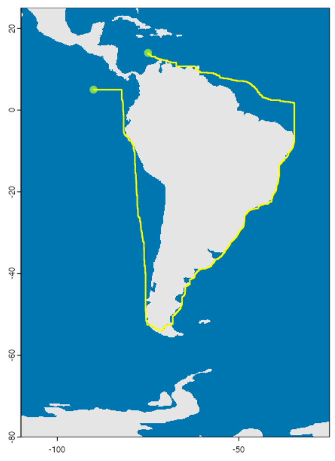


Figure 4. Example of a least cost track between two points on the ocean.

In both the assessment of geographical and environmental outliers, we employed a threshold of 3 times the interquantile range to identify extreme points. However, to avoid removing points in excess, we only marked the most extreme outliers up to a limit of 1% of the total points. This approach ensured that only the most significant outliers were flagged.

The process of downloading data from OBIS and GBIF, as well as standardizing it according to the project structure, is documented in the GitHub repository **mpaeu_sdm** (https://github.com/iobis/mpaeu_sdm) available also on the Appendix (see page 24 for more details), in Zenodo community “MPA Europe Project” (https://zenodo.org/communities/mpaeurope/), and in the project website (https://mpa-europe.eu/). It's important to note that the quality control step mentioned here is part of the upcoming deliverable 3.2 (“Publish maps and models of species environmental niches and geographic distribution in Europe on OBIS”), and a more comprehensive report on it will be provided. Both the quality control procedures and data download were performed using functions from a dedicated package specifically developed for this project. Although the package is still under development, it can be accessed at https://github.com/iobis/mpaeu_msdm, and in the Zenodo community “MPA Europe”, and the project website.

Data harvesting

An integral part of the Deliverable 3.1 was the harvesting of new data not available on OBIS. While OBIS contains a substantial amount of marine data (more than 28 million records only within our study area), many datasets are exclusively housed in other repositories such as GBIF. However, searching through multiple sources adds an extra burden for marine researchers and other OBIS users. Additionally, OBIS, being a biodiversity database focused on the ocean, incorporates specific quality control procedures for marine data and conducts taxonomic matching with WoRMS, ensuring datasets with up-to-date taxonomic information that may not be present in other databases.

We searched for potential datasets that would be relevant for the MPA Europe project using different approaches:

1. **Peer contribution**
   MPA Europe team suggested potential datasets based on their expertise and previous interactions with relevant sources.

2. **BioTIME**
   BioTIME is a database of ecological data time-series across all realms. While a significant portion of the marine datasets available on this platform was obtained from OBIS, some relevant datasets were published only there. To identify these marine studies in the European region that were not present in OBIS, we performed a fuzzy matching of study titles with dataset titles on OBIS. The pre-selected datasets were then manually reviewed to confirm their relevance.

3. **Literature**
   We searched on Web of Science for articles that could potentially contain datasets valuable for our project. We used the following topic search (TS) string: TS=((marine OR ocean* OR coastal) AND (("biodiversity data") OR (dataset) OR ("time series" OR time-series)) AND (species OR occurrence OR biodiversity OR fauna) AND (Europe* OR global)). From the returned list (~2000 articles) we (1) cross-referenced the titles with the dataset names or bibliographic citations from OBIS to determine if the dataset was already included in OBIS, and (2) screened (manually) to identify if the dataset was valuable. Note that this is not a systematic review, but an exploratory search. Because the number of records was considerably large and the screening involves evaluating the data quality and the methods that generated it, in this first phase of the project we screened only the first 100 records (ordered by relevance) and will keep screening in the following months.

4. **Data repositories**
   We searched the data repositories FigShare, Zenodo, and Dryad for datasets related to marine data in our study region (see Appendix, page 25 for more details). Each repository had a distinct search strategy based on its structure. We performed a fuzzy matching of dataset names with dataset titles on OBIS and assessed the relevance of datasets that were

identified as not available in OBIS. In this initial phase, we screened the first 50 records out of approximately 500 and will continue screening in the following months.

5. **Additional data from GBIF**
   In all the previous approaches, once a dataset was identified as relevant it was internally listed for ingestion (i.e. in our control table). For datasets found on GBIF, we followed a separate pipeline based on the principle of "publish once, harvest many times." First, we identified the datasets from which the downloaded GBIF data for species in our study area originated. We selected datasets with at least 50,000 records within the downloaded GBIF data relevant to our study area. Next, we performed a fuzzy matching of the dataset names with those in OBIS. We manually reviewed all the datasets to identify the ones pertinent to our project, which were marked for incorporation into OBIS. The inclusion process involves review and approval by an OBIS node before the dataset can be made available on the OBIS platform. An important aspect to highlight is that we developed code and tools specifically designed to streamline the download of data from both OBIS and GBIF within the project. While it may not be feasible to include all marine data from GBIF in OBIS, this development enables a more efficient and straightforward process for gathering data from multiple sources.

Table 2 (at the end of this document) provides a list of datasets identified for incorporation onto OBIS. We anticipate a contribution of over 3.3 million new occurrence records solely through the processing of non-GBIF datasets. If all GBIF datasets are approved for inclusion by the OBIS nodes, the number of new records could exceed 14 million. Only datasets complying with the license requirements of OBIS (https://manual.obis.org/policy.html#guidelines-on-the-sharing-and-use-of-data-in-obis-1) were listed for ingestion. Note that not all datasets were processed, but the tools created for the ingestion of the data (see next section) will speed up this process in the following months. We also highlight that some of the datasets may be discarded during processing if its quality does not fit OBIS standards.

Planning of data harvesting and ingestion

The datasets listed in Table 1 obtained through peer contribution and from BioTIME are scheduled to undergo complete processing and ingestion by **mid-July 2023**. Datasets listed on the table sourced from data repositories and literature are expected to be processed and ingested by **mid-August 2023**. The screening of the datasets available on data repositories will be concluded by the **end of August 2023,** and the processing will be carried out in sequence.

Ingestion of additional datasets

In adherence to the MPA Europe data management plan and the FAIR principle (Findable, Accessible, Interoperable, and Reusable), we ensured complete documentation of the data ingestion process. To facilitate this, we developed the **obisdi** R package (Figure 5), which establishes a standardized project folder structure and includes files documenting the data processing. This project structure was directly inspired from the “checklist recipe” of the Tracking Invasive Alien Species (TrIAS) project[3].

The **obisdi** package's key functionality is a RMarkdown file, which combines plain text and R code, enabling the mapping of original data to the Darwin Core format required by OBIS. Additionally, the package offers functions for data retrieval from data repositories and the acquisition and addition of metadata. By utilizing this package, users will at the end obtain a folder containing the raw and processed data files, along with documentation/code and metadata. This folder can be transformed into a GitHub repository, and a website can be deployed using GitHub Pages. One example of data processing using the package is available on the Appendix, (see page 33 for more details).

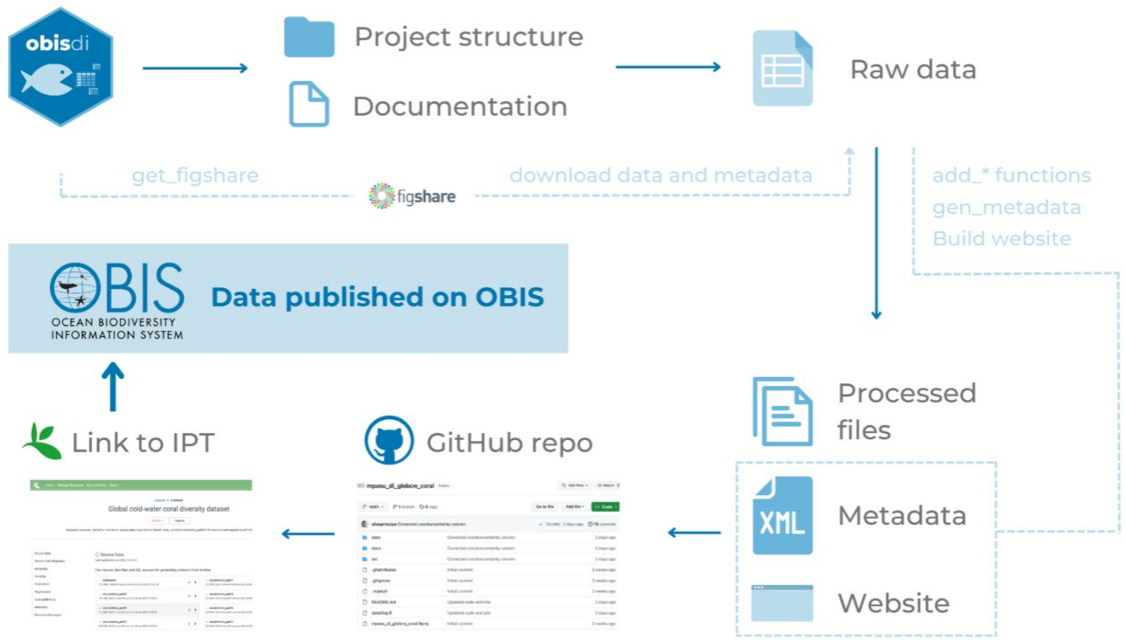


Figure 5. The workflow of using the obisdi package to ingest datasets into OBIS.

Once the data is uploaded to the GitHub repository, it can be directly linked to the IPT (Integrated Publishing Toolkit), which serves as the gateway for incorporating data into OBIS. This approach simplifies the process of providing updates to the dataset, if needed. When a file is modified on GitHub, it is automatically updated on the IPT, ensuring that the most recent version of the dataset is available.

The obisdi package is available on the GitHub repository **obisdi** (https://iobis.github.io/obisdi/), a general description of the first version in Zenodo community “MPA Europe Project” (https://zenodo.org/communities/mpaeurope/), and in the project website (https://mpa-europe.eu/).

[3]Available in https://github.com/trias-project/checklist-recipe

Documentation

Each step of the Working Package 3 is being documented in an open manner. The documentation is primarily carried out through three pathways:

1. "MPA Europe docs": this involves a GitHub repository and a companion webpage specifically dedicated to providing a detailed textual description of the work conducted. It serves as a central resource for accessing comprehensive information about the contribution from OBIS to the project.
2. R packages: the R packages being developed here implements a standardized documentation format for functions. This documentation is designed to facilitate reuse by anyone who downloads the package, ensuring transparency and promoting the use of the methods applied by MPA Europe.
3. Code and README files: The data processing procedures and the application of functions are described in code and accompanying README files. These documents provide thorough explanations of the steps undertaken and help in reproducing the results and understanding the methods.

By employing these multiple instances of documentation, the OBIS team aims to ensure transparency, facilitate knowledge sharing, and enable others to replicate and build upon the work conducted in Working Package 3 of the MPA Europe project.

Table 2. List of datasets selected for ingestion in the OBIS database. In 'Processing', "G" denotes a GitHub repository generated with obisdi and available from https://github.com/iobis/, "NR" means that the dataset is awaiting an OBIS node approval, and "Q" means that it's on queue for processing. Datasets marked with ● on 'P' are those that were already fully published to OBIS, following the ingestion; those with ♦ are being processed; and those empty are on queue. In NB is the number of additional points or events that will be included in OBIS from the dataset (provided that all data is of sufficient quality). "~" means approximate, and refers to datasets that may have a percentage of data in non-marine taxa and that would thus need filtering. In the cases the acronym is not translated, the name reflects the exact name of the dataset (and for which no translation can be supplied).

Dataset	Source	Processing	P	NB
Global cold-water coral diversity dataset	Peer	G-mpaeu_di_globcw_coral	●	552166
Long term monitoring of fish abundances from coastal Skagerrak	Peer/BioTIME	G-mpaeu_di_fish_abund_skagerrak	●	99656
A fine-tuned global distribution dataset of marine forests	Peer	G-mpaeu_di_marine_forests	♦	2467083
Marine microplankton diversity database	BioTIME	G-mpaeu_di_microp_diversity	♦	37524
Shetland Oil Terminal Environmental Advisory Group (SOTEAG) Rocky Shore Survey (Sullom Voe)	BioTIME	G-mpaeu_di_soteag_rocky_shore	♦	91491
Spatial and temporal organisation of a coastal lagoon fish community	BioTIME	G-mpaeu_di_spat_temp_lagoon	♦	819
Environmental Monitoring database (MOD) DNV	GBIF	NR	♦	2372473
Continuous Plankton Recorder Dataset - Sir Alister Hardy Foundation for Ocean Science (SAHFOS)	GBIF	Q		2629628
The archive for marine species and habitats data (DASSH) Data Archive Centre - Statutory Surveys	GBIF	Q		471660
The Danish Environmental Portal, species and habitats-database "Danmarks Miljøportals Naturdatabase"	GBIF	Q		Checking overlaps
Natural England Marine Monitoring surveys	GBIF	Q		469900
Department of Agriculture Environment and Rural Affairs (DAERA) Marine and Fisheries Division Marine Survey Data	GBIF	Q		291105
Plankton Diversity Along the Portuguese Coast in the 1970s	GBIF	Q		408604
Banco de Datos de la Biodiversidad de la Comunitat Valenciana	GBIF	Q		~680000
Conchological Society of Great Britain & Ireland: marine mollusc records	GBIF	Q		178686
Sveriges lantbruksuniversitet (SLU) Aqua Institute of Coastal Research Database for Coastal Fish - KUL	GBIF	Q		274291
Marine benthic dataset (version 1) commissioned by United Kingdom Offshore Operators Association (UKOOA)	GBIF	Q		181714
Yellow-legged gull movements in South Western Iberian Peninsula 2022	GBIF	Q		180939
Suffolk Biodiversity Information Service (SBIS) Dataset	GBIF	Q		~70000
Calf of Man (Isle of Man) Bird Observatory Daily Logs, 1959-2004	GBIF	Q		~400000
Norfolk Biodiversity Information Service (NBIS) Records to December 2016	GBIF	Q		45000
Seasearch Marine Surveys in Wales	GBIF	Q		111966
Botanical Museum, Copenhagen, the Phycology Herbarium	GBIF	Q		106550

Dataset	Source	Processing	P	NB
Agri-Food and Biosciences Institute Marine Surveys	GBIF	Q		100775
Biodiversity of Luiz Saldanha Marine Park	GBIF	Q		84390
Marine invertebrate collection of the Norwegian University of Science and Technology (NTNU) University Museum	GBIF	Q		114650
Ulster Museum Marine Surveys	GBIF	Q		79550
Visiolittoral : Base de données de la communauté de travail du Conservatoire du littoral - Visiolittoral : surveillance naturaliste des sites du Conservatoire du littoral	GBIF	Q		558001
Invertebrates Collection of the Swedish Museum of Natural History	GBIF	Q		106578
Programme de cartographie des habitats marins (CARTHAM): Inventaire biologique dans le cadre de Natura 2000 en Mer	GBIF	Q		70960
Marine Offshore Seabed Survey data held by Joint Nature Conservation Committee (JNCC)	GBIF	Q		86675
Seasearch Marine Surveys in Ireland	GBIF	Q		69997
Waterbird census in Spain	GBIF	Q		246253
Dreal Occitanie - Base Biodiv SINP - Données faune pour le rapportage européen DHFF-DO 2018	GBIF	Q		Checking overlaps
Azorean Biodiversity Portal	GBIF	Q		~10000
Dutch Foundation for Applied Water Research (STOWA) - Limnodata Neerlandica	GBIF	Q		3359932
SILENE-FAUNE-PACA - CEN_PACA_2017_12_18 (from the Conservatoire d'espaces naturels Provence-Aples-Côtes d'Azur)	GBIF	Q		939753
Native biodiversity collapse in the Eastern Mediterranean	Dryad	Q		62000
Global patterns and drivers of beta diversity facets of reef fish faunas	Dryad	Q		To be verified with authors
Microscopic species make the diversity: a checklist of marine flora and fauna around the Island of Sylt in the North Sea	Literature	Q		2758
Using stochastic population process models to predict the impact of climate change	Literature	Q		To be verified with authors
Cross-continental analysis of coastal biodiversity change	Literature	Q		To be verified with authors
Meta-analysis of multidecadal biodiversity trends in Europe	Literature	Q		Higher taxa richness information
A 14-year time series of marine megafauna bycatch in the Italian midwater pair trawl fishery	Literature	Q		> 10000
Half a century of global decline in oceanic	Literature	Q		Abundance

Dataset	Source	Processing	P	NB
sharks and rays				of 58 species

The report represents the most up to date status of the information. Any potential upcoming information may be considered for incorporation in future deliverables to improve the scientific knowledge.

Appendix: R code

Obtaining list of species through R

Code for obtaining the list of species from OBIS

```
#### MPA EUROPE PROJECT ####
# June 2023
# OBIS contribution to the MPA Europe project
# s.principe@unesco.org

#### Get list of marine species occurring on the study area on GBIF

# Load packages
library(sf)
library(terra)
library(tidyverse)
library(robis)
library(worrms)
sf_use_s2(FALSE)

# Load study area
starea <- st_read("data/shapefiles/mpa_europe_starea_v2.shp")
starea <- st_buffer(starea, 0.5)

starea_bbox <- st_bbox(starea)

# Because the study area is quite large, we divide it in blocks to run
# each one separately
grid <- rast(ext(starea_bbox), nrows = 25, ncols = 25)
grid[] <- 1:625
grid <- as.polygons(grid)
grid <- st_as_sf(grid)

starea_grid <- st_intersection(starea, grid)

# Get the checklist of species for each area
for (i in 1:nrow(starea_grid)) {
  cat(i)
  dat <- checklist(geometry = st_as_text(st_geometry(starea_grid[i,])))
  if (i == 1) {
    sp_list <- dat
  } else {
    sp_list <- bind_rows(sp_list, dat)
  }
  rm(dat)
}

# Filter according to project
final_list <- sp_list %>%
  filter(taxonRank == "Species") %>%
  filter(!kingdom %in% c("Archaea", "Bacteria", "Fungi", "Protozoa")) %>%
  filter(taxonomicStatus == "accepted") %>%
  filter(!is.na(is_marine) | !is.na(is_brackish) | !is.na(is_terrestrial) |
!is.na(is_freshwater)) %>%
  rowwise() %>%
  mutate(btm_na = ifelse(sum(is.na(c(is_brackish, is_terrestrial, is_marine))) < 3,
1, NA)) %>%
  mutate(is_freshwater = ifelse(is.na(is_freshwater), FALSE, is_freshwater)) %>%
  filter(!is_freshwater & !is.na(btm_na)) %>%
  filter(!is.na(phylum)) %>%
  distinct(scientificName, .keep_all = T)

# Check which ones are extant species by getting the full information from WoRMS
# We need to do in chunks, otherwise the server returns an error
```

```
breaks <- seq(1, nrow(final_list), by = 49)
breaks <- c(breaks, nrow(final_list))
breaks <- breaks[-1]
st <- 1

cli::cli_progress_bar("Checking species", total = length(breaks))
for (i in 1:length(breaks)) {
  info <- wm_record(final_list$taxonID[st:breaks[i]])
  if (nrow(info) < length(final_list$taxonID[st:breaks[i]])) {
    stop("Not equal number retrieved")
  }
  if (i == 1) {
    is_ext <- info[,c("AphiaID", "isExtinct")]
  } else {
    is_ext <- bind_rows(is_ext, info[,c("AphiaID", "isExtinct")])
  }
  st <- breaks[i]+1
  cli::cli_progress_update()
}
cli::cli_progress_done()

# Add information to the main table
colnames(is_ext)[1] <- "taxonID"

# Save a flagged version
final_list <- left_join(final_list, is_ext, by = "taxonID")
final_list$isExtinct[is.na(final_list$isExtinct)] <- -1

write.csv(final_list,
          paste0("data/obis_splist_flagged_", format(Sys.Date(), "%Y%m%d"),
".csv"),
          row.names = F)

# Save one without the extincts
final_list <- final_list[final_list$isExtinct != 1, ]

write.csv(final_list,
          paste0("data/obis_splist_", format(Sys.Date(), "%Y%m%d"), ".csv"),
          row.names = F)

# END
```

Code for obtaining the list of species from GBIF

```
#### MPA EUROPE PROJECT ####
# June 2023
# OBIS contribution to the MPA Europe project
# s.principe@unesco.org

#### Get list of marine species occurring on the study area on GBIF

# Load packages
library(arrow)
library(sf)
library(terra)
library(tidyverse)
library(worrms)
sf_use_s2(FALSE)

# Load study area
starea <- st_read("data/shapefiles/mpa_europe_starea_v2.shp")
starea <- st_buffer(starea, 0.5)

starea_bbox <- st_bbox(starea)

# Because the study area is quite large, we divide it in blocks to run
# each one separately
grid <- rast(ext(starea_bbox), nrows = 15, ncols = 15)
grid[] <- 1:225
grid <- as.polygons(grid)
grid <- st_as_sf(grid)

starea_grid <- st_intersection(starea, grid)

# Get information from GBIF
# You can both use the AWS full export, or download first a local parquet with all
the
# records in the study area. (see get_data_gbif.R)
gbif_snapshot <- "s3://gbif-open-data-eu-central-1/occurrence/2023-06-
01/occurrence.parquet"
#gbif_snapshot <- "~/Research/mpa_europe/mpaeu_msdm/data/gbif.parquet"
df <- open_dataset(gbif_snapshot)

for (i in 1:nrow(starea_grid)) {
  cat(i, "\n")
  # Define area
  starea_gr_bbox <- st_bbox(starea_grid[i,])

  # Obtain data
  dat <- df %>%
    select(genus, species, scientificname, decimallongitude, decimallatitude,
taxonrank, kingdom) %>%
    #slice_head(n=1) %>%
    filter(
      decimallongitude >= starea_gr_bbox["xmin"] & decimallongitude <=
starea_gr_bbox["xmax"],
      decimallatitude >= starea_gr_bbox["ymin"] & decimallatitude <=
starea_gr_bbox["ymax"]
    ) %>%
    filter(taxonrank == "SPECIES") %>%
    filter(!kingdom %in% c("Archaea", "Bacteria", "Fungi", "Protozoa")) %>%
    #select(genus, species, scientificname, decimallongitude, decimallatitude) %>%
    collect()

  # Convert to SF
  dat_sf <- st_as_sf(dat, coords = c("decimallongitude", "decimallatitude"))
  st_crs(dat_sf) <- st_crs(starea_grid)

  # See if is inside the study area
  inter <- st_contains(starea_grid[i,1], dat_sf, sparse = F)

  dat <- dat[inter[1,],]

  # Retrieve only unique species
  dat <- dat %>%
    distinct(scientificname, .keep_all = T) %>%
    select(-decimallongitude, -decimallatitude, -taxonrank)

  if (i == 1) {
    sp_list <- dat
  } else {
    sp_list <- bind_rows(sp_list, dat)
  }
  rm(dat)
}

# Get only the unique species
sp_list <- sp_list %>%
  distinct(scientificname, .keep_all = T)

# Check which are marine species
breaks <- seq(1, nrow(sp_list), by = 50)
breaks <- c(breaks, nrow(sp_list))
breaks <- breaks[-1]
st <- 1

# Get one example of result to use when nothing is returned
ex <- wm_records_names("Acanthurus chirurgus")
ex <- ex[[1]]
ex[,] <- NA

# Run for all
cli::cli_progress_bar("Checking species", total = length(breaks))
for (i in 1:length(breaks)) {
  recs <- try(wm_records_names(sp_list$species[st:breaks[i]]))
  if (class(recs) != "try-error") {
    recs <- lapply(1:length(recs), function(z){
      x <- recs[[z]]
      if (nrow(x) < 1) {
        x <- ex
      } else {
        if (nrow(x) > 1) {
          x <- x[x$status != "unaccepted",]
          if (nrow(x) > 1) {
            if ("accepted" %in% x$status) {
              x <- x[x$status == "accepted",]
            }
          }
          if (nrow(x) > 1) {
            auth <- sp_list$scientificname[st:breaks[i]][z]
            auth <- str_trim(gsub("\\(|\\)", "", auth))
            full_names <- mutate(x, f_nam = paste(scientificname, authority))
            ldist <- adist(auth, full_names$f_nam)[1,]
            x <- x[which.min(ldist),]
            warning("Multiple matches for ", auth, ".\nMatched with: ",
x$scientificname, " ", x$authority)
          }
          if (nrow(x) < 1) {
            x <- ex
          }
        } else {
          x
        }
      }
      return(x)
    })
    recs <- bind_rows(recs)
    recs$gbif_species <- sp_list$species[st:breaks[i]]
    recs$gbif_names <- sp_list$scientificname[st:breaks[i]]
    if (i == 1) {
      full_recs <- recs
    } else {
      full_recs <- bind_rows(full_recs, recs)
    }
  } else {
    recs <- ex
    recs <- tibble::add_row(recs, AphiaID = rep(NA, length(st:breaks[i])-1))
    recs$gbif_species <- sp_list$species[st:breaks[i]]
    recs$gbif_names <- sp_list$scientificname[st:breaks[i]]
    if (i == 1) {
      full_recs <- recs
    } else {
      full_recs <- bind_rows(full_recs, recs)
    }
  }
  st <- breaks[i]+1
  cli::cli_progress_update()
}
```

```
cli::cli_progress_done()

# Remove those that are not marine
final_list <- full_recs[!is.na(full_recs$AphiaID),]
final_list$isExtinct <- ifelse(is.na(final_list$isExtinct),
                               -1, final_list$isExtinct)

# Save one with the flags
write.csv(final_list,
          paste0("data/gbif_splist_flagged_", format(Sys.Date(), "%Y%m%d"),
".csv"),
          row.names = F)

# Save one without the extincts and with only the unique species
final_list <- final_list[final_list$isExtinct != 1, ]
final_list <- final_list %>%
  distinct(valid_name, .keep_all = T)

write.csv(final_list,
          paste0("data/gbif_splist_", format(Sys.Date(), "%Y%m%d"), ".csv"),
          row.names = F)

# END
```

Downloading species data

Download of data from OBIS

```
#### MPA EUROPE PROJECT ####
# June 2023
# OBIS contribution to the MPA Europe project
# s.principe@unesco.org

### Get the data from OBIS
# Load packages
library(robis)
library(obissdm)

# Load species list
splist <- read.csv("data/obis_splist_20230622.csv")

# Get data from OBIS and save
mp_get_obis(sci_names = splist$taxonID, mode = "full")

# END
```

Download of data from GBIF

```
#### MPA EUROPE PROJECT ####
# June 2023
# OBIS contribution to the MPA Europe project
# s.principe@unesco.org

### Get the data from GBIF
# Load packages
library(rgbif)
library(worrms)
library(obissdm)
library(dplyr)

# Load species list
splist <- read.csv("data/gbif_splist_20230623.csv")

splist$full_name <- paste(
  splist$scientificname, splist$authority
)
```

```
colnames(splist)[1] <- "taxonID"

# Check GBIF taxonomic backbone (to be removed on next version of code
# as it will be possible to get the ID directly from GBIF in the new version
# of the "check_gbif_species.R")
check_names <- name_backbone_checklist(splist$full_name,
                                       strict = T)
check_names$taxonID <- splist$taxonID

# See if any name was not matched
sum(is.na(check_names$usageKey))

non_match <- splist[is.na(check_names$usageKey),]

# Manually correct those
# Check using only the species name
check_nm <- name_backbone_checklist(non_match$scientificname,
                                    strict = T)
check_nm$taxonID <- non_match$taxonID

# Bind those that are already ok
splist_match <- left_join(splist, check_names[,c("taxonID", "usageKey")],
                          by = "taxonID")
splist_match <- left_join(splist_match, check_nm[,c("taxonID", "usageKey")],
                          by = "taxonID")
splist_match$usageKey <- splist_match$usageKey.x
splist_match$usageKey[!is.na(splist_match$usageKey.y)] <-
splist_match$usageKey.y[!is.na(splist_match$usageKey.y)]

splist_match <- select(splist_match, -usageKey.x, -usageKey.y)

non_match <- check_nm[is.na(check_nm$usageKey),]

# Get data from GBIF and save
mp_get_gbif(sci_names = splist_match$usageKey)

# END
```

Data harvesting

Obtain list of datasets from multiple sources

```
# Get records from FigShare using API connection
library(httr)

# Create a function to retrieve the records for a certain query
query_fig <- '{
  "item_type": 3,
  "search_for": "(:title: marine OR :title: ocean OR :title: coastal) AND (:title:
europe OR :title: global) AND (:title: species OR :title: biodiversity)",
  "limit": 1000,
  "offset": 0
}'

get_figshare <- function(query, maxtry = 7000){

  off <- seq(0, maxtry, by = 1000)

  retnum <- 1000

  k <- 1

  allres <- list()

  while(retnum == 1000 & k <= length(off)) {
    query <- gsub('"offset": [[:digit:]]*', paste0('"offset": ', off[k]), query)

    response <- POST("https://api.figshare.com/v2/articles/search", body=query,
                     httr::add_headers(`accept` = 'application/json'),
                     httr::content_type('application/json'))

    if (response$status_code != 200) {
      results <- data.frame(title = NA, doi = NA, resource_title = NA)
      retnum <- 1000
    } else {
      t_resp <- httr::content(response, "parsed", encoding = "UTF-8")

      results <- lapply(t_resp, function(x){
        data.frame(title = x$title, doi = x$doi, resource_title = x$resource_title)
      })

      results <- do.call("rbind", results)

      retnum <- nrow(results)
    }

    allres[[k]] <- results

    k <- k + 1

  }

  return(allres)

}

fig_q1 <- get_figshare(query_fig)

# Bind all results
fig_results <- do.call("rbind", fig_q1)

write.csv(fig_results, paste0("source_lists/fig_", format(Sys.Date(), "%d%m%Y"),
".csv"),
          row.names = F)



# Get records from Zenodo using API connection
# Create a function to retrieve the records for a certain query
get_zenodo <- function(query){
  response <- httr::GET('https://zenodo.org/api/records',
                        query = list(q = query,
                                     size = 2000, page = 1))

  t_resp <- httr::content(response, "parsed", encoding = "UTF-8")

  results <- lapply(t_resp, function(x){
    data.frame(title = x$title, doi = x$doi)
  })

  results <- do.call("rbind", results)

  return(results)

}

zen_results <- get_zenodo("+access_right:open +resource_type.type:dataset
+title:marine +title:species")

write.csv(zen_results, paste0("source_lists/zen_", format(Sys.Date(), "%d%m%Y"),
".csv"),
          row.names = F)


# Get records from Dryad using API connection
library(httr)

# Create a function to retrieve the records for a certain query
get_dryad <- function(query, maxtry = 2000, addstop = T, verbose = T){

  off <- seq(1, ceiling(maxtry/100))

  retnum <- 100

  k <- 1

  allres <- list()

  while(retnum == 100 & k <= length(off)) {

    if (verbose) cat("Downloading page", k, "\n")

    response <- httr::GET('https://datadryad.org/api/v2/search',
                          query = list(q = query,
                                       per_page = 100, page = k))

    if (response$status_code != 200) {
      results <- data.frame(title = NA, doi = NA, resource_title = NA)
      retnum <- 100
    } else {
      t_resp <- httr::content(response, "parsed", encoding = "UTF-8")

      results <- lapply(t_resp$`_embedded`$`stash:datasets`, function(x){
        id <- x$identifier
        title <- x$title
        if (is.null(title)) {
          title <- "NOT FOUND"
        }
        data.frame(title = title, doi = id)
      })

      results <- do.call("rbind", results)

      retnum <- nrow(results)
    }

    allres[[k]] <- results

    k <- k + 1

    if (addstop) {
      Sys.sleep(5)
    }

  }

  return(allres)

}

dry_q1 <- get_dryad("marine species europe")
dry_q1 <- do.call("rbind", dry_q1)

dry_q2 <- get_dryad("marine species global")
dry_q2 <- do.call("rbind", dry_q2)
```

```
# Bind all results
dry_results <- rbind(dry_q1, dry_q2)

write.csv(dry_results, paste0("source_lists/dry_", format(Sys.Date(), "%d%m%Y"),
".csv"),
          row.names = F)
```

Compare title of datasets from data repositories with those from OBIS

```
### Data repositories
library(readxl)
library(tidyverse)
library(sf)

zen <- read.csv("source_lists/zen_23062023.csv")
dry <- read.csv("source_lists/dry_23062023.csv")
fig <- read.csv("source_lists/fig_23062023.csv")

full <- rbind(
  zen[,c("title", "doi")],
  dry[,c("title", "doi")],
  fig[,c("title", "doi")]
)

# Get OBIS datasets
# Open study area shapefile
starea <-
st_read("~/Research/mpa_europe/mpaeu_studyarea/data/shapefiles/mpa_europe_starea_v2
.shp")
starea <- st_bbox(starea)

# Download list of all obis datasets in the study area
datasets <- robis::dataset(
  geometry = st_as_text(st_geometry(st_as_sfc(st_bbox(starea))))
)

#### PYTHON IMPLEMENTATION
library(reticulate)
use_python("/usr/local/bin/python3")

fuz <- import("rapidfuzz")

sources <- tolower(full$title)

compare <- tolower(datasets$title)

match_frat <- match_title <- rep(NA, length(sources))

cli::cli_progress_bar("Running fuzzy matching...", total = length(sources))

for (s in 1:length(sources)) {
  frat <- rep(NA, length(compare))

  for (z in 1:length(compare)) {
    frat[z] <- fuz$fuzz$ratio(sources[s], compare[z])
  }

  match_title[s] <- compare[which.max(frat)]
  match_frat[s] <- max(frat, na.rm = T)

  cli::cli_progress_update()
}
```

```
cli::cli_progress_done()

cross_check <- full
cross_check$match_titles <- match_title
cross_check$fuzzy_ratio <- match_frat
#### END OF PYTHON IMPLEMENTATION

# Save for external edition
write_csv(cross_check, "final_lists/datarepo_datasets_comparison.csv")
```

Compare title of datasets from BioTIME with those from OBIS

```
# Load packages ----
library(tidyverse)
library(arrow)
library(sf)

# Open BioTIME dataset (downloaded using full export of BioTIME)
bt <- open_csv_dataset("source_lists/biotime/BioTIMEQuery_24_06_2021.csv")

# Open BioTIME metadata
meta <- read.csv("source_lists/biotime/BioTIMEMetadata_24_06_2021.csv")

# Open study area shapefile
starea <-
st_read("~/Research/mpa_europe/mpaeu_studyarea/data/shapefiles/mpa_europe_starea_v2
.shp")
starea <- st_bbox(starea)

# Get studies
bt_sel <- bt %>%
  select(STUDY_ID, LONGITUDE, LATITUDE) %>%
  collect()

# Filter to remove those out of study area and get the count of occurrences
bt_sel <- bt_sel %>%
  filter(LONGITUDE >= starea["xmin"] & LONGITUDE <= starea["xmax"]) %>%
  filter(LATITUDE >= starea["ymin"] & LATITUDE <= starea["ymax"]) %>%
  group_by(STUDY_ID) %>%
  count()

# Get the realm and name of studies
meta <- meta %>%
  select(STUDY_ID, TITLE, REALM, LICENSE)

# Join both
bt_sel_f <- left_join(bt_sel, meta, by = "STUDY_ID")

# Filter to retain only marine
bt_sel_f <- filter(bt_sel_f, REALM == "Marine")

# Remove those that have "obis" on the name
bt_sel_f <- filter(bt_sel_f, !grepl("obis", TITLE, ignore.case = T))

# Order by the number of records
bt_sel_f[order(bt_sel_f$n, decreasing = T),]

# Download list of all obis datasets in the study area
datasets <- robis::dataset(
  geometry = st_as_text(st_geometry(st_as_sfc(st_bbox(starea))))
)

head(datasets)

# Perform fuzzy matching to see if there are similar titles
```

```
#### PYTHON IMPLEMENTATION
library(reticulate)
use_python("/usr/local/bin/python3")

fuz <- import("rapidfuzz")

sources <- tolower(bt_sel_f$TITLE)

compare <- tolower(datasets$title)

match_frat <- match_title <- rep(NA, length(sources))

cli::cli_progress_bar("Running fuzzy matching...", total = length(sources))

for (s in 1:length(sources)) {
  frat <- rep(NA, length(compare))

  for (z in 1:length(compare)) {
    frat[z] <- fuz$fuzz$ratio(sources[s], compare[z])
  }

  match_title[s] <- compare[which.max(frat)]
  match_frat[s] <- max(frat, na.rm = T)

  cli::cli_progress_update()
}
cli::cli_progress_done()

cross_check <- bt_sel_f
cross_check$match_titles <- match_title
cross_check$fuzzy_ratio <- match_frat
#### END OF PYTHON IMPLEMENTATION

# Save for external edition
write_csv(cross_check, "biotime_datasets_comparison.csv")
```

Compare title of datasets from GBIF with those from OBIS

```
library(tidyverse)
library(sf)
library(arrow)

# Get OBIS datasets
# Open study area shapefile
starea <-
st_read("~/Research/mpa_europe/mpaeu_studyarea/data/shapefiles/mpa_europe_starea_v2
.shp")
starea_bbox <- st_bbox(starea)

# Download list of all obis datasets in the study area
datasets <- robis::dataset(
  geometry = st_as_text(st_geometry(st_as_sfc(st_bbox(starea))))
)

gbif <- read.csv("source_lists/gbif_splist_20230623.csv")

gbif_snapshot <- "s3://gbif-open-data-eu-central-1/occurrence/2023-06-
01/occurrence.parquet"

df <- open_dataset(gbif_snapshot)

gbif_datasets <- df %>%
  select(-mediatype,-issue) %>%
  filter(
    decimallongitude >= starea_bbox["xmin"] & decimallongitude <= 
starea_bbox["xmax"],
    decimallatitude >= starea_bbox["ymin"] & decimallatitude <= starea_bbox["ymax"]
  ) %>%
  filter(species %in% gbif$gbif_species) %>%
  group_by(datasetkey, species) %>%
  count() %>%
  collect()

gbif_data_count <- gbif_datasets %>%
  group_by(datasetkey) %>%
  summarise(total = sum(n)) %>%
  filter(total >= 10000)

gbif_data_count$title <- NA
gbif_data_count$description <- NA
gbif_data_count$license <- NA

for (i in 1:nrow(gbif_data_count)) {
  cat(i, "\n")
  data_info <- httr::GET(
    paste0("https://api.gbif.org/v1/dataset/", gbif_data_count$datasetkey[i])
  )

  if (data_info$status_code == 200) {
    t_resp <- httr::content(data_info, "parsed", encoding = "UTF-8")

    gbif_data_count$title[i] <- t_resp$title
    if ("description" %in% names(t_resp)) {
      gbif_data_count$description[i] <- t_resp$description
    }
    if ("license" %in% names(t_resp)) {
      gbif_data_count$license[i] <- t_resp$license
    }
  }

}

#### PYTHON IMPLEMENTATION
library(reticulate)
use_python("/usr/local/bin/python3")

fuz <- import("rapidfuzz")

sources <- tolower(gbif_data_count$title)

compare <- tolower(datasets$title)

match_frat <- match_title <- rep(NA, length(sources))

cli::cli_progress_bar("Running fuzzy matching...", total = length(sources))

for (s in 1:length(sources)) {
  frat <- rep(NA, length(compare))

  for (z in 1:length(compare)) {
    frat[z] <- fuz$fuzz$ratio(sources[s], compare[z])
  }

  match_title[s] <- compare[which.max(frat)]
  match_frat[s] <- max(frat, na.rm = T)

  cli::cli_progress_update()
}
cli::cli_progress_done()
```

```
cross_check <- gbif_data_count
cross_check$match_titles <- match_title
cross_check$fuzzy_ratio <- match_frat
#### END OF PYTHON IMPLEMENTATION

# Save for external edition
write_csv(cross_check, "final_lists/gbif_datasets_comparison.csv")
```

Compare title of literature works with those from OBIS datasets

```
library(readxl)
library(tidyverse)
library(sf)

wos <- lapply(list.files("source_lists/", pattern = "wos", full.names = T),
read_xls)

wos <- do.call("rbind", wos)

# Get OBIS datasets
# Open study area shapefile
starea <-
st_read("~/Research/mpa_europe/mpaeu_studyarea/data/shapefiles/mpa_europe_starea_v2
.shp")
starea <- st_bbox(starea)

# Download list of all obis datasets in the study area
datasets <- robis::dataset(
  geometry = st_as_text(st_geometry(st_as_sfc(st_bbox(starea))))
)

#### PYTHON IMPLEMENTATION
library(reticulate)
use_python("/usr/local/bin/python3")

fuz <- import("rapidfuzz")

sources <- tolower(wos$`Article Title`)

compare <- tolower(datasets$title)

match_frat <- match_title <- rep(NA, length(sources))

cli::cli_progress_bar("Running fuzzy matching...", total = length(sources))

for (s in 1:length(sources)) {
  frat <- rep(NA, length(compare))

  for (z in 1:length(compare)) {
    frat[z] <- fuz$fuzz$ratio(sources[s], compare[z])
  }
  match_title[s] <- compare[which.max(frat)]
  match_frat[s] <- max(frat, na.rm = T)
  cli::cli_progress_update()
}
cli::cli_progress_done()

cross_check <- data.frame(source_tiles = sources,
                          source_doi = wos$DOI)
cross_check$match_titles <- match_title
cross_check$fuzzy_ratio <- match_frat
#### END OF PYTHON IMPLEMENTATION

# Save for external edition
write_csv(cross_check, "final_lists/wos_datasets_comparison.csv")
```

Example of data ingestion using the **obisdi** package: Long term monitoring of fish abundances from coastal Skagerrak

Notes for readers:

Names marked with `` refers to R objects/code snippets

Code are differentiated by the font, but are also wrapped between ``` {r}   ```

Marine biodiversity data ingestion for OBIS (DwC translation)

Long term monitoring of fish abundances from coastal Skagerrak

This document describes how we map the checklist data to Darwin Core. The source file for this document can be found at https://github.com/iobis/mpaeu_di_fish_abund_skagerrak/blob/master/src/obisdi_general.Rmd.

~~~
```{r setup, include = FALSE}
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)
```
~~~

Load libraries:

~~~
```{r}
library(tidyverse)      # Data manipulation
library(obisdi)         # Tools for data ingestion for OBIS
library(here)           # Get paths (important!)
library(arrow)          # To deal with parquet files
library(worrms)         # Taxonomy checking
```
~~~

Read source data

This dataset was downloaded from the BioTIME (https://biotime.st-andrews.ac.uk/) database and can be found in this link: https://biotime.st-andrews.ac.uk/selectStudy.php?study=428. Two files are available on this link: the raw data and the metadata.

~~~
```{r}
# Get the path to data/raw
raw_path <- here("data", "raw")

list.files(raw_path)
```
~~~

Preprocessing

We first read the raw data:

~~~
```{r}
dataset <- read.csv(paste0(raw_path, "/raw_data_428.csv"))

str(dataset)

head(dataset)
```
~~~

By reading the `metadata` we can see that there is no biomass information, so we are interested in the abundance information:

~~~
```{r include=FALSE}
metadata <- read.csv(paste0(raw_path, "/metadata_428.csv"))
```
~~~

> *`r metadata$DATA_SOURCE[1]`*

Samples are collected from the Skagerrak coast and each station (we have coordinates) is a beach seine haul (5-700m2) where all species are counted. Unit of abundance = IndCountInt, Unit of biomass = NA

We select only the relevant columns. We also add a unique ID to each occurrence to be able to track back any transformation on data.

~~~
```{r}
dataset <- select(dataset, -BIOMAS, -SAMPLE_DESC)

dataset$uniqueID <- paste0("biotime_study248_", 1:nrow(dataset))
```
~~~

Darwin Core mapping

We will need to generate two files for the submission:

1. event - describe the sampling event

2. occurrence - the occurrence of species in an event

`event` table

We start by getting each unique event. Both the occurrences and the events table need to have the same eventID and locationID, so we still work with the `dataset` object.

~~~
```{r}
dataset <- dataset %>%
  arrange(-DAY, -MONTH, -YEAR)

dataset <- dataset %>%
  group_by(LATITUDE, LONGITUDE, PLOT, DAY, MONTH, YEAR) %>%
  mutate(dwc_eventID = paste0("skag_ev_", cur_group_id())) %>%
  group_by(LATITUDE, LONGITUDE, PLOT) %>%
  mutate(dwc_locationID = paste0("skag_lo_", cur_group_id()))
```
~~~

Correct the name of Longitude/Latitude fields, and date.

~~~
```{r}
dataset <- dataset %>%
  mutate(dwc_decimalLatitude = LATITUDE,
         dwc_decimalLongitude = LONGITUDE) %>%
  mutate(dwc_eventDate = paste(YEAR, MONTH, DAY, sep = "-")) %>%
  mutate(dwc_day = DAY, dwc_year = YEAR, dwc_month = MONTH)
```
~~~

Now we create a new object called `events` that will be our events table. We add other columns that are needed:

~~~
```{r}
events <- dataset %>%
  mutate(dwc_type = "Event",
         dwc_ownerInstitutionCode = "HAVFORSKNINGSINSTITUTTET",
         dwc_samplingProtocol = "Beach seine haul (40m long, mesh size 1.5cm),
nearshore (<15m depth) - all species counted.",
         dwc_sampleSizeUnit = "square metre",
         dwc_sampleSizeValue = "5-700",
         dwc_samplingEffort = "transect",
         dwc_country = "Norway",
         dwc_countryCode = "NO",
         dwc_locality = "Skagerrak",
         dwc_datasetID = paste0("biotime_study", metadata$STUDY_ID[1]),
         dwc_datasetName = "Long term monitoring of fish abundances from coastal
Skagerrak.")
```
~~~

Get only the unique events:

~~~
```{r}
events <- events %>%
  ungroup() %>%
  distinct(dwc_eventID, .keep_all = T)
```
~~~

`occurrence` table

We first create a new object that will have our occurrences:

~~~
```{r}
occurrences <- dataset
```
~~~

Then we add the needed columns:

~~~
```{r}
occurrences <- occurrences %>%
  mutate(dwc_type = "Event",
         dwc_ownerInstitutionCode = "HAVFORSKNINGSINSTITUTTET",
         dwc_occurrenceID = uniqueID,
         dwc_basisOfRecord = "HumanObservation",
         #dwc_scientificName = GENUS_SPECIES, # We will add it later, corrected
         dwc_individualCount = ABUNDANCE,
         dwc_organismQuantity = ABUNDANCE,
         dwc_organismQuantityType = "individuals")
```
~~~

We have no information if this dataset was checked with WoRMS for consistency of taxonomy. Thus, we do a check here:

~~~
```{r}
# Look into WoRMS for each name
name_checking <- wm_records_names(unique(occurrences$GENUS_SPECIES))

# Verify if there is any record with not valid names
table(unlist(lapply(name_checking, function(x){x$status})))

# We see that there are three unaccepted records. For these, we will extract the
correct names.
~~~

~~~
# We also check if there are records for which we were unable to find information:
no_match <- unique(occurrences$GENUS_SPECIES)[unlist(lapply(name_checking, nrow))
== 0]
no_match
```
~~~

We have some records that needs updated names and also 4 records for which there was no match. One of them is a unknown taxa and one is a non marine species[4], both which we will exclude, while for the other 2 we manually correct the information:

~~~
```{r}
name_checking <- lapply(name_checking, function(x){
  if (nrow(x) == 0) {
    x <- name_checking[[1]][1,]
    x[1,] <- NA
  }
  x
})

name_info <- bind_rows(name_checking)

name_info <- name_info %>%
  select(valid_name, valid_AphiaID, valid_authority,
              kingdom, phylum, class, order, family, genus, rank) %>%
  mutate(scientificName = valid_name,
         scientificNameAuthorship = valid_authority,
         scientificNameID = paste0("urn:lsid:marinespecies.org:taxname:",
valid_AphiaID),
         taxonRank = rank) %>%
  select(-valid_name, -valid_authority, -valid_AphiaID, -rank)

name_info$GENUS_SPECIES <- unique(occurrences$GENUS_SPECIES)

occurrences <- occurrences %>%
  filter(!GENUS_SPECIES %in% c("Manetyngel unknown", "Polioptila caerulea"))
name_info <- name_info %>%
  filter(!GENUS_SPECIES %in% c("Manetyngel unknown", "Polioptila caerulea"))

name_info[name_info$GENUS_SPECIES == "Sygnathus typhle",1:10] <-
  list("Animalia", "Chordata", "Teleostei", "Syngnathiformes", "Syngnathidae",
"Syngnathus", "Syngnathus typhle", "(Linnaeus, 1758)",
       "urn:lsid:marinespecies.org:taxname:127393", "Species")

name_info[name_info$GENUS_SPECIES == "Labrus bimaculatus",1:10] <-
  list("Animalia", "Chordata", "Teleostei", NA, "Labridae", "Labrus", "Labrus
mixtus", "(Linnaeus, 1758)",
       "urn:lsid:marinespecies.org:taxname:151501", "Species") # According to
FishBase

# Merge correct names
colnames(name_info)[1:10] <- paste0("dwc_", colnames(name_info)[1:10])

# Add originalNameUsage for the one we searched on FishBase
name_info$dwc_originalNameUsage <- NA
name_info$dwc_originalNameUsage[name_info$GENUS_SPECIES == "Labrus bimaculatus"] <-
  "Labrus bimaculatus"

occurrences <- left_join(occurrences, name_info, by = "GENUS_SPECIES")
~~~

[4] The species *Polioptila caerulea* returns a bird in all our searches. It's possible that this is a mistake when the name was written in the table.

~~~
occurrences$dwc_taxonRank <- tolower(occurrences$dwc_taxonRank)
```
~~~

Now we check if both tables have the same number of unique `locationID` and `eventID`:

~~~
```{r}
all.equal(unique(occurrences$dwc_eventID), unique(occurrences$dwc_eventID))

all.equal(unique(occurrences$dwc_locationID), unique(occurrences$dwc_locationID))
```
~~~

Post-processing

Remove unused columns and change names of those with **dwc_**:

~~~
```{r}
events <- events %>%
  mutate(dwc_eventRemarks = paste("Plot", PLOT)) %>%
  select(starts_with("dwc_"))

occurrences <- occurrences %>%
  ungroup() %>%
  select(starts_with("dwc_"))

# Change column names
colnames(events) <- str_remove(colnames(events), "dwc_")

colnames(occurrences) <- str_remove(colnames(occurrences), "dwc_")

# Remove columns from the occurrences table
occurrences <- occurrences %>%
  select(-day, -month, -year, -decimalLatitude, -decimalLongitude, -eventDate,
         -locationID)
```
~~~

We now have our final files:

~~~
```{r}
occurrences

events
```
~~~

Export final files

We export in `csv` format:

~~~
```{r}
proc_path <- here("data", "processed")

write_csv(events, paste0(proc_path, "/events.csv"))
write_csv(occurrences, paste0(proc_path, "/occurrences.csv"))
```
~~~

2023