BEAST database

Modelling of Biodiversity Lab (MOBI Lab)

2025-07-04

Table of contents

Chapter 1

Index

Chapter 2

Summary

2.1 Objective

We face an unprecedented threat from global alteration of nature and biodiversity, but we still lack rigorous estimates of how fast, where, and at which scales biodiversity changes. Studies report fragmented and seemingly contradictory results, suffer from mismatches in biodiversity metrics, mismatches in temporal and spatial grains, and are constrained by huge data gaps. Moreover, local loss and gain of biodiversity is decoupled from changes in countries or continents, with opposing directions at different scales being plausible. A quantitative synthesis that connects all this, and bridges the gaps, is needed. The objective of BEAST is to map and interpolate temporal biodiversity change in Europe, the US, and the world, across continuous space, time, and their grains, from locations as small as 1 m, to countries and continents, over the last ca 40 years, for birds, plants, and butterflies.To do this we will combine data from local time series with high-quality gridded atlas data from countries and continents. We will use a new cross-scale model to interpolate biodiversity change jointly across space and time, and across the data gaps. We will test if temporal change of diversity, distributions, and turnover can be estimated from: (i) static patterns of diversity and distributions, (ii) from data lacking temporal replication, (iii) from space-for-time substitution of spatial vs temporal species turnover, (iv) from spaceborne remotely sensed spectral diversity and turnover. These methods will enable integration of heterogeneous and messy biodiversity data, and they will improve estimates of change in data-poor regions of the global South. BEAST will deliver the first integrative statistical model revealing, for the first time, how multiple facets of biodiversity change across scales. It will show which regions, habitats, and biomes undergo the most pronounced change, which is critical for informed large-scale conservation policy.

2.2 Funding

HORIZON.1.1 - European Research Council (ERC) Main Programme

ERC-2021-COG - ERC CONSOLIDATOR GRANTS

2.3 MOBI Lab team roles (to be updated)

Petr Keil - principal investigator, Gabriel Ortega Solis - data manager, Carmen Soria - data acquisition, Kateřina Tschernosterová - data acquisition

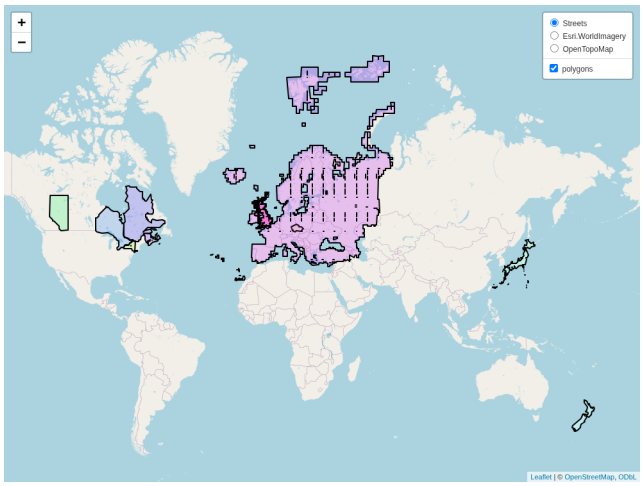
2.4 Regions covered

file:////tmp/RtmpbPIfOW/file9ff5185d22fc/widget9ff5567bffea.html screenshot completed

Your file couldn’t be accessed

It may have been moved, edited or deleted.

ERR_FILE_NOT_FOUND

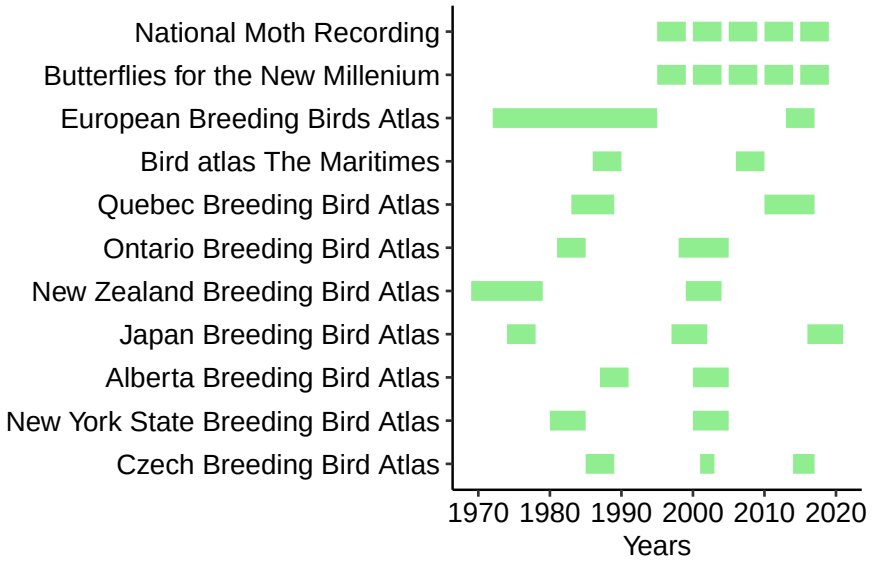


2.5 Current datasets

Dataset	Publisher	Taxa	dataset ID	Records count	Species count
Alberta Breeding Bird Atlas	The Federation of Alberta Naturalists	Aves	7	160197	371
Bird atlas	Bird Studies Canada	Aves	23	204566	228
The Maritimes Butterflies for the New Millenium	Butterfly Conservation	Butterflies	36	2914994	59
Czech Breeding Bird Atlas	Czech Society for Ornithology	Aves	5	205422	233
European Breeding Birds Atlas	European Bird Census Council	Aves	26	935392	622
Japan Breeding Bird Atlas	Japan Bird Research Association	Aves	13	150660	403
National Moth Recording	Butterfly Conservation	Moths	37	4976386	2360

Dataset	Publisher	Taxa	datasets	Records	Species count
New York State Breeding Bird Atlas	New York State Department of Environmental Conservation, New York Natural Heritage Program	Aves	6	744250	255
New Zealand Breeding Bird Atlas	The Ornithological Society of New Zealand	Aves	17	195682	276
Ontario Breeding Bird Atlas	The Federation of Ontario Naturalists	Aves	18	510170	289
Quebec Breeding Bird Atlas	Environment Canada's Québec Regional Office	Aves	20	362620	298

2.5.0.1 Sampling periods covered



2.6 Taxonomy

2.6.1 Birds

Naming follows HBW and BirdLife International (2024). Handbook of the Birds of the World and BirdLife International digital checklist of the birds of the world.

Version 9. Available at: http://datazone.birdlife.org/userfiles/file/Species/Taxonomy/HBW-BirdLife_Checklist_v9_Oct24.zip.

2.6.2 Butterflies and moths

Naming follows GBIF Secretariat (2022). GBIF Backbone Taxonomy. Checklist dataset https://doi.org/10.15468/56cv-7w97.

Chapter 3

Creating database

3.1 Purpose

Design a database for species occurrence and probability records extracted from atlases. The database should be reasonably small, compatible with multiple software, able to store spatial data, and flexible to accommodate different types of sampling efforts and methodologies. We chose PostgreSQL with PostGIS extension because it is open source, free, and it has extensive documentation.

3.2 PostgreSQL installation

Enable PostgreSQL repository for Ubuntu. It is neccesary to first install postgresql-common and then run the bash script it brings.

```
sudo apt install -y postgresql-common
sudo /usr/share/postgresql-common/pgdg/apt.postgresql.org.sh
```

Install packages postgresql, postgis and pgloader.

```
sudo apt install postgresql postgis postgresql-postgis* pgloader
```

It could be useful to also install pgAdmin. Instructions can be found here.

3.3 Create database and user roles

Identify as postgres (default user)

```
sudo -i -u postgres
```

Create a database that can be managed by the user atlasadmin.

```
createuser atlasadmin
createdb MOBI_atlases_v1 -O atlasadmin
```

Connect to the database.

```
psql -d MOBI_atlases_v1
```

Add a password for the user atlasadmin.

```
ALTER USER atlasadmin PASSWORD 'myPassword';
```

Enable PostGIS.

```
CREATE EXTENSION postgis;
```

Check PostGIS version

```
SELECT PostGIS_version();
```

Create a readonly role for users.

```
CREATE ROLE readonly WITH
  NOLOGIN
  NOSUPERUSER
  INHERIT
  NOCREATEDB
  NOCREATEROLE
  NOREPLICATION
  NOBYPASSRLS;
```

Exit database.

```
\q
```

Exit postgres user.

```
exit
```

3.4 Libraries

Load libraries to connect to the database.

```
pacman::p_load(RPostgres, DBI, askpass)
```

3.5 Connect R to database

Open an SSH tunnel. The following command works on Linux to open a tunnel in the background. The latter section of the code is the port redirection in the for `localport:localhost:remoteport`. Normally you would use `5432:localhost:5432` which redirects the remote port 5432 to the same port

number in your personal computer. However, the remote 5432 can be redirected to any local port available (5434 here).

```
screen -dmS postgresql_tunnel ssh ortega@srv-asus-fzp.science.fzp.czu.cz -L 5434:localhost:5432
```

Create an R object that will store the SQL connection.

```
con <- dbConnect(Postgres(),
  dbname = "MOBI_atlases_v1",
  host = "localhost",
  port = 5434,
  user = "atlasadmin",
  password = askpass()
)
```

Set the default SQL connection to "con" for every SQL cell.

```
knitr::opts_chunk$set(connection = con)
```

3.6 Tables

Variables were adopted from SPARSE 1.0, Darwin Core (DWC) and Humboldt core (HC).

3.6.1 Code books

1. **Licenses**

Information about data licenses including descriptions and links to external sources if available.

Table 3.1: Code book table for licensing information.

Variable	Description	Source	Type
licenseID	Primary key. Unique identifier.	BEAST	INTEGER
license	The legal description giving official permission to do something with the dataset.	DWC	TEXT
licenseDescription	Short verbal definition of the license type.	SPARSE	TEXT

Variable	Description	Source	Type
licenseURL	A full URL (web link) to the license type, if available.	SPARSE	TEXT

```
CREATE TABLE IF NOT EXISTS public."CB_license"
(
    "licenseID" integer NOT NULL,
    license text COLLATE pg_catalog."default" NOT NULL,
    "licenseDescription" text COLLATE pg_catalog."default" NOT NULL,
    "licenseURL" text COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "CB_license_PK" PRIMARY KEY ("licenseID")
)

TABLESPACE pg_default;
```

2. **Sampling effort (codes)**

   Descriptions and codes for sampling effort types adopted from Humboldt Core.

Table 3.2: Code book for sampling effort types.

Variable	Description	Source	Type
samplingEffortID	Primary key. Unique identifier.	BEAST	INTEGER
samplingEffortProtocol	A description of or reference (publication or URL) to the methods used to determine the sampling effort.	HC	
samplingEffortUnit	Units of sampling effort.	HC	TEXT

```
CREATE TABLE IF NOT EXISTS public."CB_sampling_effort"
(
    "samplingEffortID" integer NOT NULL,
    "samplingEffortProtocol" text COLLATE pg_catalog."default" NOT NULL,
    "samplingEffortUnit" text COLLATE pg_catalog."default" NOT NULL,
```

```
    CONSTRAINT "CB_sampling_effort_PK" PRIMARY KEY ("samplingEffortID"),
    CONSTRAINT "unique_samplingEffortProtocol" UNIQUE ("samplingEffortProtocol")
)
```

3. **Taxonomy**

   Includes taxonomic information for the taxa reported in every atlas dataset. We followed BirdLife 2024 for birds and GBIF 2022 for invertebrates.

Table 3.3: Code book for taxonomic information.

Variable	Description	Source	Type
scientificNameID	An identifier for the nomenclatural (not taxonomic) details of a scientific name.	DWC	INTEGER
scientificName	Binary latin scientific name. The lowest taxonomic rank included is species. Infraspecific names are reported in the column infraspecificEpithet	DWC	TEXT
scientificNameAuthorship	The authorship information for the scientificName formatted according to the conventions of the applicable nomenclaturalCode.	DWC	TEXT

Variable	Description	Source	Type
kingdom	The full scientific name of the kingdom in which the taxon is classified.	DWC	TEXT
phylum	The full scientific name of the phylum or division in which the taxon is classified.	DWC	TEXT
class	The full scientific name of the class in which the taxon is classified.	DWC	TEXT
order	The full scientific name of the order in which the taxon is classified.	DWC	TEXT
family	The full scientific name of the family in which the taxon is classified.	DWC	TEXT
genus	The full scientific name of the genus in which the taxon is classified.	DWC	TEXT
specificEpithet	The name of the first or species epithet of the scientificName.	DWC	TEXT

Variable	Description	Source	Type
infraspecificEpithet	The name of the lowest or terminal infraspecific epithet of the scientificName, excluding any rank designation.	DWC	TEXT
taxonRank	The taxonomic rank of the most specific name in the scientificName.	DWC	TEXT

```
CREATE TABLE IF NOT EXISTS public."CB_taxonomy"
(
    "scientificNameID" integer NOT NULL DEFAULT nextval('"CB_taxonomy_scientificNameID_seq"'::reg
    "scientificName" text COLLATE pg_catalog."default" NOT NULL,
    "scientificNameAuthorship" text COLLATE pg_catalog."default",
    kingdom text COLLATE pg_catalog."default" NOT NULL,
    phylum text COLLATE pg_catalog."default",
    class text COLLATE pg_catalog."default",
    "order" text COLLATE pg_catalog."default",
    family text COLLATE pg_catalog."default",
    genus text COLLATE pg_catalog."default",
    "specificEpithet" text COLLATE pg_catalog."default",
    "infraspecificEpithet" text COLLATE pg_catalog."default",
    "taxonRank" text COLLATE pg_catalog."default" NOT NULL,
    "taxonomicAuthority" text COLLATE pg_catalog."default" NOT NULL,
    "taxonomicSources" text COLLATE pg_catalog."default",
    CONSTRAINT "CB_taxonomy_PK" PRIMARY KEY ("scientificNameID"),
    CONSTRAINT "Unique_species_check" UNIQUE ("scientificName", kingdom, phylum, class)
)

TABLESPACE pg_default;
```

4. **Species equivalence table**

   Matches taxas provided by the data holders with accepted names according to BirdLife 2024 (birds) and GBIF 2022 (invertebrates).

Table 3.4: Code book for equivalences between verbatim organisms identification and accepted taxonomic information.

Variable	Description	Source	Type
verbatimIdentificationID	Part of composite primary key. Unique identifier for verbatim names per dataset.	BEAST	INTEGER
verbatimIdentification	A string representing the taxonomic identification written in the original record.	DWC	TEXT
datasetID	Part of composite primary key.		INTEGER
identificationReferences	A list (concatenated and separated) of references (publication, global unique identifier, URI).	DWC	TEXT
scientificNameID	Inherited from "CB_taxonomy".	DWC	INTEGER

```
CREATE TABLE IF NOT EXISTS public."CB_verbatim_name_equivalence"
(
    "verbatimIdentificationID" integer NOT NULL,
    "verbatimIdentification" text COLLATE pg_catalog."default" NOT NULL,
    "datasetID" integer NOT NULL,
    "scientificNameID" integer,
    CONSTRAINT "CB_verbatim_name_equivalence_PK" PRIMARY KEY ("verbatimIdentificationI
    CONSTRAINT unique_check_verbatim_equivalence UNIQUE ("verbatimIdentificationID", "
    CONSTRAINT "CB_verbatim_name_equivalence_CB_taxonomy_fk" FOREIGN KEY ("scientificNa
        REFERENCES public."CB_taxonomy" ("scientificNameID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
```

```
TABLESPACE pg_default;
```

5. **Models**

   Holds information about models used to produce occurrence probability estimations per dataset.

Table 3.5: Code book for occupancy models per atlas.

Variable	Description	Source	Type
modelID	Primary key. Unique identifier.	BEAST	INTEGER
modelName	Descriptive name of the model used.	BEAST	TEXT
predictorVariables	List of predictor variables used in the model.	BEAST	TEXT
bibliographicCitation	Bibliographic sources.	BEAST	TEXT

```
CREATE TABLE IF NOT EXISTS public."CB_model"
(
    "occurrenceModelID" integer NOT NULL,
    "modelName" text COLLATE pg_catalog."default",
    "predictorVariables" text COLLATE pg_catalog."default",
    "bibliographicCitation" text COLLATE pg_catalog."default",
    CONSTRAINT "CB_model_PK" PRIMARY KEY ("occurrenceModelID")
)

TABLESPACE pg_default;
```

3.6.2 Data tables

1. **Datasets**

   Contains information for data administration purposes including the names of publishers and providers of each atlas, bibliographic citation, and co-authors required and suggested.

Table 3.6: Datasets information.

Variable	Description	Source	Type
datasetID	Primary key. Unique identifier.	BEAST	INTEGER
datasetName	The name identifying the data set from which the record was derived. Title of atlas in most cases.	DWC	TEXT
datasetPublisher	The entity (person, organization or institution) making the dataset available.	HC	TEXT
datasetPublisherContact	Email contact to the entity making the dataset available.	SPARSE	TEXT
licenseID	An identifier from the codebook table CB_license defining publication license of the dataset source.	SPARSE	INTEGER
rightsHolder	The entity (person, organization or institution) that holds the rights to the dataset. Can be the datasetPublisher.	DWC	TEXT

Variable	Description	Source	Type
bibliographicCitation	A bibliographic reference for the resource as a statement indicating how this record should be cited (attributed) when used.	DWC	TEXT
citationIdentifier	A reference unique identifier of the associated document. DOI is preferred but a URI or ISBN number is adequate.	HC	TEXT
provider	Name of the person(s) or institution(s) providing the data.	BEAST	TEXT
shareable	YES: the dataset can be shared. NO: sharing is forbidden.	BEAST	TEXT
coauthorshipRequired	YES: offering co-authorship to selected data providers is mandatory. NO: co-authorship offers are not required.	BEAST	TEXT
coauthors	List of mandatory co-authors.	BEAST	TEXT

Variable	Description	Source	Type
coauthorshipSuggested	List of people within the MOBI team that acquired and processed the data and metadata.	BEAST	TEXT
isSamplingEffortReported	YES: some form of sampling effort is reported. NO: there is no sampling effort reported.	BEAST	TEXT
isOccurrenceProbability	YES: Available occurrence probability was mod-eled/reported. NO: no occurrence probability reported.	BEAST	TEXT
occurrenceModelID	Inherited from CB_model.	BEAST	INTEGER
recordFilterMeaning	Information about data quality provided by the data holders. Higher values mean more trusted records.	BEAST	TEXT

```
CREATE TABLE IF NOT EXISTS public."MOBI_dataset"
(
    "datasetID" integer NOT NULL,
    "datasetName" text COLLATE pg_catalog."default" NOT NULL,
    "datasetPublisher" text COLLATE pg_catalog."default" NOT NULL,
    "datasetPublisherContact" text COLLATE pg_catalog."default" NOT NULL,
    "licenseID" integer NOT NULL,
    "rightsHolder" text COLLATE pg_catalog."default",
    "bibliographicCitation" text COLLATE pg_catalog."default",
```

```
    "citationIdentifier" text COLLATE pg_catalog."default",
    provider text COLLATE pg_catalog."default",
    shareable text COLLATE pg_catalog."default",
    "coauthorshipRequired" text COLLATE pg_catalog."default",
    coauthors text COLLATE pg_catalog."default",
    "coauthorshipSuggested" text COLLATE pg_catalog."default",
    "isSamplingEffortReported" text COLLATE pg_catalog."default" NOT NULL,
    "isOccurrenceProbabilityAvailable" text COLLATE pg_catalog."default" NOT NULL,
    "occurrenceModelID" integer,
    "recordFilterMeaning" text COLLATE pg_catalog."default",
    taxa text COLLATE pg_catalog."default",
    CONSTRAINT "MOBI_dataset_PK" PRIMARY KEY ("datasetID"),
    CONSTRAINT "MOBI_dataset_CB_licenses_FK" FOREIGN KEY ("licenseID")
        REFERENCES public."CB_license" ("licenseID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT "MOBI_dataset_CB_model_FK" FOREIGN KEY ("occurrenceModelID")
        REFERENCES public."CB_model" ("occurrenceModelID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;
```

Create indexes

```
CREATE INDEX IF NOT EXISTS idx_mobi_dataset_datasetid
    ON public."MOBI_dataset" USING btree
    ("datasetID" ASC NULLS LAST)
    TABLESPACE pg_default;
```

```
CREATE INDEX IF NOT EXISTS idx_mobi_dataset_licenseid
    ON public."MOBI_dataset" USING btree
    ("licenseID" ASC NULLS LAST)
    TABLESPACE pg_default;
```

2. **Sites**

   Contains information about the spatial location of each sampling site.

Variable	Description	Source	Type
verbatimSiteID	Original site identificator provided by the data owners.	BEAST	TEXT

Variable	Description	Source	Type
verbatimSiteID	Original site identificator provided by the data owners.	BEAST	TEXT
datasetID	Dataset/atlas identifier. Inherited from MOBI_dataset	BEAST	INTEGER
footPrintSRS	The ellipsoid, geodetic datum, or spatial reference system (SRS). Standardized to WGS84 in our database.	DWC	TEXT
verbatimFootprintSRS	Original ellipsoid, geodetic datum, or spatial reference system (SRS) provided by the data owners. Analogous to verbatimSRS in DWC.	BEAST	TEXT
geometry	Original spatial object re-projected to WGS84.	BEAST	GEOMETRY
croppedGeometry	Original spatial object re-projected to WGS84 and cropped to landmasses and study area boundaries.	BEAST	GEOMETRY

: Sites information.

```
CREATE TABLE IF NOT EXISTS public."MOBI_site"
(
    "verbatimSiteID" text COLLATE pg_catalog."default" NOT NULL,
    "datasetID" integer NOT NULL,
    "footprintSRS" text COLLATE pg_catalog."default",
    "verbatimFootprintSRS" text COLLATE pg_catalog."default",
    geometry geometry,
    "croppedGeometry" geometry,
    CONSTRAINT "MOBI_siteFIN_pkey" PRIMARY KEY ("verbatimSiteID", "datasetID"),
    CONSTRAINT fk_site_dataset FOREIGN KEY ("datasetID")
        REFERENCES public."MOBI_dataset" ("datasetID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
) PARTITION BY LIST ("datasetID");
```

Create indexes

```
CREATE INDEX IF NOT EXISTS idx_mobi_site_croppedgeometry
    ON public."MOBI_site" USING gist
    ("croppedGeometry");
```

```
CREATE INDEX IF NOT EXISTS idx_mobi_site_datasetid
    ON public."MOBI_site" USING btree
    ("datasetID" ASC NULLS LAST);
```

```
CREATE INDEX IF NOT EXISTS idx_mobi_site_geometry
    ON public."MOBI_site" USING gist
    (geometry);
```

```
CREATE INDEX IF NOT EXISTS idx_mobi_site_verbatimsiteid
    ON public."MOBI_site" USING btree
    ("verbatimSiteID" COLLATE pg_catalog."default" ASC NULLS LAST)
    WITH (deduplicate_items=True);
```

3. **Scaling resolutions**

This table "translates" between the original cell resolution at the records level and every coarsening (scaling) resolution.

Variable	Description	Source	Type
verbatimSiteID	Original site identificator provided by the data owners. Inherited from MOBI_site.	BEAST	TEXT

Variable	Description	Source	Type
datasetID	Dataset/atlas identifier. Inherited from MOBI_dataset.	BEAST	INTEGER
scalingID	Grid cell spatial resolution. Filled with numbers in a doubling sequence of powers of 2 as follows: 1 (original resolution), 2 (2x2 cells grid), 4 (4x4)…	BEAST	INTEGER
siteID	New id assigned to cells in every resolution (scalingID). It allows forming groups of cells, mapping the original ones to the id of the bigger cells encompassing them.	BEAST	INTEGER

```
CREATE TABLE IF NOT EXISTS public."MOBI_scaling_table"
(
    "verbatimSiteID" text COLLATE pg_catalog."default" NOT NULL,
    "datasetID" integer NOT NULL,
    "scalingID" integer NOT NULL,
    "siteID" integer,
    CONSTRAINT "MOBI_scaling_table_PK" PRIMARY KEY ("verbatimSiteID", "datasetID", "sca
    CONSTRAINT "NEW_scaling_table_verbatimSiteID_datasetID_fkey" FOREIGN KEY ("verbatim
        REFERENCES public."MOBI_site_part1" ("verbatimSiteID", "datasetID") MATCH SIMPL
) PARTITION BY LIST ("datasetID");
```

Create indexes

```
CREATE INDEX IF NOT EXISTS idx_mobi_scaling_table_composite_all_but_verbatimsiteid
    ON public."MOBI_scaling_table" USING btree
    ("datasetID" ASC NULLS LAST, "scalingID" ASC NULLS LAST, "siteID" ASC NULLS LAST)
```

```
    INCLUDE("verbatimSiteID")
    WITH (deduplicate_items=True)
    TABLESPACE pg_default;
```

```
CREATE INDEX IF NOT EXISTS idx_mobi_scaling_table_verbatimsiteid
    ON public."MOBI_scaling_table" USING btree
    ("verbatimSiteID" COLLATE pg_catalog."default" ASC NULLS LAST)
    WITH (deduplicate_items=True)
    TABLESPACE pg_default;
```

4. **Records**

    1. ***Events***

    Sampling event attributes including 1 or multiple sampling effort values and start and end year of the sampling period per cell.

Table 3.9: Events information.

Variable	Description	Source	Type
verbatimSiteID	Part of composite primary key. Foreign key inherited from table “site”.	BEAST	TEXT
datasetID	Part of composite primary key. Foreign key inherited from table “site”.	BEAST	INTEGER
samplingPeriodID	Part of composite primary key.	BEAST	INTEGER
startYear	Initial year of the sampling period in every atlas.	BEAST	INTEGER
endYear	Initial year of the sampling period in every atlas.	BEAST	INTEGER
samplingEffortID	Foreign key inherited from table “CB_sampling_effort”.	BEAST	INTEGER
samplingEffortValue	The amount of effort expended for sampling during an Event.	HC	NUMERIC

Variable	Description	Source	Type
samplingEffort2ID	Foreign key inherited from table "CB_sampling_effort".	BEAST	INTEGER
samplingEffort2Value	The amount of effort expended for sampling during an Event.	HC	NUMERIC
samplingEffort3ID	Foreign key inherited from table "CB_sampling_effort".	BEAST	INTEGER
samplingEffort3Value	The amount of effort expended for sampling during an Event.	HC	NUMERIC

```
CREATE TABLE IF NOT EXISTS public."MOBI_event"
(
    "verbatimSiteID" text COLLATE pg_catalog."default" NOT NULL,
    "datasetID" integer NOT NULL,
    "samplingPeriodID" integer NOT NULL,
    "startYear" integer NOT NULL,
    "endYear" integer NOT NULL,
    "samplingEffortID" integer,
    "samplingEffortValue" numeric,
    remarks text COLLATE pg_catalog."default",
    "samplingEffort2ID" integer,
    "samplingEffort2Value" numeric,
    "samplingEffort3ID" integer,
    "samplingEffort3Value" numeric,
    CONSTRAINT "MOBI_event_PK" PRIMARY KEY ("verbatimSiteID", "datasetID", "samplingPe
    CONSTRAINT "MOBI_event2_CB_sampling_effort_FK" FOREIGN KEY ("samplingEffort2ID")
        REFERENCES public."CB_sampling_effort" ("samplingEffortID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
) PARTITION BY LIST ("datasetID");
```

Create indexes

```
CREATE INDEX "idx_event_verbatimSiteID" ON "MOBI_event" ("verbatimSiteID");
```

```
CREATE INDEX "idx_event_datasetID" ON "MOBI_event" ("datasetID");
```

```
CREATE INDEX "idx_event_samplingPeriodID" ON "MOBI_event" ("samplingPeriodID");
```

```
CREATE INDEX "idx_event_samplingEffortID" ON "MOBI_event" ("samplingEffortID");
```

2. ***Presences table***

Standardized original presence records handed over by the data providers.

Table 3.10: Presence information.

Variable	Description	Source	Type
verbatimIdentificationID	Numeric identificator of each verbatimIdentification. Foreign key inherited from table "CB_verbatim_name_equivalence".	BEAST	INTEGER
verbatimSiteID	Original identificator code for grid cells. Foreign key inherited from table "event".	BEAST	TEXT
datasetID	Numerical identificator of the atlas/dataset. Foreign key inherited from tables "event" and "CB_verbatim_name_equivalence".	BEAST	INTEGER
samplingPeriodID	Foreign key inherited from table "MOBI_event".	BEAST	INTEGER

Variable	Description	Source	Type
recordFilter	Maximum evidence of a species inhabiting a site. Filter values used by the data providers were re-coded as numbers from low (1) to high (>1) trust. NULL means no record filter was provided.	BEAST	INTEGER

```
CREATE TABLE "MOBI_presence" (
    "verbatimIdentificationID"  INTEGER NOT NULL,
    "verbatimSiteID"    TEXT NOT NULL,
    "datasetID" INTEGER NOT NULL,
    "samplingPeriodID"  INTEGER NOT NULL,
    "recordFilter" INTEGER,
    CONSTRAINT "MOBI_presence_PK" PRIMARY KEY("verbatimIdentificationID","verbatimSitel
    CONSTRAINT "MOBI_presence_MOBI_event_FK" FOREIGN KEY("verbatimSiteID","datasetID","
    CONSTRAINT "MOBI_presence_CB_verbatim_name_equivalence_FK" FOREIGN KEY("verbatimIde
    ) PARTITION BY LIST ("datasetID");
```

Create indexes.

```
CREATE INDEX "idx_presence_verbatimIdentificationID" ON "MOBI_presence" ("verbatimIdent
```

```
CREATE INDEX "idx_presence_verbatimSiteID" ON "MOBI_presence" ("verbatimSiteID");
```

```
CREATE INDEX "idx_presence_datasetID" ON "MOBI_presence" ("datasetID");
```

```
CREATE INDEX "idx_presence_samplingPeriodID" ON "MOBI_presence" ("samplingPeriodID");
```

```
3.  **Probabilities table**

    Standardized probability records produced by the data providers or MOBI team membe
```

Table 3.11: Probabilities information.

Variable	Description	Source	Type
presenceID	Primary key. Unique identifier.	BEAST	INTEGER
verbatimIdentificationID	Foreign key inherited from table "CB_verbatim_name_equivalence".	BEAST	INTEGER
siteID	Foreign key inherited from table "event".	BEAST	TEXT
scalingID	Foreign key inherited from table "event".	BEAST	INTEGER
datasetID	Foreign key inherited from tables "event" and "CB_verbatim_name_equivalence".	BEAST	INTEGER
samplingPeriodID	Foreign key inherited from table "event".	BEAST	INTEGER
probability	Occurrence probability.	BEAST	NUMERIC
modelID	Foreign key inherited from table "CB_model".	BEAST	INTEGER

```
CREATE TABLE "MOBI_probability" (
    "verbatimIdentificationID"  INTEGER NOT NULL,
    "verbatimSiteID"    TEXT NOT NULL,
    "datasetID" INTEGER NOT NULL,
    "samplingPeriodID"  INTEGER NOT NULL,
    "probability" NUMERIC NOT NULL,
    CONSTRAINT "MOBI_probability_PK" PRIMARY KEY("verbatimIdentificationID","verbatimSiteID","dat
    CONSTRAINT "MOBI_probability_CB_verbatim_name_equivalence_FK" FOREIGN KEY("verbatimIdentifica
    ) PARTITION BY LIST ("datasetID");
```

Create indexes

```
CREATE INDEX "idx_probability_verbatimIdentificationID" ON "MOBI_probability" ("verbatimIdentific
```

```
CREATE INDEX "idx_probability_verbatimSiteID" ON "MOBI_probability" ("verbatimSiteID");
```

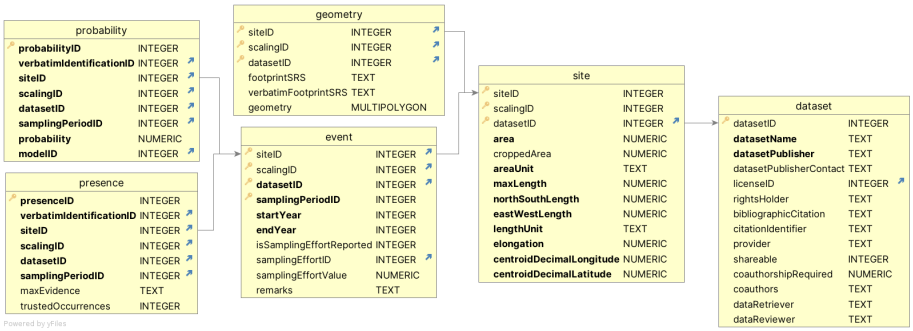
```
CREATE INDEX "idx_probability_datasetID" ON "MOBI_probability" ("datasetID");
```

```
CREATE INDEX "idx_probability_samplingPeriodID" ON "MOBI_probability" ("samplingPeriodI
```

3.7 Close the database connection

```
dbDisconnect(con)
```

3.8 ER diagram



(a)

(b)

Figure 3.1: Entity relationship (ER) diagram. It Shows from left to right the most dependent tables to the least dependent ones. The key symbol represents a primary key. Multiple key symbols in a table represent composite primary keys. Blue arrows at the right border of each table mark foreign keys inherited from a table upstream. Variable types are shown in capital letters to the right. Bold letters are columns defined as NOT NULL. The links between main tables are shown in Figure 3.1a while the whole ER diagram is in Figure 3.1b.

Chapter 4

Birds of Japan

4.1 MOBI Lab team roles

Responsibilities	Name	Date (MM/YY)
Data acquisition	Carmen Soria	03/23
Metadata preparation	Kateřina Tschernosterová	04/25
	Gabriel Ortega	
Data standardization	Gabriel Ortega	08/24
Data processing	Gabriel Ortega	01/25
	Carmen Soria	

4.2 Data providers, metadata and co-authorship

Always check the updated information here.

The original documents folder including books, publications or email exchanges are here.

4.3 Data description

- Existing temporal replications of atlases
  - BBA1 - sampling period 1974 - 1978
  - BBA2 - sampling period 1997 - 2002
  - BBA3 - sampling period 2016 - 2021
- The data holder can only provide comparable field survey data for the last two atlases (BBA2 and BBA3).

- The first atlas combines results from fieldwork and questionnaires, where people were asked to report bird sightings in specific locations. Questionnaire data cannot be separated from the fieldwork data in this case.
- The second atlas also combines mix of fieldwork and questionnaire data. However, fieldwork data can be extracted from a separate fieldwork file, and questionnaire-based records can be assumed as those not matching the fieldwork data (but will require confirmation with M. Ueta).
- The third atlas clearly separates fieldwork data from questionnaire data.
- Questionnaires for all three atlases also includes literature records where species, location, time, and reproductive status were clearly established and fell within the atlas sampling period timeframe.

4.3.1 Licensing and access

- The dataset is classified as "restricted" in the BEAST database because there is no explicit license declared. However, it is publicly accessible without restrictions for general use.
- For detailed information or specific permissions regarding data usage, contact: Japan Bird Research Association (info@bird-research.jp), Mutsuyuki Ueta (mj-ueta@bird-research.jp).

4.3.2 Sampling methodology

Fieldwork

- Each survey was conducted by 1–2 people in most locations.
- Observations were made at two fixed points for 30 minutes each, followed by a 3 km walk at a speed of approximately 2 km/h.
- The total survey time per location was about 2.5 hours.

Questionnaire surveys

- Casual observations of species before and after the field survey days.
- Records of observations at arbitrary points.
- Literature information.
- Records registered in the bird observation database of the Japan Bird Research Association.

Data Gaps

- There are cells in northern Japan (Kuril Islands) without bird data.

4.3.3 Sampling effort

The effort is filled with NA in the database because no specific value per cell was reported. However, it could be considered standard if you only use fieldwork data.

4.4 Input Data

4.4.1 Compressed original data folder

- `20km_mesh.zip`: 20km grid cells files.
- `Birds_Atlas_Japan_1-2_Data.xlsx`: file contains records for the first and second atlas. No information on the source of records.
- `Birds_Atlas_Japan_2-3_Field survey data.xlsx`: file contains field records for the second and third atlas.
- `all_data2010.xlsx`: file contains all records for the second and third atlas. Includes a column that is used to separate field surveys and questionnaires. No information on whether the questionnaire data comes from literature.
- `modified_grid.gpkg`: made by MOBI team. Includes the position of grid cells across different spatial resolutions.
- `other files.zip`: set of records downloaded from Japan's Ministry of Environment.
- `past data.xlsx`: same as `Birds_Atlas_Japan_1-2_Data.xlsx`.
- `species_list.csv`: species list provided in Japanese.
- `survey data.xlsx`: same as `Birds_Atlas_Japan_2-3_Field survey data.xlsx`.

4.4.2 Docs folder

- `Birds_Atlas_Japan_1-3_Readme.txt`: Notes on data quality and limitations. Mostly imported into the present document.
- `Birds_Atlas_Japan_3_Book.pdf`: third atlas report. Includes distribution maps.
- `Birds_Atlas_Japan_3_Data paper.pdf`: description of the third atlas dataset.
- `Birds_Atlas_Japan_3_Documentation.pdf`: survey instructions.
- `ENG_Birds_Atlas_Japan_3_Data paper.pdf`: translation of `Birds_Atlas_Japan_3_Data paper.pdf`.
- `ENG_Birds_Atlas_Japan_3_Documentation.pdf`: translation of `Birds_Atlas_Japan_3_Documentation.pdf`.
- `Gmail - Japanese bird atlas data enquiry.pdf`: email exchange.

4.4.3 Checked data folder

- `Birds_atlas_Japan_beast_data.rds`: occurrence records from all atlases of Japan.
- `Birds_atlas_Japan_cells_corr.rds`: correspondence table assigning the original cell grids to coarser grid resolutions.
- `Birds_atlas_Japan_grid.gpkg`: spatial geometries.
- `Birds_atlas_Japan_spnames_INFO.csv`: species list including old matches to multiple taxonomic sources.

4.5 Data standardization and processing comments

Sources of records were ranked as: "NULL" = No information, 1 = questionnaires, and 2 = field surveys. Higher numbers mean more trustable sources of information. The decision to consider fieldwork as a more trusted source is based on the certainty of identifying a list of species on a given location and date under a standardized methodology.

The third atlas was extracted from `all_data2010.xlsx`.

The second atlas data was extracted from `Birds_Atlas_Japan_1-2_Data.xlsx`. The data were matched with `Birds_Atlas_Japan_2-3_Field survey data.xlsx` to get a subset of records from the second atlas corresponding to fieldwork surveys.

4.6 Libraries

```
pacman::p_load(
  sf, terra, data.table, tidyverse, tidyterra, tidytable, knitr,
  tictoc, RPostgres, DBI, dbplyr, parallel,
  geodata
)
```

4.6.1 Preparation

The data were first prepared as rds and csv files before designing our database. This section starts from there but the whole data pipeline from original data to database input will be added during the next database iteration.

```
# Variables
seldata <- "data/Birds_Japan/Birds_atlas_Japan"
datasetID <- 13
licenseID <- 1
verbatimFootprintSRS <- "epsg:4326"

# Data processing
dataset <- readRDS(paste0(seldata, "_beast_data.rds")) %>%
  as.data.table() %>%
  mutate(questionnaires = recode(questionnaires, `0` = NA, `1` = 2)) %>%
  mutate(recordFilter = recode(rowSums(across(c(field_surveys, questionnaires)), na.rm
  mutate(recordFilter = ifelse(is.na(field_surveys) & is.na(questionnaires), NA, recor

spnamesinfo <- read.csv(paste0(seldata, "_spnames_INFO.csv")) %>%
  as.data.table() %>%
  mutate(cellID = str_split(cellID, pattern = ",")) %>%
```

```
  unnest(cellID) %>%
  select(cellID, name_in_data, original_verbatim_name) %>%
  mutate(cellID = cellID) %>%
  unique()

grid <- st_layers(paste0(seldata, "_grid.gpkg"))$name %>%
  sapply(., USE.NAMES = T, simplify = F, function(x) {
    st_read(paste0(seldata, "_grid.gpkg"), layer = x)
  })

corrs <- readRDS(paste0(seldata, "_cells_corr.rds")) %>%
  as.data.table() %>%
  pivot_longer(-cellID, names_to = "cell_grouping", values_to = "cell_label") %>%
  mutate(cell_grouping = as.numeric(cell_grouping))

full_data <- left_join(dataset, corrs, by = c("cell_grouping", "cell_label"), relationship = "mar
  left_join(., filter(spnamesinfo, !is.na(cellID)), by = c("cellID" = "cellID", "verbatim_name" =
  mutate(verbatim_name = ifelse(is.na(original_verbatim_name), verbatim_name, original_verbatim_n
  select(-original_verbatim_name) %>%
  unique() %>%
  mutate(
    datasetID = datasetID,
    licenseID = licenseID,
    verbatimIdentificationID = dense_rank(verbatim_name),
    samplingPeriod = dense_rank(start_year),
    # Double check original effort columns before running this
    effort = NA,
    samp_effort_type = NA
  ) %>%
  unique()

gc()
```

4.6.2 Database connection

```
source("scripts/dbcon.R")
```

4.6.3 List tables in database

```
dbListTables(con) %>%
  purrr::keep(., ~ grepl("^CB|^MOBI", .)) %>%
  kable(col.names = "Tables")
```

4.6.4 Write code books

4.6.4.1 CB_license

```
table <- "CB_license" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  licenseID = licenseID,
  license = "Closed",
  licenseDescription = "Publicly available on a data portal. No license specified.",
  licenseURL = "none"
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check the table

```
tbl(con, table) %>% kable(align = "c")
```

4.6.4.2 CB_sampling_effort

```
table <- "CB_sampling_effort" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  samplingEffortID = "",
```

```
  samplingEffortProtocol = "",
  samplingEffortUnit = ""
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table.

```
tbl(con, table) %>% kable(align = "c")
```

4.6.4.3 CB_model

```
table <- "CB_model" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  occurrenceModelID = "",
  modelName = "",
  predictorVariables = "",
  bibliographicCitation = ""
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table

```
tbl(con, table) %>% kable(align = "c")
```

4.6.4.4 CB_taxonomy

```
table <- "CB_taxonomy" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  scientificNameID = "",
  scientificName = "",
  scientificNameAuthorship = "",
  kingdom = "",
  phylum = "",
  class = "",
  family = "",
  order = "",
  genus = "",
  specificEpitet = "",
  infraspecificEpitet = "",
  taxonRank = ""
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

4.6.4.5 CB_verbatim_name_equivalence

```
table <- "CB_verbatim_name_equivalence" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  verbatimIdentificationID = full_data$verbatimIdentificationID,
  verbatimIdentification = full_data$verbatim_name,
  datasetID = full_data$datasetID,
  identificationReferences = "",
  scientificNameID = NA
) %>% unique()
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

4.6.5 Write dataset tables

4.6.5.1 Dataset

```
table <- "MOBI_dataset" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  datasetID = datasetID,
  datasetName = "Japan Breeding Bird Atlas",
  datasetPublisher = "Japan Bird Research Association",
  datasetPublisherContact = "mj-ueta@bird-research.jp | info@bird-research.jp",
```

```
  licenseID = licenseID,
  rightsHolder = "Japan Bird Research Association",
  bibliographicCitation = "Mutsuyuki Ueta, Shingo Uemura, Hayama Seiji, Teppei Ara, Shi
  citationIdentifier = "",
  provider = "Japan Bird Research Association",
  shareable = "NO",
  coauthorshipRequired = "YES",
  coauthors = "Mutsuyuki Ueta - mj-ueta@bird-research.jp",
  coauthorshipSuggested = "Carmen Soria - carmendianasoria@gmail.com | Kateřina Tscherr
  isSamplingEffortReported = "NO",
  isOccurrenceProbabilityAvailable = "NO",
  recordFilterMeaning = "NULL = No information, 1 = Field surveys, 2 = Questionnaires"
)
```

Write into the database

```
try({
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>
```

4.6.5.2 Site

```
table <- "MOBI_site" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_site_13" PARTITION OF "MOBI_site"
FOR VALUES IN (13);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  siteID = full_data$cell_label,
  scalingID = full_data$cell_grouping,
  datasetID = full_data$datasetID,
```

```
  area = full_data$area,
  croppedArea = full_data$area_cropped,
  areaUnit = "km2",
  maxLength = NA,
  northSouthLength = NA,
  eastWestLength = NA,
  lengthUnit = "km",
  centroidDecimalLongitude = full_data$cell_long,
  centroidDecimalLatitude = full_data$cell_lat,
  samplingRepetitions = full_data$repeated
) %>% unique()
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(
  con,
  sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))
) %>%
  head(n = 20) %>%
  kable(align = "c")
```

4.6.5.3 Geometry

```
table <- "MOBI_geometry" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_geometry_13" PARTITION OF "MOBI_geometry"
FOR VALUES IN (13);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
```

```
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- grid %>%
  dplyr::bind_rows() %>%
  ungroup() %>%
  rename(
    siteID = cell_label,
    scalingID = cell_grouping,
    geometry = geom
  ) %>%
  mutate(
    siteID = as.integer(siteID),
    scalingID = as.integer(scalingID),
    datasetID = as.integer(datasetID),
    footprintSRS = "epsg:4326",
    verbatimFootprintSRS = verbatimFootprintSRS
  ) %>%
  select(
    ., siteID, scalingID, datasetID,
    footprintSRS, verbatimFootprintSRS, geometry
  )

data_table$geometry <- st_cast(data_table$geometry, "MULTIPOLYGON")
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Write records
  st_write(obj = data_table, dsn = con, layer = table, append = T)
  # Remove the data_table from the environment
  # rm(data_table)
})
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>
  head() %>%
  kable(align = "c")
```

4.6.6 Write helper tables

4.6.6.1 Scaling table

This intermediate table enables the joins of records data to the different resolutions of sites.

```
table <- "MOBI_scaling_table" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_scaling_table_13" PARTITION OF "MOBI_scaling_table"
FOR VALUES IN (13);
```

Check table colnames.

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

```
data_table <- data.table(
  siteID = full_data$cell_label,
  scalingID = full_data$cell_grouping,
  datasetID = datasetID,
  verbatimSiteID = full_data$cellID
) %>% unique()
```

Write into database.

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

4.6.7 Write records tables

4.6.7.1 Event

```
table <- "MOBI_event" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_event_13" PARTITION OF "MOBI_event"
FOR VALUES IN (13);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- filter(full_data, cell_grouping == 1) %>%
  mutate(
    verbatimSiteID = cellID,
    datasetID = datasetID,
    samplingPeriodID = samplingPeriod,
    startYear = start_year,
    endYear = end_year,
    samplingEffortID = samp_effort_type,
    samplingEffortValue = effort
  ) %>%
  select(
    verbatimSiteID,
    datasetID,
    samplingPeriodID,
    startYear,
    endYear,
    samplingEffortID,
    samplingEffortValue
  ) %>%
  unique()

# if (nrow(unique(data_table)) != nrow(eff)) {
#   stop("Stopping execution due to row number mismatch")
# } else {
#   print("OK")
# }
```

```
data_table <- filter(data_table, !is.na(verbatimSiteID))
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

4.6.7.2 Presence

```
table <- "MOBI_presence" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_presence_13" PARTITION OF "MOBI_presence"
FOR VALUES IN (13);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- filter(full_data, cell_grouping == 1) %>%
  mutate(
    verbatimIdentificationID = verbatimIdentificationID,
    verbatimSiteID = cellID,
    datasetID = datasetID,
    samplingPeriodID = samplingPeriod
  ) %>%
  select(
    verbatimIdentificationID,
```

```
    verbatimSiteID,
    datasetID,
    samplingPeriodID,
    recordFilter
  ) %>%
  unique()
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>
  head() %>%
  kable(align = "c")
```

4.6.7.3 Probability

```
table <- "MOBI_probability" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_probability_13" PARTITION OF "MOBI_probability"
FOR VALUES IN (13);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  verbatimIdentificationID = "",
  verbatimSiteID = "",
  datasetID = "",
```

```
  samplingPeriodID = "",
  probability = ""
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

Chapter 5

Birds of Alberta

5.1 MOBI Lab team roles

Responsibilities	Name	Date (MM/YY)
Data acquisition	Carmen Soria	08/23
Metadata preparation	Kateřina Tschernosterová	
Data standardization	Kateřina Tschernosterová	08/24
Data processing	Gabriel Ortega	
	Carmen Soria	

5.2 Data providers metadata and co-authorships

Always check the updated information here, in the table Atlases status.

The original documents folder including books, publications or email exchanges are here.

5.3 Data description

- Existing temporal replications of atlases
  - BBA1 - sampling period 1987 - 1992
  - BBA2 - sampling period 2000 - 2005
- Observation data by grid cell, by species and by observation card (year, sampling event and data collector).
- Presence is reported, no data on abundance or absence.

- Original data detail: Scientific species name, Common species name, Survey area identifier, Decimal latitude and longitude of survey area, Year (month, day) collected, Collector number, Sampling event identifier, Effort in hours (BBA2 only, not for all events), Breeding bird atlas code.
- Data was downloaded through NatureCounts website - https://naturecounts.ca/nc/default/explore.jsp#download.
  - Downloaded dataset with BBA1 data - Alberta Breeding Bird Atlas (1987-1992) [NatureCounts (hosted datasets)].
  - Downloaded dataset with BBA2 data - Alberta Breeding Bird Atlas (2000-2005) [NatureCounts (hosted datasets)].
  - Another available dataset that was not used - Alberta Bird Records [NatureCounts (hosted datasets)].
- Nature Alberta (formerly known as Federation of Alberta Naturalists), Birding in Alberta website - https://naturealberta.ca/birding/.
- All the newsletters of the Breeding Bird Atlas of Alberta: Update Project were downloaded from website - https://andrelegris.com/environmental-science/.
- Nature Alberta should have additional bird records database available on eBird, NatureCounts or on request (Database Administrator) - records should be from the first Breeding Bird Atlas project, from individual naturalists, from the May Species Counts, from the Christmas Birds Count and all other data submitted through the Alberta Birdlist program.
- Related regional websites devoted to birds are listed in BBA2_Newsletter_1 pdf, p.8.

5.3.1 Licensing and access

- The dataset is classified as "restricted" in the database because there is no explicit license declared. However, it is publicly accessible.

5.3.2 Sampling methodology

- Data was collected in sampling squares by volunteer atlasers.
- Sampling atlas squares are 10 km by 10 km (100 km^2), there are 6623 Universal Transverse Mercator (UTM) squares in the province.
- The overall goal of the first atlas project was to survey 25% of the survey squares in the settled area of the province and 10% of the squares in the more remote areas.
- BBA1 survey data was collected during the February-August period over five field seasons (1987 to 1991) by 943 volunteer atlasers. These volunteers spent about 40,000 hours in the field, collecting 122,400 records of bird breeding activity. Of the 6623 atlas squares in Alberta, 2206 (33%) received adequate survey coverage.
- The goal of the Atlas Update Project is to survey not only more squares than were surveyed in the initial Atlas but to ensure that all priority squares identified in the first atlas are also surveyed.

- BBA2 Remote Areas Program - implemented to increase our understanding of bird populations in the northern regions of Alberta. The program is led by researchers from the University of Alberta, the program will intensively survey specific atlas squares with the goal of analyzing the relationships between bird distribution and abundance, and associated habitats and landscape patterns (BBA2_Newsletter_1 pdf, p.9).
- More Atlas Grid System related information can be found in BBA2_Newsletter_1 pdf, p.13.
- A detailed description of how the atlas surveys are done can be found in the NORAC Atlasser's Handbook: https://naturecounts.ca/norac/index.jsp?targetpg=handbook.

5.3.3 Sampling effort

- In BBA1 data there is no information on duration of sampling events, therefore sampling effort is reported as number of sampling events per grid cell.
- For BBA2 there is sampling duration in hours reported for some sampling events - but for some events the value of duration hours is negative and for some events the value is missing. The value of sampling event duration can be used, but only with restrictions.
- Therefore sampling effort for BBA2 is also reported as number of sampling events per grid cell.

5.4 Input Data

5.4.1 Compressed original data folder

- `Birds_Atlas_Alberta_1-2_Column names.txt`: text file with the list of original data column names.
- `Birds_Atlas_Alberta_1_Data.txt`: text file with data for BBA1 1984-1989.
- `Birds_Atlas_Alberta_2_Data.txt`: text file with data for BBA2 2000-2005, corrected.
- `Birds_Atlas_Alberta_2_Data original.txt`: text file with original data for BBA2 2000-2005, with several mistakes (listed below in Data standardization part).
- `alberta_gridmap.shp`: original grid shapefile.

5.4.2 Docs folder

- `Birds_Atlas_Alberta_Readme Fixing Data.txt`: readme file, text was copied to the section Data standardization and processing comments of this document.
- `Birds_Atlas_Alberta_2_Newsletter_1.pdf`: the newsletters of the Breeding Bird Atlas of Alberta, Update Project (April 2002).

- `Birds_Atlas_Alberta_2_Newsletter_2.pdf`: the newsletters of the Breeding Bird Atlas of Alberta, Update Project (February 2003).
- `Birds_Atlas_Alberta_2_Newsletter_3.pdf`: the newsletters of the Breeding Bird Atlas of Alberta, Update Project (July 2003).
- `Birds_Atlas_Alberta_2_Newsletter_4.pdf`: the newsletters of the Breeding Bird Atlas of Alberta, Update Project (March 2004).
- `Birds_Atlas_Alberta_2_Newsletter_5.pdf`: the newsletters of the Breeding Bird Atlas of Alberta, Update Project (July 2004).

5.4.3 Checked data folder

- `effFin.csv`: final_grid.gpkg: integrated effort from both sampling periods.
- `grids_correspondence_table.csv`: equivalence table to map the original grid cells to coarser resolutions.
- `modified_grid.gpkg`: wide version of correspondence table including spatial geometries.
- `occFin.csv`: consolidated occurrence records for both sampling periods.
- `samp_cells.csv`: list of cells with records or effort reported.

5.5 Data standardization and processing comments

README written by Carmen Soria - Modifications to Bird_Atlas_Alberta_2_Data.txt (the original is named as Bird_Atlas_Alberta_2_Data original.txt). There were mistakes in the original data file of the second Atlas of Alberta, which have been FIXED. - Some rows had misplaced columns. This is because in the txt they had put two localities while keeping the number of tab separations. I fixed that by separating the info of those localities by a space and not a tab. The rows that had this were: 79720, 79721, 79769, 79770, 162864, 162865, 166830, 176022, 184714, 184715, 184716, 186566, 186567, 186568, 199957, 199958, 199959, 199960, 199961, 199963, 199964. - Row 132159 had an issue in which in one cell it said “many”. That made the cell contain info for around 50 rows. Changed ’ to ” and it was fixed. - There were 1333 rows that didn’t have info from the column locality onward. This is because the text for that locality had #. Deleted the # and it was fixed.

!!! IMPORTANT !!! the sampling effort in hours is not complete.

5.6 Libraries

```
pacman::p_load(
  sf, terra, tidyverse, tidyterra, knitr,
  tictoc, RPostgres, DBI, dbplyr, parallel,
```

```
  geodata
)
```

5.7 Grid processing

```
unzip data/Birds_Alberta/Original_data.zip -d /tmp/
```

```
glimpse(vect("/tmp/Original_data/alberta_gridmap.shp"))
```

```
grid <- st_read("/tmp/Original_data/alberta_gridmap.shp") %>%
  st_make_valid() %>%
  select(utm_sqr, utm_stn, utm_nrt, utm_zon) %>%
  rename(cellID = utm_sqr) %>%
  filter(!is.na(cellID))

length(grid$cellID) == length(unique(grid$cellID))
```

```
grid_grouping <- function(grid, col1, col2, num) {
  grid <- st_drop_geometry(grid)
  mincol1 <- min(grid[[col1]])
  maxcol1 <- max(grid[[col1]])
  mincol2 <- min(grid[[col2]])
  maxcol2 <- max(grid[[col2]])
  breaksA <- seq(mincol1, maxcol1, num)
  breaksB <- seq(mincol2, maxcol2, num)
  if (length(breaksA) > 1) {
    A <- cut(grid[[col1]], breaks = breaksA, right = F)
  } else {
    A <- 1
  }
  if (length(breaksB) > 1) {
    B <- cut(grid[[col2]], breaks = breaksB, right = F)
  } else {
    B <- 1
  }
  res <- paste0(grid$utm_zon, "_", A, B)
  return(res)
}
```

Create labels for rescaling

```
for (dist in c(2, 4, 8, 16, 32, 64, 128)) {
  num <- 10000 * dist
  grid[[paste0(dist)]] <- grid_grouping(grid, col1 = "utm_stn", col2 = "utm_nrt", num = num)
}
```

```
grid$`1` <- dense_rank(grid$cellID)

grid <- grid %>% mutate(across(matches("^\\d"), dense_rank))

st_write(grid, "/tmp/alberta.gpkg", delete_dsn = T)
```

The grid was manually checked in QGIS to merge small cells with their neighbors. Small or even linear cells occur in borders of UTM zones in the original grid. We also reassigned some cells to groupings in every scale so that they end up mostly aligned and closer to the expected standard sizes per scale. The resulting grid was saved as modified_grid.gpkg.

```
qgis /tmp/alberta.gpkg
```

We have different percentages of overlap between cells and real country borders. Therefore, we should have a way to correct the areas.

```
grid <- st_read("data/Birds_Alberta/Checked_data/modified_grid.gpkg") %>%
  mutate(across(matches("^X\\d"), dense_rank))

adm_boundary <- gadm("CAN", path = "/tmp/") %>%
  select(ISO_1) %>%
  filter(ISO_1 == "CA-AB")

int <- terra::intersect(select(vect(grid), cellID), adm_boundary)

int <- group_by(int, cellID) %>% summarise()

int$cropped_area <- expanse(int, unit = "km")

grid <- as.data.frame(int) %>%
  left_join(grid, ., by = "cellID")
```

Calculate cell area with and without correction by landmasses or country borders.

```
grid$cell_area <- expanse(vect(grid), unit = "km")
grid$cell_area_proportion <- grid$cropped_area / grid$cell_area %>% round(., digits = 1
```

```
# Pivot cells to a longer format for easier use in models.
final_grids <- grid %>%
  pivot_longer(matches("^X\\d"),
    names_to = "cell_grouping",
    values_to = "cell_label"
  ) %>%
  mutate(cell_grouping = str_remove_all(cell_grouping, "X") %>%
    as.numeric())
```

```
final_grids <- split(final_grids, final_grids$cell_grouping)
```

Aggregate cells and summarize the area.

```
final_grids <- final_grids %>%
  lapply(., function(x) {
    res <- x %>%
      group_by(cell_grouping, cell_label) %>%
      summarise(
        area = sum(cell_area, na.rm = T),
        area_cropped = sum(cropped_area, na.rm = T)
      ) %>%
      ungroup()
  })
```

Functions to add additional variables of cell shapes

```
# Function to get cell shape attributes

fdistances <- function(x, type = NULL) {
  x <- vect(x)
  pol <- as.data.frame(crds(as.points(ext(project(x, "epsg:4326")))))
  if (type == "ew") {
    sw <- pol %>%
      summarise(x = min(x), y = min(y)) %>%
      as.matrix()
    se <- pol %>%
      summarise(x = max(x), y = min(y)) %>%
      as.matrix()
    res <- distance(sw, se, lonlat = T)[[1]] / 1000
  }
  if (type == "ns") {
    sw <- pol %>%
      summarise(x = min(x), y = min(y)) %>%
      as.matrix()
    nw <- pol %>%
      summarise(x = min(x), y = max(y)) %>%
      as.matrix()
    res <- distance(sw, nw, lonlat = T)[[1]] / 1000
  }
  if (type == "max") {
    res <- distance(pol, lonlat = T) %>% max()
    res <- res / 1000
  }
  res <- as.numeric(res)
  return(res)
```

```
}
```

Add cell max length, distance south-north, and distance east-west.

```
tic()

add_cell_lengths <- function(x) {
  res <- x
  list <- res %>%
    select(cell_label)
  list <- terra::split(list, list$cell_label)

  res$cell_max_length <- lapply(list, function(.x) fdistances(.x, type = "max")) %>%
    do.call(rbind, .) %>%
    as.numeric()

  res$cell_ns_length <- lapply(list, function(.x) fdistances(.x, type = "ns")) %>%
    do.call(rbind, .) %>%
    as.numeric()

  res$cell_ew_length <- lapply(list, function(.x) fdistances(.x, type = "ew")) %>%
    do.call(rbind, .) %>%
    as.numeric()

  res <- st_transform(res, st_crs(x))
  return(res)
}

# Create a cluster
cl <- makeCluster(getOption("cl.cores", detectCores() - 1))

# Load necessary packages on each worker
clusterEvalQ(cl, {
  library(dplyr)
  library(terra)
  library(sf)
})

# Export the function to the cluster
clusterExport(cl, varlist = "fdistances")

# Use parSapply with the cluster
final_grids <- parSapply(cl, final_grids, add_cell_lengths, simplify = FALSE, USE.NAMES

# Stop the cluster
stopCluster(cl)
```

```
toc()
```

```
plet(vect(final_grids[[2]]))
```

Export results.

```
grid %>%
  st_drop_geometry() %>%
  select(matches("^cell|^X")) %>%
  write_csv(., "data/Birds_Alberta/Checked_data/grids_correspondence_table.csv")

file.remove("data/Birds_Alberta/Checked_data/final_grid.gpkg")

for (x in 1:length(final_grids)) {
  st_write(final_grids[[x]], layer = paste0("grid_", x), dsn = "data/Birds_Alberta/Checked_data/f
}
```

5.8 Data processing

```
files <- c("/tmp/Original_data/Birds_Atlas_Alberta_1_Data.txt", "/tmp/Original_data/Birds_Atlas_A

columns <- c("ScientificName", "DecimalLongitude", "DecimalLatitude", "YearCollected", "MonthColl

data <- lapply(files, function(x) {
  read_delim(x) %>%
    select(any_of(columns)) %>%
    unique()
}) %>%
  do.call(rbind, .) %>%
  mutate(index = paste0(DecimalLongitude, "_", DecimalLatitude)) %>%
  rename(verbatim_name = ScientificName) %>%
  mutate(CollectorNumber = max(CollectorNumber) + 1)

index <- select(data, index, DecimalLongitude, DecimalLatitude) %>%
  unique() %>%
  st_as_sf(., coords = c("DecimalLongitude", "DecimalLatitude"), crs = 4326)

index <- select(grid, cellID) %>%
  st_join(., index) %>%
  st_drop_geometry()

data <- left_join(data, index, by = "index") %>%
  select(-index, -DecimalLongitude, -DecimalLatitude) %>%
  unique() %>%
```

```
  mutate(
    start_year = case_when(
      YearCollected >= 1987 & YearCollected <= 1991 ~ 1987,
      YearCollected >= 2000 & YearCollected <= 2005 ~ 2000
    ),
    end_year = case_when(
      YearCollected >= 1987 & YearCollected <= 1991 ~ 1991,
      YearCollected >= 2000 & YearCollected <= 2005 ~ 2005
    )
  )

occ <- select(data, cellID, verbatim_name, start_year, end_year) %>% unique()

write_csv(occ, "data/Birds_Alberta/Checked_data/occFin.csv")

eff <- select(data, -verbatim_name) %>%
  group_by(across(-CollectorNumber)) %>%
  summarise(effort = n_distinct(CollectorNumber)) %>%
  ungroup() %>%
  group_by(cellID, start_year, end_year) %>%
  summarise(effort = sum(effort)) %>%
  ungroup()

write_csv(eff, "data/Birds_Alberta/Checked_data/effFin.csv")

samp_cells <- select(occ, cellID, start_year) %>%
  rbind(., select(eff, cellID, start_year)) %>%
  unique() %>%
  mutate(
    start_year = dense_rank(start_year),
    value = 1
  ) %>%
  pivot_wider(
    names_from = start_year,
    values_from = value,
    values_fill = 0
  ) %>%
  mutate(repeated = rowSums(select(., matches("^\\d")))) %>%
  select(cellID, repeated)

write_csv(samp_cells, "data/Birds_Alberta/Checked_data/samp_cells.csv")
```

5.9 Adding records to the database

5.9.1 Data

Import the processed data and grid.

```
data <- read_csv("data/Birds_Alberta/Checked_data/occFin.csv") %>%
  mutate(
    verbatimIdentificationID = dense_rank(verbatim_name),
    samplingPeriodID = dense_rank(start_year)
  ) %>%
  rename() %>%
  filter(!is.na(cellID))

sp_names <- data %>%
  select(verbatimIdentificationID, verbatim_name) %>%
  unique()

corrs <- read_csv("data/Birds_Alberta/Checked_data/grids_correspondence_table.csv") %>%
  mutate(across(matches("^X\\d"), dense_rank)) %>%
  select(cellID, matches("^X\\d"))

eff <- read_csv("data/Birds_Alberta/Checked_data/effFin.csv") %>%
  mutate(samplingPeriodID = dense_rank(start_year))

scaling_table <- corrs %>%
  pivot_longer(-cellID, names_to = "scalingID", values_to = "siteID") %>%
  mutate(scalingID = str_remove_all(scalingID, "^X") %>%
    as.numeric()) %>%
  ungroup()

samp_cells <- read_csv("data/Birds_Alberta/Checked_data/samp_cells.csv") %>%
  left_join(., corrs, by = "cellID") %>%
  pivot_longer(-c(cellID, repeated), names_to = "cell_grouping", values_to = "cell_label") %>%
  mutate(cell_grouping = str_remove_all(cell_grouping, "^X") %>%
    as.numeric()) %>%
  na.omit() %>%
  group_by(cell_grouping, cell_label) %>%
  summarise(repeated = max(repeated)) %>%
  ungroup()

grid <- st_layers("data/Birds_Alberta/Checked_data/final_grid.gpkg")$name %>%
  mclapply(., function(x) {
    st_read("data/Birds_Alberta/Checked_data/final_grid.gpkg",
      layer = x, quiet = T
    )
```

```
  }, mc.cores = 10) %>%
  do.call(rbind, .) %>%
  ungroup() %>%
  st_make_valid() %>%
  mutate(coords = st_centroid(geom)) %>%
  mutate(
    cell_lat = st_coordinates(coords)[, 2],
    cell_long = st_coordinates(coords)[, 1]
  )


grid_table <- st_drop_geometry(grid) %>%
  left_join(., samp_cells)
```

5.9.1.1 Basic double check

```
# Check that all cellID in data are also in the corrs cellID
if (length(setdiff(unique(data$cellID), unique(corrs$cellID))) == 0) {
  print("OK")
} else {
  print("FAIL")
}
# Check that number of rows in grid and corrs are the same
nrow(filter(grid, cell_grouping == 1)) == nrow(corrs)

# Check that years and sampling periods are the same in data and eff
nrow(setdiff(
  unique(select(data, start_year, samplingPeriodID)),
  unique(select(eff, start_year, samplingPeriodID))
))

# Check uniqueness of scaling_table
nrow(unique(scaling_table)) == nrow(scaling_table)

# Check that all cellID in data are also in the grid (corrs)
nrow(setdiff(
  unique(data$cellID),
  unique(corrs$cellID)
))

# Check that all cellID in eff are also in the grid (corrs)
nrow(setdiff(
  unique(eff$cellID),
  unique(corrs$cellID)
))
```

```
# RESULTS SHOULD BE: OK, TRUE, 0, TRUE, NULL, NULL
```

5.9.1.2 Atlas variables

```
datasetID <- 7
effortID <- 1
licenseID <- 1
shareable <- "YES"
verbatimFootprintSRS <- "epsg:4326"
```

All cells match between the different dataset files.

```
final_data <- inner_join(data, corrs, by = "cellID")
```

5.9.2 Database connection

```
source("scripts/dbcon.R")
```

5.9.3 List tables in database

```
dbListTables(con) %>%
  purrr::keep(., ~ grepl("^CB|^MOBI", .)) %>%
  kable(col.names = "Tables")
```

5.9.4 Write code books

5.9.4.1 CB_license

```
table <- "CB_license" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  licenseID = licenseID,
  license = "Open",
  licenseDescription = "Publicly available on a data portal",
```

```
  licenseURL = "none"
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check the table

```
tbl(con, table) %>% kable(align = "c")
```

5.9.4.2 CB_sampling_effort

```
table <- "CB_sampling_effort" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  samplingEffortID = effortID,
  samplingEffortProtocol = "Visits per gridcell",
  samplingEffortUnit = "Visits"
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table.

```
tbl(con, table) %>% kable(align = "c")
```

5.9.4.3 CB_model

```
table <- "CB_model" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  occurrenceModelID = "",
  modelName = "",
  predictorVariables = "",
  bibliographicCitation = ""
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table

```
tbl(con, table) %>% kable(align = "c")
```

5.9.4.4 CB_taxonomy

```
table <- "CB_taxonomy" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  scientificNameID = "",
  scientificName = "",
  scientificNameAuthorship = "",
  kingdom = "",
  phylum = "",
  class = "",
  family = "",
```

```
  order = "",
  genus = "",
  specificEpitet = "",
  infraspecificEpitet = "",
  taxonRank = ""
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

5.9.4.5 CB_verbatim_name_equivalence

```
table <- "CB_verbatim_name_equivalence" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimIdentificationID = sp_names$verbatimIdentificationID,
  verbatimIdentification = sp_names$verbatim_name,
  datasetID = datasetID,
  identificationReferences = "",
  scientificNameID = NA
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

5.9.5 Write dataset tables

5.9.5.1 Dataset

```
table <- "MOBI_dataset" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  datasetID = datasetID,
  datasetName = "Alberta Breeding Bird Atlas",
  datasetPublisher = "The Federation of Alberta Naturalists",
  datasetPublisherContact = "info@fanweb.ca/info@naturealberta.ca",
  licenseID = licenseID,
  rightsHolder = "The Federation of Alberta Naturalists",
  bibliographicCitation = "BBA1 data: Nature Alberta. 2018. Alberta Breeding Bird Atlas (1987-199
  citationIdentifier = "",
  provider = "The Federation of Alberta Naturalists",
  shareable = "NO",
  coauthorshipRequired = "NO",
  coauthors = "",
  coauthorshipSuggested = "Carmen Soria - carmendianasoria@gmail.com; Kateřina Tschernosterová -
  isSamplingEffortReported = "YES",
  isOccurrenceProbabilityAvailable = "NO"
)
```

Write into the database

```
try({
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>
```

5.9.5.2 Site

```
table <- "MOBI_site" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_site_7" PARTITION OF "MOBI_site"
FOR VALUES IN (7);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  siteID = grid_table$cell_label,
  scalingID = grid_table$cell_grouping,
  datasetID = datasetID,
  area = grid_table$area,
  croppedArea = grid_table$area_cropped,
  areaUnit = "km2",
  maxLength = grid_table$cell_max_length,
  northSouthLength = grid_table$cell_ns_length,
  eastWestLength = grid_table$cell_ew_length,
  lengthUnit = "km",
  centroidDecimalLongitude = grid_table$cell_long,
  centroidDecimalLatitude = grid_table$cell_lat,
  samplingRepetitions = grid_table$repeated
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
```

```
rm(data_table)
```

Check table.

```
tbl(
  con,
  sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))
) %>%
  head(n = 20) %>%
  kable(align = "c")
```

5.9.5.3 Geometry

```
table <- "MOBI_geometry" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_geometry_7" PARTITION OF "MOBI_geometry"
FOR VALUES IN (7);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- grid %>%
  ungroup() %>%
  rename(
    siteID = cell_label,
    scalingID = cell_grouping,
    geometry = geom
  ) %>%
  mutate(
    siteID = as.integer(siteID),
    scalingID = as.integer(scalingID),
    datasetID = as.integer(datasetID),
    footprintSRS = "epsg:4326",
    verbatimFootprintSRS = verbatimFootprintSRS
  ) %>%
  select(
    ., siteID, scalingID, datasetID,
    footprintSRS, verbatimFootprintSRS, geometry
  )
```

```
data_table$geometry <- st_cast(data_table$geometry, "MULTIPOLYGON")
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Write records
  st_write(obj = data_table, dsn = con, layer = table, append = T)
  # Remove the data_table from the environment
  rm(data_table)
})
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>
  head() %>%
  kable(align = "c")
```

5.9.6 Write helper tables

5.9.6.1 Scaling table

This intermediate table enables the joins of records data to the different resolutions of sites.

```
table <- "MOBI_scaling_table" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_scaling_table_7" PARTITION OF "MOBI_scaling_table"
FOR VALUES IN (7);
```

Check table colnames.

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

```
data_table <- data.frame(
  siteID = scaling_table$siteID,
  scalingID = scaling_table$scalingID,
  datasetID = datasetID,
  verbatimSiteID = scaling_table$cellID
)
```

Write into database.

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

5.9.7 Write records tables

5.9.7.1 Event

```
table <- "MOBI_event" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_event_7" PARTITION OF "MOBI_event"
FOR VALUES IN (7);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimSiteID = eff$cellID,
  datasetID = datasetID,
  samplingPeriodID = eff$samplingPeriodID,
  startYear = eff$start_year,
  endYear = eff$end_year,
  samplingEffortID = effortID,
  samplingEffortValue = eff$effort
)
```

```
if (nrow(unique(data_table)) != nrow(eff)) {
  stop("Stopping execution due to row number mismatch")
} else {
  print("OK")
}

data_table <- filter(data_table, !is.na(verbatimSiteID))
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>
  head() %>%
  kable(align = "c")
```

5.9.7.2 Presence

```
table <- "MOBI_presence" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_presence_7" PARTITION OF "MOBI_presence"
FOR VALUES IN (7);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimIdentificationID = final_data$verbatimIdentificationID,
```

```
  verbatimSiteID = final_data$cellID,
  datasetID = datasetID,
  samplingPeriodID = final_data$samplingPeriodID,
  recordFilter = NA
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

5.9.7.3 Probability

```
table <- "MOBI_probability" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_probability_7" PARTITION OF "MOBI_probability"
FOR VALUES IN (7);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimIdentificationID = "",
  verbatimSiteID = "",
  datasetID = "",
  samplingPeriodID = "",
```

```
  probability = ""
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

5.10 Finish session

```
sessionInfo()
```

5.10.1 Clean up workspace

```
dbDisconnect(con)
rm(list = ls())
gc()
.rs.restartR()
```

Chapter 6

Birds of Ontario

6.1 MOBI team roles

Responsibilities	Name	Date (MM/YY)
Data acquisition		08/23
Metadata preparation	Kateřina Tschernosterová	07/24
Data standardization	Kateřina Tschernosterová	12/24
Data processing	Gabriel Ortega	

6.2 Data providers metadata and co-authorships

Always check the updated information here.

The original documents folder including books, publications or email exchanges are here.

6.3 Data description

- Existing temporal replications of atlases
  - BBA1 - sampling period 1981-1985
  - BBA2 - sampling period 2001-2005
- Observation data by grid cell, by species and by observation card (year, sampling event and data collector).
- Presence is reported, no data on abundance or absence.

- Original data detail - `Scientific species name`, `Common species name`, `Survey area identifier`, `Decimal latitude and longitude of survey area`, `Year (month, day) collected`, `Collector number`, `Sampling event identifier`, `Effort in hours` (BBA2 only, not for all events), `Breeding bird atlas code`.
- Data collection for the Ontario Breeding Bird Atlas 3 began on January 1, 2021 for next 5 years.
- Data was downloaded through NatureCounts website - https://naturecounts.ca/nc/default/explore.jsp#download and https://naturecounts.ca/nc/onatlas/main.jsp.
  - Downloaded dataset with BBA1 data - `Ontario Breeding Bird Atlas (1981-1985): raw breeding evidence` [Ontario Breeding Bird Atlas].
  - Downloaded dataset with BBA2 data - `Ontario Breeding Bird Atlas (2001-2005): raw breeding evidence` [Ontario Breeding Bird Atlas].
  - Another available datasets that were not used - `Ontario Breeding Bird Atlas (2001-2005): point count data` [Ontario Breeding Bird Atlas] and `Ontario Bird Feeder Survey` [Project FeederWatch].
  - Available datasets with BBA3 data (currently in progress): `Ontario Breeding Bird Atlas (2021-2025), checklist data` [Ontario Breeding Bird Atlas] and `Ontario Breeding Bird Atlas (2021-2025), point count data` [Ontario Breeding Bird Atlas].
- Data request together with available datasets and data release conditions are here.
- BBA1 website (with online BBA1 book) - https://www.birdsontario.org/jsp/firstatlas.jsp.
- BBA1 online book - https://www.birdsontario.org/jsp/atlasbook.jsp?pg=toc.
- BBA2 website - https://www.birdsontario.org/atlas-2/.
- BBA3 website - https://www.birdsontario.org/about/.

6.3.1 Sampling methodology

- Data was collected by volunteers following the guide for participants (`Birds_Atlas_Ontario_2_Guide`).
- The goal was to provide adequate coverage of every 10-km square in southern Ontario, and of every 100-km block in northern Ontario. Data was recorded on a 10-km basis wherever possible in the north (`BBA_2_Guide`, p.3 and 6).

- The minimum effort required was 20 hours of accumulated surveys per square over the 5 years. At the same time, it was important to ensure that all habitats within the square were properly covered (`BBA_2_Guide`, p.12).
- Atlasers were also given the option of collecting information on the relative abundance of species in their square by doing `Point Counts` survey (`BBA_2_Guide`, p.3).
- If `Point Counts` was done (optional), the time doing them could be included in total hours of coverage (`BBA_2_Guide`, p.12).

6.3.2 Sampling effort

- Sampling duration in hours is reported for both atlases, but for some sampling events the value is missing. Therefore the value of sampling event duration can be used, but only with restrictions.
- Effort for both atlases is also reported as number of sampling events per grid square.
- Sampling effort was not equally distributed among all squares during the surveys, the number of sampling events is different for each sampling square.
- In the northern part of Ontario, where travel is more restricted, the maps are probably a less accurate account of the species actually breeding there.
- During the first atlas, squares in southern Ontario averaged over 50 hours of coverage (`BBA_2_Guide`, p.12).
- The source of following data and tables is BBA1 online book, chapter Final coverage - Atlaser Effort, p.21.

The number of atlasers per year

	1981	1982	1983	**1984**	**1985**
Southern Ontario	379	643	706	629	722
All of Ontario	433	730	828	720	811

Total number of atlasers contributing data = 1352.

The number of field work hours per year

	1981	**1982**	**1983**	**1984**	**1985**
Southern Ontario	11,400	20,702	21,894	23,003	28,523
All of Ontario	13,436	23,446	26,543	27,827	32,627
Hours per atlaser	31.0	32.1	32.1	38.6	40.2

Total number of field work hours in Southern Ontario = 105,522. Total number of field work hours in Ontario = 123,879. Average of field work hours per atlaser over the 5 years period = 92.

6.3.3 Compressed data folder

- `Birds_Atlas_Ontario_1-2_Column names.txt` - text file with list of columns used for raw data.
- `Birds_Atlas_Ontario_1_Data_raw breeding evidence.txt` - raw data for BBA1.
- `Birds_Atlas_Ontario_2_Data_raw breeding evidence.txt` - raw data for BBA2.
- `naturecounts_ontario shapefile` - grid extracted from Naturecounts.
- `obba_squares shapefile` - alternative grid file.

6.3.4 Docs folder

- `Birds_Atlas_Ontario_1-2_Data policy.html` - policy on use and publication of data.
- `Birds_Atlas_Ontario_2_Book.pdf` - pdf version of the Atlas.
- `Birds_Atlas_Ontario_2_Guide.pdf` - guide for participants (April 2001).
- `Birds_Atlas_Ontario_3_Guide.pdf` - instructions for general atlasing (April 2021), the third atlas was not published yet, so we did not include information from the guide.

6.3.5 Checked data folder

- `Bird_Atlas_Ontario_1_OriginalData.csv` - raw data for BBA1 in csv format.
- `Bird_Atlas_Ontario_1_1981-1985_eff_data.xlsx` - standardized effort data for BBA1.
- `Bird_Atlas_Ontario_1_1981-1985_occ_data.xlsx` - standardized occurrence data for BBA1.
- `Bird_Atlas_Ontario_2_OriginalData.csv` - raw data for BBA2 in csv format.
- `Bird_Atlas_Ontario_2_1998-2005_eff_data.xlsx` - standardized effort data for BBA2.
- `Bird_Atlas_Ontario_2_1998-2005_occ_data.xlsx` - standardized occurrence data for BBA2.
- `Bird_Atlas_Ontario_CheckedData_working.xlsx` - effort and occurrence data standardization, working file.
- `XX_Birds_of_Ontario_data description.qmd` - Quarto document, contains metadata, processing notes and codes, part of BEAST database documentation.

6.4 Data standardization and processing comments

!!! IMPORTANT !!! the sampling effort in hours is not complete.

6.5 Libraries

```
pacman::p_load(
    sf, terra, tidyverse, lubridate, janitor,
    tictoc, RPostgres, DBI, dbplyr, parallel,
    geodata, data.table, tidytable, tidyverse, tidyterra, knitr
)
```

6.6 Grid processing

```
rm -rf /tmp/Original_data2
unzip data/Birds_Ontario/Original_data2.zip -d /tmp/Original_data2

rm -rf /tmp/Original_data
unzip data/Birds_Ontario/Original_data.zip -d /tmp/Original_data
```

```
newgrid <- vect("/tmp/Original_data2/obba_squares.shp")
oldgrid <- vect("/tmp/Original_data/obba_squares.shp")
oldgrid2 <- vect("/tmp/Original_data/obba_squares.shp")
```

```
grid <- vect("/tmp/Original_data/obba_squares.shp") %>%
    project(., "epsg:4326")

grid <- makeValid(grid)

plot(grid)
```

The grid was manually checked in QGIS to merge small cells with their neighbors. Small or even linear cells occur in borders of UTM zones in the original grid. We also reassigned some cells to groupings in every scale so that they end up mostly aligned and closer to the expected standard sizes per scale. The resulting table was saved as a csv file.

```
corrs <- read.csv("data/Birds_Ontario/Checked_data/scaling_table.csv") %>%
pivot_wider(
    names_from = scalingID,
    values_from = siteID
)

head(corrs)
```

Compare ids in the grid and corr

```
setdiff(unique(corrs$verbatimSiteID), unique(grid$SQUARE_ID))

not_found <- setdiff(unique(grid$SQUARE_ID), unique(corrs$verbatimSiteID))
```

```
grid <- filter(grid, SQUARE_ID %in% not_found)
```

6.7 Data processing

```
data_files <- list.files("/tmp/Original_data/", pattern = "*raw_naturecounts_data.txt"

system(str_glue("sed 's/, /,/g' {data_files[[1]]} > /tmp/cleaned1.txt"))

system(str_glue("sed 's/, /,/g' {data_files[[2]]} > /tmp/cleaned2.txt"))

system(str_glue("head -n 2 /tmp/cleaned1.txt"))

system(str_glue("head -n 2 /tmp/cleaned2.txt"))
```

Data double checked in Libreoffice

```
problems1 <- problems(read_csv("data/Birds_Ontario/Checked_data/cleaned1.csv", col_name

problems2 <- problems(read_csv("data/Birds_Ontario/Checked_data/cleaned2.csv", col_name

data1 <- read_csv("data/Birds_Ontario/Checked_data/cleaned1.csv", col_names = TRUE) %>%
        mutate(across(where(is.character), str_squish)) %>%
        unique() %>%
        as.data.table()

data2 <- read_csv("data/Birds_Ontario/Checked_data/cleaned2.csv", col_names = TRUE) %>%
        mutate(across(where(is.character), str_squish)) %>%
        unique() %>%
        as.data.table()

data <- bind_rows(data1, data2) %>%
        as.data.table() %>%
        mutate(SurveyAreaIdentifier = toupper(SurveyAreaIdentifier))

effort <- read_csv("data/Birds_Ontario/Checked_data/effort.csv", col_names = TRUE) %>%
        mutate(across(where(is.character), str_squish)) %>%
        unique() %>%
        as.data.table()
```

6.8 Adding records to the database

6.8.1 Atlas variables

```
datasetID <- 18
effortID <- 1
licenseID <- 1
shareable <- "YES"
verbatimFootprintSRS <- "epsg:4326"
```

All cells match between the different dataset files.

```
final_data <- filter(data, SurveyAreaIdentifier %in% corrs$verbatimSiteID)
no_join_data <- filter(data, !SurveyAreaIdentifier %in% corrs$verbatimSiteID)

setdiff(unique(no_join_data$ScientificName), unique(final_data$ScientificName))

setdiff(unique(no_join_data$SurveyAreaIdentifier), not_found)
```

6.8.2 Database connection

```
source("scripts/dbcon.R")
```

6.8.3 List tables in database

```
dbListTables(con) %>%
    purrr::keep(., ~ grepl("^CB|^MOBI", .)) %>%
    kable(col.names = "Tables")
```

6.8.4 Write dataset tables

6.8.4.1 Site

```
table <- "MOBI_site" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
    as.data.frame() %>%
    colnames() %>%
    paste(., ' = ""', collapse = ",\n") %>%
    cat()
```

Create data table

```
data_table <- grid %>%
    st_as_sf() %>%
```

```
    mutate(datasetID = datasetID, verbatimSiteID = SQUARE_ID, footprintSRS = "epsg:4326
    select(datasetID, verbatimSiteID, footprintSRS, verbatimFootprintSRS, gridID, geome
    unique()

data_table$geometry <- st_cast(data_table$geometry, "MULTIPOLYGON")

if (nrow(data_table) != nrow(grid)) {
    stop("Stopping execution due to row number mismatch")
} else {
    print("OK")
}
```

Write into the database

```
try({
    # Remove records where datasetID is the current dataset
    dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

    # Write records
    st_write(obj = data_table, dsn = con, layer = table, append = T)
    # Remove the data_table from the environment
    rm(data_table)
})
```

```
tbl(
    con,
    sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))
) %>%
    head(n = 20) %>%
    kable(align = "c")
```

6.8.5 Write records tables

6.8.5.1 Event

```
table <- "MOBI_event" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
    as.data.frame() %>%
    colnames() %>%
    paste(., ' = ""', collapse = ",\n") %>%
    cat()
```

Create data table

```
data_table <- event %>%
    mutate(gridID = 1) %>%
    mutate(across(everything(), ~na_if(., "NULL")))

if (nrow(unique(data_table)) != nrow(event)) {
    stop("Stopping execution due to row number mismatch")
} else {
    print("OK")
}

data_table <- filter(data_table, !is.na(verbatimSiteID))
```

Write into the database

```
try({
    # Remove records where datasetID is the current dataset
    dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

    # Copy data to the database
    copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
# rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
    head() %>%
    kable(align = "c")
```

6.8.5.2 Presence

```
table <- "MOBI_presence" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_presence_18" PARTITION OF "MOBI_presence"
FOR VALUES IN (18);
```

Check table colnames

```
tbl(con, table) %>%
    as.data.frame() %>%
    colnames() %>%
    paste(., ' = ""', collapse = ",\n") %>%
    cat()
```

Create data table

```
data_table <- data.frame(
    verbatimIdentificationID = final_data$verbatimIdentificationID,
    verbatimSiteID = final_data$cellID,
    datasetID = datasetID,
    samplingPeriodID = final_data$samplingPeriodID,
    recordFilter = NA
)
```

Write into the database

```
try({
    # Remove records where datasetID is the current dataset
    dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

    # Copy data to the database
    copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
    head() %>%
    kable(align = "c")
```

6.8.5.3 Probability

```
table <- "MOBI_probability" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_probability_18" PARTITION OF "MOBI_probability"
FOR VALUES IN (18);
```

Check table colnames

```
tbl(con, table) %>%
    as.data.frame() %>%
    colnames() %>%
    paste(., ' = ""', collapse = ",\n") %>%
    cat()
```

Create data table

```
data_table <- data.frame(
    verbatimIdentificationID = "",
```

```
    verbatimSiteID = "",
    datasetID = "",
    samplingPeriodID = "",
    probability = ""
)
```

Write into the database

```
try({
    # Remove records where datasetID is the current dataset
    dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

    # Copy data to the database
    copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
    head() %>%
    kable(align = "c")
```

6.9 Finish session

```
sessionInfo()
```

6.9.1 Clean up workspace

```
dbDisconnect(con)
rm(list = ls())
gc()
.rs.restartR()
```

Chapter 7

Birds of Quebec

7.1 MOBI team roles

Responsibilities	Name	Date (MM/YY)
Data acquisition		09/23
Metadata preparation	Kateřina Tschernosterová	07/24
Data standardization	Kateřina Tschernosterová	12/24
Data processing	Gabriel Ortega	12/24

7.2 Data providers, metadata and co-authorship

Always check the updated information here.

The original documents folder including books, publications or email exchanges are here.

7.3 Data description

- Existing temporal replications of atlases
  - BBA1 - sampling period 1984 - 1989
  - BBA2 - sampling period 2010 - 2014
- Observation data by grid cell, by species and by observation card (year, sampling event and data collector).
- Presence is reported, no data on abundance or absence.
- Original data detail: Scientific species name, Common species name, Survey area identifier, Decimal latitude and longitude of survey area, Year

(month, day) collected, Collector number, Sampling event identifier, Effort in hours (BBA2 only, not for all events), Breeding bird atlas code.

- Atlas data use agreement: https://www.atlas-oiseaux.qc.ca/convention_data_v3_sept_2018_en.jsp.
- Data was downloaded through NatureCounts website: https://naturecounts.ca/nc/qcatlas/main.jsp.
  - Downloaded BBA1 dataset: `Québec Breeding Bird Atlas (1984-1989): raw breeding evidence [Québec Breeding Bird Atlas]`.
  - Downloaded BBA2 dataset: `Québec Breeding Bird Atlas (2010-2014): raw breeding evidence [Québec Breeding Bird Atlas]`.
  - Another available datasets that were not used:
    - `Québec Breeding Bird Atlas (2010-2014): daily observations [Québec Breeding Bird Atlas]`.
    - `Québec Breeding Bird Atlas (2010-2014): point count data [Québec Breeding Bird Atlas]`.
    - `Québec Breeding Bird Atlas (2010-2014): rare species and colonial data [Québec Breeding Bird Atlas]`.
    - `Québec Breeding Bird Atlas (northern project): daily observations [Québec Breeding Bird Atlas]`.
    - `Québec Breeding Bird Atlas (northern project): raw breeding evidence [Québec Breeding Bird Atlas]`.
    - `Québec Breeding Bird Atlas (northern project): point count data [Québec Breeding Bird Atlas]`.
- BBA1 - The Breeding Birds of Québec: Atlas of the Breeding Birds of Southern Québec.
  - French version was published 1995, English version was published 1996.
  - BBA1 website: https://www.atlas-oiseaux.qc.ca/1eratlas_en.jsp.
  - The Atlas field campaign for the Quebec territory located north of latitude 50°30' N has not been completed, no data for squares which are north of latitude 50°30' N.
- BBA2 - The Second Atlas of Breeding Birds of Southern Quebec.
  - BBA2 website: https://www.atlas-oiseaux.qc.ca/index_en.jsp.
  - Book was published in April 2019.
  - You can view the book online using QuébecOiseaux's miLibris reader: https://quebecoiseauxkiosk.milibris.com/atlas.
  - The Atlas field campaign for the Quebec territory located north of latitude 50°30' N has been completed, but there are no plans to publish the results in a book.
  - However, the data collected north of latitude 50°30' N as part of the Atlas campaign are available through NatureCounts - you can consult these on distribution maps or even download them (Atlas of breeding birds of Quebec (northern region)).

7.3.1 Sampling methodology

- Data was collected in sampling squares by volunteer atlasers, sampling atlas squares are 10 km by 10 km (100 km2).
- BBA1 book and NatureCounts database contains data only for southern Quebec, northern Quebec data are not available.
- BBA2 book contains data only for southern Quebec, but data for northern Quebec are available in NatureCounts database.

7.3.2 Sampling effort

- Participants were encouraged to spend about 20 hours looking for breeding evidence in a square. It is estimated that it takes 16 to 20 hours of observation to find approximately 75% of the bird species in a square (BBA2 pdf, p.13).
- TABLE 3.1 (BBA2 pdf, p.21) – Statistics on atlasers participation in the fieldwork for BBA1 and BBA2 (note: effort in hours is simple sum of hours per event, but some events are missing this information - no data or hours = 0).

Metric	First Atlas (1984-1989)	Second Atlas (2010-2014)
Number of participants	932	1,805
Observation effort (hours)	67,587	97,625
Number of squares with records	2,462	4,033
Number of data forms filled	9,041	31,679
Number of records	202,521	566,834
Number of point counts	N/A	34,502

7.3.3 Compressed original data folder

- `qcatlas1be_raw_field_names.txt`: column names used in BBA1.
- `qcatlas1be_raw_naturecounts_data.txt`: raw data for BBA1 1984-1989.
- `qcatlas2be_raw_field_names.txt`: column names used in BBA2.
- `qcatlas2be_raw_naturecounts_data.txt`: raw data for BBA2 2010-2017.
- `grid.kmz`: grid map with 10x10 km survey squares grouped into 100 km by 100 km blocks - based on the Universal Transverse Mercator (UTM) grid; covers whole area of Quebec (southern and northern territory).

7.3.4 Docs folder

- `Birds_Atlas_Quebec_1-2_Data policy.html`: NatureCounts policy on use and publication of atlas data (version May 2018).

- `Birds_Atlas_Quebec_2_Guide.pdf`: handbook with instructions for volunteer atlasers (April 2001).
- `Second Atlas of the Breeding Birds of Southern Quebec.pdf`: Pdf version of atlas for Southern Quebec territories, atlas for North Quebec territories was not printed out.

7.3.5 Checked data folder

- `Birds_Atlas_Quebec_1_1984-1989_eff_data.xlsx`: standardized effort data.
- `Birds_Atlas_Quebec_1_1984-1989_occ_data.xlsx`: standardized occurrence data.
- `Birds_Atlas_Quebec_2_2010-2017_eff_data.xlsx`: standardized effort data.
- `Birds_Atlas_Quebec_2_2010-2017_occ_data.xlsx`: standardized occurrence data.
- `Birds_Atlas_Quebec_CheckedData_working.xlsx`: effort and occurrence data standardization, working file. `-grids_correspondence_table.csv`: equivalence table to map the original grid cells to coarser resolutions.
- `modified_grid.gpkg`: wide version of correspondence table including spatial geometries.

7.4 Data standardization and processing comments

- Sampling effort reported in NatureCounts data is not complete - effort in hours is missing or is equal to 0, for some sampling events - therefore sum of hours does not make sense.
- We decided that sampling effort as number of events per sampling square is more reasonable - data was aggregated per event.
- BBA1 data are covering only southern Quebec, BBA2 data are covering whole area of Quebec (as mentioned in methodology part).

7.5 Libraries

```
pacman::p_load(
  sf, terra, tidyverse, tidyterra, knitr,
  tictoc, RPostgres, DBI, dbplyr, parallel,
  geodata
)
```

7.6 Grid processing

```
rm -rf /tmp/Original_data
unzip data/Birds_Quebec/Original_data.zip -d /tmp/
```

```
glimpse(vect("/tmp/Original_data/grid.kmz"))
```

```
# Custom ceiling function
custom_ceiling <- function(x) {
  return(ceiling(x / 5000) * 5000)
}

# Function to import and modify the grid file on one pass
read_grid <- function(shpfile) {
  grid <- st_read(shpfile)
  grid$row_ID <- paste0("ID", 1:nrow(grid))
  grid$cellID <- grid$Name
  grid$utm_zon <- str_extract(grid$Name, "^\\d{2}")
  temp <- lapply(unique(grid$utm_zon), function(x) {
    res <- filter(grid, utm_zon == x) %>%
      st_transform(., as.numeric(paste0(269, x))) %>%
      st_centroid() %>%
      mutate(
        utm_stn = unlist(map(.$geometry, 1)),
        utm_nrt = unlist(map(.$geometry, 2)) %>% custom_ceiling(.)
      ) %>%
      st_drop_geometry() %>%
      select(row_ID, utm_stn, utm_nrt)
  }) %>% do.call(rbind, .)

  grid <- left_join(grid, temp, by = "row_ID")
  return(grid)
}

# Apply to data grid
grid <- read_grid("/tmp/Original_data/grid.kmz") %>% select(row_ID, cellID, utm_stn, utm_nrt, utm
```

```
grid_grouping <- function(grid, col1, col2, num) {
  grid <- st_drop_geometry(grid)
  mincol1 <- min(grid[[col1]])
  maxcol1 <- max(grid[[col1]])
  mincol2 <- min(grid[[col2]])
  maxcol2 <- max(grid[[col2]])
  breaksA <- seq(mincol1, maxcol1, num)
  breaksB <- seq(mincol2, maxcol2, num)
  if (length(breaksA) > 1) {
```

```
    A <- cut(grid[[col1]], breaks = breaksA, right = F)
  } else {
    A <- 1
  }
  if (length(breaksB) > 1) {
    B <- cut(grid[[col2]], breaks = breaksB, right = F)
  } else {
    B <- 1
  }
  res <- paste0(grid$utm_zon, "_", A, B)
  return(res)
}
```

Create labels for rescaling

```
for (dist in c(2, 4, 8, 16, 32, 64, 128)) {
  num <- 10000 * dist
  grid[[paste0(dist)]] <- grid_grouping(grid, col1 = "utm_stn", col2 = "utm_nrt", num =
}

grid$`1` <- dense_rank(grid$cellID)
grid$`256` <- 1

grid <- grid %>% mutate(across(matches("^\\d"), dense_rank))

st_write(grid, "/tmp/Quebec.gpkg", delete_dsn = T)
```

The grid was manually checked in QGIS to merge small cells with their neighbors. Small or even linear cells occur in borders of UTM zones in the original grid. We also reassigned some cells to groupings in every scale so that they end up mostly aligned and closer to the expected standard sizes per scale. The resulting grid was saved as modified_grid.gpkg.

```
qgis /tmp/Quebec.gpkg
```

We have different percentages of overlap between cells and real country borders. Therefore, we should have a way to correct the areas.

```
grid <- st_read("data/Birds_Quebec/Checked_data/modified_grid.gpkg") %>%
  mutate(across(matches("^X\\d"), dense_rank))

adm_boundary <- gadm("CAN", path = "/tmp/", resolution = 1) %>%
  select(HASC_1) %>%
  filter(HASC_1 == "CA.QC") %>%
  project(., "epsg:4326")

int <- terra::intersect(select(vect(grid), cellID), adm_boundary)
```

```
int <- group_by(int, cellID) %>% summarise()

int$cropped_area <- expanse(int, unit = "km")

grid <- as.data.frame(int) %>%
  left_join(grid, ., by = "cellID")
```

Calculate cell area with and without correction by landmasses or country borders.

```
grid$cell_area <- expanse(vect(grid), unit = "km")
grid$cell_area_proportion <- grid$cropped_area / grid$cell_area %>% round(., digits = 2)
```

Remove cells without ID (if any)

```
noID <- filter(grid, is.na(cellID))
grid <- filter(grid, !is.na(cellID))
```

```
# Pivot cells to a longer format for easier use in models.
final_grids <- grid %>%
  pivot_longer(matches("^X\\d"),
    names_to = "cell_grouping",
    values_to = "cell_label"
  ) %>%
  mutate(cell_grouping = str_remove_all(cell_grouping, "X") %>%
    as.numeric())

final_grids <- split(final_grids, final_grids$cell_grouping)
```

Aggregate cells and summarize the area.

```
final_grids <- final_grids %>%
  lapply(., function(x) {
    res <- x %>%
      group_by(cell_grouping, cell_label) %>%
      summarise(
        area = sum(cell_area, na.rm = T),
        area_cropped = sum(cropped_area, na.rm = T)
      ) %>%
      ungroup()
  })
```

Functions to add additional variables of cell shapes

```
# Function to get cell shape attributes

fdistances <- function(x, type = NULL) {
  x <- vect(x)
```

```
  pol <- as.data.frame(crds(as.points(ext(project(x, "epsg:4326")))))
  if (type == "ew") {
    sw <- pol %>%
      summarise(x = min(x), y = min(y)) %>%
      as.matrix()
    se <- pol %>%
      summarise(x = max(x), y = min(y)) %>%
      as.matrix()
    res <- distance(sw, se, lonlat = T)[[1]] / 1000
  }
  if (type == "ns") {
    sw <- pol %>%
      summarise(x = min(x), y = min(y)) %>%
      as.matrix()
    nw <- pol %>%
      summarise(x = min(x), y = max(y)) %>%
      as.matrix()
    res <- distance(sw, nw, lonlat = T)[[1]] / 1000
  }
  if (type == "max") {
    res <- distance(pol, lonlat = T) %>% max()
    res <- res / 1000
  }
  res <- as.numeric(res)
  return(res)
}
```

Add cell max length, distance south-north, and distance east-west.

```
tic()

add_cell_lengths <- function(x) {
  res <- x
  list <- res %>%
    select(cell_label)
  list <- terra::split(list, list$cell_label)

  res$cell_max_length <- lapply(list, function(.x) fdistances(.x, type = "max")) %>%
    do.call(rbind, .) %>%
    as.numeric()

  res$cell_ns_length <- lapply(list, function(.x) fdistances(.x, type = "ns")) %>%
    do.call(rbind, .) %>%
    as.numeric()

  res$cell_ew_length <- lapply(list, function(.x) fdistances(.x, type = "ew")) %>%
```

```
    do.call(rbind, .) %>%
    as.numeric()

  res <- st_transform(res, st_crs(x))
  return(res)
}

# Create a cluster
cl <- makeCluster(getOption("cl.cores", detectCores() - 5))

# Load necessary packages on each worker
clusterEvalQ(cl, {
  library(dplyr)
  library(terra)
  library(sf)
})

# Export the function to the cluster
clusterExport(cl, varlist = "fdistances")

# Use parSapply with the cluster
final_grids <- parSapply(cl, final_grids, add_cell_lengths, simplify = FALSE, USE.NAMES = TRUE)

# Stop the cluster
stopCluster(cl)

toc()
```

```
plet(vect(final_grids[[8]]))
```

Export results.

```
grid %>%
  st_drop_geometry() %>%
  select(matches("^cell|^X")) %>%
  write_csv(., "data/Birds_Quebec/Checked_data/grids_correspondence_table.csv")

file.remove("data/Birds_Quebec/Checked_data/final_grid.gpkg")

for (x in 1:length(final_grids)) {
  st_write(final_grids[[x]], layer = paste0("grid_", x), dsn = "data/Birds_Quebec/Checked_data/fi
}
```

7.7 Data processing

```
# files <- c("/tmp/Original_data/Birds_Atlas_Quebec_1_Data_raw\ breeding\ evidence.txt"
#
# columns <- c("ScientificName", "Locality", "SamplingEventIdentifier", "RouteIdentifie
#
# data <- lapply(files, function(x) read_delim(x) %>%
#                  select(any_of(columns)) %>%
#                  unique()) %>% do.call(rbind,.) #%>%
#   mutate(index = paste0(DecimalLongitude,"_",DecimalLatitude)) %>%
#   rename(verbatim_name = ScientificName) %>%
#   mutate(CollectorNumber = max(CollectorNumber)+1)
#
# View(summarise(data, across(everything(), ~sum(is.na(.)))))
#
# index <- select(data, index, DecimalLongitude, DecimalLatitude) %>%
#   unique() %>%
#   st_as_sf(., coords = c("DecimalLongitude","DecimalLatitude"), crs = 4326)
#
# index <- select(grid, cellID) %>%
#   st_join(., index) %>%
#   st_drop_geometry()
#
# data <- left_join(data, index, by = "index") %>% select(-index,-DecimalLongitude, -De
#   mutate(start_year = case_when(
#     YearCollected >= 1987 & YearCollected <= 1991 ~ 1987,
#     YearCollected >= 2000 & YearCollected <= 2005 ~ 2000
#   ),
#   end_year = case_when(
#     YearCollected >= 1987 & YearCollected <= 1991 ~ 1991,
#     YearCollected >= 2000 & YearCollected <= 2005 ~ 2005
#   ))
#
# occ <- select(data, cellID, verbatim_name, start_year, end_year) %>% unique()
#
# write_csv(occ, "data/Birds_Quebec/Checked_data/occFin.csv")
#
# eff <- select(data, -verbatim_name) %>%
#   group_by(across(-CollectorNumber)) %>%
#   summarise(effort = n_distinct(CollectorNumber)) %>%
#   ungroup() %>%
#   group_by(cellID, start_year, end_year) %>%
#   summarise(effort = sum(effort)) %>%
#   ungroup()
#
```

```
# write_csv(eff, "data/Birds_Quebec/Checked_data/effFin.csv")
#
# samp_cells <- select(occ, cellID, start_year) %>%
#   rbind(., select(eff, cellID, start_year)) %>%
#   unique() %>%
#   mutate(start_year = dense_rank(start_year),
#          value = 1) %>%
#   pivot_wider(names_from = start_year,
#               values_from = value,
#               values_fill = 0) %>%
#   mutate(repeated = rowSums(select(., matches("^\\d")))) %>%
#   select(cellID, repeated)
#
# write_csv(samp_cells, "data/Birds_Quebec/Checked_data/samp_cells.csv")
```

7.8 Adding records to the database

7.8.1 Data

Import the processed data and grid.

```
occ_files <- list.files("data/Birds_Quebec/Checked_data/", pattern = "_occ_data.xlsx", full.names

data <- lapply(occ_files, function(x) {
  readxl::read_xlsx(x) %>%
    filter(!is.na(cellID))
}) %>%
  do.call(rbind, .) %>%
  mutate(
    verbatimIdentificationID = dense_rank(verbatim_name),
    samplingPeriodID = dense_rank(start_year)
  )

sp_names <- data %>%
  select(verbatimIdentificationID, verbatim_name) %>%
  unique()

corrs <- read_csv("data/Birds_Quebec/Checked_data/grids_correspondence_table.csv") %>%
  mutate(across(matches("^X\\d"), dense_rank)) %>%
  select(cellID, matches("^X\\d"))

eff_files <- list.files("data/Birds_Quebec/Checked_data/", pattern = "_eff_data.xlsx", full.names

eff <- lapply(eff_files, function(x) {
  readxl::read_xlsx(x) %>%
```

```
    filter(!is.na(cellID))
}) %>% do.call(rbind, .)

eff <- select(data, cellID, start_year, end_year) %>%
  unique() %>%
  setdiff(., select(eff, cellID, start_year, end_year)) %>%
  mutate(effort_1 = NA, effort_1_unit = NA_character_) %>%
  bind_rows(eff, .) %>%
  mutate(samplingPeriodID = dense_rank(start_year))

scaling_table <- corrs %>%
  pivot_longer(-cellID, names_to = "scalingID", values_to = "siteID") %>%
  mutate(scalingID = str_remove_all(scalingID, "^X") %>%
    as.numeric()) %>%
  ungroup()

samp_cells <- select(data, cellID, start_year) %>%
  rbind(., select(eff, cellID, start_year)) %>%
  unique() %>%
  group_by(cellID) %>%
  mutate(repeated = n_distinct((start_year))) %>%
  ungroup() %>%
  select(!start_year) %>%
  unique() %>%
  left_join(., corrs, by = "cellID") %>%
  pivot_longer(-c(cellID, repeated), names_to = "cell_grouping", values_to = "cell_labe
  mutate(cell_grouping = str_remove_all(cell_grouping, "^X") %>%
    as.numeric()) %>%
  na.omit() %>%
  group_by(cell_grouping, cell_label) %>%
  summarise(repeated = max(repeated)) %>%
  ungroup()

grid <- st_layers("data/Birds_Quebec/Checked_data/final_grid.gpkg")$name %>%
  mclapply(., function(x) {
    st_read("data/Birds_Quebec/Checked_data/final_grid.gpkg",
      layer = x, quiet = T
    )
  }, mc.cores = 10) %>%
  do.call(rbind, .) %>%
  ungroup() %>%
  st_make_valid() %>%
  mutate(coords = st_centroid(geom)) %>%
  mutate(
    cell_lat = st_coordinates(coords)[, 2],
```

```
    cell_long = st_coordinates(coords)[, 1]
  )


grid_table <- st_drop_geometry(grid) %>%
  left_join(., samp_cells)
```

7.8.1.1 Basic double check

```
# Check that all cellID in data are also in the corrs cellID
if (length(setdiff(unique(data$cellID), unique(corrs$cellID))) == 0) {
  print("OK")
} else {
  print("FAIL")
}
# Check that number of rows in grid and corrs are the same
nrow(filter(grid, cell_grouping == 1)) == nrow(corrs)

# Check that years and sampling periods are the same in data and eff
nrow(setdiff(
  unique(select(data, start_year, samplingPeriodID)),
  unique(select(eff, start_year, samplingPeriodID))
))

# Check uniqueness of scaling_table
nrow(unique(scaling_table)) == nrow(scaling_table)

# Check that all cellID in data are also in the grid (corrs)
length(setdiff(
  unique(data$cellID),
  unique(corrs$cellID)
))

# Check that all cellID in eff are also in the grid (corrs)
length(setdiff(
  unique(eff$cellID),
  unique(corrs$cellID)
))

# RESULTS SHOULD BE: OK, TRUE, 0, TRUE, NULL, NULL
```

7.8.1.2 Atlas variables

```
datasetID <- 20
effortID <- 2
licenseID <- 0
shareable <- "NO"
verbatimFootprintSRS <- "epsg:4326"
```

All cells match between the different dataset files.

```
final_data <- inner_join(data, corrs, by = "cellID")
no_join_data <- anti_join(data, corrs, by = "cellID")
```

7.8.2 Database connection

```
source("scripts/dbcon.R")
```

7.8.3 List tables in database

```
dbListTables(con) %>%
  purrr::keep(., ~ grepl("^CB|^MOBI", .)) %>%
  kable(col.names = "Tables")
```

7.8.4 Write code books

7.8.4.1 CB_license

```
table <- "CB_license" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  licenseID = licenseID,
  license = "Restricted",
  licenseDescription = "Closed data. Special request or purchase required",
  licenseURL = "none"
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check the table

```
tbl(con, table) %>% kable(align = "c")
```

7.8.4.2 CB_sampling_effort

```
table <- "CB_sampling_effort" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  samplingEffortID = 2,
  samplingEffortProtocol = "Sampling events per gridcell",
  samplingEffortUnit = "Sampling events"
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table.

```
tbl(con, table) %>% kable(align = "c")
```

7.8.4.3 CB_model

```
table <- "CB_model" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
```

```
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  occurrenceModelID = "",
  modelName = "",
  predictorVariables = "",
  bibliographicCitation = ""
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table

```
tbl(con, table) %>% kable(align = "c")
```

7.8.4.4 CB_taxonomy

```
table <- "CB_taxonomy" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  scientificNameID = "",
  scientificName = "",
  scientificNameAuthorship = "",
  kingdom = "",
  phylum = "",
  class = "",
  family = "",
  order = "",
  genus = "",
  specificEpitet = "",
  infraspecificEpitet = "",
```

```
  taxonRank = ""
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

7.8.4.5 CB_verbatim_name_equivalence

```
table <- "CB_verbatim_name_equivalence" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimIdentificationID = sp_names$verbatimIdentificationID,
  verbatimIdentification = sp_names$verbatim_name,
  datasetID = datasetID,
  identificationReferences = "",
  scientificNameID = NA
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
```

```
  kable(align = "c")
```

7.8.5 Write dataset tables

7.8.5.1 Dataset

```
table <- "MOBI_dataset" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  datasetID = datasetID,
  datasetName = "Quebec Breeding Bird Atlas",
  datasetPublisher = "Environment Canada's Québec Regional Office",
  datasetPublisherContact = "info@atlas-oiseaux.qc.ca",
  licenseID = licenseID,
  rightsHolder = "Environment Canada's Québec Regional Office",
  bibliographicCitation = "Québec Breeding Bird Atlas. 2024. Data accessed from Nature
  citationIdentifier = "",
  provider = "Environment Canada's Québec Regional Office",
  shareable = "NO",
  coauthorshipRequired = "NO",
  coauthors = "",
  coauthorshipSuggested = "Carmen Soria - carmendianasoria@gmail.com; Kateřina Tscherno
  isSamplingEffortReported = "YES",
  isOccurrenceProbabilityAvailable = "NO"
)
```

Write into the database

```
try({
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>
```

7.8.5.2 Site

```
table <- "MOBI_site" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_site_20" PARTITION OF "MOBI_site"
FOR VALUES IN (20);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  siteID = grid_table$cell_label,
  scalingID = grid_table$cell_grouping,
  datasetID = datasetID,
  area = grid_table$area,
  croppedArea = grid_table$area_cropped,
  areaUnit = "km2",
  maxLength = grid_table$cell_max_length,
  northSouthLength = grid_table$cell_ns_length,
  eastWestLength = grid_table$cell_ew_length,
  lengthUnit = "km",
  centroidDecimalLongitude = grid_table$cell_long,
  centroidDecimalLatitude = grid_table$cell_lat,
  samplingRepetitions = grid_table$repeated
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(
  con,
  sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))
) %>%
  head(n = 20) %>%
  kable(align = "c")
```

7.8.5.3 Geometry

```
table <- "MOBI_geometry" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_geometry_20" PARTITION OF "MOBI_geometry"
FOR VALUES IN (20);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- grid %>%
  ungroup() %>%
  rename(
    siteID = cell_label,
    scalingID = cell_grouping,
    geometry = geom
  ) %>%
  mutate(
    siteID = as.integer(siteID),
    scalingID = as.integer(scalingID),
    datasetID = as.integer(datasetID),
    footprintSRS = "epsg:4326",
    verbatimFootprintSRS = verbatimFootprintSRS
  ) %>%
  select(
    ., siteID, scalingID, datasetID,
    footprintSRS, verbatimFootprintSRS, geometry
  )

data_table$geometry <- st_cast(data_table$geometry, "MULTIPOLYGON")
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Write records
  st_write(obj = data_table, dsn = con, layer = table, append = T)
  # Remove the data_table from the environment
  rm(data_table)
})
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

7.8.6 Write helper tables

7.8.6.1 Scaling table

This intermediate table enables the joins of records data to the different resolutions of sites.

```
table <- "MOBI_scaling_table" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_scaling_table_20" PARTITION OF "MOBI_scaling_table"
FOR VALUES IN (20);
```

Check table colnames.

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

```
data_table <- data.frame(
  siteID = scaling_table$siteID,
  scalingID = scaling_table$scalingID,
  datasetID = datasetID,
  verbatimSiteID = scaling_table$cellID
)
```

Write into database.

```
try({
  # Remove records where datasetID is the current dataset
```

```
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

7.8.7 Write records tables

7.8.7.1 Event

```
table <- "MOBI_event" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_event_20" PARTITION OF "MOBI_event"
FOR VALUES IN (20);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimSiteID = eff$cellID,
  datasetID = datasetID,
  samplingPeriodID = eff$samplingPeriodID,
  startYear = eff$start_year,
  endYear = eff$end_year, samplingEffortID = 2,
  samplingEffortValue = ifelse(eff$effort_1 == 0, NA,
    eff$effort_1
  ),
  samplingEffort2ID = 4,
  samplingEffort2Value = ifelse(eff$effort_2 == 0, NA,
    eff$effort_2
```

```
  )
)

if (nrow(unique(data_table)) != nrow(eff)) {
  stop("Stopping execution due to row number mismatch")
} else {
  print("OK")
}

data_table <- filter(data_table, !is.na(verbatimSiteID))
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

7.8.7.2 Presence

```
table <- "MOBI_presence" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_presence_20" PARTITION OF "MOBI_presence"
FOR VALUES IN (20);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimIdentificationID = final_data$verbatimIdentificationID,
  verbatimSiteID = final_data$cellID,
  datasetID = datasetID,
  samplingPeriodID = final_data$samplingPeriodID,
  recordFilter = NA
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>
  head() %>%
  kable(align = "c")
```

7.8.7.3 Probability

```
table <- "MOBI_probability" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_probability_20" PARTITION OF "MOBI_probability"
FOR VALUES IN (20);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimIdentificationID = "",
  verbatimSiteID = "",
```

```
  datasetID = "",
  samplingPeriodID = "",
  probability = ""
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

7.9 Finish session

```
sessionInfo()
```

7.9.1 Clean up workspace

```
dbDisconnect(con)
rm(list = ls())
gc()
.rs.restartR()
```

Chapter 8

Birds of The Maritimes

8.1 MOBI team roles

Responsibilities	Name	Date (MM/YY)
Data acquisition		08/23
Metadata preparation	Kateřina Tschernosterová	
Data standardization	Kateřina Tschernosterová	07/24
Data processing	Gabriel Ortega	

8.2 Data providers metadata and co-authorships

Always check the updated information here.

The original documents folder including books, publications or email exchanges are here.

8.3 Data description

- Existing temporal replications of atlases
  - BBA1 - sampling period 1986 - 1990
  - BBA2 - sampling period 2006 - 2010
- `Observation data`: by grid cell, by species and by observation card (sampling event and data collector).
- `Presence is reported`: no data on abundance or absence.
- `Original data detail`: Scientific species name, Common species name, Survey area identifier, Decimal latitude and longitude of survey area, Year

collected, Collector number, Sampling event identifier, Effort in hours (BBA1 only, not for all events), Breeding bird atlas code.
- `The Maritimes region`: includes provinces New Brunswick, Nova Scotia and Prince Edward Island.
- `Data was downloaded`: through NatureCounts website - https://naturecounts.ca/nc/default/explore.jsp#download.
  - `Downloaded dataset`: with BBA1 data - Maritimes Breeding Bird Atlas (1986-1990): raw breeding evidence [Maritimes Breeding Bird Atlas].
  - `Downloaded dataset`: with BBA2 data - Maritimes Breeding Bird Atlas (2006-2010): raw breeding evidence [Maritimes Breeding Bird Atlas].
  - `Another available dataset`: that was not used - Maritimes Breeding Bird Atlas (2006-2010): point count data [Maritimes Breeding Bird Atlas].
- `BBA1 website`: https://www.mba-aom.ca/first-atlas/.
- `BBA2 website`: https://www.mba-aom.ca/.
- `BBA2 online book`: https://www.mba-aom.ca/jsp/toc.jsp.

8.3.1 Sampling methodology

- Data was collected in sampling squares by volunteer atlasers.
- For both atlases, the basic sampling unit was a 10 km by 10 km (100 km2) square from the Universal Transverse Mercator (UTM) grid.
- The UTM coordinates of the squares were almost identical in the first and second atlases, but they shifted slightly because the UTM reference parameters changed between atlases. For the first atlas, the North American Datum of 1927 (NAD27) was used, whereas for the second atlas the North American Datum of 1983 (NAD83) was used. (BBA2 pdf, p. 35).

8.3.2 Sampling effort

- Sampling square was considered fully surveyed, or "complete," once it received 20 hours of cumulative survey effort or when a substantial number of species was detected in the square. (BBA2 pdf, p. 37).
- For BB1, sampling duration (effort) in hours is reported for some sampling events, but for some events the value of effort in hours is equal to 0 and for some events the value is missing. The value of sampling event duration is not complete, so it can be used only with restrictions.
- Therefore, sampling effort for BBA1 is also reported as the number of sampling events per sampling square.
- In BBA2 data, there is no information on the duration of sampling events. Therefore, sampling effort is reported only as the number of sampling events per sampling square.

8.4 Input Data

8.4.1 Compressed original data folder

- `Birds_Atlas_The Maritimes_1_Column names.txt`: text file with the list of original data column names.
- `Birds_Atlas_The Maritimes_1_OriginalData.csv`: csv file with raw observation data for BBA1 1986-1990.
- `Birds_Atlas_The Maritimes_2_OriginalData.csv`: csv file with raw observation data for BBA2 2006-2010.
- `maritimes_gridmap shapefile`: original grid shapefile.

8.4.2 Docs folder

- `Birds_Atlas_The Maritimes_2_Documentation.pdf`: Breeding Bird Atlas project final report.
- `Birds_Atlas_The Maritimes_2_Guide.pdf`: handbook with instructions for volunteer atlasers.
- `Birds_Atlas_The Maritimes_1-2_Data policy.html`: NatureCounts policy on use and publication of atlas data.
- `First Breeding Birds Atlas of the Maritimes Provinces, 1992.pdf`: Pdf version of atlas.
- `Second Atlas of Breeding Birds of the Maritime Provinces, 2015`: Folder with pdf version of atlas divided into 17 pdf parts.

8.4.3 Checked data folder

- `Birds_Atlas_TheMaritimes_CheckedData_working.xlsx`: effort and occurrence data standardization, working file.
- `Birds_Atlas_TheMaritimes_1_1986-1990_occ_data.xlsx`: standardized occurrence data.
- `Birds_Atlas_TheMaritimes_1_1986-1990_eff_data.xlsx`: standardized effort data.
- `Birds_Atlas_TheMaritimes_2_2006-2010_occ_data.xlsx`: standardized occurrence data.
- `Birds_Atlas_TheMaritimes_2_2006-2010_eff_data.xlsx`: standardized effort data.

8.5 Data standardization and processing comments

- !!! IMPORTANT !!! The sampling effort in hours for BBA1 is not complete.

8.6 Libraries

```
pacman::p_load(
  sf, terra, tidyverse, tidyterra, knitr,
  tictoc, RPostgres, DBI, dbplyr, parallel,
  geodata
)
```

8.7 Grid processing

```
rm -rf /tmp/Original_data
unzip data/Birds_The_Maritimes/Original_data.zip -d /tmp/
```

```
glimpse(vect("/tmp/Original_data/maritimes_gridmap.shp"))
```

There were two cells whose BLOCK id was NA. One was in the sea and the other covering lake Oneida.

```
grid <- st_read("/tmp/Original_data/maritimes_gridmap.shp") %>%
  select(utm_sqr, utm_stn, utm_nrt, utm_zon) %>%
  rename(cellID = utm_sqr) %>%
  mutate(cellID = str_c(str_sub(cellID, 1, 2), str_sub(cellID, 4))) %>%
  unique()
```

```
grid_grouping <- function(grid, col1, col2, num) {
  grid <- st_drop_geometry(grid)
  mincol1 <- min(grid[[col1]])
  maxcol1 <- max(grid[[col1]])
  mincol2 <- min(grid[[col2]])
  maxcol2 <- max(grid[[col2]])
  breaksA <- seq(mincol1, maxcol1, num)
  breaksB <- seq(mincol2, maxcol2, num)
  if (length(breaksA) > 1) {
    A <- cut(grid[[col1]], breaks = breaksA, right = F)
  } else {
    A <- 1
  }
  if (length(breaksB) > 1) {
    B <- cut(grid[[col2]], breaks = breaksB, right = F)
  } else {
    B <- 1
  }
  res <- paste0(grid$utm_zon, "_", A, B)
  return(res)
}
```

Create labels for rescaling

```
for (dist in c(2, 4, 8, 16, 32, 64)) {
  num <- 10000 * dist
  grid[[paste0(dist)]] <- grid_grouping(grid, col1 = "utm_stn", col2 = "utm_nrt", num = num)
}

grid$`1` <- dense_rank(grid$cellID)
grid$`128` <- 1

grid <- grid %>% mutate(across(matches("^\\d"), dense_rank))

st_write(grid, "/tmp/The_Maritimes.gpkg", delete_dsn = T)
```

```
qgis /tmp/The_Maritimes.gpkg
```

We have different percentages of overlap between cells and real country borders. Therefore, we should have a way to correct the areas.

```
grid <- st_read("data/Birds_The_Maritimes/Checked_data/modified_grid.gpkg") %>%
  mutate(across(matches("^X\\d"), dense_rank))

adm_boundary <- gadm("CAN", path = "/tmp/", resolution = 1) %>%
  select(ISO_1) %>%
  filter(ISO_1 %in% c("CA-NB", "CA-NS", "CA-PE")) %>%
  project(., "epsg:4326")

int <- terra:intersect(select(vect(grid), cellID), adm_boundary)

int <- group_by(int, cellID) %>% summarise()

int$cropped_area <- expanse(int, unit = "km")

grid <- as.data.frame(int) %>%
  left_join(grid, ., by = "cellID")
```

Calculate cell area with and without correction by landmasses or country borders.

```
grid$cell_area <- expanse(vect(grid), unit = "km")
grid$cell_area_proportion <- grid$cropped_area / grid$cell_area %>% round(., digits = 2)
```

Remove cells without ID (if any)

```
noID <- filter(grid, is.na(cellID))
grid <- filter(grid, !is.na(cellID))
```

```
# Pivot cells to a longer format for easier use in models.
final_grids <- grid %>%
```

```
  pivot_longer(matches("^X\\d"),
    names_to = "cell_grouping",
    values_to = "cell_label"
  ) %>%
  mutate(cell_grouping = str_remove_all(cell_grouping, "X") %>%
    as.numeric())

final_grids <- split(final_grids, final_grids$cell_grouping)
```

Aggregate cells and summarize the area.

```
final_grids <- final_grids %>%
  lapply(., function(x) {
    res <- x %>%
      group_by(cell_grouping, cell_label) %>%
      summarise(
        area = sum(cell_area, na.rm = T),
        area_cropped = sum(cropped_area, na.rm = T)
      ) %>%
      ungroup()
  })
```

Functions to add additional variables of cell shapes

```
# Function to get cell shape attributes

fdistances <- function(x, type = NULL) {
  x <- vect(x)
  pol <- as.data.frame(crds(as.points(ext(project(x, "epsg:4326")))))
  if (type == "ew") {
    sw <- pol %>%
      summarise(x = min(x), y = min(y)) %>%
      as.matrix()
    se <- pol %>%
      summarise(x = max(x), y = min(y)) %>%
      as.matrix()
    res <- distance(sw, se, lonlat = T)[[1]] / 1000
  }
  if (type == "ns") {
    sw <- pol %>%
      summarise(x = min(x), y = min(y)) %>%
      as.matrix()
    nw <- pol %>%
      summarise(x = min(x), y = max(y)) %>%
      as.matrix()
    res <- distance(sw, nw, lonlat = T)[[1]] / 1000
  }
```

```
  if (type == "max") {
    res <- distance(pol, lonlat = T) %>% max()
    res <- res / 1000
  }
  res <- as.numeric(res)
  return(res)
}
```

Add cell max length, distance south-north, and distance east-west.

```
tic()

add_cell_lengths <- function(x) {
  res <- x
  list <- res %>%
    select(cell_label)
  list <- terra::split(list, list$cell_label)

  res$cell_max_length <- lapply(list, function(.x) fdistances(.x, type = "max")) %>%
    do.call(rbind, .) %>%
    as.numeric()

  res$cell_ns_length <- lapply(list, function(.x) fdistances(.x, type = "ns")) %>%
    do.call(rbind, .) %>%
    as.numeric()

  res$cell_ew_length <- lapply(list, function(.x) fdistances(.x, type = "ew")) %>%
    do.call(rbind, .) %>%
    as.numeric()

  res <- st_transform(res, st_crs(x))
  return(res)
}

# Create a cluster
cl <- makeCluster(getOption("cl.cores", detectCores() - 5))

# Load necessary packages on each worker
clusterEvalQ(cl, {
  library(dplyr)
  library(terra)
  library(sf)
})

# Export the function to the cluster
clusterExport(cl, varlist = "fdistances")
```

```
# Use parSapply with the cluster
final_grids <- parSapply(cl, final_grids, add_cell_lengths, simplify = FALSE, USE.NAMES

# Stop the cluster
stopCluster(cl)

toc()
```

```
plet(vect(final_grids[[6]]))
```

Export results.

```
grid %>%
  st_drop_geometry() %>%
  select(matches("^cell|^X")) %>%
  write_csv(., "data/Birds_The_Maritimes/Checked_data/grids_correspondence_table.csv")

file.remove("data/Birds_The_Maritimes/Checked_data/final_grid.gpkg")

for (x in 1:length(final_grids)) {
  st_write(final_grids[[x]], layer = paste0("grid_", x), dsn = "data/Birds_The_Maritime
}
```

8.8 Data processing

```
files <- c("/tmp/Original_data/Birds_Atlas_The_Maritimes_1_Data_raw\ breeding\ evidence

columns <- c("ScientificName", "Locality", "SamplingEventIdentifier", "RouteIdentifier"

data <- lapply(files, function(x) read_delim(x) %>%
                 select(any_of(columns)) %>%
                 unique()) %>% do.call(rbind,.) #%>%
  mutate(index = paste0(DecimalLongitude,"_",DecimalLatitude)) %>%
  rename(verbatim_name = ScientificName) %>%
  mutate(CollectorNumber = max(CollectorNumber)+1)

View(summarise(data, across(everything(), ~sum(is.na(.)))))

index <- select(data, index, DecimalLongitude, DecimalLatitude) %>%
  unique() %>%
  st_as_sf(., coords = c("DecimalLongitude","DecimalLatitude"), crs = 4326)

index <- select(grid, cellID) %>%
  st_join(., index) %>%
```

```
  st_drop_geometry()

data <- left_join(data, index, by = "index") %>% select(-index,-DecimalLongitude, -DecimalLatitud
  mutate(start_year = case_when(
    YearCollected >= 1987 & YearCollected <= 1991 ~ 1987,
    YearCollected >= 2000 & YearCollected <= 2005 ~ 2000
  ),
  end_year = case_when(
    YearCollected >= 1987 & YearCollected <= 1991 ~ 1991,
    YearCollected >= 2000 & YearCollected <= 2005 ~ 2005
  ))

occ <- select(data, cellID, verbatim_name, start_year, end_year) %>% unique()

write_csv(occ, "data/Birds_The_Maritimes/Checked_data/occFin.csv")

eff <- select(data, -verbatim_name) %>%
  group_by(across(-CollectorNumber)) %>%
  summarise(effort = n_distinct(CollectorNumber)) %>%
  ungroup() %>%
  group_by(cellID, start_year, end_year) %>%
  summarise(effort = sum(effort)) %>%
  ungroup()

write_csv(eff, "data/Birds_The_Maritimes/Checked_data/effFin.csv")

samp_cells <- select(occ, cellID, start_year) %>%
  rbind(., select(eff, cellID, start_year)) %>%
  unique() %>%
  mutate(start_year = dense_rank(start_year),
         value = 1) %>%
  pivot_wider(names_from = start_year,
              values_from = value,
              values_fill = 0) %>%
  mutate(repeated = rowSums(select(., matches("^\\d")))) %>%
  select(cellID, repeated)

write_csv(samp_cells, "data/Birds_The_Maritimes/Checked_data/samp_cells.csv")
```

8.9 Adding records to the database

8.9.1 Data

Import the processed data and grid.

```
occ_files <- list.files("data/Birds_The_Maritimes/Checked_data/", pattern = "_occ_data

data <- lapply(occ_files, function(x) {
  readxl:read_xlsx(x) %>%
    filter(!is.na(cellID))
}) %>%
  do.call(rbind, .) %>%
  mutate(
    verbatimIdentificationID = dense_rank(verbatim_name),
    samplingPeriodID = dense_rank(start_year)
  )

sp_names <- data %>%
  select(verbatimIdentificationID, verbatim_name) %>%
  unique()

corrs <- read_csv("data/Birds_The_Maritimes/Checked_data/grids_correspondence_table.csv
  mutate(across(matches("^X\\d"), dense_rank)) %>%
  select(cellID, matches("^X\\d"))

eff_files <- list.files("data/Birds_The_Maritimes/Checked_data/", pattern = "_eff_data

eff <- lapply(eff_files, function(x) {
  df <- readxl:read_xlsx(x) %>%
    filter(!is.na(cellID))
  if ("effort_1" %in% colnames(df)) {
    colnames(df)[colnames(df) == "effort_1"] <- "effort"
    colnames(df)[colnames(df) == "effort_1_unit"] <- "effort_unit"
  }
  return(df)
}) %>% data.table:rbindlist(fill = TRUE)


eff <- select(data, cellID, start_year, end_year) %>%
  unique() %>%
  setdiff(., select(eff, cellID, start_year, end_year)) %>%
  mutate(effort = 1, effort_unit = "n_events") %>%
  bind_rows(eff, .) %>%
  mutate(samplingPeriodID = dense_rank(start_year))

scaling_table <- corrs %>%
  pivot_longer(-cellID, names_to = "scalingID", values_to = "siteID") %>%
  mutate(scalingID = str_remove_all(scalingID, "^X") %>%
    as.numeric()) %>%
  ungroup()
```

```
samp_cells <- select(data, cellID, start_year) %>%
  rbind(., select(eff, cellID, start_year)) %>%
  unique() %>%
  group_by(cellID) %>%
  mutate(repeated = n_distinct((start_year))) %>%
  ungroup() %>%
  select(!start_year) %>%
  unique() %>%
  left_join(., corrs, by = "cellID") %>%
  pivot_longer(-c(cellID, repeated), names_to = "cell_grouping", values_to = "cell_label") %>%
  mutate(cell_grouping = str_remove_all(cell_grouping, "^X") %>%
    as.numeric()) %>%
  na.omit() %>%
  group_by(cell_grouping, cell_label) %>%
  summarise(repeated = max(repeated, na.rm = T)) %>%
  ungroup()

grid <- st_layers("data/Birds_The_Maritimes/Checked_data/final_grid.gpkg")$name %>%
  mclapply(., function(x) {
    st_read("data/Birds_The_Maritimes/Checked_data/final_grid.gpkg",
      layer = x, quiet = T
    )
  }, mc.cores = 10) %>%
  do.call(rbind, .) %>%
  ungroup() %>%
  st_make_valid() %>%
  mutate(coords = st_centroid(geom)) %>%
  mutate(
    cell_lat = st_coordinates(coords)[, 2],
    cell_long = st_coordinates(coords)[, 1]
  )


grid_table <- st_drop_geometry(grid) %>%
  left_join(., samp_cells)
```

8.9.1.1 Basic double check

```
# Check that all cellID in data are also in the corrs cellID
if (length(setdiff(unique(data$cellID), unique(corrs$cellID))) == 0) {
  print("OK")
} else {
  print("FAIL")
}
```

```
# Check that number of rows in grid and corrs are the same
nrow(filter(grid, cell_grouping == 1)) <= nrow(corrs)

# Check that years and sampling periods are the same in data and eff
nrow(setdiff(
  unique(select(data, start_year, samplingPeriodID)),
  unique(select(eff, start_year, samplingPeriodID))
))

# Check uniqueness of scaling_table
nrow(unique(scaling_table)) == nrow(scaling_table)

# Check that all cellID in data are also in the grid (corrs)
length(setdiff(
  unique(data$cellID),
  unique(corrs$cellID)
))

# Check that all cellID in eff are also in the grid (corrs)
length(setdiff(
  unique(eff$cellID),
  unique(corrs$cellID)
))

# RESULTS SHOULD BE: OK, TRUE, 0, TRUE, NULL, NULL
```

8.9.1.2 Atlas variables

```
datasetID <- 23
effortID <- 2
licenseID <- 0
shareable <- "NO"
verbatimFootprintSRS <- "epsg:4326"
```

All cells match between the different dataset files.

```
final_data <- inner_join(data, corrs, by = "cellID")
no_join_data <- anti_join(data, corrs, by = "cellID")
```

8.9.2 Database connection

```
source("scripts/dbcon.R")
```

8.9.3 List tables in database

```
dbListTables(con) %>%
  purrr:keep(., ~ grepl("^CB|^MOBI", .)) %>%
  kable(col.names = "Tables")
```

8.9.4 Write code books

8.9.4.1 CB_license

```
table <- "CB_license" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  licenseID = licenseID,
  license = "Restricted",
  licenseDescription = "Closed data. Special request or purchase required",
  licenseURL = "none"
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check the table

```
tbl(con, table) %>% kable(align = "c")
```

8.9.4.2 CB_sampling_effort

```
table <- "CB_sampling_effort" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  samplingEffortID = 2,
  samplingEffortProtocol = "Sampling events per gridcell",
  samplingEffortUnit = "Sampling events"
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table.

```
tbl(con, table) %>% kable(align = "c")
```

8.9.4.3 CB_model

```
table <- "CB_model" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  occurrenceModelID = "",
  modelName = "",
  predictorVariables = "",
  bibliographicCitation = ""
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table

```
tbl(con, table) %>% kable(align = "c")
```

8.9.4.4 CB_taxonomy

```
table <- "CB_taxonomy" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  scientificNameID = "",
  scientificName = "",
  scientificNameAuthorship = "",
  kingdom = "",
  phylum = "",
  class = "",
  family = "",
  order = "",
  genus = "",
  specificEpitet = "",
  infraspecificEpitet = "",
  taxonRank = ""
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

8.9.4.5 CB_verbatim_name_equivalence

```
table <- "CB_verbatim_name_equivalence" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimIdentificationID = sp_names$verbatimIdentificationID,
  verbatimIdentification = sp_names$verbatim_name,
  datasetID = datasetID,
  identificationReferences = "",
  scientificNameID = NA
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>
  head() %>%
  kable(align = "c")
```

8.9.5 Write dataset tables

8.9.5.1 Dataset

```
table <- "MOBI_dataset" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  datasetID = datasetID,
  datasetName = "Bird atlas The Maritimes",
  datasetPublisher = "Bird Studies Canada",
  datasetPublisherContact = "mba-aom@birdscanada.org",
  licenseID = licenseID,
  rightsHolder = "Bird Studies Canada",
  bibliographicCitation = "BBA1 and BBA2 data: Bird Studies Canada, Environment Canada - Canadian
  citationIdentifier = "",
  provider = "Bird Studies Canada",
  shareable = "NO",
  coauthorshipRequired = "NO",
  coauthors = "",
  coauthorshipSuggested = "Carmen Soria - carmendianasoria@gmail.com; Kateřina Tschernosterová -
  isSamplingEffortReported = "YES",
  isOccurrenceProbabilityAvailable = "NO"
)
```

Write into the database

```
try({
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>% kable(al
```

8.9.5.2 Site

```
table <- "MOBI_site" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_site_23" PARTITION OF "MOBI_site"
FOR VALUES IN (23);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  siteID = grid_table$cell_label,
  scalingID = grid_table$cell_grouping,
  datasetID = datasetID,
  area = grid_table$area,
  croppedArea = grid_table$area_cropped,
  areaUnit = "km2",
  maxLength = grid_table$cell_max_length,
  northSouthLength = grid_table$cell_ns_length,
  eastWestLength = grid_table$cell_ew_length,
  lengthUnit = "km",
  centroidDecimalLongitude = grid_table$cell_long,
  centroidDecimalLatitude = grid_table$cell_lat,
  samplingRepetitions = grid_table$repeated
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(
  con,
  sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))
) %>%
  head(n = 20) %>%
  kable(align = "c")
```

8.9.5.3 Geometry

```
table <- "MOBI_geometry" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_geometry_23" PARTITION OF "MOBI_geometry"
FOR VALUES IN (23);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- grid %>%
  ungroup() %>%
  rename(
    siteID = cell_label,
    scalingID = cell_grouping,
    geometry = geom
  ) %>%
  mutate(
    siteID = as.integer(siteID),
    scalingID = as.integer(scalingID),
    datasetID = as.integer(datasetID),
    footprintSRS = "epsg:4326",
    verbatimFootprintSRS = verbatimFootprintSRS
  ) %>%
  select(
    ., siteID, scalingID, datasetID,
    footprintSRS, verbatimFootprintSRS, geometry
  )

data_table$geometry <- st_cast(data_table$geometry, "MULTIPOLYGON")
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Write records
  st_write(obj = data_table, dsn = con, layer = table, append = T)
  # Remove the data_table from the environment
```

```
  rm(data_table)
})
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>
  head() %>%
  kable(align = "c")
```

8.9.6 Write helper tables

8.9.6.1 Scaling table

This intermediate table enables the joins of records data to the different resolutions of sites.

```
table <- "MOBI_scaling_table" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_scaling_table_23" PARTITION OF "MOBI_scaling_table"
FOR VALUES IN (23);
```

Check table colnames.

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

```
data_table <- data.frame(
  siteID = scaling_table$siteID,
  scalingID = scaling_table$scalingID,
  datasetID = datasetID,
  verbatimSiteID = scaling_table$cellID
)
```

Write into database.

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

8.9.7 Write records tables

8.9.7.1 Event

```
table <- "MOBI_event" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_event_23" PARTITION OF "MOBI_event"
FOR VALUES IN (23);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimSiteID = eff$cellID,
  datasetID = datasetID,
  samplingPeriodID = eff$samplingPeriodID,
  startYear = eff$start_year,
  endYear = eff$end_year,
  samplingEffortID = effortID,
  samplingEffortValue = eff$effort,
  samplingEffort2ID = 4,
  samplingEffort2Value = eff$effort_2
)

if (nrow(unique(data_table)) != nrow(eff)) {
  stop("Stopping execution due to row number mismatch")
} else {
  print("OK")
}

data_table <- filter(data_table, !is.na(verbatimSiteID))
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

8.9.7.2 Presence

```
table <- "MOBI_presence" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_presence_23" PARTITION OF "MOBI_presence"
FOR VALUES IN (23);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimIdentificationID = final_data$verbatimIdentificationID,
  verbatimSiteID = final_data$cellID,
  datasetID = datasetID,
  samplingPeriodID = final_data$samplingPeriodID,
  recordFilter = NA
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
```

```
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

8.9.7.3 Probability

```
table <- "MOBI_probability" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_probability_23" PARTITION OF "MOBI_probability"
FOR VALUES IN (23);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimIdentificationID = "",
  verbatimSiteID = "",
  datasetID = "",
  samplingPeriodID = "",
  probability = ""
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
```

```
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

8.10 Finish session

```
sessionInfo()
```

8.10.1 Clean up workspace

```
dbDisconnect(con)
rm(list = ls())
gc()
.rs.restartR()
```

Chapter 9

European Breeding Birds Atlas

9.1 MOBI Lab team roles

Responsibilities	Name	Date
Data acquisition	Carmen Soria, Petr Keil	
Metadata preparation	Kateřina Tschernosterová	
Data processing	Gabriel Ortega	01/24

9.2 Data providers metadata and co-authorships

Always check the updated information here.

The original documents folder including books, publications or email exchanges are here.

9.3 Data description

- Existing temporal replications of atlases
  - BBA1 - sampling period 1972 - 1995
  - BBA2 - sampling period 2013 - 2017

This dataset corresponds to the first and second edition of the European Breeding Birds Atlas. EBBA2 encompassed all of Europe, including: The European

parts of Russia and Kazakhstan. Transcaucasia (including Armenia and Azerbaijan). The entirety of Turkey and Cyprus. Nearby archipelagos in the Atlantic, Arctic, and Mediterranean seas.

Expanded Coverage Compared to EBBA1: EBBA2 included additional regions such as all of Turkey, Cyprus, and the Canary Islands. Special focus was given to eastern Europe, where previous data were limited or based on expert knowledge rather than field surveys. We received raw data and a special dataset of occurrences considered trusted to evaluate changes between EBBA1 and 2.

Change data coverage: The change maps for EBBA2 were restricted to a well-defined, continuous geographical area based on coverage in EBBA1.

Areas excluded from change maps: Cyprus, the Asian part of Turkey, and the Canary Islands (not included in EBBA1). Most of the European parts of Russia and Kazakhstan, Georgia, Armenia, and Azerbaijan (only partially surveyed in EBBA1).

Criteria for exclusion from change data:

- 50-km squares with more than 70% of their area in these regions were excluded. Special cases in square allocation were considered to ensure consistency. The final change map covered a total of 2,972 50-km squares.
- Richness comletenness: The study analyzed species richness patterns in the European Breeding Bird Atlas 1 (EBBA1) and EBBA2 using Generalized Linear Models (GLMs) with the following considerations: Model Structure: Dependent variable: Observed species richness per 50-km square.

```
- Predictors:
    - Geographic variables (central latitude/longitude, island status, distance to
    - Environmental variables (land cover diversity, total annual precipitation, e
    - Polynomial terms and interactions (e.g., latitude × longitude) were included

    Final models explained 58% deviance (EBBA1) and 75% deviance (EBBA2), adjusted

    Exclusion Criteria: Squares with residuals below the 5th percentile (indicatin
```

9.3.1 Licensing and access

The dataset is classified as “restricted” in the database because it is not covered by an open license, nor is it freely available on the internet. Our copy was purchased from the data holders.

9.3.2 Sampling methodology

All terrestrial habitats were included as potential breeding areas for birds—ranging from urbanized zones to deserts and mountain tops. Artificial structures

near water (e.g., harbors or oil platforms) were considered but not systematically surveyed.

Sampling grid: The study area for EBBA2 was divided into 5,303 grid cells. Most grid cells were regular squares with a side length of 50 km and an area of 2,500 km². Some grid cells were irregular in shape and size, being either larger or smaller than the regular squares. For simplicity, all grid cells were referred to as squares, regardless of their actual shape or size. Data collection efforts varied across Europe for each 50-km square. A measure of survey completeness was developed based on expert knowledge.

9.3.3 Sampling effort

All terrestrial habitats were included as potential breeding areas for birds—ranging from urbanized zones to deserts and mountain tops. Artificial structures near water (e.g., harbors or oil platforms) were considered but not systematically surveyed. Survey completeness was taken as a measure of **effort** or quality of sampling. It was categorized into five levels: - Category 1: 0–20% of habitats in the square well surveyed. - Category 2: 20–40% of habitats well surveyed. - Category 3: 40–60% of habitats well surveyed. - Category 4: 60–80% of habitats well surveyed. - Category 5: 80–100% of habitats well surveyed. When preparing the effort across spatial scales, we took the median effort of all the small cells that formed a bigger one.

9.3.4 Compressed original data folder

- `ebba1_occurrence.csv`: raw occurrence data for EBBA 1.
- `ebba2_data_change_50km.csv`: selected change records for 50 km grid cells.
- `EBBA1 & EBBA2 richness and survey completeness.xlsx`: richness and survey completeness of the two surveys.
- `EBBA1_EBBA2_change_metadata (1).xlsx`: metadata of the change records.
- `EBBA2_50km_occurrence_data.xlsx`: occurrence data for 50 km grid cells in EBBA2.

9.3.5 Docs folder

- `Agreement on EBBA data provision_2023_09_18.pdf`: agreement on data provision between the European Bird Census Council and the Czech University of Life Sciences, MOBIlab team.
- `BEAST_data_management_plan_version_1.1.pdf`: data management plan for BEAST project.
- `Birds_Atlas_EBBA_Data_use_conditions.docx`: data use conditions for EBBA 2.

- `Birds_Atlas_EBBA_Emails.docx`: emails from the European Bird Census Council to MOBIlab team regarding the data provision agreement and data management plan.
- `EBBA2 book chapter Methods.pdf`: Methods section of EBBA2.
- `EBBA2 book chapter Patterns of distribution and change.pdf`: Patterns of distribution and change between EBBA1 and EBBA2.
- `Wintermonitoring.pdf`: Report on winter monitoring data. Not included in our database.

9.3.6 Checked data folder

- `Birds_atlas_EBBA_beast_data_CHANGE.rds`: occurrences extracted from change data.
- `Birds_atlas_EBBA_beast_data.rds`: occurrences extracted from raw data.
- `Birds_atlas_EBBA_cells_corr.rds`: correspondence table to assign original grid cells to coarser resolutions.
- `Birds_atlas_EBBA_grid.gpkg`: spatial geometries.
- `Birds_atlas_EBBA_spnames_INFO_change_data.csv`: species names and synonyms for change data.
- `Birds_atlas_EBBA_spnames_INFO.csv`: species names and synonyms for raw occurrence data.
- `Previous data processing documents.zip`: just for reference. Old processed files.

9.4 Data standardization/processing comments

There are NAs in the effort column in both sampling periods. NAs are located in southern Ukraine and Ihlas Selvagens (Portugal). Effort is a qualitative estimation of habitat survey completeness ranging from 1 to 5. The effort was prepared for every resolution as the median of effort in every group of cells.

9.5 Libraries

```
pacman::p_load(
  sf, terra, tidyverse, tidyterra, knitr,
  tictoc, RSQLite, DBI, dbplyr, parallel
)
```

9.6 Grid preparation

This grid was derived from the original EBBA grid, but manually prepared in QGIS to match different cell resolutions. Since the original grid spans more

than one UTM zone and contains non square cells, it was not possible to keep the cells squared while coarsening the resolutions.

9.7 Import the processed grid

```
grid <- vect("data/EBBA/modified_grid.gpkg") %>% mutate(across(matches("^X"), dense_rank))
crs(grid)
```

9.7.1 Calculate cell area

Get the landmass of Europe

```
tempCRS <- crs(grid)
ext <- project(grid, "epsg:4326") %>% ext()

adm_boundary <- rnaturalearth::ne_countries(
  continent = c("europe", "asia"),
  returnclass = "sv", scale = "large"
) %>%
  crop(., ext)

minor_islands <- rnaturalearth::ne_download(
  type = "minor_islands",
  category = "physical", scale = "large",
  destdir = "/tmp/EBBA/", returnclass = "sv"
) %>%
  crop(., ext)

adm_boundary <- rbind(adm_boundary, minor_islands) %>%
  buffer(., width = 1000) %>%
  project(., tempCRS) %>%
  aggregate()

map <- plet(adm_boundary) %>%
  plet(grid, map = .)
map

rm(minor_islands)
```

Intersect the grid with the landmass of Europe

```
int <- terra::intersect(select(grid, cellID), adm_boundary)
int <- group_by(int, cellID) %>% summarise()
int$cropped_area <- expanse(int, unit = "km")
grid <- as.data.frame(int) %>%
  left_join(grid, ., by = "cellID")
```

Calculate cell areas with and without correction by landmasses or country borders

```
grid$cell_area <- expanse(grid, unit = "km")
grid$cell_area_proportion <- grid$cropped_area / grid$cell_area %>% round(., digits = 2
```

9.7.2 Check cell areas

```
hist(grid$cell_area)
```

```
grid <- read_csv("data/EBBA/samp_cells.csv") %>%
  select(cellID, starts_with("T")) %>%
  left_join(grid, ., by = "cellID") %>%
  mutate(across(starts_with("T"), ~ if_else(is.na(.x), 0, .x)))
```

```
# Pivot cells to a longer format for easier use in models.
final_grids <- grid %>%
  as_sf() %>%
  pivot_longer(matches("^X\\d+\\b"),
    names_to = "cell_grouping",
    values_to = "cell_label"
  ) %>%
  mutate(cell_grouping = str_extract(cell_grouping, "\\d+") %>%
    as.numeric())

final_grids <- split(final_grids, final_grids$cell_grouping)
```

Aggregate cells and summarize the area. Areas with an "s" at the end are "the area sampled in every atlasing period". It is not the truly sampled area but a correction according to the sampled cells informed in each time period.

```
# Define the function to process each grid
summarise_grid_cells <- function(x) {
  res <- x %>%
    select(matches("cell_*|cropp*|T\\d")) %>%
    group_by(cell_grouping, cell_label) %>%
    summarise(
      area = sum(cell_area, na.rm = TRUE),
      area1s = sum(T1 * cell_area, na.rm = TRUE),
      area2s = sum(T2 * cell_area, na.rm = TRUE),
      area_cropped = sum(cropped_area, na.rm = TRUE)
    ) %>%
    ungroup() %>%
    mutate(
      cell_perimeter_km = as.numeric(units::set_units(st_length(st_cast(st_make_valid(
```

```
    )
  return(res)
}

cl <- makeCluster(getOption("cl.cores", 10))

# Load necessary packages on each worker
clusterEvalQ(cl, {
  library(tidyverse)
  library(terra)
  library(sf)
})

final_grids <- parSapply(cl, final_grids, summarise_grid_cells, simplify = FALSE, USE.NAMES = TRU

stopCluster(cl)
```

Functions to add additional variables of cell shapes

```
# Function to get cell shape attributes

fdistances <- function(x, type = NULL) {
  x <- vect(x)
  pol <- as.data.frame(crds(as.points(ext(project(x, "epsg:4326")))))
  if (type == "ew") {
    sw <- pol %>%
      summarise(x = min(x), y = min(y)) %>%
      as.matrix()
    se <- pol %>%
      summarise(x = max(x), y = min(y)) %>%
      as.matrix()
    res <- distance(sw, se, lonlat = T)[[1]] / 1000
  }
  if (type == "ns") {
    sw <- pol %>%
      summarise(x = min(x), y = min(y)) %>%
      as.matrix()
    nw <- pol %>%
      summarise(x = min(x), y = max(y)) %>%
      as.matrix()
    res <- distance(sw, nw, lonlat = T)[[1]] / 1000
  }
  if (type == "max") {
    res <- distance(pol, lonlat = T) %>% max()
    res <- res / 1000
  }
```

```
  res <- as.numeric(res)
  return(res)
}
```

Add cell max length, distance south-north, and distance east-west.

```
tic()

add_cell_lengths <- function(x) {
  res <- x
  list <- res %>%
    select(cell_label)
  list <- terra::split(list, list$cell_label)

  res$cell_max_length <- lapply(list, function(.x) fdistances(.x, type = "max")) %>%
    do.call(rbind, .) %>%
    as.numeric()

  res$cell_ns_length <- lapply(list, function(.x) fdistances(.x, type = "ns")) %>%
    do.call(rbind, .) %>%
    as.numeric()

  res$cell_ew_length <- lapply(list, function(.x) fdistances(.x, type = "ew")) %>%
    do.call(rbind, .) %>%
    as.numeric()

  res <- st_transform(res, st_crs(x))
  return(res)
}

# Create a cluster
cl <- makeCluster(getOption("cl.cores", 10))

# Load necessary packages on each worker
clusterEvalQ(cl, {
  library(dplyr)
  library(terra)
  library(sf)
})

# Export the function to the cluster
clusterExport(cl, varlist = "fdistances")

# Use parSapply with the cluster
final_grids <- parSapply(cl, final_grids, add_cell_lengths, simplify = FALSE, USE.NAMES
```

```
# Stop the cluster
stopCluster(cl)

toc()
```

Get longitude and latitude coordinates of each cell.

```
final_grids <- lapply(final_grids, function(x) {
  x <- vect(x)
  x <- project(x, "epsg:4326")
  res <- centroids(x) %>%
    geom() %>%
    bind_spat_cols(x, .)
  res <- rename(res, cell_lat = y, cell_long = x)
  res <- select(res, -geom, -part, -hole)
  return(res)
})
```

Plot grids

```
# for (x in final_grids) {
#   plot(x)
# }
```

9.8 Export grids

```
wide_out_grid <- "data/EBBA/grids_correspondence_table.csv"
file.remove(wide_out_grid)

grid %>%
  st_drop_geometry() %>%
  select(matches("cellID|^X\\d")) %>%
  as.data.frame() %>%
  write_csv(., wide_out_grid)
```

```
out_grid <- "data/EBBA/final_grids.gpkg"
file.remove(out_grid)

# Define the grid layers you want to write
grid_layers <- c(1, 2, 4, 8, 16, 32, 64, 128)

# Loop through each grid layer and write to the output
for (layer in grid_layers) {
  st_write(st_as_sf(final_grids[[as.character(layer)]]), out_grid, layer = paste0("cell", layer,
}
```

9.9 Adding records to the database

9.9.1 Data

Import the processed data and grid.

```
# Function to split, sort and rejoin every string in a vector
splitSort <- function(vector, splpat, joinpat) {
  res <- sapply(vector, function(x) {
    if (is.na(x)) {
      return(NA)
    } else {
      split <- stringr::str_split_1(x, pattern = splpat)
      order <- stringr::str_sort(split, decreasing = T)
      paste <- paste0(order, collapse = joinpat)
      return(paste)
    }
  })
}

# Import raw data
data <- list("data/EBBA/data1.csv", "data/EBBA/data2.csv") %>%
  lapply(., function(file) read_csv(file)) %>%
  do.call(rbind, .) %>%
  mutate(
    verbatimIdentificationID = dense_rank(verbatim_name),
    samplingPeriodID = dense_rank(start_year),
    composite_cells = str_detect(cellID, "-"),
    cellID = splitSort(cellID, "-", "-")
  )

# Import change data
change <- read_csv("data/EBBA/prepared_change.csv", col_select = -1) %>%
  mutate(
    composite_cells = str_detect(cellID, "-"),
    cellID = splitSort(cellID, "-", "-")
  )

data <- left_join(data, change)
no_join <- anti_join(change, data)

sp_names <- data %>%
  select(verbatimIdentificationID, verbatim_name) %>%
  unique()

eff <- read_csv("data/EBBA/effort.csv", col_select = -1) %>%
```

```
  mutate(samplingPeriodID = dense_rank(start_year)) %>%
  group_by(across(-effort)) %>%
  summarise(effort = max(effort))

corrs <- read_csv("data/EBBA/grids_correspondence_table.csv")

scaling_table <- corrs %>%
  pivot_longer(-cellID, names_to = "scalingID", values_to = "siteID") %>%
  mutate(scalingID = str_remove_all(scalingID, "^X") %>%
    as.numeric())

grid <- st_layers("data/EBBA/final_grids.gpkg")$name %>%
  mclapply(., function(x) {
    st_read("data/EBBA/final_grids.gpkg",
      layer = x, quiet = T
    )
  }, mc.cores = 10) %>%
  do.call(rbind, .)

grid_table <- st_drop_geometry(grid)
```

9.9.1.1 Periodic table check

```
# Check that all cellID in data are also in the corrs cellID
if (length(setdiff(unique(data$cellID), unique(corrs$cellID))) == 0) {
  print("OK")
} else {
  print("FAIL")
}
# Check that number of rows in grid and corrs are the same
nrow(filter(grid, cell_grouping == 1)) == nrow(corrs)

# Check that years and sampling periods are the same in data and eff
nrow(setdiff(
  unique(select(data, start_year, samplingPeriodID)),
  unique(select(eff, start_year, samplingPeriodID))
))

# Check uniqueness of scaling_table
nrow(unique(scaling_table)) == nrow(scaling_table)

# RESULTS SHOULD BE: OK, TRUE, 0, TRUE
```

9.9.1.2 Atlas variables

```
datasetID <- 26
effortID <- 2
licenseID <- 0
shareable <- 0
verbatimFootprintSRS <- "epsg:3035"
```

The previous step indicates that there are cells not joined to the grid. Such cells correspond to locations in the sea or outside the countries border.

```
final_data <- inner_join(data, corrs, by = "cellID")
no_gridcell_data <- anti_join(data, corrs, by = "cellID")
```

9.9.2 Database connection

```
if (exists("con")) {
  dbDisconnect(con)
}

con <- dbConnect(RSQLite::SQLite(),
  dbname = "MOBI_rep_atlases.sqlite"
)

# Enable SpatiaLite extension
dbExecute(con, "SELECT load_extension('mod_spatialite')")

knitr::opts_chunk$set(connection = con)
```

9.9.3 List tables in database

```
dbListTables(con) %>%
  purrr::keep(., ~ grepl("^CB|^MOBI", .)) %>%
  kable(col.names = "Tables")
```

9.9.4 Write code books

9.9.4.1 CB_license

```
table <- "CB_license" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
```

```
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  licenseID = licenseID,
  license = "Restricted",
  licenseDescription = "Closed data. Special request or purchase required",
  licenseURL = "none"
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check the table

```
tbl(con, table) %>% kable(align = "c")
```

9.9.4.2 CB_sampling_effort

```
table <- "CB_sampling_effort" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  samplingEffortID = effortID,
  samplingEffortProtocol = "Habitats survey completeness",
  samplingEffortUnit = "Survey completeness"
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table.

```
tbl(con, table) %>% kable(align = "c")
```

9.9.4.3 CB_model

```
table <- "CB_model" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  occurrenceModelID = "",
  modelName = "",
  predictorVariables = "",
  bibliographicCitation = ""
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table

```
tbl(con, table) %>% kable(align = "c")
```

9.9.4.4 CB_taxonomy

```
table <- "CB_taxonomy" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  scientificNameID = "",
  scientificName = "",
  scientificNameAuthorship = "",
  kingdom = "",
  phylum = "",
  class = "",
  family = "",
  order = "",
  genus = "",
  specificEpitet = "",
  infraspecificEpitet = "",
  taxonRank = ""
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

9.9.4.5 CB_verbatim_name_equivalence

```
table <- "CB_verbatim_name_equivalence" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimIdentificationID = sp_names$verbatimIdentificationID,
  verbatimIdentification = sp_names$verbatim_name,
  datasetID = datasetID,
  identificationReferences = "",
  scientificNameID = 0
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
```

```
  dbExecute(con, paste("DELETE FROM", table, "WHERE datasetID =", datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste("SELECT * FROM", table, "WHERE datasetID =", datasetID))) %>% kable
```

9.9.5 Write dataset tables

9.9.5.1 Dataset

```
table <- "MOBI_dataset" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  datasetID = datasetID,
  datasetName = "European Breeding Birds Atlas",
  datasetPublisher = "European Bird Census Council",
  datasetPublisherContact = "Sergi Herrando (vice-chair, European Bird Census Council)
  licenseID = licenseID,
  rightsHolder = "European Bird Census Council",
  bibliographicCitation = "EBBA1 book: Hagemeijer, W.J.M. & Blair, M.J. (1997). The EBC
EBBA1 dataset/web: Hagemeyer W, Blair M, Loos W (2016). EBCC Atlas of European Breeding
EBBA2 book: Keller, V., Herrando, S., Voříšek, P., Franch, M., Kipson, M., Milanesi, P
EBBA2 web: EBCC (2022). European Breeding Bird Atlas 2 website. European Bird Census Co
  citationIdentifier = "",
  provider = "European Bird Census Council",
  shareable = "NO",
  coauthorshipRequired = "YES",
  coauthors = "Sergi Herrando  - ornitologia@ornitologia.org; Petr Voříšek - Vorisek@el
  coauthorshipSuggested = "Carmen Soria - carmendianasoria@gmail.com; Kateřina Tscherno
  isSamplingEffortReported = "YES",
```

```
  isOccurrenceProbabilityAvailable = "NO",
  occurrenceModelID = "",
  recordFilterMeaning = "Trusted occurrences selected to evaluate changes in species distributior
)
```

Write into the database

```
try({
  dbExecute(con, paste("DELETE FROM", table, "WHERE datasetID =", datasetID))
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table.

```
tbl(con, sql(paste("SELECT * FROM", table, "WHERE datasetID =", datasetID))) %>% kable(align = "c
```

9.9.5.2 Site

```
table <- "MOBI_site" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  siteID = grid_table$cell_label,
  scalingID = grid_table$cell_grouping,
  datasetID = datasetID,
  area = grid_table$area,
  croppedArea = grid_table$area_cropped,
  areaUnit = "km2",
  maxLength = grid_table$cell_max_length,
  northSouthLength = grid_table$cell_ns_length,
  eastWestLength = grid_table$cell_ew_length,
  lengthUnit = "km",
  centroidDecimalLongitude = grid_table$cell_long,
  centroidDecimalLatitude = grid_table$cell_lat
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste("DELETE FROM", table, "WHERE datasetID =", datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(
  con,
  sql(paste("SELECT * FROM", table, "WHERE datasetID =", datasetID))
) %>%
  head(n = 20) %>%
  kable(align = "c")
```

9.9.5.3 Geometry

```
table <- "MOBI_geometry" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- grid %>%
  rename(
    siteID = cell_label,
    scalingID = cell_grouping,
    geometry = geom
  ) %>%
  mutate(
    siteID = as.integer(siteID),
    scalingID = as.integer(scalingID),
    datasetID = as.integer(datasetID),
    footprintSRS = "epsg:4326",
    verbatimFootprintSRS = verbatimFootprintSRS
  ) %>%
```

```
  select(
    ., siteID, scalingID, datasetID,
    footprintSRS, verbatimFootprintSRS, geometry
  )

data_table$geometry <- st_cast(data_table$geometry, "MULTIPOLYGON")

st_write(data_table, "/tmp/grid.gpkg", delete_dsn = T)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste("DELETE FROM", table, "WHERE datasetID =", datasetID))

  # Remove the data_table from the environment
  rm(data_table)
})
```

```
ogr2ogr -update -append -f SQLite -dsco SPATIALITE=YES -nln MOBI_geometry MOBI_rep_atlases.sqlite
```

Check table.

```
tbl(
  con,
  sql(paste("SELECT * FROM", table, "WHERE datasetID =", datasetID))
) %>%
  head(n = 20) %>%
  kable(align = "c")
```

9.9.6 Write helper tables

9.9.6.1 Scaling table

This intermediate table enables the joins of records data to the different resolutions of sites.

```
table <- "MOBI_scaling_table" ## Table of interest
```

Check table colnames.

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

```
data_table <- data.frame(
  siteID = scaling_table$siteID,
```

```
  scalingID = scaling_table$scalingID,
  datasetID = datasetID,
  verbatimSiteID = scaling_table$cellID
)
```

Write into database.

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste("DELETE FROM", table, "WHERE datasetID =", datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(
  con,
  sql(paste("SELECT * FROM", table, "WHERE datasetID =", datasetID))
) %>%
  head(n = 20) %>%
  kable(align = "c")
```

9.9.7 Write records tables

9.9.7.1 Event

```
table <- "MOBI_event" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimSiteID = eff$cellID,
  datasetID = datasetID,
  samplingPeriodID = eff$samplingPeriodID,
  startYear = eff$start_year,
```

```
  endYear = eff$end_year,
  samplingEffortID = effortID,
  samplingEffortValue = eff$effort
)

if (nrow(unique(data_table)) != nrow(eff)) {
  stop("Stopping execution due to row number mismatch")
} else {
  print("OK")
}
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste("DELETE FROM", table, "WHERE datasetID =", datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(
  con,
  sql(paste("SELECT * FROM", table, "WHERE datasetID =", datasetID))
) %>%
  head(n = 20) %>%
  kable(align = "c")
```

9.9.7.2 Presence

```
table <- "MOBI_presence" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimIdentificationID = final_data$verbatimIdentificationID,
  verbatimSiteID = final_data$cellID,
  datasetID = datasetID,
  samplingPeriodID = final_data$samplingPeriodID
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste("DELETE FROM", table, "WHERE datasetID =", datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(
  con,
  sql(paste("SELECT * FROM", table, "WHERE datasetID =", datasetID))
) %>%
  head(n = 20) %>%
  kable(align = "c")
```

9.9.7.3 Probability

```
table <- "MOBI_probability" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.frame() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.frame(
  verbatimIdentificationID = "",
  verbatimSiteID = "",
  datasetID = "",
  samplingPeriodID = "",
```

```
  probability = ""
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste("DELETE FROM", table, "WHERE datasetID =", datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(
  con,
  sql(paste("SELECT * FROM", table, "WHERE datasetID =", datasetID))
) %>%
  head(n = 20) %>%
  kable(align = "c")
```

9.10 Finish session

```
sessionInfo()
```

9.10.1 Clean up workspace

```
dbDisconnect(con)
rm(list = ls())
gc()
.rs.restartR()
```

Chapter 10

New York Birds Atlas

10.1 MOBI Lab team roles

Responsibilities	Name	Date (MM/YY)
Data acquisition	Petr Keil	02/21
Metadata preparation	Kateřina Tschernosterová	04/24
Data standardization		02/24
Data processing	Gabriel Ortega	10/24

10.2 Data providers, metadata and co-authorship

Always check the updated information here, in the table Atlases status

The original documents folder including books, publications or email exchanges are here.

10.3 Data description

- Existing temporal replications of atlases
  - BBA1 - sampling period 1980 - 1985
  - BBA2 - sampling period 2000 - 2005
  - BBA3 - next atlas is in progress, presumed sampling period 2020 - 2024
- Observation data by grid cell, by species and by sampling event year (for some events in BBA2 there is also date of observation - day, month and year)

- Presence is reported, no data on abundance or absence
- Original data detail - Sampling cell ID, Link to the pdf map of sampling cell, Scientific species name, Common species name, NYS Protection Status, Breeding Behavior, Breeding status, Effort in hours per sampling cell, Total number of observed species per sampling cell
- Atlases data for BBA1 and BBA2 (together) can be downloaded here: https://data.ny.gov/Energy-Environment/Breeding-Bird-Atlases/vk8g-ypxi/about_data
- An official website of New York State Breeding Bird Atlases is available here: https://dec.ny.gov/nature/animals-fish-plants/birds/breeding-bird-atlas/past-data-maps
- Effort data and sampling cell shapefile were provided by email sent by Julie Hart (atlas coordinator, New York State Department of Environmental Conservation).
- BBA3 website is here: https://ebird.org/atlasny/about and https://dec.ny.gov/nature/animals-fish-plants/birds/breeding-bird-atlas

10.3.1 Licensing and access

- Julie Hart's email says the data are open in the sense that are freely available to download and the data use conditions do not restrict uses (https://extapps.dec.ny.gov/cfmx/extapps/bba/DataUsePolicy.cfm). However, the dataset webpage explicitly says that the data are not covered by any license, therefore, they are restricted by default.
- For detailed information or specific permissions regarding data usage, contact New York State Department of Environmental Conservation, New York Natural Heritage Program (Julie Hart - atlas coordinator, julie.hart@dec.ny.gov) or NYSBBAIII (nybbba3@gmail.com)

10.3.2 Sampling methodology

- The mapping for the Atlas is based on a grid system that divides the state into 10 km x 10 km map squares, each of which is identified by a four-digit number. Each map square is in turn divided into four 5 km x 5 km (9.65 sq mi) Atlas blocks, which are designated by the letters A, B, C, and D (Figure 1). The Atlas block is the basic unit for conducting surveys for the Atlas. (BBA2 handbook, p.2)
- You should set a goal of recording at least 76 species in each survey block, and confirming breeding for about half of those species. (BBA2 handbook, p.3)
- Paid field workers were used in both atlases to ensure coverage in the remote blocks of some regions. Approximately 25 percent of 1980-1985

Atlas coverage was completed by paid workers. About 20 percent of 2000-2005 Atlas coverage was completed by paid workers. (BBA data overview, p.3)

- Though the statewide average of time spent in a block was similar in the two Atlases, this variability among observers must not be discounted when examining changes in distribution. (BBA data overview, p.3)
- While a record in the database indicates the presence of a species in a block, the absence of a record does not necessarily indicate absence of the species in that block. (BBA data overview, p.3)

10.3.3 Sampling effort

- Dates of observations were not collected during the first Atlas, but in the second Atlas, observers were instructed to record the first date and breeding code for an observed species. If a subsequent visit to the block resulted in an observation that allowed the use of a higher breeding code, the lower breeding code and associated date were replaced. (BBA data overview, p.3)
- Atlasers agreed to survey one or more blocks and were expected to spend at least eight hours in each block, visiting each habitat represented and including at least one nighttime visit to document nocturnal species. (BBA data overview, p.3)
- During the first Atlas, a block was considered to have been adequately surveyed when a total of at least 76 species had been identified from the block with half confirmed as breeders. This same standard was provided to atlasers in the second Atlas project as well. (BBA data overview, p.3)

10.4 Input Data

10.4.1 Compressed original data folder

- Birds_Atlas_New York_1-2_Data.csv - file with observation data for both atlases
- Birds_Atlas_New York_1_Data.csv - file with observation data for BBA1
- Birds_Atlas_New York_1_Survey effort.xlsx - file with sampling effort in hours per sampling cell for BBA1
- Birds_Atlas_New York_2_Data.csv - file with observation data for BBA2
- Birds_Atlas_New York_2_Survey effort.xlsx - file with sampling effort in hours per sampling cell for BBA2
- Birds_Atlas_New York_1-2_Received data.zip - file with observation data for both atlases together provided by Julia Hart

- BBA_2000_blocks shapefile, grid.qgz, modified_grid.gpkg - original grid shapefile for sampling area

10.4.2 Docs folder

- Birds_Atlas_New York_1-2_Download data confirmation.pdf - printscreen of downloading website
- Birds_Atlas_New York_1-2_Data Dictionary.pdf - file with explanation of Data Label, Data Type and Data Description
- Birds_Atlas_New York_1-2_Data Overview.pdf - short general data description, methodology, statistics and limitations of the data
- Birds_Atlas_New York_2_Handbook for workers.pdf - guide for atlasers with practical information and methodology explanation
- emails_Birds_Atlas_US_New York_terms of use.docx - email conversation about data acqusition and data use conditions

10.4.3 Checked data folder xxxxxxxxxxxxxxxxxxx Gabriel check

- Birds_Atlas_New_York_beast_data.rds - R object with atlas data
- Birds_Atlas_New_York_cells_corr.rds - R object sampling cells coordinates
- Birds_Atlas_New_York_data_INFO_spnames.csv - species names check
- Birds_Atlas_New_York_grid.gpkg

10.5 Data standardization and processing comments xxxxxxxxxxxxxxxx Gabriel check

XX Use the notes/methods section to put notes on data processing.

10.6 Libraries

```
pacman::p_load(
  sf, terra, data.table, tidyverse, tidyterra, tidytable, knitr,
  tictoc, RPostgres, DBI, dbplyr, parallel,
  geodata
)
```

10.6.1 Preparation

```
# Variables
seldata <- "data/4processedAtlases/Birds_Atlas_New_York"
datasetID <- 6
licenseID <- 1
verbatimFootprintSRS <- "epsg:26918"

# Data processing
dataset <- readRDS(paste0(seldata, "_beast_data.rds")) %>% as.data.table()

spnamesinfo <- read.csv(paste0(seldata, "_data_INFO_spnames.csv")) %>%
  as.data.table() %>%
  mutate(cellID = str_split(cellID, pattern = ",")) %>%
  unnest(cellID) %>%
  select(cellID, name_in_data, original_verbatim_name) %>%
  mutate(cellID = cellID) %>%
  unique()

grid <- st_layers(paste0(seldata, "_grid.gpkg"))$name %>%
  sapply(., USE.NAMES = T, simplify = F, function(x) {
    st_read(paste0(seldata, "_grid.gpkg"), layer = x)
  })

corrs <- readRDS(paste0(seldata, "_cells_corr.rds")) %>%
  as.data.table() %>%
  pivot_longer(-cellID, names_to = "cell_grouping", values_to = "cell_label") %>%
  mutate(cell_grouping = as.numeric(cell_grouping))

full_data <- left_join(dataset, corrs, by = c("cell_grouping", "cell_label"), relationship = "mar
  left_join(., filter(spnamesinfo, !is.na(cellID)), by = c("cellID" = "cellID", "verbatim_name" =
  mutate(verbatim_name = ifelse(is.na(original_verbatim_name), verbatim_name, original_verbatim_n
  select(-original_verbatim_name) %>%
  unique() %>%
  mutate(
    datasetID = datasetID,
    licenseID = licenseID,
    verbatimIdentificationID = dense_rank(verbatim_name),
    samplingPeriod = dense_rank(start_year),
    effort = ifelse(effort == 0, NA, effort),
    samp_effort_type = ifelse(is.na(samp_effort_type), NA, 4)
  ) %>%
  unique()

gc()
```

10.6.2 Database connection

```
source("scripts/dbcon.R")
```

10.6.3 List tables in database

```
dbListTables(con) %>%
  purrr::keep(., ~ grepl("^CB|^MOBI", .)) %>%
  kable(col.names = "Tables")
```

10.6.4 Write code books

10.6.4.1 CB_license

```
table <- "CB_license" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  licenseID = licenseID,
  license = "Closed",
  licenseDescription = "Closed data. Special request or purchase required",
  licenseURL = "none"
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check the table

```
tbl(con, table) %>% kable(align = "c")
```

10.6.4.2 CB_sampling_effort

```
table <- "CB_sampling_effort" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  samplingEffortID = effortID,
  samplingEffortProtocol = "Visits per gridcell",
  samplingEffortUnit = "Visits"
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table.

```
tbl(con, table) %>% kable(align = "c")
```

10.6.4.3 CB_model

```
table <- "CB_model" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  occurrenceModelID = "",
  modelName = "",
  predictorVariables = "",
  bibliographicCitation = ""
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table

```
tbl(con, table) %>% kable(align = "c")
```

10.6.4.4 CB_taxonomy

```
table <- "CB_taxonomy" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  scientificNameID = "",
  scientificName = "",
  scientificNameAuthorship = "",
  kingdom = "",
  phylum = "",
  class = "",
  family = "",
  order = "",
  genus = "",
  specificEpitet = "",
  infraspecificEpitet = "",
  taxonRank = ""
)
```

Write into the database

```
try({
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

10.6.4.5 CB_verbatim_name_equivalence

```
table <- "CB_verbatim_name_equivalence" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  verbatimIdentificationID = full_data$verbatimIdentificationID,
  verbatimIdentification = full_data$verbatim_name,
  datasetID = full_data$datasetID,
  identificationReferences = "",
  scientificNameID = NA
) %>% unique()
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

10.6.5 Write dataset tables

10.6.5.1 Dataset

```
table <- "MOBI_dataset" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  datasetID = datasetID,
  datasetName = "New York State Breeding Bird Atlas",
  datasetPublisher = "New York State Department of Environmental Conservation, New Yor
  datasetPublisherContact = "nybbba3@gmail.com | julie.hart@dec.ny.gov",
  licenseID = licenseID,
  rightsHolder = "New York State Department of Environmental Conservation",
  bibliographicCitation = "BBA1 data: New York State Breeding Bird Atlas [Internet]. 1
  citationIdentifier = "",
  provider = "New York State Department of Environmental Conservation",
  shareable = "NO",
  coauthorshipRequired = "NO",
  coauthors = "",
  coauthorshipSuggested = "Carmen Soria - carmendianasoria@gmail.com | Kateřina Tscher
  isSamplingEffortReported = "YES",
  isOccurrenceProbabilityAvailable = "NO"
)
```

Write into the database

```
try({
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))
  copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %
```

10.6.5.2 Site

```
table <- "MOBI_site" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_site_6" PARTITION OF "MOBI_site"
FOR VALUES IN (6);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  siteID = full_data$cell_label,
  scalingID = full_data$cell_grouping,
  datasetID = full_data$datasetID,
  area = full_data$area,
  croppedArea = full_data$area_cropped,
  areaUnit = "km2",
  maxLength = NA,
  northSouthLength = NA,
  eastWestLength = NA,
  lengthUnit = "km",
  centroidDecimalLongitude = full_data$cell_long,
  centroidDecimalLatitude = full_data$cell_lat,
  samplingRepetitions = full_data$repeated
) %>% unique()
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(
  con,
  sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))
) %>%
  head(n = 20) %>%
  kable(align = "c")
```

10.6.5.3 Geometry

```
table <- "MOBI_geometry" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_geometry_6" PARTITION OF "MOBI_geometry"
FOR VALUES IN (6);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- grid %>%
  dplyr::bind_rows() %>%
  ungroup() %>%
  rename(
    siteID = cell_label,
    scalingID = cell_grouping,
    geometry = geom
  ) %>%
  mutate(
    siteID = as.integer(siteID),
    scalingID = as.integer(scalingID),
    datasetID = as.integer(datasetID),
    footprintSRS = "epsg:4326",
    verbatimFootprintSRS = verbatimFootprintSRS
  ) %>%
  select(
    ., siteID, scalingID, datasetID,
    footprintSRS, verbatimFootprintSRS, geometry
  )

data_table$geometry <- st_cast(data_table$geometry, "MULTIPOLYGON")
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Write records
  st_write(obj = data_table, dsn = con, layer = table, append = T)
```

```
  # Remove the data_table from the environment
  # rm(data_table)
})
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

10.6.6 Write helper tables

10.6.6.1 Scaling table

This intermediate table enables the joins of records data to the different resolutions of sites.

```
table <- "MOBI_scaling_table" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_scaling_table_6" PARTITION OF "MOBI_scaling_table"
FOR VALUES IN (6);
```

Check table colnames.

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

```
data_table <- data.table(
  siteID = full_data$cell_label,
  scalingID = full_data$cell_grouping,
  datasetID = datasetID,
  verbatimSiteID = full_data$cellID
) %>% unique()
```

Write into database.

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})
```

```
# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>
  head() %>%
  kable(align = "c")
```

10.6.7 Write records tables

10.6.7.1 Event

```
table <- "MOBI_event" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_event_6" PARTITION OF "MOBI_event"
FOR VALUES IN (6);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- filter(full_data, cell_grouping == 1) %>%
  mutate(
    verbatimSiteID = cellID,
    datasetID = datasetID,
    samplingPeriodID = samplingPeriod,
    startYear = start_year,
    endYear = end_year,
    samplingEffortID = samp_effort_type,
    samplingEffortValue = effort
  ) %>%
  select(
    verbatimSiteID,
    datasetID,
    samplingPeriodID,
    startYear,
    endYear,
    samplingEffortID,
    samplingEffortValue
```

```
  ) %>%
  unique()

# if (nrow(unique(data_table)) != nrow(eff)) {
#   stop("Stopping execution due to row number mismatch")
# } else {
#   print("OK")
# }

data_table <- filter(data_table, !is.na(verbatimSiteID))
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

10.6.7.2 Presence

```
table <- "MOBI_presence" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_presence_6" PARTITION OF "MOBI_presence"
FOR VALUES IN (6);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- filter(full_data, cell_grouping == 1) %>%
  mutate(
    verbatimIdentificationID = verbatimIdentificationID,
    verbatimSiteID = cellID,
    datasetID = datasetID,
    samplingPeriodID = samplingPeriod,
    recordFilter = NA
  ) %>%
  select(
    verbatimIdentificationID,
    verbatimSiteID,
    datasetID,
    samplingPeriodID,
    recordFilter
  ) %>%
  unique()
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %
  head() %>%
  kable(align = "c")
```

10.6.7.3 Probability

```
table <- "MOBI_probability" ## Table of interest
```

Create partition

```
CREATE TABLE "MOBI_probability_6" PARTITION OF "MOBI_probability"
FOR VALUES IN (6);
```

Check table colnames

```
tbl(con, table) %>%
  as.data.table() %>%
  colnames() %>%
  paste(., ' = ""', collapse = ",\n") %>%
  cat()
```

Create data table

```
data_table <- data.table(
  verbatimIdentificationID = "",
  verbatimSiteID = "",
  datasetID = "",
  samplingPeriodID = "",
  probability = ""
)
```

Write into the database

```
try({
  # Remove records where datasetID is the current dataset
  dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

  # Copy data to the database
  copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
  head() %>%
  kable(align = "c")
```

Chapter 11

Czech Birds Atlas

11.1 MOBI Lab team roles

Responsibilities	Name	Date (MM/YY)
Data acquisition	Francois Leroy	
Metadata preparation	Kateřina Tschernosterová	
Data standardization		
Data processing	Gabriel Ortega	

11.2 Data providers metadata and co-authorships

Always check the updated information here, in the table Atlases status

The original documents folder including books, publications or email exchanges are here.

11.3 Data description

- Existing temporal replications of atlases
  - BBA1 - sampling period 1973 - 1977
  - BBA2 - sampling period 1985 - 1989
  - BBA3 - sampling period 2001 - 2003
  - BBA4 - sampling period 2014 - 2017

The first mapping (1973–77) used 10×10 km squares, with data collected on paper maps.

Later mappings (1985–89, 2001–03, 2014–17) used a unified grid of 12×11.1 km squares (approx. 133 km²), resulting in 678 quadrats for the Czech Republic. Some border squares were merged or excluded for consistency.

The 2014–17 atlas transitioned to electronic data collection via the Faunistic Database of the Czech Ornithological Society (atlas.birds.cz), allowing real-time data entry and access for collaborators.

We received a set of excel and shp files containing the records, grid cells, and identifiers of reporting cards and volunteers.

The original data columns are:

- BBA1: ID, OID, BIRD_LOK, MAX_N_KODH, KVADRAT, SPECIES, ID_TAX, KOD_HN, ID_CODE, ATLAS_code, ACTIVITY_code.
- BBA2: ID, ID_Z94, SPECIES, KVADRAT, KATEGORIE, ID_TAX, ATLAS_code, ACTIVITY_code.
- BBA3: ID, ID_Z94, SPECIES, KVADRAT, KATEGORIE, ID_TAX, ATLAS_code, ACTIVITY_code.
- BBA4: ID, ID_Z94, SPECIES, KVADRAT, KATEGORIE, ID_TAX, ATLAS_code, ACTIVITY_code.

The datasets were provided by Vladimír Bejček & Karel Šťastný (bejcek@fzp.czu.cz, stastny@fzp.czu.cz, CZU Prague).

BBA1 was treated as a separate subset of the data and allow users to pull it into their analyses separated from the most recent data.

11.3.1 Licensing and access

- Restricted

11.3.2 Sampling methodology

Fieldwork and Data Collection described for the 2014-17 atlas.

Each basic quadrant (12×11.1 km) was subdivided into 16 smaller squares (approx. 3×2.8 km) to increase spatial resolution.

Observers aimed to confirm breeding of as many species as possible, categorizing evidence as:

- A: Possible breeding (e.g., species seen in breeding habitat during the breeding season)
- B: Probable breeding (e.g., pairs observed, territorial behavior, nest building)
- C: Confirmed breeding (e.g., occupied nests, eggs, young, adults feeding young)

Fieldwork spanned all habitats in each quadrant, with attention to rare or nocturnal species.

11.3.3 Sampling effort

- *XX important notes from emails (only statements), links to related websites*
- a

11.4 Input Data

11.4.1 Compressed original data folder

The data were pre-processed by Francois Leroy and used in his Ph.D thesis. We received the original files from him after an agreement with the data owners.

- M4_accdb.RData contains datasets of the 4th atlas period.
- SPECIES_allM.RData contains datasets from the atlas periods 1 to 4. We used this file as the basis for BEAST.
- Other_shapefiles includes grids at different resolutions.
- Volunteers_Atlas_M4.xlsx includes grid cells identifications and data of the atlasers names and dates of visits.
- grids contains original grid shapefiles for the first atlas (M1) and subsequent ones (M2_3_4).
- shp_for_predictions (Francois).zip holds grids at 50 and 10000 squared kilometers.

11.4.2 Docs folder

- *Methods_Czech_Atlas_birds.pdf: original methods section (in Czech).*
- 4_Metodika_atlas.pdf: methods publication in Aythia 5 (2014), before the 2014 - 2017 atlas period (in Czech).
- ENG_Metodika_atlas_2014-2017: translation of methods published in Aythia 5 (2014).
- Czech_Methods_1-4 en-US: translation of methods section of the atlas book (2014 - 2017)

11.4.3 Checked data folder

- *There is no Checked data folder. The data were loaded directly into R. The data preparation steps can be seen below.*

11.5 Data standardization and processing comments

The information we require was spread in several files and dataframes. We checked the match between species occurrences and grid cells and removed records that didn't have a corresponding grid cell id inside the territory of the Czech Republic.

Grid files for the first and current atlases were coarsened to different resolutions. Since the cells in both shapefiles have different sizes, we attempted to standardize as much as possible across different spatial scales, but still there is a inevitable mismatch remaining.

11.6 Libraries

```
pacman::p_load(
    sf, terra, tidyverse, tidyterra, knitr,
    tictoc, RPostgres, DBI, dbplyr, parallel,
    geodata, readxl
)
```

11.6.1 Preparation

11.6.1.1 Database connection

```
source("scripts/dbcon.R")
```

11.6.1.2 Extract original data

```
system("unzip data/Birds_Czechia/Original_data.zip -d /tmp")
system("ls -l -h /tmp/Original_data/")
```

11.7 Load datasets

```
load("/tmp/Original_data/SPECIES_allM.RData")
```

```
## Load observer list

Volunteers_Atlas_M4 <-
    read_excel("/tmp/Original_data/Volunteers_Atlas_M4.xlsx") %>%
    split(., ifelse(.$ActiveFrom == "all years", "T", "F"))

Volunteers_Atlas_M4[["F"]]$ActiveFrom <- as.numeric(Volunteers_Atlas_M4[["F"]]$ActiveFr
```

```
Volunteers_Atlas_M4[["F"]]$ActiveTo <- as.numeric(Volunteers_Atlas_M4[["F"]]$ActiveTo)

Volunteers_Atlas_M4[["T"]]$ActiveFrom <- min(Volunteers_Atlas_M4[["F"]]$ActiveFrom)

Volunteers_Atlas_M4[["T"]]$ActiveTo <- max(Volunteers_Atlas_M4[["F"]]$ActiveTo)

Volunteers_Atlas_M4 <- do.call(rbind, Volunteers_Atlas_M4)

#### Load the databases

to_load <- list.files("/tmp/Original_data/M4", full.names = T)

for (i in 1:length(to_load)) {
    suppressWarnings(
        assign(
            stringr::str_extract(to_load[i], "(?<=M4/).+?(?=.xlsx)"),
            readxl::read_xlsx(to_load[i])
        )
    )
}

## I need to erase the double spacings in the observers name of AVIF_all
AVIF_all$OBSERVER <- stringr::str_squish(AVIF_all$OBSERVER)
```

```
## Create the list of observers

observers <- Volunteers_Atlas_M4 %>%
    unite("OBSERVER", c(Name, Surname), sep = " ") %>%
    pull(OBSERVER) %>%
    unique()

# Now I need to filter the data by the observers
M4_speciesNumber <-
    AVIF_all %>%
    filter(OBSERVER %in% observers) %>%
    select(POLE_original, ATLAS_name) %>%
    rbind(NDOP_all %>% filter(if_any(starts_with("OBSERVER"), ~ . %in% observers)) %>% select(POL
    rbind(ADDITIONAL_data %>% select(POLE_original, ATLAS_name)) %>%
    rbind(SPECIES_cards %>% select(POLE_original, ATLAS_name) %>%
        mutate(POLE_original = gsub("[^0-9.-]", "", SPECIES_cards$POLE_original))) %>%
    ## And compute the species richness
    group_by(POLE_original) %>%
    distinct(ATLAS_name) %>%
    count(POLE_original, name = "species_number_M4") %>%
    rename(KVADRAT = POLE_original) %>%
```

```
    mutate_at("KVADRAT", as.character)
```

```
M4 <-
    AVIF_all %>%
    filter(OBSERVER %in% observers) %>%
    select(POLE_original, ATLAS_name, ATLAS_code) %>%
    rbind(NDOP_all %>% filter(if_any(starts_with("OBSERVER"), ~ . %in% observers)) %>%
    rbind(ADDITIONAL_data %>% select(POLE_original, ATLAS_name, ATLAS_code)) %>%
    rbind(SPECIES_cards %>% select(POLE_original, ATLAS_name, ATLAS_code) %>%
        mutate(POLE_original = gsub("[^0-9.-]", "", SPECIES_cards$POLE_original))) %>%
    ## And compute the species richness
    group_by(POLE_original) %>%
    distinct(ATLAS_name, ATLAS_code) %>%
    rename(KVADRAT = POLE_original, SPECIES = ATLAS_name)
```

11.7.1 Process and export

```
# Add the start and end years to each dataset
M1$start_year <- 1973
M1$end_year <- 1977
M2$start_year <- 1985
M2$end_year <- 1989
M3$start_year <- 2001
M3$end_year <- 2003
M4$start_year <- 2014
M4$end_year <- 2017
```

```
## Load the sampling effort data
sampling_effort_M2 <- read_xlsx("/tmp/Original_data/Atlas_samplingeffort.xlsx", sheet =

colnames(sampling_effort_M2) <- gsub(".0$", "", colnames(sampling_effort_M2))

## Sum the number of cards
M2 <-
    sampling_effort_M2 %>%
    select(KVADRAT, "1985", "1986", "1987", "1988", "1989") %>%
    mutate(effort = rowSums(across("1985":"1989"), na.rm = T)) %>%
    filter(!is.na(KVADRAT)) %>%
    left_join(M2, ., by = "KVADRAT")

sampling_effort_M3 <- read_xlsx("/tmp/Original_data/Atlas_samplingeffort.xlsx", sheet =

colnames(sampling_effort_M3) <- gsub(".0$", "", colnames(sampling_effort_M3))

## Sum the number of cards
```

```
M3 <-
    sampling_effort_M3 %>%
    select(KVADRAT, "2001", "2002", "2003") %>%
    mutate(effort = rowSums(across("2001":"2003"), na.rm = T)) %>%
    filter(!is.na(KVADRAT)) %>%
    left_join(M3, ., by = "KVADRAT")

# Volunteers count M4
sampling_effort_M4 <-
    AVIF_all %>%
    filter(OBSERVER %in% observers) %>%
    select(POLE_original, OBSERVER) %>%
    rbind(
        NDOP_all %>% select_at(vars(contains(c("original", "OBSE")))) %>%
            pivot_longer(cols = contains("OBSER"), values_to = "OBSERVER", values_drop_na = T) %>
            select(-name)
    ) %>%
    rbind(
        ADDITIONAL_data %>% select_at(vars(contains(c("original", "OBSE")))) %>%
            pivot_longer(cols = contains("OBSER"), values_to = "OBSERVER", values_drop_na = T) %>
            select(-name)
    ) %>%
    rbind(
        SPECIES_cards %>% select(POLE_original, OBSERVER) %>%
            mutate(POLE_original = gsub("[^0-9.-]", "", SPECIES_cards$POLE_original))
    ) %>%
    group_by(POLE_original) %>%
    distinct(OBSERVER) %>%
    count(POLE_original, name = "effort") %>%
    rename(KVADRAT = POLE_original) %>%
    left_join(M4_speciesNumber, by = "KVADRAT")

M4 <- left_join(M4, select(sampling_effort_M4, KVADRAT, effort), by = "KVADRAT")
```

```
M1$effort <- NA
M1$samp_effort_type <- NA
M2$samp_effort_type <- "n_cards"
M3$samp_effort_type <- "n_cards"
M4$samp_effort_type <- "n_observer"
```

```
M1 <- mutate(M1, EBBA_code = paste0(ATLAS_code, ACTIVITY_code)) %>%
    select(all_of(c("KVADRAT", "SPECIES", "EBBA_code", "start_year", "end_year", "effort", "samp_
    unique()

M2 <- mutate(M2, EBBA_code = paste0(ATLAS_code, ACTIVITY_code)) %>%
```

```
    select(all_of(c("KVADRAT", "SPECIES", "EBBA_code", "start_year", "end_year", "effo
    unique()

M3 <- mutate(M3, EBBA_code = paste0(ATLAS_code, ACTIVITY_code)) %>%
    select(all_of(c("KVADRAT", "SPECIES", "EBBA_code", "start_year", "end_year", "effo
    unique()

M4 <- mutate(M4, EBBA_code = ATLAS_code) %>%
    select(all_of(c("KVADRAT", "SPECIES", "EBBA_code", "start_year", "end_year", "effo
    unique()
```

```
data <- lapply(list(M1, M2, M3, M4), function(x) {
    res <- rename(x,
        cellID = matches("KVADRAT|POLE"),
        verbatim_name = matches("SPECIES|ATLAS_name")
    )

    res <- select(res, cellID, verbatim_name, EBBA_code, start_year, end_year, effort,
    return(res)
}) %>%
    do.call(rbind, .) %>%
    filter(!is.na(cellID)) %>%
    mutate(verbatim_name = str_squish(verbatim_name))
```

11.8 Taxonomy

```
taxonomy <- dbGetQuery(con, 'SELECT * FROM public."CB_verbatim_name_equivalence" WHERE

ids <- data %>%
    rownames_to_column("row") %>%
    left_join(., taxonomy, by = c("verbatim_name" = "verbatimIdentification")) %>%
    group_by(row) %>%
    slice_min(datasetID, with_ties = FALSE) %>%
    ungroup()

table(is.na(ids$scientificNameID))

write.csv(ids, "/tmp/ids.csv")
```

The data were manually checked in Libreoffice and then re-imported into R.

```
ids <- read.csv("/tmp/ids.csv") %>%
    select(-X, -row) %>%
    mutate(gridID = ifelse(start_year == 1973, 1, 2))
```

11.9 Grid preparation

```
grid1 <- vect("/tmp/Original_data/grids/M1") %>%
    select(POLE) %>%
    rename(cellID = POLE)
grid2 <- vect("/tmp/Original_data/grids/M2_3_4") %>%
    select(KVADRAT) %>%
    rename(cellID = KVADRAT)
grid1 <- project(grid1, crs(grid2))
```

Check match between grids and data

```
setdiff(filter(ids, gridID == 1) %>% pull(cellID) %>% unique(), unique(grid1$cellID))
setdiff(filter(ids, gridID == 2) %>% pull(cellID) %>% unique(), unique(grid2$cellID))
```

```
grid_scaling <- function(grid, dist) {
    if (inherits(grid, "sf")) {
        grid <- vect(grid)
    }
    label <- project(grid, "epsg:3857") %>% # Project to Pseudo-Mercator (cilindrical with meter
        centroids() %>%
        geom() %>%
        as.data.frame() %>%
        cbind(centroids(grid), .) %>%
        mutate(
            bandx = str_extract(x, "^\\d{3}") %>% # Extract part of the latitude coordinates to h
                rank(., ties.method = "min") %>%
                as.factor() %>% as.numeric(),
            bandy = str_extract(y, "^\\d{3}") %>% # Extract part of the longitude coordinates to
                rank(., ties.method = "min") %>%
                as.factor() %>% as.numeric()
        ) %>%
        mutate(
            seqx = cut_interval(bandx, length = dist, labels = F),
            seqy = cut_interval(bandy, length = dist, labels = F)
        ) %>%
        mutate("{dist}" := paste(seqx, seqy) %>%
            factor() %>%
            as.numeric()) %>%
        project(., crs(grid)) %>%
        select(last_col())

    grid <- cbind(grid, label)
}
```

Create labels for rescaling

```r
for (dist in c(2, 4, 8, 64)) {
    grid1 <- grid_scaling(grid1, dist)
}

grid1$`1` <- grid1$cellID %>% rank()

for (dist in c(2, 4, 8, 16, 32, 64)) {
    grid2 <- grid_scaling(grid2, dist)
}

grid2 <- st_as_sf(grid2)
grid1 <- st_as_sf(grid1)
grid1 <- st_join(grid1, select(grid2, `16`, `32`), largest = T)

grid2$`1` <- grid2$cellID %>% rank()
```

```r
library(ggplot2)
library(sf)
library(cowplot)

fcol <- "2"

library(viridis)

# Shuffle fill colors by randomly assigning factor levels
set.seed(42)
shuffle_levels <- function(x) {
    lvls <- unique(as.factor(x))
    shuffled <- sample(lvls, length(lvls))
    factor(x, levels = shuffled)
}

p1 <- ggplot(grid1) +
    geom_sf(aes(fill = shuffle_levels(.data[[fcol]])), color = "black", lwd = 0.1) +
    scale_fill_viridis_d(option = "turbo", direction = 1, begin = 0, end = 1) +
    theme_minimal() +
    ggtitle("grid1") +
    guides(fill = "none")

p2 <- ggplot(grid2) +
    geom_sf(aes(fill = shuffle_levels(.data[[fcol]])), color = "black", lwd = 0.1) +
    scale_fill_viridis_d(option = "turbo", direction = 1, begin = 0, end = 1) +
    theme_minimal() +
    ggtitle("grid2") +
    guides(fill = "none")
```

```
plot_grid(p1, p2, labels = c("A", "B"), ncol = 2)
```

```
grid1$gridID <- 1
grid2$gridID <- 2

scale1 <- st_drop_geometry(grid1) %>%
    pivot_longer(cols = -c(cellID, gridID), names_to = "scalingID", values_to = "siteID")

scale2 <- st_drop_geometry(grid2) %>%
    pivot_longer(cols = -c(cellID, gridID), names_to = "scalingID", values_to = "siteID")

scaling_table <- rbind(scale1, scale2)
```

11.10 Import into database

11.10.1 Database connection

```
source("scripts/dbcon.R")
```

11.10.2 List tables in database

```
dbListTables(con) %>%
    purrr::keep(., ~ grepl("^CB|^MOBI", .)) %>%
    purrr::keep(., ~ !grepl("\\d+$|default$", .)) %>%
    kable(col.names = "Tables")
```

11.10.3 Write code books

11.10.3.1 CB_license

```
tbl(con, "CB_license") %>%
    # head(5) %>%
    collect() %>%
    kable()
```

11.10.3.2 CB_sampling_effort

```
tbl(con, "CB_sampling_effort") %>%
    # head(5) %>%
    collect() %>%
    knitr::kable()
```

11.10.3.3 CB_model

```
tbl(con, "CB_model") %>%
    # head(5) %>%
    collect() %>%
    knitr::kable()
```

11.10.3.4 CB_taxonomy

```
tbl(con, "CB_taxonomy") %>%
    head(5) %>%
    collect() %>%
    knitr::kable()
```

11.10.3.5 CB_verbatim_name_equivalence

```
tbl(con, "CB_verbatim_name_equivalence") %>%
    head(5) %>%
    collect() %>%
    knitr::kable()
```

11.10.4 Write dataset tables

11.10.4.1 Dataset

```
table <- "MOBI_dataset" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
    as.data.frame() %>%
    colnames() %>%
    paste(., ' = ""', collapse = ",\n") %>%
    cat()
```

Create data table

```
datasetID <- 5
data_table <- data.frame(
    datasetID = datasetID,
    datasetName = "Czech Breeding Bird Atlas",
    datasetPublisher = "Czech Society for Ornithology",
    datasetPublisherContact = "bejcek@fzp.czu.cz | stastny@fzp.czu.cz",
    licenseID = 0,
    rightsHolder = "Vladimír Bejček & Karel Šťastný",
    bibliographicCitation = "BBA1: Šťastný, Karel; Randík, Aladár; Hudec, Karel. 1987.
```

```
    citationIdentifier = "ISBN 21-003-87 | ISBN 80-86022-18-8 | ISBN 80-86858-19-7 | ISBN 978-80-
    provider = "Vladimír Bejček | Karel Šťastný",
    shareable = "NO",
    coauthorshipRequired = "YES",
    coauthors = "Vladimír Bejček - bejcek@fzp.czu.cz | Karel Šťastný - stastny@fzp.czu.cz | BBA4:
    coauthorshipSuggested = "Carmen Soria - carmendianasoria@gmail.com | Kateřina Tschernosterová
    isSamplingEffortReported = "YES",
    isOccurrenceProbabilityAvailable = "NO",
    occurrenceModelID = NA,
    trustFilterMeaning = "NULL = No information",
    taxa = "Aves",
    cellSizeCategory = 10,
    samplingYears = "1973-1977, 1985-1989, 2001-2003, 2014-2017",
    dataSubsetMeaning = "1 = first Czech atlas. Grid different from subsequent atlases. 2 = atlas
)
```

Write into the database

```
try({
    dbExecute(con, str_glue('DELETE FROM "{table}" WHERE "datasetID" = {datasetID}'))
    copy_to(con, data_table, table, append = T)
})
rm(data_table)
```

Check table.

```
tbl(con, sql(str_glue('SELECT * FROM "{table}" WHERE "datasetID" = {datasetID}'))) %>% kable(alig
```

11.10.4.2 Site

```
table <- "MOBI_site" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
    as.data.frame() %>%
    colnames() %>%
    paste(., ' = ""', collapse = ",\n") %>%
    cat()
```

Create data table

```
data_table <- rbind(grid1, grid2) %>%
    st_transform(., 4326) %>%
    select(cellID, gridID) %>%
    rename(verbatimSiteID = cellID) %>%
    mutate(
        datasetID = datasetID,
```

```
        footprintSRS = "EPSG:4326",
        verbatimFootprintSRS = "EPSG:5514",
        croppedGeometry = st_sfc(
            lapply(seq_len(nrow(.)), function(i) st_multipolygon()),
            crs = st_crs(4326)
        )
    )
```

Write into the database

```
try({
    # Remove records where datasetID is the current dataset
    dbExecute(con, str_glue('DELETE FROM "{table}" WHERE "datasetID" = {datasetID}'))

    # Copy data to the database
    st_write(obj = data_table, dsn = con, layer = table, append = T)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(
    con,
    sql(str_glue('SELECT * FROM "{table}" WHERE "datasetID" = {datasetID}'))
) %>%
    head(n = 20) %>%
    kable(align = "c")
```

11.10.5 Write helper tables

11.10.5.1 Scaling table

This intermediate table enables the joins of records data to the different resolutions of sites.

```
table <- "MOBI_scaling_table" ## Table of interest
```

Check table colnames.

```
tbl(con, table) %>%
    as.data.frame() %>%
    colnames() %>%
    paste(., ' = ""', collapse = ",\n") %>%
    cat()
```

```
data_table <- data.frame(
    verbatimSiteID = scaling_table$cellID,
```

```
    datasetID = datasetID,
    scalingID = scaling_table$scalingID,
    siteID = scaling_table$siteID,
    gridID = scaling_table$gridID
) %>% unique()
```

Write into database.

```
try({
    # Remove records where datasetID is the current dataset
    dbExecute(con, str_glue('DELETE FROM "{table}" WHERE "datasetID" = {datasetID}'))

    # Copy data to the database
    copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(str_glue('SELECT * FROM "{table}" WHERE "datasetID" = {datasetID}'))) %>%
    head() %>%
    kable(align = "c")
```

11.10.6 Write records tables

11.10.6.1 Event

```
table <- "MOBI_event" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
    as.data.frame() %>%
    colnames() %>%
    paste(., ' = ""', collapse = ",\n") %>%
    cat()
```

Create data table

```
data_table <- data.frame(
    verbatimSiteID = ids$cellID,
    datasetID = datasetID,
    samplingPeriodID = dense_rank(ids$start_year),
    startYear = ids$start_year,
    endYear = ids$end_year,
    samplingEffortID = ifelse(ids$samp_effort_type == "n_cards", 5, ifelse(ids$samp_effort_type =
```

```
    samplingEffortValue = ids$effort,
    remarks = NA,
    samplingEffort2ID = NA,
    samplingEffort2Value = NA,
    samplingEffort3ID = NA,
    samplingEffort3Value = NA
) %>%
    filter(!verbatimSiteID %in% c(5414, 6135)) %>%
    unique()

# if (nrow(unique(data_table)) != nrow(eff)) {
#   stop("Stopping execution due to row number mismatch")
# } else {
#   print("OK")
# }

data_table <- filter(data_table, !is.na(verbatimSiteID))
```

Write into the database

```
try({
    # Remove records where datasetID is the current dataset
    dbExecute(con, str_glue('DELETE FROM "{table}" WHERE "datasetID" = {datasetID}'))

    # Copy data to the database
    copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(str_glue('SELECT * FROM "{table}" WHERE "datasetID" = {datasetID}'))) %>%
    head() %>%
    kable(align = "c")
```

11.10.6.2 Presence

```
table <- "MOBI_presence" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
    as.data.frame() %>%
    colnames() %>%
    paste(., ' = ""', collapse = ",\n") %>%
```

```
    cat()
```

Create data table

```
data_table <- data.frame(
    verbatimIdentificationID = ids$verbatimIdentificationID,
    verbatimSiteID = ids$cellID,
    datasetID = datasetID,
    samplingPeriodID = dense_rank(ids$start_year),
    trustFilter = NA,
    breedingEvidence = ids$EBBA_code,
    season = 1,
    datasetSubset = ifelse(ids$start_year == 1973, 1, 2),
    gridID = ids$gridID
) %>%
    filter(!verbatimSiteID %in% c(5414, 6135)) %>%
    unique()
```

Write into the database

```
try({
    # Remove records where datasetID is the current dataset
    dbExecute(con, str_glue('DELETE FROM "{table}" WHERE "datasetID" = {datasetID}'))

    # Copy data to the database
    copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(str_glue('SELECT * FROM "{table}" WHERE "datasetID" = {datasetID}'))) %>%
    head() %>%
    kable(align = "c")
```

11.10.6.3 Probability

```
table <- "MOBI_probability" ## Table of interest
```

Check table colnames

```
tbl(con, table) %>%
    as.data.frame() %>%
    colnames() %>%
    paste(., ' = ""', collapse = ",\n") %>%
    cat()
```

Create data table

```
data_table <- data.frame(
    verbatimIdentificationID = "",
    verbatimSiteID = "",
    datasetID = "",
    samplingPeriodID = "",
    trustFilter = "",
    breedingEvidence = "",
    season = "",
    datasetSubset = "",
    gridID = ""
)
```

Write into the database

```
try({
    # Remove records where datasetID is the current dataset
    dbExecute(con, paste0('DELETE FROM "', table, '" WHERE "datasetID" = ', datasetID))

    # Copy data to the database
    copy_to(con, data_table, table, append = TRUE)
})

# Remove the data_table from the environment
rm(data_table)
```

Check table.

```
tbl(con, sql(paste0('SELECT * FROM "', table, '" WHERE "datasetID" = ', datasetID))) %>%
    head() %>%
    kable(align = "c")
```