Retrieve standard data from Seatrack

Jens Åström

2018-08-09

Contents

Connecting to the database

Users at NINA and the Polarinstitute (using a computer that is within these networks' IP-addresses) can connect to the Seatrack database. It is a PostgreSQL (9.6) database answering to the address `seatrack.nina.no`, on the standard port 5432. Users should use their individual login user names and passwords. Contact Jens Åström (jens.astrom@nina.no) for details about usernames and passwords.

This instruction deals with the preferred way of connecting to the database, using R and the `seatrackR` package. Another option is to connect through a dedicated database management software, such as Pgadmin3 (or 4), HeidiSQL, or similar. Some users may prefer to use the MS Access interface.

To simplify the connection, use the convenience function `connectSeatrack`. This creates a connection named `con` by using the packages `DBI` and `RPostgres`.

```
require(seatrackR)
```

```
## Loading required package: seatrackR
```

```
connectSeatrack(Username = "testreader", Password = "testreader")
```

Custom queries

As of now, 4 functions exist to retreive data from the database through prebuilt queries. Apart from that, users are free to use their own queries through the functions in `DBI` and `dplyr`, using the connection named `con` made by the `connectSeatrack()`-function.

It is perfecty fine to download data through your own custom queries. Creating interesting queries requires some knowledge about the structure of the database however. Pgadmin3(4) would be a useful tool to get further info on that. For now, we show a simple query involving just one table. Here we get the different locations currently recorded from the Faroe Islands (Coordinates not updated). Note that you have to load the `DBI` package and use its query functions.

```
require(DBI)
```

```
## Loading required package: DBI
```

```
myQuery <- "SELECT * from metadata.location
             WHERE colony_int_name = 'Faroe Islands'"

faroeLocations <- dbGetQuery(con, myQuery)
head(faroeLocations)
```

```
##                                     id  location_name colony_int_name
## 1 b7e16a70-0bf0-11e8-82b0-005056b165f3     Gassadalur   Faroe Islands
## 2 b7e1f8fa-0bf0-11e8-82b0-005056b165f3     Glyvursnes   Faroe Islands
## 3 b7e67e98-0bf0-11e8-82b0-005056b165f3    Havnardalur   Faroe Islands
## 4 b7ee5884-0bf0-11e8-82b0-005056b165f3 Kirkjubøholmur   Faroe Islands
## 5 b7f25f38-0bf0-11e8-82b0-005056b165f3 Lamba grotbrot   Faroe Islands
## 6 b7f52fce-0bf0-11e8-82b0-005056b165f3     Leynavatni   Faroe Islands
##   colony_nat_name    lat    lon geom
## 1         Føroyar 66.585 12.229 <NA>
## 2         Føroyar 66.585 12.229 <NA>
## 3         Føroyar 66.585 12.229 <NA>
## 4         Føroyar 66.585 12.229 <NA>
## 5         Føroyar 66.585 12.229 <NA>
## 6         Føroyar 66.585 12.229 <NA>
```

Position data

The primary data of the positions of the birds is stored in the table `positions.postable`. This includes all entered positions in the database.

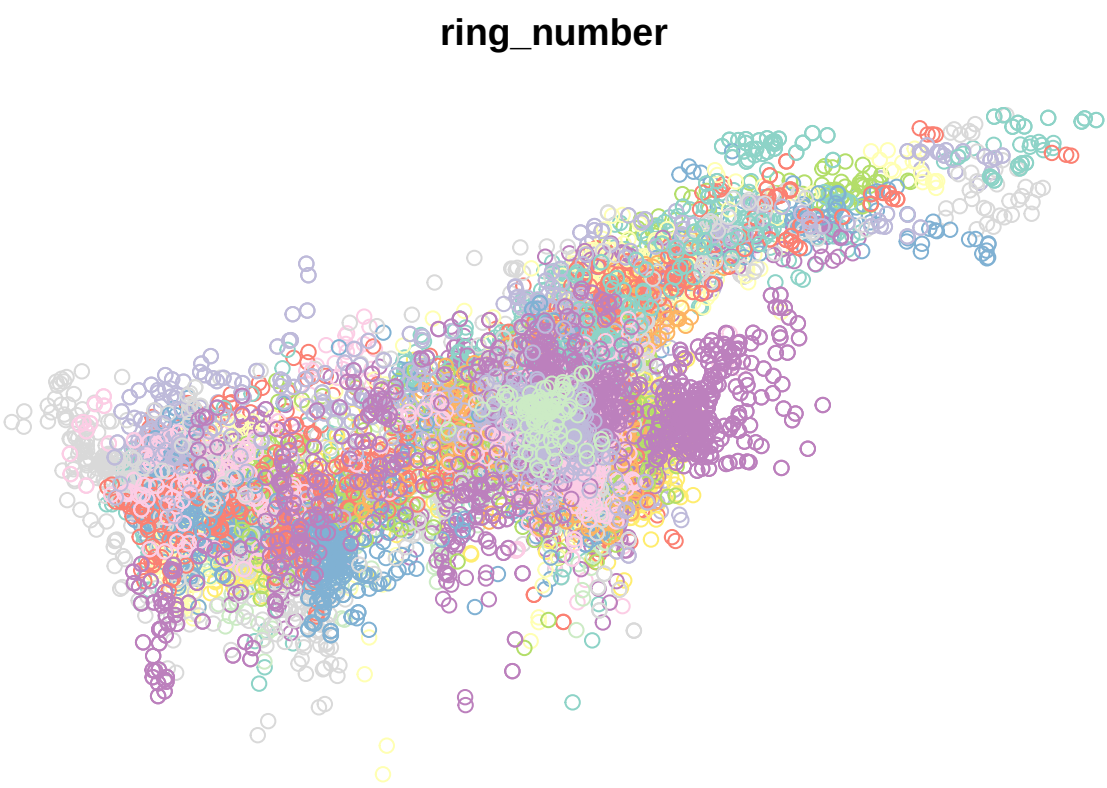
The `getPosdata` function retrieves this table, with options to subselect only specific species, colonies, responsible contact person, specific ring numbers, and years. There is also an option to limit the records to a set number of rows, and to load the position coordinates as a spatial object.

```
eynhallowPositions <- getPosdata(selectColony = "Eynhallow", loadGeometries = T)
eynhallowPositions
```

```
## Simple feature collection with 33713 features and 44 fields
## geometry type:  POINT
## dimension:      XY
## bbox:           xmin: -66.59141 ymin: 36.96776 xmax: 62.64313 ymax: 78.75672
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
## # A tibble: 33,713 x 45
##    id       date_time           logger logger_id logger_model year_tracked
##    <chr>    <dttm>              <chr>  <chr>     <chr>        <chr>
##  1 72139fc~ 2015-10-25 00:07:55 M722_~ M722      c250         2015_16
##  2 2a98e3c~ 2015-10-25 12:15:33 M722_~ M722      c250         2015_16
##  3 7c0427f~ 2015-10-26 00:11:33 M722_~ M722      c250         2015_16
##  4 bbd9e6f~ 2015-10-26 12:21:33 M722_~ M722      c250         2015_16
##  5 e73e24a~ 2015-10-27 00:22:33 M722_~ M722      c250         2015_16
##  6 695a169~ 2015-10-27 12:26:33 M722_~ M722      c250         2015_16
##  7 69e33c6~ 2015-10-28 00:21:33 M722_~ M722      c250         2015_16
##  8 a027776~ 2015-10-28 12:18:17 M722_~ M722      c250         2015_16
```

```
##  9 0e31475~ 2015-10-29 00:29:47 M722_~ M722      c250         2015_16
## 10 ad4c10d~ 2015-10-29 12:36:34 M722_~ M722      c250         2015_16
## # ... with 33,703 more rows, and 39 more variables: session_id <int>,
## #   year_deployed <int>, year_retrieved <int>, ring_number <chr>,
## #   euring_code <chr>, species <chr>, colony <chr>, lon_raw <dbl>,
## #   lat_raw <dbl>, lon_smooth1 <dbl>, lat_smooth1 <dbl>,
## #   lon_smooth2 <dbl>, lat_smooth2 <dbl>, disttocol_s2 <dbl>,
## #   eqfilter1 <int>, eqfilter2 <int>, eqfilter3 <int>,
## #   lat_smooth2_eqfilt3 <dbl>, sex <chr>, morph <chr>, subspecies <chr>,
## #   age <chr>, col_lon <dbl>, col_lat <dbl>, tfirst <dttm>,
## #   tsecond <dttm>, twl_type <int>, conf <int>, sun <dbl>, software <chr>,
## #   light_threshold <int>, analyzer <chr>, data_responsible <chr>,
## #   logger_yeartracked <chr>, posdata_file <chr>, import_date <date>,
## #   data_version <int>, database_version <int>, geometry <POINT [°]>
```

```
plot(eynhallowPositions["ring_number"])
```



Position data for export

The data sent to the Polar institute also have the subspecies names added to the records. The export ready positions data can be retrieved most easily through a specific export view. Note that this will download all records, and will take some time. Note that this export does not contain information on the used and deleted uuids.

```
newExport <- dbReadTable(con, Id(schema = "views", table = "export"))
nrow(newExport)
```

```
## [1] 1759448
```

```
write.csv(newExport, file = "seatrack_export_2018-08-09.csv")
```

If you are interested in knowing separate old, deleted rows, these are found in the table `positions.deleted_uuid`.

```
deletedUuids <- dbReadTable(con, Id(schema = "positions", table = "deleted_uuid"))
nrow(deletedUuids)
```

```
## [1] 1047730
```

```
write.csv(deletedUuids, file = "deletedUuids_2018-08-09.csv")
```

Other functions for download

There are some more convenience functions for retrieving information from the database as well. Here follows a quich demo.

The `getFileArchiveSummary` function

This function pulls together data from several tables with focus on the file archive. It should contain enough information to know what the individual raw files contain.

```
eynhallowFiles <- getFileArchiveSummary(selectColony = "Eynhallow")
eynhallowFiles
```

```
## # A tibble: 0 x 9
## # ... with 9 variables: file_id <int>, session_id <int>, colony <chr>,
## #   ring_number <chr>, euring_code <chr>, year_tracked <chr>,
## #   logger_serial_no <chr>, logger_model <chr>, filename <chr>
```

The `getIndividInfo` function

This function summarizes all observation data for the individual birds. We can subselect the colony and year interval the bird where tracked.

```
eynhallowIndivids <- getIndividInfo(selectColony = "Eynhallow", selectYear = "2014_15")
eynhallowIndivids
```

```
## # A tibble: 0 x 31
## # ... with 31 variables: session_id <int>, colony <chr>,
## #   year_tracked <chr>, ring_number <chr>, euring_code <chr>,
## #   color_ring <chr>, species <chr>, subspecies <chr>, morph <chr>,
## #   status_age <chr>, status_sex <chr>, status_sexing_method <chr>,
## #   status_date <date>, weight <dbl>, scull <dbl>, tarsus <dbl>,
## #   wing <dbl>, breeding_stage <chr>, eggs <int>, chicks <int>,
## #   hatching_success <lgl>, breeding_success <lgl>,
## #   breeding_success_criterion <chr>, data_responsible <chr>,
## #   back_on_nest <lgl>, comment <chr>, latest_sex <chr>,
## #   latest_sexing_method <chr>, latest_age <chr>, latest_info_date <date>,
## #   eventType <chr>
```

!Note the weird duplicate records here! **TO BE FIXED**

```
eynhallowIndivids %>% print(width = Inf)
```

```
## # A tibble: 0 x 31
## # ... with 31 variables: session_id <int>, colony <chr>,
## #   year_tracked <chr>, ring_number <chr>, euring_code <chr>,
## #   color_ring <chr>, species <chr>, subspecies <chr>, morph <chr>,
## #   status_age <chr>, status_sex <chr>, status_sexing_method <chr>,
## #   status_date <date>, weight <dbl>, scull <dbl>, tarsus <dbl>,
## #   wing <dbl>, breeding_stage <chr>, eggs <int>, chicks <int>,
## #   hatching_success <lgl>, breeding_success <lgl>,
## #   breeding_success_criterion <chr>, data_responsible <chr>,
## #   back_on_nest <lgl>, comment <chr>, latest_sex <chr>,
## #   latest_sexing_method <chr>, latest_age <chr>, latest_info_date <date>,
## #   eventType <chr>
```