

Retrieve standard data from Seatrack

Jens Åström

2018-08-21

Contents

Connecting to the database	1
Custom queries	1
Position data	2
Position data for export	3
Other functions for download	4
The <code>getFileArchiveSummary</code> function	4
The <code>getIndividInfo</code> function	4
Commonly used info	6

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
## Warning: package 'seatrackR' was built under R version 3.4.4
connectSeatrack(Username = "testreader", Password = "testreader")
```

Custom queries

As of now, 4 functions exist to retrieve 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 perfectly 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 grottbrot 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)
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

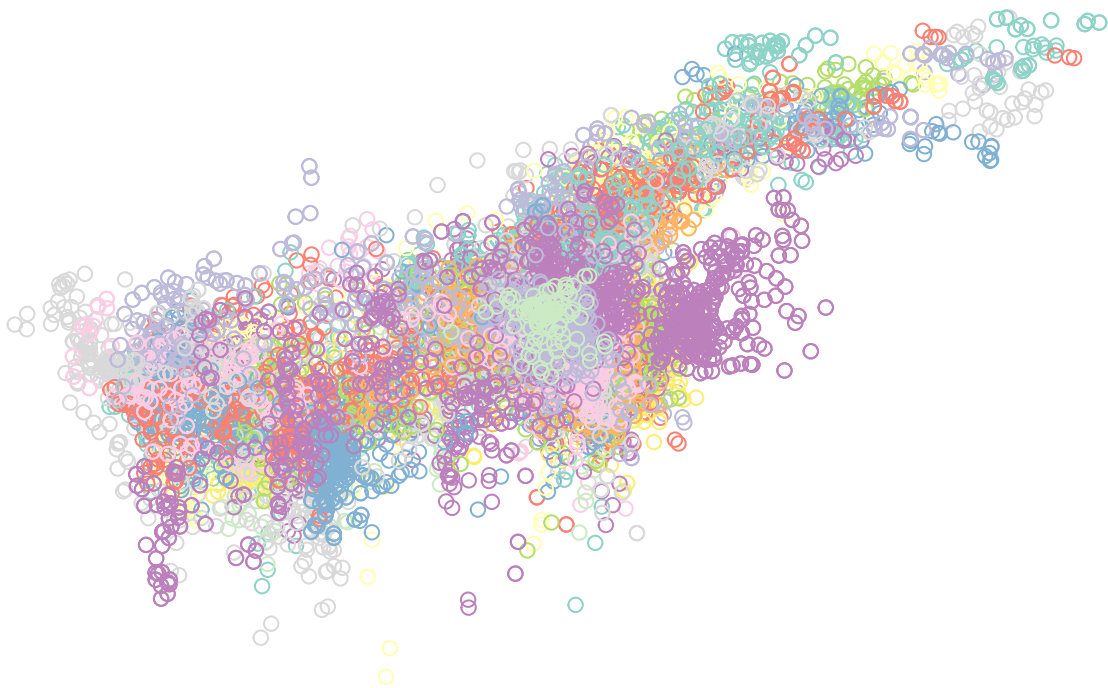
```
eynhallowPositions

## 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> <dtm> <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 <dtm>,
## #   tsecond <dtm>, 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"])
```

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)
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)
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 quick 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.

```
hornoyaIndivids <- getIndividInfo(selectColony = "Hornoya", selectYear = "2014_15")
hornoyaIndivids
```

```
## # A tibble: 205 x 31
##   session_id colony year_tracked ring_number euring_code color_ring
##   <int> <chr> <chr> <chr> <chr> <chr>
## 1 2903 Hornoya 2014_15 5185355 NOS W(NL1)
## 2 2903 Hornoya 2014_15 5185355 NOS W(NL1)
## 3 2924 Hornoya 2014_15 5185357 NOS W(NP1)
## 4 2924 Hornoya 2014_15 5185357 NOS W(NP1)
## 5 2890 Hornoya 2014_15 5185359 NOS W(NC1)
## 6 2890 Hornoya 2014_15 5185359 NOS W(NC1)
## 7 2890 Hornoya 2014_15 5185359 NOS W(NC1)
## 8 2922 Hornoya 2014_15 5185360 NOS W(ND1)
## 9 2922 Hornoya 2014_15 5185360 NOS W(ND1)
## 10 2900 Hornoya 2014_15 5185361 NOS W(NV1)
## # ... with 195 more rows, and 25 more variables: 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**

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

```
## # A tibble: 205 x 31
##   session_id colony year_tracked ring_number euring_code color_ring
##   <int> <chr> <chr> <chr> <chr> <chr>
## 1 2903 Hornoya 2014_15 5185355 NOS W(NL1)
## 2 2903 Hornoya 2014_15 5185355 NOS W(NL1)
## 3 2924 Hornoya 2014_15 5185357 NOS W(NP1)
## 4 2924 Hornoya 2014_15 5185357 NOS W(NP1)
## 5 2890 Hornoya 2014_15 5185359 NOS W(NC1)
## 6 2890 Hornoya 2014_15 5185359 NOS W(NC1)
## 7 2890 Hornoya 2014_15 5185359 NOS W(NC1)
## 8 2922 Hornoya 2014_15 5185360 NOS W(ND1)
## 9 2922 Hornoya 2014_15 5185360 NOS W(ND1)
## 10 2900 Hornoya 2014_15 5185361 NOS W(NV1)
##   species subspecies morph status_age status_sex
##   <chr> <chr> <chr> <chr> <chr>
## 1 Atlantic puffin <NA> <NA> adult_unknown <NA>
## 2 Atlantic puffin <NA> <NA> adult_unknown <NA>
## 3 Atlantic puffin <NA> <NA> adult_unknown <NA>
## 4 Atlantic puffin <NA> <NA> adult_unknown <NA>
## 5 Atlantic puffin <NA> <NA> adult_unknown <NA>
## 6 Atlantic puffin <NA> <NA> adult_unknown <NA>
## 7 Atlantic puffin <NA> <NA> adult_unknown <NA>
## 8 Atlantic puffin <NA> <NA> adult_unknown <NA>
## 9 Atlantic puffin <NA> <NA> adult_unknown <NA>
## 10 Atlantic puffin <NA> <NA> adult_unknown <NA>
##   status_sexing_method status_date weight scull tarsus wing
##   <chr> <date> <dbl> <dbl> <dbl> <dbl>
## 1 dna 2014-07-06 510 82.0 31.0 176
## 2 none_yet 2015-06-11 500 NA NA NA
## 3 dna 2014-07-06 540 78.0 29.5 182
## 4 none_yet 2015-07-01 530 NA NA NA
## 5 dna 2014-07-06 520 83.0 29.5 180
## 6 none_yet 2015-06-11 545 NA NA NA
## 7 none_yet 2016-06-05 535 83.0 29.6 183
## 8 dna 2014-07-06 455 80.0 28.4 175
## 9 none_yet 2015-06-25 460 NA NA NA
## 10 dna 2014-07-06 450 80.0 28.7 177
##   breeding_stage eggs chicks hatching_success breeding_success
##   <chr> <int> <int> <lgl> <lgl>
## 1 incubating 1 NA NA NA
## 2 rearing chicks NA NA T NA
## 3 incubating 1 NA NA NA
```

```
## 4 rearing chicks NA NA T NA
## 5 incubating 1 NA NA NA
## 6 rearing chicks NA NA T NA
## 7 breeding/stage_unknown NA NA NA NA
## 8 incubating 1 NA NA NA
## 9 breeding/stage_unknown NA NA NA NA
## 10 incubating 1 NA NA NA
## breeding_success_criterion data_responsible back_on_nest
## <chr> <chr> <lgl>
## 1 <NA> Rob Barret NA
## 2 none Kjell Einar Erikstad NA
## 3 <NA> Rob Barret NA
## 4 none Kjell Einar Erikstad NA
## 5 <NA> Rob Barret NA
## 6 none Kjell Einar Erikstad NA
## 7 none Kjell Einar Erikstad NA
## 8 <NA> Rob Barret NA
## 9 none Kjell Einar Erikstad NA
## 10 <NA> Rob Barret NA
## comment latest_sex latest_sexing_method latest_age
## <chr> <chr> <chr> <chr>
## 1 BLOOD FOR SEXING <NA> dna adult_unknown
## 2 <NA> <NA> dna adult_unknown
## 3 BLOOD FOR SEXING <NA> dna adult_unknown
## 4 <NA> <NA> dna adult_unknown
## 5 BLOOD FOR SEXING <NA> dna adult_unknown
## 6 <NA> <NA> dna adult_unknown
## 7 <NA> <NA> dna adult_unknown
## 8 BLOOD FOR SEXING <NA> dna adult_unknown
## 9 <NA> <NA> dna adult_unknown
## 10 BLOOD FOR SEXING <NA> dna adult_unknown
## latest_info_date eventType
## <date> <chr>
## 1 2016-06-18 Deployment
## 2 2016-06-18 Retrieval
## 3 2016-07-05 Deployment
## 4 2016-07-05 Retrieval
## 5 2016-06-05 Deployment
## 6 2016-06-05 Retrieval
## 7 2016-06-05 <NA>
## 8 2015-06-25 Deployment
## 9 2015-06-25 Retrieval
## 10 2016-06-28 Deployment
## # ... with 195 more rows
```

Commonly used info

I have made a couple of views for som common information, that are displayed on the shiny app <http://view.nina.no/seatrack/>. These can be found like this as well.

```
shorttable <- dbReadTable(con, Id(schema = "views", table = "shorttable"))
shorttable
```

```
## Antall.arter Antall.kolonier Antall.år Antall.positions Antall.individer
```

```
## 1      11      36      9      1759448      2123
##  Antall.tracks..logger_yeartracked.
## 1      3392

shorttableeqfilter3 <- dbReadTable(con, Id(schema = "views", table = "shorttableeqfilter3"))
shorttableeqfilter3

##  Antall.artar Antall.kolonier Antall.år Antall.positions Antall.individer
## 1      11      36      9      1126791      2121
##  Antall.tracks..logger_yeartracked.
## 1      3388
```