Example script for SpatialDeltaGLMM for spatio-temporal analysis of catch-rate data

James Thorson

October 10, 2016

Contents

1 Overview

This tutorial will walk through a simple example of how to use `SpatialDeltaGLMM` for estimating abundance indices, distribution shifts, and range expansion.

2 Getting started

First, we install necessary packages. We also have to install TMB as appropriate for the operating system (see directions elsewhere).

```
devtools::install_github("nwfsc-assess/geostatistical_delta-GLMM")
devtools::install_github("james-thorson/utilities")
```

Next load libraries…

```
library(TMB)               # Can instead load library(TMBdebug)
library(SpatialDeltaGLMM)
```

… and set location for saving files

```
DateFile = paste0(getwd(),'/SpatialDeltaGLMM_example/')
  dir.create(DateFile)
```

3 Settings

First chose an example data set for this script, as archived with package

```
Data_Set = c("Chatham_rise_hake", "Iceland_cod", "WCGBTS_canary",
    "GSL_american_plaice", "BC_pacific_cod", "EBS_pollock",
    "GOA_Pcod", "GOA_pollock", "GB_spring_haddock",
    "GB_fall_haddock", "SAWC_jacopever", "Aleutian_islands_POP")[6]
```

Next use latest version for CPP code

```
Version = "geo_index_v4b"
```

3.1 Spatial settings

The following settings define the spatial resolution for the model, and whether to use a grid or mesh approximation

```
Method = c("Grid", "Mesh")[2]
grid_size_km = 25
n_x = c(100, 250, 500, 1000, 2000)[1] # Number of stations
Kmeans_Config = list( "randomseed"=1, "nstart"=100, "iter.max"=1e3 )
```

3.2 Model settings

The following settings define whether to include spatial and spatio-temporal variation, whether its autocorrelated, and whether there's overdispersion

```
FieldConfig = c(Omega1 = 1, Epsilon1 = 1, Omega2 = 1,
    Epsilon2 = 1)
RhoConfig = c(Beta1 = 0, Beta2 = 0, Epsilon1 = 0, Epsilon2 = 0)
VesselConfig = c(Vessel = 0, VesselYear = 0)
ObsModel = 2
```

3.3 Stratification for results

We also define any potential stratification of results, and settings specific to any case-study data set

```
# Default
if (Data_Set %in% c("GSL_american_plaice", "BC_pacific_cod",
    "EBS_pollock", "SAWC_jacopever", "Chatham_rise_hake",
    "Aleutian_islands_POP")) {
    strata.limits <- data.frame(STRATA = "All_areas")
}
# Specific (useful as examples)
if (Data_Set %in% c("WCGBTS_canary", "Sim")) {
    # In this case, it will calculate a coastwide
    # index, and also a separate index for each state
    # (although the state lines are approximate)
    strata.limits <- data.frame(STRATA = c("Coastwide",
        "CA", "OR", "WA"), north_border = c(49, 42,
        46, 49), south_border = c(32, 32, 42, 46),
        shallow_border = c(55, 55, 55, 55), deep_border = c(1280,
            1280, 1280, 1280))
    # Override default settings for vessels
    VesselConfig = c(Vessel = 0, VesselYear = 1)
}
if (Data_Set %in% c("GOA_Pcod", "GOA_pollock")) {
    # In this case, will calculating an unrestricted
    # index and a separate index restricted to west of
    # -140W
    strata.limits <- data.frame(STRATA = c("All_areas",
        "west_of_140W"), west_border = c(-Inf, -Inf),
        east_border = c(Inf, -140))
}
if (Data_Set %in% c("GB_spring_haddock", "GB_fall_haddock")) {
    # For NEFSC indices, strata must be specified as a
    # named list of area codes
    strata.limits = list(Georges_Bank = c(1130, 1140,
        1150, 1160, 1170, 1180, 1190, 1200, 1210, 1220,
        1230, 1240, 1250, 1290, 1300))
}
if (Data_Set %in% c("Iceland_cod")) {
    strata.limits = data.frame(STRATA = "All_areas")
    # Turn off all spatial, temporal, and
    # spatio-temporal variation in probability of
    # occurrence, because they occur almost everywhere
    FieldConfig = c(Omega1 = 0, Epsilon1 = 0, Omega2 = 1,
        Epsilon2 = 1)
    RhoConfig = c(Beta1 = 3, Beta2 = 0, Epsilon1 = 0,
```

```
        Epsilon2 = 0)
}
```

3.4 Derived objects

Depending on the case study, we define a `Region` used when extrapolating or plotting density estimates. If its a different data set, it will define `Region="Other"`, and this is a recognized level for all uses of `Region` (which attempts to define reasonable settings based on the location of sampling). For example `Data_Set="Iceland_cod"` has no associated meta-data for the region, so it uses `Region="Other"` by default.

```
Region = switch( Data_Set, "Chatham_rise_hake"="New_Zealand",
                 "WCGBTS_canary"="California_current",
                 "GSL_american_plaice"="Gulf_of_St_Lawrence",
                 "BC_pacific_cod"="British_Columbia",
                 "EBS_pollock"="Eastern_Bering_Sea",
                 "GOA_Pcod"="Gulf_of_Alaska",
                 "GOA_pollock"="Gulf_of_Alaska",
                 "GB_spring_haddock"="Northwest_Atlantic",
                 "GB_fall_haddock"="Northwest_Atlantic",
                 "SAWC_jacopever"="South_Africa",
                 "Aleutian_islands_POP"="Aleutian_Islands",
                 "Other")
```

I also like to save all settings for later reference

```
Record = ThorsonUtilities::bundlelist(c("Data_Set",
    "strata.limits", "Region", "Version", "Method",
    "grid_size_km", "n_x", "FieldConfig", "RhoConfig",
    "VesselConfig", "ObsModel", "Kmeans_Config"))
save(Record, file = file.path(DateFile, "Record.RData"))
capture.output(Record, file = paste0(DateFile, "Record.txt"))
```

4 Prepare the data

4.1 Data-frame for catch-rate data

Depending upon the `Data_Set` chosen, we load different data sets for this example. Each results in a data-frame `Data_Geostat` with a standardized set of columns. For a new data set, the user is responsible for formatting `Data_Geostat` appropriately to match the example format.

```
if (Data_Set == "WCGBTS_canary") {
    data(WCGBTS_Canary_example, package = "SpatialDeltaGLMM")
    Year = as.numeric(sapply(WCGBTS_Canary_example[,
        "PROJECT_CYCLE"], FUN = function(Char) {
        strsplit(as.character(Char), " ")[[1]][2]
    }))
    Data_Geostat = data.frame(Catch_KG = WCGBTS_Canary_example[,
        "HAUL_WT_KG"], Year = Year, Vessel = WCGBTS_Canary_example[,
        "VESSEL"], AreaSwept_km2 = WCGBTS_Canary_example[,
        "AREA_SWEPT_HA"]/100, Lat = WCGBTS_Canary_example[,
```

```
        "BEST_LAT_DD"], Lon = WCGBTS_Canary_example[,
        "BEST_LON_DD"], Pass = WCGBTS_Canary_example[,
        "PASS"] - 1.5)
}
if (Data_Set %in% c("BC_pacific_cod")) {
    data(BC_pacific_cod_example, package = "SpatialDeltaGLMM")
    Data_Geostat = data.frame(Catch_KG = BC_pacific_cod_example[,
        "PCOD_WEIGHT"], Year = BC_pacific_cod_example[,
        "Year"], Vessel = "missing", AreaSwept_km2 = BC_pacific_cod_example[,
        "TOW.LENGTH..KM."]/100, Lat = BC_pacific_cod_example[,
        "LAT"], Lon = BC_pacific_cod_example[, "LON"],
        Pass = 0)
}
if (Data_Set %in% c("GSL_american_plaice")) {
    data(GSL_american_plaice, package = "SpatialDeltaGLMM")
    Print_Message("GSL_american_plaice")
    Data_Geostat = data.frame(Year = GSL_american_plaice[,
        "year"], Lat = GSL_american_plaice[, "latitude"],
        Lon = GSL_american_plaice[, "longitude"], Vessel = "missing",
        AreaSwept_km2 = GSL_american_plaice[, "swept"],
        Catch_KG = GSL_american_plaice[, "biomass"] *
            GSL_american_plaice[, "vstd"])
}
if (Data_Set == "EBS_pollock") {
    data(EBS_pollock_data, package = "SpatialDeltaGLMM")
    Data_Geostat = data.frame(Catch_KG = EBS_pollock_data[,
        "catch"], Year = EBS_pollock_data[, "year"],
        Vessel = "missing", AreaSwept_km2 = 0.01, Lat = EBS_pollock_data[,
            "lat"], Lon = EBS_pollock_data[, "long"],
        Pass = 0)
}
if (Data_Set == "GOA_Pcod") {
    data(GOA_pacific_cod, package = "SpatialDeltaGLMM")
    Data_Geostat = data.frame(Catch_KG = GOA_pacific_cod[,
        "catch"], Year = GOA_pacific_cod[, "year"],
        Vessel = "missing", AreaSwept_km2 = 0.01, Lat = GOA_pacific_cod[,
            "lat"], Lon = GOA_pacific_cod[, "lon"],
        Pass = 0)
}
if (Data_Set == "GOA_pollock") {
    data(GOA_walleye_pollock, package = "SpatialDeltaGLMM")
    Data_Geostat = data.frame(Catch_KG = GOA_walleye_pollock[,
        "catch"], Year = GOA_walleye_pollock[, "year"],
        Vessel = "missing", AreaSwept_km2 = 0.01, Lat = GOA_walleye_pollock[,
            "lat"], Lon = GOA_walleye_pollock[, "lon"],
        Pass = 0)
}
if (Data_Set == "Aleutian_islands_POP") {
    data(AI_pacific_ocean_perch, package = "SpatialDeltaGLMM")
    Data_Geostat = data.frame(Catch_KG = AI_pacific_ocean_perch[,
        "cpue..kg.km.2."], Year = AI_pacific_ocean_perch[,
        "year"], Vessel = "missing", AreaSwept_km2 = 1,
        Lat = AI_pacific_ocean_perch[, "start.latitude"],
```

```
        Lon = AI_pacific_ocean_perch[, "start.longitude"],
        Pass = 0)
}
if (Data_Set == "GB_spring_haddock") {
    data(georges_bank_haddock_spring, package = "SpatialDeltaGLMM")
    Print_Message("GB_haddock")
    Data_Geostat = data.frame(Catch_KG = georges_bank_haddock_spring[,
        "CATCH_WT_CAL"], Year = georges_bank_haddock_spring[,
        "YEAR"], Vessel = "missing", AreaSwept_km2 = 0.0112 *
        1.852^2, Lat = georges_bank_haddock_spring[,
        "LATITUDE"], Lon = georges_bank_haddock_spring[,
        "LONGITUDE"])
}
if (Data_Set == "GB_fall_haddock") {
    data(georges_bank_haddock_fall, package = "SpatialDeltaGLMM")
    Print_Message("GB_haddock")
    Data_Geostat = data.frame(Catch_KG = georges_bank_haddock_fall[,
        "CATCH_WT_CAL"], Year = georges_bank_haddock_fall[,
        "YEAR"], Vessel = "missing", AreaSwept_km2 = 0.0112 *
        1.852^2, Lat = georges_bank_haddock_fall[,
        "LATITUDE"], Lon = georges_bank_haddock_fall[,
        "LONGITUDE"])
}
if (Data_Set == "SAWC_jacopever") {
    data(south_africa_westcoast_jacopever, package = "SpatialDeltaGLMM")
    Data_Geostat = data.frame(Catch_KG = south_africa_westcoast_jacopever[,
        "HELDAC"], Year = south_africa_westcoast_jacopever[,
        "Year"], Vessel = "missing", AreaSwept_km2 = south_africa_westcoast_jacopever[,
        "area_swept_nm2"] * 1.852^2, Lat = south_africa_westcoast_jacopever[,
        "cen_lat"], Lon = south_africa_westcoast_jacopever[,
        "cen_long"])
}
if (Data_Set %in% c("Iceland_cod")) {
    # WARNING: This data set has not undergone much
    # evaluation for spatio-temporal analysis
    data(iceland_cod, package = "SpatialDeltaGLMM")
    Data_Geostat = data.frame(Catch_KG = iceland_cod[,
        "Catch_b"], Year = iceland_cod[, "year"], Vessel = 1,
        AreaSwept_km2 = iceland_cod[, "towlength"],
        Lat = iceland_cod[, "lat1"], Lon = iceland_cod[,
            "lon1"])
}
if (Data_Set %in% c("Chatham_rise_hake")) {
    data(chatham_rise_hake, package = "SpatialDeltaGLMM")
    Data_Geostat = data.frame(Catch_KG = chatham_rise_hake[,
        "Hake_kg_per_km2"], Year = chatham_rise_hake[,
        "Year"], Vessel = 1, AreaSwept_km2 = 1, Lat = chatham_rise_hake[,
        "Lat"], Lon = chatham_rise_hake[, "Lon"])
}
Data_Geostat = na.omit(Data_Geostat)
```

4.2 Extrapolation grid

We also generate the extrapolation grid appropriate for a given region. For new regions, we use `Region="Other"`.

```
if (Region %in% c("California_current", "Eastern_Bering_Sea",
    "Gulf_of_Alaska", "Aleutian_Islands", "Northwest_Atlantic",
    "Gulf_of_St_Lawrence", "New_Zealand")) {
    Extrapolation_List = Prepare_Extrapolation_Data_Fn(Region = Region,
        strata.limits = strata.limits)
}
if (Region == "British_Columbia") {
    Extrapolation_List = Prepare_Extrapolation_Data_Fn(Region = Region,
        strata.limits = strata.limits, strata_to_use = c("HS",
            "QCS"))
}
if (Region == "South_Africa") {
    Extrapolation_List = Prepare_Extrapolation_Data_Fn(Region = Region,
        strata.limits = strata.limits, region = "west_coast")
}
if (Region == "Other") {
    Extrapolation_List = Prepare_Extrapolation_Data_Fn(Region = Region,
        strata.limits = strata.limits, observations_LL = Data_Geostat[,
            c("Lat", "Lon")], maximum_distance_from_sample = 15)
}
```

4.3 Derived objects for spatio-temporal estimation

And we finally generate the information used for conducting spatio-temporal parameter estimation, bundled in list `Spatial_List`

```
Spatial_List = Spatial_Information_Fn(grid_size_km = grid_size_km,
    n_x = n_x, Method = Method, Lon = Data_Geostat[,
        "Lon"], Lat = Data_Geostat[, "Lat"], Extrapolation_List = Extrapolation_List,
    randomseed = Kmeans_Config[["randomseed"]], nstart = Kmeans_Config[["nstart"]],
    iter.max = Kmeans_Config[["iter.max"]], DirPath = DateFile)
# Add knots to Data_Geostat
Data_Geostat = cbind(Data_Geostat, Spatial_List$loc_UTM,
    knot_i = Spatial_List$knot_i)
```

5 Build and run model

To estimate parameters, we first build a list of data-inputs used for parameter estimation. `Data_Fn` has some simple checks for buggy inputs, but also please read the help file `?Data_Fn`.

```
TmbData = Data_Fn(Version = Version, FieldConfig = FieldConfig,
    RhoConfig = RhoConfig, ObsModel = ObsModel, b_i = Data_Geostat[,
        "Catch_KG"], a_i = Data_Geostat[, "AreaSwept_km2"],
    v_i = as.numeric(Data_Geostat[, "Vessel"]) - 1,
    s_i = Data_Geostat[, "knot_i"] - 1, t_i = Data_Geostat[,
        "Year"], a_xl = Spatial_List$a_xl, MeshList = Spatial_List$MeshList,
```

```r
    GridList = Spatial_List$GridList, Method = Spatial_List$Method,
    Options = c(SD_site_density = 0, SD_site_logdensity = 0,
        Calculate_Range = 1, Calculate_evenness = 0,
        Calculate_effective_area = 1))
```

We then build the TMB object…

```r
TmbList = Build_TMB_Fn(TmbData = TmbData, RunDir = DateFile,
    Version = Version, RhoConfig = RhoConfig, VesselConfig = VesselConfig,
    loc_x = Spatial_List$loc_x)
Obj = TmbList[["Obj"]]
```

…and use a gradient-based nonlinear minimizer to identify maximum likelihood estimates for fixed-effects

```r
Opt = TMBhelper::Optimize(obj = Obj, lower = TmbList[["Lower"]],
    upper = TmbList[["Upper"]], getsd = TRUE, savedir = DateFile,
    bias.correct = FALSE)
```

Here I print the diagnostics generated during parameter estimation, and I confirm that (1) no parameter is hitting an upper or lower bound and (2) the final gradient for each fixed-effect is close to zero.

```r
print( Opt$diagnostics[,c('Param','Lower','MLE','Upper','final_gradient')] )
```

```
##          Param  Lower    MLE Upper final_gradient
## 1  ln_H_input -50.00  0.232 50.00       1.17e-03
## 2  ln_H_input -50.00 -0.966 50.00       1.36e-03
## 3     beta1_t -50.00  4.120 50.00      -2.23e-04
## 4     beta1_t -50.00  4.229 50.00       7.92e-04
## 5     beta1_t -50.00  4.323 50.00      -5.53e-05
## 6     beta1_t -50.00  5.093 50.00       1.11e-04
## 7     beta1_t -50.00  5.428 50.00       3.64e-04
## 8     beta1_t -50.00  4.105 50.00      -6.86e-04
## 9     beta1_t -50.00  5.056 50.00      -4.68e-05
## 10    beta1_t -50.00  4.168 50.00       1.64e-04
## 11    beta1_t -50.00  4.333 50.00       4.75e-04
## 12    beta1_t -50.00  5.989 50.00       1.79e-04
## 13    beta1_t -50.00  4.524 50.00      -3.08e-04
## 14    beta1_t -50.00  5.265 50.00       2.01e-04
## 15    beta1_t -50.00  5.647 50.00       4.70e-05
## 16    beta1_t -50.00  4.886 50.00      -3.30e-04
## 17    beta1_t -50.00  5.073 50.00      -2.26e-04
## 18    beta1_t -50.00  4.753 50.00      -7.90e-04
## 19    beta1_t -50.00  4.997 50.00       9.88e-04
## 20    beta1_t -50.00  6.219 50.00       1.41e-04
## 21    beta1_t -50.00  5.124 50.00      -3.53e-04
## 22    beta1_t -50.00  5.707 50.00      -1.71e-05
## 23    beta1_t -50.00  4.809 50.00      -2.35e-04
## 24    beta1_t -50.00  4.535 50.00       3.42e-04
## 25    beta1_t -50.00  5.454 50.00      -4.44e-04
## 26    beta1_t -50.00  4.746 50.00      -1.07e-03
## 27    beta1_t -50.00  4.572 50.00       8.06e-04
## 28    beta1_t -50.00  4.198 50.00      -1.20e-04
```

```
## 29    beta1_t -50.00  2.877 50.00       6.07e-04
## 30    beta1_t -50.00  3.426 50.00      -1.00e-03
## 31    beta1_t -50.00  2.986 50.00      -5.09e-06
## 32    beta1_t -50.00  4.660 50.00       1.91e-04
## 33    beta1_t -50.00  4.657 50.00      -6.31e-04
## 34    beta1_t -50.00  5.190 50.00       5.49e-04
## 35    beta1_t -50.00  6.231 50.00       3.24e-05
## 36   logetaE1 -50.00 -1.240  3.34      -1.94e-03
## 37   logetaO1 -50.00 -1.932  3.34      -7.60e-04
## 38  logkappa1  -6.01 -4.120 -2.57      -6.21e-04
## 39    beta2_t -50.00  7.517 50.00       8.06e-05
## 40    beta2_t -50.00  8.740 50.00      -8.44e-05
## 41    beta2_t -50.00  7.844 50.00       4.19e-05
## 42    beta2_t -50.00  8.535 50.00       8.66e-05
## 43    beta2_t -50.00  8.097 50.00       3.48e-05
## 44    beta2_t -50.00  8.459 50.00       1.28e-04
## 45    beta2_t -50.00  8.287 50.00       2.48e-05
## 46    beta2_t -50.00  8.243 50.00       6.68e-05
## 47    beta2_t -50.00  8.046 50.00      -1.23e-04
## 48    beta2_t -50.00  8.170 50.00      -6.58e-06
## 49    beta2_t -50.00  8.063 50.00       8.62e-05
## 50    beta2_t -50.00  8.212 50.00      -1.49e-04
## 51    beta2_t -50.00  8.008 50.00      -1.23e-05
## 52    beta2_t -50.00  7.516 50.00       5.43e-05
## 53    beta2_t -50.00  7.730 50.00       1.37e-04
## 54    beta2_t -50.00  7.887 50.00       2.14e-05
## 55    beta2_t -50.00  7.663 50.00       2.91e-04
## 56    beta2_t -50.00  7.405 50.00       1.17e-04
## 57    beta2_t -50.00  8.198 50.00      -3.93e-06
## 58    beta2_t -50.00  8.166 50.00      -9.63e-05
## 59    beta2_t -50.00  7.847 50.00       1.20e-04
## 60    beta2_t -50.00  8.542 50.00      -2.22e-04
## 61    beta2_t -50.00  7.983 50.00       5.57e-05
## 62    beta2_t -50.00  7.833 50.00      -1.80e-04
## 63    beta2_t -50.00  7.130 50.00      -1.54e-05
## 64    beta2_t -50.00  6.996 50.00      -1.34e-04
## 65    beta2_t -50.00  6.544 50.00      -2.92e-04
## 66    beta2_t -50.00  6.056 50.00       2.96e-04
## 67    beta2_t -50.00  7.291 50.00      -1.15e-04
## 68    beta2_t -50.00  7.546 50.00       3.98e-05
## 69    beta2_t -50.00  7.248 50.00      -1.29e-05
## 70    beta2_t -50.00  7.513 50.00       9.57e-05
## 71    beta2_t -50.00  8.565 50.00      -1.94e-04
## 72   logetaE2 -50.00 -1.382  3.34       6.04e-04
## 73   logetaO2 -50.00 -1.367  3.34       1.04e-03
## 74  logkappa2  -6.01 -4.535 -2.57      -1.11e-03
## 75  logSigmaM -50.00  0.168 10.00      -5.69e-03
```

Finally, we bundle and save output

```
Report = Obj$report()
Save = list("Opt"=Opt, "Report"=Report, "ParHat"=Obj$env$parList(Opt$par), "TmbData"=TmbData)
save(Save, file=paste0(DateFile,"Save.RData"))
```

6 Save plots

Last but not least, we generate useful plots by first determining which years to plot (`Years2Include`), and labels for each plotted year (`Year_Set`)

```
Year_Set = seq(min(Data_Geostat[,'Year']),max(Data_Geostat[,'Year']))
Years2Include = which( Year_Set %in% sort(unique(Data_Geostat[,'Year'])))
```

We then run a set of pre-defined plots for visualizing results

6.1 Direction of "geometric anisotropy"

We can visualize which direction has faster or slower decorrelation (termed "geometric anisotropy")

```
PlotAniso_Fn( FileName=paste0(DateFile,"Aniso.png"), Report=Report, TmbData=TmbData )
```

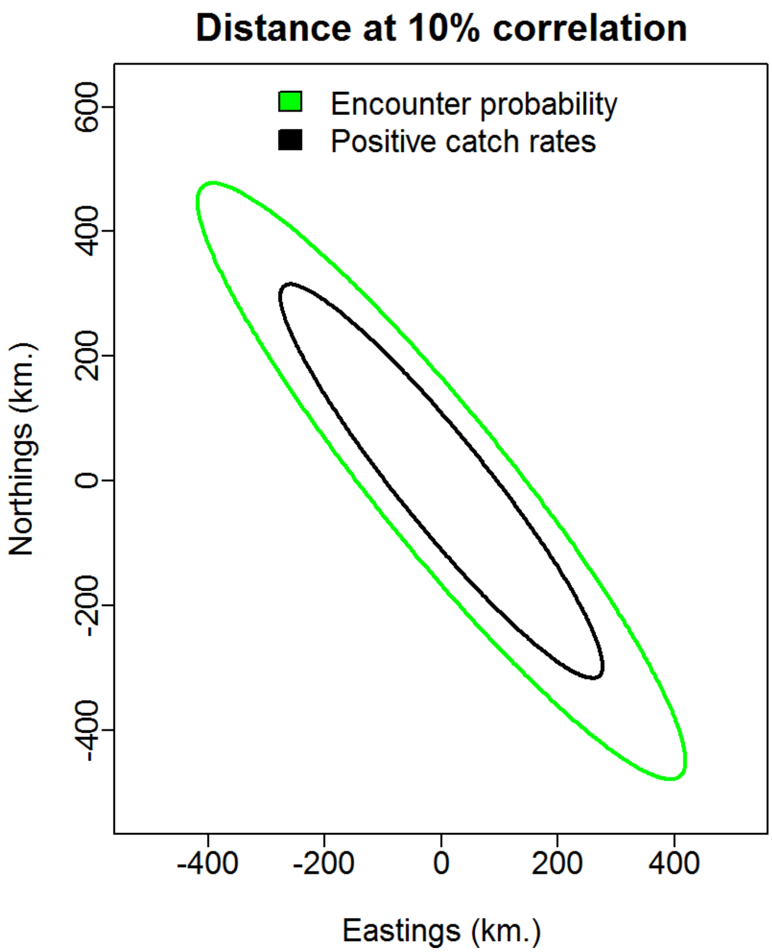

Figure 1: Decorrelation distance for different directions

6.2 Density surface for each year

We can visualize many types of output from the model. Here I only show predicted density, but other options are obtained via other integers passed to `plot_set` as described in `?PlotResultsOnMap_Fn`

```
# Get region-specific settings for plots
MapDetails_List = MapDetails_Fn(Region = Region, NN_Extrap = Spatial_List$PolygonList$NN_Extrap,
    Extrapolation_List = Extrapolation_List)
# Plot maps representing density or other variables
PlotResultsOnMap_Fn(plot_set = c(3), MappingDetails = MapDetails_List[["MappingDetails"]],
    Report = Report, Sdreport = Opt$SD, PlotDF = MapDetails_List[["PlotDF"]],
    MapSizeRatio = MapDetails_List[["MapSizeRatio"]],
    Xlim = MapDetails_List[["Xlim"]], Ylim = MapDetails_List[["Ylim"]],
    FileName = DateFile, Year_Set = Year_Set, Years2Include = Years2Include,
    Rotate = MapDetails_List[["Rotate"]], Cex = MapDetails_List[["Cex"]],
    Legend = MapDetails_List[["Legend"]], zone = MapDetails_List[["Zone"]],
    mar = c(0, 0, 2, 0), oma = c(3.5, 3.5, 0, 0), cex = 1.8)
```

6.3 Index of abundance

The index of abundance is generally most useful for stock assessment models.

```
PlotIndex_Fn(DirName = DateFile, TmbData = TmbData,
    Sdreport = Opt[["SD"]], Year_Set = Year_Set, Years2Include = Years2Include,
    strata_names = strata.limits[, 1], use_biascorr = TRUE)
```

6.4 Center of gravity and range expansion/contraction

We can detect shifts in distribution or range expansion/contraction.

```
Plot_range_shifts(Report = Report, TmbData = TmbData,
    Sdreport = Opt[["SD"]], Znames = colnames(TmbData$Z_xm),
    PlotDir = DateFile)
```

6.5 Vessel effects if included

Most example data-sets don't have vessel effects, so this plot is generally skipped

```
Return = Vessel_Fn(TmbData = TmbData, Sdreport = Opt[["SD"]],
    FileName_VYplot = paste0(DateFile, "VY-effect.jpg"))
```

```
## Not plotting vessel effects because none are present
```

6.6 Quantile-quantile plot for positive-catch-rate component

We can visualize fit to residuals of catch-rates given encounters using a Q-Q plot. A good Q-Q plot will have residuals along the one-to-one line.

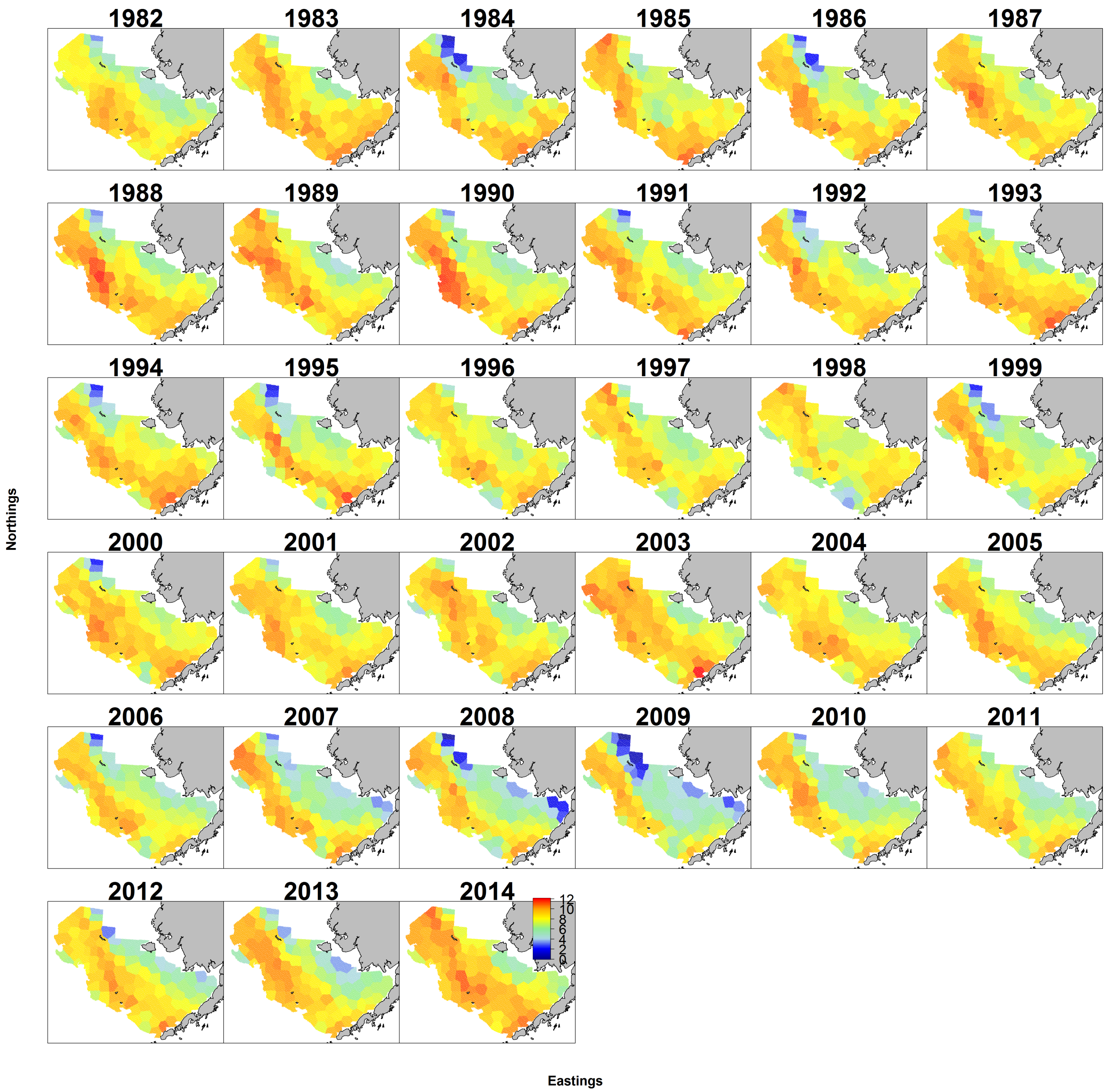

Figure 2: Density maps for each year

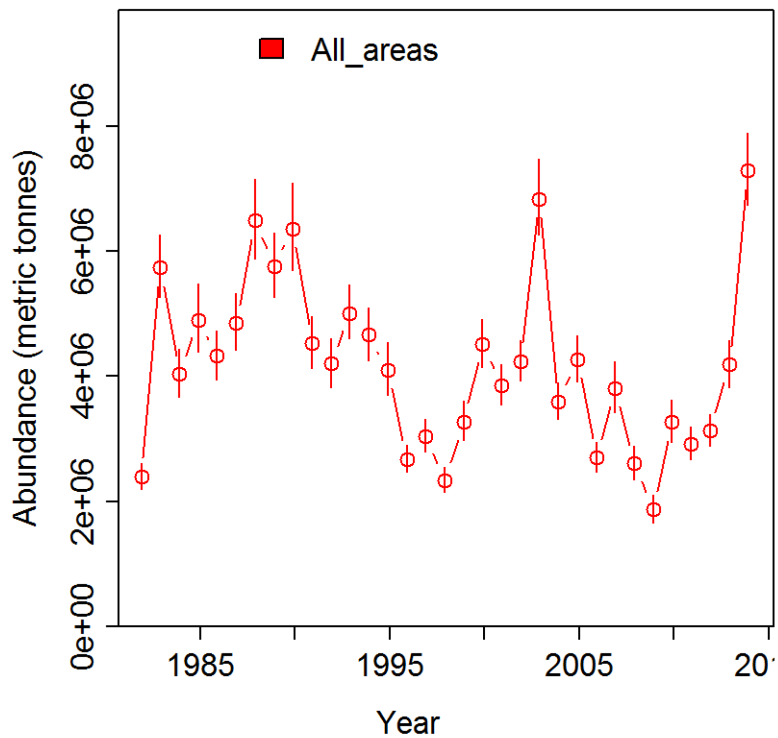

Figure 3: Index of abundance plus/minus 1 standard error

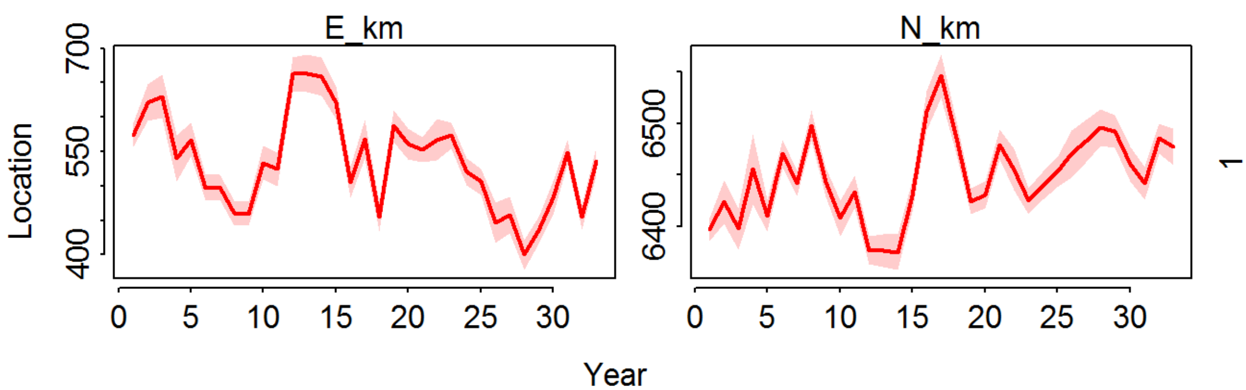

Figure 4: Center of gravity (COG) indicating shifts in distribution plus/minus 1 standard error

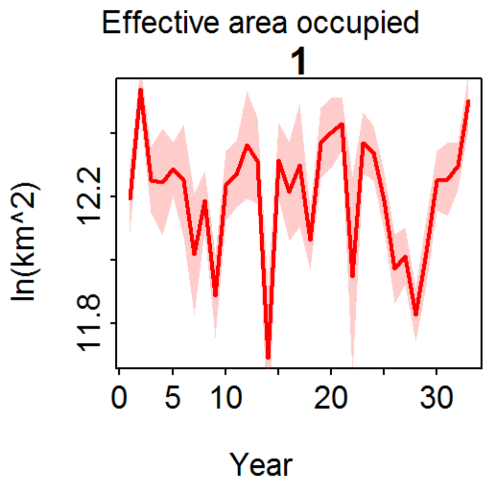

Figure 5: Effective area occupied indicating range expansion/contraction plus/minus 1 standard error

```
Q = QQ_Fn(TmbData = TmbData, Report = Report, FileName_PP = paste0(DateFile,
    "Posterior_Predictive.jpg"), FileName_Phist = paste0(DateFile,
    "Posterior_Predictive-Histogram.jpg"), FileName_QQ = paste0(DateFile,
    "Q-Q_plot.jpg"), FileName_Qhist = paste0(DateFile,
    "Q-Q_hist.jpg"))
```

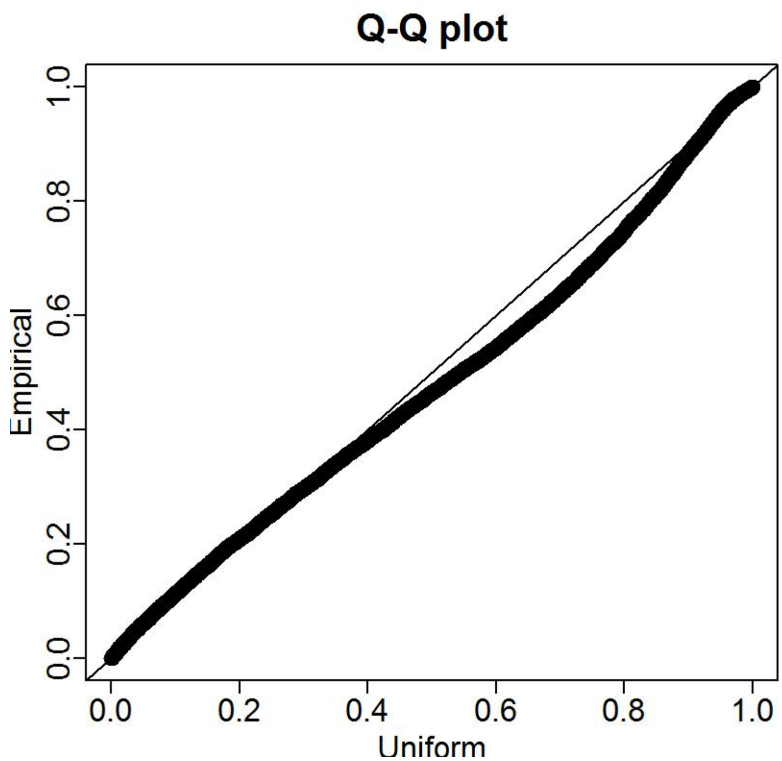

Figure 6: Quantile-quantile plot indicating residuals for “positive catch rate” component