Draft

harvesStrategy R Package

Notes on usage

Dr Stephen E. Lane[1]

[1]Chief Scientist, interadata

March 25, 2018

interadata
bespoke data science.

Contents

List of Code Examples

1 Introduction

This report provides instructions on how to use the `harvestStrategy` R package, developed for Dr Harry Gorfine on behalf of the Victorian Fisheries Authority. Full documentation of the code is provided as part of the `harvestStrategy` package, along with a vignette demonstrating its use. This document provides the key steps to get up and running with the package.

During the development of this package, extra data requirements were discovered. We detail those requirements in Section 4 of this report.

2 Installation

The first step is to install the package (Code Chunk 2.1). The package should install all dependencies required, however the `devtools` R package is required to install from interadata's Bitbucket[1] website.

R Chunk 2.1.

```r
## Make sure that devtools is installed. If the following command fails, install
## devtools with: install.packages("devtools")
library(devtools)
install_bitbucket("interadata/harvestStrategy", build_vignettes = TRUE)
```

3 Loading the Package and Getting Help

The `harvestStrategy` package is fully documented. Code Chunk 3.1 shows how to load the package, how to get help on the `categoriseIndicator` function, and how to open the vignette.

R Chunk 3.1.

```r
library(harvestStrategy)
?categoriseIndicator
vignette("intro-harvest-strategy", "harvestStrategy")
```

4 Data Requirements

If you read through the vignette (see Code Chunk 3.1), you'll notice that the package requires additional data to what was provided to interadata in the "HS_DRAFT_180131_Data for reference points_AWG.xlsx" and "HS_DRAFT_SPREADSHEET_AWG_SPECS.xlsx" spreadsheets. The extra data relate to the historical threshold results. These data are required due to the harvest strategy framework as depicted in Figure 1 in the "Draft abalone harvest strategy_V2.docx" report (Draft HS report) provided to interadata.

What this means, is that when a harvest strategy has been decided upon for a new year, the Catch Per Unit Effort (CPUE) threshold (as per Figure 1, Draft HS report)

[1] Bitbucket is a code-hosting website, similar to GitHub.

become part of the recorded data for the new year, along with CPUE etc when the season has completed.

Note: due to the setup of the `harvestStrategy` package, the CPUE threshold data are to be entered exactly as shown in the Figure 1 flowchart: *Green*, *Yellow*, and *Red*. The vignette shows how this is used, and we provide an example in the next section.

5 Using `harvestStrategy`

Once the library is loaded (as per Code Chunk 3.1), we need to load in the data. We use the "HS_DRAFT_180131_Data for reference points_AWG.xlsx" spreadsheet as an example. In Code Chunk 5.1, you'll notice that there are no CPUE thresholds entered in the data — we will assume that for the first year that the harvest strategy is run, all thresholds are *Green*.

R Chunk 5.1.

```
library(dplyr)
hs_data

## # A tibble: 874 x 11
##    SMU         Year Catch_79 OptimumTarget Nom_CPUE_79_16 LegalBiomass_199~
##    <chr>      <dbl>    <dbl>         <dbl>          <dbl>             <dbl>
##  1 CZ_BACK.~  1979    45295            NA           50.3                NA
##  2 CZ_BACK.~  1980    53028            NA           44.1                NA
##  3 CZ_BACK.~  1981    56376            NA           44.5                NA
##  4 CZ_BACK.~  1982    39496            NA           51.8                NA
##  5 CZ_BACK.~  1983    53044            NA           51.9                NA
##  6 CZ_BACK.~  1984    54990            NA           51.3                NA
##  7 CZ_BACK.~  1985    61839            NA           50.9                NA
##  8 CZ_BACK.~  1986    53767            NA           64.0                NA
##  9 CZ_BACK.~  1987    51434            NA           72.7                NA
## 10 CZ_BACK.~  1988    53512            NA           76.8                NA
## # ... with 864 more rows, and 5 more variables: SSB_1995_75 <dbl>,
## #   `Under_Count_1995(20mm)` <dbl>, `FIS mean weight` <dbl>, `Logger mean
## #   weight` <dbl>, `Logger kg/ha` <dbl>
```

We will demonstrate the steps of the harvest strategy framework as per the Draft HS report, on the 2016 data.

5.1 Step 1: Determine the CPUE Category

Step 1 is to determine the CPUE category. This determines the *Green*, *Yellow*, or *Red* classification. First, we require CPUE reference limits, as per Table 1 of the Draft HS report. These need to be entered in tabular form (e.g. in a spreadsheet) which interadata has done. Code Chunk 5.2 shows the required format.

R Chunk 5.2.

```
cpue_ref

## # A tibble: 23 x 4
##    SMU                     Limit Threshold Target
##    <chr>                   <int>     <int>  <int>
##  1 WZ_JULIA.PERCY.ISLAND      30        50     90
##  2 WZ_PORT.FAIRY              30        50     70
##  3 WZ_PORTLAND                20        40    100
##  4 WZ_WARRNAMBOOL             30        50     70
##  5 CZ_BACK.BEACHES            50        70    100
##  6 CZ_CAPE.LIPTRAP            40        60    120
##  7 CZ_CAPE.OTWAY              50        70    100
##  8 CZ_CLIFFY.GROUP            40        60    110
##  9 CZ_FLINDERS                50        70    100
## 10 CZ_KILCUNDA                50        70    110
## # ... with 13 more rows
```

Code Chunk 5.3 shows how to use the `catchControlStrategy` function.

R Chunk 5.3.

```
library(purrr)
cpue_thresh <- hs_data %>%
    filter(Year < 2016) %>%
    mutate(thresh = "Green") %>%
    select(SMU, Year, thresh)
hs_2016 <- hs_data %>%
    filter(Year == 2016) %>%
    mutate(
        thresh = "Green",
        ccr = purrr::pmap(list(SMU, Year, Nom_CPUE_79_16), catchControlStrategy,
                          hist_ccs = cpue_thresh,
                          cpue_ref = cpue_ref,
                          ccs_warning_nm = "thresh")
    ) %>%
    tidyr::unnest()
hs_2016 %>%
    select(SMU, Year, Nom_CPUE_79_16, CCS_Warning, CCS)

## # A tibble: 23 x 5
##    SMU                Year Nom_CPUE_79_16 CCS_Warning CCS
##    <chr>             <dbl>          <dbl> <chr>       <chr>
##  1 CZ_BACK.BEACHES    2016           81.3 Green       CCR1
##  2 CZ_CAPE.LIPTRAP    2016           75.7 Green       CCR1
##  3 CZ_CAPE.OTWAY      2016           73.4 Green       CCR1
##  4 CZ_CLIFFY.GROUP    2016           78.3 Green       CCR1
##  5 CZ_FLINDERS        2016           61.3 Yellow      CCR1
##  6 CZ_KILCUNDA        2016           65.8 Yellow      CCR1
##  7 CZ_PHILLIP.ISLAND  2016           76.0 Green       CCR1
##  8 CZ_PPB             2016           NA   Yellow      CCR1
##  9 CZ_PROM.EASTSIDE   2016           51.8 Green       CCR1
## 10 CZ_PROM.WESTSIDE   2016           62.3 Yellow      CCR1
## # ... with 13 more rows
```

There are a couple of things to note in Code Chunk XXX: first, we create the `cpue_thresh` data from years 2015 and earlier — this would already exist, and include the `thresh` column as data; second, we use `purrr::map` to run the `catchControlStrategy` function over all of the Spatial Management Units (SMU's). Enter `?purrr::map` on the R console to get help on that function.

Also included in the output is the catch control rule (in the CCS column). This is used to implement the lower and upper ranges of the optimal target in Step 5 (Section 5.5).

5.2 Step 2: Calculate the Relative Change in the Performance Indicators

In this step, we calculate the relative change in the performance indicators. We will demonstrate this using the relative change of the four-year gradient (linear regression model) as the primary indicator, the year-to-year ratio as the secondary indicator, and the relative change of the FIS unders as the tertiary ratio.

Code Chunk 5.4 demonstrates the two steps involved: calculate the four-year gradient indicator from the raw data, then calculate the relative change.

R Chunk 5.4.

```
hs_indicators_2016 <- hs_data %>%
    group_by(SMU) %>%
    mutate(
        CPUE_gradient4 = movingIndicator(Nom_CPUE_79_16, movingGradient, 4),
        CPUE_gradient_ratio = ratioIndicator(CPUE_gradient4),
        CPUE_ratio = ratioIndicator(Nom_CPUE_79_16),
        Unders_ratio = ratioIndicator(`Under_Count_1995(20mm)`)
    ) %>%
    filter(Year == 2016)
hs_2016 <- left_join(
    hs_2016,
    hs_indicators_2016 %>%
    select(SMU, CPUE_gradient_ratio, CPUE_ratio, Unders_ratio),
    by = "SMU"
)
hs_2016 %>%
    select(SMU, CPUE_gradient_ratio, CPUE_ratio, Unders_ratio)

## # A tibble: 23 x 4
##    SMU                  CPUE_gradient_ratio CPUE_ratio Unders_ratio
##    <chr>                              <dbl>      <dbl>        <dbl>
##  1 CZ_BACK.BEACHES                   0.190       0.911        0.542
##  2 CZ_CAPE.LIPTRAP                 - 0.466       0.816       NA
##  3 CZ_CAPE.OTWAY                     0.671       1.09         1.30
##  4 CZ_CLIFFY.GROUP                 - 1.25        1.07        NA
##  5 CZ_FLINDERS                       0.200       0.905        1.16
##  6 CZ_KILCUNDA                       0.0953      1.04        NA
##  7 CZ_PHILLIP.ISLAND                 0.294       0.894        0.564
##  8 CZ_PPB                           NA          NA           NA
##  9 CZ_PROM.EASTSIDE                  1.65        0.885       NA
## 10 CZ_PROM.WESTSIDE                  0.827       1.03         1.24
## # ... with 13 more rows
```

Notice we use the whole data set (`hs_data`) here to calculate the indicators. Once the indicators are calculated, they are restricted to the year of interest (2016) and merged back on to the `hs_2016` dataset.

5.3 Step 3: Calculate the Indicator Categorisations

The next step is to calculate the categorisations of the indicators. We use the bounds as given in the Draft HS report to set whether an indicator is Increasing ($> 5\%$ change), Decreasing ($< -5\%$ change), or else Stable. Code Chunk 5.5 demonstrates the setting of these categorisations.

R Chunk 5.5.

```
hs_2016 <- hs_2016 %>%
    mutate(
        CPUE_gradient_cat = categoriseIndicator(
            CPUE_gradient_ratio, c(-0.05, 0.05)
        ),
        CPUE_cat = categoriseIndicator(
            CPUE_ratio, c(-0.05, 0.05)
        ),
        Unders_cat = categoriseIndicator(
            Unders_ratio, c(-0.05, 0.05)
        )
    )
hs_2016 %>%
    select(SMU, CPUE_gradient_cat, CPUE_cat, Unders_cat)

## # A tibble: 23 x 4
##    SMU               CPUE_gradient_cat CPUE_cat   Unders_cat
##    <chr>             <chr>             <chr>      <chr>
##  1 CZ_BACK.BEACHES   Increasing        Increasing Increasing
##  2 CZ_CAPE.LIPTRAP   Decreasing        Increasing Stable
##  3 CZ_CAPE.OTWAY     Increasing        Increasing Increasing
##  4 CZ_CLIFFY.GROUP   Decreasing        Increasing Stable
##  5 CZ_FLINDERS       Increasing        Increasing Increasing
##  6 CZ_KILCUNDA       Increasing        Increasing Stable
##  7 CZ_PHILLIP.ISLAND Increasing        Increasing Increasing
##  8 CZ_PPB            Stable            Stable     Stable
##  9 CZ_PROM.EASTSIDE  Increasing        Increasing Stable
## 10 CZ_PROM.WESTSIDE  Increasing        Increasing Increasing
## # ... with 13 more rows
```

If data is missing (as shown in Code Chunk 5.4), the default categorisation of Stable will be applied.

5.4 Step 4: Calculate the Final Categorisation

We can now calculate the final categorisation of each SMU. This will be a compound categorisation, based on decision matrices provided by the VFA. There are two decision matrices required; these are shown in Tables 3 and 4 of the Draft HS. These tables have been hard-coded into the `harvestStrategy` package; they are required to be loaded using `data` statements, which are shown in Code Chunk 5.6.

R Chunk 5.6.

```
data(prim_sec)
data(compound_tertiary)
prim_sec

## # A tibble: 9 x 3
##   primary_category secondary_category decision
##   <chr>            <chr>              <chr>
## 1 Decreasing       Decreasing         Decreasing
## 2 Stable           Decreasing         Decreasing
## 3 Increasing       Decreasing         Stable
## 4 Decreasing       Stable             Decreasing
## 5 Stable           Stable             Stable
## 6 Increasing       Stable             Increasing
## 7 Decreasing       Increasing         Decreasing
## 8 Stable           Increasing         Stable
## 9 Increasing       Increasing         Increasing

compound_tertiary

## # A tibble: 9 x 3
##   compound_category tertiary_category decision
##   <chr>             <chr>             <chr>
## 1 Decreasing        Decreasing        Decreasing
## 2 Stable            Decreasing        Stable
## 3 Increasing        Decreasing        Stable
## 4 Decreasing        Stable            Decreasing
## 5 Stable            Stable            Stable
## 6 Increasing        Stable            Increasing
## 7 Decreasing        Increasing        Stable
## 8 Stable            Increasing        Stable
## 9 Increasing        Increasing        Increasing
```

With these decision matrices, it is then a two-step calculation to arrive at the final categorisation: (i) combine the primary and secondary categorisations into a compound categorisation, and (ii) combine the compound categorisation and tertiary categorisation into the final categorisation. These steps are shown in Code Chunk 5.7.

R Chunk 5.7.

```
hs_2016 <- hs_2016 %>%
    mutate(prim_sec_cat = purrr::map2_chr(
                                  CPUE_gradient_cat,
                                  CPUE_cat,
                                  compoundCategorisation,
                                  decisionRule = prim_sec),
           final_cat = purrr::map2_chr(
                                  prim_sec_cat,
                                  Unders_cat,
                                  compoundCategorisation,
                                  decisionRule = compound_tertiary,
                                  primary_category = "compound_category",
                                  secondary_category = "tertiary_category"
                              )
           )
hs_2016 %>%
    select(SMU, prim_sec_cat, final_cat)

## # A tibble: 23 x 3
##    SMU                prim_sec_cat final_cat
##    <chr>              <chr>        <chr>
##  1 CZ_BACK.BEACHES    Increasing   Increasing
##  2 CZ_CAPE.LIPTRAP    Decreasing   Decreasing
##  3 CZ_CAPE.OTWAY      Increasing   Increasing
##  4 CZ_CLIFFY.GROUP    Decreasing   Decreasing
##  5 CZ_FLINDERS        Increasing   Increasing
##  6 CZ_KILCUNDA        Increasing   Increasing
##  7 CZ_PHILLIP.ISLAND Increasing   Increasing
##  8 CZ_PPB             Stable       Stable
##  9 CZ_PROM.EASTSIDE   Increasing   Increasing
## 10 CZ_PROM.WESTSIDE   Increasing   Increasing
## # ... with 13 more rows
```

5.5 Step 5: Calculate the Optimum Target Range for the SMU

The last step is to calculate the optimum target range for the SMU. This is a combination of Steps 5 and 6 in the Draft HS report. Similar to Step 4 (Section 5.4), we require multiplier data to exist to calculate the appropriate ranges. Once again, these are included in the `harvestStrategy` package as per Table 5 in the Draft HS report. Code Chunk 5.8 shows how to load these multipliers.

R Chunk 5.8.

```
data(range_rules)
range_rules

## # A tibble: 6 x 4
##   CCS   Final_Category Lower Upper
##   <chr> <chr>          <dbl> <dbl>
## 1 CCR1  Increasing     1.00  1.15
## 2 CCR1  Stable         0.950 1.05
## 3 CCR1  Decreasing     0.850 0.950
## 4 CCR2  Increasing     0.950 1.05
## 5 CCR2  Stable         0.850 0.950
## 6 CCR2  Decreasing     0.750 0.850
```

Having loaded the multiplier rules, we can use all previous steps to calculate the new optimum target ranges to be applied in the next season (which will be 2017 based on our example). Code Chunk 5.9 demonstrates this, and stores the output in a new dataset, hs_2017, which can be used for reporting.

R Chunk 5.9.

```
hs_2017 <- hs_2016 %>%
    mutate(
        ranges = purrr::pmap(list(OptimumTarget, CCS, final_cat),
                             catchControlRange,
                             range_rules = range_rules)
    ) %>%
    tidyr::unnest()
hs_2017 %>%
    select(SMU, Year, CCS, final_cat, OptimumTarget, OT_Lower, OT_Upper)

## # A tibble: 23 x 7
##    SMU                 Year CCS   final_cat OptimumTarget OT_Lower OT_Upper
##    <chr>              <dbl> <chr> <chr>             <dbl>    <dbl>    <dbl>
##  1 CZ_BACK.BEACHES     2016 CCR1  Increasi~         66.2     66.2     76.1
##  2 CZ_CAPE.LIPTRAP     2016 CCR1  Decreasi~         12.5     10.6     11.9
##  3 CZ_CAPE.OTWAY       2016 CCR1  Increasi~         61.5     61.5     70.7
##  4 CZ_CLIFFY.GROUP     2016 CCR1  Decreasi~          5.00     4.25     4.75
##  5 CZ_FLINDERS         2016 CCR1  Increasi~         30.0     30.0     34.5
##  6 CZ_KILCUNDA         2016 CCR1  Increasi~         14.1     14.1     16.2
##  7 CZ_PHILLIP.ISLAND  2016 CCR1  Increasi~         41.7     41.7     48.0
##  8 CZ_PPB              2016 CCR1  Stable             0        0        0
##  9 CZ_PROM.EASTSIDE    2016 CCR1  Increasi~          5.00     5.00     5.75
## 10 CZ_PROM.WESTSIDE    2016 CCR1  Increasi~         20.0     20.0     23.0
## # ... with 13 more rows
```