Vignette: Getting Started with COYU R package

Introduction

The combined-over-years uniformity (COYU) criterion is used to assess uniformity for measured, quantitative characteristics in DUS trials. Uniformity is one of the criteria to be assessed before granting protection of a new plant variety under the UPOV system (see www.upov.int).

COYU is used for measured characteristics, assessed over trials over two or three years. The between-plant variability for a new variety is compared to existing varieties planted in the same trials. To take into account any relationship between level of expression and variability, a spline-based adjustment is used. An older version of this criterion used moving-averages rather than splines. The method is described in UPOV document TGP/8 (version 5 and later) "Trial Design and Techniques Used in the Examination of Distinctness, Uniformity and Stability", which can be found at www.upov.int/tgp.

This package allows implementation of COYU. It requires variety means in each trial for two or three years, along with pooled within-plot standard deviations. It is freely available from the authors at Biomathematics & Statistics Scotland (BioSS), via GitHub (see below).

Installation

The R package has been written for R version 3.0.2. If it is running in an older version, it may not run. Successful application of the package will require a reasonable level of R experience.

Firstly, the package '`devtools`' should be installed from CRAN. Then run the following code

```
library(devtools)
install_github("BiomathematicsAndStatisticsScotland/coyus
/package/coyu")
```

The second command accesses the current version of the COYU from GitHub.

The package does require the installation of another package `lme4` and its dependencies, which are available from CRAN. So please ensure that `lme4` is installed prior to installing the COYU package. We may investigate automating this at a later stage.

For every session, the R package should be called using:

```
library(coyu)
```

There is help for each of the key functions along with a vignette "GettingStarted". Note that the vignettes for the package may listed using the command `vignettes(package='coyu')`.

COYU package functions

The coyu package contains a number of functions designed to help produce the required format for input and to structure output as well as running the COYU with spline analysis. There are also a number of background functions that you will not need to access directly. Here is a list of the key functions along with their use:

`COYU_all_results`	Core-function: carries the COYU analysis with spline adjustment and generates a results set from trial data, parameters and probability set objects.
`COYU_data_skeleton`	Input data manipulation: creates a "skeleton" data-frame of the right size and type.
`COYU_parameters`	Input data manipulation: creates a parameter object with the correct type. Usually called by `COYU_parameters_from_df`.
`COYU_parameters_from_df`	Input data manipulation: Infer parameters from the trial data data-frame. You just need to supply the candidate AFP numbers.
`COYU_probability_set`	Input data manipulation: generates a probability set object from the lists of probability values for 2 and 3 year rejection decisions along with 2 year acceptance decisions.
`COYU_sanity_check`	Input data manipulation: checks consistency of input data
`COYU_print_results`	Output formatting: prints summary results following a run of `COYU_all_results`
`COYU_results_as_dataframe`	Output formatting: converts output from `COYU_all_results` to a convenient data-frame format. This can then be written to a file, e.g. comma-separated value format using `write.csv` (csv files can easily be read by Excel).
`COYU_plot_results`	Output formatting: produces plots for each character and year of the spline curves fitted by `COYU_all_results`

Specification of input

In the first instance, we recommend that you use the function `COYU_all_results` to run the COYU analyses using spline adjustment. This function requires three arguments to define the input for the analysis and it is suggested that you use the constructor functions to create them:

1) `trial_data`: trial data in a data –frame with columns `year`, AFP, `variety`, one column for each character (UP prefix followed by numeric identifier of character) containing mean values, and one column for each character (sUP prefix followed by numeric identifier of character) containing within-plot standard deviations (mean over plots). `Year` is numeric e.g. 2001. AFP denotes the numeric unique identifier for the varieties and must be a factor. `Variety` is an optional text label for each variety. Missing values are denoted NA as usual in R. This object should have the S3 class "COYUs9TrialData"

2) `coyu_parameters`: a list defines which elements of the trial data are to be included in the analysis. It is a list with elements `candidates` (a numeric vector specifying AFP numbers of candidates), `references` (a numeric vector specifying AFP numbers of reference varieties), `characters` (a numeric vector specifying numbers of characters) and `num_trial_years` (a number stating the number of years). This object should have the S3 class "COYUs9Parameters".

3) `probability_sets`: a two-dimensional array with each row defining a set of probability values for decisions. The ability to define more than one set may be useful in this practical exercise. There are three columns: `3_year_reject` (probability value for rejection after three years), `2_year_reject` (probability value for rejection after two years) and `2_year_accept` (probability value for acceptance after two years – this will be a higher value than `2_year_reject`). Note that, with a 2 year DUS test, `3_year_reject` and `2_year_accept` are redundant but the software still requires values for these. This object should have the S3 class "COYUs9ProbabilitySet".

To facilitate setting up of these files we have included utility functions `COYU_data_skeleton`, `COYU_parameters`, `COYU_parameters_from_df` and `COYU_probability_set`. See below for an example of their use. Further information on the data format can be found by using the command `help(COYU_dataset_format,package="coyu")`.

Output

Running function `COYU_all_results` produces a complex output data set. This consists of a list with one element for each probability set. To facilitate presentation of results, there are three functions: `COYU_print_results`, `COYU_results_as_dataframe` and `plotResults`.

The function `COYU_print_results` produces accessible tables of results – however at the moment these are just summaries. For this exercise the function `COYU_results_as_dataframe` will be more useful. This collates some of the more useful results in a data-frame with one row for each candidate-by-character combination. The function `plotResults` produces plots for each character and year combination showing the spline curve fits. We ask that you review these to see that the method is working sensibly.

Example of use

Example 1

The first example takes a pre-existing data-set, available within the package, and shows how to run the COYU analysis. In the code below, `pathname` is a suitable working directory. The data command accesses the example data-set - you may wish to examine this (more information in "Getting Started" vignette).

```
setwd("pathname")
library(coyu)
data(test_2_year,package="coyu")
     # from example data set in package
results1<-COYU_all_results(test_2_year$trial_data,
                 test_2_year$coyu_parameters,
                 test_2_year$probability_sets)[[1]]
     # note [[1]] selects the results for the first probability set
COYU_print_results(results1,            test_2_year$coyu_parameters,
test_2_year$character_key, test_2_year$probability_set[1,])
     # note test_2_year$probability_set[1,] gives the probabilities
     for this set
write.csv(COYU_results_as_dataframe(results1,      "2_year_reject"),
"tester.csv")
COYU_plot_results(results1,             character_key              =
test_2_year$character_key, plot_file="MyPlots.pdf")
     # results sent to a pdf file.
```

Example 2

The second example illustrates how to use the input utility function to create input files. You may wish to examine the objects created at each stage,

```
setwd("pathname")

library(coyu)

years=c(2011,2012)

characters=c(1,2,3)

varieties=c(10,11,12)

COYU_data_skeleton(years, characters, varieties)

fake_mean_data<-rbind(matrix(3,3,3), matrix(8,3,3))

fake_stddev_data<-rbind(matrix(0.1,3,3), matrix(0.8,3,3))

trial_data<-COYU_data_skeleton(years,     characters,     varieties,
mean_data = fake_mean_data, stddev_data = fake_stddev_data)

COYU_parameters_from_df(trial_data,c(10))
```

```
y=COYU_probability_set(reject_3_year = c(0.05,0.02,0.01), reject_2_year = c(0.05,0.02,0.01), accept_2_year = c(0.1,0.05,0.02))
```

```
COYU_sanity_check(trial_data = trial_data, coyu_parameters = COYU_parameters_from_df(trial_data,c(10)))
```

Starting from data in DUSTNT format

If data is available in DUSTNT format, then there are tools to assist in transforming the data ready for use with the coyu package. Note DUSTNT is software to assist with statistical analysis for DUS testing. It is supplied by AFBI in Northern Ireland.

It is first necessary to download further code from the GitHub repository. The file ‘`IOroutines.R`’ can be found in the `dust_stub` subdirectory at at https://github.com/BiomathematicsAndStatisticsScotland/coyus.

A DUSTNT file is required to link the information together for the run. This is the input file, called COYU9.DAT by default, as described in the instructions for the DUSTNT COYU9 and COYUS9 modules in the DUSNT manual. This links to a UX format file. All directory references need to be valid. The function `readCoyu9File` is accessed from `IORoutines.R`.

The code below shows how it can be used.

```
setwd("pathname")
library(coyu)
source("IORoutines.R")
input_file="C:/TEMP/COYUS9.DAT" # insert own example
file.exists(input_file)
setwd(dirname(input_file))
data1<-readCoyu9File(basename(input_file))
results1<-COYU_all_results(data1$trial_data,

                           data1$coyu_parameters,

                           data1$probability_sets)[[1]]

COYU_print_results(results1,

                   data1$coyu_parameters,

                   data1$character_key[1,],

                   data1$probability_sets[1,])
```