Tutorial to ORNAMENTAL tool

1 Introduction

Wellcome to ORNAMENTAL. The ORNAMENTAL program is written in C++ and runs on Linux in command-line mode. The results of the program are output to text and image terminals.

2 Program installation

2.1 Hardware requirements

Hardware requirements are low, and the program can run on any PC. For example, the program was developed and tested on an Intel Core i7-3770K CPU @ 3.50 GHz PC with 32G of memory. Note that, execution of any command was instantaneous (<1s), memory consumption did not exceed 0.1%.

2.2 Requirements for installed programs

The program is developed in Ubuntu 16.04, without using any features of this operating system. It is expected to run on any other unix-like system, or in an environment that mimics it.

The following packages and libraries must be installed on the system:

- c++ compiler that supports the standard C++17 (g++-7 –version >= 7.5.0)
- boost-1.74 (>= boost-1.72)
- gsl (>= 2.1)
- ncurses (>= libncurses5)
- gnuplot (>= 4.6)
- any pdf viewer, like evince and/or okular
- any program to convert svg files to pdf files, for example, inkscape

Although any program can be used to view pdf (for example, libreoffice, firefox, etc.), but in reality there is a limitation related to usability. Since ORNAMENTAL generates a large number of plots, it is essential that the viewer automatically reloads files that have been modified on disk.

2.3 Compiling and running the program

2.3.1 Compiling from the command line

After downloading and unpacking the file ornamental.zip, go to the directory ornamental: *cd ornamental*

Compile with the following commands:

```
g++-7 -c -O2 -std=c++17 -I./include -o main.o main.cpp
g++-7 -o orn main.o -L /usr/local/lib -lgsl -lgslcblas
      -lcurses
```

Run the program with the command:

./orn or

./orn <file>,

where <file> is the name of the text file containing the configuration and pre-execution commands. The sample file is named *ornamental.init*.

2.3.2 Compilation in the C++ development framework

You can compile and run it in any C++ development framework. Users who choose this way do not need a description of how to do it.

Note. When using Qt Creator, you must insert in the *.pro file the next line:

```
CONFIG += c++1z # C++17.
```

2.3.3 Input data

The program processes *.csv files (where data is separated by the comma symbol), as well as *.orn files. The *.csv format follows the data description in table 1, and the *.orn format follows table 2 (see Fomin ES and Fomin ai. A nonparametric model for analysis of flowering patterns of herbaceous multiflowered monocarpic shoots. Submitted to Bulletin of Mathematical Biology, 2020). The use of *.csv files is more natural for practicing plant biologists, but leads to a high level of redundancy in detailed data files; *. orn files do not have this disadvantage.

3 Basic commands

3.1 Commands of ORNAMENTAL

Any command of ORNAMENTAL consists of a command name followed by a list of parameters with values. Parameters differ by name, and the order of such parameters is arbitrary. Parameters and values are separated by a space character.

$$cmd\ [[param_1]\ value_1]\ ..\ [[param_N]\ value_N]$$

where *cmd* is the full or abbreviated command name, *param* is the full or abbreviated parameter name, and *value* is the parameter value. Abbreviations do not conflict with each other. Optional parameter names and values are shown in square brackets. The command may have no parameters or have unnamed parameters. Unnamed parameters always have a fixed order.

3.2 Linux commands

You can use both ORNAMENTAL and linux commands on the program's command line. If the command is not recognized as an ORNAMENTAL command, it is run as a linux command. The purpose of using linux commands is to get information about applications installed on the system when configuring ORNAMENTAL. The linux command format has not been modified.

ATTENTION! Careless use of linux commands can cause permanent damage to the user's files, since they do not provide warnings. It is also possible that the app may seem to "hang". In this case, press Ctrl-C. Some linux commands cannot be run in the ORNAMENTAL terminal. These are the linux commands that cannot be run correctly with the command *system()*.

3.3 List of commands

The following characters are used in the list of commands:

- '|' : the OR operator allows you to select one or another option
- «[ ]» : optional parameters are enclosed in square brackets
- « < > » : variable names of parameters and their values are written in angle brackets
- fixed values are written without parentheses

The '! ' symbol is used, which asks the user to pay attention to the following comment.

3.3.1 help|h

The command lists all the ORNAMENTAL commands used. There are no parameters.

3.3.2 environment|env [<var> <value>]

The command without parameters gives a list of all environment variables. A single-parameter command lets you change the value of a specified environment variable.

The following environment variables are used:

- LOCALE (abbrev. L|l) is the localization
- WORK_DIR (abbrev. W|w) is the working directory (user data files)
- OUTPUT_DIR (abbrev. O|o) is the directory to output the results
- PDF_VIEW (abbrev. PDF|pdf|P|p) is the tool to view pdf files
- SVG_TO_PDF (abbrev. SVG|svg|S|s) is the tool to convert svg files to pdf files

Examples

- env LOCALE ru_RU-UTF-8
  # installing the russian locale

- env LOCALE C
  # installing the "C"locale (for advanced users)
- env WORK_DIR
  # the working directory /home/<user>
- env WORK_DIR tmp/dir
  # the working directory /home/<user>/tmp/dir (! no slash before tmp/)
- env WORK_DIR /tmp/dir1
  # the working directory /tmp/dir (! slash before tmp/)
- env PDF_VIEW evince %s.pdf
  # the evince as pdf viewer (! postfix .pdf after %s)
- env SVG_TO_PDF inkscape –export-pdf=%s.pdf %s.svg
  # the inkscape to convert files *.svg -> *.pdf (! the order of tokens %s.pdif %s.svg is strictly fixed)

Attention! There may be problems with some locales. If after setting the locale, there are distortions when drawing graphs, try setting the locale C.

3.3.3 window|w

Open the graphical terminal to view the model.

3.3.4 fload|f< <filename> [<fileID>]

The command loads a data file in the *.csv or *.orn format for processing. After loading, the dataset gets the name <fileID>, if specified. The <fileID> name can be any string or numeric identifier. This <fileID> is used to access this data set. If <fileID> is omitted, the data set is named *noname*. The loaded data set is assigned the status R (for reading), and if the downloaded file is the only one, then it is assigned the status W (for writing). You can upload any number of files or the same file multiple times. All uploaded datasets must have different <fileID>. When <fileID> is duplicated, previously loaded dataset disappear.

3.3.5 fadd|f+ <fileID>

The command creates a new empty dataset with the name <file ID> and sets it's status to W (for writing). An empty data set can be used to accumulate sprout samples generated by models (see the command *models*).

3.3.6 files|f

The command lists all loaded/created files with their statuses.

3.3.7 files|f <fileID> R|W|RW

This command allows you to change the status of the specified file.

Any number of datasets can be created or loaded, but only one dataset can have the R status. This particular dataset is used for reading data and generating models. Only one dataset can also have the W status. It is this dataset that the

samples generated by models are written to. All other datasets have an undefined status. In fact, statuses show which datasets are active. Setting a status for a certain dataset automatically removes this status from another set.

Note. There is also the status '∧' which means that the data in the file has been improved by using the MLM method (the command *festimate*).

3.3.8 fsave|f> <fileID> <filename>

Save the <fileID> dataset to the <filename> file. The file is placed in OUTPUT_DIR. (!) You don't need to specify the path to the file.

3.3.9 fdel|f- <fileID>

Destroys the <fileID> dataset. (!) No warnings are given.

3.3.10 festimate|f∧ MLM|random <fileID>

To improve input data <fileID>. Due to the high expenditure of time and effort, observations of the development of shoots may not be performed daily, that is, they may have random or fixed (observations every two or three days) omissions. As a result, there is uncertainty in determining the bounds of phases of development of all the flowers of shoots. This command allows you to improve the available data.

The command requires you to specify the data processing method.

- *MLM* is the maximum likelihood method. There is assumed that the data is distributed according to the normal law.
- *random* is the random selection of a values in the known uncertainty intervals.

After refining the data using the *MLM* method, the file gets the status '∧', after using the *random* method the '∧' status is removed.

3.3.11 madd|m+ <fileID:sproutID> [<modelID>]

Generating a model with the specified name <modelID> from the <fileID>. For generation, an <sproutID> is used as a reference sample. After the model is generated, it is assigned the 'R' status (see the models <modelID>command).

A model is a set of different relations between structural/dynamic parameters of a shoot that allows to completely restore the structure and dynamics of the shoot development with statistical accuracy in the absence of initial data, that is, to solve the inverse—from the model to the data—problem. During generation, not only the model—the minimum required set of relations—is created, but also a number of additional dependencies for a more complete view of the model.

When creating a model, the development data for the <sproutID> is used. In addition, the structural data of all the shoots in the <fileID> is collected and the corresponding numerical distributions are constructed allowing to generate model shoots with different structures.

3.3.12 mdel|m- <modelID>

Delete the specified model.

3.3.13 models|m

View the full list of all models.

3.3.14 models|m <modelID>

Set the 'R' status for the specified model. This status means that all plots will be generated for this model.

3.3.15 plots|p

Show a list of all <plotID>s. <plotID> can be used to view, destroy, or regenerate the plot.

3.3.16 plots|p <plotID>

Show a plot with the <plotID> for the active model. (!) All curves are generated with the minimum possible smoothing window, the width of which is determined by the number of points in the data.

3.3.17 padd|p+ <plotID> [w <value>]

ADD a plot to the diagram with the specified relative width of the data smoothing window. (!) The width of the window is relative. (!) It is not possible to set a very small width, it is always limited by the number of data. This command is auxiliary and allows you to "play around"with different ways for generating plots. Using the command without the width parameter generates a plot with the default width of the smoothing window. To clear the diagram, use the command *pdel* (see below).

3.3.18 pdel|p- [<plotID>]

Delete the plot with the specified <plotID>, or the last drawn plot, if <plotID> is missing. (!) After deleting the plot, the model may become incomplete and not allow the generation of model shoots. Therefore, you need to restore the deleted plot using the command *plot <plot ID>*.

3.3.19 psave|p> <filename>

Save the current plot to OUTPUT_DIR with the specified <filename>. (!) The <filename> is specified without a path.

3.3.20 test|t <plotID> [n]

Run statistical tests for the specified <plotID>.

- If the plot is a distribution, then testing for the normality of the distribution is performed and some statistics are printed. Testing uses the specified number of n points. If the n parameter is omitted, the number of points n is equal to the number of points in the input data.

- If the plot is a regression, then the Pearson correlation coefficient r and its p-value, the coefficient of determination $R2$ are output, and the residuals that are not described by the regression are tested for autocorrelations and the residuals are tested for normality. In this case, the $[n]$ parameter is ignored and the input data is used for testing.

3.3.21 scheme <modelID>|<fileID:sproutID> <date|day>

Display the shoot structure for the <modelID> or for <sproutID> in the <fileID> and the state of development of sprout flowers on the specified date or on the relative date from the beginning of flowering. The date is given in the DAY-MONTH format, for example 4-Jul. The relative date is set as an integer.

Examples

- scheme <modelID> 4-Jul
  # <modelID> on 4-Jul.
- scheme <modelID> 0
  # <modelID> on the date of the flowering start
- scheme <modelID> -2
  # <modelID> on the date of 2 days before the flowering start

3.3.22 video <modelID>|<fileID:sproutID> <date1:date2|day1:day2>

Display the shoot flower development sequences for the <modelID> or sproutID in the <fileID> in the specified date range or days from the beginning of flowering. Dates are set in the DAY-MONTH format, relative dates are set as integers.

3.3.23 generate|g <modelID> N [<n1>:<n2>]

Generate N model shoots using the <modelID>. The result is written to a dataset with the status W. The parameters <n1>—number of flowers of the 1st order—and <n2>—number of flowers of the second order—set restrictions on the structure of the generated shoots. A sprout is generated with the number of flowers of the corresponding orders as close as possible to the specified values. Specifying zero values means that the numbers must be taken the same as in the sprout that was used to create the model. Specifying values as *:* means that there are no restrictions. In this case, these values are generated from the corresponding distributions and regressions.

3.3.24 quit|exit

Exit the program.

4 Scripts

The application is launched by the command:

```
ornamental [<filename>]
```

where the optional <filename> parameter contains the name of the script file.

At startup, the application checks for the <filename> in the command line. When it is detected, the application first executes all the commands written in <filename>, and then switches to interactive mode. If the script contains the *exit/quit* command, the program terminates without switching to interactive mode.

Script files can be created in any text editor, or by editing the *.usr file, which records all the commands that the user entered during execution. *.usr file is located in the /home/<user>/.ornamental directory, which is created automatically when the app is launched. Also in this directory there are *.log files that log the received commands and their results.

4.1 Configuration script

Configuration script allows you to avoid configuring the app every time you launch it. When creating the script manually, you need to enter a sequence of configuration commands in it.

Sample configuration file:

```
env LOCALE ru_RU-UTF-8 \# set locale
env PDF_VIEW evince %s.pdf \# pdf viewer
env SVG_TO_PDF inkscape --export-pdf=%s.pdf %s.svg
\# svg->pdf convertion
env WORK_DIR ~ \# set workdir = /home/<user>
env OUTPUT_DIR ~/tmp/orn \# set outputdir = ~/tmp/orn
```

Useful tips:

1. when working with ORNAMENTAL, many temporary pdf files are generated and modified. To view them, use applications that automatically reload files when they change on disk, such as evince and okular. Do not use applications such as xpdf, qpdfview, firefox, loffice, and a number of others, because they need to manually reload files.

2. When setting the PDF_VIEW environment variable, don't forget to specify the regular expression '%s.pdf' to allow file name substitution. ! The .pdf postfix is required.

3. the command 'apropos viewer' lets you find out if there are any viewer programs on your system (see https://www.geeksforgeeks.org/apropos-command-in-linux-with-examples/). It can be run in the ORNAMENTAL terminal. The command

   ```
   "dpkg --list | grep-i '[ (]pdf[ )].*viewer'"
   ```

   can also be used for search.

4. when working with ORNAMENTAL, inflorescence diagram files are generated in svg format. To view them in the same graphical pdf terminal, inflorescence diagram files are converted to pdf format. To convert, specify the SVG_TO_PDF environment variable. (!) When setting the SVG_TO_PDF environment variable, don't forget to specify the regular expressions '%s.pdf' and '%s.svg' to allow file names to be substituted (their order is fixed, first .pdf and then .svg).

4.2 Creating a configuration script from the app

Launch the app and run commands (for example, 4.1) in it. Exit the app. Find the file /home/<user>/.ornamental/terminal.usr, edit it, and save it under a different name. To enable interactive mode after the script finishes, delete the *exit*/*quit* command from the file.

Editing *.usr files is the easiest and most reliable way to create working scripts, since the application responds to commands entered incorrectly.

4.3 A work script. Example

The example uses abbreviated command names, which is more convenient when working.

```
# setting environment variables and opening a graphical terminal
env LOCALE ru\_RU-UTF-8
env PDF\_VIEW evince \%s.pdf
env SVG\_TO\_PDF inkscape --export-pdf=\%s.pdf \%s.svg
env WORK\_DIR ~
env OUTPUT\_DIR ~/tmp/orn
window # open window
h # help
#
# loading and refining data, building a model of the 3rd sprout
f+ C.sarmatica.csv S
f^ MLM S
m+ S:3 m3
#
# viewing plot IDs, drawing the plot 9 and saving it to disk
p
p 9
p> flowering\_curves1.pdf
#
# creating empty file, generaing 100 model sprouts
# and inserting their to 'S'(C.sarmatica)
f+ model_sprouts
generate m3 100
#
# viewing the file for the numbers of the newly inserted sprouts,
# and selecting a sprout (for example, 101);
# building a model based on this new model sprout
# drawing the plot 9 and saving it to disk
f
m+ S:101 m101
p 9
p> flowering\_curves1.pdf

# viewing the structure of the models on the day
# of flowering start and saving them
scheme m1 0
p> scheme1.pdf
```

```
scheme m101 0
p> scheme101.pdf

exit
```

5 Detected shortcomings

- called third-party programs may attempt to output warnings to the terminal (such as gnuplot). As a result, this causes the screen to flicker.
- In some locales, for example, in empty locale, there may be artifacts when plotting graphs and inflorescence diagrams. The locales C, en_US.UTF-8, and ru_RU-UTF-8 work without such artifacts.
- erroneous source data can lead to artifacts on the diagrams. For example, plot constructed for the C.saramtica.csv file (sprout:2) have artifacts associated with missing data for flowers from 1 to 5. This defect is corrected after executing the command *festimate*, and rebuilding the model.

Please report any bugs you find to my email address. I would also be grateful for proposals and suggestions for improving the functionality of the program.

Yours sinceraly, Eduard Fomin.
fomined@gmail.com