tvst

Data Science

PyVisualFields: A Python Package for Visual Field Analysis

Mohammad Eslami[1], Saber Kazeminasab[1], Vishal Sharma[1], Yangjiani Li[1], Mojtaba Fazli[1], Mengyu Wang[1], Nazlee Zebardast[2], and Tobias Elze[1]

[1] Harvard Ophthalmology AI Lab, Schepens Eye Research Institute of Massachusetts Eye and Ear, Harvard Medical School, Boston, MA, USA
[2] Massachusetts Eye and Ear Infirmary, Harvard Medical School, Boston, MA, USA

Correspondence: Mohammad Eslami, Harvard Ophthalmology AI Lab, Schepens Eye Research Institute of Massachusetts Eye and Ear, Harvard Medical School, 20 Staniford Street, Boston, MA 02114, USA.
e-mail: mohammad_eslami@meei.harvard.edu

Received: July 21, 2022
Accepted: November 24, 2022
Published: February 6, 2023

Keywords: software package; visual field; python; R; progression

Citation: Eslami M, Kazeminasab S, Sharma V, Li Y, Fazli M, Wang M, Zebardast N, Elze T. PyVisualFields: A python package for visual field analysis. Transl Vis Sci Technol. 2023;12(2):6, https://doi.org/10.1167/tvst.12.2.6

Purpose: Artificial intelligence (AI) methods are changing all areas of research and have a variety of capabilities of analysis in ophthalmology, specifically in visual fields (VFs) to detect or predict vision loss progression. Whereas most of the AI algorithms are implemented in Python language, which offers numerous open-source functions and algorithms, the majority of algorithms in VF analysis are offered in the R language. This paper introduces *PyVisualFields*, a developed package to address this gap and make available VF analysis in the Python language.

Methods: For the first version, the R libraries for VF analysis provided by *vfprogression* and *visualFields* packages are analyzed to define the overlaps and distinct functions. Then, we defined and translated this functionality into Python with the help of the wrapper library *rpy2*. Besides maintaining, the subsequent versions' milestones are established, and the third version will be R-independent.

Results: The developed Python package is available as open-source software via the GitHub repository and is ready to be installed from *PyPI*. Several *Jupyter* notebooks are prepared to demonstrate and describe the capabilities of the *PyVisualFields* package in the categories of data presentation, normalization and deviation analysis, plotting, scoring, and progression analysis.

Conclusions: We developed a Python package and demonstrated its functionality for VF analysis and facilitating ophthalmic research in VF statistical analysis, illustration, and progression prediction.

Translational Relevance: Using this software package, researchers working on VF analysis can more quickly create algorithms for clinical applications using cutting-edge AI techniques.

Introduction

Standard automated perimetry visual field (VF) examination is one of the most widely used quantitative methods to evaluate the visual functions of the eyes, to detect and monitor their progress.[1] Whereas cataract and myopia are more common than glaucoma, glaucoma is the most common cause of irreversible blindness in the world that changes the structure of the optic nerve head (ONH) and causes retinal ganglion cell (RGC) death[2] which affects the visual function, showing specific defects in VF tests.[3] It is crucial to detect early glaucoma to prevent irreversible vision loss.

Artificial intelligence (AI) and deep learning (DL) is an emerging area of science that is revolutionizing the development of almost every field that it touches. The AI algorithms have made the analysis of the data much simpler, and it has also brought new tools for further analysis and understanding of the features behind data. Ophthalmology is one of the areas in which AI can help to enhance better capability in the detection, prediction, and progression prediction of ocular diseases like glaucoma. Most AI algorithms are offered in Python, an open-source programming language. It is expanding extremely fast with the easy contribution of software developers around the world. However, the functions to analyze the VF are mainly offered in the R language. There are some wrapping methodologies for using R

Table. The Functions Provided in *PyVisualFields* Library

	Function	Source	Description	
Pre-defined DATA	data_vfpwgRetest24d2()	*visualFields*	Exemplary VF data	
	data_vfctrSunyiu24d2()		Several pre-saved VF data are available to show the data structure and required columns in DataFrames. Some are a series data, and some are just cross-sectional data.	
	data_vfpwgSunyiu24d2()			
	data_vfctrSunyiu10d2()			
	data_vfctrIowaPC26()			
	data_vfctrIowaPeri()			
	data_vfseries()	*vfprogression*		
	data_vfi()			
	data_cigts()			
	data_plrnouri2012()			
	data_schell2014()			
Progression Analysis	get_score_AGIS()	*vfprogression*	Compute and get AGIS and CIGTS visual field defect scores[11]	
	get_score_CIGTS()			
	progression_cigts()	*vfprogression*	Progression analysis based on different criteria. The input data is a series of VFs with at least 5 tests. The returned value would be stable, worsening or improving for each eye (left/rights).	
	progression_plrnouri2012()			
	progression_vfi()			
	progression_schell2014()			
	progression_agis()			
	gettdp()			
	glr()	*visualFields*	Linear regression	
	plr()		pointwise linear regression (PLR)	
	poplr()		PoPLR regression analysis (permutation based)	
Plotting	plotValues()	*vfprogression*	Plot/save sensitivity or deviation values	PNG, PDF and SVG can be used as the saving format.
	plotProbabilities()		Plot/save deviation probability maps (e.g. tdp, pdp)	
	vfplot()	*visualFields*	Input is VF sensitivity and can plot and save values and probabilities based on its own design. We also prepared some aliases.	
	Vfplotsparklines()		Plot and save sparklines of a series data. We also prepared some aliases.	
	Vfsfa()		Generates one-page report of single field analyses as a pdf file.	
Deviation Analysis	getnv()	*visualFields*	Get current normative setting deployed	These functions are helpful to compute deviation values based on some predefined normalization settings. It is also possible to train/generate a new normative setting based on a new set of VF data.
	get_info_normvals()	*visualFields*	Get all available predefined normalization settings	
	getallvalues()	*visualFields*	Compute all td, pd, pdp, tdp, gl, gh, glp based on the current normative setting. Input is the VF sensitivity values.	
	gettd()	*visualFields*	alias for *getallvalues* only to compute td	
	getgl()	*visualFields*	alias for *getallvalues* only to compute gl	
	getpd()	*visualFields*	alias for *getallvalues* only to compute pd	
	gettdp()	*visualFields*	alias for *getallvalues* only to compute tdp	
	getgh()	*visualFields*	alias for *getallvalues* only to compute gh	
	getpdp()	*visualFields*	alias for *getallvalues* only to compute pdp	
	getglp()	*visualFields*	alias for *getallvalues* only to compute glp	
	setnv()	*visualFields*	change and set normalization setting to a predefined or new computed setting	
	nvgenerate()	*visualFields*	Compute new normalization values based on new data and generate a normalization setting	

scripts and libraries in Python (e.g. pyRserve, rpy2 [comparable to R package *reticulate* for calling Python scripts/commands in R]), however, changing the scripts to make use of them takes time, especially when transferring data and variables between R and Python and addressing several data-structures is required. Hence, it is useful to make available functions to analyze the VF in python to have a broader audience of researchers and help to further develop VF analysis approaches.

In this paper, we offer a python package with more than 50 functions to make VF analysis available for Python developers. The first version is based on the previously developed collection of tools in R by Elze et al. (so-called: *vfprogression*[4]) and Marin-Franch et al. (so-called: *visualFields*).[5,6] We present the functions in four categories including predefined data, VF demonstration, scoring, and progression analysis, as well as normalization and deviation analysis.

This developed library, named *PyVisualFields*, and used in our recent research to investigate the clinical applicability of deep learning models for predicting the ocular VFs while taking into account

A

```
from PyVisualFields import visualFields
import pandas

df_VFs_py = visualFields.data_vfpwgRetest24d2()
df_VFs_py.head()
```

	id	eye	date	time	age	type	fpr	fnr	fl	duration	...	l45	l46	l47	l48	l49	l50	l51	l52	l53	l54
1	1	OD	2008-08-13	00:00:00	53	pwg	0	0.0	0.00	00:00:00	...	26	29	26	27	28	30	25	25	27	31
2	1	OD	2008-08-20	00:00:00	53	pwg	0	0.0	0.06	00:00:00	...	28	28	29	29	30	31	26	25	28	31
3	1	OD	2008-08-27	00:00:00	53	pwg	0	0.0	0.00	00:00:00	...	26	27	28	25	23	28	24	26	26	25
4	1	OD	2008-09-03	00:00:00	53	pwg	0	0.0	0.06	00:00:00	...	26	27	26	25	25	27	25	24	26	29
5	1	OD	2008-09-10	00:00:00	53	pwg	0	0.0	0.13	00:00:00	...	28	28	27	25	25	27	25	24	25	26

5 rows × 64 columns

B

```
from PyVisualFields import vfprogression
import pandas

df_VFs_py = vfprogression.data_vfseries()
df_VFs_py.head()
```

	eyeid	nvisit	yearsfollowed	distprev	age	righteye	duration	falsenegrate	falseposrate	malfixrate	...	pdp45	pdp46	pdp
1	1.0	1	0.00	NaN	64.98	1	287	0.0	0.00	0.071429	...	1.0	1.00	1
2	1.0	2	0.73	0.73	65.71	1	295	0.0	0.01	0.142857	...	1.0	1.00	1
3	1.0	3	1.77	1.04	66.75	1	291	0.0	0.01	0.062500	...	1.0	1.00	1
4	1.0	4	2.78	1.01	67.76	1	347	0.0	0.02	0.066667	...	1.0	1.00	1
5	1.0	5	4.01	1.23	68.99	1	309	0.0	0.01	0.000000	...	1.0	0.05	0

5 rows × 278 columns

Figure 1. Two exemplary provided VF data.

possible biases and unidentified worsening cases.[7] The library will be extended by adding DL methods for training and predicting the VFs in the second version. It is intended to decrease our R dependency in each release and the third version will be fully independent to R. Version 4 and above will provide user interfaces (UIs) to make the library easy to use for those people who may not be well versed with programming.

Nomenclature and Abbreviations

The abbreviations used in this report are as follows: Artificial intelligence (AI), optic nerve head (ONH), retinal ganglion cell (RGC), visual field (VF), Collaborative Initial Glaucoma Treatment Study (CIGTS), static automated perimetry (SAP), total deviation (TD), total deviation probability (TDP), pattern deviation (PD), pattern deviation map (PDP), Advanced Glaucoma Intervention Study (AGIS), pointwise linear regression (PLR), retinal nerve fiber layer (RFNL), intra-ocular pressure (IOP), open-angle glaucoma (OAG), permutation analyses of pointwise linear regression (PoPLR), global indices (gl), general height (gh), global indices probability (glp), and Normalization Value (NV).

Methods

Open-Source Languages

To develop the first version of the package, we used Python and R which are open-source languages. Then with the wrapper library "*rpy2*," which is a python interface for the R language,[8] we define new functions to translate and call the corresponding R-written functions into python. In addition, we used the open-source dependencies that are required in python for the associated functions to run. A full list of all dependencies is mentioned in the GitHub repository, and they will be installed by default if *pip* installation is being used. In future releases, the third version of the package will be R and rpy2-independent.

A

```python
from PyVisualFields import visualFields

df_VFs_py = visualFields.data_vfpwgRetest24d2()

df_td = visualFields.gettd(df_VFs_py) # get TD values
df_gi = visualFields.getgl(df_VFs_py) # get global indices
df_tdp = visualFields.gettdp(df_td) # get TD probability values
df_pd = visualFields.getpd(df_td) # get PD values
gh = visualFields.getgh(df_td) # get the general height
df_pdp = visualFields.getpdp(df_pd) # get PD probability values
df_gip = visualFields.getglp(df_gi) # get global indices probability values
```

B

```python
from PyVisualFields import visualFields
import pickle

######## caculate new nv values and set it to be used:
df_VFs_py = visualFields.data_vfctrSunyiu24d2()
newNV_r, newNV_py = visualFields.nvgenerate(df_VFs_py, method = "pointwise",
                              name = "our_NV",
                              perimetry = "something",
                              strategy = "something",
                              size = "tmp")
visualFields.setnv(newNV_r)
print('===> current NV setting: ')
currentNV = visualFields.getnv() # check it is set correctly:

#''' notcie: this normalization will not be saved.
#We need to set for each session
#so we need to save and load them seperately, e.g.: '''
newNV_dict = { "newNV_r": newNV_r, "newNV_py": newNV_py }
pickle.dump( newNV_dict, open( "our_NV.pkl", "wb" ) )

loaded_dict = pickle.load( open( "our_NV.pkl", "rb" ) )
newNV_r = loaded_dict['newNV_r']
newNV_py = loaded_dict['newNV_py']
visualFields.setnv(newNV_r) # set it to be used
```

```
===> current NV setting:
$name

[1] "our_NV"
```

Figure 2. (**A**) Compute the deviation values and probabilities. (**B**) Compute, create, and set a new normalization setting based on new data.

Development/Software Iteration Cycles

Because the *PyVisualFields* package version 1.x was determined to wrap the two available R libraries, a waterfall development model was used. Maintenance and subsequent version developments are based on the Agile methodology.

Intended Input Format

In this package, the VF data need to be in Pandas' DataFrame format.[9] Therefore, developers are flexible to have any input file format (e.g. *txt*, *csv*, etc.), but one needs to prepare them as a DataFrame. It should be mentioned that using DataFrame has several required columns (depending on the methods) which are shown by exemplary provided data and functions. Functions based on *vfprogression* library allow 24-2 or 30-2 VF measurements, whereas functions based on *visualFields* library also accept 10-2 measurements.

How External Usability of the Library was Assessed

We examined the functionality of this library by developing test and demo scripts through interactive applications (Jupyter notebook[10] and Spyder[11]) with both Linux and Windows operating systems. Because *PyVisualFields* is a development library, usability tests regarding UI and UX are not applicable.

Software Repository

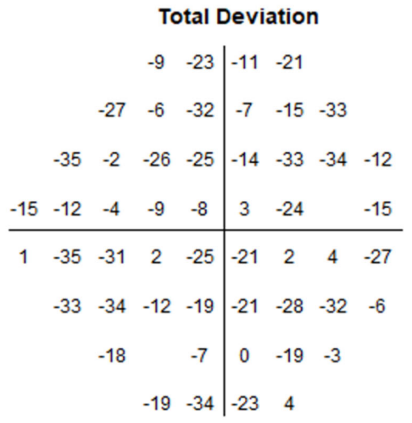
The source code of the *PyVisualFields* package, and its license declaration is available on GitHub: https://github.com/mohaEs/PyVisualField.

This repository also contains demonstration notebooks including examples for introducing functions and arguments.

A

```
from PyVisualFields import vfprogression
import numpy as np

td = np.random.randint(low=-35, high=5, size=(54,))
vfprogression.plotValues(td, title= 'Total Deviation',
                                save=True, filename='td', fmt='png')
```



B

```
from PyVisualFields import vfprogression
import numpy as np

df_VFs_py = vfprogression.data_vfseries()
vf = df_VFs_py.iloc[[5]] # get a one VF
ind_tdp_start= vf.columns.get_loc("tdp1")
ind_tdp_end= vf.columns.get_loc("tdp54")
tdp = vf.iloc[0, ind_tdp_start:ind_tdp_end+1].to_numpy().astype(np.float16())
vfprogression.plotProbabilities(tdp, title= 'Total Deviation Probablity',
                                save=True, filename='tdp', fmt='png')
```

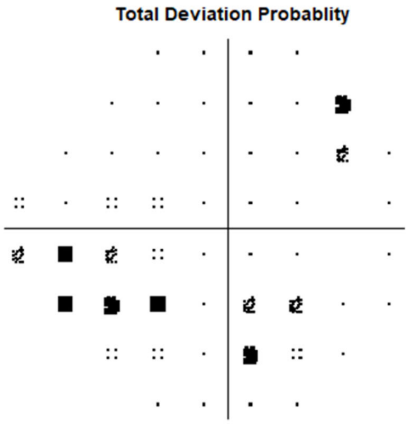


Figure 3. Plotting the values and probabilities using *vfprogression* sub-package.

Software Summary

Open-Source License

The open-source software follows the MIT license.[12]

Comparison with Other Open-Source Tools as a Baseline Comparison

To the best of our knowledge, there is no released Python package for VF analysis in this manner. Another available Python package related to VF is the *hvf-extraction-script* it is used to extract Humphrey visual fields (HVFs) from report files in Dicom and PDF.[13] We will provide a choice in our library to receive input data that is in line with the output of the *hvf-extraction-script*. *PyVF* is a still-under-development VF extraction and simulation program and there is no released version of it on *PyPI*.[14]

As mentioned before, two available open-source R packages (*vfprogression* and *visualField*) wrapped in our library. There are also more R packages related to VF analysis and simulation, such as *binovisual-*

```
from PyVisualFields import visualFields

df_VFs_py = visualFields.data_vfpwgRetest24d2() #df_VFs_py is acquired using vfpwgRetest24d2() functior
vf = df_VFs_py.iloc[[0]] #Lets get the first VF at the first row of df_VFs_py

visualFields.vfplot_s(vf, save=True, filename='file', fmt='png') # alias for vfplot(type='s')
visualFields.vfplot_td(vf, save=True, filename='file', fmt='png') # alias for vfplot(type='td')
visualFields.vfplot_pd(vf, save=True, filename='file', fmt='pdf') # alias for vfplot(type='pd')
visualFields.plotProbColormap(save=True, filename='file', fmt='png') # show colormap of probablies (frc
```

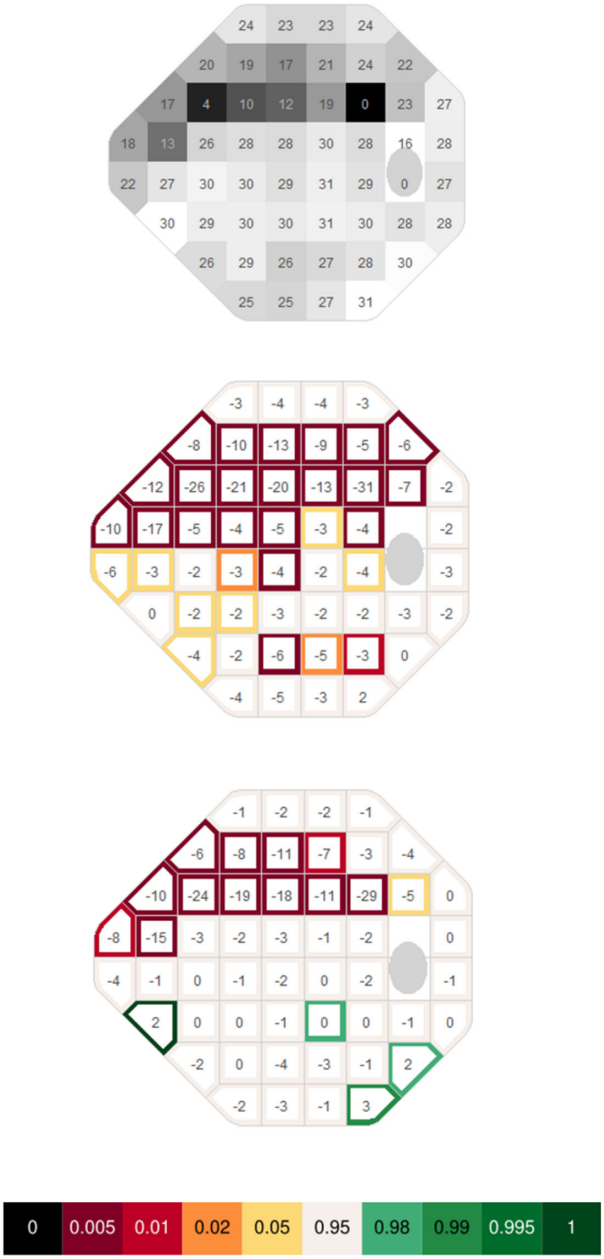


Figure 4. Plotting based on *visualFields* sub-package. Up to bottom are sensitivity, total deviation, pattern deviation, and the colormap of the deviation probabilities.

fields[15,16] and *spCP*[17,18] and adding their functionalities to our package is a milestone for future developments.

External Usability Results

Not applicable.

Figure 5. (**A**) Getting the AGIS and CIGTS defect scores. (**B**) progression analysis of VF series by different criteria.

Submission to Package Registry (pip, gems, CRAN, CPAN, etc.)

The developed *PyVisualFields* package is submitted and available on the Python Package Index (PyPI) website, which is the main repository of software for the Python programming language. PyPI helps people to find and install software and libraries: https://pypi.org/project/PyVisualFields/.

Generated Outputs From the Software Library

The list of functions provided in this library with a short introduction is provided in the Table. The corresponding source R package is also mentioned for finding further information if it is required. This package can be partitioned into four categories as follows. One part is for restoring the pre-saved data to show exemplary VF samples/series and introducing the required data structure and columns of the DataFrame. Figure 1 shows two exemplary VF data. Normalization and deviation analysis is another part of the package which can be used to calculate deviation values and probabilities as well as compute and set new normalization parameters and setting based on a new population. Figure 2 shows two Jupyter cells for these reasons. The third part of the package is organized to do plotting of the test values and probabilities. Plots can also be saved in different file formats,

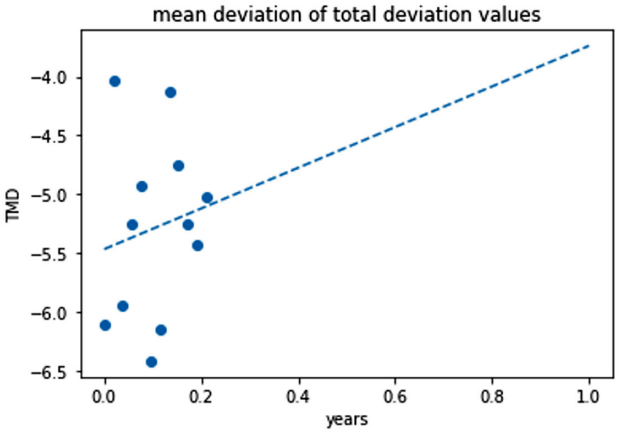


Figure 6. Printout of function *glr(*args)* from *visualFields* package that performs learn regression on the mean deviation of VF tests over 1 year.

including PNG, PDF, and SVG formats. Figures 3 and 4 show two examples of plotting the values and probabilities by each sub-package. VF evaluation and progression analysis are the last part of the packages. Users can get AGIS and CIGTS defect scores for each VF (Fig. 5A).[19] Progression analysis is also provided by five different criteria to determine the worsening, stable, and improving cases. Figure 5B shows three progression analysis with three different criteria. Linear regression analysis is also available in this module as well as pointwise and permutation-based alternatives. Figure 6 shows an example of linear regression analysis.

Discussion

Potential Uses for the Software

The developed library is beneficial for the researchers working in the field of ophthalmology, VF analysis, and AI to develop further algorithms in Python language that offer tons of built-in functions in different open-source libraries.

Limitations of the Software

The library calls the functions from the mentioned R libraries and therefore it is still needed that R core and packages, *vfprogression* and *visualFields,* be installed and available on the machine. For this reason, detailed installation and verification procedures are made available at the repository as well as a video demonstration. It is intended to be R-independent in the third version.

Acknowledgments

Supported by the National Institutes of Health (NIH: R01 EY030575 and P30 EY003790).

Disclosure: **M. Eslami**, None; **S. Kazeminasab**, None; **V. Sharma**, None, **Y. Li**, None; **M. Fazli**, None; **M. Wang**, None; **N. Zebardast**, None; **T. Elze**, None

References

1. Fankhauser F, Koch P, Roulier A. On automation of perimetry. *Albrecht Von Graefes Arch Klin Exp Ophthalmol*. 1972;184(2):126–150.
2. Yohannan J, Boland MV. The Evolving Role of the Relationship between Optic Nerve Structure and Function in Glaucoma. *Ophthalmology*. 2017;124(12S):S66–S70.
3. Christopher M, Bowd C, Belghith A, et al. Deep Learning Approaches Predict Glaucomatous Visual Field Damage from OCT Optic Nerve Head En Face Images and Retinal Nerve Fiber Layer Thickness Maps. *Ophthalmology*. 2020; 127(3):346–356.
4. CRAN - Package vfprogression. https://cran.r-project.org/web/packages/vfprogression/index.html. Accessed February 14, 2022.
5. CRAN - Package visualFields. https://cran.r-project.org/web/packages/visualFields/index.html. Accessed February 14, 2022.
6. Marín-Franch I, Swanson WH. The visualFields package: a tool for analysis and visualization of visual fields. *J Vis*. 2013;13(4):10.
7. Eslami M, Kim JA, Zhang M, et al. Visual Field Prediction: Evaluating the Clinical Relevance of Deep Learning Models. *Ophthalmol Sci*. 2022;3(1):100222.
8. rpy2: Python interface to the R language from Python package index (PyPI). Available online at: https://pypi.org/project/rpy2/. Accessed February 14, 2022.
9. Pandas: a python package for data structures handling from python package index (PyPI). Available online at: https://pypi.org/project/pandas/. Accessed May 22, 2022.
10. Project Jupyter: a web-based interactive development environment for notebooks, code, and data. Available online at: https://jupyter.org/. Accessed May 22, 2022.
11. Spyder: A Scientific Python Development Environment from Python Package Index (PyPI). Available

online at: https://pypi.org/project/spyder/. Accessed May 22, 2022.
12. MIT License: Open-Source Initiative License Copyright. Available online at: https://mit-license.org/. Accessed May 22, 2022.
13. Saifee M, Wu J, Liu Y, et al. Development and validation of automated visual field report extraction platform using computer vision tools. *Front Med (Lausanne)*. 2021;8:625487.
14. PyVF: Python Visual Field simulation. Available online at: https://github.com/constructor-s/PyVF. Accessed May 22, 2022.
15. Liu P, McKendrick A, Ma-Wyatt A, Turpin A. A depth-dependent integrated VF simulation for analysis and visualization of glaucomatous VF defects. *Transl Vis Sci Technol*. 2020;9(3):8.
16. CRAN - Package binovisualfields. Available online at: https://cran.r-project.org/web/packages/binovisualfields/. Accessed May 22, 2022.
17. Berchuck SI, Mwanza JC, Warren JL. A spatially varying change points model for monitoring glaucoma progression using visual field data. *Spat Stat*. 2019;30:1–26.
18. CRAN - Package spCP. Available online at: https://cran.r-project.org/web/packages/spCP. Accessed May 22, 2022.
19. Naka M, Kanamori A, Tatsumi Y, et al. Comparison of mean deviation with AGIS and CIGTS scores in association with structural parameters in glaucomatous eyes. *J Glaucoma*. 2009;18(5):379–384.