popeye documentation

Release 0.1

Kevin DeSimone

Sep 21, 2017

Contents

Welcome to the documentation for **popeye**!

Contents:

CHAPTER 1

What is a population receptive field?

A population receptive field (pRF) is a quantitative model of the cumulative response of the population of cells contained within a single fMRI voxel[1]. The pRF model can be used to estimate the response properties of populations of neurons using other measures, such as EcOG and EEG[2].

The pRF model allows us to interpret and predict the responses of a voxel to different stimuli. Such models can be designed to describe various sensory[3] and cognitive[4] processes. More recently, we have used the pRF model to map the retinotopic organization of multiple subcortical nuclei[5].

Below are parameter maps detailing the response properties of the human lateral geniculate nucleus (LGN). The model parameter estimates are overlaid on the PD image for the left and right LGN. Separate subjects are shown in A and B. Columns illustrate the polar angle, eccentricity, pRF size, and HRF delay estimates. A, anterior; P, posterior.

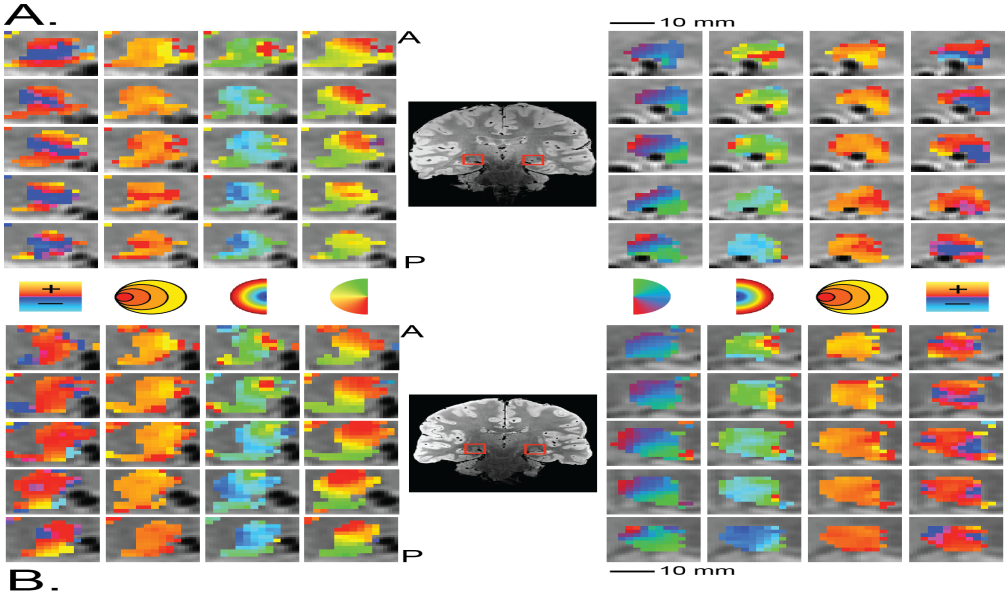
[1] Dumoulin S, and Wandell BA (2008). Population receptive field estimates in human visual cortex. Neuroimage 39: 647-60.

[2] Winawer J and Wandell BA (2015). Computational neuroimaging and population receptive fields. Trends in Cognitive Sciences 19: 349-357.

[3] Thomas JM, Huber E, Stecker E, Boynton G, Saenz M, Fine I (2014) Population receptive field estimates in human auditory cortex. NeuroImage 105: 428-439.

[4] Harvey BM, Klein BP, Petridou N, Dumoulin SO (2013) Topographic organization of numerosity in the human parietal cortex. Science 341: 1123-1126.

[5] DeSimone K, Viviano JD, Schneider KA (2015) Population receptive field estimation reveals two new maps in human subcortex. Journal of Neuroscience 35: 9836-9847.



References

CHAPTER 2

What is **popeye**?

popeye is an **open-source** project for developing, estimating, and validating pRF models. **popeye** was created by Kevin DeSimone with significant guidance and contributions from Ariel Rokem. **popeye** was developed because there was no open-source software for pRF modeling and estimation. The are several freely available MATLAB toolboxes for doing pRF model estimation, including mrVista and analyzePRF, but since these toolboxes are written in MATLAB they are not truly open-source. This motivated us to create **popeye** as project open to critique and colloboration.

The pRF model is what is known as a forward encoding model. We present the participant with a stimulus that varies over time in the tuning dimension(s) of interest. If we're interested in the retinotopic organization of some brain region, we would use a visual stimulus that varies over the horizontal and vertical dimensions of the visual field. Often experimenters will use a sweeping bar stimulus that makes several passes through the visual field to encode the visuotopic location of the measured response.

The **StimulusModel**, **PopulationModel**, and **PopulationFit** represent the fundamental units of **popeye**. Each of these are defined in **popeye.base**. **StimulusModel** is an abstract class containing the stimulus array—a NumPy ndarray—along with some supporting information about the stimulus timing and encoding. **PopulationModel** takes a **StimulusModel** as input and defines a **generate_prediction** method for operating on the stimulus via some user-defined receptive field model to produce a model prediction given some set of input parameters. **PopulationFit** takes a **PopulationModel** and a data timeseries (1D NumPy ndarray), estimating the the set of parameters of the **PopulationModel** that best fits the data.

CHAPTER 3

Getting started

Welcome to **popeye**! This section of the documentation will help you install **popeye** and its dependencies and then walks you through a small demonstration of how to use **popeye** for fitting some simulated timeseries data.

Installation

Download the popeye source code from the GitHub repository here. Using popeye requires that you have installed NumPy, SciPy, statsmodel, Cython, nibabel, and matplotlib.

Once you've downloaded the popeye source code and installed the dependencies, install popeye and build the Cython extensions I've written for speeding up the analyses.

```
:math:` cd popeye
` sudo python setup.py install build_ext
:math:` cd ~ # this ensures we test the install instead of version in the cwd
` python
>>> import popeye
>>> popeye.__version__
'0.1.0.dev'
```

Demo

Below is a small demonstration of how to interact with the popeye API. Here, we'll generate our stimulus and simulate the BOLD response of a Gaussian pRF model estimate we'll just invent. Normally, we'd be analyzing the BOLD timeseries that we collect while we present a participant with a visual stimulus.

```
import ctypes, multiprocessing
import numpy as np
import sharedmem
import popeye.og_hrf as og
import popeye.utilities as utils
```

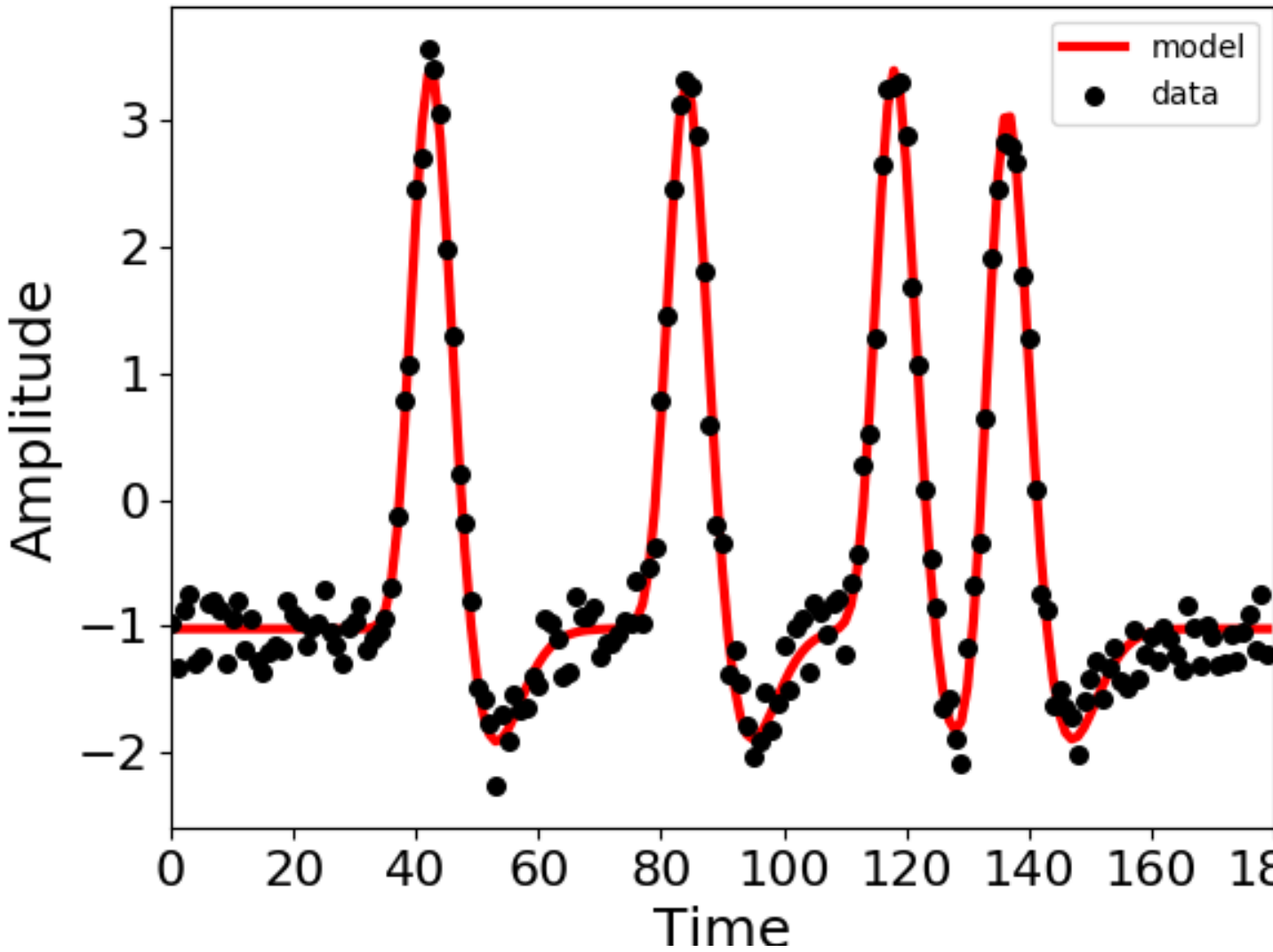
```
from popeye.visual_stimulus import VisualStimulus, simulate_bar_stimulus

# seed random number generator so we get the same answers ...
np.random.seed(2764932)

### STIMULUS
## create sweeping bar stimulus
sweeps = np.array([-1,0,90,180,270,-1]) # in degrees, -1 is blank
bar = simulate_bar_stimulus(100, 100, 40, 20, sweeps, 30, 30, 10)

## create an instance of the Stimulus class
stimulus = VisualStimulus(bar, 50, 25, 0.50, 1.0, ctypes.c_int16)

### MODEL
## initialize the gaussian model
model = og.GaussianModel(stimulus, utils.double_gamma_hrf)

## generate a random pRF estimate
x = -5.24
y = 2.58
sigma = 1.24
hrf_delay = -0.25
beta = 0.55
baseline = -0.88

## create the time-series for the invented pRF estimate
data = model.generate_prediction(x, y, sigma, hrf_delay, beta, baseline)

## add in some noise
data += np.random.uniform(-data.max()/10,data.max()/10,len(data))

### FIT
## define search grids
# these define min and max of the edge of the initial brute-force search.
x_grid = (-10,10)
y_grid = (-10,10)
s_grid = (1/stimulus.ppd + 0.25,5.25)
b_grid = (0.1,1.0)
h_grid = (-1.0,1.0)

## define search bounds
# these define the boundaries of the final gradient-descent search.
x_bound = (-12.0,12.0)
y_bound = (-12.0,12.0)
s_bound = (1/stimulus.ppd, 12.0) # smallest sigma is a pixel
b_bound = (1e-8,None)
u_bound = (None,None)
h_bound = (-3.0,3.0)

## package the grids and bounds
grids = (x_grid, y_grid, s_grid, h_grid)
bounds = (x_bound, y_bound, s_bound, h_bound, b_bound, u_bound,)

## fit the response
# auto_fit = True fits the model on assignment
# verbose = 0 is silent
# verbose = 1 is a single print
# verbose = 2 is very verbose
```

```
fit = og.GaussianFit(model, data, grids, bounds, Ns=3,
                     voxel_index=(1,2,3), auto_fit=True,verbose=2)
```

Inspecting the results

Below is the output of the model fit we invoked in the code block above. We also include some matplotlib code for plotting the simulated data and the predicted timeseries. Explore the attributes of the **fit** object to get a sense of thekinds of measures gleaned from the pRF model.

```
## plot the results
import matplotlib.pyplot as plt
plt.plot(fit.prediction,c='r',lw=3,label='model',zorder=1)
plt.scatter(range(len(fit.data)),fit.data,s=30,c='k',label='data',zorder=2)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.xlabel('Time',fontsize=18)
plt.ylabel('Amplitude',fontsize=18)
plt.xlim(0,len(fit.data))
plt.legend(loc=0)
```


model
data
Amplitude
Time
3
2
1
0
−1
−2
0
20
40
60
80
100
120
140
160

CHAPTER 4

API Reference

popeye

auditory

AuditoryFit
AuditoryModel
PopulationFit
PopulationModel
interp1d
auto_attr
fftconvolve
generate_rf_timeseries_1D

AuditoryFit

AuditoryModel

PopulationFit

PopulationModel

interp1d

auto_attr

fftconvolve

generate_rf_timeseries_1D

auditory_stimulus

AuditoryStimulus
StimulusModel
auto_attr
generate_spectrogram
imresize
specgram

AuditoryStimulus

StimulusModel

auto_attr

generate_spectrogram

imresize

specgram

base

PopulationFit
PopulationModel
StimulusModel
auto_attr
set_verbose

PopulationFit

PopulationModel

StimulusModel

auto_attr

set_verbose

css

CompressiveSpatialSummationFit
CompressiveSpatialSummationModel
PopulationFit
PopulationModel
auto_attr
fftconvolve
generate_og_receptive_field
generate_rf_timeseries

CompressiveSpatialSummationFit

CompressiveSpatialSummationModel

PopulationFit

PopulationModel

auto_attr

fftconvolve

generate_og_receptive_field

generate_rf_timeseries

dog

DifferenceOfGaussiansFit
DifferenceOfGaussiansModel
PopulationFit
PopulationModel
auto_attr
fftconvolve
generate_og_receptive_field

Continued on next page

Table 4.6 – continued from previous page

generate_rf_timeseries
linregress
trapz

DifferenceOfGaussiansFit

DifferenceOfGaussiansModel

PopulationFit

PopulationModel

auto_attr

fftconvolve

generate_og_receptive_field

generate_rf_timeseries

linregress

trapz

gabor

GaborFit
GaborModel
PopulationFit
PopulationModel
interp1d
auto_attr
brute
fmin_powell
generate_ballpark_prediction
linregress

GaborFit

GaborModel

PopulationFit

PopulationModel

interp1d

auto_attr

brute

fmin_powell

generate_ballpark_prediction

linregress

og

GaussianFit
GaussianModel
PopulationFit
PopulationModel
auto_attr
fftconvolve
generate_og_receptive_field
generate_rf_timeseries

GaussianFit

GaussianModel

PopulationFit

PopulationModel

auto_attr

fftconvolve

generate_og_receptive_field

generate_rf_timeseries

ogb

GaussianFit
GaussianModel
PopulationFit
PopulationModel
auto_attr
fftconvolve
generate_og_receptive_field
generate_rf_timeseries

GaussianFit

GaussianModel

PopulationFit

PopulationModel

auto_attr

fftconvolve

generate_og_receptive_field

generate_rf_timeseries

onetime

OneTimeProperty

Continued on next page

Table 4.10 – continued from previous page

ResetMixin
auto_attr
setattr_on_read

OneTimeProperty

ResetMixin

auto_attr

setattr_on_read

plotting

Circle
LogFormatterMathtext
LogNorm
gaussian_kde
beta_hist
eccentricity_hist
eccentricity_sigma_fill
eccentricity_sigma_scatter
field_coverage
find
generate_og_receptive_field
hexbin_location_map
hrf_delay_hist
hrf_delay_kde
lazy_field_coverage
linregress
location_and_size_map
location_estimate_jointdist
make_movie_calleable
make_movie_static
polar_angle_plot
sigma_hrf_delay_scatter

Circle

LogFormatterMathtext

LogNorm

gaussian_kde

beta_hist

eccentricity_hist

eccentricity_sigma_fill

eccentricity_sigma_scatter

field_coverage

find

generate_og_receptive_field

hexbin_location_map

hrf_delay_hist

hrf_delay_kde

lazy_field_coverage

linregress

location_and_size_map

location_estimate_jointdist

make_movie_calleable

make_movie_static

polar_angle_plot

sigma_hrf_delay_scatter

reconstruction

Process

Continued on next page

Table 4.12 – continued from previous page

interp1d	
Array	
Queue	A multi-producer, multi-consumer queue.
find	
multiprocess_stimulus_reconstruction	
multiprocess_stimulus_reconstruction_realtime	
reconstruct_stimulus	
reconstruct_stimulus_realtime	
reconstruct_stimulus_realtime_smoothing	

Process

interp1d

Array

Queue

find

multiprocess_stimulus_reconstruction

multiprocess_stimulus_reconstruction_realtime

reconstruct_stimulus

reconstruct_stimulus_realtime

reconstruct_stimulus_realtime_smoothing

simulation

VisualStimulus
repeat
error_function
fmin
fmin_powell
generate_og_receptive_field
generate_og_receptive_fields
generate_scatter_volume
parallel_simulate_neural_sigma
recast_simulation_results
resample_stimulus
shuffle
simulate_bar_stimulus
simulate_neural_sigma

VisualStimulus

repeat

error_function

fmin

fmin_powell

generate_og_receptive_field

generate_og_receptive_fields

generate_scatter_volume

parallel_simulate_neural_sigma

recast_simulation_results

resample_stimulus

shuffle

simulate_bar_stimulus

simulate_neural_sigma

spectrotemporal

PopulationFit
PopulationModel
SpectroTemporalFit
SpectroTemporalModel
interp1d
auto_attr
brute
compute_model_ts
decimate
fftconvolve
fmin_powell
gaussian_1D
gaussian_2D
generate_rf_timeseries_1D
generate_strf_timeseries
imresize
linregress
median_filter

Continued on next page

Table 4.14 – continued from previous page

parallel_fit
recast_estimation_results
romb
trapz

PopulationFit

PopulationModel

SpectroTemporalFit

SpectroTemporalModel

interp1d

auto_attr

brute

compute_model_ts

decimate

fftconvolve

fmin_powell

gaussian_1D

gaussian_2D

generate_rf_timeseries_1D

generate_strf_timeseries

imresize

linregress

median_filter

parallel_fit

recast_estimation_results

romb

trapz

utilities

repeat

Continued on next page

Table 4.15 – continued from previous page

Array
brute
brute_force_search
double_gamma_hrf
error_function
fmin
fmin_powell
gaussian_2D
generate_shared_array
gradient_descent_search
imresize
make_nifti
multiprocess_bundle
normalize
parallel_fit
percent_change
recast_estimation_results
romb
trapz
zscore

repeat

Array

brute

brute_force_search

double_gamma_hrf

error_function

fmin

fmin_powell

gaussian_2D

generate_shared_array

gradient_descent_search

imresize

make_nifti

multiprocess_bundle

normalize

parallel_fit

percent_change

recast_estimation_results

romb

trapz

zscore

version

---

visual_stimulus

StimulusModel
VisualStimulus
gaussian_2D
generate_coordinate_matrices
imread
imresize
loadmat
resample_stimulus
simulate_bar_stimulus
simulate_checkerboard_bar
simulate_movie_bar
simulate_sinflicker_bar
square
zoom

StimulusModel

VisualStimulus

gaussian_2D

generate_coordinate_matrices

imread

imresize

loadmat

resample_stimulus

simulate_bar_stimulus

simulate_checkerboard_bar

simulate_movie_bar

simulate_sinflicker_bar

square

zoom

xvalidation

coeff_of_determination

Continued on next page

Table 4.18 – continued from previous page

deepcopy
kfold_xval
parallel_xval
recast_xval_results

coeff_of_determination

deepcopy

kfold_xval

parallel_xval

recast_xval_results

CHAPTER 5

Indices and tables

- genindex
- modindex
- search