

# ontoFAST: getting started

*Sergei Tarasov*

*2017-10-17*

## Installing

To install ontoFAST package, you must have Rstudio and devtools package installed. The latter can be done by executing `install.packages("devtools")` in Rstudio. If devtools is intalled you can use it to install ontoFAST directly from GitHub:

```
# !!! uncomment to run
#library(devtools)
#install_github("sergeitarasov/ontoFAST@main")
library(ontoFAST)

## Loading required package: pbapply
## Loading required package: ontologyIndex
## Loading required package: dplyr
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##   filter, lag
## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union
## Loading required package: shiny
## Loading required package: shinydashboard
##
## Attaching package: 'shinydashboard'
## The following object is masked from 'package:graphics':
##
##   box
## Loading required package: visNetwork
## Loading required package: stringr
## Loading required package: magrittr
## Loading required package: devtools
##
## Attaching package: 'devtools'
## The following object is masked from 'package:ontologyIndex':
##
##   check
```

# Working with ontoFAST

## Assembling data

Let's first read in ontology. In this example, I use Hymenoptera anatomy ontology that is available as embedded data set:

```
hao_obo<-HAO
```

Alternatively ontology can be parsed directly from .obo file:

```
hao_obo=get_OBO(system.file("data_onto", "HAO.obo", package = "ontoFAST"),  
                 extract_tags="everything", propagate_relationships = c("BF0:0000050", "is_a"))
```

Not, let's read in character statements. I prefer using a simple table format to store characters and their states. So, it can be imported in R for example using `read.csv` function. Here, I use embedded data set from Sharkey et.al. (2011):

```
char_et_states<-Sharkey_2011
```

This data sets contains 392 characters. Let's create IDs for them, which will be used to call the characters. We keep all our data in `ontologyIndex` (here `hao_obo`) object that stores the entire ontology.

```
id_characters<-paste("CHAR:",c(1:392), sep="")  
hao_obo$id_characters<-id_characters
```

Now, let's add character statements to `hao_obo` and associate them with IDs.

```
name_characters<-char_et_states[,1]  
names(name_characters)<-id_characters  
hao_obo$name_characters<-name_characters
```

To make automatic annotation more efficient, we can use the synonyms of terms in the ontology. The synonyms are stored in `hao_obo$synonym` but they have to be preprocessed to be available for automatic annotation. To do it we use `syn_extract()` function.

```
hao_obo$parsed_synonyms<-syn_extract(hao_obo)
```

Now, we can run the automatic annotation. You can skip this step if you do not want to use synonyms.

```
hao_obo$auto_annot_characters<-annot_all_chars(hao_obo, use.synonyms=TRUE, min_set=TRUE)
```

```
## [1] "Doing automatic annotation of characters with ontology terms..."
```

## Running ontoFAST interactively

The final step before running ontoFAST in interactive mode, requires creating the global `shiny_in` object that will store all the annotations and associated data. To create the global object use the assignment operator `<-`

```
shiny_in<-make_shiny_in(hao_obo)
```

Now we can run ontoFAST interactively by executing the line below. It may take a few seconds until all characters are loaded.

```
# !!! uncomment to run  
#runOntoFast()
```

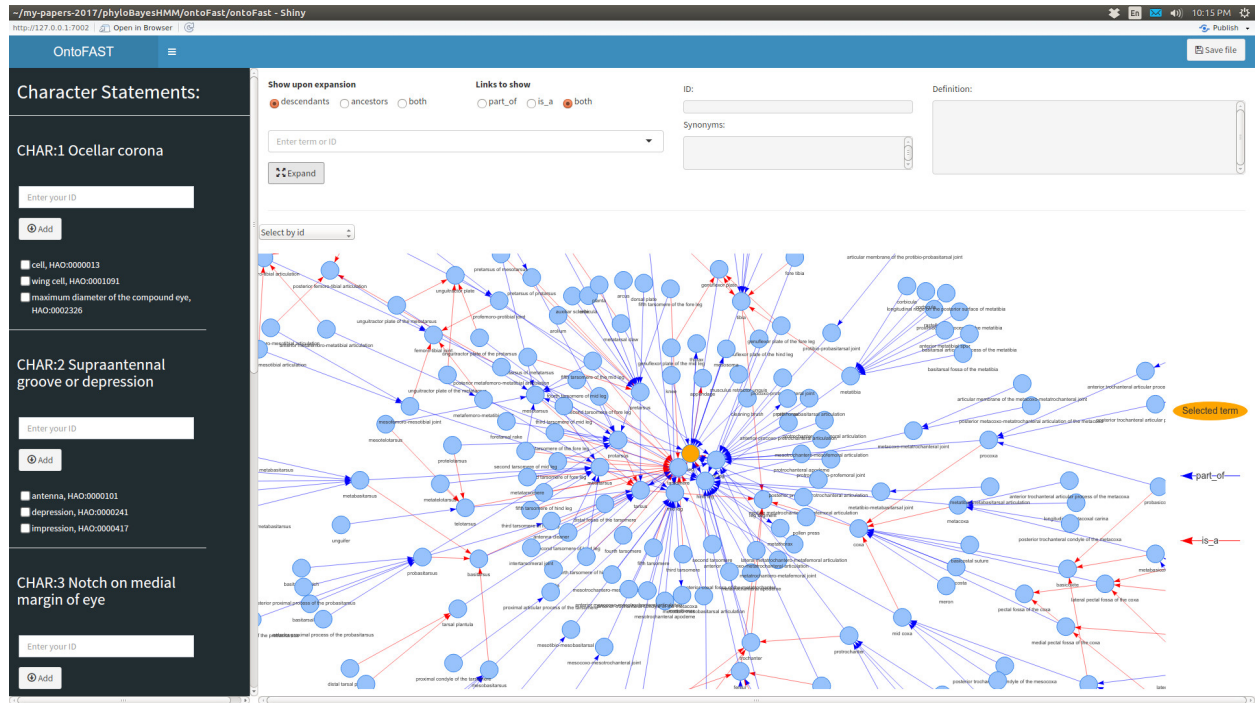


Figure 1: ontoFAST in interactive mode

By default ontoFAST displays all the characters in the data set. Using argument `nchar=N` you may restrict the visualization to N characters. You may also use ontoFAST as an ontology browser without loading characters by specifying argument `show.chars=FALSE`.