GPTCelltype: Reference-free and cost-effective automated cell type annotation with GPT-4 in single-cell RNA-seq analysis

Wenpin Hou[1,*], Zhicheng Ji[2,*]
[1]Department of Biostatistics, Columbia University Mailman School of Public Health Health
[2]Department of Biostatistics and Bioinformatics, Duke University School of Medicine
[*] corresponding authors

Introductions

Cell type annotation is an essential step in single-cell RNA-seq analysis. However, it is a time-consuming process that often requires expertise in collecting canonical marker genes and manually annotating cell types. Automated cell type annotation methods typically require the acquisition of high-quality reference datasets and the development of additional pipelines. We demonstrated that GPT-4, a highly potent large language model, can automatically and accurately annotate cell types by utilizing marker gene information generated from standard single-cell RNA-seq analysis pipelines in this manuscript. We developed this software, **GPTCelltype**, to provide an automated cell type annotation approach using GPT-4 for single-cell RNA-seq analysis.

Installation

GPTCelltype can be installed by following this instruction on Github.

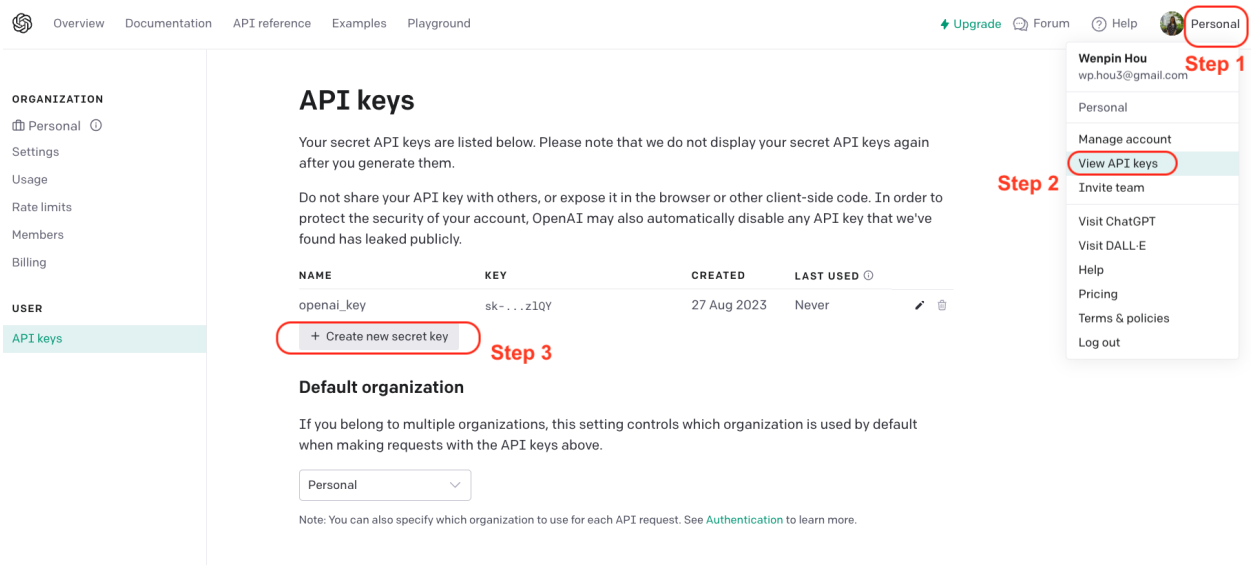
```
remotes::install_github("Winnie09/GPTCelltype")
```

GPTCelltype depends on the R package openai. Please make sure it is installed as well.

```
install.packages("openai")
```

OpenAI Key

GPTCelltype integrates the OpenAI API into the software. To connect to OpenAI API, a secret API key is required. You can generate your API key in your OpenAI account webpage: log in to OpenAI, click on "Personal" on the upper right corner, click on "View API keys" in the break-down list, and then click on "Create new secret key" which directs you to the API key page. Copy the key and paste it on a note for further use. Users need to pass their secret API key to GPTCelltype functions as one of the inputs to get cell type annotations. Avoid sharing your API key with others or uploading it to public spaces. Make sure it's not visible in browsers or any client-side scripts.



Run GPTCelltype

First of all, please load the packages.

```
library(GPTCelltype)
library(openai)
```

We demonstrate how to run GPTCelltype as follows. The main function is **`gptcelltype()`**. It can annotate cell types by OpenAI GPT models in a Seurat pipeline or with a custom gene list. If **`gptcelltype()`** is used in a Seurat pipeline, Seurat **`FindAllMarkers()`** function needs to be run first and the differential gene table generated by Seurat will serve as the input. If the input is a custom list of genes, one cell type is identified for each element in the list.

Among the input arguments, **`input`** can either be the differential gene table returned by Seurat **`FindAllMarkers()`** function, or a list of genes. **`tissuename`** (optional) is a tissue name. **`openai_key`** is your OpenAI key obtained from API key page (see above section). **`model`** is a valid GPT-4 or GPT-3.5 model name listed on Models page. Default is 'gpt-4'. **`topgenenumber`** is the number of top differential genes to be used when the input is Seurat differential genes. The output is a vector of cell types.

Example 1: Seurat object as input GPTCelltype integrates seamlessly with the Seurat pipeline. It can take an Seurat object as input, if the Seurat object has marker genes information. Specifically, this can be achieved after running the Seurat function `FindAllMarkers()`. Here follows an example.

Load the Seurat package.

```
library(Seurat, quietly = TRUE)
```

```
## Attaching SeuratObject
```

Set up your OpenAI key:

```
openaikey <- 'your openai key here'
```

In the below example, we are going to use a Seurat object called 'pbmc_small' provided by the Seurat package. In real applications, a Seurat project obtained after running the standard Seurat pipeline should be prepared. The Seurat project should have cell clustering available. Use FindAllMarkers() function to generate the differential gene table if you haven't done so:

```
data("pbmc_small")
all.markers <- FindAllMarkers(object = pbmc_small)
```

```
## Calculating cluster 0
```

```
## Calculating cluster 1
```

```
## Calculating cluster 2
```

Perform cell type annotation by GPT-4 using the gptcelltype() function. Here you can optionally provide the actual name of the tissue for your dataset.

```
res <- gptcelltype(all.markers,
            tissuename = 'human PBMC',
            openai_key = openaikey,
            ## Note: Please provide your OpenAI key to get cell type annotations;
            ## or otherwise the output is the prompt itself.
            model = 'gpt-4'
)
```

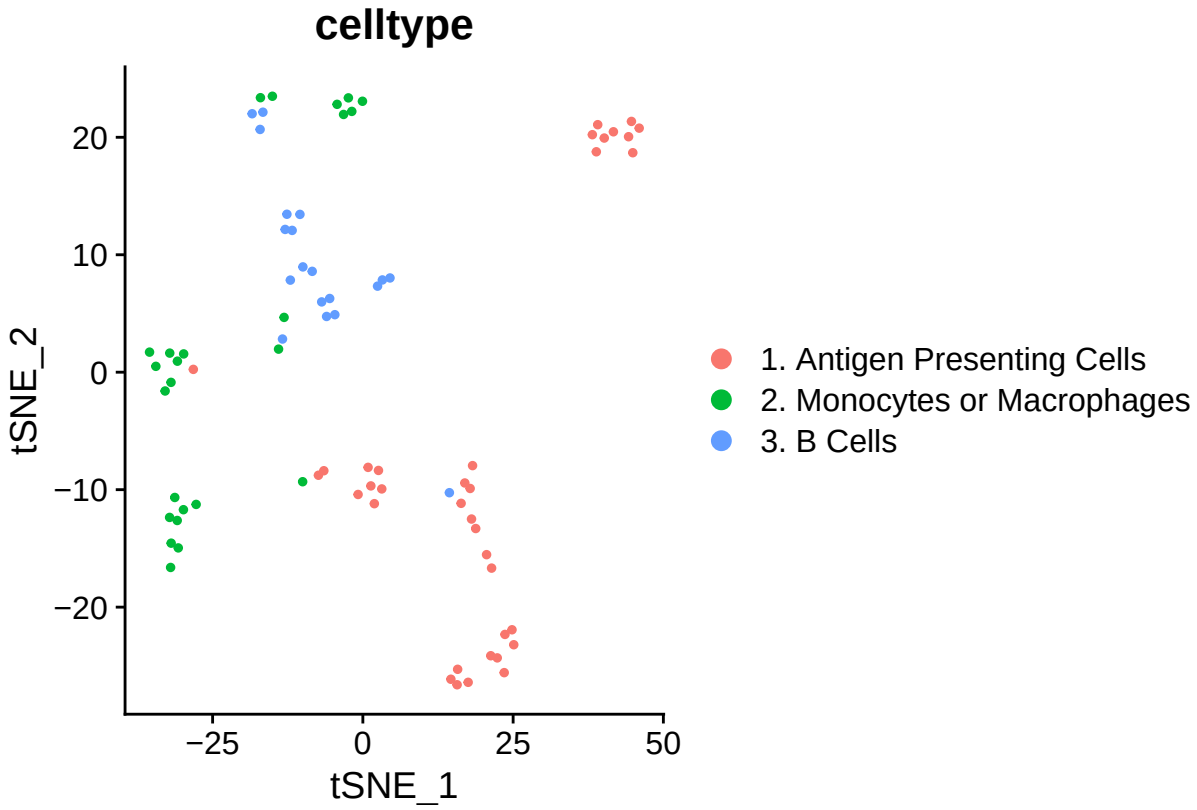
It is always recommended to check the results returned by GPT-4 in case of AI hallucination, before going to down-stream analysis.

```
res
```

```
##                             0                             1
## "1. Antigen Presenting Cells" "2. Monocytes or Macrophages"
##                             2
##                 "3. B Cells"
```

If the results make sense, we can assign the cell type annotations back to the Seurat object and visualize the cell type annotations on the UMAP:

```
pbmc_small$celltype <- res[as.character(Idents(pbmc_small))]
DimPlot(pbmc_small,group.by='celltype')
```



If the results need to be fine-tuned, it is easy to reassign cell type annotations for some clusters. For example, to change the cell type annotation for cluster 0:

```
res[1] <- 'Classical monocytes'
pbmc_small$celltype <- res[as.character(Idents(pbmc_small))]
```

If you prefer not to link to GPT-4 API or do not have OpenAI key, you can set `openai_key = NA`. In this case, the gptcelltype() function will print the prompt directly, which can be copied and pasted into the GPT-4 or ChatGPT online user interface to obtain cell type annotations.

```
data("pbmc_small")
all.markers <- FindAllMarkers(object = pbmc_small)
```

```
## Calculating cluster 0
## Calculating cluster 1
## Calculating cluster 2
```

```
res <- gptcelltype(all.markers,
            tissuename = 'human PBMC',
            openai_key = NA,
            ## Note: When NA, the output is the prompt itself.
            model = 'gpt-4'
)
cat(res)
```

```
## Identify cell types of human PBMC cells using the following markers separately for each row. Only pr
## 0:HLA-DPB1,HLA-DRB1,HLA-DPA1,HLA-DRA,HLA-DRB5,HLA-DQB1,LYZ,TYMP,HLA-DQA1,HLA-DMB
## 1:S100A8,TYMP,S100A9,LYZ,CST3,FCGRT,LST1,AIF1,TYROBP,IFITM3
## 2:HLA-DPB1,MS4A1,HLA-DQB1,HLA-DRB1,HLA-DRA,TCL1A,CD79A,CD79B,HLA-DPA1,HLA-DRB5
```

Example 2: use a list of genes as input If we provide a list of two gene vectors: the first vector contains *CD4* and *CD3D*, and the second vector contains *CD14*, then we can call the function in this way:

```
res <- gptcelltype(
  input = list(cluster1 = c('CD4, CD3D'), cluster2 = 'CD14'),
  tissuename = 'human PBMC',
  openai_key = openaikey,
  ## Note: Please provide your OpenAI key to get cell type annotations;
  ## or otherwise the output is the prompt itself.
  model = 'gpt-4'
)
res
```

```
##         cluster1         cluster2
## "T Helper Cells"      "Monocytes"
```

```
sessionInfo()
```

Session Info

```
## R version 4.0.2 (2020-06-22)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS  10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] SeuratObject_4.1.3 Seurat_4.3.0.1     openai_0.4.1       GPTCelltype_1.0
##
## loaded via a namespace (and not attached):
##   [1] Rtsne_0.16             colorspace_2.1-0       deldir_1.0-9
##   [4] ellipsis_0.3.2         ggridges_0.5.4         rstudioapi_0.14
##   [7] spatstat.data_3.0-1    farver_2.1.1           leiden_0.4.3
##  [10] listenv_0.9.0          ggrepel_0.9.3          fansi_1.0.4
##  [13] codetools_0.2-19       splines_4.0.2          knitr_1.42
##  [16] polyclip_1.10-4        jsonlite_1.8.4         ica_1.0-3
##  [19] cluster_2.1.4          png_0.1-8              uwot_0.1.14
##  [22] shiny_1.7.4            sctransform_0.3.5      spatstat.sparse_3.0-1
##  [25] compiler_4.0.2         httr_1.4.6             assertthat_0.2.1
##  [28] Matrix_1.5-4.1         fastmap_1.1.1          lazyeval_0.2.2
##  [31] limma_3.46.0           cli_3.6.1              later_1.3.1
##  [34] htmltools_0.5.5        tools_4.0.2            igraph_1.4.2
##  [37] gtable_0.3.3           glue_1.6.2             RANN_2.6.1
##  [40] reshape2_1.4.4         dplyr_1.1.2            Rcpp_1.0.10
##  [43] scattermore_1.1        vctrs_0.6.2            spatstat.explore_3.2-1
##  [46] nlme_3.1-162           progressr_0.13.0       lmtest_0.9-40
##  [49] spatstat.random_3.1-5  xfun_0.39              stringr_1.5.0
```

```
##  [52] globals_0.16.2         mime_0.12              miniUI_0.1.1.1
##  [55] lifecycle_1.0.3        irlba_2.3.5.1          goftest_1.2-3
##  [58] future_1.32.0          MASS_7.3-60            zoo_1.8-12
##  [61] scales_1.2.1           promises_1.2.0.1       spatstat.utils_3.0-3
##  [64] parallel_4.0.2         RColorBrewer_1.1-3     curl_5.0.0
##  [67] yaml_2.3.7             reticulate_1.28        pbapply_1.7-0
##  [70] gridExtra_2.3          ggplot2_3.4.2          stringi_1.7.12
##  [73] highr_0.10             rlang_1.1.1            pkgconfig_2.0.3
##  [76] matrixStats_0.63.0     evaluate_0.21          lattice_0.21-8
##  [79] ROCR_1.0-11            purrr_1.0.1            tensor_1.5
##  [82] labeling_0.4.2         patchwork_1.1.2        htmlwidgets_1.6.2
##  [85] cowplot_1.1.1          tidyselect_1.2.0       parallelly_1.35.0
##  [88] RcppAnnoy_0.0.20       plyr_1.8.8             magrittr_2.0.3
##  [91] R6_2.5.1               generics_0.1.3         withr_2.5.0
##  [94] pillar_1.9.0           fitdistrplus_1.1-11    survival_3.5-5
##  [97] abind_1.4-5            sp_1.6-0               tibble_3.2.1
## [100] future.apply_1.10.0    KernSmooth_2.23-21     utf8_1.2.3
## [103] spatstat.geom_3.2-1    plotly_4.10.1          rmarkdown_2.21
## [106] grid_4.0.2             data.table_1.14.8      digest_0.6.31
## [109] xtable_1.8-4           tidyr_1.3.0            httpuv_1.6.11
## [112] munsell_0.5.0          viridisLite_0.4.2
```