Introduction to mLLMCelltype

```r
library(mLLMCelltype)
```

Overview

mLLMCelltype is an R package that leverages various large language models (LLMs) for automated cell type annotation in single-cell RNA sequencing data. It implements a consensus-based approach where multiple LLMs collaborate to provide more reliable annotations. For more details, please refer to our paper: https://www.biorxiv.org/content/10.1101/2025.04.10.647852v1.

Installation

You can install the released version of mLLMCelltype from CRAN:

```r
install.packages("mLLMCelltype")
```

Or install the development version from GitHub:

```r
# install.packages("devtools")
devtools::install_github("cafferychen777/mLLMCelltype")
```

Setting Up API Keys

Before using mLLMCelltype, you need to set up API keys for the LLMs you want to use. The package supports multiple LLM providers including OpenAI, Anthropic, Google (Gemini), and X.AI (Grok).

```r
# Set API keys (recommended to use .Renviron or .env file)
# OpenAI API key
Sys.setenv(OPENAI_API_KEY = "your-openai-api-key")

# Anthropic API key
Sys.setenv(ANTHROPIC_API_KEY = "your-anthropic-api-key")

# Google API key
Sys.setenv(GOOGLE_API_KEY = "your-google-api-key")

# X.AI API key
Sys.setenv(XAI_API_KEY = "your-xai-api-key")

# Alternatively, use .env file
# dotenv::load_dot_env()
```

Basic Usage

Annotating Cell Types with Seurat Object

The main function of mLLMCelltype is to annotate cell types in a Seurat object:

```r
# Load required packages
library(mLLMCelltype)
library(Seurat)
library(dplyr)

# Load your preprocessed Seurat object
# pbmc <- readRDS("your_seurat_object.rds")

# Find marker genes for each cluster
# pbmc_markers <- FindAllMarkers(pbmc,
#                            only.pos = TRUE,
#                            min.pct = 0.25,
#                            logfc.threshold = 0.25)

# Set up cache directory to speed up processing
cache_dir <- "./mllmcelltype_cache"
dir.create(cache_dir, showWarnings = FALSE, recursive = TRUE)

# Run LLMCelltype annotation with multiple LLM models
consensus_results <- interactive_consensus_annotation(
  input = pbmc_markers,
  tissue_name = "human PBMC",  # provide tissue context
  models = c(
    "claude-3-7-sonnet-20250219",  # Anthropic
    "gpt-4o",                      # OpenAI
    "gemini-2.5-pro"               # Google
  ),
  api_keys = list(
    anthropic = Sys.getenv("ANTHROPIC_API_KEY"),
    openai = Sys.getenv("OPENAI_API_KEY"),
    gemini = Sys.getenv("GOOGLE_API_KEY")
  ),
  top_gene_count = 10,
  controversy_threshold = 1.0,
  entropy_threshold = 1.0,
  cache_dir = cache_dir
)

# Add annotations to Seurat object
# Get cell type annotations from consensus_results$final_annotations
cluster_to_celltype_map <- consensus_results$final_annotations

# Create new cell type identifier column
cell_types <- as.character(Idents(pbmc))
for (cluster_id in names(cluster_to_celltype_map)) {
  cell_types[cell_types == cluster_id] <- cluster_to_celltype_map[[cluster_id]]
}

# Add cell types to Seurat object
pbmc$mLLM_cell_type <- cell_types
```

Visualizing Results

```r
# Plot UMAP with cell type annotations
library(Seurat)
library(ggplot2)
library(cowplot)

# Basic visualization
p1 <- DimPlot(pbmc, group.by = "mLLM_cell_type", label = TRUE) +
  ggtitle("mLLMCelltype Consensus Annotations")

# Visualize consensus proportion (confidence metric)
pbmc$consensus_proportion <- 0
for (cluster_id in names(consensus_results$initial_results$consensus_results)) {
  cluster_cells <- which(Idents(pbmc) == cluster_id)
  pbmc$consensus_proportion[cluster_cells] <-
    consensus_results$initial_results$consensus_results[[cluster_id]]$consensus_proportion
}

p2 <- FeaturePlot(pbmc, features = "consensus_proportion",
                  min.cutoff = 0, max.cutoff = 1) +
  scale_color_gradientn(colors = c("blue", "green", "red")) +
  ggtitle("Consensus Proportion") +
  theme(legend.position = "right")

# Print consensus summary
print_consensus_summary(consensus_results)
```

Supported Models

mLLMCelltype supports various LLM models:

- **OpenAI**: gpt-4o, gpt-4o-mini, gpt-4.1, gpt-4.1-mini, gpt-4.1-nano, gpt-4-turbo, gpt-3.5-turbo, o1, o1-mini, o1-preview, o1-pro
- **Anthropic**: claude-3-7-sonnet-20250219, claude-3-5-sonnet-latest, claude-3-5-haiku-latest, claude-3-opus
- **DeepSeek**: deepseek-chat, deepseek-reasoner
- **Google**: gemini-2.5-pro, gemini-2.0-flash, gemini-2.0-flash-exp, gemini-1.5-pro, gemini-1.5-flash
- **Qwen**: qwen-max-2025-01-25
- **Stepfun**: step-2-mini, step-2-16k, step-1-8k
- **Zhipu**: glm-4-plus, glm-3-turbo
- **MiniMax**: minimax-text-01
- **Grok**: grok-3, grok-3-latest, grok-3-fast, grok-3-fast-latest, grok-3-mini, grok-3-mini-latest, grok-3-mini-fast, grok-3-mini-fast-latest
- **OpenRouter**: Access to models from multiple providers through a single API

Advanced Usage

Using a Single LLM Model

If you only want to use a single LLM model instead of the consensus approach:

```r
# Run annotation with a single model
single_model_results <- annotate_with_single_model(
  input = pbmc_markers,
  tissue_name = "human PBMC",
  model = "claude-3-7-sonnet-20250219",
  api_key = Sys.getenv("ANTHROPIC_API_KEY"),
  top_gene_count = 10,
  cache_dir = cache_dir
)

# Add annotations to Seurat object
pbmc$single_model_cell_type <- plyr::mapvalues(
  x = as.character(Idents(pbmc)),
  from = as.character(0:(length(single_model_results)-1)),
  to = single_model_results
)

# Visualize results
DimPlot(pbmc, group.by = "single_model_cell_type", label = TRUE) +
  ggtitle("Cell Types Annotated by Single LLM Model")
```

Customizing Consensus Parameters

```r
# Customize consensus parameters
custom_consensus_results <- interactive_consensus_annotation(
  input = pbmc_markers,
  tissue_name = "human PBMC",
  models = c("claude-3-7-sonnet-20250219", "gpt-4o", "gemini-2.5-pro"),
  api_keys = list(
    anthropic = Sys.getenv("ANTHROPIC_API_KEY"),
    openai = Sys.getenv("OPENAI_API_KEY"),
    gemini = Sys.getenv("GOOGLE_API_KEY")
  ),
  top_gene_count = 15,                # Number of top marker genes to use
  controversy_threshold = 0.7,        # Threshold for controversy detection
  entropy_threshold = 0.7,            # Entropy threshold for controversy detection
  max_discussion_rounds = 2,          # Maximum rounds of discussion
  cache_dir = cache_dir
)
```

Using Custom Providers

You can register custom LLM providers:

```r
# Register a custom provider
register_custom_provider(
  provider_name = "my_custom_provider",
  process_function = my_process_function,
  models = c("my-model-1", "my-model-2")
)

# Use the custom provider
results <- annotate_cell_types(
  seurat_obj = seurat_obj,
  models = c("my-model-1", "claude-3-7-sonnet-20250219"),
  # other parameters
)
```

Caching Results

mLLMCelltype includes a caching system to save API costs:

```r
# Enable caching
Sys.setenv(MLLM_CACHE_ENABLED = "TRUE")

# Set cache directory
Sys.setenv(MLLM_CACHE_DIR = "path/to/cache")

# Clear cache if needed
clear_cache()
```

Conclusion

mLLMCelltype provides a powerful and flexible framework for automated cell type annotation using large language models. By leveraging the consensus of multiple models, it achieves more reliable annotations than single-model approaches.

For bug reports or feature requests, please visit: https://github.com/cafferychen777/mLLMCelltype/issues.