自定义分析模板

这是一个用于创建您自己的 AlphaGenome 分析的模板 notebook。

如何使用此模板：

1. 将此 notebook 复制到您的个人工作空间（`~/work/`）
2. 重新命名以描述您的分析
3. 修改下面的代码以满足您的需求
4. 定期保存您的结果

分析元数据

为您的自定义分析填写这些详细信息：

In [1]:
```
ANALYSIS_NAME = "我的自定义分析"
ANALYSIS_DATE = "2025-02-08"
ANALYST = "您的名字"
DESCRIPTION = "在此描述您的分析"

print(f"分析: {ANALYSIS_NAME}")
print(f"日期: {ANALYSIS_DATE}")
print(f"分析员: {ANALYST}")
print(f"描述: {DESCRIPTION}")
```

```
分析: 我的自定义分析
日期: 2025-02-08
分析员: 您的名字
描述: 在此描述您的分析
```

1. 导入库

In [2]:
```
# ======================
# 标准库
# ======================
import os
import sys
import json
from pathlib import Path
from datetime import datetime

# ======================
# 数据处理
# ======================
import numpy as np
import pandas as pd

# ======================
# 可视化
# ======================
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# ======================
# AlphaGenome
# ======================
from alphagenome.data import genome
from alphagenome.models import dna_client

# ======================
# 自定义工具
# ======================
sys.path.insert(0, '/shared/tools')
from alphagenome_tools import (
    batch_predict_variants,
    batch_predict_sequences,
    load_variants_from_csv,
    load_intervals_from_csv,
    monitor_api_quota,
    save_results,
    export_to_csv,
    export_to_excel
)

print("✓ 所有库已导入")
print(f"✓ 准备好进行: {ANALYSIS_NAME}")
```

```
Warning: alphagenome package not installed. Install with: pip install alph
agenome
✓ 所有库已导入
✓ 准备好进行: 我的自定义分析
```

```
/usr/local/lib/python3.10/dist-packages/tqdm/auto.py:21: TqdmWarning: IPro
gress not found. Please update jupyter and ipywidgets. See https://ipywidg
ets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

2. 配置

In [3]:

```
# ======================
# 分析参数
# ======================

# 输出设置
OUTPUT_DIR = Path.home() / 'work' / 'results' / ANALYSIS_NAME.lower().rep
OUTPUT_DIR.mkdir(parents=True, exist_ok=True)

# 分析设置
WINDOW_SIZE = 100000  # 变异分析的窗口大小（bp）
ONTOLOGY_TERMS = []  # 可选: ['UBERON:0001157']
OUTPUT_TYPES = None  # 默认值为 None，或指定: [dna_client.OutputType.RNA_SEQ

# 打印配置
print("配置:")
print(f"  输出目录: {OUTPUT_DIR}")
print(f"  窗口大小: {WINDOW_SIZE:,} bp")
print(f"  本体术语: {ONTOLOGY_TERMS if ONTOLOGY_TERMS else '未指定'}")
print(f"  输出类型: {OUTPUT_TYPES if OUTPUT_TYPES else '默认'}")
```

```
配置:
  输出目录: /home/admin/work/results/我的自定义分析
  窗口大小: 100,000 bp
  本体术语: 未指定
  输出类型: 默认
```

3. 连接到 AlphaGenome

In [4]:
```python
# 从环境变量获取 API 密钥
api_key = os.environ.get('ALPHAGENOME_API_KEY')
if not api_key:
    raise ValueError(
        "环境变量中未找到 ALPHAGENOME_API_KEY! "
        "请确保在 docker-compose.yml 中已设置"
    )

# 创建模型连接
model = dna_client.create(api_key=api_key)

print("✓ 已连接到 AlphaGenome")

# 检查 API 配额
monitor = monitor_api_quota()
print(f"\n{monitor}")
```

```
✓ 已连接到 AlphaGenome

API Usage: 32/1,000,000 calls (999,968 remaining)
```

4. 加载您的数据

选择下面的选项之一或创建您自己的数据加载逻辑。

选项 A: 从 CSV 加载变异

In [5]:
```python
# 取消注释并修改以从 CSV 加载

# csv_path = 'path/to/your/variants.csv'
# variants = load_variants_from_csv(csv_path)
# print(f"✓ 从 {csv_path} 加载了 {len(variants)} 个变异")

print("取消注释上面的代码以从 CSV 加载变异")
```

```
取消注释上面的代码以从 CSV 加载变异
```

选项 B: 从 CSV 加载区间

In [6]:
```python
# 取消注释并修改以加载区间

# csv_path = 'path/to/your/intervals.csv'
# intervals = load_intervals_from_csv(csv_path)
# print(f"✓ 从 {csv_path} 加载了 {len(intervals)} 个区间")

print("取消注释上面的代码以从 CSV 加载区间")
```

```
取消注释上面的代码以从 CSV 加载区间
```

选项 C: 以编程方式定义数据

In [7]:
```
# 示例: 定义您自己的变异

# variants = [
#     genome.Variant('chr22', 36201698, 'A', 'C'),
#     genome.Variant('chr22', 36202000, 'G', 'T'),
#     # 添加更多变异...
# ]

print("在此定义您的变异或区间")
```

在此定义您的变异或区间

选项 D: 从其他来源加载（VCF、BED 等）

In [8]:
```
# 示例: 从 VCF 文件加载

# def load_vcf(filepath):
#     """从 VCF 文件加载变异。"""
#     variants = []
#     with open(filepath, 'r') as f:
#         for line in f:
#             if line.startswith('#'):
#                 continue
#             fields = line.strip().split('\t')
#             chrom = fields[0]
#             pos = int(fields[1])
#             ref = fields[3]
#             alts = fields[4].split(',')
#             for alt in alts:
#                 variants.append(genome.Variant(chrom, pos, ref, alt))
#     return variants

# variants = load_vcf('path/to/your/file.vcf')

print("在此定义自定义加载函数")
```

在此定义自定义加载函数

5. 预览您的数据

In [9]:
```
# 加载数据后，在此预览它

# 对于变异:
# if 'variants' in locals():
#     variants_df = pd.DataFrame([{
#         'chromosome': v.chromosome,
#         'position': v.position,
#         'reference': v.reference_bases,
#         'alternate': v.alternate_bases
#     } for v in variants])
#     print(f"总计: {len(variants)} 个变异")
#     display(variants_df.head())

# 对于区间:
# if 'intervals' in locals():
```

```
#     intervals_df = pd.DataFrame([{
#         'chromosome': i.chromosome,
#         'start': i.start,
#         'end': i.end,
#         'length': i.end - i.start
#     } for i in intervals])
#     print(f"总计: {len(intervals)} 个区间")
#     display(intervals_df.head())

print("在此预览您加载的数据")
```

在此预览您加载的数据

6. 运行分析

修改此部分以执行您的自定义分析。

In [10]:
```
# ======================
# 您的自定义分析代码
# ======================

# 示例: 批量预测变异
# if 'variants' in locals():
#     results = batch_predict_variants(
#         variants=variants,
#         model=model,
#         ontology_terms=ONTOLOGY_TERMS,
#         requested_outputs=OUTPUT_TYPES,
#         show_progress=True,
#         monitor=True
#     )
#     print("\n✓ 分析完成!")
#     print(f"成功: {results['success'].sum()}/{len(results)}")

# 示例: 批量预测区间
# if 'intervals' in locals():
#     results = batch_predict_sequences(
#         intervals=intervals,
#         model=model,
#         requested_outputs=OUTPUT_TYPES,
#         show_progress=True,
#         monitor=True
#     )
#     print("\n✓ 分析完成!")

print("在此添加您的自定义分析代码")
```

在此添加您的自定义分析代码

7. 分析结果

In [11]:
```
# ======================
# 自定义结果分析
# ======================

# 示例: 汇总统计
# if 'results' in locals():
#     print("汇总统计:")
```

```
#     print(results.describe())
#
#     # 按染色体分组
#     if 'chromosome' in results.columns:
#         print("\n按染色体:")
#         print(results.groupby('chromosome').size())

print("在此添加您的结果分析代码")
```

在此添加您的结果分析代码

8. 可视化结果

In [12]:
```
# ======================
# 自定义可视化
# ======================

# 示例: 创建自定义图表
# if 'results' in locals():
#     fig, ax = plt.subplots(figsize=(12, 6))
#     # 您的绘图代码
#     plt.show()

print("在此添加您的可视化代码")
```

在此添加您的可视化代码

9. 保存结果

In [13]:
```
# ======================
# 保存所有结果
# ======================

# 保存数据
# if 'results' in locals():
#     export_to_csv(results, OUTPUT_DIR / 'results.csv')
#     export_to_excel(results, OUTPUT_DIR / 'results.xlsx')
#     print(f"✓ 结果已保存到 {OUTPUT_DIR}")

# 保存图表
# if 'fig' in locals():
#     fig.savefig(OUTPUT_DIR / 'figure.png', dpi=300, bbox_inches='tight'
#     fig.savefig(OUTPUT_DIR / 'figure.pdf', bbox_inches='tight')
#     print(f"✓ 图表已保存到 {OUTPUT_DIR}")

# 保存元数据
metadata = {
    'analysis_name': ANALYSIS_NAME,
    'date': ANALYSIS_DATE,
    'analyst': ANALYST,
    'description': DESCRIPTION,
    'timestamp': datetime.now().isoformat()
}

with open(OUTPUT_DIR / 'metadata.json', 'w') as f:
    json.dump(metadata, f, indent=2)
```

```
print(f"✓ 元数据已保存")
print(f"\n所有输出在: {OUTPUT_DIR}")
```

✓ 元数据已保存

所有输出在: /home/admin/work/results/我的自定义分析

10. 自定义函数

在此为您的分析定义任何可重用的函数。

In [14]:
```
# ======================
# 您的自定义函数
# ======================

# 示例:
# def my_custom_analysis_function(data):
#     """描述您的函数做什么。"""
#     # 您的代码
#     return result

print("在此定义您的自定义函数")
```

在此定义您的自定义函数

11. 注释和文档

分析计划:

1. 描述您的第一步
2. 描述您的第二步
3. 等等

预期结果:

- 您期望找到什么
- 关键假设

注释:

- 添加有关问题或发现的注释
- 记录与计划的任何偏差

下一步:

- 后续分析
- 要运行的其他实验

模板完成!

您已到达自定义分析模板的末尾。

使用此模板的提示：

1. **从简单开始**：从基本分析开始，然后添加复杂性
2. **频繁保存**：在每个主要步骤后导出结果
3. **详细文档**：添加注释说明您的方法
4. **小范围测试**：先尝试使用小数据集
5. **监控 API**：跟踪您的配额使用

相关 Notebooks：

- **01_quickstart.ipynb** - 学习基础知识
- **02_variant_analysis.ipynb** - 单个变异示例
- **03_batch_analysis.ipynb** - 批量处理示例
- **04_visualization.ipynb** - 高级绘图技术

需要帮助?

- 查看 AlphaGenome 文档
- 查看示例 notebooks
- 咨询您的团队

祝您分析顺利！