Automated Bioinformatics Analysis via AutoBA

Juexiao Zhou[1,2,†], Bin Zhang[1,2,†], Xiuying Chen[1,2], Haoyang Li[1,2], Xiaopeng Xu[1,2], Siyuan Chen[1,2], Xin Gao[1,2,*]

Abstract—With the fast-growing and evolving omics data, the demand for streamlined and adaptable tools to handle the analysis continues to grow. In response to this need, we introduce Auto Bioinformatics Analysis (AutoBA), an autonomous AI agent based on a large language model designed explicitly for conventional omics data analysis. AutoBA simplifies the analytical process by requiring minimal user input while delivering detailed step-by-step plans for various bioinformatics tasks. Through rigorous validation by expert bioinformaticians, AutoBA's robustness and adaptability are affirmed across a diverse range of omics analysis cases, including whole genome sequencing (WGS), RNA sequencing (RNA-seq), single-cell RNA-seq, ChIP-seq, and spatial transcriptomics. AutoBA's unique capacity to self-design analysis processes based on input data variations further underscores its versatility. Compared with online bioinformatic services, AutoBA deploys the analysis locally, preserving data privacy. Moreover, different from the predefined pipeline, AutoBA has adaptability in sync with emerging bioinformatics tools. Overall, AutoBA represents a convenient tool, offering robustness and adaptability for complex omics data analysis.

Index Terms—Bioinformatics, Omics analysis, Large language model, Agent.

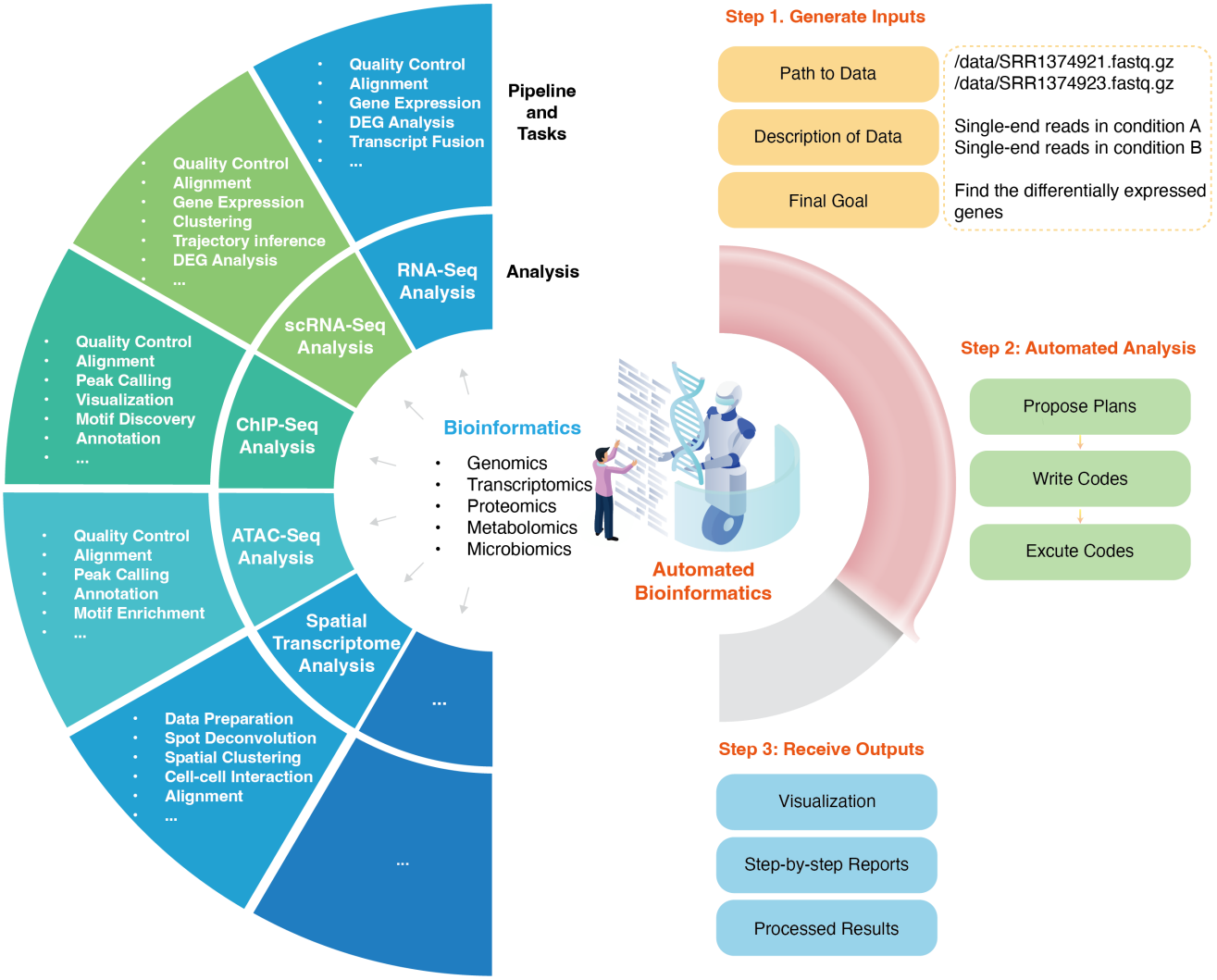
1 Introduction

Bioinformatics is an interdisciplinary field that encompasses computational, statistical, and biological approaches to analyze, understand and interpret complex biological data [1], [2], [3]. With the rapid growth of gigabyte-sized biological data generated from various high-throughput technologies, bioinformatics has become an essential tool for researchers to make sense of these massive datasets and extract meaningful biological insights. The applications of bioinformatics typically cover diverse fields such as genome analysis [4], [5], [6], structural bioinformatics [7], [8], [9], systems biology [10], data and text mining [11], [12], [13], phylogenetics [14], [15], [16], and population analysis [17], [18], which has further enabled significant advances in personalized medicine [19], and drug discovery [5].

In broad terms, bioinformatics could be categorized into two primary domains: the development of innovative algorithms to address various biological challenges [20], [21], [22], [23], [24], and the application of established tools to analyze extensive biological datasets [25], [26]. Developing new bioinformatics software requires a substantial grasp of biology and programming expertise. Alongside the development of novel computational methods, one of the most prevalent applications of bioinformatics is the investigation of biological data using the existing tools and pipelines [27], [28], which typically involves a sequential, flow-based analysis of omics data, encompassing a variety types of datasets like whole genome sequencing (WGS) [29], whole exome sequencing (WES), RNA sequencing (RNA-seq) [30], single-cell RNA-seq (scRNA-Seq) [31], transposase-accessible chromatin with sequencing (ATAC-Seq) [32], ChIP-seq [33], and spatial transcriptomics [34].

For example, the conventional analytical framework for bulk RNA-seq involves a meticulously structured sequence of computational steps [35]. This intricate pipeline reveals its complexity through a series of carefully orchestrated stages. It begins with quality control [36], progresses to tasks such as adapter trimming [37] and the removal of low-quality reads, and then moves on to critical steps like genome or transcriptome alignment [38]. Furthermore, it extends to some advanced tasks, including the identification of splice junctions [39], quantification through read counting [40], and the rigorous examination of differential gene expression [41]. Moreover, the pipeline delves into the intricate domain of alternative splicing [42] and isoform analysis [43]. This progressive journey ultimately ends in downstream tasks like the exploration of functional enrichment [44], providing a comprehensive range of analytical pursuits. Compared to bulk RNA-seq, ChIP-seq involves distinct downstream tasks, such as peak calling [45], motif discovery [46], peak annotation [47] and so on. In summary, the analysis of different types of omics data requires professional skills and an understanding of the corresponding field. Moreover, the

[1]*Computer Science Program, Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Kingdom of Saudi Arabia*
[2]*Computational Bioscience Research Center, King Abdullah University of Science and Technology, Thuwal 23955-6900, Kingdom of Saudi Arabia*
[†]*These authors contributed equally to this work.*
[*]*To whom correspondence should be addressed; E-mail: xin.gao@kaust.edu.sa.*



Methods	Easy to Master	Save Time	Rapid Redevelopment	Generalizability	Robustness	Privacy
AutoBA	✓✓✓	✓✓✓	✓✓✓	✓✓✓	✓✓✓	✓✓✓
Conventional Bioinformatics Tools	✓	✓	✓✓	✓	✓✓	✓✓✓
Online Webserver	✓✓	✓✓	✗	✓	✓	✗

Fig. 1. **Design of AutoBA.** AutoBA stands as the first autonomous AI agent meticulously crafted for traditional bioinformatics analysis. Remarkably user-friendly, AutoBA necessitates only three inputs from users: data path, data description, and analysis objective. With these inputs, it autonomously generates comprehensive analysis plans, authors code, executes code, and conducts the analysis, offering a streamlined and efficient solution for bioinformatics tasks.

methods and pipelines might vary across different bioinformaticians and they even may evolve with the development of more advanced algorithms.

In the context described above, the bioinformatics community grapples with essential concerns regarding the standardization, portability, and reproducibility of analysis pipelines [48], [49], [50]. Moreover, achieving proficiency in utilizing these pipelines for data analysis demands additional training, posing challenges for many wet lab researchers due to its potential complexity and time-consuming nature. Even dry-lab researchers may find the repetitive process of running and debugging these pipelines to be quite tedious [51]. Consequently, there is a growing anticipation within the community for the development of a more user-friendly, low-code, multi-functional, automated, and natural language-driven intelligent tool tailored for bioinformatics analysis. Such a tool has the potential to generate significant excitement and benefit researchers across the field.

Over the past few months, the rapid advancement of Large Language Models (LLMs) [52] has raised substantial expectations for the enhancement of scientific research,

particularly in the field of biology [53], [54], [55]. These advancements hold promise for applications such as disease diagnosis [56], [57], [58], [59], drug discovery [60], and all. In the realm of bioinformatics, LLMs, such as ChatGPT, also demonstrate immense potential in tasks related to bioinformatics education [61] and code generation [62]. While researchers have found ChatGPT to be a valuable tool in facilitating bioinformatics research, such as data analysis, there remains a strong requirement for human involvement in the process. AutoGPT [63], as a recently developed, advanced, and experimental open-source autonomous AI agent, has the capacity to string together LLM-generated "thoughts" to autonomously achieve user-defined objectives. Nevertheless, given the intricate and specialized nature of bioinformatics tasks, the direct application of AutoGPT in this field still presents significant challenges.

Therefore, in this work, we present Auto Bioinformatics Analysis (AutoBA), the first autonomous AI agent meticulously crafted for conventional bioinformatics analysis. AutoBA streamlines user interactions by soliciting just three inputs: the data path, the data description, and the final objective. AutoBA possesses the capability to autonomously generate analysis plans, write codes, execute codes, and perform subsequent data analysis. In essence, AutoBA marks the pioneering application of LLMs and automated AI agents in the realm of bioinformatics, showcasing the immense potential to expedite future bioinformatics research endeavors.

2 METHODS

2.1 The overall framework design of AutoBA

AutoBA is the first autonomous AI agent tailor-made for conventional bioinformatics analysis. As illustrated in Figure 1, conventional bioinformatics typically entails the use of pipelines to analyze diverse data types such as WGS, WES, RNA-seq, single-cell RNA-seq, ChIP-seq, ATAC-seq, spatial transcriptomics, and more, all requiring the utilization of various software tools. Users are traditionally tasked with selecting the appropriate software tools based on their specific analysis needs. In practice, this process involves configuring the environment, installing software, writing code, and addressing code-related issues, which are time-consuming and labour-intensive.

With the advent of AutoBA, this labor-intensive process is revolutionized. Users are relieved from the burden of dealing with multiple software packages and need only provide three key inputs: the data path (e.g., */data/SRR1374921.fasta.gz*), data description (e.g., *single-end reads in condition A*), and the ultimate analysis goal (e.g., *identify differentially expressed genes*). AutoBA takes over by autonomously analyzing the data, generating comprehensive step-by-step plans, composing code for each step, executing the generated code, and conducting in-depth analysis. Depending on the complexity and difficulty of the tasks, users can expect AutoBA to complete the tasks within a matter of minutes to a few hours, all without the need for additional manual labor.

2.2 Prompt engineering of AutoBA

To initiate AutoBA, users provide three essential inputs: the data path, data description, and the previously mentioned analysis objective. AutoBA comprises two distinct phases: a planning phase and an execution phase. During the planning phase, AutoBA meticulously outlines a comprehensive step-by-step analysis blueprint. This blueprint includes details such as the software name and version to be used at each step, along with guided actions and specific subgoals for each stage. Subsequently, in the execution phase, AutoBA systematically follows the plan from the initial step onward. This entails tasks like configuring the environment, installing necessary software, writing code, and executing the generated code. In light of this workflow, AutoBA incorporates two distinct prompts: one tailored for the planning phase and the other for the execution phase. Experience has shown that these two sets of cues are essential for the proper functioning of AutoBA in automated bioinformatics analysis tasks.

The prompt for the planning phase is displayed as follows:

```
prompt = {
    "role": "Act as a bioinformatician, the
        rules must be strictly followed!",
    "rules": [
        "When acting as a bioinformatician,
            you strictly cannot stop acting
            as a bioinformatician.",
        "All rules must be followed strictly
            .",
        "You should use information in input
             to write a detailed plan to
            finish your goal.",
        f"You should include the software
            name and should not use those
            software: {blacklist}.",
        "You should only respond in JSON
            with the required format.",
        "Your JSON should enclosed in double
             quotes."
    ],
    "input": [
        "You have the following information
            in a list with the format file
            path: file description. I
            provide those files to you, so
            you don't need to prepare the
            data.",
        data_list
    ],
    "goal": current_goal,
    "format": {
        "plan": [
```

```
            "Your detailed step-by-step sub-
                tasks to finish your goal."
        ]
    }
}
```

The prompt for the execution phase is displayed as follows:

```
prompt = {
    "role": "Act as a bioinformatician, the
        rules must be strictly followed!",
    "rules": [
        "When acting as a bioinformatician,
            you strictly cannot stop acting
            as a bioinformatician.",
        "All rules must be followed strictly
            .",
        "You are provided a system with
            specified constraints."
        "The history of what you have done
            is provided, you should take the
             name changes of some files into
             account, or use some output
            from previous steps.",
        "You should use all information you
            have to write bash codes to
            finish your current task.",
        "All code requirements must be
            followed strictly when you write
             codes.",
        "You should only respond in JSON
            with the required format.",
        "Your JSON should enclosed in double
             quotes."
    ],
    "system": [
        "You have a Ubuntu 18.04 system",
        "You have a conda environment named
            abc",
        "You do not have any other software
            installed"
    ],
    "input": [
        "You have the following information
            in a list with the format file
            path: file description. I
            provide those files to you, so
            you don't need to prepare the
            data.",
        data_list
    ],
    "history": history_summary,
    "current task": current_goal,
    "code requirement": [
        f"You should not use those software:
             {blacklist}.",
        'You should always source activate
            the environment abc first.',
        'You should always install
            dependencies with -y with conda
            or pip.',
        'You should pay attention to the
            number of input files and do not
             miss any.',
        'You should process each file
            independently and can not use
            the FOR loop.',
        'You should use the path for all
            files according to input and
            history.',
        'You should use the default values
            for all parameters that are not
            specified.',
        'You should not repeat what you have
             done in history.',
        'You should only use software
            directly you installed with
            conda.',
    ],
    "format": {
        "tool": "name of the tool you use",
        "code": "bash code to finish the
            current task"
    }
}
```

In the two aforementioned prompt designs, the term *blacklist* pertains to the user's personalized list of prohibited software. Meanwhile, *data list* encompasses the inputs necessary for AutoBA, encompassing data paths and data descriptions. The term *current goal* serves as the final objective during the planning phase and as the sub-goal in the execution phase, while *history summary* encapsulates AutoBA's memory of previous actions and information.

2.3 Memory management of AutoBA

A memory mechanism is incorporated within AutoBA to enable it to generate code more effectively by drawing from past actions, thus avoiding unnecessary repetition of certain steps. AutoBA meticulously logs the outcome of each step in a specific format, and all these historical records become part of the input for the subsequent prompt. In the planning phase, memories are structured as follows: "Firstly, you provided input in the format 'file path: file description' in a list: <data list>. You devised a detailed plan to accomplish your overarching objective. Your overarching goal is <global goal>. Your plan involves <tasks>." In the execution phase, memories follow this format: "Then, you successfully completed the task: <task> with the corresponding code: <code>."

2.4 Evaluation of AutoBA

The results produced by AutoBA undergo thorough validation by an expert bioinformatician. This validation process encompasses a comprehensive review of the generated code, execution of the code, and confirmation of the results for accuracy and reliability. AutoBA's development and validation are built upon a specific environment and software stack, which includes Ubuntu version 18.04, Python 3.10.0, and openai version 0.27.6. These environment and software specifications form the robust foundation for AutoBA's functionality in the field of bioinformatics, ensuring its reliability and effectiveness.

3 Results

3.1 AutoBA accurately tackles RNA-seq data analysis tasks

As an example, in Figure 2, the user supplied four RNA-Seq samples: two from the LoGlu group (SRR1374921.fastq.gz and SRR1374922.fastq.gz) and two from the HiGlu group (SRR1374923.fastq.gz and SRR1374924.fastq.gz). Additionally, the user furnished the mouse reference genome (mm39.fa) and genome annotation (mm39.ncbiRefSeq.gtf). The primary objective of this case was to identify differentially expressed genes between the two data groups. Using textual inputs only, AutoBA generated a detailed, step-by-step analysis plan during the planning phase, as outlined below:

```
1. First, I will use the 'Trimmomatic'
   software to trim the adapters from the
   single-end mouse rna-seq reads. The
   input files for this step will be './
   examples/case1.1/data/SRR1374921.fastq.
   gz', './examples/case1.1/data/SRR1374922
   .fastq.gz', './examples/case1.1/data/
   SRR1374923.fastq.gz', './examples/case1
   .1/data/SRR1374924.fastq.gz' and the
   adapter file './examples/case1.1/data/
   TruSeq3-SE.fa'.

2. Next, I will use the 'Hisat2' software to
    align the trimmed reads to the mouse
   mm39 genome. The input files for this
   step will be the output files from the
   previous step and the genome file './
   examples/case1.1/data/mm39.fa'.

3. Then, I will use the 'Samtools' software
   to convert the alignment output files
   from Hisat2 (in SAM format) to BAM
   format.

4. After that, I will use the 'HTSeq'
   software to count the number of reads
   mapped to each gene. The input files for
    this step will be the BAM files from
   the previous step and the annotation
   file './examples/case1.1/data/mm39.
   ncbiRefSeq.gtf'.

5. Finally, I will use the 'DESeq2' software
    to identify the differentially
   expressed genes. The input files for
   this step will be the count files from
   the previous step.
```

Within each step of the plan outlined above, AutoBA provides precise instructions regarding the required software, including names like Trimmomatic, Hisat2, Samtools, HTSeq, and DESeq2, along with clear sub-goals for each analytical stage.

During the execution phase, AutoBA generates bash scripts for every step of the plan established in the planning phase. These scripts encompass environment setup, software installation, and tailored code for software utilization. Parameters and data paths specific to the software are meticulously incorporated. The accuracy of AutoBA's analysis procedure and results has undergone independent verification by an expert bioinformatician, confirming its reliability.

3.2 AutoBA adeptly manages similar tasks with robustness

In practice, even when dealing with the same data types, such as RNA-Seq, bioinformatics analyses can exhibit variations primarily driven by differences in the characteristics of input data and analysis objectives. As exemplified in Figures 2 to 4, when performing RNA-Seq analysis, users may have distinct final goals, necessitating adjustments in software and parameter selection during the actual execution.

In comparison to case 1.1, AutoBA introduces a 6th step in case 1.2, tailored for screening top differentially expressed genes to fulfill the user's specific requirements. In case 1.3, AutoBA demonstrates its capability to autonomously devise novel analysis processes based on varying input data, showcasing its adaptability to diverse input data and analysis objectives.

3.3 AutoBA generalizes for multi-omics bioinformatics analysis

To evaluate the robustness of AutoBA, we conducted assessments involving a total of 10 cases spanning four distinct types of omics data: bulk RNA-seq (case 1.1: identifying differentially expressed genes; case 1.2: pinpointing the top 5 down-regulated genes in the HiGlu group; case 1.3: predicting fusion genes), single-cell RNA-seq (case 2.1: detecting differentially expressed genes; case 2.2: performing clustering; case 2.3: identifying the top 5 marker genes), ChIP-Seq (case 3.1: calling peaks; case 3.2: discovering motifs within the peaks; case 3.3: conducting functional enrichment analysis), and spatial transcriptomics (case 4.1: neighborhood enrichment analysis) as shown from Figrues 2 to 11.

Each case underwent an independent analysis process conducted by AutoBA and was subsequently subjected to validation by an expert bioinformatics expert. The collective results underscore the versatility and robustness of AutoBA across a spectrum of multi-omics analysis procedures in the field of bioinformatics.

4 Discussion

To our knowledge, AutoBA is the first and a pioneering autonomous AI agent tailored explicitly for conventional bioinformatics analysis for omics data. AutoBA streamlines the analytical process, requiring minimal user input while providing detailed step-by-step plans for various bioinformatics tasks. The results of our investigation reveal that AutoBA excels in accurately handling a diverse array of omics

TABLE 1
Summary of AutoBA application scenarios in bioinformatics multi-omics analysis.

Bioinformatics Pipelines	Tasks	Types of Omics	Validation Progress
WGS data analysis	Genome assembly	Genomics	ongoing
WGS/WES data analysis	Somatic SNV+indel calling	Genomics	validated
WGS/WES data analysis	Somatic SNV+indel calling and annotation	Genomics	validated
WGS/WES data analysis	Structure variation identification	Genomics	ongoing
ChIP-seq data analysis	Peak calling	Genomics	validated
ChIP-seq data analysis	Motif discovery for binding sites	Genomics	validated
ChIP-seq data analysis	Functional enrichment of target gene	Genomics	validated
Bisulfite-Seq data analysis	Identifying DNA methylation	Genomics	ongoing
ATAC-seq data analysis	Identifying open chromatin regions	Genomics	ongoing
DNase-seq data analysis	Identifying Dnasel hypersensitive site	Genomics	ongoing
4C-seq data analysis	Find genomics interactions	Genomics	ongoing
Nanopore DNA sequencing data analysis	Genome assembly	Genomics	ongoing
Nanopore DNA sequencing data analysis	Tandem repeats variation identification	Genomics	ongoing
PacBio DNA sequencing data analysis	Genome assembly	Genomics	ongoing
RNA-Seq data analysis	Find Differentially expressed genes	Transcriptomics	validated
RNA-Seq data analysis	Find enriched pathways of differentially expressed gene	Transcriptomics	validated
RNA-Seq data analysis	Predict Fusion gene with annotation	Transcriptomics	validated
RNA-Seq data analysis	Isoform expression	Transcriptomics	ongoing
RNA-Seq data analysis	Splicing analysis	Transcriptomics	ongoing
RNA-Seq data analysis	APA analysis	Transcriptomics	ongoing
RNA-Seq data analysis	RNA editing	Transcriptomics	ongoing
RNA-Seq data analysis	Circular RNA identification	Transcriptomics	ongoing
Small RNA sequencing data analysis	microRNA quantification	Transcriptomics	ongoing
Small RNA sequencing data analysis	microRNA prediction	Transcriptomics	ongoing
CAGE-seq data analysis	TSS identification	Transcriptomics	ongoing
3′ end-seq data analysis	PAS (polyadenylation site) identification	Transcriptomics	ongoing
Nanopore RNA sequencing data analysis	Isoform expression	Transcriptomics	ongoing
PacBio RNA sequencing data analysis	Isoform expression	Transcriptomics	ongoing
eCLIP-seq data analysis	Identifying binding site	Transcriptomics	ongoing
eCLIP-seq data analysis	Find enriched binding motif	Transcriptomics	ongoing
SHAPE-seq data analysis	RNA secondary structure identification	Transcriptomics	ongoing
single-cell RNA-seq data analysis	Cell clustering from fastq data	Transcriptomics	ongoing
single-cell RNA-seq data analysis	Find differentially expressed genes based on count matrix	Transcriptomics	validated
single-cell RNA-seq data analysis	Find marker genes based on count matrix	Transcriptomics	validated
single-cell RNA-seq data analysis	Cell clustering and visualization	Transcriptomics	validated
Spatial transcriptomics	Find differentialy expressed genes based on count matrix	Transcriptomics	ongoing
Spatial transcriptomics	Find marker genes based on count matrix	Transcriptomics	ongoing
Spatial transcriptomics	Cell clustering and visulization	Transcriptomics	ongoing
Spatial transcriptomics	Cell-cell communications	Transcriptomics	validated
Spatial transcriptomics	Quantification of metabolites	Transcriptomics	ongoing
Spatial transcriptomics	Single-cell mapping	Transcriptomics	ongoing
Spatial transcriptomics	Cell type deconvolution	Transcriptomics	ongoing
Mass spectrometry data analysis	Protein expression quantification	Proteomics	ongoing
Mass spectrometry data analysis	Identifying protein modification	Proteomics	ongoing
Mass spectrometry data analysis	Quantification of metabolites	Metabolomics	ongoing
Mass spectrometry data analysis	Dimension reduction based on metabolites concentration	Metabolomics	ongoing

analysis tasks, such as RNA-seq, scRNA-seq, ChIP-seq, and spatial transcriptomics, among others. One of the key strengths of AutoBA is its adaptability to variations in analysis objectives. As demonstrated in the cases presented, even with similar data types, such as RNA-Seq, users often have distinct goals, necessitating modifications in software and parameter selection during execution. AutoBA effectively accommodates these variations, allowing users to tailor their analyses to specific research needs without compromising accuracy. Furthermore, AutoBA's versatility is highlighted by its ability to self-design new analysis processes based on differing input data. This autonomous adaptability makes AutoBA a valuable tool for bioinformaticians working on novel or unconventional research questions, as it can adjust its approach to the unique characteristics of the data.

Online bioinformatics analysis platforms are currently in vogue, but they often necessitate the uploading of either raw data or pre-processed statistics by users, which could potentially give rise to privacy concerns and data leakage risks. In contrast, AutoBA offers a local solution that ef-

RNA-Seq Analysis **Case 1.1: Find differentially expressed genes**

User's Inputs

Data and Description

SRR1374921.fastq.gz: single-end mouse rna-seq reads, replicate 1 in LoGlu group
SRR1374922.fastq.gz: single-end mouse rna-seq reads, replicate 2 in LoGlu group
SRR1374923.fastq.gz: single-end mouse rna-seq reads, replicate 1 in HiGlu group
SRR1374924.fastq.gz: single-end mouse rna-seq reads, replicate 2 in HiGlu group
TruSeq3-SE.fa: trimming adapter
mm39.fa: mouse mm39 genome fasta
mm39.ncbiRefSeq.gtf: mouse mm39 genome annotation

Goal

Find the differentially expressed genes

Generated Plans by AutoBA

1. First, I will use the 'Trimmomatic' software to trim the adapters from the single-end mouse rna-seq reads. The input files for this step will be './examples/case1.1/data/SRR1374921.fastq.gz', './examples/case1.1/data/SRR1374922.fastq.gz', './examples/case1.1/data/SRR1374923.fastq.gz', './examples/case1.1/data/SRR1374924.fastq.gz' and the adapter file './examples/case1.1/data/TruSeq3-SE.fa'.
2. Next, I will use the 'Hisat2' software to align the trimmed reads to the mouse mm39 genome. The input files for this step will be the output files from the previous step and the genome file './examples/case1.1/data/mm39.fa'.
3. Then, I will use the 'Samtools' software to convert the alignment output files from Hisat2 (in SAM format) to BAM format.
4. After that, I will use the 'HTSeq' software to count the number of reads mapped to each gene. The input files for this step will be the BAM files from the previous step and the annotation file './examples/case1.1/data/mm39.ncbiRefSeq.gtf'.
5. Finally, I will use the 'DESeq2' software to identify the differentially expressed genes. The input files for this step will be the count files from the previous step.

Generated Codes by AutoBA

```
# Step 1
source activate abc
conda install -y trimmomatic
trimmomatic SE -phred33 ./examples/case1.1/data/SRR1374921.fastq.gz ./examples/case1.1/output/SRR1374921_trimmed.fastq.gz
ILLUMINACLIP:./examples/case1.1/data/TruSeq3-SE.fa:2:30:10
trimmomatic SE -phred33 ./examples/case1.1/data/SRR1374922.fastq.gz ./examples/case1.1/output/SRR1374922_trimmed.fastq.gz
ILLUMINACLIP:./examples/case1.1/data/TruSeq3-SE.fa:2:30:10
trimmomatic SE -phred33 ./examples/case1.1/data/SRR1374923.fastq.gz ./examples/case1.1/output/SRR1374923_trimmed.fastq.gz
ILLUMINACLIP:./examples/case1.1/data/TruSeq3-SE.fa:2:30:10
trimmomatic SE -phred33 ./examples/case1.1/data/SRR1374924.fastq.gz ./examples/case1.1/output/SRR1374924_trimmed.fastq.gz
ILLUMINACLIP:./examples/case1.1/data/TruSeq3-SE.fa:2:30:10

# Step 2
source activate abc
conda install -y hisat2
hisat2 -x ./examples/case1.1/data/mm39.fa -U ./examples/case1.1/output/SRR1374921_trimmed.fastq.gz -S ./examples/case1.1/output/SRR1374921.sam
hisat2 -x ./examples/case1.1/data/mm39.fa -U ./examples/case1.1/output/SRR1374922_trimmed.fastq.gz -S ./examples/case1.1/output/SRR1374922.sam
hisat2 -x ./examples/case1.1/data/mm39.fa -U ./examples/case1.1/output/SRR1374923_trimmed.fastq.gz -S ./examples/case1.1/output/SRR1374923.sam
hisat2 -x ./examples/case1.1/data/mm39.fa -U ./examples/case1.1/output/SRR1374924_trimmed.fastq.gz -S ./examples/case1.1/output/SRR1374924.sam

# Step 3
source activate abc
conda install -y samtools
samtools view -S -b ./examples/case1.1/output/SRR1374921.sam > ./examples/case1.1/output/SRR1374921.bam
samtools view -S -b ./examples/case1.1/output/SRR1374922.sam > ./examples/case1.1/output/SRR1374922.bam
samtools view -S -b ./examples/case1.1/output/SRR1374923.sam > ./examples/case1.1/output/SRR1374923.bam
samtools view -S -b ./examples/case1.1/output/SRR1374924.sam > ./examples/case1.1/output/SRR1374924.bam

# Step 4
source activate abc
conda install -y htseq
htseq-count -f bam -s no ./examples/case1.1/output/SRR1374921.bam ./examples/case1.1/data/mm39.ncbiRefSeq.gtf
> ./examples/case1.1/output/SRR1374921.count
htseq-count -f bam -s no ./examples/case1.1/output/SRR1374922.bam ./examples/case1.1/data/mm39.ncbiRefSeq.gtf
> ./examples/case1.1/output/SRR1374922.count
htseq-count -f bam -s no ./examples/case1.1/output/SRR1374923.bam ./examples/case1.1/data/mm39.ncbiRefSeq.gtf
> ./examples/case1.1/output/SRR1374923.count
htseq-count -f bam -s no ./examples/case1.1/output/SRR1374924.bam ./examples/case1.1/data/mm39.ncbiRefSeq.gtf
> ./examples/case1.1/output/SRR1374924.count

# Step 5
source activate abc
conda install -y bioconductor-deseq2
Rscript -e "library('DESeq2'); countData <- read.table('./examples/case1.1/output/SRR1374921.count', './examples/case1.1/output/SRR1374922.count',
'./examples/case1.1/output/SRR1374923.count', './examples/case1.1/output/SRR1374924.count', header=TRUE, row.names=1); colData <-
DataFrame(condition=factor(c('LoGlu', 'LoGlu', 'HiGlu', 'HiGlu'))); dds <- DESeqDataSetFromMatrix(countData=countData, colData=colData, design=~condition);
dds <- DESeq(dds); res <- results(dds); write.csv(as.data.frame(res), file='./examples/case1.1/output/differential_expression_results.csv')"
```

Fig. 2. **Case 1.1. Find differentially expressed genes.** In this case, the user supplied four RNA-seq datasets, comprising two from the LoGlu group and two from the HiGlu group. The primary objective of this analysis was to identify differentially expressed genes across the two datasets.

fectively addresses these privacy issues. Moreover, AutoBA showcases its adaptability in sync with emerging bioinformatics tools, with LLM seamlessly incorporating these latest tools into the database. Furthermore, AutoBA is inclined towards selecting the most popular analytical frameworks or widely applicable tools in the planning phase, underscoring its robustness. Another distinguishing feature is AutoBA's transparent and interpretable execution process. This transparency allows professional bioinformaticians to easily modify and customize AutoBA's outputs, leveraging AutoBA to expedite the data analysis process.

Given that classical bioinformatic analysis encompasses a far broader spectrum of tasks and challenges than the 10 cases studied in this work, it is essential to conduct additional real-world applications to further comprehensively validate the robustness of AutoBA as shown in Table 1. Furthermore, taking into account the timeliness of the training data used for large language models, it's important to note that some of the most recently proposed methods in the field of bioinformatics may still pose challenges in automatically generating code by AutoBA. Therefore, a future endeavor to train a real-time large language model explicitly tailored for bioinformatics can significantly enhance AutoBA's ability to maintain up-to-date code generation capabilities. Nevertheless, AutoBA represents a significant advancement in the field of bioinformatics, offering a user-friendly, efficient, and adaptable solution for a wide range of omics analysis tasks. Its capacity to handle diverse data types and analysis goals, coupled with its robustness and adaptability, positions AutoBA as a valuable asset in the pursuit of accelerating bioinformatics research. We anticipate that AutoBA will find extensive utility in the scientific community, supporting researchers in their quest to extract meaningful insights from complex biological data.

5 Data availability

RNA-seq: The dataset for case 1.1 and case 1.2 could be downloaded with IDs: SRR1374921, SRR1374922, SRR1374923, and SRR1374924. The dataset for case 1.3 could be downloaded from https://github.com/STAR-Fusion/STAR-Fusion-Tutorial/wiki.

scRNA-seq: The dataset for case 2.1 to 2.3 could be downloaded from http://cf.10xgenomics.com/samples/cell-exp/1.1.0/pbmc3k/pbmc3k_filtered_gene_bc_matrices.tar.gz.

ChIP-seq: The dataset for case 3.1 to 3.3 could be downloaded with IDs: SRR620204, SRR620205, SRR620206, and SRR620208.

Spatial Transcriptomics: The dataset for case 4.1 could downloaded from https://doi.org/10.5281/zenodo.6334774.

6 Code availability

The AutoBA software is publicly available at https://github.com/JoshuaChou2018/AutoBA.

7 Acknowledgements

Juexiao Zhou, Bin Zhang, Xiuying Chen, Haoyang Li, Xiaopeng Xu, Siyuan Chen and Xin Gao were supported in part by grants from the Office of Research Administration (ORA) at King Abdullah University of Science and Technology (KAUST) under award number FCC/1/1976-44-01, FCC/1/1976-45-01, REI/1/5202-01-01, REI/1/5234-01-01, REI/1/4940-01-01, RGC/3/4816-01-01, and REI/1/0018-01-01.

8 Author contributions statement

Conceptualization: J.Z. and X.G. Design: J.Z., B.Z. and X.G. Code implementation: J.Z. Application: J.Z., B.Z., X.C., H.L. Drafting of the manuscript: J.Z. and B.Z. Critical revision of the manuscript for important intellectual content: J.Z., B.Z., X.X., S.C., X.G. Supervision: J.Z. and X.G. Funding acquisition: X.G.

9 Competing Interests

The authors have declared no competing interests.

References

[1] N. M. Luscombe, D. Greenbaum, and M. Gerstein, "What is bioinformatics? a proposed definition and overview of the field," *Methods of information in medicine*, vol. 40, no. 04, pp. 346–358, 2001.

[2] J. Gauthier, A. T. Vincent, S. J. Charette, and N. Derome, "A brief history of bioinformatics," *Briefings in bioinformatics*, vol. 20, no. 6, pp. 1981–1996, 2019.

[3] A. D. Baxevanis, G. D. Bader, and D. S. Wishart, *Bioinformatics*. John Wiley & Sons, 2020.

[4] P. Munk, C. Brinch, F. D. Møller, T. N. Petersen, R. S. Hendriksen, A. M. Seyfarth, J. S. Kjeldgaard, C. A. Svendsen, B. Van Bunnik, F. Berglund *et al.*, "Genomic analysis of sewage from 101 countries reveals global landscape of antimicrobial resistance," *Nature Communications*, vol. 13, no. 1, p. 7251, 2022.

[5] F. Hemmerling and J. Piel, "Strategies to access biosynthetic novelty in bacterial genomes for drug discovery," *Nature Reviews Drug Discovery*, vol. 21, no. 5, pp. 359–378, 2022.

[6] E. H. Lips, T. Kumar, A. Megalios, L. L. Visser, M. Sheinman, A. Fortunato, V. Shah, M. Hoogstraat, E. Sei, D. Mallo *et al.*, "Genomic analysis defines clonal relationships of ductal carcinoma in situ and recurrent invasive breast cancer," *Nature genetics*, vol. 54, no. 6, pp. 850–860, 2022.

[7] G. Orlando, D. Raimondi, R. Duran-Romaña, Y. Moreau, J. Schymkowitz, and F. Rousseau, "Pyuul provides an interface between biological structures and deep learning algorithms," *Nature communications*, vol. 13, no. 1, p. 961, 2022.

[8] D. T. Jones and J. M. Thornton, "The impact of alphafold2 one year on," *Nature methods*, vol. 19, no. 1, pp. 15–20, 2022.

[9] M. L. Hekkelman, I. de Vries, R. P. Joosten, and A. Perrakis, "Alphafill: enriching alphafold models with ligands and cofactors," *Nature Methods*, vol. 20, no. 2, pp. 205–213, 2023.

[10] N. Sapoval, A. Aghazadeh, M. G. Nute, D. A. Antunes, A. Balaji, R. Baraniuk, C. Barberan, R. Dannenfelser, C. Dun, M. Edrisi *et al.*, "Current progress and open challenges for applying deep learning across the biosciences," *Nature Communications*, vol. 13, no. 1, p. 1728, 2022.

[11] T. Gupta, M. Zaki, and N. A. Krishnan, "Matscibert: A materials domain language model for text mining and information extraction," *npj Computational Materials*, vol. 8, no. 1, p. 102, 2022.

[12] A. Santos, A. R. Colaço, A. B. Nielsen, L. Niu, M. Strauss, P. E. Geyer, F. Coscia, N. J. W. Albrechtsen, F. Mundt, L. J. Jensen *et al.*, "A knowledge graph to interpret clinical proteomics data," *Nature Biotechnology*, vol. 40, no. 5, pp. 692–702, 2022.

[13] Z. Zeng, Y. Yao, Z. Liu, and M. Sun, "A deep-learning system bridging molecule structure and biomedical text with comprehension comparable to human professionals," *Nature communications*, vol. 13, no. 1, p. 862, 2022.

[14] S. W. Attwood, S. C. Hill, D. M. Aanensen, T. R. Connor, and O. G. Pybus, "Phylogenetic and phylodynamic approaches to understanding and combating the early sars-cov-2 pandemic," *Nature Reviews Genetics*, vol. 23, no. 9, pp. 547–562, 2022.

[15] N. De Maio, P. Kalaghatgi, Y. Turakhia, R. Corbett-Detig, B. Q. Minh, and N. Goldman, "Maximum likelihood pandemic-scale phylogenetics," *Nature Genetics*, pp. 1–7, 2023.

[16] A. S. Chanderbali, L. Jin, Q. Xu, Y. Zhang, J. Zhang, S. Jian, E. Carroll, D. Sankoff, V. A. Albert, D. G. Howarth *et al.*, "Buxus and tetracentron genomes help resolve eudicot genome history," *Nature communications*, vol. 13, no. 1, p. 643, 2022.

[17] J. Rhodes, A. Abdolrasouli, K. Dunne, T. R. Sewell, Y. Zhang, E. Ballard, A. P. Brackin, N. van Rhijn, H. Chown, A. Tsitsopoulou *et al.*, "Population genomics confirms acquisition of drug-resistant aspergillus fumigatus infection by humans from the environment," *Nature microbiology*, vol. 7, no. 5, pp. 663–674, 2022.

[18] R. J. Rockett, J. Draper, M. Gall, E. M. Sim, A. Arnott, J. E. Agius, J. Johnson-Mackinnon, W. Fong, E. Martinez, A. P. Drew *et al.*, "Co-infection with sars-cov-2 omicron and delta variants revealed by genomic surveillance," *Nature communications*, vol. 13, no. 1, p. 2745, 2022.

[19] A. Heinken, J. Hertel, G. Acharya, D. A. Ravcheev, M. Nyga, O. E. Okpala, M. Hogan, S. Magnúsdóttir, F. Martinelli, B. Nap *et al.*, "Genome-scale metabolic reconstruction of 7,302 human microorganisms for personalized medicine," *Nature Biotechnology*, pp. 1–12, 2023.

[20] J. Zhou, B. Zhang, H. Li, L. Zhou, Z. Li, Y. Long, W. Han, M. Wang, H. Cui, J. Li *et al.*, "Annotating tsss in multiple cell types based on dna sequence and rna-seq data via deerect-tss," *Genomics, Proteomics & Bioinformatics*, vol. 20, no. 5, pp. 959–973, 2022.

[21] H. Li, H. Li, J. Zhou, and X. Gao, "Sd2: spatially resolved transcriptomics deconvolution through integration of dropout and spatial information," *Bioinformatics*, vol. 38, no. 21, pp. 4878–4884, 2022.

[22] T. Zhang, L. Li, H. Sun, D. Xu, and G. Wang, "Deepicsh: a complex deep learning framework for identifying cell-specific silencers and their strength from the human genome," *Briefings in Bioinformatics*, p. bbad316, 2023.

[23] Z. Li, E. Gao, J. Zhou, W. Han, X. Xu, and X. Gao, "Applications of deep learning in understanding gene regulation," *Cell Reports Methods*, 2023.

[24] Y. Long, B. Zhang, S. Tian, J. J. Chan, J. Zhou, Z. Li, Y. Li, Z. An, X. Liao, Y. Wang *et al.*, "Accurate transcriptome-wide identification and quantification of alternative polyadenylation from rna-seq data with apaiq," *Genome Research*, vol. 33, no. 4, pp. 644–657, 2023.

[25] A. F. Bardet, Q. He, J. Zeitlinger, and A. Stark, "A computational pipeline for comparative chip-seq analyses," *Nature protocols*, vol. 7, no. 1, pp. 45–61, 2012.

[26] B. Vieth, S. Parekh, C. Ziegenhain, W. Enard, and I. Hellmann, "A systematic evaluation of single cell rna-seq analysis pipelines," *Nature communications*, vol. 10, no. 1, p. 4667, 2019.

[27] M. D. Luecken and F. J. Theis, "Current best practices in single-cell rna-seq analysis: a tutorial," *Molecular systems biology*, vol. 15, no. 6, p. e8746, 2019.

[28] F. C. Grandi, H. Modi, L. Kampman, and M. R. Corces, "Chromatin accessibility profiling by atac-seq," *Nature protocols*, vol. 17, no. 6, pp. 1518–1552, 2022.

[29] P. C. Ng and E. F. Kirkness, "Whole genome sequencing," *Genetic variation: Methods and protocols*, pp. 215–226, 2010.

[30] Z. Wang, M. Gerstein, and M. Snyder, "Rna-seq: a revolutionary tool for transcriptomics," *Nature reviews genetics*, vol. 10, no. 1, pp. 57–63, 2009.

[31] A.-E. Saliba, A. J. Westermann, S. A. Gorski, and J. Vogel, "Single-cell rna-seq: advances and future challenges," *Nucleic acids research*, vol. 42, no. 14, pp. 8845–8860, 2014.

[32] J. D. Buenrostro, B. Wu, H. Y. Chang, and W. J. Greenleaf, "Atac-seq: a method for assaying chromatin accessibility genome-wide," *Current protocols in molecular biology*, vol. 109, no. 1, pp. 21–29, 2015.

[33] P. J. Park, "Chip–seq: advantages and challenges of a maturing technology," *Nature reviews genetics*, vol. 10, no. 10, pp. 669–680, 2009.

[34] D. J. Burgess, "Spatial transcriptomics coming of age," *Nature Reviews Genetics*, vol. 20, no. 6, pp. 317–317, 2019.

[35] A. Conesa, P. Madrigal, S. Tarazona, D. Gomez-Cabrero, A. Cervera, A. McPherson, M. W. Szcześniak, D. J. Gaffney, L. L. Elo, X. Zhang *et al.*, "A survey of best practices for rna-seq data analysis," *Genome biology*, vol. 17, no. 1, pp. 1–19, 2016.

[36] L. Wang, S. Wang, and W. Li, "Rseqc: quality control of rna-seq experiments," *Bioinformatics*, vol. 28, no. 16, pp. 2184–2185, 2012.

[37] M. Martin, "Cutadapt removes adapter sequences from high-throughput sequencing reads," *EMBnet. journal*, vol. 17, no. 1, pp. 10–12, 2011.

[38] A. Dobin and T. R. Gingeras, "Mapping rna-seq reads with star," *Current protocols in bioinformatics*, vol. 51, no. 1, pp. 11–14, 2015.

[39] C. Trapnell, L. Pachter, and S. L. Salzberg, "Tophat: discovering splice junctions with rna-seq," *Bioinformatics*, vol. 25, no. 9, pp. 1105–1111, 2009.

[40] Y. Liao, G. K. Smyth, and W. Shi, "featurecounts: an efficient general purpose program for assigning sequence reads to genomic features," *Bioinformatics*, vol. 30, no. 7, pp. 923–930, 2014.

[41] F. Rapaport, R. Khanin, Y. Liang, M. Pirun, A. Krek, P. Zumbo, C. E. Mason, N. D. Socci, and D. Betel, "Comprehensive evaluation of differential gene expression analysis methods for rna-seq data," *Genome biology*, vol. 14, no. 9, pp. 1–13, 2013.

[42] S. Shen, J. W. Park, Z.-x. Lu, L. Lin, M. D. Henry, Y. N. Wu, Q. Zhou, and Y. Xing, "rmats: robust and flexible detection of differential alternative splicing from replicate rna-seq data," *Proceedings of the National Academy of Sciences*, vol. 111, no. 51, pp. E5593–E5601, 2014.

[43] Y. Katz, E. T. Wang, E. M. Airoldi, and C. B. Burge, "Analysis and design of rna sequencing experiments for identifying isoform regulation," *Nature methods*, vol. 7, no. 12, pp. 1009–1015, 2010.

[44] X. Wang and M. J. Cairns, "Gene set enrichment analysis of rna-seq data: integrating differential expression and splicing," in *BMC bioinformatics*, vol. 14, no. 5. BioMed Central, 2013, pp. 1–10.

[45] R. Thomas, S. Thomas, A. K. Holloway, and K. S. Pollard, "Features that define the best chip-seq peak calling algorithms," *Briefings in bioinformatics*, vol. 18, no. 3, pp. 441–450, 2017.

[46] T. L. Bailey, "Dreme: motif discovery in transcription factor chip-seq data," *Bioinformatics*, vol. 27, no. 12, pp. 1653–1659, 2011.

[47] G. Yu, L.-G. Wang, and Q.-Y. He, "Chipseeker: an r/bioconductor package for chip peak annotation, comparison and visualization," *Bioinformatics*, vol. 31, no. 14, pp. 2382–2383, 2015.

[48] S. Roy, C. Coldren, A. Karunamurthy, N. S. Kip, E. W. Klee, S. E. Lincoln, A. Leon, M. Pullambhatla, R. L. Temple-Smolkin, K. V. Voelkerding *et al.*, "Standards and guidelines for validating next-generation sequencing bioinformatics pipelines: a joint recommendation of the association for molecular pathology and the college of american pathologists," *The Journal of Molecular Diagnostics*, vol. 20, no. 1, pp. 4–27, 2018.

[49] P. A. Ewels, A. Peltzer, S. Fillinger, H. Patel, J. Alneberg, A. Wilm, M. U. Garcia, P. Di Tommaso, and S. Nahnsen, "The nf-core framework for community-curated bioinformatics pipelines," *Nature biotechnology*, vol. 38, no. 3, pp. 276–278, 2020.

[50] L. Wratten, A. Wilm, and J. Göke, "Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers," *Nature methods*, vol. 18, no. 10, pp. 1161–1168, 2021.

[51] E. B. Işık, M. D. Brazas, R. Schwartz, B. Gaeta, P. M. Palagi, C. W. van Gelder, P. Suravajhala, H. Singh, S. L. Morgan, H. Zahroh *et al.*, "Grand challenges in bioinformatics education and training," *Nature Biotechnology*, vol. 41, no. 8, pp. 1171–1174, 2023.

[52] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler *et al.*, "Emergent abilities of large language models," *arXiv preprint arXiv:2206.07682*, 2022.

[53] A. J. Thirunavukarasu, D. S. J. Ting, K. Elangovan, L. Gutierrez, T. F. Tan, and D. S. W. Ting, "Large language models in medicine," *Nature Medicine*, pp. 1–11, 2023.

[54] A. Madani, B. Krause, E. R. Greene, S. Subramanian, B. P. Mohr, J. M. Holton, J. L. Olmos Jr, C. Xiong, Z. Z. Sun, R. Socher *et al.*, "Large language models generate functional protein sequences across diverse families," *Nature Biotechnology*, pp. 1–8, 2023.

[55] B. Meskó and E. J. Topol, "The imperative for regulatory oversight of large language models (or generative ai) in healthcare," *npj Digital Medicine*, vol. 6, no. 1, p. 120, 2023.

[56] S. Wang, Z. Zhao, X. Ouyang, Q. Wang, and D. Shen, "Chatcad: Interactive computer-aided diagnosis on medical image using large language models," *arXiv preprint arXiv:2302.07257*, 2023.

[57] J. Zhou, X. He, L. Sun, J. Xu, X. Chen, Y. Chu, L. Zhou, X. Liao, B. Zhang, and X. Gao, "Skingpt-4: An interactive dermatology diagnostic system with visual large language model," *medRxiv*, pp. 2023–06, 2023.

[58] J. Zhou, X. Chen, and X. Gao, "Path to medical agi: Unify domain-specific medical llms with the lowest cost," *arXiv preprint arXiv:2306.10765*, 2023.

[59] T. Tu, S. Azizi, D. Driess, M. Schaekermann, M. Amin, P.-C. Chang, A. Carroll, C. Lau, R. Tanno, I. Ktena *et al.*, "Towards generalist biomedical ai," *arXiv preprint arXiv:2307.14334*, 2023.

[60] D. Flam-Shepherd, K. Zhu, and A. Aspuru-Guzik, "Language models can learn complex molecular distributions," *Nature Communications*, vol. 13, no. 1, p. 3293, 2022.

[61] E. Shue, L. Liu, B. Li, Z. Feng, X. Li, and G. Hu, "Empowering beginners in bioinformatics with chatgpt," *bioRxiv*, pp. 2023–03, 2023.

[62] S. R. Piccolo, P. Denny, A. Luxton-Reilly, S. Payne, and P. G. Ridge, "Many bioinformatics programming tasks can be automated with chatgpt," *arXiv preprint arXiv:2303.13528*, 2023.

[63] S. Gravitas, "Auto-gpt: An autonomous gpt-4 experiment," 2023.

RNA-Seq Analysis **Case 1.2: Identify top5 down-regulated genes in HiGlu group**

User's Inputs

Data and Description

SRR1374921.fastq.gz: single-end mouse rna-seq reads, replicate 1 in LoGlu group
SRR1374922.fastq.gz: single-end mouse rna-seq reads, replicate 2 in LoGlu group
SRR1374923.fastq.gz: single-end mouse rna-seq reads, replicate 1 in HiGlu group
SRR1374924.fastq.gz: single-end mouse rna-seq reads, replicate 2 in HiGlu group
TruSeq3-SE.fa: trimming adapter
mm39.fa: mouse mm39 genome fasta
mm39.ncbiRefSeq.gtf: mouse mm39 genome annotation

Goal

Identify top5 down-regulated genes in HiGlu group

Generated Plans by AutoBA

1. First, we will use the 'Trimmomatic' software to trim the adapters from the RNA-seq reads. The input for this step will be the RNA-seq reads files ('./examples/case1.2/data/SRR1374921.fastq.gz', './examples/case1.2/data/SRR1374922.fastq.gz', './examples/case1.2/data/SRR1374923.fastq.gz', './examples/case1.2/data/SRR1374924.fastq.gz') and the trimming adapter file ('./examples/case1.2/data/TruSeq3-SE.fa').
2. Next, we will use the 'Hisat2' software to align the trimmed reads to the mouse mm39 genome. The input for this step will be the trimmed reads from the previous step and the mouse mm39 genome fasta file ('./examples/case1.2/data/mm39.fa').
3. After alignment, we will use the 'Samtools' software to convert the alignment output to a sorted BAM file. The input for this step will be the alignment output from the previous step.
4. Then, we will use the 'HTSeq' software to count the number of reads mapped to each gene. The input for this step will be the sorted BAM file from the previous step and the mouse mm39 genome annotation file ('./examples/case1.2/data/mm39.ncbiRefSeq.gtf').
5. After obtaining the read counts, we will use the 'DESeq2' software to perform differential gene expression analysis between the LoGlu and HiGlu groups. The input for this step will be the read counts from the previous step.
6. Finally, we will use the 'pheatmap' software to visualize the top 5 down-regulated genes in the HiGlu group. The input for this step will be the differential gene expression results from the previous step.

Generated Codes by AutoBA

```
# Step 1
source activate abc
conda install -y trimmomatic
trimmomatic SE -phred33 ./examples/case1.2/data/SRR1374921.fastq.gz ./examples/case1.2/output/SRR1374921_trimmed.fastq.gz
ILLUMINACLIP:./examples/case1.2/data/TruSeq3-SE.fa:2:30:10
trimmomatic SE -phred33 ./examples/case1.2/data/SRR1374922.fastq.gz ./examples/case1.2/output/SRR1374922_trimmed.fastq.gz
ILLUMINACLIP:./examples/case1.2/data/TruSeq3-SE.fa:2:30:10
trimmomatic SE -phred33 ./examples/case1.2/data/SRR1374923.fastq.gz ./examples/case1.2/output/SRR1374923_trimmed.fastq.gz
ILLUMINACLIP:./examples/case1.2/data/TruSeq3-SE.fa:2:30:10
trimmomatic SE -phred33 ./examples/case1.2/data/SRR1374924.fastq.gz ./examples/case1.2/output/SRR1374924_trimmed.fastq.gz
ILLUMINACLIP:./examples/case1.2/data/TruSeq3-SE.fa:2:30:10

# Step 2
source activate abc
conda install -y hisat2
hisat2 -x ./examples/case1.2/data/mm39.fa -U ./examples/case1.2/output/SRR1374921_trimmed.fastq.gz -S ./examples/case1.2/output/SRR1374921.sam
hisat2 -x ./examples/case1.2/data/mm39.fa -U ./examples/case1.2/output/SRR1374922_trimmed.fastq.gz -S ./examples/case1.2/output/SRR1374922.sam
hisat2 -x ./examples/case1.2/data/mm39.fa -U ./examples/case1.2/output/SRR1374923_trimmed.fastq.gz -S ./examples/case1.2/output/SRR1374923.sam
hisat2 -x ./examples/case1.2/data/mm39.fa -U ./examples/case1.2/output/SRR1374924_trimmed.fastq.gz -S ./examples/case1.2/output/SRR1374924.sam

# Step 3
source activate abc
conda install -y samtools
samtools view -S -b ./examples/case1.2/output/SRR1374921.sam > ./examples/case1.2/output/SRR1374921.bam
samtools sort ./examples/case1.2/output/SRR1374921.bam -o ./examples/case1.2/output/SRR1374921_sorted.bam
samtools view -S -b ./examples/case1.2/output/SRR1374922.sam > ./examples/case1.2/output/SRR1374922.bam
samtools sort ./examples/case1.2/output/SRR1374922.bam -o ./examples/case1.2/output/SRR1374922_sorted.bam
samtools view -S -b ./examples/case1.2/output/SRR1374923.sam > ./examples/case1.2/output/SRR1374923.bam
samtools sort ./examples/case1.2/output/SRR1374923.bam -o ./examples/case1.2/output/SRR1374923_sorted.bam
samtools view -S -b ./examples/case1.2/output/SRR1374924.sam > ./examples/case1.2/output/SRR1374924.bam
samtools sort ./examples/case1.2/output/SRR1374924.bam -o ./examples/case1.2/output/SRR1374924_sorted.bam

# Step 4
source activate abc
conda install -y htseq
htseq-count -f bam -s no ./examples/case1.2/output/SRR1374921_sorted.bam ./examples/case1.2/data/mm39.ncbiRefSeq.gtf
> ./examples/case1.2/output/SRR1374921_counts.txt
htseq-count -f bam -s no ./examples/case1.2/output/SRR1374922_sorted.bam ./examples/case1.2/data/mm39.ncbiRefSeq.gtf
> ./examples/case1.2/output/SRR1374922_counts.txt
htseq-count -f bam -s no ./examples/case1.2/output/SRR1374923_sorted.bam ./examples/case1.2/data/mm39.ncbiRefSeq.gtf
> ./examples/case1.2/output/SRR1374923_counts.txt
htseq-count -f bam -s no ./examples/case1.2/output/SRR1374924_sorted.bam ./examples/case1.2/data/mm39.ncbiRefSeq.gtf
> ./examples/case1.2/output/SRR1374924_counts.txt

# Step 5
source activate abc
conda install -y -c bioconda bioconductor-deseq2
Rscript -e "library('DESeq2'); counts <- read.table('./examples/case1.2/output/SRR1374921_counts.txt', './examples/case1.2/output/SRR1374922_counts.txt',
'./examples/case1.2/output/SRR1374923_counts.txt', './examples/case1.2/output/SRR1374924_counts.txt', header=TRUE, row.names=1); colData <-
DataFrame(condition=factor(c('LoGlu', 'LoGlu', 'HiGlu', 'HiGlu'))); dds <- DESeqDataSetFromMatrix(countData=counts, colData=colData, design=~condition); dds <-
DESeq(dds); res <- results(dds); write.csv(res, './examples/case1.2/output/differential_expression_results.csv')"

# Step 6
source activate abc
conda install -y -c conda-forge r-pheatmap
Rscript -e "library('pheatmap'); library('DESeq2'); res <- read.csv('./examples/case1.2/output/differential_expression_results.csv', row.names=1); res_ordered <-
res[order(res$log2FoldChange),]; top5_downregulated <- head(res_ordered, 5); pheatmap(top5_downregulated)"
```

Fig. 3. **Case 1.2. Identify top5 down-regulated genes in HiGlu group.** In this case, the user supplied four RNA-seq datasets, comprising two from the LoGlu group and two from the HiGlu group. The primary objective of this analysis was to identify top5 down-regulated genes in HiGlu group.

RNA-Seq Analysis **Case 1.3: Predict Fusion genes**

User's Inputs

Data and Description
rnaseq_1.fastq.gz: RNA-Seq read 1 data (left read)
rnaseq_2.fastq.gz: RNA-Seq read 2 data (right read)
CTAT_HumanFusionLib.mini.dat.gz: a small fusion annotation library
minigenome.fa: small genome sequence consisting of ~750 genes
minigenome.gtf: transcript structure annotations for these genes

Goal
Predict Fusion genes

Generated Plans by AutoBA

1. Load the RNA-Seq read 1 data (left read) from './examples/case1.3/data/rnaseq_1.fastq.gz
2. Load the RNA-Seq read 2 data (right read) from './examples/case1.3/data/rnaseq_2.fastq.gz
3. Load the small fusion annotation library from './examples/case1.3/data/CTAT_HumanFusionLib.mini.dat.gz'
4. Load the small genome sequence consisting of ~750 genes from './examples/case1.3/data/minigenome.fa'.
5. Load the transcript structure annotations for these genes from './examples/case1.3/data/minigenome.gtf'.
6. Use a software alternative to STAR and java, such as HISAT2, to align the RNA-Seq reads to the small genome sequence.
7. Use the aligned reads and the fusion annotation library to predict fusion genes. This can be done using a software like FusionCatcher.
8. Validate the predicted fusion genes using the transcript structure annotations.
9. Document the results and the process for future reference and reproducibility.

Generated Codes by AutoBA

```
# Step 1-5
source activate abc
conda install -y gunzip
gunzip -c ./examples/case1.3/data/rnaseq_1.fastq.gz > ./examples/case1.3/data/rnaseq_1.fastq
gunzip -c ./examples/case1.3/data/rnaseq_2.fastq.gz > ./examples/case1.3/data/rnaseq_2.fastq
gunzip -c ./examples/case1.3/data/CTAT_HumanFusionLib.mini.dat.gz > ./examples/case1.3/data/CTAT_HumanFusionLib.mini.dat
gunzip -c ./examples/case1.3/data/minigenome.fa > ./examples/case1.3/data/minigenome

# Step 6
source activate abc
conda install -y hisat2
hisat2-build ./examples/case1.3/data/minigenome ./examples/case1.3/data/minigenome_index
hisat2 -x ./examples/case1.3/data/minigenome_index -1 ./examples/case1.3/data/rnaseq_1.fastq -2 ./examples/case1.3/data/rnaseq_2.fastq -
S ./examples/case1.3/output/aligned.sam

# Step 7
source activate abc
conda install -y fusioncatcher
fusioncatcher -d ./examples/case1.3/data/CTAT_HumanFusionLib.mini.dat -i ./examples/case1.3/output/aligned.sam -
o ./examples/case1.3/output/fusioncatcher_out

# Step 8
source activate abc
conda install -y gffcompare
gffcompare -r ./examples/case1.3/data/minigenome.gtf -o ./examples/case1.3/output/gffcompare_out ./examples/case1.3/output/fusioncatcher_out/final-
list_candidate-fusion-genes.GRCh37.gff3

# Step 9
source activate abc
echo 'The results and the process for future reference and reproducibility are documented in the output directory: ./examples/case1.3/output'
```

Fig. 4. **Case 1.3. Predict Fusion genes.** In this case, the user supplied one RNA-seq dataset, and a small fusion annotation library. The primary objective of this analysis was to predict fusion genes.

scRNA-Seq Analysis **Case 2.1: Find the differentially expressed genes**

User's Inputs

Data and Description
./examples/case2.1/data/filtered_gene_bc_matrices/hg19: path to 10x mtx data

Goal
Use scanpy to find the differentially expressed genes

Generated Plans by AutoBA

1. Load the 10x mtx data using the Scanpy's read_10x_mtx function. The file path is './examples/case2.1/data/filtered_gene_bc_matrices/hg19'.
2. Preprocess the data using the Scanpy's pp.filter_cells and pp.filter_genes functions to remove cells and genes with too few counts.
3. Normalize the data using the Scanpy's pp.normalize_total function.
4. Logarithmize the data using the Scanpy's pp.log1p function.
5. Identify highly-variable genes using the Scanpy's pp.highly_variable_genes function.
6. Scale the data to unit variance and zero mean using the Scanpy's pp.scale function.
7. Run PCA using the Scanpy's tl.pca function to reduce the dimensionality of the data.
8. Compute the neighborhood graph using the Scanpy's pp.neighbors function.
9. Run UMAP using the Scanpy's tl.umap function for visualization.
10. Cluster the cells using the Scanpy's tl.leiden function.
11. Find marker genes for each cluster using the Scanpy's tl.rank_genes_groups function.
12. Finally, identify the differentially expressed genes using the results from the rank_genes_groups function.

Generated Codes by AutoBA

```
# Step 1
#!/bin/bash

# Activate the conda environment
source activate abc

# Install scanpy with conda
conda install -y scanpy

# Load the 10x mtx data using the Scanpy's read_10x_mtx function
python -c "import scanpy as sc; adata = sc.read_10x_mtx('./examples/case2.1/data/filtered_gene_bc_matrices/hg19')"

# Step 2-12
#!/bin/bash

# Activate the conda environment
source activate abc

# Identify the differentially expressed genes using the results from the rank_genes_groups function
python -c "import scanpy as sc; adata = sc.read_10x_mtx('./examples/case2.1/data/filtered_gene_bc_matrices/hg19'); sc.pp.filter_cells(adata, min_genes=200);
sc.pp.filter_genes(adata, min_cells=3); sc.pp.normalize_total(adata, target_sum=1e4); sc.pp.log1p(adata); sc.pp.highly_variable_genes(adata); sc.pp.scale(adata);
sc.tl.pca(adata); sc.pp.neighbors(adata); sc.tl.umap(adata); sc.tl.leiden(adata); sc.tl.rank_genes_groups(adata, 'leiden');
print(adata.uns['rank_genes_groups']['names'])"
```

Fig. 5. **Case 2.1. Find the differentially expressed genes.** In this case, the user supplied one 10x mtx data. The primary objective of this analysis was to find the differentially expressed genes.

scRNA-Seq Analysis **Case 2.2: Clustering and Visualization**

User's Inputs

Data and Description
./examples/case2.1/data/filtered_gene_bc_matrices/hg19: path to 10x mtx data

Goal
Use scanpy to perform clustering and visualize the expression level of gene PPBP in the UMAP.

Generated Plans by AutoBA

1. Load the 10x mtx data from the provided file path './examples/case2.2/data/filtered_gene_bc_matrices/hg19' using the scanpy.read_10x_mtx function.
2. Normalize the data using the scanpy.pp.normalize_total function.
3. Logarithmize the data using the scanpy.pp.log1p function.
4. Identify highly-variable genes using the scanpy.pp.highly_variable_genes function.
5. Scale the data to unit variance and zero mean using the scanpy.pp.scale function.
6. Run PCA to reduce dimensionality using the scanpy.tl.pca function.
7. Compute the neighborhood graph using the scanpy.pp.neighbors function.
8. Embed the neighborhood graph using UMAP via the scanpy.tl.umap function.
9. Cluster cells using the Leiden algorithm via the scanpy.tl.leiden function.
10. Find marker genes for each cluster using the scanpy.tl.rank_genes_groups function.
11. Visualize the expression level of gene PPBP in the UMAP using the scanpy.pl.umap function.

Generated Codes by AutoBA

```
# Step 1
# Activate the conda environment
source activate abc

# Install scanpy
conda install -y scanpy

# Load the 10x mtx data
python -c "import scanpy as sc; adata = sc.read_10x_mtx('./examples/case2.2/data/filtered_gene_bc_matrices/hg19')"

# Step 2-11
source activate abc

# Visualize the expression level of gene PPBP in the UMAP
python -c "import scanpy as sc; adata = sc.read_10x_mtx('./examples/case2.2/data/filtered_gene_bc_matrices/hg19'); sc.pp.normalize_total(adata);
sc.pp.log1p(adata); sc.pp.highly_variable_genes(adata); sc.pp.scale(adata); sc.tl.pca(adata); sc.pp.neighbors(adata); sc.tl.umap(adata); sc.tl.leiden(adata);
sc.tl.rank_genes_groups(adata, 'leiden'); sc.pl.umap(adata, color='PPBP')"
```

Fig. 6. **Case 2.2. Perform clustering.** In this case, the user supplied one 10x mtx data. The primary objective of this analysis was to perform clustering.

scRNA-Seq Analysis **Case 2.3: Identify Marker Genes**

User's Inputs

Data and Description
./examples/case2.1/data/filtered_gene_bc_matrices/hg19: path to 10x mtx data

Goal
Use scanpy to identify top5 marker genes.

Generated Plans by AutoBA

1. Load the 10x mtx data from the provided file path './examples/case2.3/data/filtered_gene_bc_matrices/hg19' using the 'read_10x_mtx' function in the Scanpy library.
2. Preprocess the data using the 'pp' module in Scanpy. This includes normalizing the data, identifying highly variable genes, scaling the data, and regressing out unwanted sources of variation.
3. Perform dimensionality reduction on the preprocessed data using the 'tl.pca' function in Scanpy.
4. Cluster the cells using the 'tl.louvain' function in Scanpy.
5. Identify marker genes for each cluster using the 'tl.rank_genes_groups' function in Scanpy. This function will rank genes for each cluster based on their differential expression.
6. Extract the top 5 marker genes for each cluster from the results of the 'rank_genes_groups' function.

Generated Codes by AutoBA

```
# Step 1
# Activate the conda environment
source activate abc

# Install scanpy using pip
pip install -y scanpy

# Load the 10x mtx data using the 'read_10x_mtx' function in the Scanpy library
python -c "import scanpy as sc; adata = sc.read_10x_mtx('./examples/case2.3/data/filtered_gene_bc_matrices/hg19')"

# Step 2-11
# Activate the conda environment
source activate abc

# Extract the top 5 marker genes for each cluster from the results of the 'rank_genes_groups' function
python -c "import scanpy as sc; adata = sc.read_10x_mtx('./examples/case2.3/data/filtered_gene_bc_matrices/hg19'); sc.pp.normalize_total(adata,
target_sum=1e4); sc.pp.log1p(adata); sc.pp.highly_variable_genes(adata, min_mean=0.0125, max_mean=3, min_disp=0.5); adata.raw = adata; adata = adata[:,
adata.var.highly_variable]; sc.pp.scale(adata, max_value=10); sc.tl.pca(adata); sc.tl.louvain(adata); sc.tl.rank_genes_groups(adata, 'louvain'); result =
adata.uns['rank_genes_groups']['names']; top5_genes = result[:, :5]; print(top5_genes)"
```

Fig. 7. **Case 2.3. Identify top5 marker genes.** In this case, the user supplied one 10x mtx data. The primary objective of this analysis was to identify top5 marker genes.

ChIP-Seq Analysis **Case 3.1: Call Peaks**

User's Inputs

Data and Description

SRR620204.fastq.gz: chip-seq data for Ring1B
SRR620205.fastq.gz: chip-seq data for cbx7
SRR620206.fastq.gz: chip-seq data for SUZ12
SRR620208.fastq.gz: chip-seq data for IgGold
mm39.ncbiRefSeq.gtf: genome annotations mouse
mm39.fa: mouse genome

Goal

Call peaks for protein cbx7 with IgGold as control

Generated Plans by AutoBA

1. Load the provided data files. The files include chip-seq data for Ring1B, cbx7, SUZ12, IgGold, genome annotations for mouse, and the mouse genome.
2. Use the software 'FastQC' to perform quality control checks on raw sequence data coming from high throughput sequencing pipelines.
3. Use the software 'Trim Galore' to trim low-quality ends from reads in addition to adapter removal.
4. Use the software 'Bowtie2' to align the trimmed reads to the mouse genome. Note: We are not using 'STAR' or 'java' based software as per the rules.
5. Use the software 'Samtools' to convert the aligned reads from SAM format to BAM format for easier handling and reduced storage space.
6. Use the software 'MACS2' to call peaks for the protein cbx7 with IgGold as control. This software will identify the regions in the genome that have been enriched with reads (peaks) from the chip-seq data.
7. Use the software 'Bedtools' to annotate the peaks with the genome annotations.
8. Finally, visualize the peaks using the software 'IGV' to confirm the peak calling.

Generated Codes by AutoBA

```
# Step 1
# Activate the conda environment
source activate abc
# Load the data files
# Chip-seq data for Ring1B
zcat ./examples/case3.1/data/SRR620204.fastq.gz
> ./examples/case3.1/data/SRR620204.fastq
# Chip-seq data for cbx7
zcat ./examples/case3.1/data/SRR620205.fastq.gz
> ./examples/case3.1/data/SRR620205.fastq
# Chip-seq data for SUZ12
zcat ./examples/case3.1/data/SRR620206.fastq.gz
> ./examples/case3.1/data/SRR620206.fastq
# Chip-seq data for IgGold
zcat ./examples/case3.1/data/SRR620208.fastq.gz
> ./examples/case3.1/data/SRR620208.fastq
# Genome annotations for mouse
zcat ./examples/case3.1/data/mm39.ncbiRefSeq.gtf.gz
> ./examples/case3.1/data/mm39.ncbiRefSeq.gtf
# Mouse genome
zcat ./examples/case3.1/data/mm39.fa.gz > ./examples/case3.1/data/mm39.fa

# Step 2
# Activate the conda environment
source activate abc
# Install FastQC
conda install -y fastqc
# Perform quality control checks on raw sequence data
fastqc ./examples/case3.1/data/SRR620204.fastq
fastqc ./examples/case3.1/data/SRR620205.fastq
fastqc ./examples/case3.1/data/SRR620206.fastq
fastqc ./examples/case3.1/data/SRR620208.fastq

# Step 3
# Activate the conda environment
source activate abc
# Install Trim Galore
conda install -y trim-galore
# Trim low-quality ends from reads in addition to adapter removal
trim_galore ./examples/case3.1/data/SRR620204.fastq -
o ./examples/case3.1/output
trim_galore ./examples/case3.1/data/SRR620205.fastq -
o ./examples/case3.1/output
trim_galore ./examples/case3.1/data/SRR620206.fastq -
o ./examples/case3.1/output
trim_galore ./examples/case3.1/data/SRR620208.fastq -
o ./examples/case3.1/output

# Step 4
# Activate the conda environment
source activate abc
# Install Bowtie2
conda install -y bowtie2
# Build the index for the mouse genome
bowtie2-build ./examples/case3.1/data/mm39.fa mm39
# Align the trimmed reads to the mouse genome
bowtie2 -x mm39 -U ./examples/case3.1/output/SRR620204_val_1.fq -
S ./examples/case3.1/output/SRR620204.sam
bowtie2 -x mm39 -U ./examples/case3.1/output/SRR620205_val_1.fq -
S ./examples/case3.1/output/SRR620205.sam
bowtie2 -x mm39 -U ./examples/case3.1/output/SRR620206_val_1.fq -
S ./examples/case3.1/output/SRR620206.sam
bowtie2 -x mm39 -U ./examples/case3.1/output/SRR620208_val_1.fq -
S ./examples/case3.1/output/SRR620208.sam

# Step 5
# Activate the conda environment
source activate abc
# Install Samtools
conda install -y samtools
# Convert the aligned reads from SAM format to BAM format
samtools view -bS ./examples/case3.1/output/SRR620204.sam
> ./examples/case3.1/output/SRR620204.bam
samtools view -bS ./examples/case3.1/output/SRR620205.sam
> ./examples/case3.1/output/SRR620205.bam
samtools view -bS ./examples/case3.1/output/SRR620206.sam
> ./examples/case3.1/output/SRR620206.bam
samtools view -bS ./examples/case3.1/output/SRR620208.sam
> ./examples/case3.1/output/SRR620208.bam

# Step 6
# Activate the conda environment
source activate abc
# Install MACS2
conda install -y macs2
# Call peaks for the protein cbx7 with IgGold as control
macs2 callpeak -t ./examples/case3.1/output/SRR620205.bam -
c ./examples/case3.1/output/SRR620208.bam -f BAM -g mm -n cbx7_vs_IgGold -B -
q 0.01 --outdir ./examples/case3.1/output

# Step 7
# Activate the conda environment
source activate abc
# Install Bedtools
conda install -y bedtools
# Annotate the peaks with the genome annotations
bedtools intersect -
a ./examples/case3.1/output/cbx7_vs_IgGold_peaks.narrowPeak -
b ./examples/case3.1/data/mm39.ncbiRefSeq.gtf
> ./examples/case3.1/output/cbx7_vs_IgGold_peaks_annotated.bed

# Step 8
# Activate the conda environment
source activate abc
# Install IGV
conda install -y igv
# Visualize the peaks
igv -
g ./examples/case3.1/data/mm39.fa ./examples/case3.1/output/cbx7_vs_IgGold_p
eaks_annotated.bed
```

Fig. 8. **Case 3.1. Call peaks.** In this case, the user supplied chip-seq data for proteins Ring1B, cbx7, SUZ12, and IgGold. The primary objective of this analysis was to call peaks for protein cbx7 with IgGold as control.

ChIP-Seq Analysis **Case 3.2: Motif Discovery**

User's Inputs

Data and Description
SRR620204.fastq.gz: chip-seq data for Ring1B
SRR620205.fastq.gz: chip-seq data for cbx7
SRR620206.fastq.gz: chip-seq data for SUZ12
SRR620208.fastq.gz: chip-seq data for IgGold
mm39.ncbiRefSeq.gtf: genome annotations mouse
mm39.fa: mouse genome

Goal
Discover motifs within the peaks of protein SUZ12 with IgGold as control

Generated Plans by AutoBA

1. Load the provided chip-seq data files for SUZ12 and IgGold using a software like 'FastQC' for quality control checks.
2. Use a software like 'Trim Galore!' to trim low quality bases and adapter sequences from the reads.
3. Align the trimmed reads to the mouse genome using a software like 'Bowtie2', avoiding the use of 'STAR' and 'java'.
4. Use a software like 'MACS2' to call peaks from the aligned reads, using IgGold data as control.
5. Use a software like 'HOMER' to discover motifs within the peaks of SUZ12.
6. Validate the discovered motifs using a software like 'MEME'.

Generated Codes by AutoBA

```
# Step 1
# Activate the conda environment
source activate abc

# Install FastQC
conda install -y fastqc

# Run FastQC on the SUZ12 data
fastqc ./examples/case3.2/data/SRR620206.fastq.gz -o ./examples/case3.2/output

# Run FastQC on the IgGold data
fastqc ./examples/case3.2/data/SRR620208.fastq.gz -o ./examples/case3.2/output

# Step 2
# Activate the conda environment
source activate abc

# Install Trim Galore!
conda install -y trim-galore

# Trim low quality bases and adapter sequences from the SUZ12 data
trim_galore ./examples/case3.2/data/SRR620206.fastq.gz -
o ./examples/case3.2/output

# Trim low quality bases and adapter sequences from the IgGold data
trim_galore ./examples/case3.2/data/SRR620208.fastq.gz -
o ./examples/case3.2/output

# Step 3
# Activate the conda environment
source activate abc

# Install Bowtie2
conda install -y bowtie2

# Build the Bowtie2 index for the mouse genome
bowtie2-
build ./examples/case3.2/data/mm39.fa ./examples/case3.2/output/mm39

# Align the trimmed reads of SUZ12 to the mouse genome
bowtie2 -x ./examples/case3.2/output/mm39 -
U ./examples/case3.2/output/SRR620206_trimmed.fq.gz -
S ./examples/case3.2/output/SRR620206.sam

# Align the trimmed reads of IgGold to the mouse genome
bowtie2 -x ./examples/case3.2/output/mm39 -
U ./examples/case3.2/output/SRR620208_trimmed.fq.gz -
S ./examples/case3.2/output/SRR620208.sam

# Step 4
# Activate the conda environment
source activate abc

# Install MACS2
conda install -y macs2

# Call peaks from the aligned reads of SUZ12 using IgGold data as control
macs2 callpeak -t ./examples/case3.2/output/SRR620206.sam -
c ./examples/case3.2/output/SRR620208.sam -f SAM -g mm -n SUZ12 -B -q 0.01 --
outdir ./examples/case3.2/output

# Step 5
# Activate the conda environment
source activate abc

# Install HOMER
conda install -y homer

# Discover motifs within the peaks of SUZ12
findMotifsGenome.pl ./examples/case3.2/output/SUZ12_peaks.narrowPeak
mm ./examples/case3.2/output/HOMER_motifs -size given

# Step 6
# Activate the conda environment
source activate abc

# Install MEME
conda install -y meme

# Validate the discovered motifs using MEME
meme ./examples/case3.2/output/HOMER_motifs -
oc ./examples/case3.2/output/MEME_motifs
```

Fig. 9. **Case 3.2. Discover motifs within the peaks.** In this case, the user supplied chip-seq data for proteins Ring1B, cbx7, SUZ12, and IgGold. The primary objective of this analysis was to discover motifs within the peaks of protein SUZ12 with IgGold as control.

ChIP-Seq Analysis **Case 3.3: Functional Enrichment**

User's Inputs

Data and Description

SRR620204.fastq.gz: chip-seq data for Ring1B
SRR620205.fastq.gz: chip-seq data for cbx7
SRR620206.fastq.gz: chip-seq data for SUZ12
SRR620208.fastq.gz: chip-seq data for IgGold
mm39.ncbiRefSeq.gtf: genome annotations mouse
mm39.fa: mouse genome

Goal

Perform functional enrichment for protein Ring1B, use protein IgGold as the control

Generated Plans by AutoBA

1. Load the provided chip-seq data files for Ring1B and IgGold using a software like FastQC for quality control checks.
2. Use a software like BWA (Burrows-Wheeler Aligner) to align the chip-seq data for Ring1B and IgGold to the mouse genome (mm39.fa).
3. Use a software like MACS (Model-based Analysis of ChIP-Seq) to identify the peaks in the chip-seq data for Ring1B and IgGold.
4. Use the genome annotations file (mm39.ncbiRefSeq.gtf) to annotate the identified peaks using a software like BEDTools.
5. Use a software like DESeq2 to perform differential binding analysis between Ring1B and IgGold.
6. The genes associated with the differential peaks will be identified and used for functional enrichment analysis.
7. Use a software like DAVID (Database for Annotation, Visualization and Integrated Discovery) or g:Profiler for functional enrichment analysis.
8. Interpret the results of the functional enrichment analysis to understand the biological processes and pathways associated with Ring1B.

Generated Codes by AutoBA

```
# Step 1
# Activate the conda environment
source activate abc
# Install FastQC
conda install -y fastqc
# Load the chip-seq data files for Ring1B and IgGold
fastqc ./examples/case3.3/data/SRR620204.fastq.gz -o ./examples/case3.3/output
fastqc ./examples/case3.3/data/SRR620208.fastq.gz -o ./examples/case3.3/output

# Step 2
# Activate the conda environment
source activate abc
# Install BWA
conda install -y bwa
# Align the chip-seq data for Ring1B to the mouse genome
bwa
mem ./examples/case3.3/data/mm39.fa ./examples/case3.3/data/SRR620204.fastq.
gz > ./examples/case3.3/output/Ring1B.sam
# Align the chip-seq data for IgGold to the mouse genome
bwa
mem ./examples/case3.3/data/mm39.fa ./examples/case3.3/data/SRR620208.fastq.
gz > ./examples/case3.3/output/IgGold.sam

# Step 3
# Activate the conda environment
source activate abc
# Install MACS
conda install -y macs
# Identify the peaks in the chip-seq data for Ring1B
macs2 callpeak -t ./examples/case3.3/output/Ring1B.sam -f SAM -g mm -n Ring1B -B
-q 0.01 --outdir ./examples/case3.3/output
# Identify the peaks in the chip-seq data for IgGold
macs2 callpeak -t ./examples/case3.3/output/IgGold.sam -f SAM -g mm -n IgGold -B
-q 0.01 --outdir ./examples/case3.3/output

# Step 4
# Activate the conda environment
source activate abc
# Install BEDTools
conda install -y bedtools
# Annotate the identified peaks for Ring1B
bedtools intersect -a ./examples/case3.3/output/Ring1B_peaks.narrowPeak -
b ./examples/case3.3/data/mm39.ncbiRefSeq.gtf -wa -wb
> ./examples/case3.3/output/Ring1B_annotated_peaks.bed
# Annotate the identified peaks for IgGold
bedtools intersect -a ./examples/case3.3/output/IgGold_peaks.narrowPeak -
b ./examples/case3.3/data/mm39.ncbiRefSeq.gtf -wa -wb
> ./examples/case3.3/output/IgGold_annotated_peaks.bed

# Step 5
# Activate the conda environment
source activate abc
# Install DESeq2
conda install -y bioconductor-deseq2
# Perform differential binding analysis between Ring1B and IgGold
Rscript -e "library('DESeq2');
ring1b_counts <-
read.table('./examples/case3.3/output/Ring1B_annotated_peaks.bed', header =
TRUE, stringsAsFactors = FALSE);
igGold_counts <-
read.table('./examples/case3.3/output/IgGold_annotated_peaks.bed', header =
TRUE, stringsAsFactors = FALSE);
counts <- cbind(ring1b_counts, igGold_counts);
colData <- data.frame(condition = factor(c('Ring1B', 'IgGold')));
dds <- DESeqDataSetFromMatrix(countData = counts, colData = colData, design = ~
condition);
dds <- DESeq(dds);
res <- results(dds);
write.table(res, file = './examples/case3.3/output/differential_binding_analysis.txt',
sep = '  ', quote = FALSE, row.names = TRUE)"

# Step 6
# Activate the conda environment
source activate abc
# Install BEDTools
conda install -y bedtools
# Identify the genes associated with the differential peaks for Ring1B
bedtools intersect -a ./examples/case3.3/output/Ring1B_annotated_peaks.bed -
b ./examples/case3.3/data/mm39.ncbiRefSeq.gtf -wa -wb
> ./examples/case3.3/output/Ring1B_genes.bed
# Identify the genes associated with the differential peaks for IgGold
bedtools intersect -a ./examples/case3.3/output/IgGold_annotated_peaks.bed -
b ./examples/case3.3/data/mm39.ncbiRefSeq.gtf -wa -wb
> ./examples/case3.3/output/IgGold_genes.bed

# Step 7
# Activate the conda environment
source activate abc
# Install g:Profiler
conda install -y gprofiler
# Perform functional enrichment analysis for Ring1B
Rscript -e "library('gProfileR');
ring1b_genes <- read.table('./examples/case3.3/output/Ring1B_genes.bed', header
= TRUE, stringsAsFactors = FALSE);
ring1b_enrichment <- gprofiler(ring1b_genes$V4, organism='mmusculus');
write.table(ring1b_enrichment, file =
'./examples/case3.3/output/Ring1B_enrichment.txt', sep = '\t', quote = FALSE,
row.names = TRUE)"
# Perform functional enrichment analysis for IgGold
Rscript -e "library('gProfileR');
igGold_genes <- read.table('./examples/case3.3/output/IgGold_genes.bed', header =
TRUE, stringsAsFactors = FALSE);
igGold_enrichment <- gprofiler(igGold_genes$V4, organism='mmusculus');

# Step 8
# Activate the conda environment
source activate abc
# Install R
conda install -y r-base
# Interpret the results of the functional enrichment analysis for Ring1B
Rscript -e "ring1b_enrichment <-
read.table('./examples/case3.3/output/Ring1B_enrichment.txt', header = TRUE,
stringsAsFactors = FALSE);
ring1b_enrichment <- ring1b_enrichment[order(-ring1b_enrichment$p.value), ];
ring1b_enrichment <- ring1b_enrichment[ring1b_enrichment$p.value < 0.05, ];
write.table(ring1b_enrichment, file =
'./examples/case3.3/output/Ring1B_enrichment_significant.txt', sep = '\t', quote =
FALSE, row.names = TRUE)"
```

Fig. 10. **Case 3.3. Functional Enrichment.** In this case, the user supplied chip-seq data for proteins Ring1B, cbx7, SUZ12, and IgGold. The primary objective of this analysis was to conduct functional enrichment analysis specifically for protein Ring1B, with protein IgGold serving as the control.

Spatial Transcriptomics **Case 4.1: Neighborhood enrichment analysis**

User's Inputs

Data and Description
slice1.h5ad: spatial transcriptomics data for slice 1 in AnnData format

Goal
Use squidpy for neighborhood enrichment analysis

Generated Plans by AutoBA

1. Load the necessary libraries for the analysis. In this case, we will use 'squidpy' and 'anndata' for the analysis. Note that 'squidpy' is a powerful tool for spatial transcriptomics data analysis.
2. Load the spatial transcriptomics data for slice 1 using the 'anndata.read_h5ad' function. The file path is './examples/case4.1/data/slice1.h5ad'.
3. Perform initial data exploration to understand the structure and content of the data. This can be done using functions such as 'adata.obs' and 'adata.var' in 'anndata'.
4. Use the 'squidpy.gr.spatial_neighbors' function to compute a spatial graph based on the spatial coordinates of the data. This function will add a spatial neighbors graph to the AnnData object.
5. Use the 'squidpy.gr.nhood_enrichment' function to compute neighborhood enrichment. This function tests for each cluster, whether its spatial neighbors are randomly distributed.
6. Visualize the neighborhood enrichment results using the 'squidpy.pl.nhood_enrichment' function. This function plots the neighborhood enrichment test results.
7. Save and export the results for further analysis or presentation.

Generated Codes by AutoBA

```
# Step 1
# Activate the conda environment
source activate abc

# Install the necessary libraries
conda install -y squidpy
conda install -y anndata

# Step 2-3
source activate abc

# Perform initial data exploration
python -c "import anndata; adata = anndata.read_h5ad('./examples/case4.1/data/slice1.h5ad'); print(adata.obs); print(adata.var)"

# Step 4-7
source activate abc

# Compute a spatial graph based on the spatial coordinates of the data
python -c "import anndata; import squidpy as sq; adata = anndata.read_h5ad('./examples/case4.1/data/slice1.h5ad'); sq.gr.spatial_neighbors(adata);
sq.gr.nhood_enrichment(adata); adata.write('./examples/case4.1/output/results.h5ad')"
```

Fig. 11. **Case 4.1. Neighborhood enrichment analysis.** In this case, the user supplied spatial transcriptomics data for slice 1 in AnnData format. The primary objective of this analysis was to use squidpy for neighborhood enrichment analysis.