实验物理的大数据方法

2024 年 7 月 16 日

目录

第一章　总论

1.1　认识大家

自我介绍

2005-2009 在数理基科学习 （科协）

2009-2018 在日本神冈地下实验室，学习研究中微子和暗物质

2018-至今 在工程物理系近代物理研究所任教

主要工作：锦屏中微子实验

合作研究

1. JUNO 江门中微子实验
2. 日本 SuperK 超级神冈实验

研究兴趣：中微子质量、核子衰变、地球中微子

爱好：大数据分析、高性能计算、系统架构运维

教学团队

助教

张爱强 工物系本科毕业、博士生

王宇逸 物理系本科毕业，工物系博士生

武益阳 物理系本科毕业，工物系博士生

徐闯 工物系本科毕业、博士生

郝传晖 物理系本科毕业，工物系博士生

刘逸祺 工物系本科毕业、博士生

陶嘉燊 物理系本科毕业

教学团队（续）

顾问

陈晟祺 计算机系本科毕业、博士生

小助教

- 请踊跃报名。

算力支持
物理系科协（somnia）、工物系科协（cat）

1.2 为什么

我为什么在这里?

培养方案要求?

- 本课程的“性价比很低”!（由某巨佬结课后评价道）

班主任的观察
（学长）每年都劝退（学弟学妹），每年都劝不住……然后劝不住的学完开始劝退下一届

听说这门课讲 Python?

- 现在猴子都会 Python，我不会的话会很焦虑

“实验物理”和“大数据方法”哪个更重要

- 课程名的中心语在“大数据方法”，所以后者重要?
- 服务于“实验物理”，所以前者重要?
- 思考：如果两者去掉其一，你能否接受?

形式逻辑与实验同等重要

- 计算机是形式逻辑演绎机，计算机的原理植根于逻辑。

爱因斯坦 1953 给 J.S.Switzer 的信
西方科学的发展是以两个伟大的成就为基础：希腊哲学家发明形式逻辑体系 (在欧几里得几何中)，以及 (在文艺复兴时期) 发现通过系统的实验可能找出因果关系。

释义

什么是“实验物理”

- 一切费电的物理!

  “理论学家费纸，实验学家费电，理论实验物理学家费米”

  - 欧洲核子中心 (CERN) 一年用电约 1.3 TW h。
    北京东城区西城区 2016 年用电总量 10.1 TW h，人口 200 万。

- 观察物理现象的学科

  - 控制物理现象发生的环境? 是 → “实验”(狭义)；否 → “观测”。
  - 实验物理的目标：发现物理规律，包括测量物理常数。
    从取得的数据出发，进行统计推断，证伪物理规律的假设。

什么是“大数据”

商业概念。“大”是相对的，大数据是指无法用一台计算机处理的数据。

实验物理与计算机

- 逻辑电路的起源

  - 盖革计数器，电离辐射通过时给出脉冲信号
  - 需要构造：顶端探测器无计数，中间有计数的逻辑
  - 真空电子管“与”、“或”、“非”等逻辑门出现
  - 真空电子管通用电子计算机：ENIAC （1945）

- 中子和辐射输运问题

  - 第一代电子计算机大多用来进行 Metropolis Monte Carlo 算法
  - 模拟预测氢弹中的输运过程

- 半导体逻辑电路

  - 晶体管替代真空电子管
  - 计算机功耗和体积大幅减小

- Monte Carlo 算法成为大型物理实验设计的一部分

计算历史：大型机

- 一台大型计算机，用户通过终端 (Terminal) 连接

计算历史：个人计算机

- 1981 年

计算历史：个人计算机组成集群

- 1990s 个人计算机组成集群 ”beowulf”

高性能计算与超级计算机

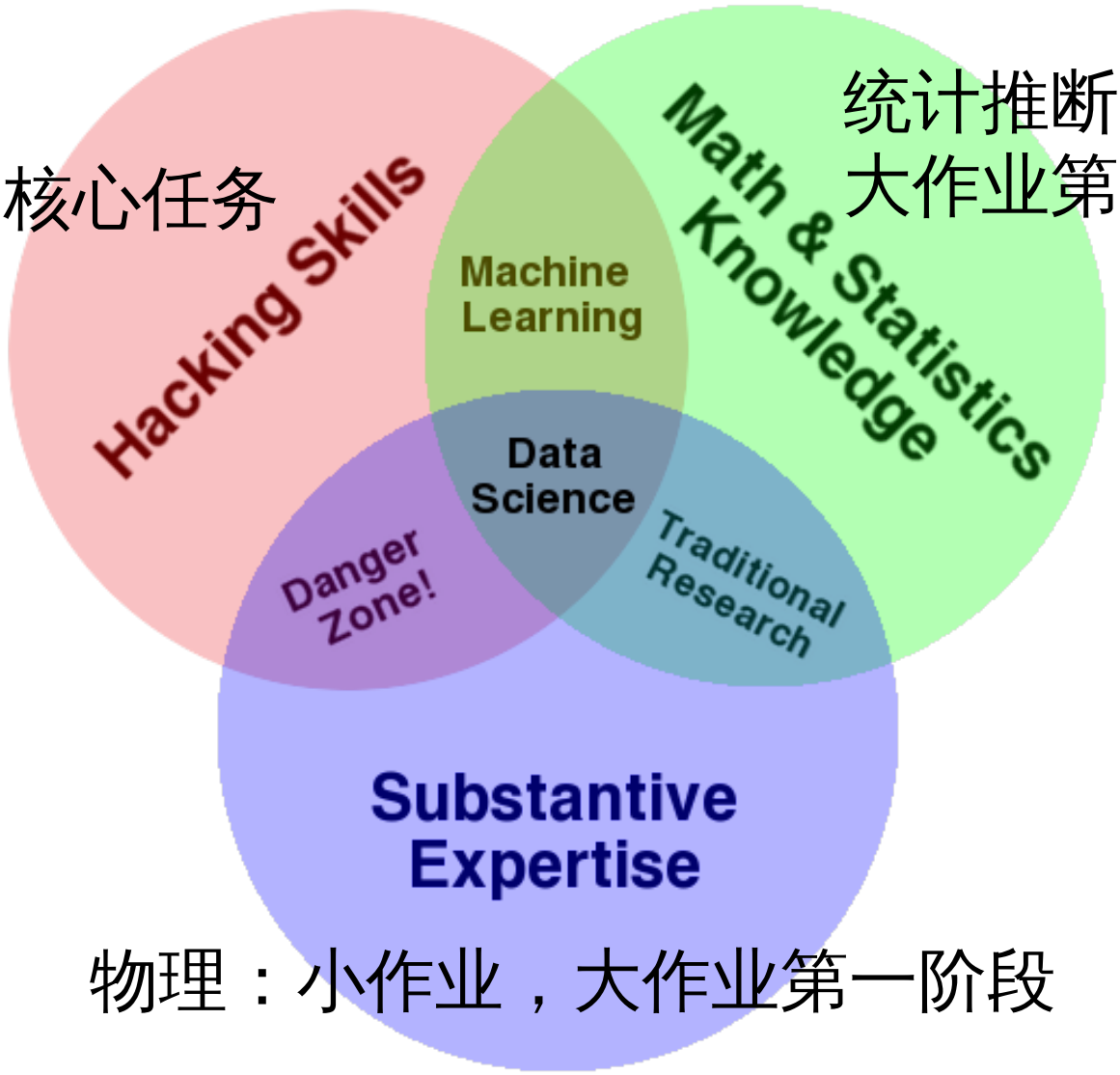
- 更大的计算机集群，优化节点间的通信，共享存储。
- 软件环境如何配置?

大物理与大数据

- 大型物理实验，产生大数据，多台计算机协同工作：分布式计算。
- 跨地理的分布式计算称为“网格”(grid)，在早期促进和影响了互联网的发展。
  - FAST 天眼：每年 $\sim$1 PB $= 1 \times 10^{15}$ B。
  - JUNO 江门中微子实验：每年 $\sim$2 PB 。
- CERN 在 90 年代决定使用 Intel 民用计算机组成的集群来完成 LHC 的计算任务，是个人电脑级硬件集群成为科学计算主流的标志。
  - X86(Intel 和 AMD) 计算集群是当今大数据工业界的主流配置。
- 近年来，大数据方法独立于实验物理，被工业界广泛采用，与神经网络结合，催生了“深度神经网络”，或“深度学习”。
  - 一系列新方法被应用到实验物理。

此课在知识体系中的地位

- 这是一个数据的时代。
- 数据时代需要 黑客技术 Hacking Skills、 数理统计 Math & Statistics Knowledge 、 专业知识 Substantive Expertise 等多方面技能。

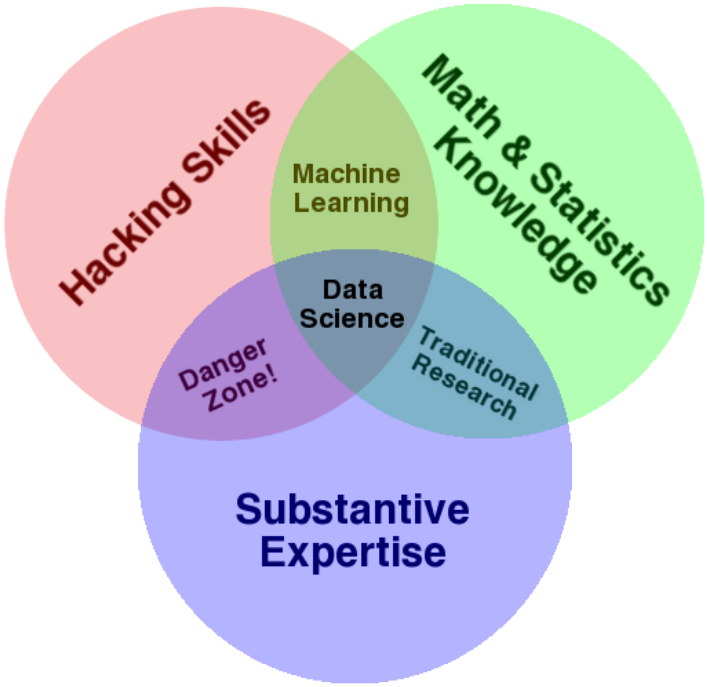


核心任务

统计推断：
大作业第二阶段

物理：小作业，大作业第一阶段

技能树



1. 本课程目标为数理大类的同学补齐 黑客技术 。
2. 警惕 *Danger Zone!* 。

   黑客技术 人类掌握工具的最高水平

   数学与统计 人类理性思维的最高水平，欢迎选修《概率统计分析及量测技术》
3. 专业知识 根据自己的兴趣选择。
   - 物理学的思维方式和实践方法是一切专业知识的标杆。
4. 本科时代要积极认真学习以上硬知识。

物理学是“硬科学”

以知识的量化程度排序：

- 0 类学科：努力区分偶然统计关联和因果关系
  - 到底哪些是决定性因素?
  - X 基因决定了此性状，Y 基因决定了此性状？XY 一起决定?
  - $X\bar{Y}$ ， $\bar{X}Y$ ？
- 1 类学科：努力确认变量之间影响的趋势
  - “房价加速增长的势头得到遏制”
- A 类学科：努力算出具体的数字

  - 地球的年龄是 46 亿年，45–47 亿年，还是 46–48 亿年?

- Z 类学科：没有不确定度的结果不是科学结论!

  - 电子磁矩，以玻尔磁子 μ_B 为单位

$$g/2 = 1.00115965218073(28)$$

    Ref: Phys.Rev.Lett.100:120801,2008

1.3 课程概论

数据分析的指导原则

“复现”原则 以人类语言和计算机语言的形式，详细记录每一步计算。

这是科学的基本精神，与可证伪性一起，是区分科学与伪科学的标志。

“透明”原则 每一步的数据都应尽量可以被人类直接阅读，比如 JSON, CSV。如果不得不用二进制模式，一定是使用最普遍最开放的格式，比如 HDF5。

“一次”原则 Single Point of Truth, Don't Repeat Yourself. 不可在分析做任何重复，任何有意义的信息都应该被共享。

“最佳工具”原则 尽量使用高级语言和语法糖，为每个子任务选择合适的工具。只有在性能分析之后，才在必要时使用低级语言进行性能加速。

- 最佳工具会随时间变化，因此课程名不是 ~~Python 数据处理与科学计算~~
- 推论：入手研究之前，应当优先调研现有工具。

课程内容

- 以实验物理为主题，学习大数据方法的基础。

  1. 理解数据的科学精神，从一开始养成良好的科研习惯
  2. 掌握典型工具 Git, Python 和常用命令行工具，鼓励自学
  3. 培养工具的品位，针对具体场景选择合适的工具

计划安排

- 实验物理的大数据方法 (1) 经管新楼 A119

  第一周 Git，Python 基础

    – 周二至周五，三四大节

  第二周 Python 科学计算与可视化

  - 周一至周四，三四大节

- 实验物理的大数据方法 (2) 经管新楼

  第三周 数据处理的命令行工具

  第四周 基于关系代数的数据组织

- 答疑：课后 4:55–5:30

课程评估 按百分制记分 录入成绩时映射为等级

平时作业 62%

- 覆盖课堂学习的知识要点
  - 平时作业中的 80% 为黑盒测试，20% 为白盒测试。
  - 白盒测试重点考察：Git commit 的内容必须有意义，程序书写规范

大作业 30%

- 取材自实验物理不同方向的真实场景。也可以由同学自行提出。
  - 分两个阶段，对应前两周“生成”和后两周“分析”，互为逆过程。

捌分 8%

- 小助教 (A)，贡献 FAQ 内容 (B)，审校讲义内容 (C)，课堂贡献 (D)，娱乐作业 (E)
- 计入总评 $\min(8, \sqrt{A^2 + B^2 + C^2 + D^2 + E^2})$

教材

- 自编讲义：《实验物理的大数据方法》

参考资料

- 往年课程录像：`http://hep.tsinghua.edu.cn/~orv/teaching/physics-data/`

辅助资料

- Allen Downey, Think Python 2e
- Fernando Perez et al., Scipy Lecture Notes
- Jeroen Janssens, Data Science at the Command Line
- David MacKay, Information Theory, Inference, and Learning Algorithms
- `https://learnxinyminutes.com/` 从已经有编程基础迅速入门 Python。

有智慧地提问

- 遇到困难多求助
  - 特别是那些可以把人“卡住”的“小”问题。一定不要自己扛，及时求助：老师、助教、小助教。
  - 学长忠告 ~~不要问我是怎么知道的系列~~
    - 不要用某度，用 Google 或 Bing 海外版
    - 以 CSDN 为代表的中文网站 99% 的信息都是错的（或过时的）
    - stackexchange 系列英文网站 70% 信息可信，其它英文网站 50% 可信
    - 官方文档 100% 可信
    - 综合多种信息来源，作出自己的判断，不要轻信小学生博主。

求助时要有智慧 作者 Eric Raymond

- 原则：让他人以最小的努力复现出你的问题
  1. 不要这样说话： ~~救命! 我的程序坏了！哪位大神快来帮帮我！急！在线等！哭了!~~
  2. 使用最简洁的语言，借助最简单的例子描述问题
- 原则：努力让你的问题使更多的人受益
  1. 非隐私问题，尽可能公开提问，公开讨论。 不要私信

如何使用 Gitlab 提问和解答

- issue （议题）
  - 创建要讨论的问题;
  - 注意指定 issue 的负责人。

GitLab issue

- `https://git.tsinghua.edu.cn/physics-data/faq/-/issues/`
- `https://git.tsinghua.edu.cn/physics-data/lecture/-/issues`
- 创建 issue
  - 标题：概括问题
  - 正文：详细描述问题
    1. 给出复现问题的方法，提供 最小可复现单元
    2. （可选）分析问题，提出可能的解决方法
- 回复 issue ，参与讨论
- 关注 issue ，打开 “Notifications”

1.4 计算环境

GNU 环境

- POSIX 是 *Portable Operating System Interface* 的缩写，是关于计算机操作系统的国际标准。它规定了操作系统的基本工具和程序接口。
- GNU 是 *GNU is not Unix* 的缩写，是自由软件运动形成的 POSIX 环境。
- 科学研究中时间和空间跨度都很大，需要使用公开的国际标准化接口，才能满足 复现 透明 和 最佳工具 原则。

可使用 GNU 环境的操作系统

GNU/Linux 天然大佬。请帮助周围的同学，并自荐小助教。

Apple macOS 使用 `UTM` 虚拟机。

-

Microsoft Windows 本身不满足，但可以使用以下扩展

1. Windows Subsystem for Linux (WSL)
2. Cygwin, MinGW

权威指南：`https://physics-data.meow.plus/faq/`

GNU 环境的基本操作

ls list structure, 列出当前路径中的文件

cd change directory, 改变当前路径

sudo 使用管理员权限执行操作

apt Advanced Package Tool 软件管理器

- `apt update`
- `apt install nano diffutils patch`

nano 文件编辑器，与 emacs, vi, VSCode 等互换

- 查找 `^W` 、替换 `^R` 、保存 `^O` 、退出 `^X`
  注： `^W` 代表按住 Ctrl 键，再按 W。

命令参数标准

- GNU 标准定义了推荐的参数格式
- `git config --global user.name "Benda Xu"`
- 命令的各个部分由 半角 空格分隔
  - 不作为分隔符的空格由 半角 引号标出
- 短参数：一条短线接一个字母
  - `ls -a`
  - `kill -9 xxxx`
  - `sudo -s`
  - 可以缩写在一起
    - `ls -lrta`
    - `sudo -sE`
- 长参数：两条短线接一个单词
  - `ls --color=tty`
  - `git show --pretty=short --show-signature`
  - 由 GNU 推荐，是在 POSIX 标准之上的扩展
- 更多参数，使用 `man` 查看文档 `man ls`, `man git`

1.5 学习建议

学术共同体意识

- 我们在一个月的时间里，共同理解物理和逻辑，沉淀下来我们的探索过程

网站的使用

- 收藏夹

邮件

- 客户端

编辑器

程序编辑器三大流派

- Emacs
- Vi
- Visual Studio Code

1.6 版本控制

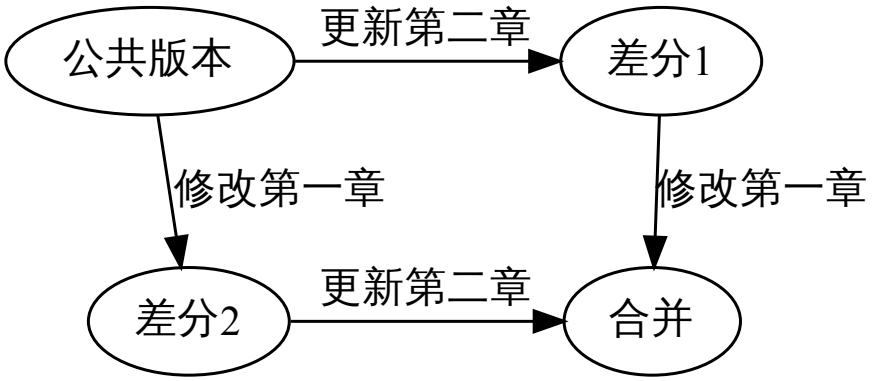
版本控制

- “我和同学合写文章，结果他是在昨天的版本上改的！我还得重改!”

发展历史

石器时代 实验报告-v1, 实验报告-v2, 实验报告-v2.2, 实验报告-v2.2-续本达更新 20190629 ……

青铜时代 diff, patch 文本差分算法



铁器时代 版本控制服务：CVS，SVN

现代 分布式版本控制：Git

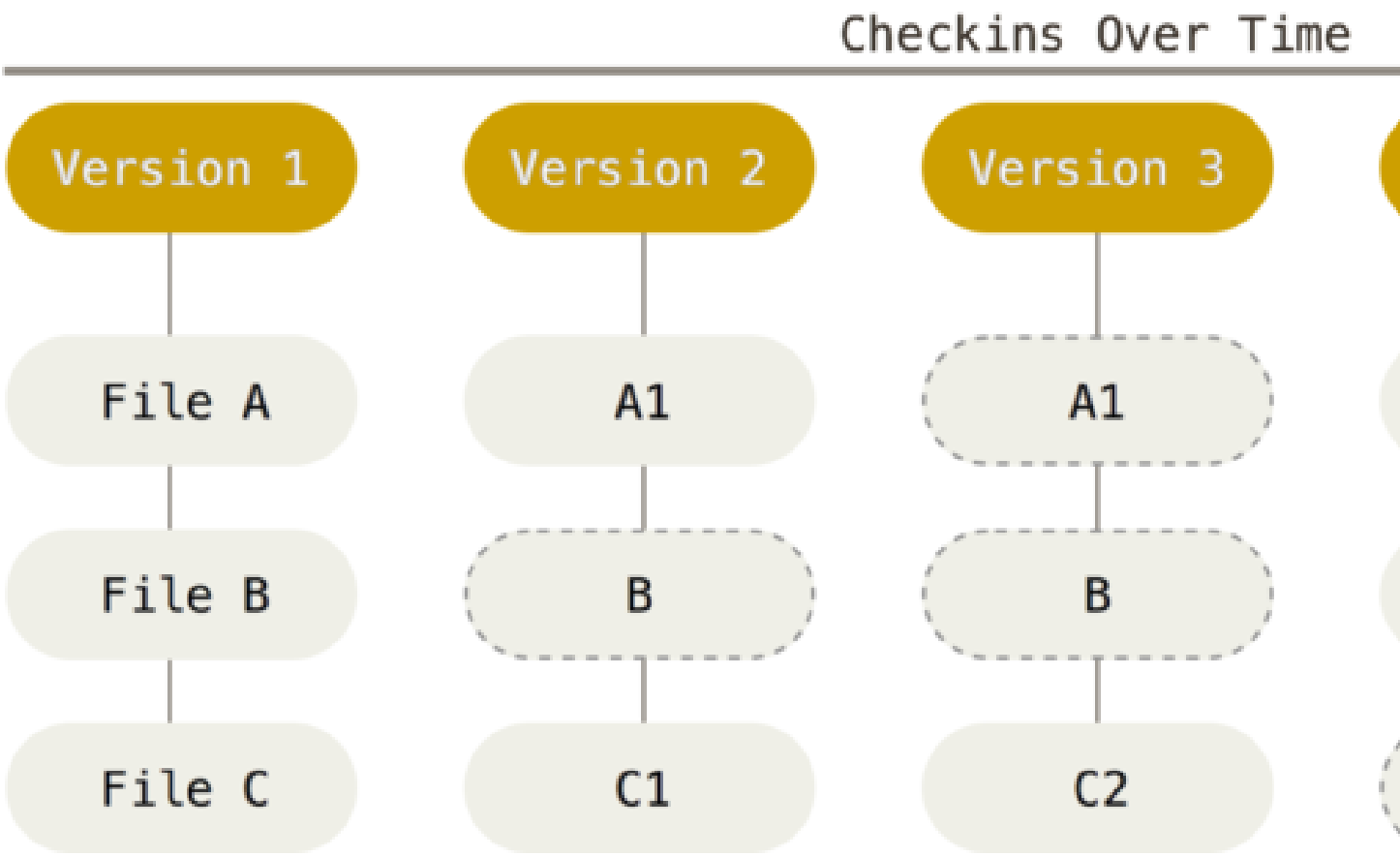
Git 案例

“我上周写的程序被不小心覆盖掉了！无法找回之前的版本。”

- 小强同学经过认真思考和试验，发明了一种巧妙的解决问题方法，并写成程序，经验证效果拔群。
  - 小强非常振奋，他把成果记了下来，并继续重构并优化解法。
  - 两个星期之后，他的优化并没有达到预期效果，反而使结果变差。
  - 令他更为懊恼的是，他已经无法达到自己两周之前的高度了，因为忘记了改之前的程序是怎么写的。
- 复现 原则被破坏。
- 解法-20190530.backup ， 解法-20190608 ？ 重复存储， 一次 原则被破坏。
- 讨论：你遇到的最难受的问题，如何解决?

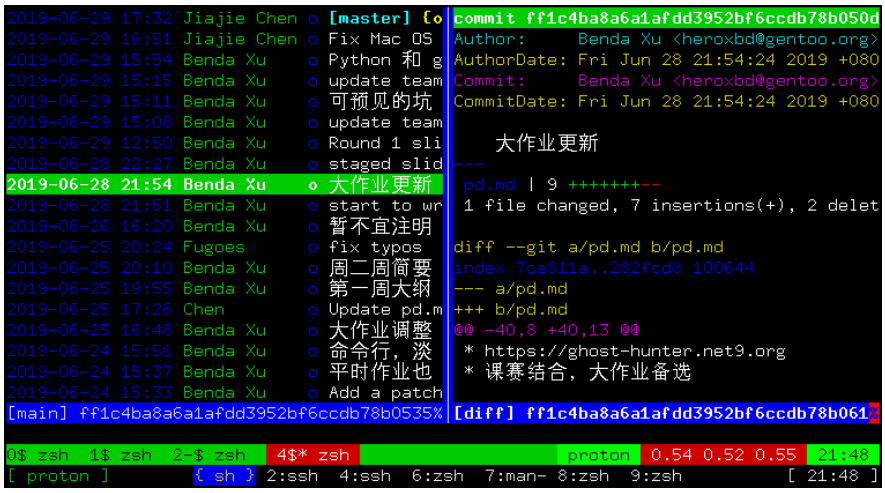
Git 基础 安装： `apt install git`

- 使用 Git，养成良好的习惯，就能解决以上问题。
- Git 由 Linus Torvalds 发明，用来支撑全世界 5000 名以上 Linux 操作系统内核开发者的协作。是我们的 最佳工具 。



- 在任意时间我们都可以选择给工作区进行一次“快照”。
- 相邻快照之间，是代表文件改动的差分量。

例子：撰写讲义



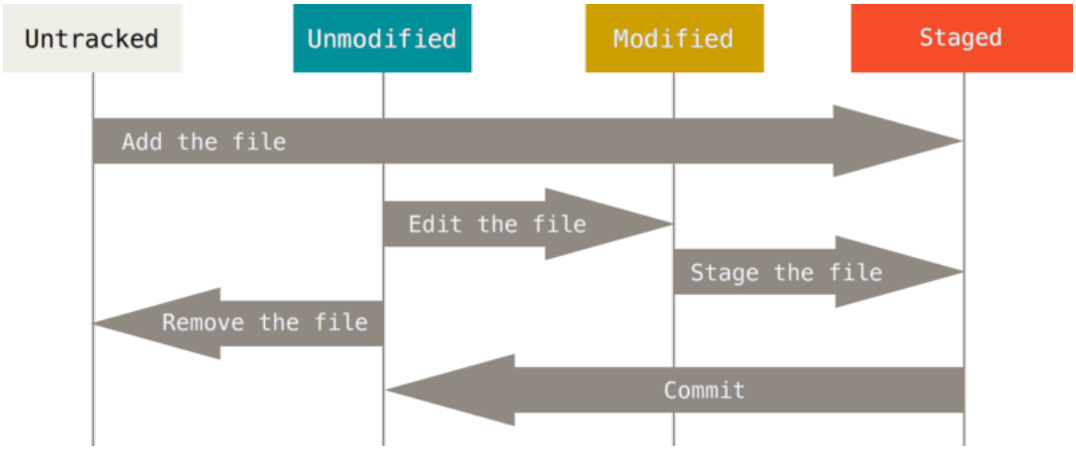
- 查看每次提交的改动。工具 `tig`，可由 `apt install tig` 安装。

Git 三种状态

已提交（committed）改动安全保存在数据库中

已修改（modified）自上一次提交，有文件被修改

已暂存（staged）已修改的文件被作了标记，将被包含在下一轮提交中



作业：Git 练习

☒ DONE 你会在 清华 GitLab 收到一个新仓库，是今天的作业

☒ DONE 设置本地到 清华 GitLab 的访问权限

- ☒ 生成 SSH 密钥对
- ☒ 上传到 `https://git.tsinghua.edu.cn`

☒ DONE 同步你的 Git 仓库，输入你的基本信息

```
我的姓名:
我的学号:
我的 Python 版本信息:

课程感言:
```

☐ TODO 查看修改 (diff)，预估得分

☐ TODO 添加修改 (add)，提交 (commit)，推送 (push)

Git 基本命令

diff 查看改动

status 查看状态

add 添加文件

commit 提交

log 查看历史

pull 从远程下载

push 推送到远程

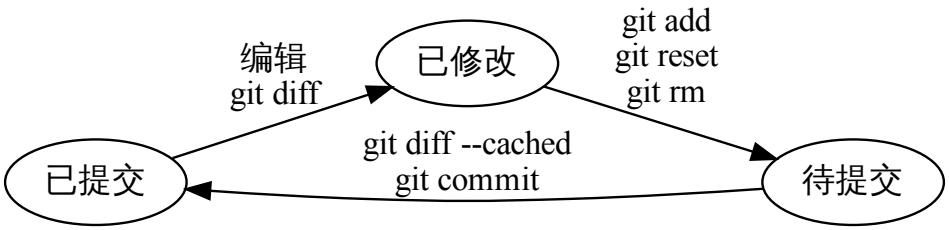
merge 合并分支

内建手册 man

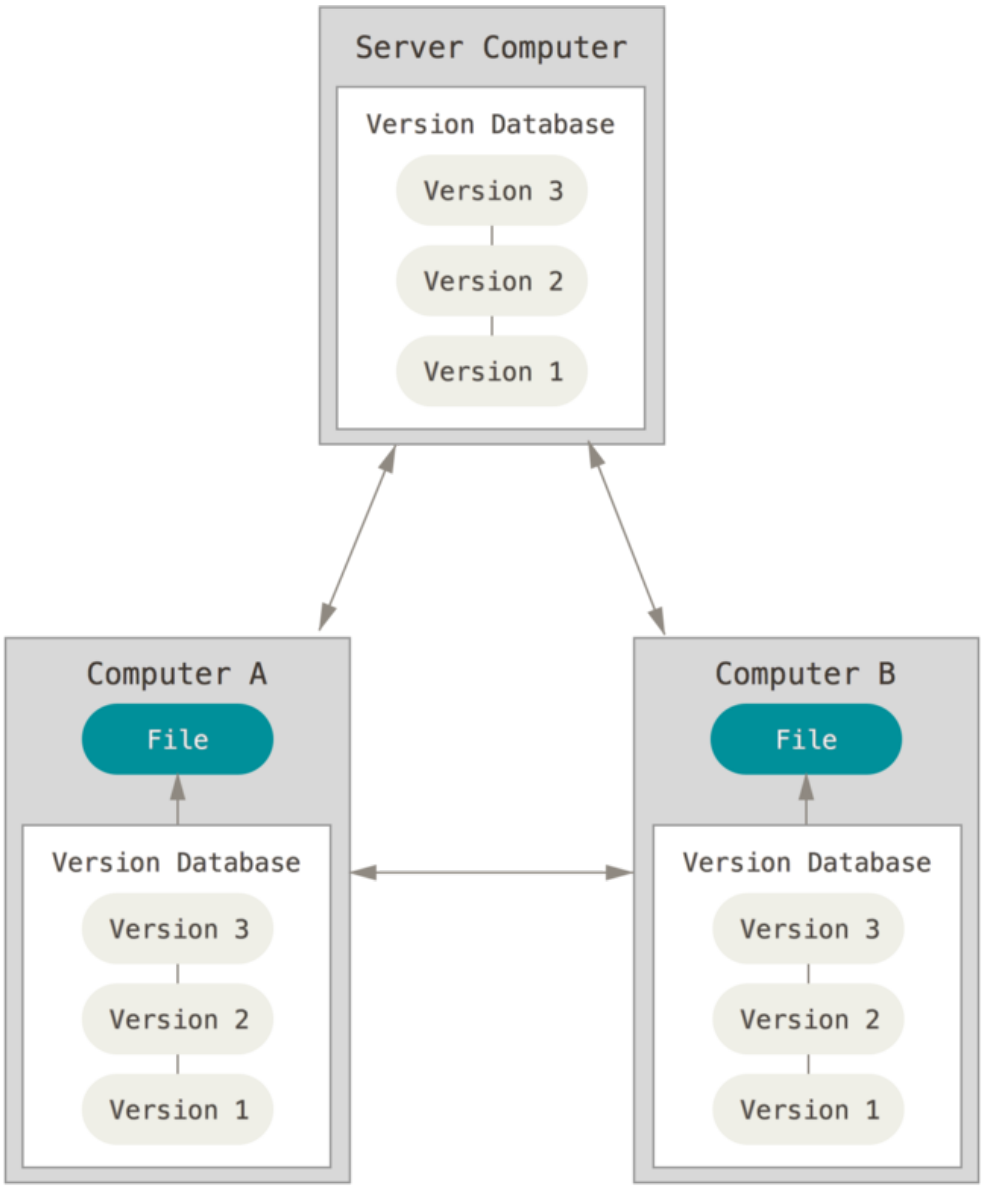
命令的具体含义可以参考 `man` 命令，例如，

```
$ man git add
```

Git 的状态与命令



Git 仓库间通信



- 建议：通过 SSH (secure shell) 通信。

SSH 通信

- 通信的认证问题：如何证明我是“我”?
  - 手机验证码
  - 一次性密码（one-time password，OTP）
  - 用户名密码
  - 非对称加密（银行 U 盾）

- 生成 SSH 非对称密钥对
  - 一个公钥：公布， `id_rsa.pub`
  - 一个私钥：私密， `id_rsa`
  - 在网页界面把 `id_rsa.pub` 交给 `git.tsinghua.edu.cn`
  - `git.tsinghua.edu.cn` 用我的公钥加密挑战码，只有我的私钥才能解密
    - `git.tsinghua.edu.cn` 知道了我是“我”
    - 我也需要知道它是真的 `git.tsinghua.edu.cn`
  - 建立了与 `git.tsinghua.edu.cn` 的可靠通信

提交说明 commit message

- `git commit` 时需要输入一段话概括本次 commit 所作的修改
- 目标：让他人和未来的自己一目了然，只通过读此信息就知道自己做了什么

commit 样例

- 不错的例子：“完善了个人信息页面，并填写了课程感言”
- 短的 ，把事情讲清楚，把涉及到的人讲清楚。
- 较长的 ，把所有的改动概括清楚。
- 更长的 ，在 commit 里说明为什么要这样做，引用相关的讨论。

精心撰写提交说明

- 说明既要给人类读，也要易于自动处理
- 第一行：高度概括改动
- 空一行后，撰写具体改动内容
  - 形式不限，可分段、分要点。
  - 篇幅不限，以描述清楚为目标。
- 备注：感谢其他贡献者。

何时提交 commit ？

- 完成一件事，有一个较独立的“版本”：添加一个功能，修复一个问题，重构一块代码。
- 初学者可尝试多提交，可以充分练习 Git 操作，也可以获得更细致的开发历史。
  - 不要 把多个不相关的修改攒成一次巨大的提交。

反例

1. 数数：1,2,3
2. 复读：Update,Update,Update
3. 欺负键盘：aaaaa,wwwww,asdf

- 不太好的例子：“完成作业”

来自学长的忠告：作业得分不重要，重要的是优雅地得分
我看到的无意义 commit message 有：“这是一个描述”，“xX”，“1234567899”等等。由于 Git 的特性，后续的提交没有办法覆盖这些无意义的 commit ，所以这个操作在很大程度上是不可逆的。也就是说，批阅作业的助教可以看到这些无意义的提交信息，然后或许会给你的白盒扣上 2 分。虽然我相信批改作业的助教不会这么残忍，但请一定要有“提交信息要有意义”的意识。

请查看白盒标准：`https://physics-data.meow.plus/faq/rules/whitebox/`

- 如果想要自己玩一下 Git 的相关操作（强烈推荐这么做，尤其是要学会 add 和 commit 如何取消。我个人对于一个操作是否可逆是有追求的），可以自己在 GitLab 上面创

建一个项目练习。

Git 与编辑器的整合

禁止使用 Gitlab 网页上传和修改工具

- 远远次于命令行的功能，无推广价值
- 禁止使用，防止养成不良习惯

在命令行调用相应的编辑器

- `git` 会调用 环境设定 的编辑器
  - 临时调整：使用 `EDITOR=xxxxx` 前缀
    - `EDITOR="code --wait"` 见 李禹锋的 issue 88
    - `EDITOR="vim"`
    - `EDITOR="emacsclient"` 见 黄宇同的 issue 109
- Debian 上永久调整系统的默认编辑器，管理员权限执行

```
update-alternatives --config editor
```

  选择需要的编辑器

- 配置 git 的编辑器：查看文档配置，关键字是 `core.editor`

```
man git config
```

参考资料

- Git 提交说明的白盒采分项:

  `https://physics-data.meow.plus/faq/whitebox/#git`

- 写好提交说明的七大原则 by cbeams `https://chris.beams.io/posts/git-commit/`

- 提交说明编写指南 by 阮一峰

  `http://www.ruanyifeng.com/blog/2016/01/commit_message_change_log.html`

1.7 对 FAQ 和讲义作贡献

有突出贡献的同学可得 8% 左右的总评鼓励

- 课程 FAQ
  - `https://git.tsinghua.edu.cn/physics-data/faq/`
  - 提出好问题，整理问题解答，提交 merge request，……
- 课程讲义
  - 修正错别字、语言错误、常识错误
  - 指出逻辑不明的地方
  - 补充课堂讲授但讲义遗漏的内容
  - ……
- 贡献方式
  - ☒ Git
  - ☒ GitLab issue
  - ☐ GitLab Merge Request

分支

例子

- 关系
- merge request （合并请求）
  - 提交代码审核。

Git 仓库间通信的各阶段

fork Gitlab 类 Git 平台的术语，对应于 `git clone`

- 在本人帐号创建复本，用于自己修改和提交
- 提交说明的备注：关闭哪个 issue

remote Git 的远程仓库

```
git remote -v
```

```
origin  git@git.tsinghua.edu.cn:physics-data/lecture.git (fetch)
origin  git@git.tsinghua.edu.cn:physics-data/lecture.git (push)
```

pull, push 从远程下载和上传改动，单元为差分 commit

merge 与其他仓库或分支通过交换差分来整合

merge request GitLab 平台术语，发送 merge 请求。请他人整合自己的差分贡献。

第二章　Python 基础

2.1　准备与复习

预备

- 查看是否已经安装 Python

```
python3

Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

  - 退出输入 `exit()` 或者按 `Ctrl-d` 。
  - `Ctrl-d` 代表文件（标准输入）的终止。

- 如果没有 Python（可能性不大），或想升级 Python

```
apt install python3 # Debian @ WSL
emerge -vt python:3 # Gentoo Prefix @ macOS
```

黑客审美

- 当代文明的两大支柱是 实验 和 逻辑 。
- 四个原则：
  1. 复现 - 否则成伪科学
  2. 透明 - 否则玄学黑箱
  3. 一次 - 否则到处是坑
  4. 最佳工具 - 否则效率低下
- 推论：
  1. 兼容比性能优先

     > Premature optimization is the root of all evil. – Tony Hoare, Donald Knuth

  2. 人类时间比机器时间宝贵
  3. 使用工具进行版本控制

最佳工具

- GNU 环境、 POSIX 标准：用于促进程序有几十年跨度的兼容性。

Git：当代的版本控制

- 快照与差分
- 三个状态:
  - 已提交 committed （未改动 unmodified）
  - 已修改 modified
  - 待提交 staged

程序编辑器三大流派
请完成网络学堂的“课程问卷”，计划帮助大家掌握一款编辑器。

- Emacs
- Vi
- Visual Studio Code

2.2 Python 入门

为什么用 Python

- Python 是一门“解释型语言”，相对于“编译型语言”更易调试。
- Python 的语法风格简明，即使对外行也易读。
  - 书写效率高，快速写出不错的程序。
- Python 可以直接调用 Fortran, C/C++, R 等语言库，因此也叫“胶水”语言，即把不同的程序粘合在一起。
  - 易于与已有工具整合。
  - 促进团队分工，协作。
  - 大大丰富了 Python 生态系统的功能，进一步优化程序运行效率。
  - 符合 最佳工具 原则。
- Python 是一个通用语言，不仅在科学研究，在生活中的方方面面都会有用。
- 在科学计算领域得到广泛欢迎和采用。

  `https://www.python.org/`

特点和用途

- Python 是一门“解释型语言”，相对于“编译型语言”更易调试。
- Python 的语法风格简明，即使对外行也易读，大大降低了程序设计的门槛
- Python 可以直接调用 Fortran, C/C++, R 等语言库，因此也叫“胶水”语 言，即把不同的程序粘合在一起。
- Python 是一个通用语言，不仅在科学研究，在生活中的方方面面都会有用。
  - 操作系统生成器和管理器 (Gentoo Portage)
  - 网站 (Django)

参考资料

- Allen Downey, Think Python 2e

  简明通俗的入门书
- `http://py4e.com/`

  Python for everybody，全球知名的 Python 在线教程，新手友好。
- Learn X in Y minutes

  `https://learnxinyminutes.com/docs/python3/`

  已经掌握若干门语言的同学，可以通过此提纲快速入门

Python 环境

安装了 Python 之后，在命令行界面可以直接进入 Python 的交互模式：

```
$ python3
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

IPython （Jupyter 前身）增强的额外交互功能环境

```
# apt install ipython3
$ ipython3
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.5.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
```

查看 Python 的基本信息

```
import sys
print(sys.version)
```

```
3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0]
```

```
print("\n".join(sys.path))
```

```
/usr/lib/python311.zip
/usr/lib/python3.11
/usr/lib/python3.11/lib-dynload
/usr/local/lib/python3.11/dist-packages
/usr/lib/python3/dist-packages
/usr/lib/python3.11/dist-packages
```

Python 之禅

```
import this
```

```
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

算术基本运算

- + 加， - 减， * 乘， / 除， // 整除， % 取余， ** 乘方
- 其它运算由函数来定义

```
2+1
```

```
3
```

```
2*7, 2**7, 3/2, 3//2, 3%2
```

14 128 1.5 1 1

```
import math
math.factorial(10) # 10!
```

```
3628800
```

整数是高精度的

- 计算机上标准整数运算有界;
- 高精度整数并没有硬件的底层支持, 是 Python 的软件实现, 以一些性能损失为代价给予用户便利。

```
math.factorial(66)
```

```
54434493907744306400372924024784275264429306438879887453286012686967108114841600000000000000
```

```
2**100
```

```
1267650600228229401496703205376
```

整除的基本约定

负数整除：向小的方向截断。

```
5 // 3, -5 // 3, 5.0 // 3.0, -5.0 // 3.0
```

1 -2 1.0 -2.0

```
5 % 3, -5 % 3, 5.0 % 3.0, -5.0 % 3.0
```

2 1 2.0 1.0

```
(-5 // 3) * 3 + (-5 % 3) == -5
```

```
True
```

布尔运算：真与假

Python 的设计目标是“符合直觉”

```
not True, not False
```

False True

```
True and False, False or True
```

False True

```
True + True, True * False # True 实际上是 1, False 实际上是 0
```

2 0

```
True * 8, False - 5 # 只为了说明 True 和 False 的内部表示，不要这样写代码
```

8 -5

条件判断

- 等号写两次用于判断，写一次用于赋值。

```
1 == 1, 2 == 1
```

True False

```
1 != 1, 2 != 1
```

False True

```
1 < 10, 1 > 10, 2 <= 2, 2 >= 2
```

True False True True

数据类型

`int` 指整型，没有上限； `float` 是浮点型，一般为双精度； `str` 是字符串

```
type(1)
```

```
<class 'int'>
```

```
type(1.5)
```

```
<class 'float'>
```

```
type('Hello')
```

```
<class 'str'>
```

```
print(type("a"), type("你好")) # 单个字符和汉字都是字符串
```

```
<class 'str'> <class 'str'>
```

字符串

与高精度整数一样，字符串也没有硬件的对应，是 Python 的软件实现。这极大方便了使用 Python 进行文本处理。

```
"今天" + "要下雨"
```

```
今天要下雨
```

```
"1" + "2"
```

```
12
```

标准输入输出

- 标准输出默认与屏幕连接， `print()` 默认向标准输出写
- 标准输入默认与键盘连接， `input()` 默认从标准输入读

```
q = input() # 下面由现场输入
print(q)
```

```
frog
```

Python 的注释

- 注释使用 '#' 号引出，多行注释则多用几次 '#' 号

```
# 在这个程序中，我们将使用计算球谐函数对任意
# 球面上的连续函数进行拟合
```

正式入门 Python：输出 Hello World!

```
print("Hello World!")
```

```
Hello World!
```

创建脚本文件

世界上本没有脚本，把输入的命令记录下来就成了脚本。 Python 脚本一般以 .py 结尾。

使用编辑器创建 hello.py ，写入以下内容，第一行是标注脚本由什么来解释。

```
#!/usr/bin/env python3

print("Hello World!")
```

给予脚本可执行权限并执行

```
chmod +x hello.py
./hello.py
```

2.3 Python 变量

变量创建与使用

- Python 是“弱类型语言”，创建变量可不指定类型。

```
message = "This is an new era. 新时代"
print(message)
```

```
This is an new era. 新时代
```

- 变量在使用中可以改换类型

```
message = 1
type(message)
```

```
<class 'int'>
```

字符串函数

字符串的操作相对于硬件调用，是一项高级功能。Python 有强大的字符串处理工具。

```
len('123456'), len('654321') # 字符串长度
```

```
s = "我正在上课。"
s[0], s[2:4], s[-1] # 字符串可以取子串
```

我 在上 。

```
"啊" * 10 + "哈" * 20
```

啊啊啊啊啊啊啊啊啊啊哈哈哈哈哈哈哈哈哈哈哈哈哈哈哈哈哈哈哈哈

字符串与变量的联合操作

```
"{} 乘以 {} 等于 {}".format(3, 5, 3*5) # 把其它类型的值嵌入字符串
```

3 乘以 5 等于 15

f-string 用来把变量值嵌入字符串

```
b=50
f"b 的取值是 {b}"
```

b 的取值是 50

None 值

- None 是一个特殊的值，代表空集、无、无法表达或者非法的结果

```
None
```

None

```
x = None # None 可以被赋值
1 is None, x is None # 可以被判断
```

False True

- 使用场景：判断某个结果是否是 None 有助于我们了解操作是否成功

```
bool(None) # None 也可以当作“假”被判断
```

False

2.4 阅读英语提示

不要害怕英文提示

- 人生第一次遇到英语不是“屠龙之技”的场景，值得庆贺。
- 下一个场景：阅读英语科技文献。
- 再下一个场景：与国际同行讨论争吵。

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-f40f0b3453ee> in <module>
----> 1 message + 1

TypeError: can only concatenate str (not "int") to str
```

2.5 Python 的程序结构

程序结构

head to `Python-Constructs.slides`.

- 顺序结构
- 选择结构
- 循环结构

第三章　复合类型与函数

3.1　复习

课前准备

下载选课统计数据

```
wget http://hep.tsinghua.edu.cn/~orv/teaching/physics-data/departments.csv
```

Python 基础

- 基本数据类型和基本运算
  - 整型、浮点型和字符串
- 基本程序结构
  - 顺序、选择和循环结构
- Python 的脚本

```
#!/usr/bin/env python3

# Python 的语句
# ...
# ...
```

3.2　复合数据类型

简介

- Python 的基本类型，比如整型、浮点型、布尔型、字符串。
- 它们可以复合起来，成为 tuple（元组）、列表或字典。
  - `()`, `[]`, `{}`
  - 复合类型之间可以嵌套，变化丰富

3.3 列表 list

简介

- 把单元前后排成一列，单元为任意类型。

```
[1,2,3], ["天","地","人"], ["物理",3.1415926]
```

```
([1, 2, 3], ['天', '地', '人'], ['物理', 3.1415926])
```

```
[], [[]], [[],[[]]], [[],[[],[[]]]] # 可以构造自然数
```

```
([], [[]], [[], [[]]], [[], [[], [[]]]])
```

```
xl = [1, 2, 3]
xl[1]= 5
print(xl)
```

```
[1, 5, 3]
```

用于汇总

- 生成空列表出发，逐步加入元素

```
li = []
li.append("手机")
li.append("钥匙")
li.append("身份证")
print(li)
for i in li: # 作为迭代器
    print(f"出门之前，别忘了带{i}")
```

```
['手机', '钥匙', '身份证']
出门之前，别忘了带手机
出门之前，别忘了带钥匙
出门之前，别忘了带身份证
```

元素操作

取出

```
li[0], li[1:3], li[-1]
```

```
('手机', ['钥匙', '身份证'], '身份证')
```

判断是否属于

```
' 手机' in li, "眼镜" in li
```

```
(True, False)
```

3.4 元组 tuple

简介

- 元组与列表类似，区别在于它不可修改

```
(1,2,3), ("天时","地利","人和"), (),((()), ((),((())), ((),((),((())))
```

```
((1, 2, 3), ('天时', '地利', '人和'), (), (), ((), ()), ((), ((), ((()))
```

```
xt=(1,2,3)
xt[1]=5
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [11], in <module>
      1 xt=(1,2,3)
----> 2 xt[1]=5

TypeError: 'tuple' object does not support item assignment
```

无法像列表一样 append ，但可以重新生成

```
tu = tuple(li); print(tu)
tu += ("眼镜",); print(tu)  # 对比 list.append("眼镜")
```

```
('手机', '钥匙', '身份证')
('手机', '钥匙', '身份证', '眼镜')
```

```
"眼镜" in tu, "帽子" in tu
```

```
(True, False)
```

3.5 字典 dict(ionary)

简介

- 字典是一个可以索引的数据结构

```
工物 => 19
物理 => 16
致理 => 38
未央 => 19
```

```
sc = {' 工物': 19, ' 物理': 16}
print(sc[' 工物'], sc[' 物理'])
```

```
19 16
```

在字典中填加词条

```
sc[' 致理'], sc[' 未央'] = 38, 19 # 元组赋值
print(sc["致理"], sc[' 未央'])
```

```
38 19
```

判断是否属于

```
print(' 数学' in sc, ' 工物' in sc)
```

```
False True
```

简化条件语句

- 在需要多重判断时，使用字典会有奇效。

```
OS = "macOS"
package_manager = {
    "GNU/Linux": "请用 Debian 的 APT 工具",
    "macOS": "请用 UTM，然后安装 Debian",
    "Windows": "请安装 WSL，然后安装 Debian"
}

print("大佬，", end="")
if OS in package_manager:
    print(package_manager[OS])
else:
    print("壮士，请您到主席台上来")
```

```
大佬，请用 UTM，然后安装 Debian
```

迭代器基本操作

- 从字典中取出词和值。 `list()` 把迭代器转成列表。

```
list(sc.keys()), list(sc.values())
```

工物	物理	致理	未央
19	16	38	19

```
for k in sc:
    print(k)
for v in sc.values():
    print(v)
```

```
工物
物理
致理
未央
19
16
38
19
```

元组迭代器

- `items()` 返回“词”和“值”组成的二元组。

```
for k,v in sc.items():
    print(f"教室里有{k}学生{v}人")
```

```
教室里有工物学生19人
教室里有物理学生16人
教室里有致理学生38人
教室里有未央学生19人
```

字典中的数据类型

- 词 key 应当是“不可变” immutable 类型
  - 列表不能是字典的词
  - 元组可以是字典的词
  - 字典不能是字典的词

```
坐标系 = {(0,0): "原点", (1,1): "第一象限", (-1,1): "第二象限"}
print(坐标系[(0,0)])
```

```
原点
```

- 值 value 可以是任意类型

3.6 集合 set

简介

- 集合具有互异不重复性，对其做添加操作时自动滤除重复元素。
- 集合的取交、并和差集等运算应用广泛。

创建集合

```
A = set(range(4))
B = {0, 1, 2, 3}
print(A, B)
```

```
{0, 1, 2, 3} {0, 1, 2, 3}
```

```
print(type({}), type(set()), type(dict())) # {} 是空字典
```

```
<class 'dict'> <class 'set'> <class 'dict'>
```

元素操作

```
A.add(7) # 加元素进集合
A.discard(3) # 从集合除去元素
print(A)
```

```
{0, 1, 2, 7}
```

```
A.clear() # 集合清空
print(A)
```

```
set()
```

```
B.pop() # 随机取出一个元素返回
```

```
0
```

```
B
```

1 2 3

集合运算

```
{1, 2} - {2, 3}  # 差集
```

1

```
print({1, 2}.union({2, 3}), # 并集
      {1, 2}.intersection({2, 3})) # 交集
```

```
{1, 2, 3} {2}
```

3.7 其它数据结构

defaultdict – 带有默认值的字典

```
k = "数学"

if k in sc:
    v=sc[k]
else:
    v=0
print(f"来自{k}的同学有{v}人")
```

来自数学的同学有0人

```
from collections import defaultdict

dsc = defaultdict(int)
dsc["工物"] = 42; dsc["物理"] = 52
print(dsc["工物"], dsc["数学"])
```

42 0

普通的字典不能强取

```
sc["数学"]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [39], in <module>
----> 1 sc["数学"]

KeyError: '数学'
```

Counter – 高效地计数

```
deps = ["物理","物理","工物","工物","物理"]
numbers = defaultdict(int)

for d in deps:
    numbers[d]+=1
print(numbers)
```

defaultdict(<class 'int'>, {'物理': 3, '工物': 2})

```
from collections import Counter

cnumbers = Counter(deps)
print(cnumbers)
```

Counter({'物理': 3, '工物': 2})

namedtuple – 带有元素名字的 tuple

```
from collections import namedtuple

Point = namedtuple("point", field_names=("x", "y"))
p = Point(1,2)
print(p)
```

```
point(x=1, y=2)
```

```
print(p.x, p.y)
print(p[0], p[1])
```

```
1 2
1 2
```

3.8 函数

简介

- 函数是程序的基本组成部分。
- 函数可以直观看作是多段代码组成的功能单元。
- 函数的输入输出。
- 函数方便代码复用，体现 一次 原则。

```
def add(x, y):
    print(f"x is {x} and y is {y}")
    return x + y  # Return values with a return statement
add(3, 5)
```

```
x is 3 and y is 5
8
```

函数定义互换操作

```
def swap(x, y):
    return y, x
a = '左'
b = '右'

print(a,b)
a, b = swap(a,b)
print(a,b)
a, b = b, a
print(a,b)
```

```
左 右
右 左
左 右
```

```
tmp = a; a = b; b = tmp # 对比
```

迭代器函数调用

- 使用 ‘map‘ 可以把迭代器的映射到另一个迭代器

```
def squared(x):
    return x*x

list(map(squared, [1,2,3,4,5,6]))
```

```
[1, 4, 9, 16, 25, 36]
```

无名函数

函数名不重要时，可以使用无名函数

```
list(map(lambda x: x*x, range(6)))
```

名字空间 – 函数自己的变量存储空间

```
x = 1
def scope():
    x = 2
scope()
print(x)
```

```
1
```

强行使用全局变量（不推荐!）

```
def gscope():
    global x; x = 2
gscope()
print(x)
```

```
2
```

递归调用

- 考虑循环

```
n=123
while n>=1:
    print(n)
    n //= 2 # n = n // 2 的简写
```

```
123
61
30
15
7
3
1
```

递归调用（二）

- 等价于下列的递归调用

```
def div2(n):
    print(n)
    if n > 1:
        div2(n // 2)
div2(123)
```

```
123
61
30
15
7
3
1
```

3.9 命令行参数

命令行参数从终端向 Python 内部传递信息

```
import sys

print(sys.argv)
```

调用 sys 模块， sys.argv 是一个列表，内含程序调用的参数

```
$ python3 sys-demo.py 第一个参数 第二个参数
['sys-demo.py', '第一个参数', '第二个参数']
  ^sys.argv[0]  sys.argv[1]  sys.argv[2]
```

argparse 命令行传递参数

https://docs.python.org/3/library/argparse.html

- 构造高级的命令选项
  - 数据类型
  - 格式限制

3.10 文档

文档的重要性

- 保证程序对人类友好，实现“透明”原则。
- 对人友好，就是对自己友好，给自己的程序写笔记备忘。

己所不欲，勿施于人

- 我们学习别人的代码时，最讨厌的是“看不懂”，“没说明”。

函数文档

- 在函数定义后，紧跟一个字符串，可以定义函数的文档。
- 用多行字符串很方便。

```
def spherical_harmonic_fitter(grid, order):
    "求球谐函数拟合的系数"

    # 具体实现省略
    pass

help(spherical_harmonic_fitter)
```

```
Help on function spherical_harmonic_fitter in module __main__:

spherical_harmonic_fitter(grid, order)
    求球谐函数拟合的系数
```

多行文档

```
def spherical_harmonic_fitter(grid, order):
    '''
    求球谐函数拟合的系数

    输入
    ~~~
    grid: 球面上连续函数在固定格点上的取值
    order: 拟合时球谐函数近似截断的阶数
    '''
    pass
help(spherical_harmonic_fitter)
```

```
Help on function spherical_harmonic_fitter in module __main__:

spherical_harmonic_fitter(grid, order)
    求球谐函数拟合的系数

    输入
    ~~~
    grid: 球面上连续函数在固定格点上的取值
    order: 拟合时球谐函数近似截断的阶数
```

文档的普适性

- 任何 Python 的标准函数都有文档，大家都应认真写文档

```
help(None)
```

```
Help on NoneType object:

class NoneType(object)
 |  Methods defined here:
 |
 |  __bool__(self, /)
 |      self != 0
 |
 |  __repr__(self, /)
 |      Return repr(self).
 |
 |  ----------------------------------------------------------------------
 |  Static methods defined here:
 |
 |  __new__(*args, **kwargs) from builtins.type
 |      Create and return a new object.  See help(type) for accurate signature.
```

3.11 调试测试

程序调试

- 透明原则：程序运行的中间结果应当被人类理解
- 调试方法：
  - `print()`
  - REPL 试验
  - pdb, Python Debugger
    - `breakpoint()`
    - bt 给出函数调用关系
    - next 执行下一步
    - where 给出当前位置
    - list 列出附近的程序
    - help

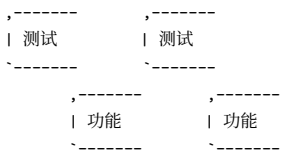
程序的正确性保证

- 如果保证修改程序不引入新的问题?
  - 修改越多，引入新问题的概率越大吗?

- 使用 Git 会有什么帮助吗?
  - 除了 Git 还需要什么?
- 实验室为何会有谁都不敢动的“祖传代码”出现?

编写测试数据

- 根据需求，用已知的输入输出检验程序是否正确
  - 单元测试：对函数进行测试
  - 集成测试：对各部分总体进行测试
- 测试程序与功能程序可以交替进行：



作业中的评分器是测试

- 评分器定义了功能，即程序期待得出的结果
  - 自动测试是目前为止，软件工程中最有效的保证程序可靠性的做法
  - 不仅保证程序逻辑正确，还要保证输入输出符合约定
- 可以在本地手动运行
- 也可以在线自动触发：课程使用的是 gitlab runner
  - 实现了持续集成（Continuous Integration）
  - 一旦哪个 commit 有了问题可以迅速通知：谁在什么时候弄坏了什么功能

实验的数据处理

- 上游程序的输出是下游程序的输入
- 程序仅仅被人类理解是不够的
- 如果不符合格式约定，后一个程序或者会崩溃，或者误读数据造成不易察觉的错误
- 一旦科学结果被发表，纠正错误将牵扯诸多非科学因素

测试框架可以促进协作

- 上游程序使用测试器模拟下游的读入
- 下游程序使用测试器模拟上游的输出
- 请主动设计测试输入样例，与同学分享刁钻的测试输入

3.12 代码风格

原则：人类阅读友好

There should be one– and preferably only one –obvious way to do it.

- 写程序，也只有一种推荐的风格
- 由 PEP-8 定义
  - PEP := Python enhancement proposal
  - `https://www.python.org/dev/peps/pep-0008/`
  - This document gives coding conventions for the Python code comprising the standard library in the main Python distribution.
- Python 哲学：Readability counts.

缩进

- 建议 4 个空格
- 避免一行过长，一切行必须在 80 字符以内
  - 人类读过长的行时，眼球不停转动，肌肉易疲劳
  - 限制行宽，方便把代码并列对比

```
# Aligned with opening delimiter.
foo = long_function_name(var_one, var_two,
                         var_three, var_four)

# 如果与开括号对齐导致空白过多，也可只加 4 个空格的缩进量。
def long_function_name(
        var_one, var_two, var_three,
        var_four):
    print(var_one)
```

多参数写法

```
my_list = [
    1, 2, 3,
    4, 5, 6,
    ]
result = some_function_that_takes_arguments(
    'a', 'b', 'c',
    'd', 'e', 'f',
    )
```

空格

- 恰当使用空格。过多或过少使用空格，都会使代码的可读性降低。
- 开括号之后和闭括号之前，不留空格。
- ",)" 之间不留空格

```
spam(ham[1], {eggs: 2})

foo = (0,)
```

- 逗号之前不留空格，之后最好由一个空格

```
a, b = b, a
```

空格：数学表达式

- 等号左右留空格
- 加号类左右留空格数应相等

```
i = i + 1
submitted += 1
x = x*2 - 1
hypot2 = x*x + y*y
c = (a+b) * (a-b)
```

- 但是，函数调用中的等号不留空格

```
magic(r=real, i=imag)
```

辅助工具

- pylint

```
apt install pylint # Debian
emerge -vt pylint # Gentoo
```

- 编辑器的自动重排功能
  - 学习一款编辑器非常重要：Emacs、Vi、Visual Studio Code

参考资料

- PEP-8
- FAQ 的相应说明

第四章　迭代器与数组

4.1　复习

课前准备

- 安装 NumPy

```
apt update
apt install python3-numpy
```

- 验证是否安装成功

```
python3 -c "from numpy import version; print(version.version)"
```

```
1.24.2
```

复合数据结构

- 列表、元组、字典
- `defaultdict` 、 `Counter` 、 `namedtuple`
- 迭代器可以逐个访问它所包含的值的。 Python 的复合数据结构都可以看成迭代器。

函数

- 定义，名字空间
- 局部变量，全局变量（不推荐！用类的变量替代）

4.2　类与对象

为什么

- Python 的内部实现
  - 一切都是对象

```
help(1)
```

给出一的是 help(int)，整数是一个类。

- 对象的要素：封装、继承、多态
- 迭代器：定义了 __iter__() 方法的类。
- 运算：

  ＋ __add__

  - __sub__

- 替代全局变量

极简例子

```
class status(object): # 继承自 "object"
    '''
    状态记录器
    '''
    def __init__(self, move=0, blank=(1,2,3)):
        self.move = move
        self.blank = blank
s = status(1)
print(s.move)
s.move = 3 # 一切成员都是 public
print(s.move)
```

```
1
3
```

- 将 s 传递给函数，避免使用全局变量

4.3 模块

简介

- 函数用来代码复用（一次 原则）
- 模块：相关的函数和类集合起来，整理到名字空间 namespace 中
  - 模块可以用 Python 实现，也可以由 C 等编译语言实现

模块导入用 import

```
import math
help(math.factorial)
```

```
Help on built-in function factorial in module math:

factorial(x, /)
    Find x!.

    Raise a ValueError if x is negative or non-integral.
```

模块别名

- 加载模块时，可自定义名称。对长模块名有用

```
import math as m
m.factorial(10)
```

```
3628800
```

多层名字加载

1. 直接使用多层名字空间
2. 使用 `from`

```
import os
help(os.path.abspath)
```

```
Help on function abspath in module posixpath:

abspath(path)
    Return an absolute path.
```

```
from os.path import abspath
from os.path import abspath as absp
abspath is os.path.abspath, abspath is absp
```

```
(True, True)
```

自定义模块

- Python 可以方便地定义模块以进行代码复用
- 每个 Python 脚本都可以当作模块使用

```
cat physics_data/script.py
```

```
    def spherical_harmonic_fitter(grid, order):
        '''
        求球谐函数拟合的系数

        输入
        ~~~
        grid: 球面上连续函数在固定格点上的取值
        order: 拟合时球谐函数近似截断的阶数

        输出
        ~~~
        拟合系数矩阵
        '''

        # 具体实现省略
        pass
```

自定义模块（二）

```
from physics_data import script
help(script.spherical_harmonic_fitter)
```

```
Help on function spherical_harmonic_fitter in module physics_data.script:

spherical_harmonic_fitter(grid, order)
    求球谐函数拟合的系数

    输入
    ~~~
    grid: 球面上连续函数在固定格点上的取值
    order: 拟合时球谐函数近似截断的阶数

    输出
    ~~~
    拟合系数矩阵
```

4.4 Python 标准库模块

fractions 有理数才是良定义的

https://docs.python.org/3/library/fractions.html

```
from fractions import Fraction
Fraction(16, -10)
```

```
Fraction(-8, 5)
```

decimal 计算机对人类的妥协

https://docs.python.org/3/library/decimal.html

```
from decimal import *
print(Decimal(1) / Decimal(7))
print(1/7)
```

```
0.14285714285714285714285714286
0.14285714285714285
```

迭代器

- 迭代器是一个对象，允许逐个访问容器（如列表、元组、集合）中的元素，而不必知道容器的内部实现细节。
  - 实现“迭代器协议”，在类中定义 `__iter__()` 函数。

与容器（复合数据类型）区别

- 容器（如列表、元组、集合）是数据的 集合 ，而迭代器是遍历这些集合中元素的 方式 。
- 迭代器提供了一种统一的访问接口，无论容器类型如何，都可以通过迭代器进行逐个访问。
  - 例子：字典可以定义多种迭代器。
- 列表和元组的迭代器恰好是按顺序访问每个元素，因此容易与列表和元组本身混淆。

应用

- `for` 循环由迭代器实现。列表、元组等能从迭代器创建。

credit: 王希呈

itertools 高级迭代器变换

`https://docs.python.org/3/library/itertools.html`

- 丰富多样的迭代器操作，巧妙运用则功能强大。

```
import itertools as it

data = [3, 4, 6, 2, 1, 9, 0, 7, 5, 8]
list(it.accumulate(data, max))
```

```
[3, 4, 6, 6, 6, 9, 9, 9, 9, 9]
```

```
list(it.permutations(range(1, 4)))
```

```
[(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)]
```

itertools 取笛卡尔积

```
list(it.product('ABCD', repeat=2))
```

```
[('A', 'A'),
 ('A', 'B'),
 ('A', 'C'),
 ('A', 'D'),
 ('B', 'A'),
 ('B', 'B'),
 ('B', 'C'),
 ('B', 'D'),
 ('C', 'A'),
 ('C', 'B'),
 ('C', 'C'),
 ('C', 'D'),
 ('D', 'A'),
 ('D', 'B'),
 ('D', 'C'),
 ('D', 'D')]
```

itertools 取可重组合

```
tuple(it.combinations_with_replacement('ABCD', 2))
```

```
(('A', 'A'),
 ('A', 'B'),
 ('A', 'C'),
 ('A', 'D'),
 ('B', 'B'),
 ('B', 'C'),
 ('B', 'D'),
 ('C', 'C'),
 ('C', 'D'),
 ('D', 'D'))
```

itertools 使用 zip 合并两个迭代器

```
for i, s in zip(range(11), it.accumulate(range(11))):
    print(f"从 0 加到 {i} 的和是 {s}")
```

```
从0加到 0 的和是 0
从0加到 1 的和是 1
从0加到 2 的和是 3
从0加到 3 的和是 6
从0加到 4 的和是 10
从0加到 5 的和是 15
从0加到 6 的和是 21
从0加到 7 的和是 28
从0加到 8 的和是 36
从0加到 9 的和是 45
从0加到 10 的和是 55
```

itertools 过滤器

```
list(it.filterfalse(lambda n: n % 13, range(100)))
```

```
[0, 13, 26, 39, 52, 65, 78, 91]
```

4.5 文件读写

大批量输入输出

- input() print() 适合少量的信息传递
- 大批量的读写宜直接操作文件

文本文件读

- 文件是迭代器，逐行。每行是字符串。

```
import itertools as it
with open("departments.csv") as f_input:
    for l in it.islice(f_input, 5):
        print(l, end="")
```

```
数学系
工物系
致理书院
致理书院
探微书院
```

- with 用来帮助在文件用完后及时关闭，防止占用和争夺资源。

```
from collections import Counter
with open("departments.csv") as f_input:
    print(Counter(f_input))
```

```
Counter({'致理书院\n': 30, '工物系\n': 20, '未央书院\n': 20, '物理系\n': 7, '上海交大\n': 4, '数学系\n': 1, '探微书院\n': 1})
```

文本文件写

```
with open("log.txt", "w") as f:
    f.write("第一天 概论\n")
    f.write("第二天 Python 基础\n")
```

```
cat log.txt
```

```
第一天 概论
第二天 Python 基础
```

字符处理

- Python 内建了丰富的字符处理函数

```
s = "今天的气温是 30 摄氏度，明天是 29 摄氏度"
print(s.count("度"))
print(s.startswith("今天"))
print(s.split("，"))
```

```
2
True
['今天的气温是 30 摄氏度', '明天是 29 摄氏度']
```

字符处理（二）

- 娱乐奥利奥生成器

```
seed = bin(2324)
print(seed)
print(seed[2:].replace('0',"奥").replace('1',"利"))
```

```
0b100100010100
利奥奥利奥奥奥利奥利奥奥
```

参考资料

- `help(str)`
- `https://docs.python.org/3.9/library/stdtypes.html#textseq`

4.6 寻找工具

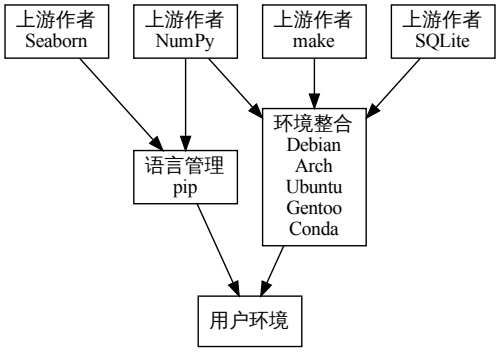
更多 Python 扩展库

- 优先使用 APT (Debian) 软件管理器

```
apt install python3-xxxxxx # Debian
```

- 混合 APT 与 pip 要极其小心，尽量不引入 pip
  - 提 issue 求助
  - 不要使用 conda

软件分发与环境整合：概念



- `pip` 没有整合测试，只适合早期试验个别最新 Python 软件，长期维护性差。
- 整体生产方案对应多种软件的稳定版本，经过应跨语言整合测试。

环境整合分发方案

- 第三方测试整合提供环境，防止被某上游垄断并偷藏私货
  - Malicious PyPI package opens backdoors on Windows, Linux, and Macs

    ```
    Allows a remote attacker to gain unauthorized access to the application.
    ```

  - `pip` 的软件没有经过第三方测试验证，使用意味者相信上游软件作者。
  - 环境整合方案会有负责安全、兼容和性能的团队，有大量同样环境的用户反馈问题。
- Debian/Arch/Gentoo 志愿者形态与 Ubuntu/Conda 公司免费形态
  - 志愿者形态的开发团队没有利益关系，人们由于共同的兴趣走到一起。
    - 无人能强制整个项目的走向，集体决策。
  - 公司免费形态利用免费服务吸引用户，构建潜在的客户池或者暗藏广告潜移默化改造用户。
    - 产品服务于公司的盈利或扩大影响的战略，代码由雇工产生。

不推荐用作科学计算和数据分析

- Ubuntu 的源代码 95% 从 Debian （合法）复制，外加公司定制和广告
- Conda 的依赖关系的计算效率极低，但在公司层面调动外宣经费公关推广

只有普通用户权限

- 安装和升级软件需要系统管理员权限，具体体现为在 `apt` 前加 `sudo` 。
- 课题组共享的服务器、超算中心的登录节点无权限怎么办？用户态软件管理

	成熟的管理工具	去中心研发	海量软件+物理学	超算中心采用	续本达召集
Gentoo Prefix	✓	✓	✓	✓	✓ ✓
nix/guix	✓	✓	✓		
Spack	✓			✓	
conda / mamba					

参考

Benda Xu, G. Amadio, F.Grffen, and M. Haubenwallner. “Gentoo Prefix as a Physics Software Manager.” EPJ Web of Conferences 245 (2020): 05036.

推荐：软件管理与环境配置

- 自己的机器用 *Debian*：工具安装用 `apt`
- 别人的机器用 *Gentoo Prefix*：工具安装用 `emerge`

4.7 Python 科学计算

NumPy: Numeric Python

- NumPy 起源于使用 Python 语言调用 fortran 进行线性代数运算的需求。
- 已经发展成为 Python 科学计算的基石
- 参考书：Scipy Lecture Notes

安装 NumPy 和相关工具

```
apt install python3-numpy python3-scipy python3-h5py
```

SciPy: Scientific Python

- NumPy 定义高效的数据结构
- SciPy 在 NumPy 的基础上提供的数值计算算法

scipy.cluster Vector quantization / Kmeans

scipy.constants Physical and mathematical constants

scipy.fftpack Fourier transform

scipy.integrate Integration routines

scipy.interpolate Interpolation

scipy.io Data input and output

scipy.linalg Linear algebra routines

scipy.ndimage n-dimensional image package

scipy.odr Orthogonal distance regression

scipy.optimize Optimization

scipy.signal Signal processing

scipy.sparse Sparse matrices

scipy.spatial Spatial data structures and algorithms

scipy.special Any special mathematical functions

scipy.stats Statistics

4.8 NumPy 数组

创建和索引数组

```
import numpy as np

nv = np.array([1,2,3,4,3,2,1])
print(nv, nv[2], nv[5:])
print(nv[-1], nv[::2])
```

```
[1 2 3 4 3 2 1] 3 [2 1]
1 [1 3 3 1]
```

数组语法与列表相似，可能相互转换，区别在于：

- 数组要求元素的数据类型被预设且一致，列表 (List) 无此要求
- 数组的存储是一段连续的内存空间，列表不是
- 以上两点使得在数值计算中，数组的效率比列表高很多

二维数组用来表示矩阵

```
ma = np.array([[1,0], [0,1]])
print(ma)
```

```
[[1 0]
 [0 1]]
```

```
type(ma), ma.shape
```

```
(numpy.ndarray, (2, 2))
```

常数数组的创建

```
print(np.ones((3, 3)))
print(np.zeros((4, 4)))
```

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

随机矩阵的创建

```
print(np.random.rand(2,5))
```

```
[[0.80632296 0.53845746 0.00714721 0.84843748 0.32654466]
 [0.56930226 0.35817586 0.45108818 0.14484227 0.65463473]]
```

创建单位矩阵

- np.eye 读与大写字母 I 相同的读法

```
e = np.eye(4)
print(e)
```

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

对角元

- 取矩阵对角元

```
print(np.diag(e))
```

```
[1. 1. 1. 1.]
```

- 由对角元生成对角矩阵

```
print(np.diag(np.arange(5)))
```

```
[[0 0 0 0 0]
 [0 1 0 0 0]
 [0 0 2 0 0]
 [0 0 0 3 0]
 [0 0 0 0 4]]
```

叠放

```
np.tile(e,2)
```

```
array([[1., 0., 0., 0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0., 1., 0., 0.],
       [0., 0., 1., 0., 0., 0., 1., 0.],
       [0., 0., 0., 1., 0., 0., 0., 1.]])
```

4.9 多维索引

数组的形状

- 创建一个 10×10 的数组

```
m = np.arange(100, dtype=int)
print(m)
m.shape = (10, 10)
print(m)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95
 96 97 98 99]
[[ 0  1  2  3  4  5  6  7  8  9]
 [10 11 12 13 14 15 16 17 18 19]
 [20 21 22 23 24 25 26 27 28 29]
 [30 31 32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47 48 49]
 [50 51 52 53 54 55 56 57 58 59]
 [60 61 62 63 64 65 66 67 68 69]
 [70 71 72 73 74 75 76 77 78 79]
 [80 81 82 83 84 85 86 87 88 89]
 [90 91 92 93 94 95 96 97 98 99]]
```

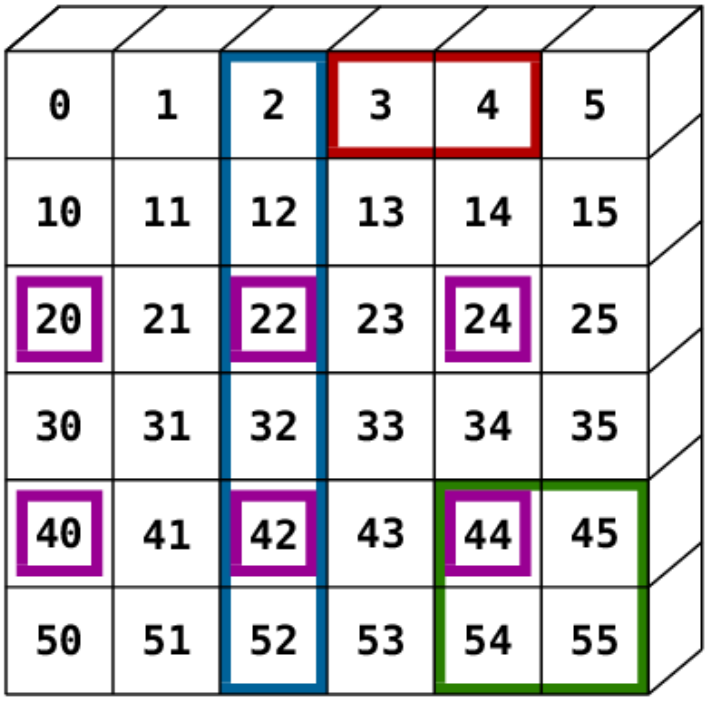
- 修改 `shape` 可以把一维数组解释成二维的

二维索引

```
m6 = m[:6, :6]
m6[:, 2]
```

```
array([ 2, 12, 22, 32, 42, 52])
```

```
>>> a[0, 3:5]
array([3, 4])

>>> a[4:, 4:]
array([[44, 45],
       [54, 55]])

>>> a[:, 2]
a([2, 12, 22, 32, 42, 52])

>>> a[2::2, ::2]
array([[20, 22, 24],
       [40, 42, 44]])
```



二维任意索引

- 分别传两个相等形状的索引数组，按索引数组形状排列对应的输出

```
print(m6)
print(m6[[1, 2, 3], [3, 4, 5]])
print(m6[[[1], [2]], [[3], [4]]])
```

```
[[ 0  1  2  3  4  5]
 [10 11 12 13 14 15]
 [20 21 22 23 24 25]
 [30 31 32 33 34 35]
 [40 41 42 43 44 45]
 [50 51 52 53 54 55]]
[13 24 35]
[[13]
 [24]]
```

4.10 数组运算

整体运算

- 一般的运算符都可以在数组上使用
- 可以省去循环，使用程序比 map 和列表生成更简明易懂

```
n = np.arange(10)
print(n**2)
print([v**2 for v in n]) # 对比
```

```
[ 0  1  4  9 16 25 36 49 64 81]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

索引与运算结合

```
nr = n[::-1]
print(nr + n)
print(n[::2] + 100)
```

```
[9 9 9 9 9 9 9 9 9 9]
[100 102 104 106 108]
```

数组的总体特征

- 求和、平均、中位数、方差、标准差

```
np.sum(n), np.mean(n), np.median(n), np.var(n), np.std(n)
```

```
(45, 4.5, 4.5, 8.25, 2.8722813232690143)
```

- 选择性地对二维数组的某个维度求值，省去两重循环。

```
np.sum(m, axis=0), np.median(m, axis=1)
```

```
(array([450, 460, 470, 480, 490, 500, 510, 520, 530, 540]),
 array([ 4.5, 14.5, 24.5, 34.5, 44.5, 54.5, 64.5, 74.5, 84.5, 94.5]))
```

数组的扩展

- 把一维数组扩展成二维，补齐形状

```
n[None, :] + n[:, None]
```

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10],
       [ 2,  3,  4,  5,  6,  7,  8,  9, 10, 11],
       [ 3,  4,  5,  6,  7,  8,  9, 10, 11, 12],
       [ 4,  5,  6,  7,  8,  9, 10, 11, 12, 13],
       [ 5,  6,  7,  8,  9, 10, 11, 12, 13, 14],
       [ 6,  7,  8,  9, 10, 11, 12, 13, 14, 15],
       [ 7,  8,  9, 10, 11, 12, 13, 14, 15, 16],
       [ 8,  9, 10, 11, 12, 13, 14, 15, 16, 17],
       [ 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]])
```

- 代替二重循环

```
np.array([x + y for x in n for y in n]).reshape(10, 10)
```

第五章　矩阵与张量运算

5.1　小助教与互助限度

小助教的职责范围

- 将本组同学的共性问题及时反馈给助教和老师，帮助老师根据实际情况调整教学方案。
- 帮助同学掌握课程内容，协助助教和老师答疑。

什么程度的帮助与借鉴可以促进学习

1. AB 讨论解决问题的思路；
2. AB 讨论关键的技巧，A 给出参考资料，引导 B 自学；
3. A 将自己的程序逐句用人类语言描述给 B，B 按照自己的理解写了一份逻辑一模一样的程序；
4. A 将自己的源码交给 B，B 在理解了原理之后丢掉了 A 的程序，自己凭记忆和理解重写了一份；
5. A 将自己的源码交给 B，B 在读懂之后进行了简单的变量名和函数定义替换；
6. A 将自己的源码交给 B，B 把程序原样提交。

抄袭零容忍

- 抄袭 过度借鉴代码
- 作业的意义
    1. 学生验证自己是否掌握课程内容
    2. 为课程分数提供客观教学评价依据
- 课程分数的意义
    1. 反馈学生对课程的掌握程度
    2. 未来的导师参考

本课程对抄袭的处理规则

1. 抄袭者与被抄袭者同样按抄袭处理，视情节严重程度
    - 警告，该次作业记 0 分；
    - 严重警告，该次作业记 0 分，倒扣 100%-200% 本次作业分数；
    - 总评记 F，报教务部门。

案例

- A 同学帮助 B 同学，面临交作业截止，B 同学还是没法及时完成；
- A 同学把代码给 B 同学参考；
- 虽然 B 向 A 承诺只是“参考”，但由于时间紧迫 B 把代码稍加修改，提交上去；
- 因为代码雷同，两位同学都疑似抄袭。

B

- 作业即使迟交，也不要复制其他同学的代码。
- 迟交作业不会在网络学堂上得分，但可以作为总评时的参考。
- 消化理解课堂样例代码、助教参考代码，形成自己的方法。
- 如果感到吃力，应当在课后投入更多精力尽更大的努力地练习。

A

- 请继续帮助同学，注意授人以渔；
- 请保持对同学的信任。

参考资料

学术诚信规则、更多案例与具体解读，详见：
`https://physics-data.meow.plus/faq/plagiarism/`

本人心目中的理想给分梯度

D 学会了用 Git，以 Git 提交了大多数作业。理解了数据处理的复现、透明、一次和最佳工具原则。

C 在同学、小/助教、老师的帮助下完成作业的基本要求。

B 遇到困难，可以独立分析出错信息，独立查清原因并解决。遇到知识盲点，可以通过自学围绕实现需要快速入门。

A 独立思考，提出大作业的创新解决方法，效果出众。

A+ 解决大作业中涉及的科学界的开放问题。

5.2 复习

课前准备

下载 asym.py

https://git.tsinghua.edu.cn/physics-data/lecture/-/blob/master/asym.py

保存到课堂练习的路径

Python 的模块

模块是一系列相关的变量、函数和类的集合，可以由程序复用。使用 import, from, as 等调用

```
import fractions
import fractions as frcs
from fractions import Fraction
from fractions import Fraction as Frc
```

模块的例子

fractions 无限精度分数

decimal 十进制小数

itertools 迭代器

NumPy 数值计算

SciPy 科学计算

单 Python 脚本式模块

```
from asym import eps

eps(4).shape
```

4 4 4 4

文件读写

- 文本文件在 Python 中被抽象成迭代器。
- 使用 with 让与外部资源交互（如文件）的代码更简洁。

```
from collections import Counter
with open("departments.csv") as f_input:
    print(Counter(f_input))
```

```
Counter({'致理书院\n': 30, '工物系\n': 20, '未央书院\n': 20, '物理系\n': 7, '上海交大\n': 4, '数学系\n': 1, '探微书院\n': 1})
```

NumPy

- 数组
  - 与元组、列表的异同
  - 创建
  - 下标的范围
  - 下标的扩展
  - 替代循环

```
import numpy as np

n = np.arange(10)
n**2
```

0 1 4 9 16 25 36 49 64 81

5.3 矩阵乘法

元素的乘法是逐个数字相乘。

元素乘法与矩阵乘法的区别

- 矩阵乘法是张量缩并的特例

$$\sum_j a_{ij}b_{jk} = c_{ik}$$

- 元素乘法是逐个数字相乘

$$a_{ij}b_{ij} = c_{ij}$$

```
a = b = np.ones((2,2))
print(a * b)
print(a @ b)
```

```
[[1. 1.]
 [1. 1.]]
[[2. 2.]
 [2. 2.]]
```

张量乘法

- `tensordot`， 可标注哪些下标进行缩并

```
a = np.arange(60.).reshape(3,4,5)
b = np.arange(24.).reshape(4,3,2)
np.tensordot(a,b, axes=([1,0],[0,1]))
```

$$\begin{matrix} 4400 & 4730 \\ 4532 & 4874 \\ 4664 & 5018 \\ 4796 & 5162 \\ 4928 & 5306 \end{matrix}$$

爱因斯坦约定

- einsum

```
a = np.arange(25).reshape(5,5)
b = np.arange(5)
np.einsum('ii', a) # 取迹
```

```
60
```

```
np.einsum('ij->i', a) # 对 'j' 方向求和
```

$$\begin{matrix} 10 & 35 & 60 & 85 & 110 \end{matrix}$$

```
np.einsum('ij,j', a, b) # 矩阵乘法
```

$$\begin{matrix} 30 & 80 & 130 & 180 & 230 \end{matrix}$$

爱因斯坦约定用于复杂张量运算

```
a = np.arange(60.).reshape(3,4,5)
b = np.arange(24.).reshape(4,3,2)
np.einsum('ijk,jil->kl', a, b)
```

$$\begin{matrix} 4400 & 4730 \\ 4532 & 4874 \\ 4664 & 5018 \\ 4796 & 5162 \\ 4928 & 5306 \end{matrix}$$

灵活缩并

```
a = np.ones(64).reshape(2,4,8)
np.einsum('ijk,ilm,njm,nlk',a,a,a,a)
```

```
4096.0
```

5.4 矩阵运算

Pauli 矩阵的定义

- 在量子力学中为自旋 $\frac{1}{2}$ 态空间下的角动量算符表示。

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Pauli 矩阵的对易关系

$$\sigma_1\sigma_2 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix} = i \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = i\sigma_3$$

同理,

$$\sigma_2\sigma_1 = -i\sigma_3\sigma_2\sigma_3 = i\sigma_1\sigma_3\sigma_2 = -i\sigma_1\sigma_3\sigma_1 = i\sigma_2\sigma_1\sigma_3 = -i\sigma_2$$

Pauli 矩阵

- 构造三个 Pauli 矩阵

```
pauli = []
pauli.append(np.array([0,1,1,0]).reshape(2,2))
pauli.append(np.array([0,-1j,1j,0]).reshape(2,2))
pauli.append(np.array([1,0,0,-1]).reshape(2,2))
for m in pauli:
    print(m)
```

```
[[0 1]
 [1 0]]
[[ 0.+0.j -0.-1.j]
 [ 0.+1.j  0.+0.j]]
[[ 1  0]
 [ 0 -1]]
```

计算迹和本征值

```
for m in pauli:
    print("trace is {};".format(np.trace(m)), "eigenvalues are: ", np.linalg.eigvals(m))
```

```
trace is 0; eigenvalues are:  [ 1. -1.]
trace is 0j; eigenvalues are:  [ 1.+0.j -1.+0.j]
trace is 0; eigenvalues are:  [ 1. -1.]
```

验证对易关系

- $[\sigma_i, \sigma_j] = 2i\epsilon_{ijk}\sigma^k$
  - (ϵ_{ijk}) 是全反称张量，指标有相同时为 0，指标奇排列为 -1 ，偶排列时为 1。

```
def commute(a,b):
    '''
    commutation operator: ab-ba
    '''
    return a@b - b@a

for i in range(3):
    l = (i+1) % 3
    m = (i+2) % 3
    if np.all(commute(pauli[i], pauli[l]) == 2j * pauli[m]):
        print(f"[ pauli_{i} , pauli_{l} ] == 2i pauli_{m}")
```

```
[ pauli_0 , pauli_1 ] == 2i pauli_2
[ pauli_1 , pauli_2 ] == 2i pauli_0
[ pauli_2 , pauli_0 ] == 2i pauli_1
```

课堂练习：彻底避免使用列表

- 构造 Pauli 矩阵时使用了列表和 for 循环，不够简洁
- 直接构造张量计算，省去交换关系验证中的循环
- 书写脚本解决

Pauli 矩阵的反对易关系验证

$$\{\sigma_i, \sigma_j\} = 2\delta_{ij}I$$

5.5 Dirac 矩阵

进阶练习：Dirac 矩阵的对易关系

$$\gamma^0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}, \quad \gamma^1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}$$

$$\gamma^2 = \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ -i & 0 & 0 & 0 \end{pmatrix}, \quad \gamma^3 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$$\{\gamma^\mu, \gamma^\nu\} = 2\eta^{\mu\nu} I$$

- $\{A, B\} = AB + BA$ 是 反对易子
- $\eta^{\mu\nu}$ 是闵可夫斯基度规 `np.diag((1, -1, -1, -1))`

Dirac 矩阵代数

- 定义 $\gamma^5 \equiv i\gamma^0\gamma^1\gamma^2\gamma^3$
- $\gamma_\mu = \eta_{\mu\nu}\gamma^\nu$
  - $\eta_{\mu\nu}$ 也是是闵可夫斯基度规 `np.diag((1, -1, -1, -1))`

NumPy 编程验证：

- γ^5 是厄米阵
- $(\gamma^5)^2 = I$
- $\{\gamma^5, \gamma^\nu\} = 0$
- $\gamma^\nu\gamma_\nu = 4I$
- $\gamma^\mu\gamma^\nu\gamma^\rho\gamma_\mu = 4\eta^{\nu\rho} I$
- $\mathrm{Tr}(\gamma^\mu) = 0$
- $\mathrm{Tr}(\gamma^\mu\gamma^\nu) = 4\eta^{\mu\nu}$
- $\mathrm{Tr}(\gamma^5) = 0$

Dirac 矩阵

```
import numpy as np
pauli = np.array([[[0,1],[1,0]],
                  [[0,-1j],[1j,0]],
                  [[1,0],[0,-1]]])
gamma0 = np.diag((1,1,-1,-1))
gamma2p = np.diag((-1j,1j,1j,-1j))
dirac = np.array((gamma0,
                  gamma0[:, ::-1],
                  gamma2p[:, ::-1],
                  np.roll(np.diag((1,-1,-1,1)), 2, axis=1)))
```

验证反对易关系

```
def anticomm(x, y):
    '''
    反对易子操作
    ===========
    传入 x（二阶） 和 y（三阶）张量，
    其后两阶可以解读为矩阵。

    返回，xy + yx 的三阶张量值。
    '''
    return x @ y + y @ x
```

Dirac 的反对易关系

```
mkov = np.diag((1,-1,-1,-1))
np.all(anticomm(dirac[None], dirac[:, None]) == 2 * np.einsum('ij,xz->ijxz', mkov, np.eye(4)))
```

```
True
```

Γ^5 是 Hermite 阵

```
gamma5 = 1j * dirac[0] @ dirac[1] @ dirac[2] @ dirac[3]
np.all(np.conj(gamma5.T) == gamma5)
```

```
True
```

Γ^5 平方是单位阵

```
np.all(gamma5 @ gamma5 == np.eye(4))
```

```
True
```

Γ^5 与 Γ^μ 的反对易关系

```
np.all(anticomm(gamma5[None], dirac) == 0)
```

```
True
```

归一性

```
np.all(np.einsum('ixy,ij,jyz->xz', dirac, mkov, dirac) == 4 * np.eye(4))
```

```
True
```

四 Γ 同框

```
np.all(np.einsum('xab,jbc,kcd,xy,yde->jkae', dirac,dirac,dirac,mkov,dirac)
       == 4 * np.einsum('ij,xy->ijxy', mkov, np.eye(4)))
```

```
True
```

迹

```
np.all(np.einsum('ixx', dirac) == 0)
```

```
True
```

```
np.trace(gamma5)
```

```
0j
```

```
np.all(np.einsum('ixy,jyx', dirac, dirac) == 4 * mkov)
```

```
True
```

5.6 备用

同学们的建议反馈（一）

- 希望可以多一点写代码的指导

  课上的例子是写代码的指导。如果不够，同学之间和小助教多讨论切磋。

- 希望自己学好，谢谢助教和老师（约占 40%）

  好！我们一起努力克服困难。

- 希望学到更多对实验物理研究有用的工具

  好！这是本人十几年来一直不变的初心。

- 希望作业能简单一点，分高一点

  作业的难度和给分高低服务于课程目标的设计。

  世界上本没有水课，太简单分太高就成了水课。

同学们的建议反馈（二）

- 我实在是太菜了，助教总是凶巴巴，可以温柔一点吗?

  有些人明白他们不该粗鲁或傲慢的提问并要求得到答复，但他们选择另一个极端 —低声下气：我知道我只是个可悲的新手，一个撸瑟，但...。这既使人困扰，也没有用，尤其是伴随着与实际问题含糊不清的描述时更令人反感。

  别用原始灵长类动物的把戏来浪费你我的时间。取而代之的是，尽可能清楚地描述背景条件和你的问题情况。这比低声下气更好地定位了你的位置。

  —Eric Raymond，提问的智慧

“菜”问题

- 解决未知问题是本课程的教学目标，也是数据时代黑客技能的基本组成部分，不要放弃宝贵的学习机会。
- 希望同学在提问之前，做一些努力，尝试先自己解决问题。
- 查找 FAQ 和群聊天记录，可能已经有人遇到并解决了同样问题。
- 提问时，把问题描述清楚，不要说自己“菜”“弱”。
- 不要遇到问题不假思索随手拍照，请先思考分析原因。
- 老师一直在暗中观察，会及时制止助教无端以不好的态度与同学对话。但是本人看到的所有助教语气激烈的情形，都是同学无视了以上的建议。
- 人非生而知之者，基础弱可以不断学习。但提菜问题必须批评。
- 幸亏助教只是凶一下，随后还是帮助了同学，如果遇到老师……

课堂豁免

- 在课堂上，没有时间作系统调研，以上规则不适用。
- 课堂上遇到疑惑，请你大声发问！这也是在帮助同学和老师。

同学们的建议反馈（三）

- 作业难度跳跃太大

  是这样的。

- 从以往的大作业项目看，强烈建议大作业中给出更加详细的公式、推导过程和计算方法，以便没有选修过相关课程的同学可以更好地理解项目的背景知识、数值计算方法等。期待不要把作业中没有讲的东西都扔给自学。

  参考老师理解的给分梯度，请在遇到知识盲点时，通过自学补足。大作业的设计目标是模拟科研环境，客观世界从不在乎你已经学过什么。

- 希望讲授的时候能够慢一些（约占 15%）

  请在课堂上遇到困难时直接发问，本人会加入更多的反馈机制。

- 希望讲授的时候能够快一些（约占 10%）

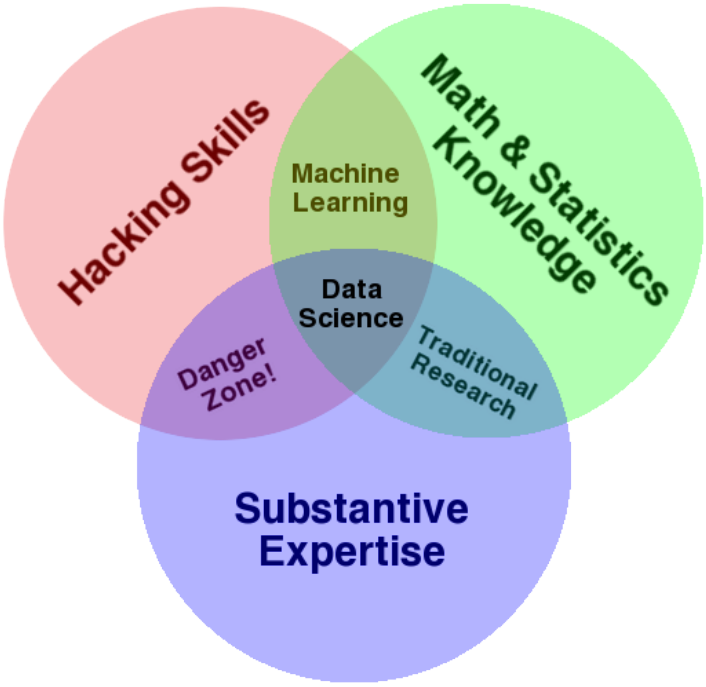
  请帮助周围的同学，或直接去解决大作业中的开放问题。

同学们的建议反馈（四）

- 希望能简单的列出之后的上课大纲，方便同学们预习

  好。

- 期待马上上手机器学习，希望学习神经网络



  - 炒菜式的机器学习非常简单，不用上课也可以学会。
  - 有了 Hacking skills 和数理统计的功底，机器学习才能发挥出作用。

同学们的建议反馈（五）

- 希望详细演示下 debug 的过程

  好！

- 希望了解更多的 linux 的骚气操作

  好！第三周。

- 希望老师多讲解每个知识点的具体应用的意义。少讲语法，突出物理

  好！

- 建议作业和课件在课程开始时就全部发放

  课件可以，作业课后发。

- 将所有同学问过的问题整理到某个地方。助教群里答疑是提供详细的说明，而非提一个从来没有听说过的关键词

  - https://physics-data.meow.plus/faq/
  - 你学习如何通过关键词提示找到答案

如何通过关键词提示找到答案

- 搜索引擎的用法演示

  https://zh.lmgtfy.com/

  https://www.google.com/

  http://www.bing.com/

  https://duckduckgo.com/

  其它的搜索引擎都强烈不推荐。

同学们的建议反馈（五）

- 希望老师多举一些例子，整理出来发给大家。优秀代码分享。

  好！

提问的智慧节选

- 如果你的问题被人无视了，请对号入座：
- 问题：我能在哪找到 X 程序或 X 资源?
  - 回答：就在我找到它的地方啊，白痴 —搜索引擎的那一头。天哪！难道还有人不会用 Google 吗?
- 问题：我怎样用 X 做 Y?
  - 回答：如果你想解决的是 Y，提问时别给出可能并不恰当的方法。这种问题说明提问者不但对 X 完全无知，也对 Y 要解决的问题糊涂，还被特定形势禁锢了思维。最好忽略这种人，等他们把问题搞清楚了再说。
- 问题：如何设定我的 shell 提示??
  - 回答：如果你有足够的智慧提这个问题，你也该有足够的智慧去 RTFM，然后自己去找出来。
- 问题：我可以用 Bass-o-matic 文件转换工具将 AcmeCorp 档案转换为 TEX 格式吗?
  - 回答：试试看就知道了。如果你试过，你既知道了答案，就不用浪费我的时间了。

提问的智慧节选（续）

- 问题：我的{程序/设定/SQL 语句}不工作
  - 回答：这不算是问题吧，我对要我问你二十个问题才找得出你真正问题的问题没兴趣 —我有更有意思的事要做呢。在看到这类问题的时候，我的反应通常不外如下三种
  - 你还有什么要补充的吗?
  - 真糟糕，希望你能搞定。
  - 这关我有什么屁事?

新规则

- 具体问题，请到 `https://git.tsinghua.edu.cn/physics-data/faq/-/issues/`
  方便遇到类似问题的同学快速找到。
- 当你的问题得到解答后，助教可能觉得此问题非常有价值。请配合助教发 Pull Request，将它收集到 `https://physics-data.meow.plus/faq/` 。

第六章　数据存储格式

6.1　复习准备

环境准备

- 加入 Debian testing 软件源。
  - Debian 中软件库的分类： `unstable`, `testing`, `stable`
    - 2024 年三者的代码分别为 `sid`, `trexie`, `bookworm`
  - Debian 12 是 `stable` ， 环境稳定，可堪重用
  - 新软件进入 Debian `unstable` 如果经过一星期没有问题，则转入 `testing`
  - 将 `testing` 定期整理发布成 `stable`
- `/etc/apt/sources.list`
- `/etc/apt/preferences.d/stable`

```
Package: *
Pin: release a=unstable
Pin-Priority: 200

Package: *
Pin: release a=testing
Pin-Priority: 300

Package: *
Pin: release a=stable-backports
Pin-Priority: 400
```

安装 HDF5、CSV 和 JSON 的相关工具

```
apt install vitables csvkit python3-h5py hdf5-tools jq python3-pandas
```

- `python3-fastparquet` 需要 `unstable` ， 未来将在 Debian 13 中。

```
apt install -t unstable python3-fastparquet python3-numpy
```

- 数据

```
wget 'http://hep.tsinghua.edu.cn/~orv/distfiles/C3--Trace--99996.txt.xz'
unxz C3--Trace--99996.txt.xz
wget 'http://hep.tsinghua.edu.cn/~orv/pd/BBH_events_v3.json'
wget 'http://hep.tsinghua.edu.cn/~orv/pd/4426232.json.xz'
unxz 4426232.json.xz
```

矩阵与张量运算

要善于使用 NumPy 的运算替代循环结构。

- `@`
- `tensordot`
- `einsum`

张量形式验证

Pauli 矩阵与 Dirac 矩阵的代数性质。

6.2 数据格式

在操作层面上，我们做实验的目标是取得数据，数据是实验结果的载体。有了数据才能获得科学结果。而数据格式的重要性，在于影响透明原则的贯彻。

透明原则

- 把大规模的数组高效地存储到磁盘上，是数据处理的关键。
  - 大时间尺度上看，数据 = 储存的文件
- 数据处理的结果，不仅在符合计算机的约定标准，还要对人类友好。
- 数据格式本质上是内存磁盘的双向数据转化，关键问题是
  - 转化过程中是否有损失?
  - 转化是否方便?

反例

- 违反透明原则。
- 只有某个操作系统上才能读取的数据文件，操作系统只能在某种计算机硬件上运行。

但世界上有诸多情况下，数据的格式会被无意或故意地做成不透明的。在此我们介绍四类推荐的透明数据格式。

自制二进制格式

- KamLAND，JUNO 中微子实验
- 问题
  1. 除了指定的 C++ 工具库，无法读取中间结果，违反“透明”原则
  2. 新成员必须学习非通用的数据接口和方法，门槛高且无用
  3. 数据格式与库深层绑定，必须同步升级，新库无法读取旧数据
- 解决方案：使用指定语言，以最小代码开发不透明数据到透明关系数据（例如 HDF5）的转换器
  - 数据格式转换器比函数跨语言调用更可靠，有效解耦合
  - 如 Python 调用 C++ 和 R 容易出现内存管理问题。

自制二进制格式（二）

- SuperK 中微子实验的 Zebra 格式
- 问题
  1. 格式已经被上游团队遗弃，除了指定的 Fortran 77 工具库，无法读取原始数据，违反“透明”原则
  2. 无法升级，必须维护旧计算环境才能运行 g77-3.4 （2004 年版本）
- 解决方案：使用新 Fortran 语言研发 Zebra 到 HDF5 的转换器

6.3 CSV

简介 comma separated values

- 文本文件，易于阅读
- 文本天然是一个表格
- 适合传递整数，文字或者对精度没有要求的浮点数
- CSV 工具

试验写 CSV

```
import numpy as np
hz = np.arange(100).reshape(10, 10)
print(hz)
```

```
[[ 0  1  2  3  4  5  6  7  8  9]
 [10 11 12 13 14 15 16 17 18 19]
 [20 21 22 23 24 25 26 27 28 29]
 [30 31 32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47 48 49]
 [50 51 52 53 54 55 56 57 58 59]
 [60 61 62 63 64 65 66 67 68 69]
 [70 71 72 73 74 75 76 77 78 79]
 [80 81 82 83 84 85 86 87 88 89]
 [90 91 92 93 94 95 96 97 98 99]]
```

```
np.savetxt("hz.csv", hz)
```

```
None
```

查看文件

- `cat` 命令，“concatenate” 的缩写，可以显示文件内容，即向标准输出写入。

```
cat hz.csv
```

```
0.00000000000000000e+00 1.00000000000000000e+00 2.00000000000000000e+00 3.00000000000000000e+00 4.00000000000000000e+00 5.00000000000000000e+00 6.00000
1.00000000000000000e+01 1.10000000000000000e+01 1.20000000000000000e+01 1.30000000000000000e+01 1.40000000000000000e+01 1.50000000000000000e+01 1.60000
2.00000000000000000e+01 2.10000000000000000e+01 2.20000000000000000e+01 2.30000000000000000e+01 2.40000000000000000e+01 2.50000000000000000e+01 2.60000
3.00000000000000000e+01 3.10000000000000000e+01 3.20000000000000000e+01 3.30000000000000000e+01 3.40000000000000000e+01 3.50000000000000000e+01 3.60000
4.00000000000000000e+01 4.10000000000000000e+01 4.20000000000000000e+01 4.30000000000000000e+01 4.40000000000000000e+01 4.50000000000000000e+01 4.60000
5.00000000000000000e+01 5.10000000000000000e+01 5.20000000000000000e+01 5.30000000000000000e+01 5.40000000000000000e+01 5.50000000000000000e+01 5.60000
6.00000000000000000e+01 6.10000000000000000e+01 6.20000000000000000e+01 6.30000000000000000e+01 6.40000000000000000e+01 6.50000000000000000e+01 6.60000
7.00000000000000000e+01 7.10000000000000000e+01 7.20000000000000000e+01 7.30000000000000000e+01 7.40000000000000000e+01 7.50000000000000000e+01 7.60000
8.00000000000000000e+01 8.10000000000000000e+01 8.20000000000000000e+01 8.30000000000000000e+01 8.40000000000000000e+01 8.50000000000000000e+01 8.60000
9.00000000000000000e+01 9.10000000000000000e+01 9.20000000000000000e+01 9.30000000000000000e+01 9.40000000000000000e+01 9.50000000000000000e+01 9.60000
```

- 不是很易读，因为 ‘np.savetxt()‘ 的默认格式是 ‘fmt=’%.18e’‘.
- ‘fmt=%d‘ 按整数输出。
  - 参考：C format specifiers

重新写 CSV

```
np.savetxt("hz.csv", hz, fmt="%d")
```

```
cat hz.csv
```

```
0 1 2 3 4 5 6 7 8 9
10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29
30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49
50 51 52 53 54 55 56 57 58 59
60 61 62 63 64 65 66 67 68 69
70 71 72 73 74 75 76 77 78 79
80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 95 96 97 98 99
```

读取 CSV

```
csv_hz = np.loadtxt("hz.csv")
print(csv_hz)
```

```
[[ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9.]
 [10. 11. 12. 13. 14. 15. 16. 17. 18. 19.]
 [20. 21. 22. 23. 24. 25. 26. 27. 28. 29.]
 [30. 31. 32. 33. 34. 35. 36. 37. 38. 39.]
 [40. 41. 42. 43. 44. 45. 46. 47. 48. 49.]
 [50. 51. 52. 53. 54. 55. 56. 57. 58. 59.]
 [60. 61. 62. 63. 64. 65. 66. 67. 68. 69.]
 [70. 71. 72. 73. 74. 75. 76. 77. 78. 79.]
 [80. 81. 82. 83. 84. 85. 86. 87. 88. 89.]
 [90. 91. 92. 93. 94. 95. 96. 97. 98. 99.]]
```

数据类型

- Numpy 数组需要指定数据类型
  - ‘np.int16‘, ‘np.int32‘, ‘np.int64‘, ‘int‘
  - ‘np.float16‘, ‘np.float32‘, ‘np.float64‘, ‘float‘
  - ‘np.complex64‘

类型的变化

```
print(hz.dtype, csv_hz.dtype)
```

```
int64 float64
```

- 数据类型从整数变成了浮点数!

正确的读取方法

```
ri_hz = np.loadtxt("hz.csv", dtype=int)
print(ri_hz)
print(ri_hz.dtype)
```

```
[[ 0  1  2  3  4  5  6  7  8  9]
 [10 11 12 13 14 15 16 17 18 19]
 [20 21 22 23 24 25 26 27 28 29]
 [30 31 32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47 48 49]
 [50 51 52 53 54 55 56 57 58 59]
 [60 61 62 63 64 65 66 67 68 69]
 [70 71 72 73 74 75 76 77 78 79]
 [80 81 82 83 84 85 86 87 88 89]
 [90 91 92 93 94 95 96 97 98 99]]
int64
```

- 需要额外信息辅助才能读入

总结 CSV 的特点

- 优点：简单直观，兼容性极强
- 缺点：需要每次都指定格式和读入时的数据类型
  - 只能表示表格
  - 因为兼容性强，可能被加上额外的非标准信息。
    - 例子： `C3--Trace--99996.txt`
- 是否可以把数据类型也存到文件中?
  - 人类还可以直接读取吗?
  - 另加约定是否可以成为标准?
  - → HDF5
- 表格之外的格式怎么办?
  - → JSON，具有额外的格式定义

6.4 HDF5

简介 Hierarchical Data Format

- 起源于高性能计算领域，目前由 The HDF Group 非盈利组织开发和维护
- 从 HDF 第 4 代起，获得广泛应用，特别是天文学
- 现在是第 5 代，因此叫做 HDF5
  - 原始表示：数据不必转换成文本。
    - 不涉及转换误差，但不再有文本文件的可读性。
  - 自我描述：数据类型写在文件中，可以被自动识别
  - 支持所有主流语言，有多种查看器
  - 缺点：对 ASCII 之外的字符支持没有标准，不保证可以处理中文。

HDF5 的结构

- 数据集 (Dataset): 多维数组
- 组 (Group): 数据集的容器
- 组可以嵌套，使用 '/' 分隔
  - `/calibration/water/waveform`
- 元数据 (Metadata)：用于描述数据集或组的特征

Python 的 HDF5 工具

- h5py: 极简的工具库，允许 Python 调用 HDF5 的 C++ 库。
  - 数据格式兼容性好，可以与其它语言交换数据。
- PyTables: 在 HDF5 之上进行了自定义格式，对读写有优化
  - 但是损失了兼容性；
  - 可以读入标准 HDF5 文件，但是容易不小心写出非标准 HDF5 文件。
- 课程选择 h5py，当兼容性和性能冲突时，优先选择兼容性。
  - “透明”原则。

写 HDF5 文件

```
import h5py

with h5py.File("hz.h5", "w") as opt:
    opt["hz"] = hz
```

```
HDF5 "hz.h5" {
GROUP "/" {
   DATASET "hz" {
      DATATYPE  H5T_STD_I64LE
      DATASPACE  SIMPLE { ( 10, 10 ) / ( 10, 10 ) }
   }
}
}
```

- 注意写入风格与 CSV 的异同
- `h5py.File` 返回的 opt 可以看作一个字典。

读 HDF5 文件

```
with h5py.File("hz.h5") as ipt:
    h5_hz = ipt["hz"][...]
print(h5_hz)
```

```
[[ 0  1  2  3  4  5  6  7  8  9]
 [10 11 12 13 14 15 16 17 18 19]
 [20 21 22 23 24 25 26 27 28 29]
 [30 31 32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47 48 49]
 [50 51 52 53 54 55 56 57 58 59]
 [60 61 62 63 64 65 66 67 68 69]
 [70 71 72 73 74 75 76 77 78 79]
 [80 81 82 83 84 85 86 87 88 89]
 [90 91 92 93 94 95 96 97 98 99]]
```

- `[...]` 或 `[()]` 代表把数据全部读入内存。
- 如果内存不够，h5py 提供了部分读入的功能，也叫做 ”out of core computing”。

在文件上直接操作

- 不在 `with` 环境有额外的功用

```
ipt = h5py.File("hz.h5")
ipt["hz"][2::3, ::5]
```

```
array([[20, 25],
       [50, 55],
       [80, 85]])
```

```
ipt.close() # 关闭之后就无法使用了
ipt["hz"][2::3, ::5]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [110], in <module>
      1 ipt.close() # 关闭之后就无法使用了
----> 2 ipt["hz"][2::3, ::5]

File h5py/_objects.pyx:54, in h5py._objects.with_phil.wrapper()

File h5py/_objects.pyx:55, in h5py._objects.with_phil.wrapper()

File /usr/lib/python3.9/site-packages/h5py/_hl/group.py:305, in Group.__getitem__(self, name)
    303         raise ValueError("Invalid HDF5 object reference")
    304 elif isinstance(name, (bytes, str)):
--> 305     oid = h5o.open(self.id, self._e(name), lapl=self._lapl)
    306 else:
    307     raise TypeError("Accessing a group is done with bytes or str, "
    308                     " not {}".format(type(name)))

File h5py/_objects.pyx:54, in h5py._objects.with_phil.wrapper()

File h5py/_objects.pyx:55, in h5py._objects.with_phil.wrapper()

File h5py/h5o.pyx:190, in h5py.h5o.open()

ValueError: Not a location (invalid object ID)
```

创建组

- 当有多个数据集时，可以通过组来对其进行归类和整理。

```
with h5py.File("hzg.h5", "w") as opt:
    opt.create_group("/demo")
    opt["demo"]["hz"] = hz
```

```
h5dump -A hzg.h5
```

```
HDF5 "hzg.h5" {
GROUP "/" {
   GROUP "demo" {
      DATASET "hz" {
```

```
            DATATYPE  H5T_STD_I64LE
            DATASPACE  SIMPLE { ( 10, 10 ) / ( 10, 10 ) }
         }
      }
   }
   }
```

6.5 复合数组

Structured Array: 数组中的复合数据类型

- 元素从简单类型 int, float 变成自定义的
- 方便像表格一样组织数据

```
import numpy as np
t = [('Height', 'f4'), ('Weight', 'f4'), ('Age', 'u2')]
r = np.empty(3, dtype=t)
r[0] = (170, 60, 20)
r[1] = (159, 45, 22)
r[2] = (185, 72, 26)
r
```

$$\begin{matrix} 170 & 60 & 20 \\ 159 & 45 & 22 \\ 185 & 72 & 26 \end{matrix}$$

本质上是一维数组

```
r.shape
```

$$3$$

取列

```
print(r['Height'])
```

```
[170. 159. 185.]
```

取行

```
r[0]
```

```
(170., 60., 20)
```

直接保存到 HDF5 表

- 复合数组可以直接保存为 HDF5 的表格

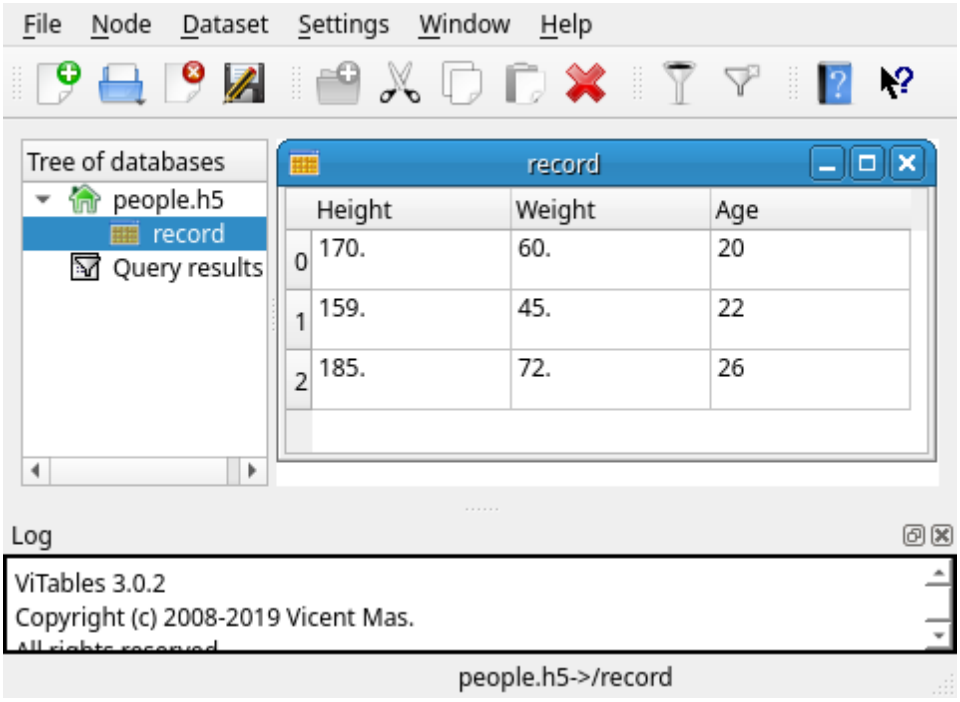
```
with h5py.File("people.h5", "w") as opt:
    opt['record'] = r
```

```
h5dump people.h5
```

```
HDF5 "people.h5" {
GROUP "/" {
   DATASET "record" {
      DATATYPE  H5T_COMPOUND {
         H5T_IEEE_F32LE "Height";
         H5T_IEEE_F32LE "Weight";
         H5T_STD_U16LE "Age";
      }
      DATASPACE  SIMPLE { ( 3 ) / ( 3 ) }
      DATA {
      (0): {
            170,
            60,
            20
         },
      (1): {
            159,
            45,
            22
         },
      (2): {
            185,
            72,
            26
         }
      }
   }
}
}
```

图形查看器 ViTables

```
vitables people.h5 # Debian
```



读复合数组

```
with h5py.File("people.h5", 'r') as ipt:
    people = ipt['record'][()]
people
```

170	60	20
159	45	22
185	72	26

DataFrame 表格

- DataFrame 特指二维表格，每行代表一条数据，每列代表一种变量。

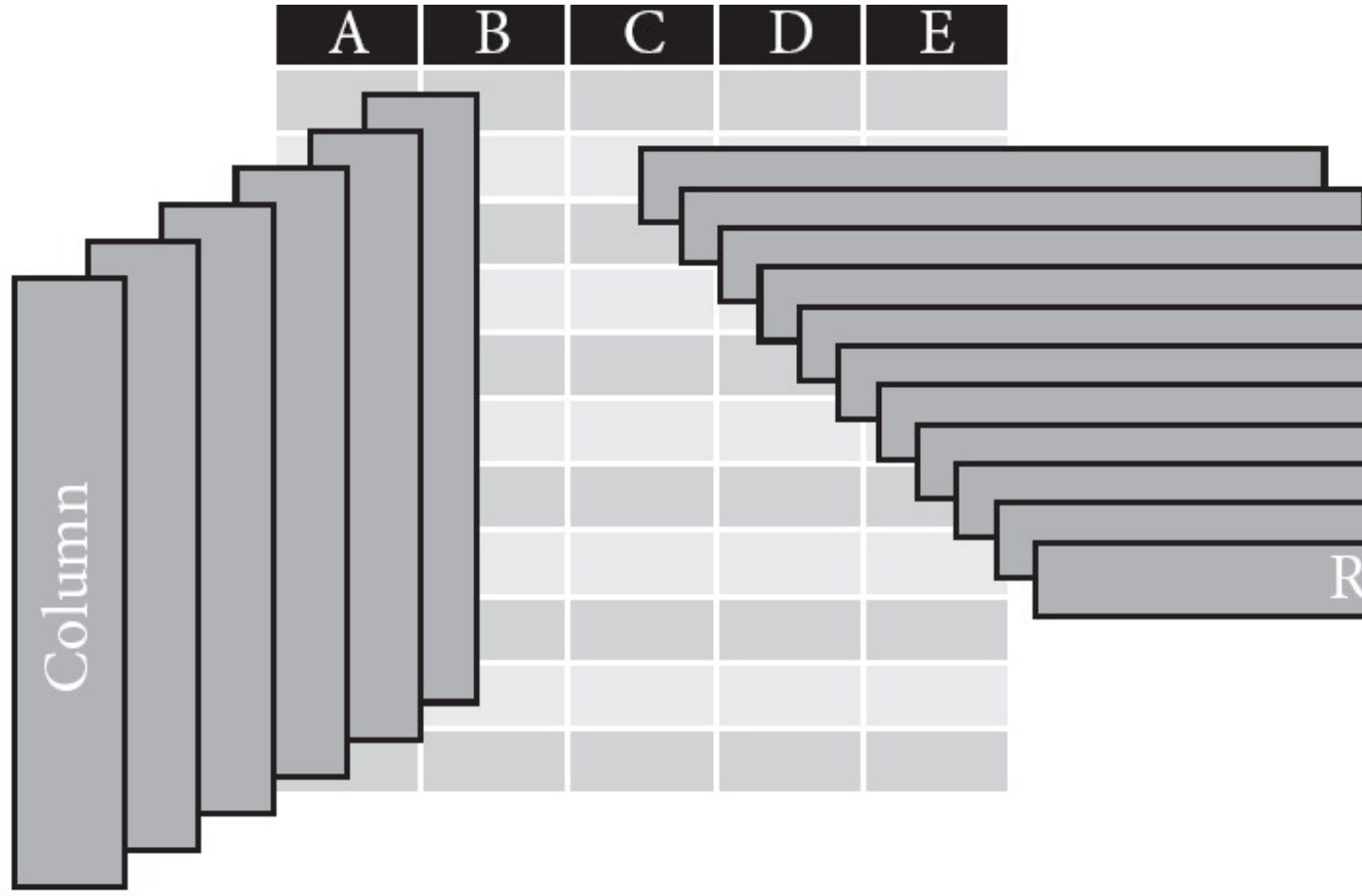
```
import pandas as pd
df_r = pd.DataFrame.from_records(r)
df_r
```

```
   Height  Weight  Age
0   170.0    60.0   20
1   159.0    45.0   22
2   185.0    72.0   26
```

- 复合数组与 DataFrame 可相互转换，逻辑结构相同

复合数组与 DataFrame 的不同

- 复合数组是逐行保存的，DataFrame 是逐列保存的。



Column-oriented Row-

- 逐行保存有利于逐步积累数据，HDF5 是逐行保存的
- 逐列保存有利于处理数据，parquet 是逐列保存的
- CSV 是逐行的还是逐列的?

CSV 表格

- 可以把复合数组写到 CSV 中，形成一个表格。

```
df_r.to_csv("people.csv", index=False)
```

```
None
```

读回来

```
pd.read_csv("people.csv")
```

```
   Height  Weight  Age
0   170.0    60.0   20
1   159.0    45.0   22
2   185.0    72.0   26
```

Apache Arrow 与 Parquet

- `Arrow` 提供了跨语言的数据交换格式。
  - 非常高效，做到 ” 零拷贝”
- `Parquet` 是 `Arrow` 在磁盘上的常用存储格式，经过了列读取优化。
  - 在 Java 语言生态的大数据平台，例如 Hadoop、Spark 上非常流行。
  - 开始被 Python (Pandas 2.0) 和 R 社区使用
- 优点：读写性能高
- 缺点：查看器等配套工具还不完善
  - 王宇逸写了 `pqdump`

读写 parquet

```
df_r.to_parquet("people.pq", index=False)
```

```
None
```

```
pq_r = pd.read_parquet('people.pq')
pq_r
```

```
   Height  Weight  Age
0   170.0    60.0   20
1   159.0    45.0   22
2   185.0    72.0   26
```

6.6 课堂练习

转化高度定制的 ~~.txt~~ CSV 文件

- 相比于 HDF5
  - CSV 的记录非常随意，无统一标准
  - CSV 占用的空间大
- 任务：把 `C3--Trace--99996.txt` 转化为 `C3--Trace--99996.h5`

读取定制的 CSV 文件

```
import numpy as np
import itertools as it

meta = {}
with open("C3--Trace--99996.txt") as metadata:
    l = next(metadata).strip() # 读第一行
    meta["model"], meta["model_param"], ds_name = l.split(",")
    l = next(metadata).strip() # 读第二行
    _, _, _, ds_length = l.split(",")
    l = next(metadata).strip() # 读第三行
    l = next(metadata).strip() # 读第四行
    _, meta["TrigTime"], _ = l.split(",")
print(meta)
```

```
{'model': 'LECROYHDO9404', 'model_param': '30486', 'TrigTime': '03-Jul-2023 14:41:14'}
```

读取数据主体

```
ds = pd.read_csv("C3--Trace--99996.txt", skiprows=4)
ds
```

```
              Time      Ampl
0     3.195045e-09 -0.012751
1     3.220045e-09 -0.016558
2     3.245045e-09 -0.013280
3     3.270045e-09 -0.005773
4     3.295045e-09  0.003425
...            ...       ...
7997  2.031200e-07  1.603280
7998  2.031450e-07  1.596830
7999  2.031700e-07  1.589380
8000  2.031950e-07  1.584830
8001  2.032200e-07  1.586100

[8002 rows x 2 columns]
```

保存成 HDF5

```
ds.to_hdf("osci.h5", ds_name)
```

```
None
```

用 PyTables 写出，兼容性较差。

标准的 HDF5 输出

```
with h5py.File("osci.h5", "w") as opt:
    opt[ds_name] = ds.to_records(index=False)
    for k, v in meta.items():
        opt[ds_name].attrs[k] = v
```

6.7 JSON

简介 JavaScript Object Notation

- JSON 最早从网站前端的 javascript 社区出现，用于代替 Extensible Markup Language (XML)。
  - 更加易于人类理解
  - 适合传递有层次的数据，特别是文本
  - 优点：与 Python 的字典结构相近
  - 缺点：数字的表达能力较弱

读 JSON

```
import json

with open("BBH_events_v3.json", "r") as ipt:
    events = json.load(ipt)
print(type(events)) # 就是一个字典
print(events.keys())
```

```
None
```

JSON 结构字典

```
events['GW150914']
```


```
{'name': 'GW150914',
 'fn_H1': 'H-H1_LOSC_4_V2-1126259446-32.hdf5',
 'fn_L1': 'L-L1_LOSC_4_V2-1126259446-32.hdf5',
 'fn_template': 'GW150914_4_template.hdf5',
 'fs': 4096,
 'tevent': 1126259462.44,
 'utcevent': '2015-09-14T09:50:45.44',
 'm1': 41.743,
 'm2': 29.237,
 'a1': 0.355,
 'a2': -0.769,
 'approx': 'lalsim.SEOBNRv2',
 'fband': [43.0, 300.0],
 'f_min': 10.0}
```


JSON 输出

```
with open("BBH_events_rewrite.json", 'w') as opt:
    json.dump(events, opt)
```


```
{"GW150914": {"name": "GW150914", "fn_H1": "H-H1_LOSC_4_V2-1126259446-32.hdf5", "fn_L1": "L-L1_LOSC_4_V2-1126259446-32.hdf5", "fn_template": "GW150914_4_temp
09-14T09:50:45.44", "m1": 41.743, "m2": 29.237, "a1": 0.355, "a2": -0.769, "approx": "lalsim.SEOBNRv2", "fband": [43.0, 300.0], "f_min": 10.0}, "LVT151012":
H1_LOSC_4_V2-1128678884-32.hdf5", "fn_L1": "L-L1_LOSC_4_V2-1128678884-32.hdf5", "fn_template": "LVT151012_4_template.hdf5", "fs": 4096, "tevent": 1128678900.
10-12T09:54:43.44", "m1": 44.111, "m2": 11.205, "a1": 0.447, "a2": -0.434, "approx": "lalsim.SEOBNRv2", "fband": [43.0, 400.0], "f_min": 10.0}, "GW151226": {
H1_LOSC_4_V2-1135136334-32.hdf5", "fn_L1": "L-L1_LOSC_4_V2-1135136334-32.hdf5", "fn_template": "GW151226_4_template.hdf5", "fs": 4096, "tevent": 1135136350.6
12-26T03:38:53.65", "m1": 19.6427, "m2": 6.7054, "a1": 0.3998, "a2": -0.0396, "approx": "lalsim.SEOBNRv2", "fband": [43.0, 800.0], "f_min": 10.0}, "GW170104"
H1_LOSC_4_V1-1167559920-32.hdf5", "fn_L1": "L-L1_LOSC_4_V1-1167559920-32.hdf5", "fn_template": "GW170104_4_template.hdf5", "fs": 4096, "tevent": 1167559936.6
01-04T10:11:58.60", "m1": 33.64, "m2": 24.82, "a1": -0.236, "a2": 0.024, "approx": "lalsim.SEOBNRv2", "fband": [43.0, 800.0], "f_min": 10.0}}
```

人类友好的 JSON 查看器

```
jq < BBH_events_rewrite.json
```


```
{
  "GW150914": {
    "name": "GW150914",
    "fn_H1": "H-H1_LOSC_4_V2-1126259446-32.hdf5",
    "fn_L1": "L-L1_LOSC_4_V2-1126259446-32.hdf5",
    "fn_template": "GW150914_4_template.hdf5",
    "fs": 4096,
    "tevent": 1126259462.44,
    "utcevent": "2015-09-14T09:50:45.44",
    "m1": 41.743,
    "m2": 29.237,
    "a1": 0.355,
    "a2": -0.769,
    "approx": "lalsim.SEOBNRv2",
    "fband": [
      43.0,
      300.0
    ],
    "f_min": 10.0
  },
  "LVT151012": {
    "name": "LVT151012",
    "fn_H1": "H-H1_LOSC_4_V2-1128678884-32.hdf5",
    "fn_L1": "L-L1_LOSC_4_V2-1128678884-32.hdf5",
    "fn_template": "LVT151012_4_template.hdf5",
    "fs": 4096,
    "tevent": 1128678900.44,
    "utcevent": "2015-10-12T09:54:43.44",
    "m1": 44.111,
    "m2": 11.205,
    "a1": 0.447,
    "a2": -0.434,
    "approx": "lalsim.SEOBNRv2",
    "fband": [
      43.0,
      400.0
    ],
    "f_min": 10.0
  },
  "GW151226": {
    "name": "GW151226",
    "fn_H1": "H-H1_LOSC_4_V2-1135136334-32.hdf5",
    "fn_L1": "L-L1_LOSC_4_V2-1135136334-32.hdf5",
    "fn_template": "GW151226_4_template.hdf5",
    "fs": 4096,
    "tevent": 1135136350.65,
    "utcevent": "2015-12-26T03:38:53.65",
    "m1": 19.6427,
    "m2": 6.7054,
    "a1": 0.3998,
    "a2": -0.0396,
    "approx": "lalsim.SEOBNRv2",
    "fband": [
      43.0,
      800.0
    ],
    "f_min": 10.0
  },
  "GW170104": {
```

```
    "name": "GW170104",
    "fn_H1": "H-H1_LOSC_4_V1-1167559920-32.hdf5",
    "fn_L1": "L-L1_LOSC_4_V1-1167559920-32.hdf5",
    "fn_template": "GW170104_4_template.hdf5",
    "fs": 4096,
    "tevent": 1167559936.6,
    "utcevent": "2017-01-04T10:11:58.60",
    "m1": 33.64,
    "m2": 24.82,
    "a1": -0.236,
    "a2": 0.024,
    "approx": "lalsim.SEOBNRv2",
    "fband": [
      43.0,
      800.0
    ],
    "f_min": 10.0
  }
}
```

美化输出

```
with open("BBH_events_indent.json", 'w') as opt:
    json.dump(events, opt, indent=2)
```

6.8 其它格式

SQLite 磁盘格式

- SQLite 数据库在磁盘上的存储格式
- 在第四周 关系代数 阶段学习
- 优点：交互界面支持 SQL 语法
- 缺点：科学和数值计算性能平庸

第七章　数据绘图

7.1　大作业

预告

- 课赛结合：Ghost Hunter 2024 中微子实验数据分析；
  - 江门中微子实验 JUNO 的光学模型。

自定义

- 如果有同学希望自定义大作业，请在 11 日 10 点之前向

  `https://git.tsinghua.edu.cn/physics-data/projects/proposal`

  仓库发送 merge request，内容至少包含

  1. 问题描述和学科背景；
  2. 预期的目标和评价方式；
  3. 时间安排；
  4. 大作业项目导师，一名获得过《实验物理的大数据方法》课程 A 以上（或有同等水平）的学长。

7.2　复习准备

安装软件

- Matplotlib（Python 绘图）、SciPy（科学计算）

```
apt install python3-matplotlib python3-tk python3-scipy
```

科学绘图

- A plot worths a thousand words. 图是描述科学对象的载体。
  - 撰写论文甚至可理解为“看图说话”：论文的局部目标是用语言和公式把图片解释清楚。

  - 绘制高质量的图，所花精力不比书写几千字少

- Matplotlib
  - 数据画图和可视化工具，风格受 Matlab 影响
  - 参考资料：Scipy Lecture Notes 1.5

X Window System 也被称为 “X11”

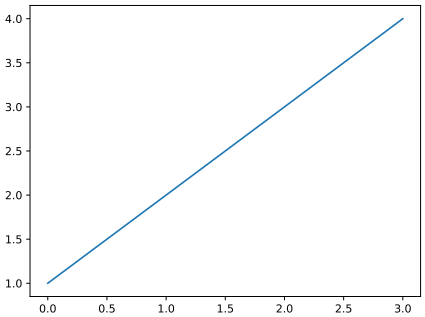
- 图形显示事实上的国际标准
- Server-client 架构
- Server 为显示图形提供服务 (`WSLg` 用的 `FreeRDP=`，或 `=VcXsrv`)
- Client 连接 X11 server，请求显示图形 (`xeyes`, `ViTables`, `Matplotlib`)

7.3 Matplotlib

一条直线

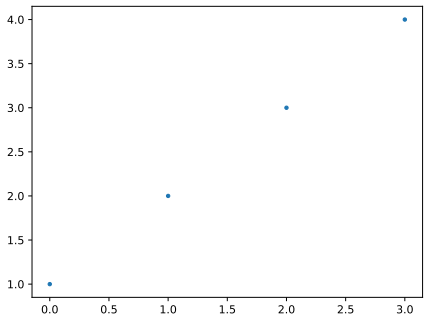
- 四点连成一条直线

```
from matplotlib import pyplot as plt

plt.plot([1,2,3,4])
```



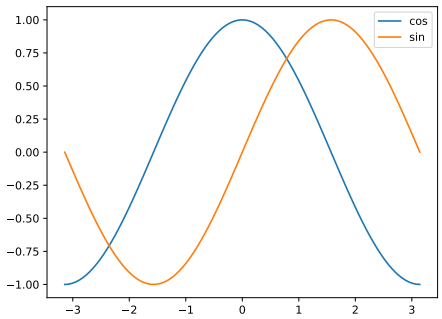
只画四点，不连线

```
plt.plot([1,2,3,4], '.')
```



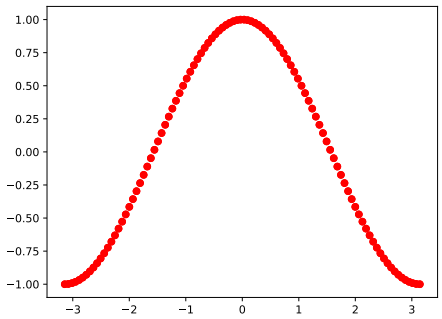
两条曲线

```
import numpy as np
t = np.linspace(-np.pi, np.pi, 100)
C, S = np.cos(t), np.sin(t)
plt.plot(t, C, label="cos")
plt.plot(t, S, label="sin")
plt.legend()
```



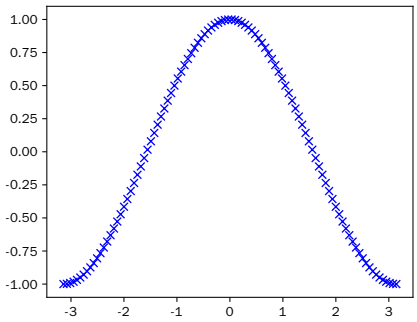
变成红色的点

```
plt.plot(t, C, "ro") # 改成红色的圆点
```



点的形状

```
plt.plot(t, C, "bx") # 改成蓝色的 x
```



点形状列表

mark	symbol
.	point
,	pixel
o	circle
v	triangle down
^	triangle up
<	triangle left
>	triangle right
1	tri down
2	tri up
3	tri left
4	tri right
8	octagon
s	square

mark	symbol
p	pentagon
P	plus (filled)
*	star marker
h	hexagon1
H	hexagon2
+	plus
x	x
X	x (filled)
D	diamond
d	$\text{thin}_{\text{diamond}}$
|	vline
_	hline

线与颜色的形状

记号	线型
-	solid
–	dashed
-.	dash-dot
:	dotted

记号	颜色
b	blue
g	green
r	red
c	cyan
m	magenta
y	yellow
k	black
w	white

7.4 图片的调整

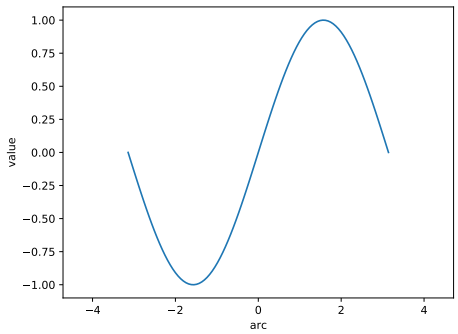
调整必要性

- 默认的图片，会有各类值得改进之处。
  - 坐标轴取值范围、风格、含义
  - 图片中的标记
  - 字体大小

坐标轴

- 显示范围、标注

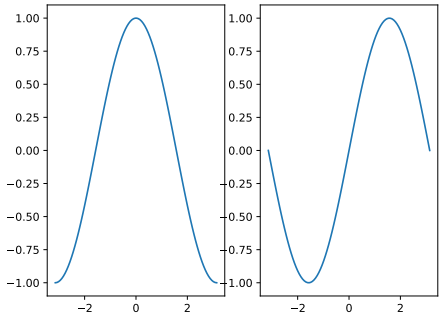
```
plt.plot(t, S)
plt.xlim(t.min() * 1.5, t.max() * 1.5)
plt.xlabel("arc")
plt.ylabel("value")
```



子图

- 并列绘图，调整图的显示范围

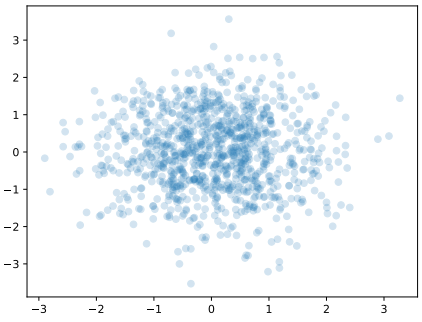
```
plt.subplot(1, 2, 1)
plt.plot(t, C)
plt.subplot(1, 2, 2)
plt.plot(t, S)
```



7.5 常用形式

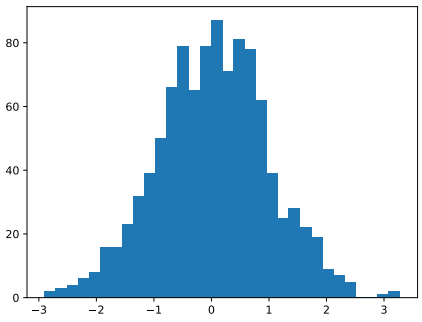
散点图

```
n = 1024
X = np.random.normal(0, 1, n)
Y = np.random.normal(0, 1, n)

plt.scatter(X, Y, alpha=0.2)
```



直方图

```
plt.hist(X, bins=32)
```



画院系的统计

```
import pandas as pd
dep = pd.read_csv("departments.csv", header=None)
plt.hist(list(dep[0]))
```

图的横轴间距不太理想。

7.6 多维元素的表现

平铺

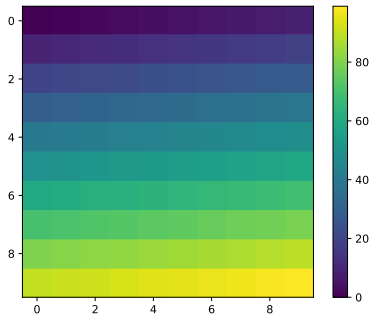
- 表现简单的三维数据

```
img = np.arange(100).reshape(10, 10)
print(img)
```

```
[[ 0  1  2  3  4  5  6  7  8  9]
 [10 11 12 13 14 15 16 17 18 19]
 [20 21 22 23 24 25 26 27 28 29]
 [30 31 32 33 34 35 36 37 38 39]
 [40 41 42 43 44 45 46 47 48 49]
 [50 51 52 53 54 55 56 57 58 59]
 [60 61 62 63 64 65 66 67 68 69]
 [70 71 72 73 74 75 76 77 78 79]
 [80 81 82 83 84 85 86 87 88 89]
 [90 91 92 93 94 95 96 97 98 99]]
```

平铺（二）

```
plt.imshow(img)
plt.colorbar()
```



高斯分布的概率密度函数

```
from scipy.stats import multivariate_normal

rv = multivariate_normal(mean=(0, 0), cov = ((1,0.5), (0.5,0.5)))
print(rv.pdf((0, 0))) # (x, y) -> f(x, y)

norm_x, norm_y = np.mgrid[-1:1:.01, -1:1:.01]
pos = np.dstack((norm_x, norm_y))
prob_density = rv.pdf(pos)
print(pos.shape, prob_density.shape)
```

```
0.31830988618379075
(200, 200, 2) (200, 200)
```

- 给 (i,j) 着色为 $f(x,y)$

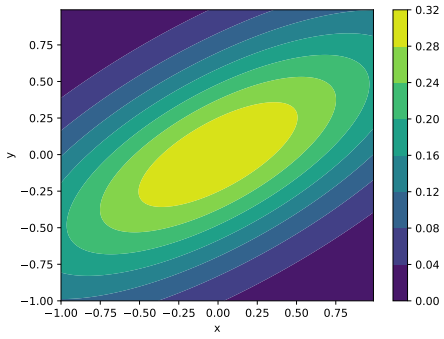
$$(i,j) \to (x,y) \to f(x,y)$$

  pos $(i,j) \to (x,y)$

  prob_density $(i,j) \to f(x,y)$

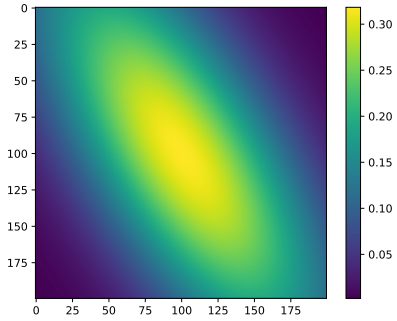
绘制等高线图

```
plt.contourf(norm_x, norm_y, prob_density)
plt.colorbar()
plt.xlabel("x")
plt.ylabel("y")
```



imshow 的原点约定不同

```
plt.imshow(prob_density)
plt.colorbar()
```



真三维图

来自 https://matplotlib.org/stable/gallery/mplot3d/contour3d_3.html#sphx-glr-gallery-mplot3d-co 例子，用来绘制二维高斯分布

```
from mpl_toolkits.mplot3d import axes3d

ax = plt.figure().add_subplot(projection='3d')
# Plot the 3D surface
ax.plot_surface(norm_x, norm_y, prob_density,
                edgecolor='royalblue', lw=0.5, rstride=8, cstride=8,
                alpha=0.3)

ax.set(xlim=(-1, 1), ylim=(-1, 1), zlim=(0, 0.4),
       xlabel='x', ylabel='y', zlabel='f(x,y)')
plt.show()
```

制作动画

- `FuncAnimation`
- 不断画图法

```
from matplotlib.backends.backend_pdf import PdfPages
with PdfPages("normal2d.pdf") as pp:
    for angle in range(0, 720, 5):
        ax.view_init(elev=angle, azim=45, roll=30)
        pp.savefig()
```

```
convert -verbose -delay 50 -density 100 normal2d.pdf normal2d.gif
```

绘制流线图

- 相图: (p, q) 的演化图, $p = m\dot{q}$
- 经典谐振子
  - $\dot{p} = -kq$
  - 取 $m = k = 1$ 每个 (p, q) 位置对应

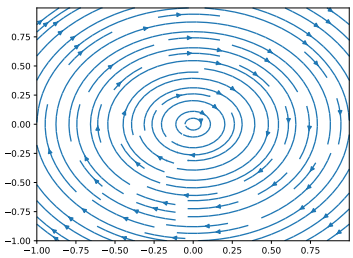
$$(\dot{p}, \dot{q}) = (-q, p)$$

绘制流线图（二）

- 表现矢量场

$$(i, j) \to (p, q) \to (\dot{p}, \dot{q})$$

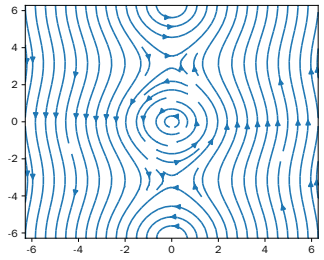
```
P, Q = np.mgrid[-1:1:.002, -1:1:.002]
dP = -Q # \dot{p}
dQ = P  # \dot{q}
plt.streamplot(Q, P, dQ, dP)
```



课堂练习：非简谐轻杆单摆的相图

$$mL\ddot{\theta} = -mg\sin\theta$$

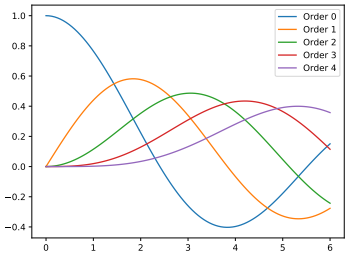
```
theta, eta = np.mgrid[-2*np.pi:2*np.pi:.005, -2*np.pi:2*np.pi:.005]
dtheta = eta
deta = -np.sin(theta)
plt.streamplot(eta, theta, deta, dtheta)
```



7.7 Bessel 函数

第一类 Bessel 函数

```
from scipy import special

x = np.linspace(0, 6, 100)
y_values = [ special.jv(n, x) for n in range(0, 5) ] # 第一类 Bessel 函数
for i, y in enumerate(y_values):
    plt.plot(x, y, label=f"Order {i}")
plt.legend()
plt.xlabel("x")
plt.ylabel("values of Bessel function")
```



7.8 更多样例

Matplotlib 官方图片展

`https://matplotlib.org/stable/gallery/index.html`

- 看到需要的图，点进去可查看代码

第八章　蒙特卡罗方法与大作业

8.1　复习准备

大作业准备

安装 GNU Make

```
apt install make-guile
```

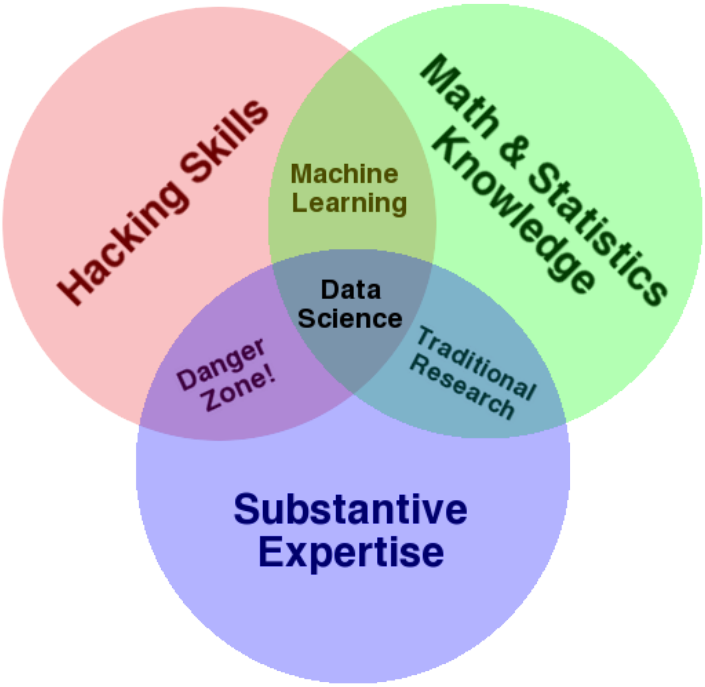
进阶使用

参见 第十讲 GNU Make 数据生产线

科学绘图

- 原则：复现、透明
  - 艺术性提升，需要花很大精力打磨
  - 代表了“实验结果”
- Matplotlib：
  - 点、线
  - 散点图、直方图
  - 标量场、向量场
  - 动画

技能树



1. 本课程目标为数理大类的同学补齐 黑客技术 。

2. 警惕 *Danger Zone!* 。

   黑客技术 人类掌握工具的最高水平

   数学与统计 人类理性思维的最高水平，欢迎选修《概率统计分析及量测技术》

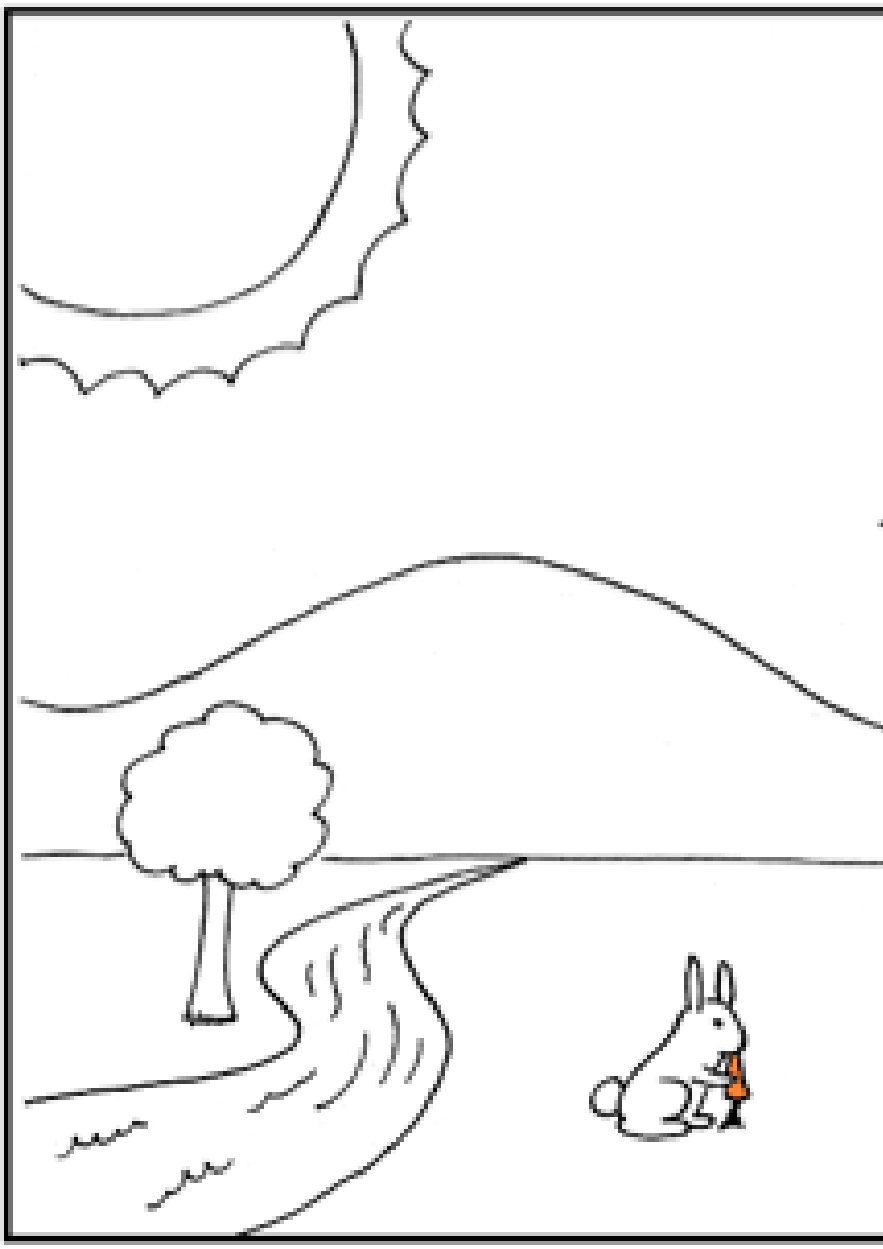
3. 专业知识 根据自己的兴趣选择。

   - 物理学的思维方式和实践方法是一切专业知识的标杆。

4. 大作业是对此技能树的一次综合检验。

8.2 弱相互作用之世界

科学家如何看待世界



投票 最多可选3项

你看到了什么？

A 流体力学

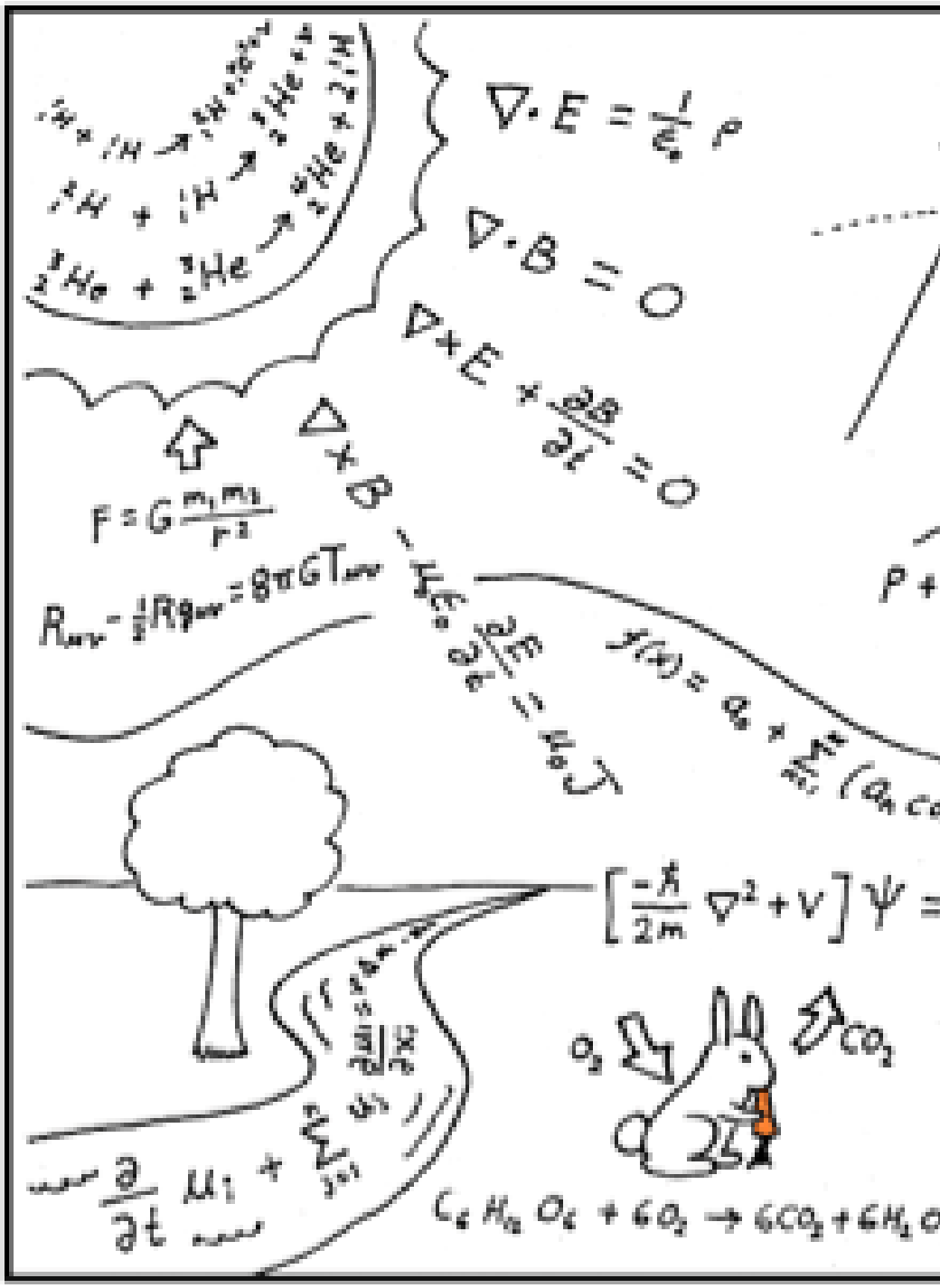
B 核聚变

C 宇宙射线

D 热辐射

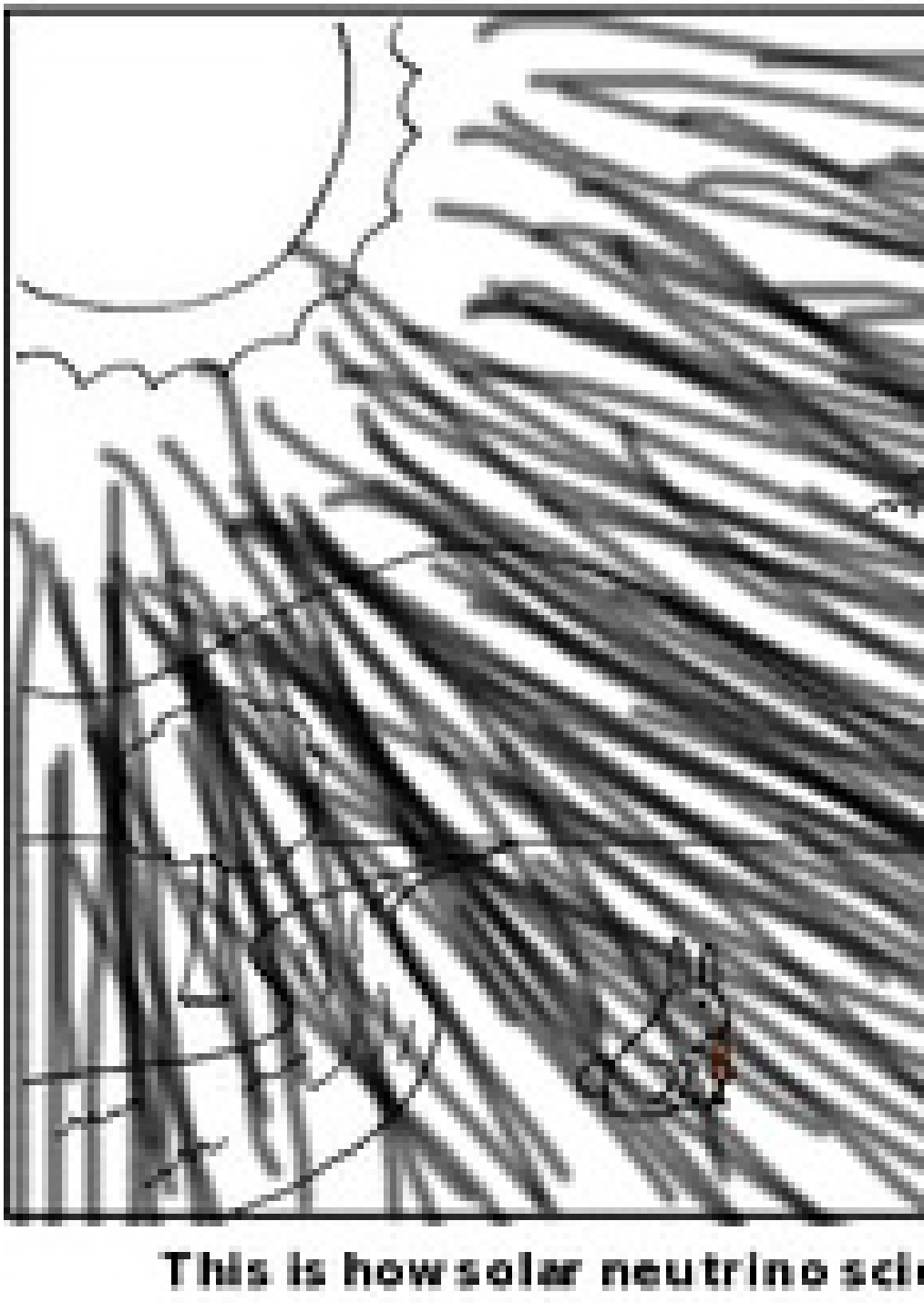
E 万有引力

科学家如何看待世界



中微子物理学家咋看

中微子物理学家咋看



中微子的基本事实

1. 电中性，自旋 1/2，只参与弱相互作用。

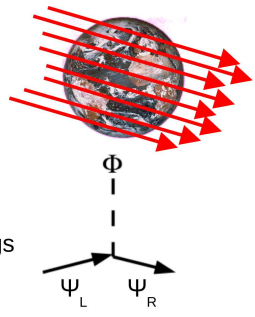
   - 太阳中微子与电子的反应截面是 $SIe - 50m^2$。
   - 类比，原子核的尺度 $SIe - 30m^2$。

2. 自然界中只观测到左手中微子（ ν_L ）和右手反中微子（ $\bar{\nu}_R$ ）。

3. 弱超荷（weak hypercharge）：ν_L -1，ν_R 0；e_L -1，e_R -2。

4. 弱同位旋（weak isospin）：ν_L 1/2，ν_R 0；e_L -1/2，e_R 0。

5. 最简的 Higgs 机制不适用：

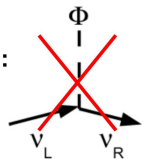


Mass terms in the Standard Model:

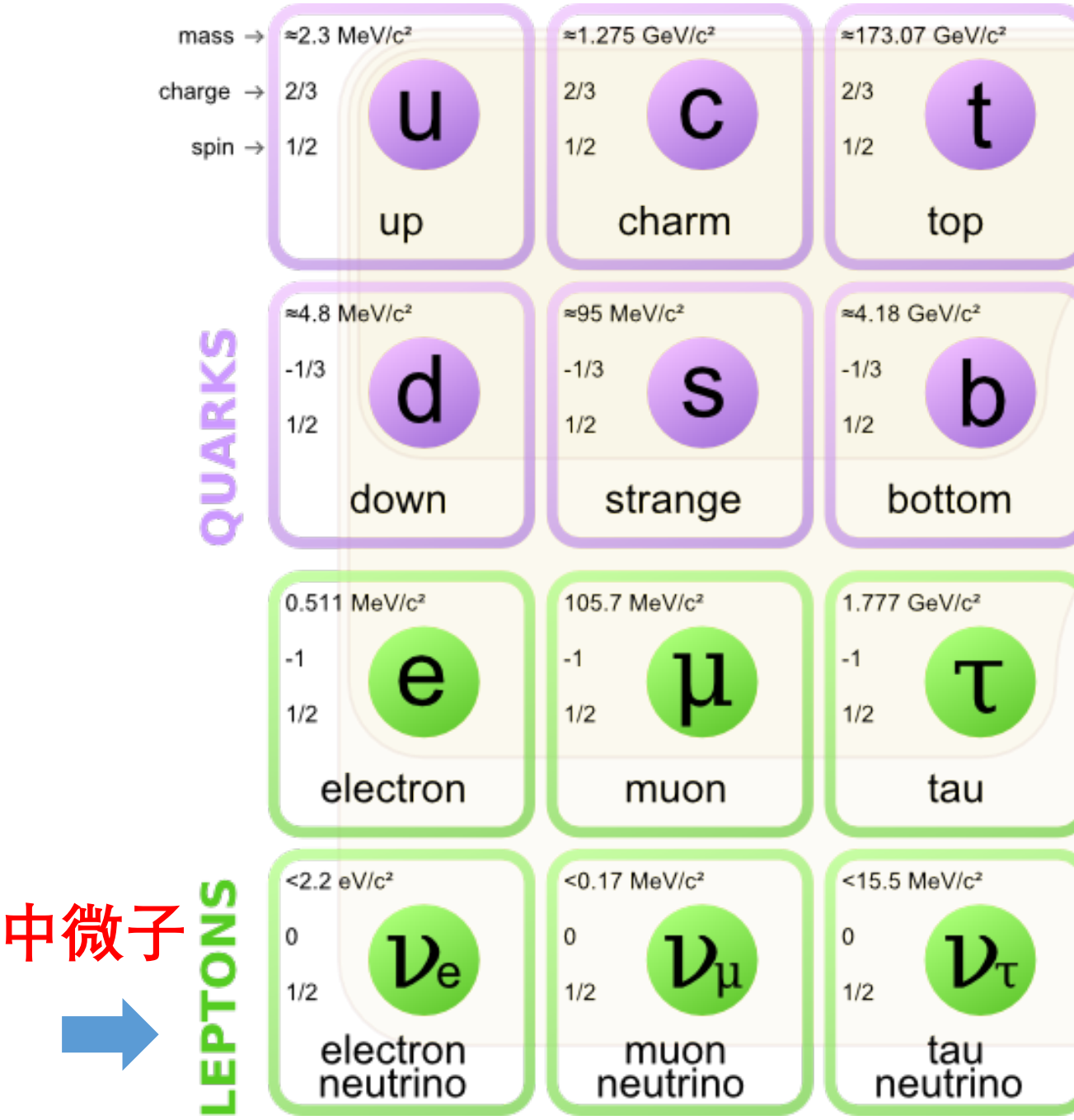
coupling to the Higgs

The most abundant particle in the universe: 336 / cm³
(together with the particle of light, the photon)

In original SM ν only left-handed: ν_L
→ difficult to account for mass term:
Yukawa coupling to the Higgs did not exist in the SM

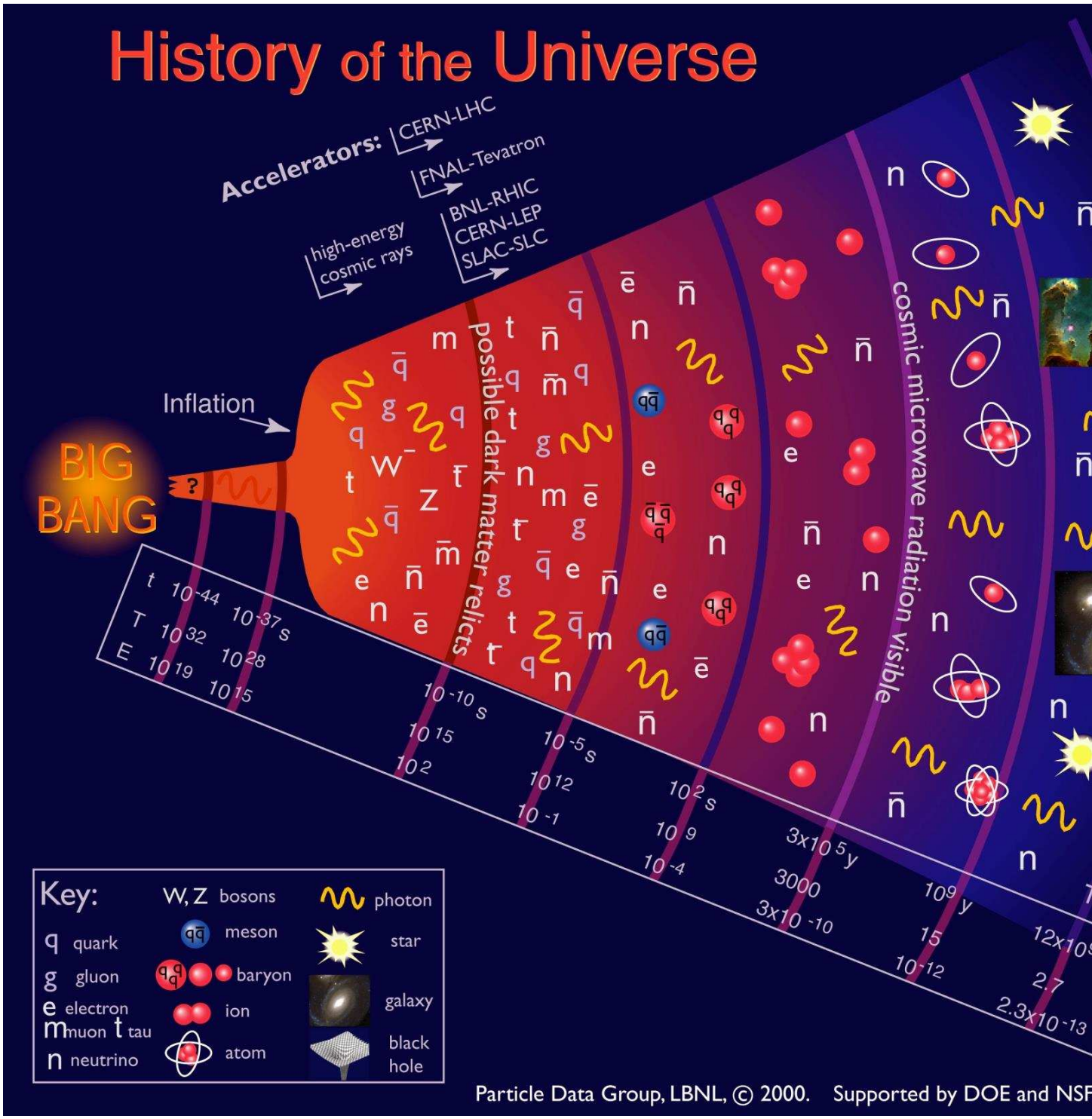


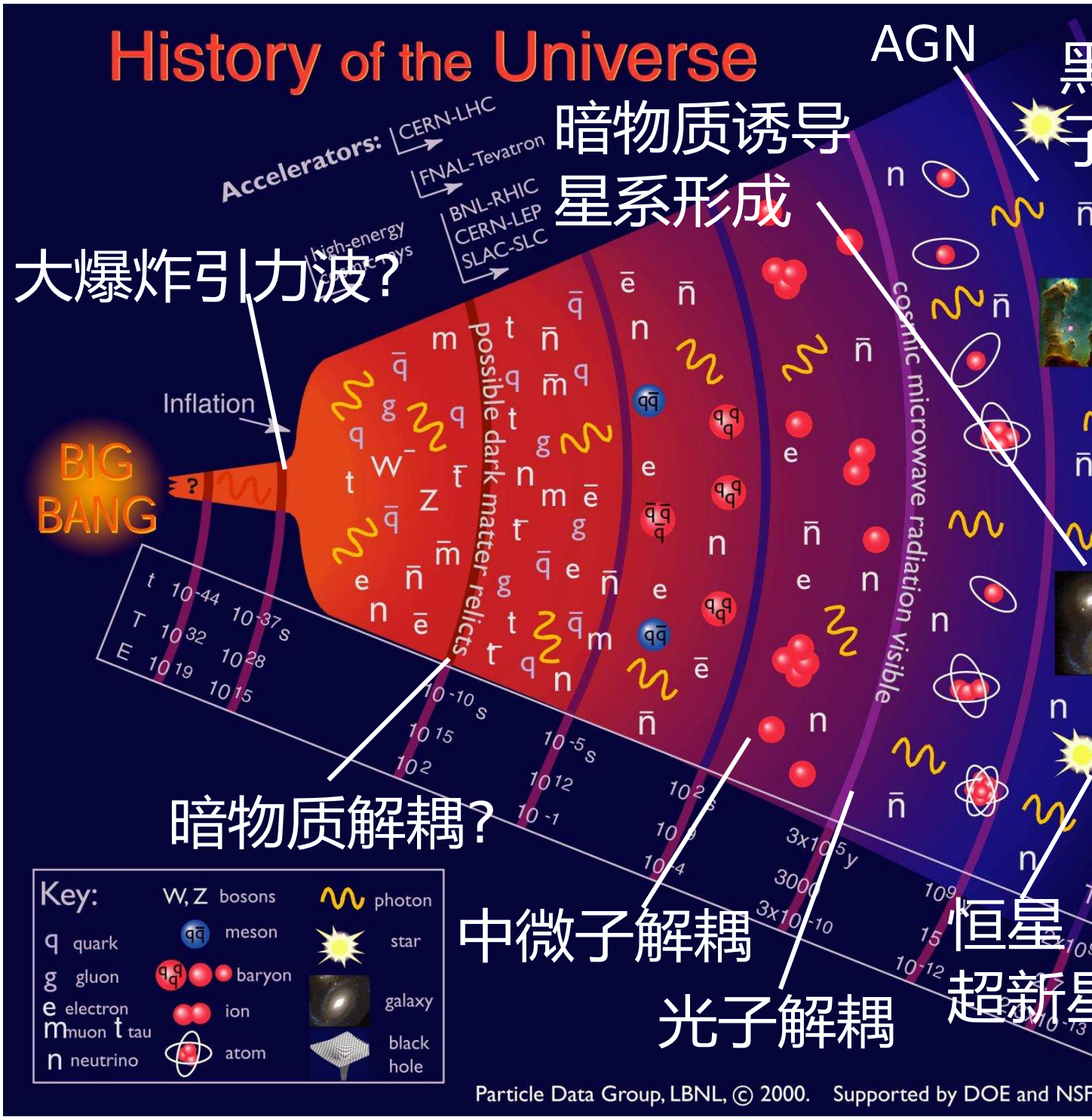
已知的基本粒


mass →
charge →
spin →
≈2.3 MeV/c²
2/3
1/2
u
up
≈1.275 GeV/c²
2/3
1/2
c
charm
≈173.07 GeV/c²
2/3
1/2
t
top
QUARKS
≈4.8 MeV/c²
-1/3
1/2
d
down
≈95 MeV/c²
-1/3
1/2
s
strange
≈4.18 GeV/c²
-1/3
1/2
b
bottom
0.511 MeV/c²
-1
1/2
e
electron
105.7 MeV/c²
-1
1/2
μ
muon
1.777 GeV/c²
-1
1/2
τ
tau
LEPTONS
<2.2 eV/c²
0
1/2
νe
electron neutrino
<0.17 MeV/c²
0
1/2
νμ
muon neutrino
<15.5 MeV/c²
0
1/2
ντ
tau neutrino

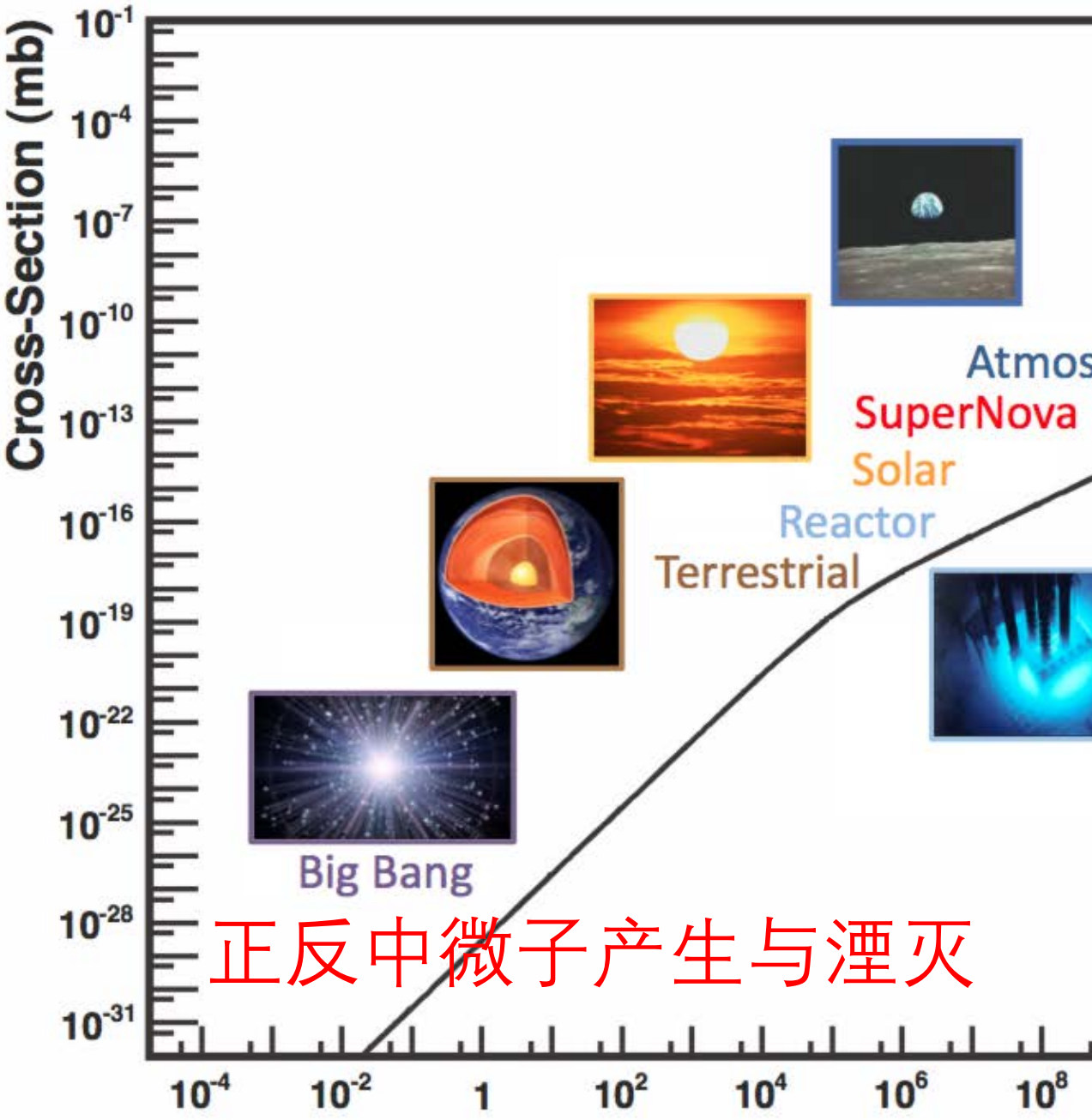
中微子

宇宙的历史


History of the Universe
Accelerators:
CERN-LHC
FNAL-Tevatron
BNL-RHIC
CERN-LEP
SLAC-SLC
high-energy cosmic rays
Inflation
BIG BANG
possible dark matter relicts
cosmic microwave radiation visible
t 10-44 10-37 s
T 1032 1028
E 1019 1015
10-10 s
1015
102
10-5 s
1012
10-1
102 s
109
10-4
3x105 y
3000
3x10-10
109 y
15
10-12
12x10
2.7
2.3x10-13
Key:
q quark
g gluon
e electron
m muon t tau
n neutrino
W, Z bosons
meson
baryon
ion
atom
photon
star
galaxy
black hole
Particle Data Group, LBNL, © 2000. Supported by DOE and NSF


History of the Universe
Accelerators:
CERN-LHC
FNAL-Tevatron
BNL-RHIC
CERN-LEP
SLAC-SLC
high-energy cosmic rays
AGN
暗物质诱导
星系形成
大爆炸引力波?
Inflation
BIG BANG
possible dark matter relicts
cosmic microwave radiation visible
暗物质解耦?
中微子解耦
光子解耦
恒星
超新星
Key:
q quark
g gluon
e electron
m muon
t tau
n neutrino
W, Z bosons
meson
baryon
ion
atom
photon
star
galaxy
black hole
Particle Data Group, LBNL, © 2000. Supported by DOE and NSF

不同能量中微子与电子



正反中微子产生与湮灭

10.1103/RevModPhys.84.1307

8.3 中微子

~~中子~~中微

“我终于找到了一个绝望拯
电中性粒子…自旋为1/2并服从

上世纪早期的核衰

$$n \rightarrow p + e$$

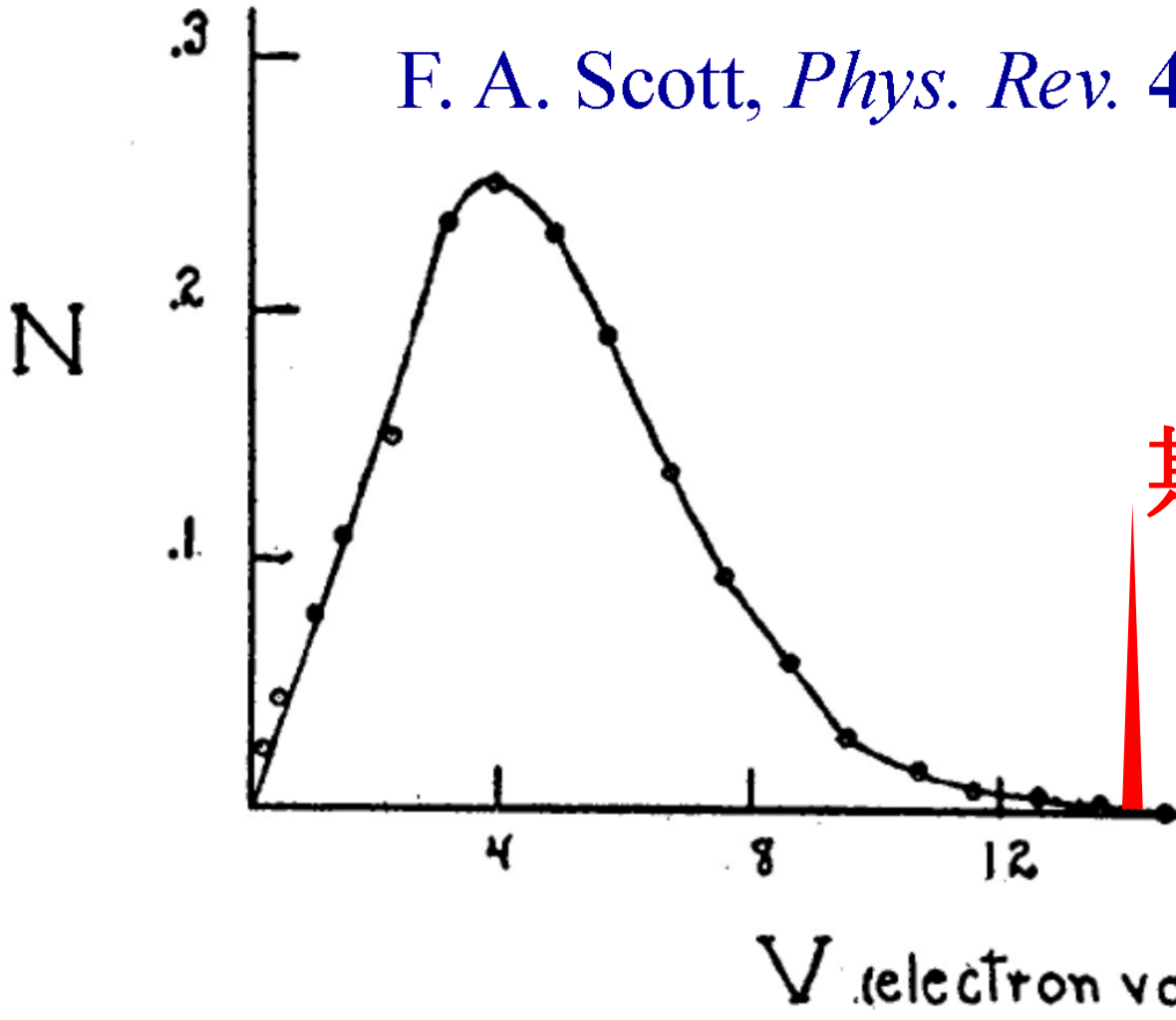


FIG. 5. Energy distribution curve

中微子:“绝望的泡

The Desperate Remedy

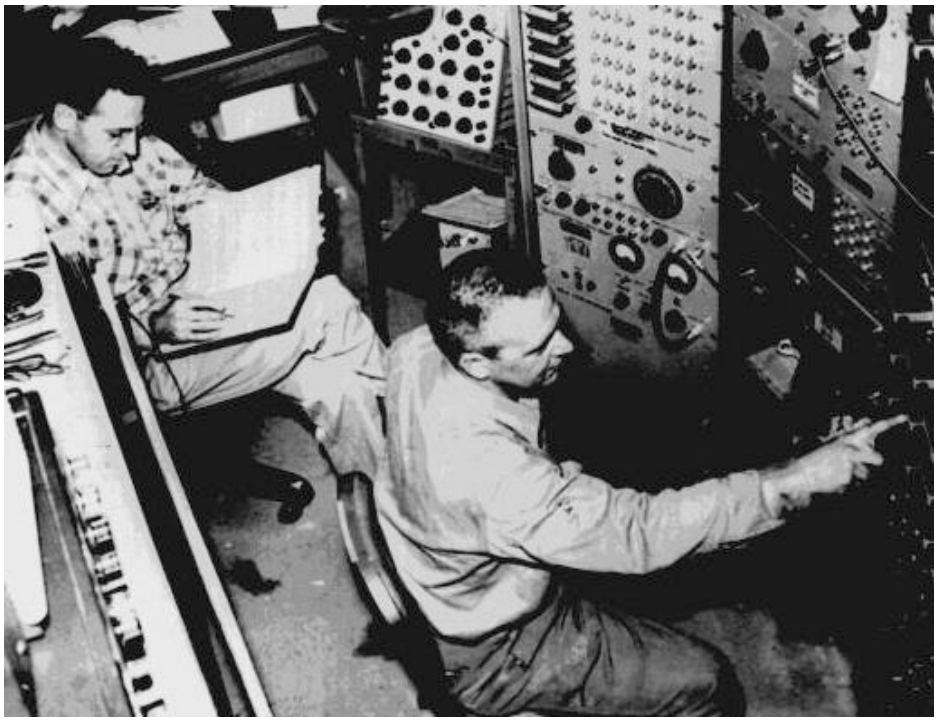
Physical Institute of the
Federal Institute of Technology (ETH)
Zürich

Dear radioactive ladies and gentlemen,

As the bearer of these lines, to whom
graciously, will explain more exactly, co
'false' statistics of N-14 and Li-6 nucle
continuous β-spectrum, I have hit upon a
to save the "exchange theorem"* of statis
theorem. Namely [there is] the possibilit
exist in the nuclei electrically neutral
wish to call neutrons,** which have spin
exclusion principle, and additionally dif
ta in that they do not travel with the v
The mass of the neutron must be of the s
tude as the electron mass and, in any ca
0.01 proton mass. The continuous β-spectr
understandable by the assumption that in
is emitted together with the electron, i
the sum of the energies of neutron and e

长达四分之一世纪

直到1956年才被Cowan & R



泡利教授：很高兴地通知你，通过
衰变，我们肯定是探测到了来自裂

大科学仪器的起源

Reines remarked that before Cowan's work, a "big" physic might use a one-liter detect background in weapons rese emboldened Reines and Cow such a large increase in expe (2 x 200L).

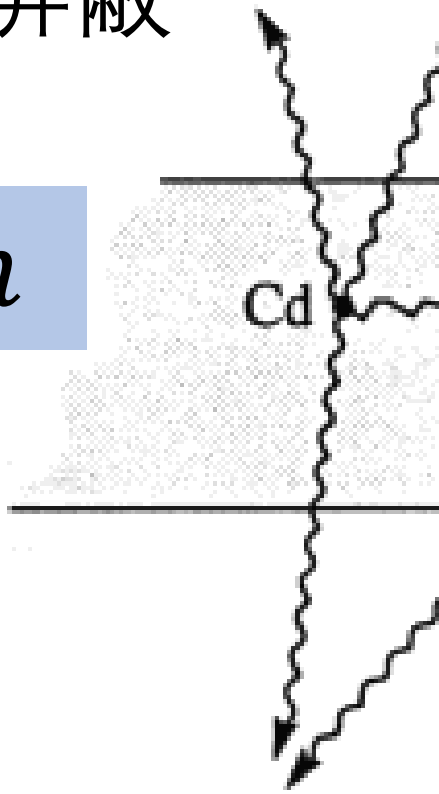
物理学从此走向

dx.doi.org/10.1103/PhysRevFocus.19.13

中微子的发现 Cowan &

- 中微子源：研究型反应堆
- 塑料闪烁体：Triethylber
- $CdCl_4$：中子俘获
- 符合探测，水屏蔽

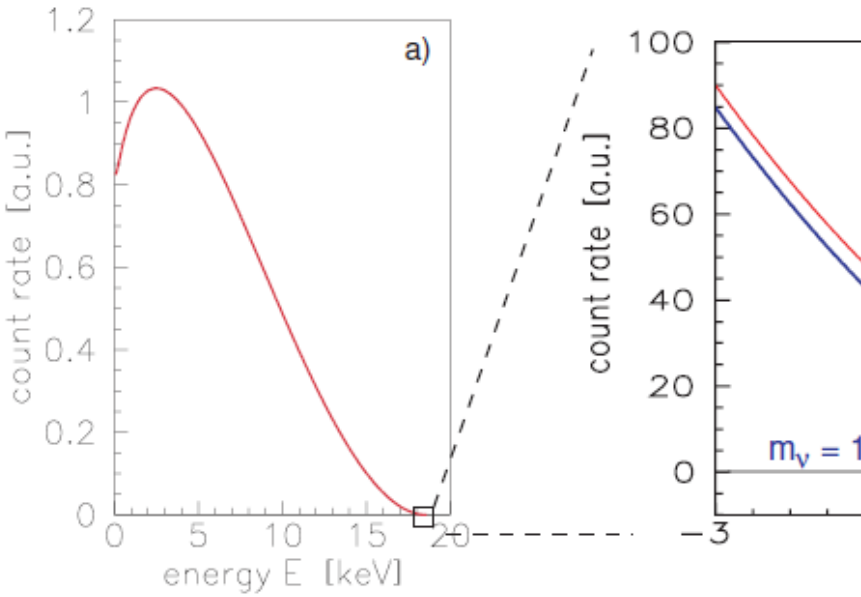
$$\bar{\nu}_e + p \rightarrow e^+ + n$$

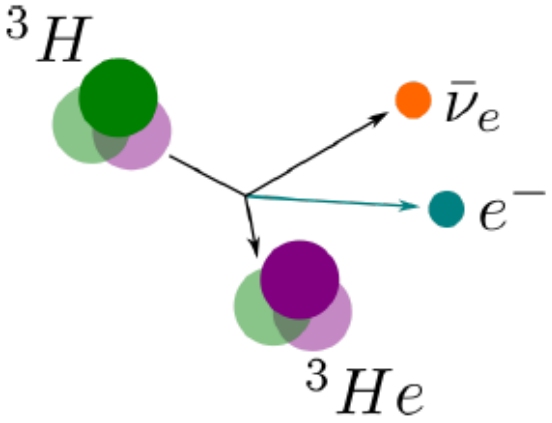


1995 Nobel Prize

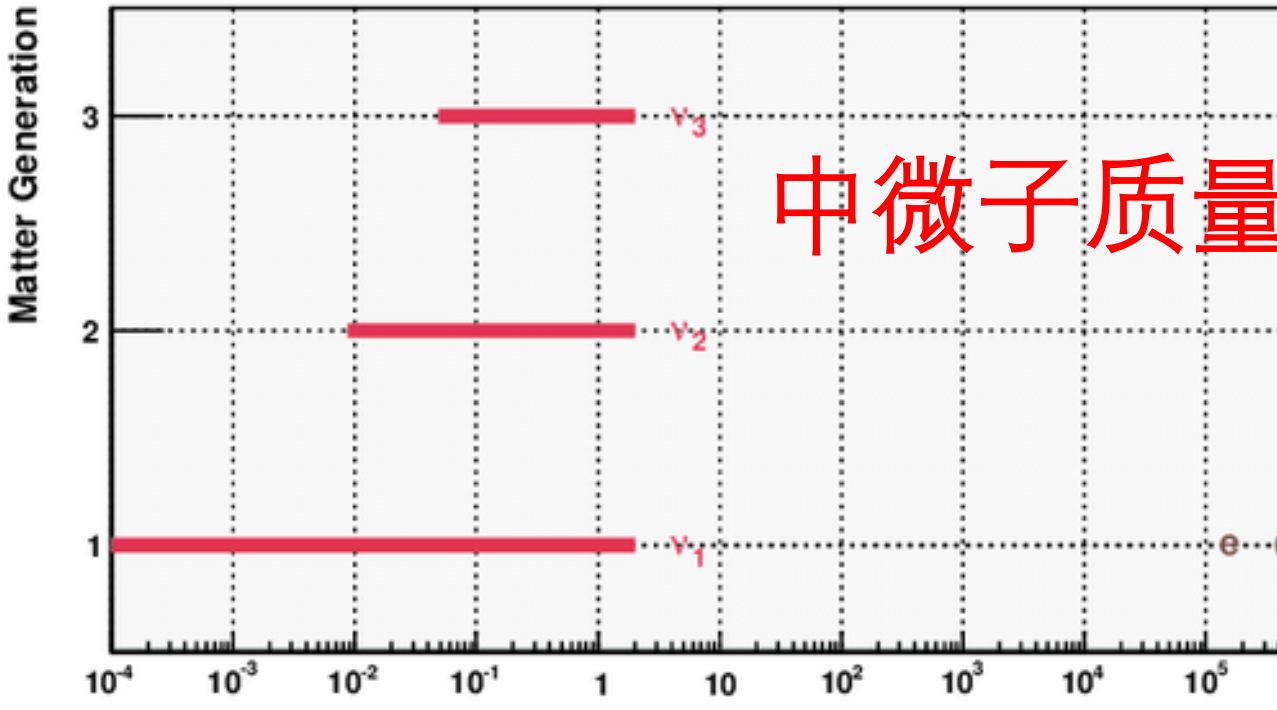
我们依旧不知道中微

- KATRIN 实验2021年最新结





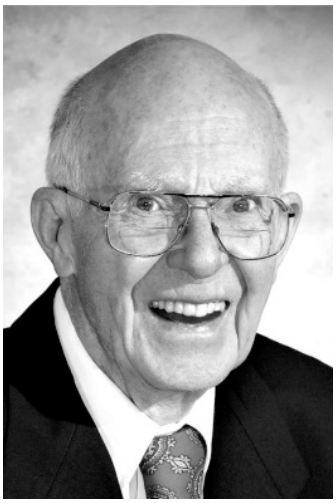
- Benefits from t
  - Short lifetime
  - Low endpoint
  - Superallowed


Matter Generation
3
2
1
ν_3
ν_2
ν_1
e
10^{-4}
10^{-3}
10^{-2}
10^{-1}
1
10
10^{2}
10^{3}
10^{4}
10^{5}
中微子质量
基本粒子质量

8.4　宇称正反中微子

正反中微子，宇
Weyl Spin

1955: Ray Davis寻



$\nu_e + {}^{37}\mathrm{Cl} \rightarrow e^- + {}^{37}$

- 氯气？有毒，气
- 四氯化碳（CCl_4液

化学放射计量，使用 Cl 捕捉

1. 使用 He 气注入液体，气
2. 将 Ar 与 He 按原子量分离
3. 使用盖革计数器观察Ar-3

$$e^- + {}^{37}\mathrm{Ar} \rightarrow \nu_e + {}^{37}\mathrm{Cl}\ (+$$

3800L四氯化碳观察原子反应

投票 最多可选1项

为什么Ray没有观测到中微子，但Reins观查到了？

A 实验计数方法不可靠

B 中微子不与 Cl 反应，核物

C 只是反应堆中微子不与 Cl

D 以上解释都不对

两个实验的对

核反应堆旁的两个实验

$$\nu_e + {}^{37}\mathrm{Cl} \rightarrow e^- + {}^{37}\mathrm{Ar}\ (\\ \bar{\nu}_e + p \rightarrow e^+ + n\ (\text{-}1.8\mathrm{M}$$

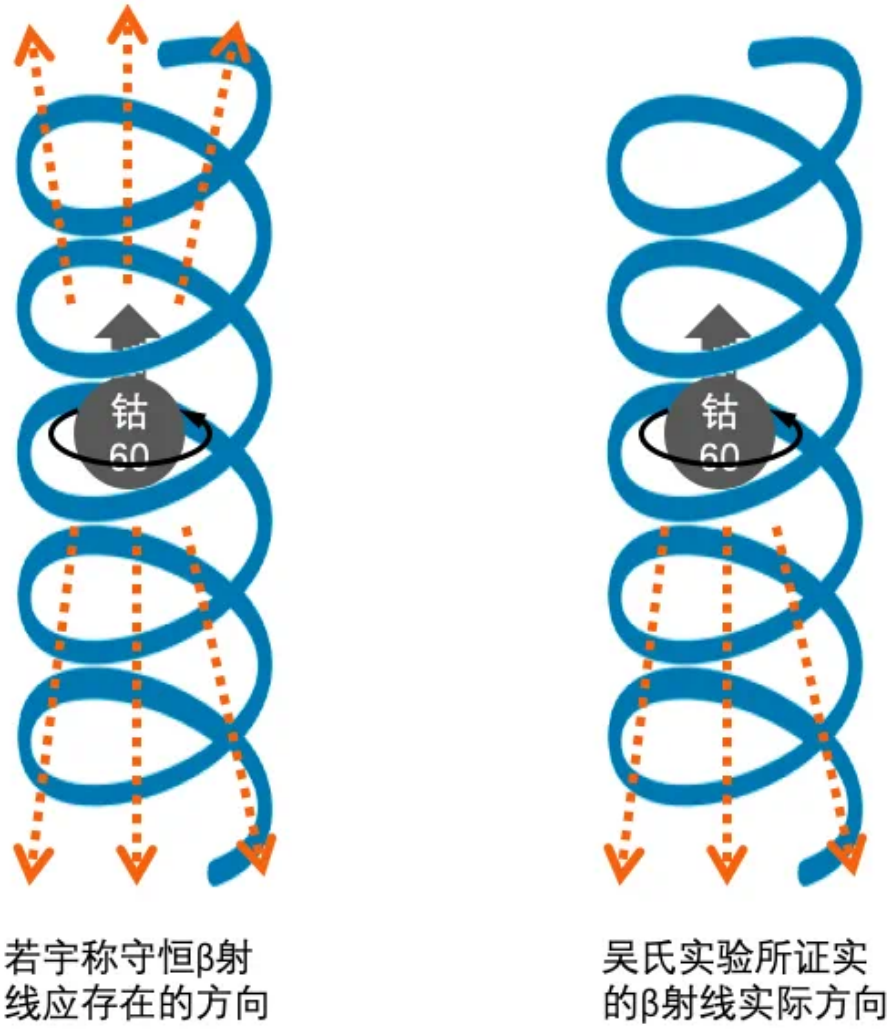
Ray 的实验证实，第一种

截面至多是第二种的$\frac{1}{20}$。

结论：存在正反两种中

电子对应。

吴健雄实验（图待更新）



吴健雄改变了磁场方向，发现电子方向始终与磁场方向相反。

$$\langle \vec{\sigma}_{\mathrm{Ni}} \cdot \vec{p}_e \rangle \neq 0$$

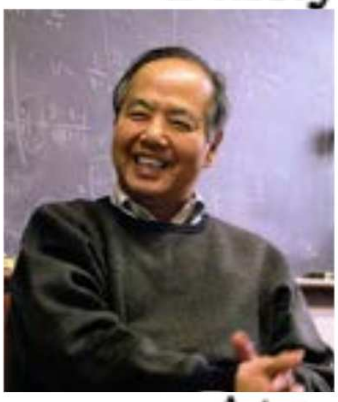
结论

弱相互作用下宇称不守恒，但为什么?

宇称不守恒驱动的新中微子理论

PHYSICAL REVIEW VOLUME 105, NU

Parity Nonconservation and a Two-Comp



T. D. LEE, *Columbia University, Ne*

AND

C. N. YANG, *Institute for Advanced Study*

(Received January 10, 1957; revised manuscrip

A two-component theory of the neutrino is discussed. The th
in interactions involving the neutrino. Various experimental
marks concerning nonconservation are made.

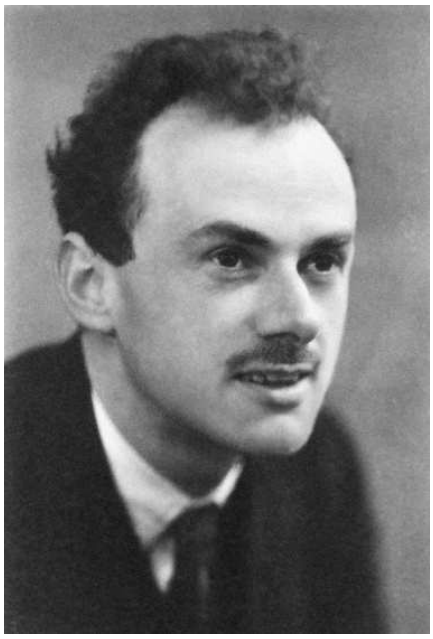
Dirac

Neutrino wave functions ne

T.D. Lee & C.N. Yang

Neutrino mass is zero and i

need only two components.



Paul Dirac

$$i\gamma^{\alpha}\partial_{\alpha}\psi -$$

Dirac 矩阵

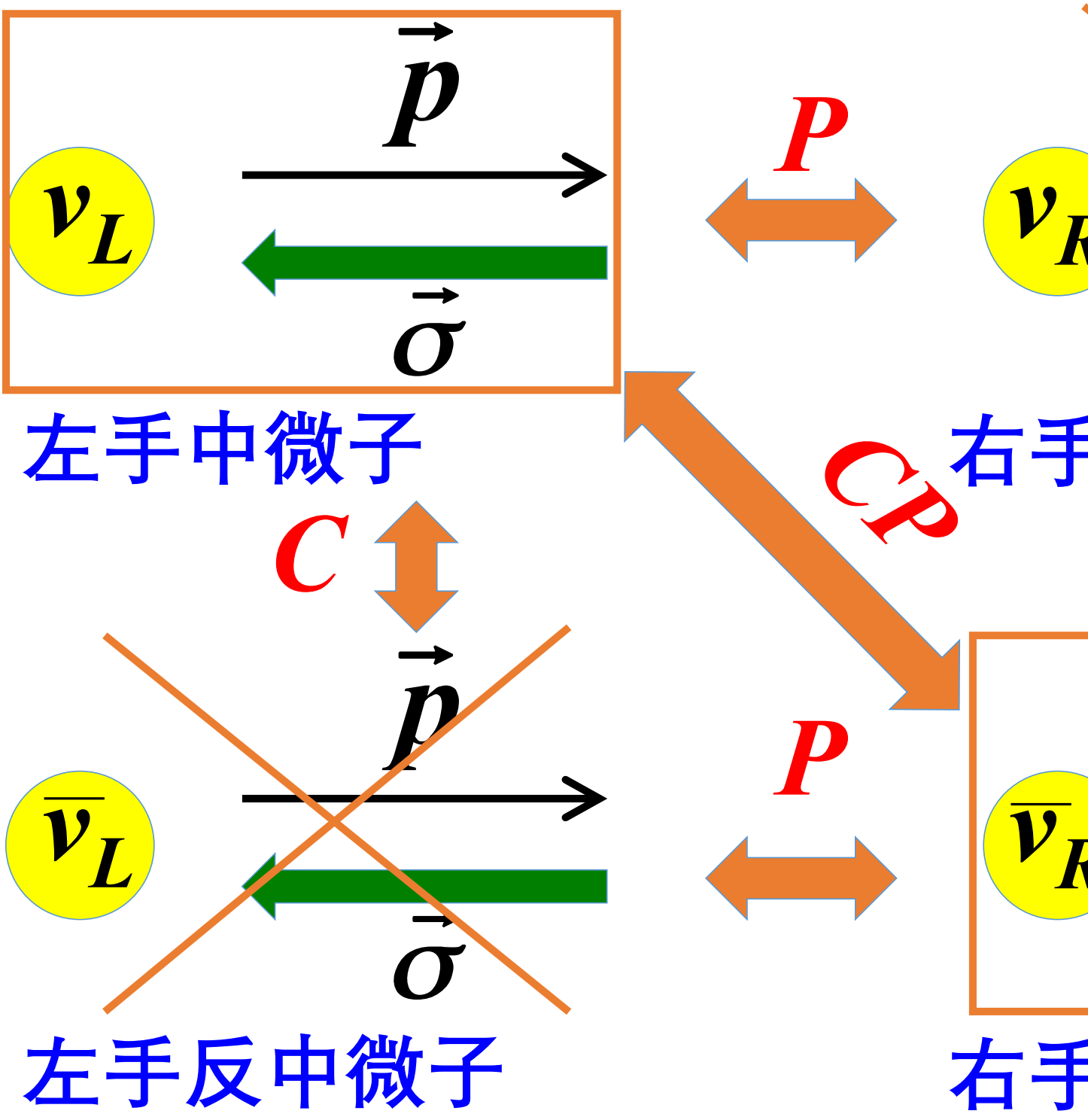
时空指标

$$i\gamma^{\alpha}\partial_{\alpha}\psi_L - m\nu\psi_R = 0$$

$$i\gamma^{\alpha}\partial_{\alpha}\psi_R - m\nu\psi_L = 0$$

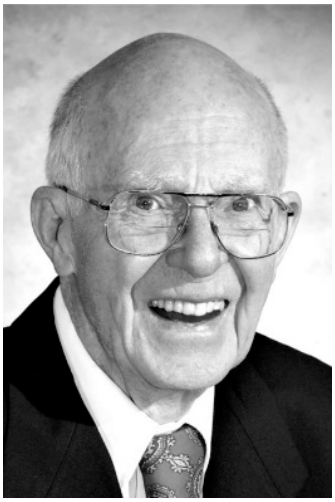
Weyl Spinor

中微子的 CP 变换


$\vec{p}$
ν_L
$\vec{\sigma}$
P
ν_R
左手中微子
CP
右手
C
$\vec{p}$
$\overline{\nu}_L$
$\vec{\sigma}$
P
$\overline{\nu}_R$
左手反中微子
右手

8.5 非零质量

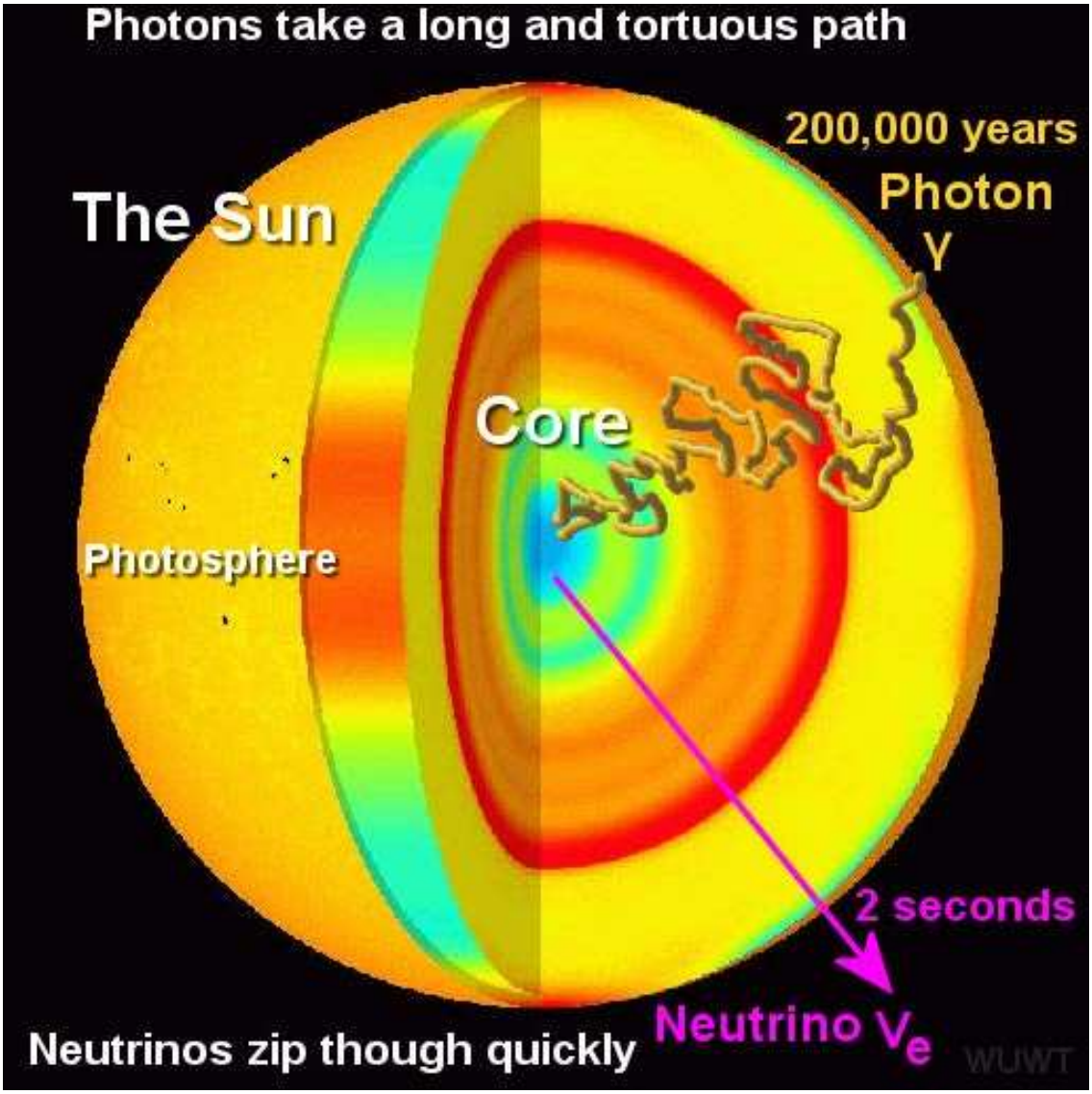
1959: 探索太阳的

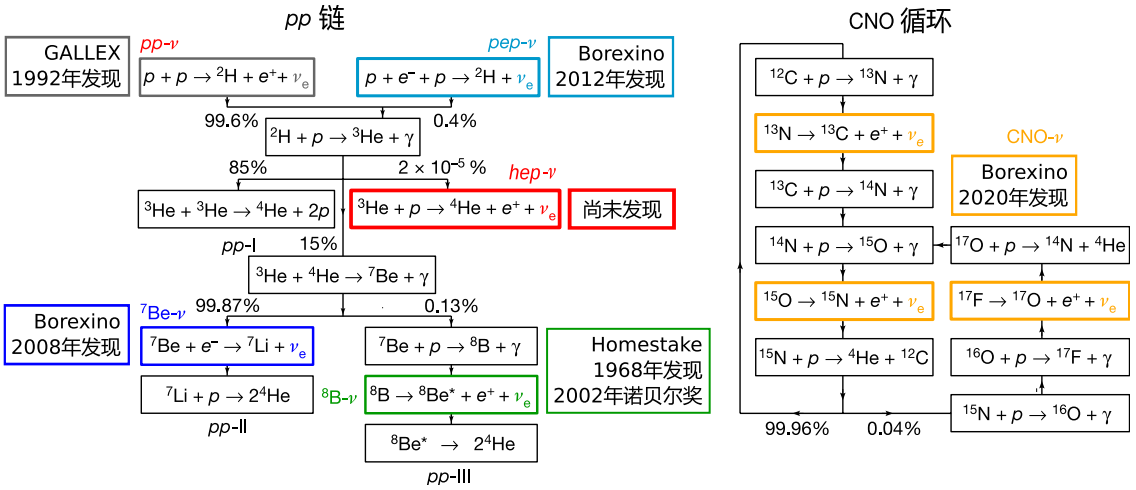


使用中微子看

- 太阳的寿命有多长？
- 太阳的能量从哪里来？
- 太阳的中微子可以揭示什

太阳核聚变与中微子

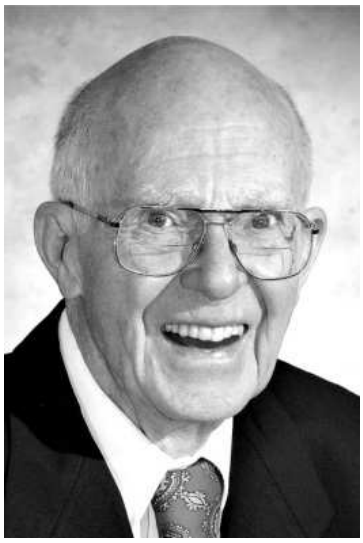




- 模型由太阳的化学成分、光度和温度间接验证。但中微子能提供直接证据。

Hans Bethe 1967 Nobel Prize

1960s–1990s 太阳中微子之谜



Ray 使用既有的探测原理，
100 倍，并放入了地下深
氯乙烯，几经碾转，终于
到了太阳中微子。

- 但观测值是理论预测的1/3。
- 观测一个月，可以回收出几十个 Ar 原子。
- 实验遭到质疑：几百吨液体中找几十个 Ar，可信吗？
- 20多年间，Ray 证实了实验可信。太阳中微子之谜越发严重。

投票 最多可选2项

如何解释 Ray Davis 实验结果

A 实验做错了

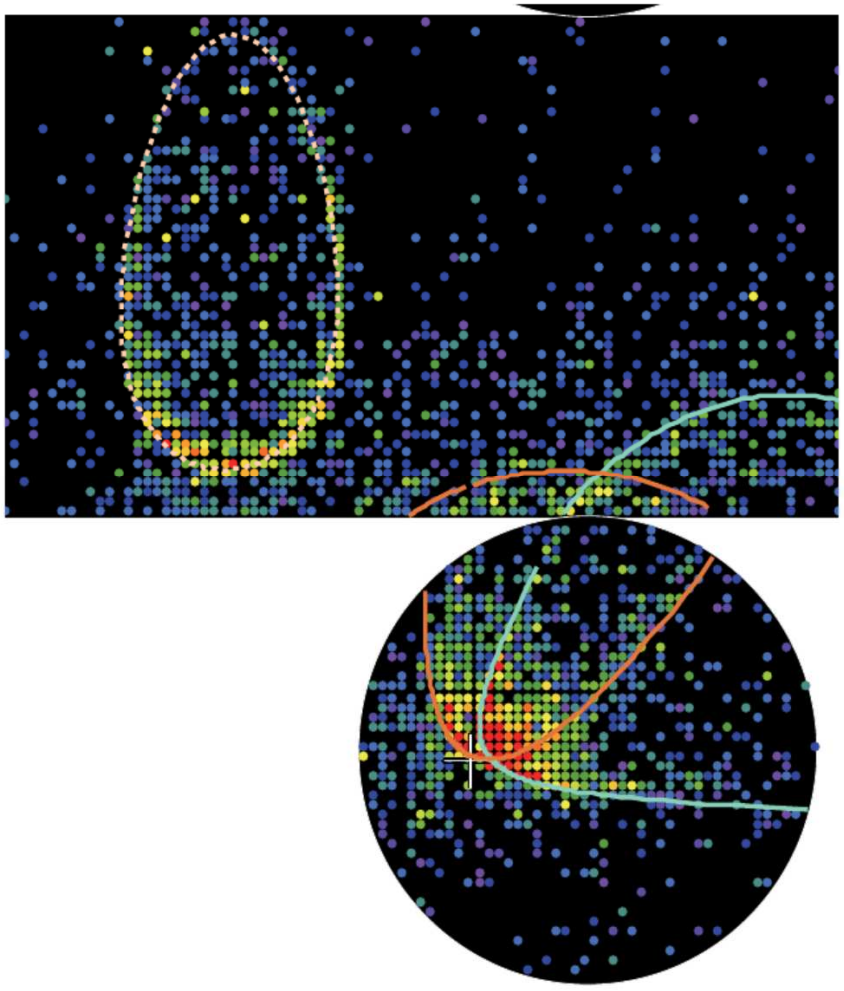
B 太阳要熄灭了

C 中微子凭空消失了

D 太阳还有其它产能机制

超级/神冈实验

- 容器内：带电粒子诱导 Cher
- 容器壁有光电倍增管，光电
- 相对化学放射计数是实时观



水 Cherenkov 实验：

- 水的实验给出，太阳中微子
  论预测的 65%。
- 实验确认了由Ray发现的太
- 为什么与理论值的差距不

- 1962年，Brookhaven μ 子
  用π^-衰变产生的中微子，
  μ^+不产生e^+。证明中微子

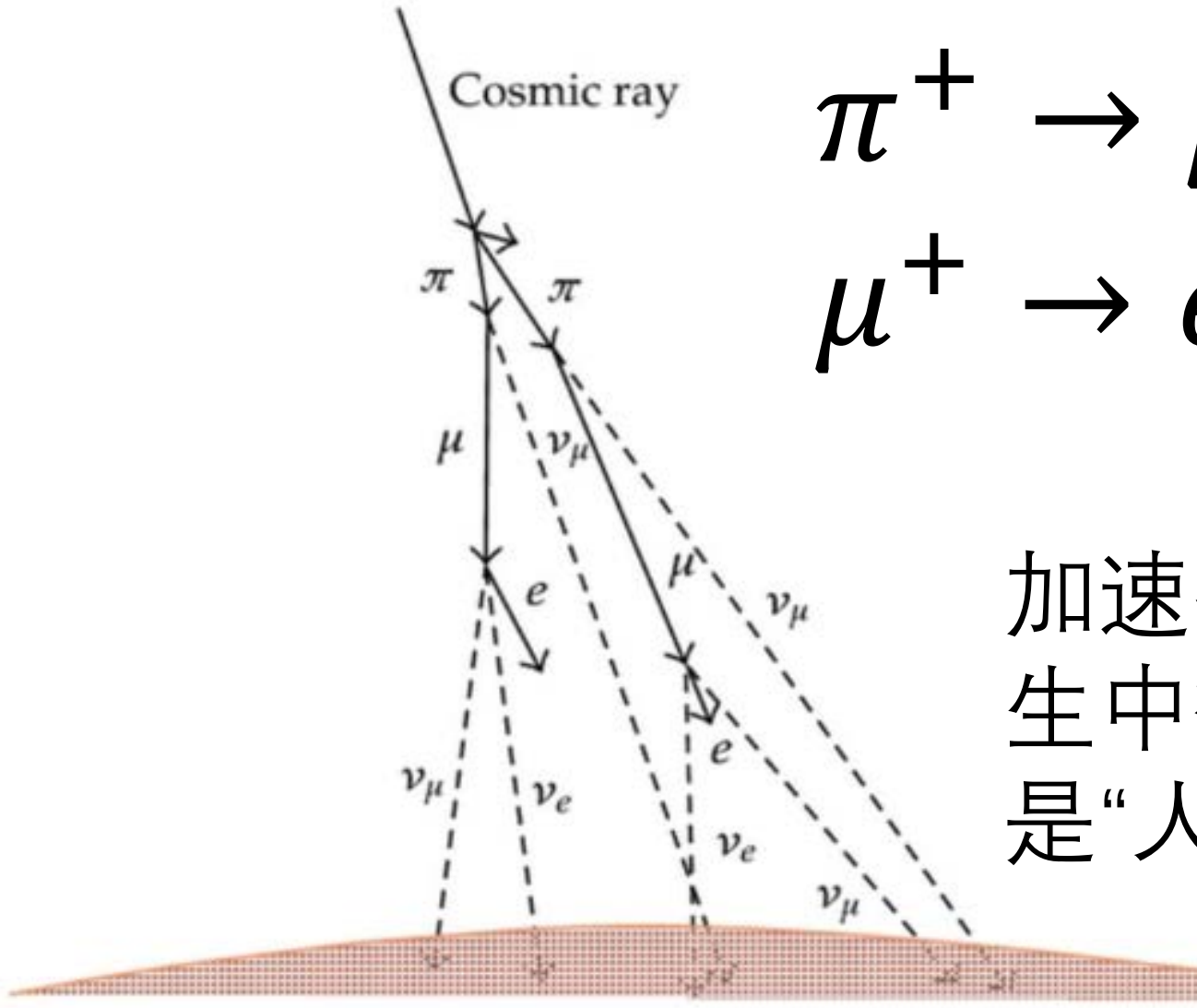
$$\bar{\nu}_\mu + p \rightarrow \mu^+ +$$

中微子有多个味，味之间可

1988年诺贝尔奖

大气和加速器中

质子加速，轰击靶，产生



$\pi^+ \rightarrow$

$\mu^+ \rightarrow$

加速

生中

是“人

Yukawa 1949 Nobel Prize

来自大气中微子的线索

$$\pi^+ \rightarrow \mu^+ \nu_\mu \ \ 26\text{ns}$$
$$\mu^+ \rightarrow e^+ \nu_e \bar{\nu}_\mu 2.2\text{us}$$

- ν_μ ν_e 比例应为2
- Kamokande 观察到中微子变
  - 从上飞来的中微子符合比例
  - 从地下飞来的中微子不符合
- 1998年：扩大15倍的Super-K

SNO 的巧妙设计：

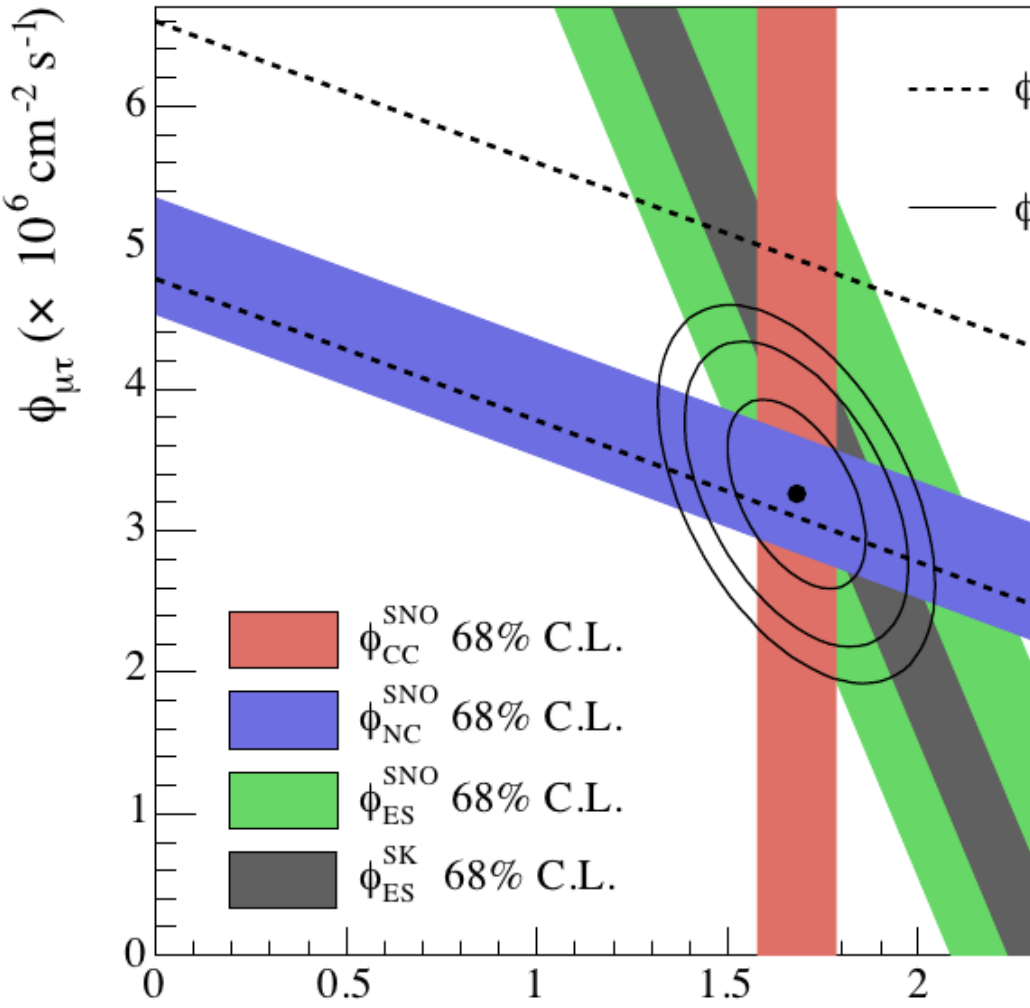
- 重水Cherenkov探测
- $\mathrm{d} + \nu \rightarrow p + n + \nu$
- 测量电子中微子流量，模拟
- $\mathrm{d} + \nu_e \rightarrow p + p + e^-$

测量所有中微子的流量，太阳中微子总流量。与理论计算相符！

Ray 发现的世纪难题终于被人类解答。

2015 Nobel Prize: Super-Kamiokande, SNO

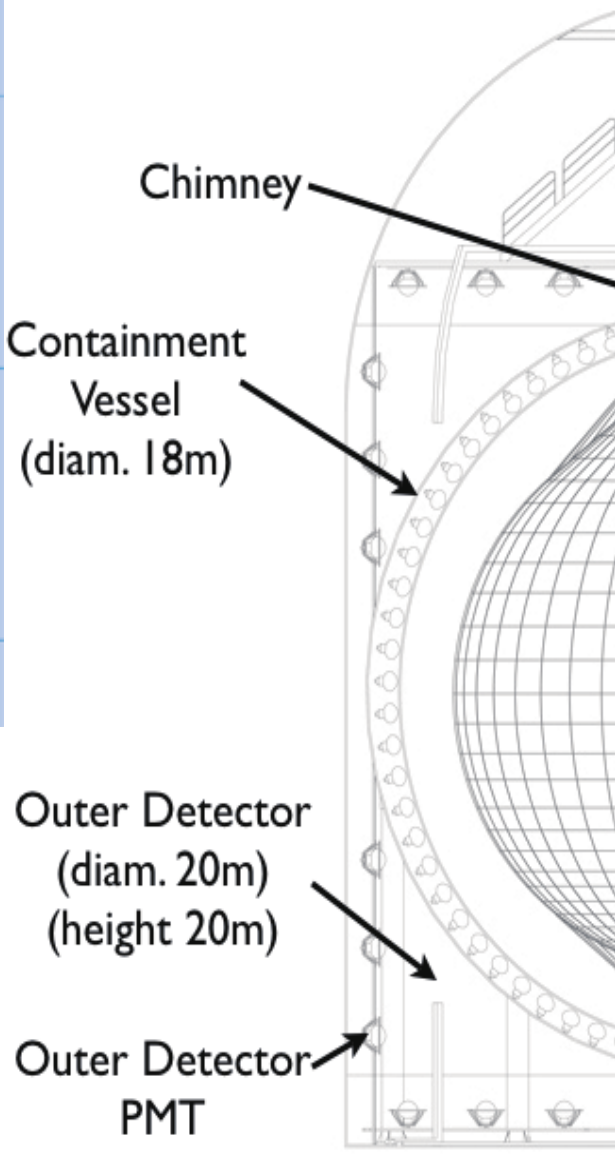
2001年太阳中微子消



BS05：太阳恒星模型
NC: neutral current
CC: charged current

SK: super-Kamiokand
SNO: Sudbury Neutri

KamL
确立中

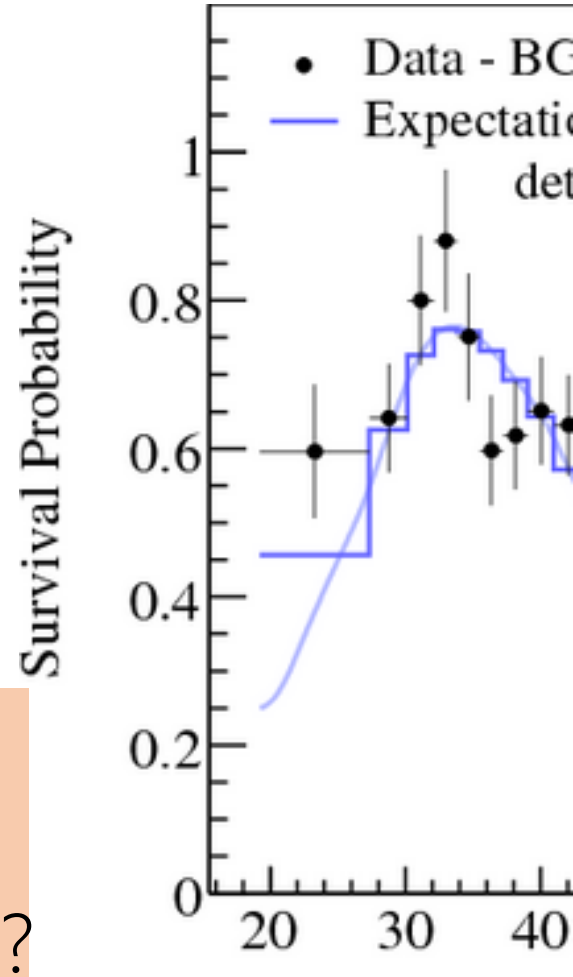


最大的液闪实验
巧妙利用反应堆
发现太阳ν振荡

中微子振荡的最

2002年，KamLAND 观测到反应堆
同时定出太阳中微子的
2005年，KamLAND 报告了完整的

L_0：中微子飞行距离
E：中微子能量



思考：反应堆中微子和太阳中微子的振荡为什么可以相互验证？

8.6 质量起源

中微子的世纪难题

1. 质量到底有多大？为什么如此之小？

   - 中微子的质量顺序，正序 $m_1 < m_2 \ll m_3$ 还是反序 $m_3 \ll m_1 < m_2$。
   - ν_3 到 ν_1, ν_2 的地球物质效应尚未观测到。
   - 江门中微子实验（JUNO）。

2. 正反中微子是同一种粒子吗

   - Dirac 型还是 Majorana 型费米子？
   - Majorana 型费米子存在吗？

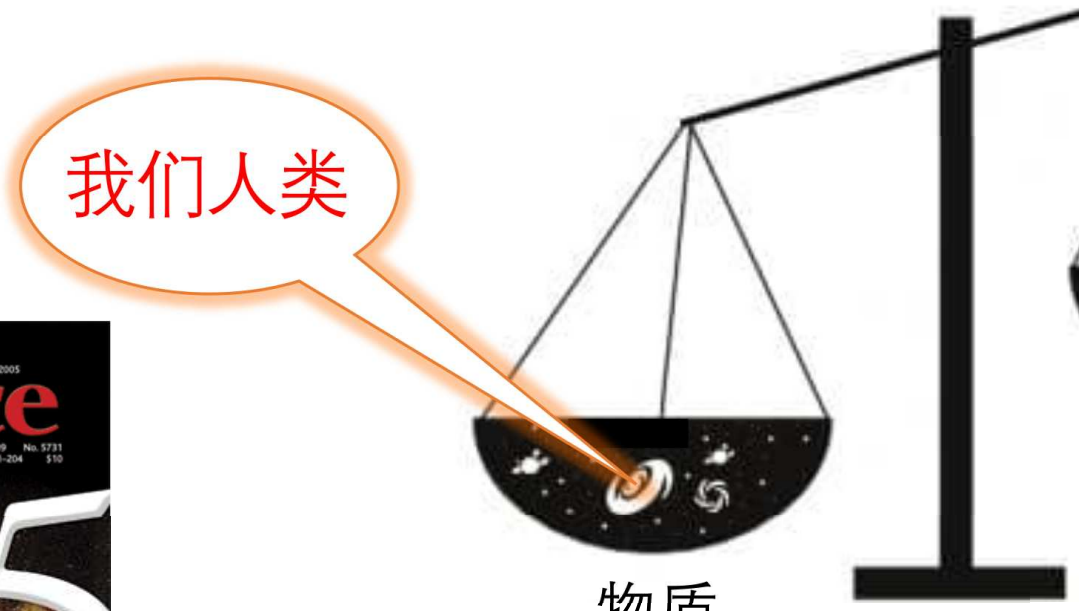
3. 轻子的 *CP* 破坏，δ_{CP} 有多大？

   - 为什么与夸克的 CP 破坏不同？
   - 可以解释宇宙由物质主导吗？

4. 中微子与 Higgs 如何作用？

   - 右手中微子存在吗？

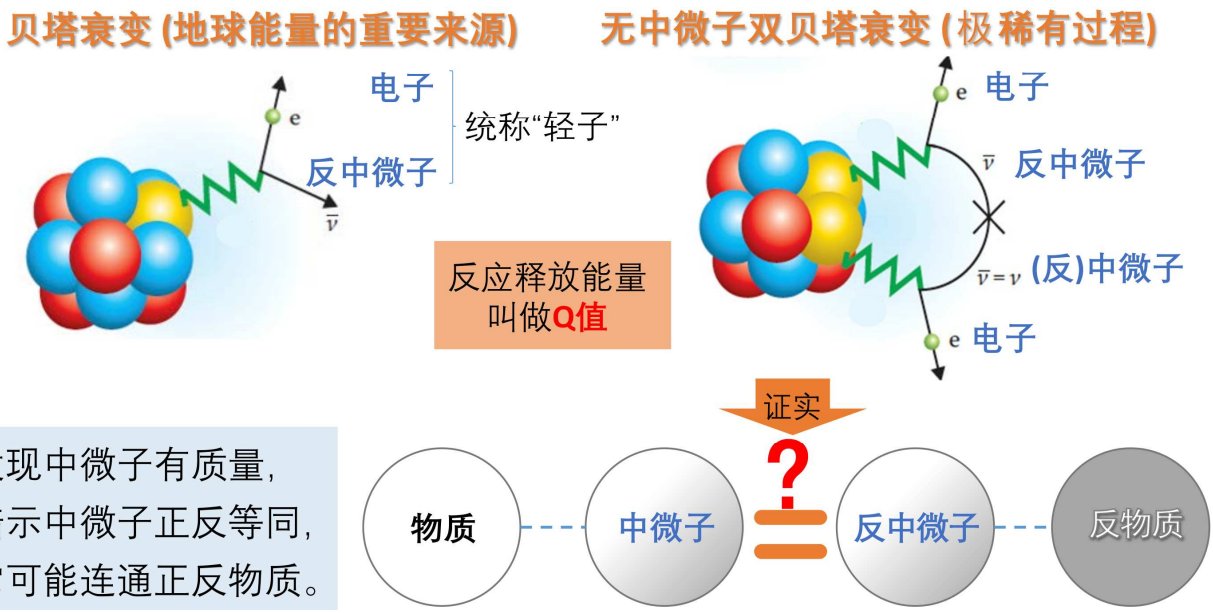
宇宙中物质从何起源？



2005和2021年均入选Science 125 个重大科学问题

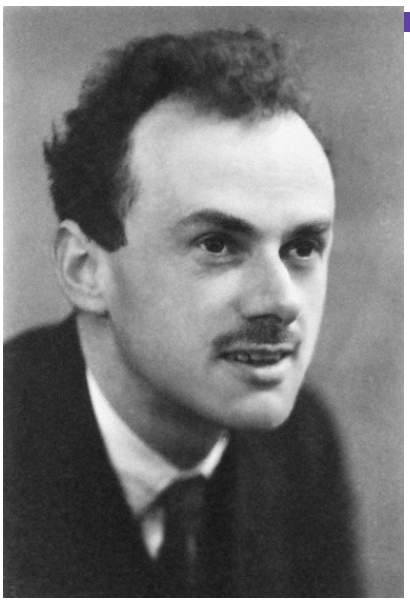
中微子质量和宇宙物质-反物质不对称的起源是什么？

——中国科协2021十大科学问题之



发现中微子有质量，
暗示中微子正反等同，
它可能连通正反物质。

Dirac方程与Majoran



Paul Dirac

$$\mathrm{i}\gamma^{\alpha}\partial_{\alpha}\psi$$

Dirac 矩阵

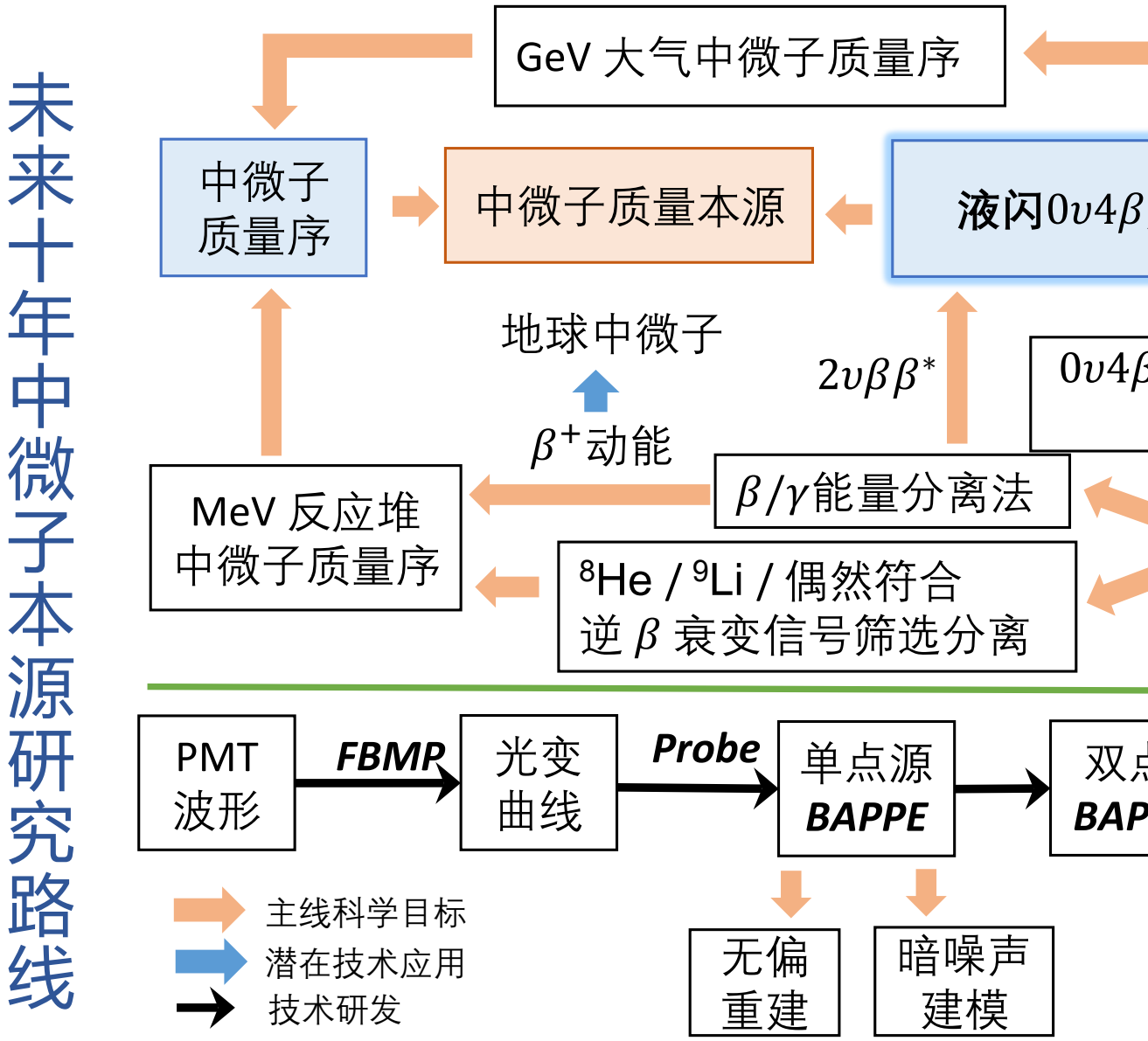
时空 指标

$$\mathrm{i}\gamma^{\alpha}\partial_{\alpha}\psi - m_{\nu}\,\psi^{*} = 0$$

$$\mathrm{i}\gamma^{\alpha}\partial_{\alpha}\psi^{*} - m_{\nu}\psi = 0$$

Majorana Spinor

中微子研究@清华



- **FBMP:** fast bayesian matching pursuit algorithm 快速贝叶斯匹
- **BAPPE:** Bayesian probe for point-like events 点源贝叶斯响应函
- **TCS:** tracking Cherenkov scintillation 径迹切伦科夫闪烁

8.7 江门-台山 ν

中微子的微小非零质量

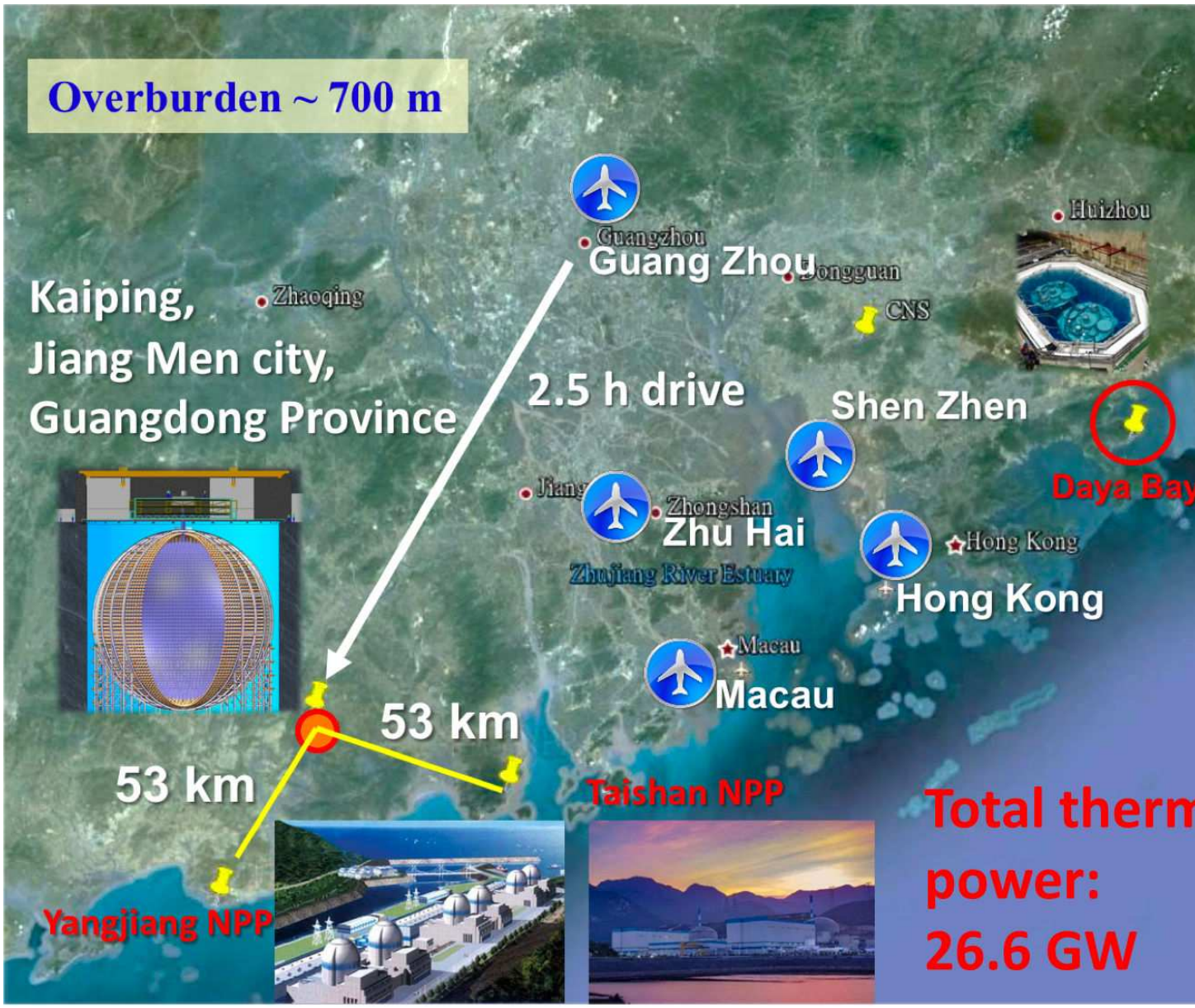
- Zero neutrino mass
  - Lee-Yang's two-component Weyl-Fermion ν model: explain the P-violation.
  - Inherited by the standard model (Weinberg)
  - Broken by solar (Homestake, Gallex/GNO/SAGE, SNO+), atmospheric (SuperK) and reactor neutrino (KamLAND) disappearance
  - Motivation: origin of the tiny non-zero mass, Majorana Fermion by GUT see-saw.

Mass ordering

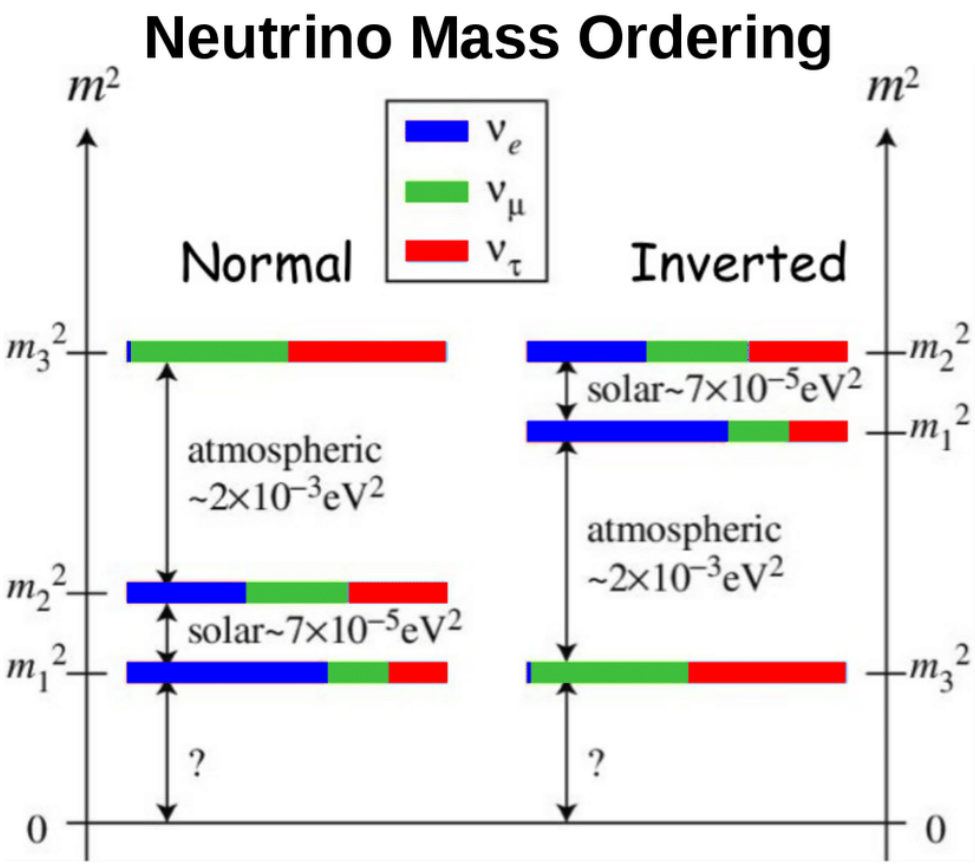
- Neutrino mass by quantum interference

广东：江门-台山地下中微子观测站

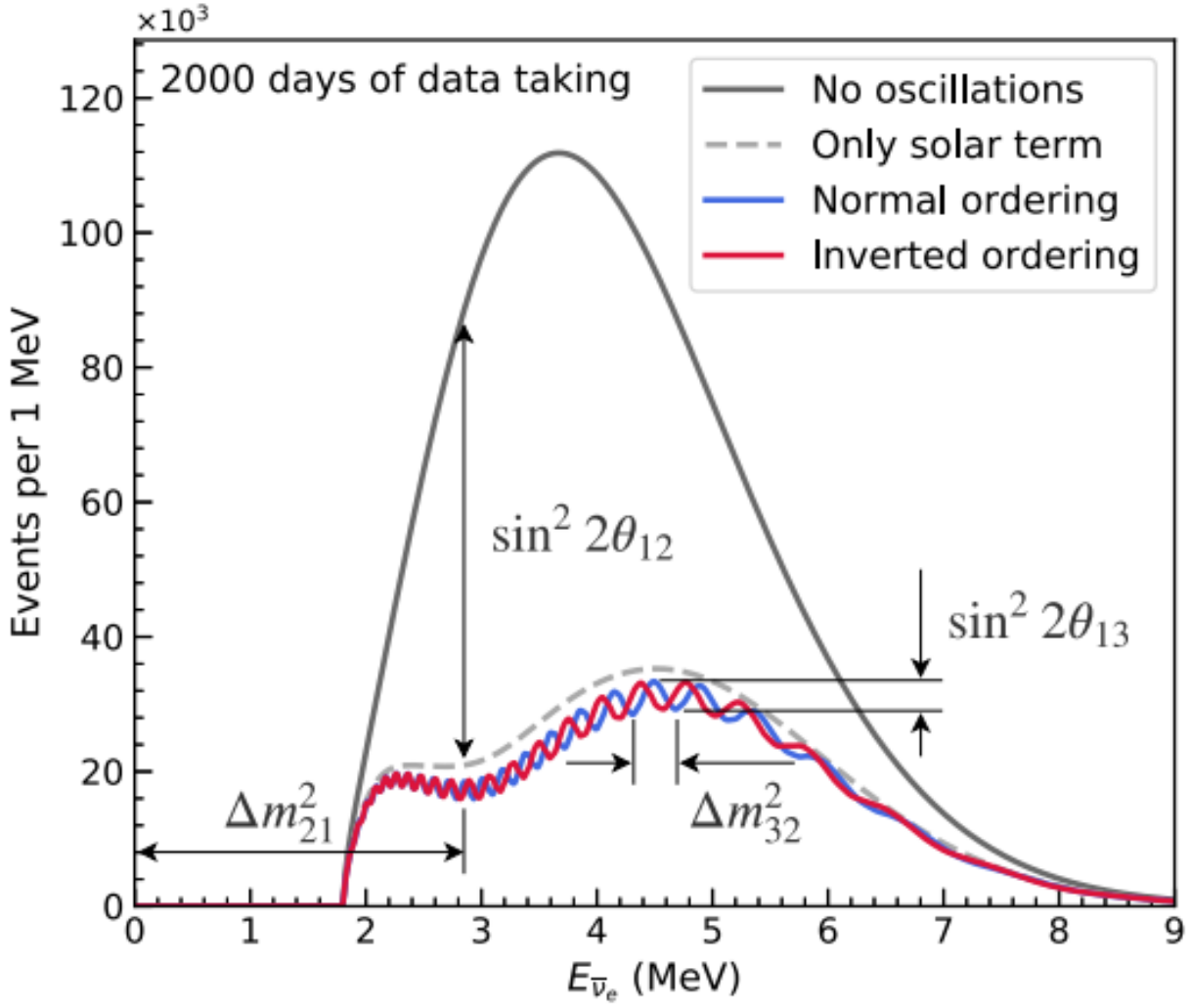
JUNO:=Jiangmen Underground Neutrino Observatory


Overburden ~ 700 m
Kaiping,
Jiang Men city,
Guangdong Province
Zhaoqing
Guangzhou
Guang Zhou
Dongguan
CNS
Huizhou
2.5 h drive
Shen Zhen
Daya Bay
Zhongshan
Zhu Hai
Zhujiang River Estuary
Hong Kong
Hong Kong
Macau
Macau
53 km
53 km
Taishan NPP
Yangjiang NPP
Total therm
power:
26.6 GW

测量质量顺序



精密测量反中微子能量是重中之重。



中微子质量顺序的意义

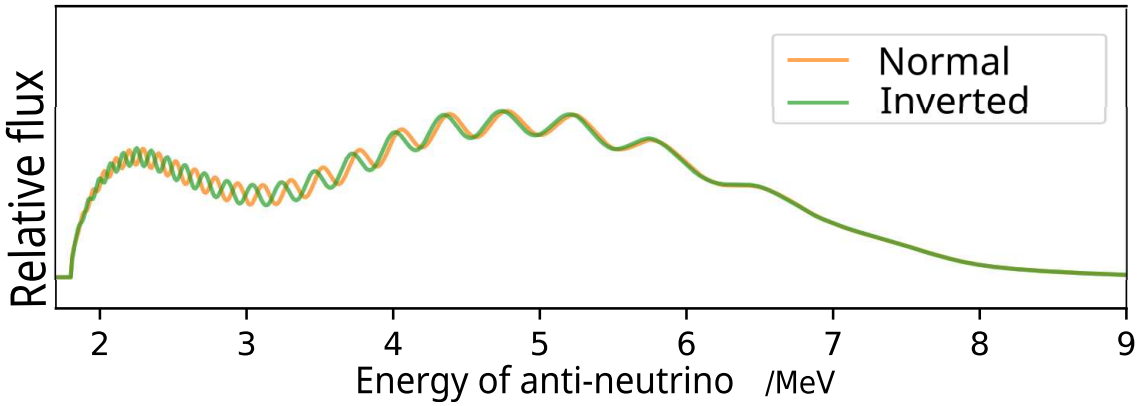
1. 质量排序本身是自然界的基本物理参数
   - 研究中微子质量本源
   - 寻找新物理规律
2. 决定超新星中微子传播过程中的振荡
   - 推定超新星爆发时 ν_e, ν_μ, ν_τ 的比份
   - 揭示超新星爆发机理
3. 决定 $0\nu\beta\beta$ 衰变实验的世界格局
   - 如果是正排序，人类还需要把现有 吨级 实验扩大 1000 倍
   - 寻找自然界中的 Majorana 费米子

- 寻找自轻子数破坏的稀有衰变

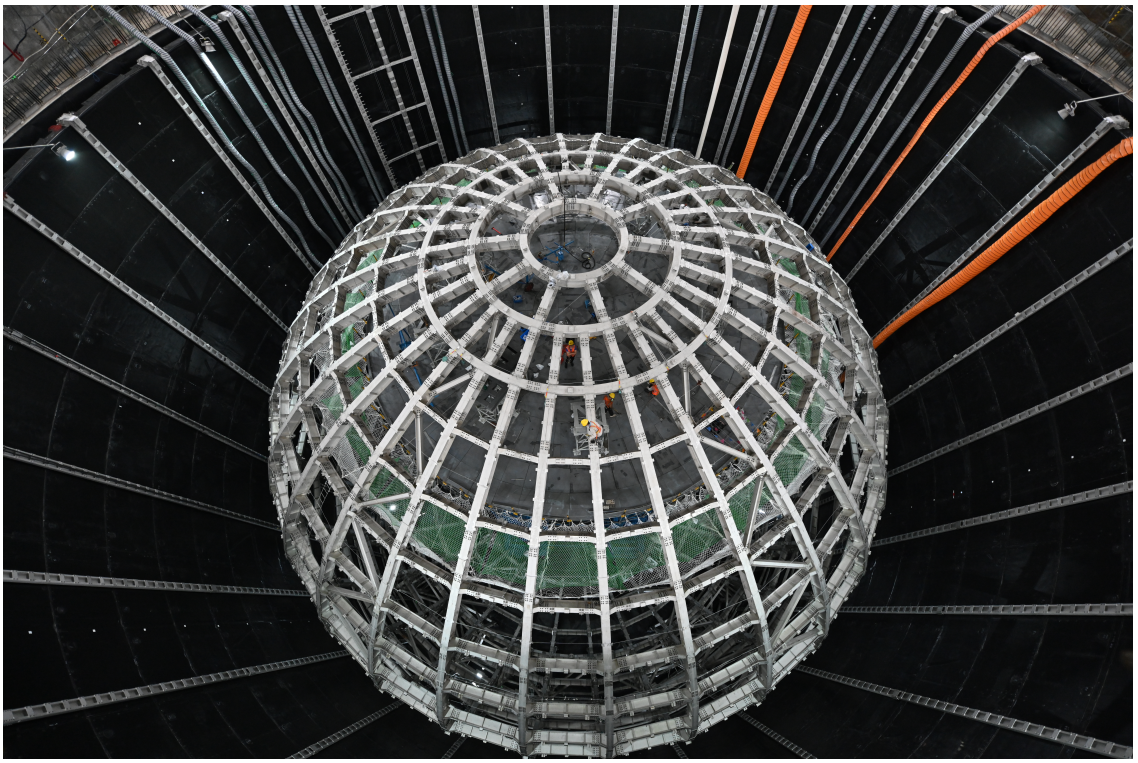
4. 与轻子 CP 破坏的测量耦合
   - 关乎自然界的“左右正反”对称在多大意义上近似成立
   - 探寻宇宙正反物质不对称的起源

能量分辨率

Energy resolution is the figure-of-merit that we push to extreme.



53 km 之外的中微子振荡后能谱

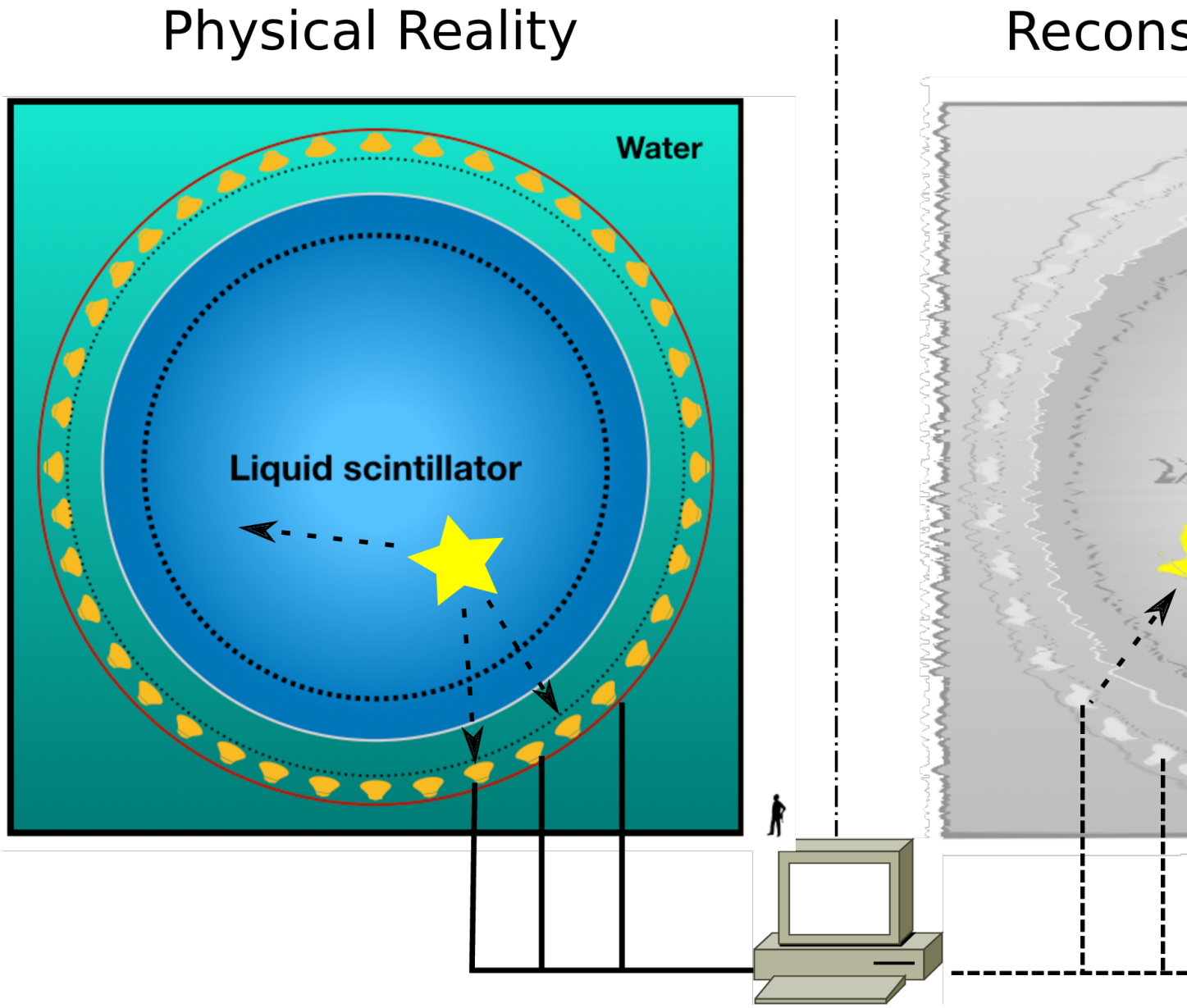


JUNO under construction

重要性能指标

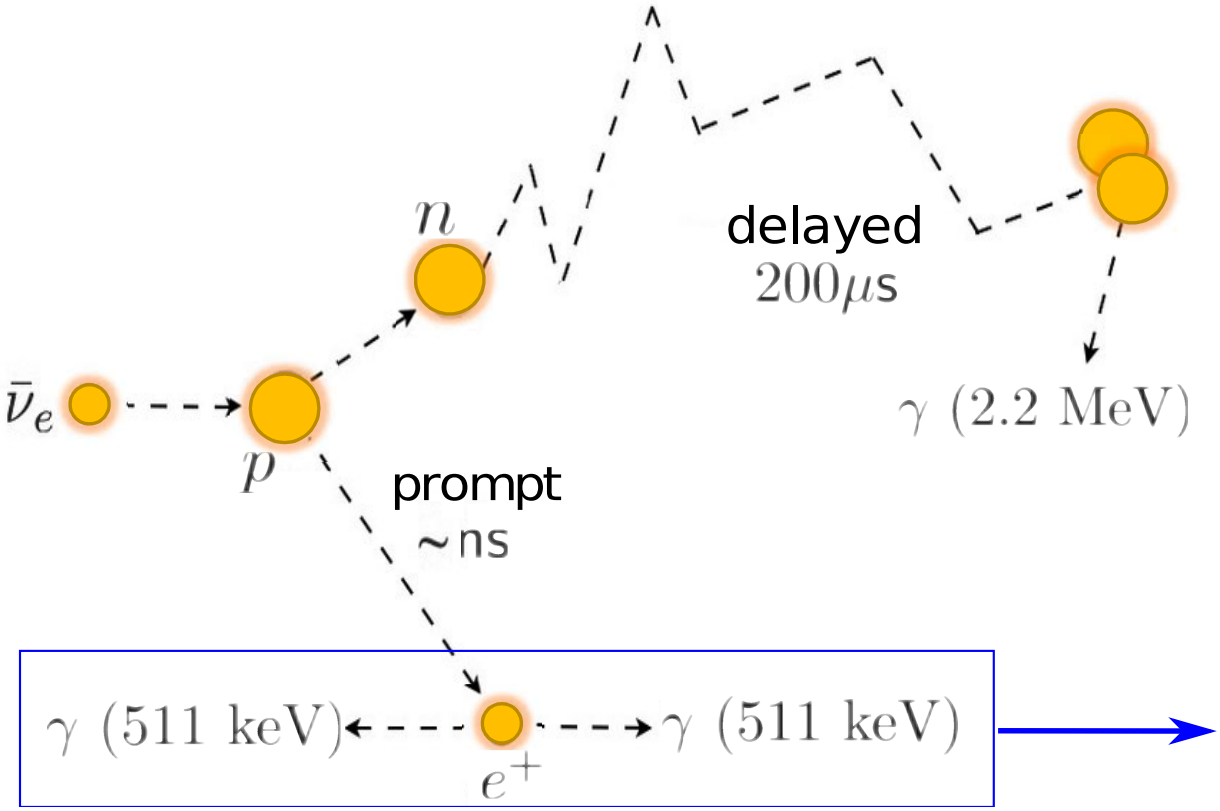
- 每提升现有能量分辨 0.5% ，都可以缩短 25% 的运行时间。

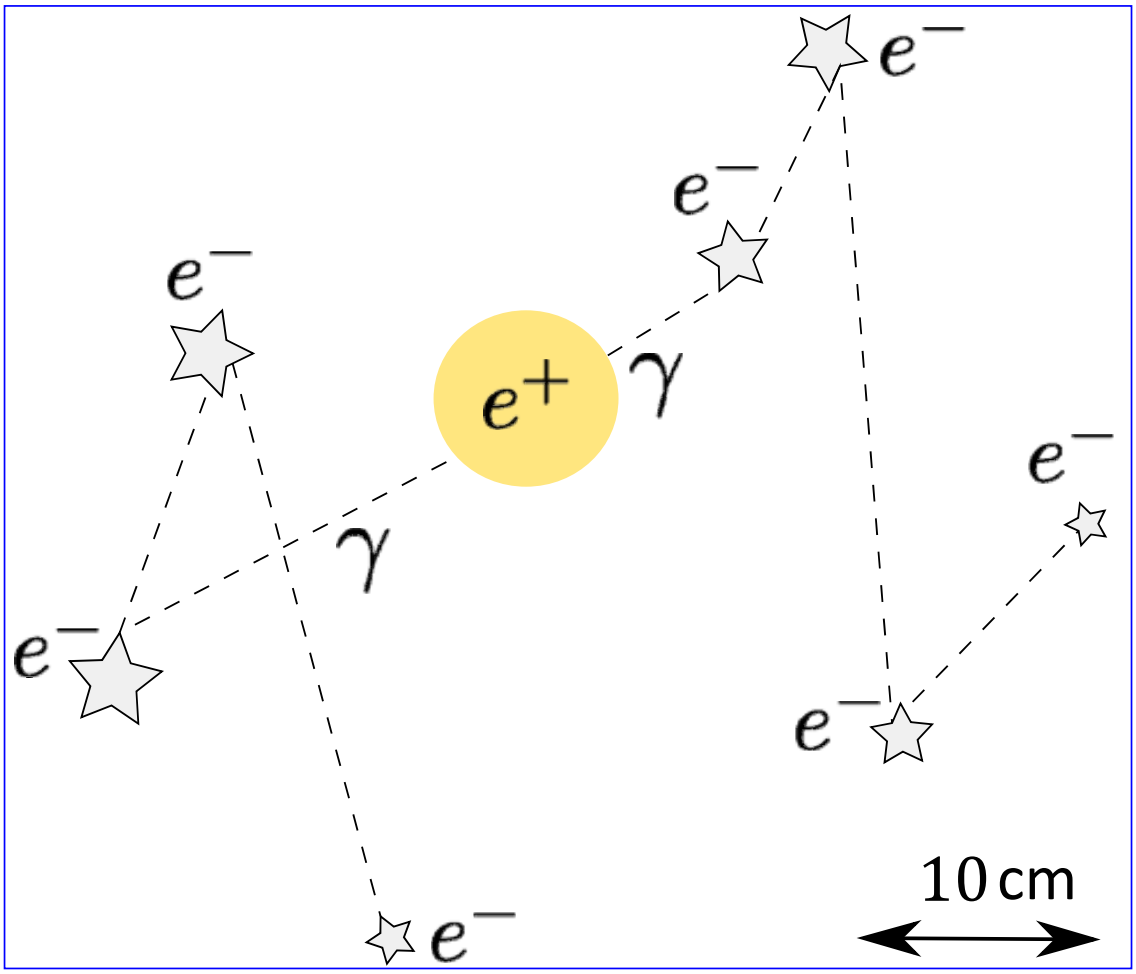
测量原理



- “非成像”光学探测器，电离发光，单个光子由光电倍增管检出。

逆贝塔衰变


n
delayed
$200\mu s$
$\bar{\nu}_e$
p
γ (2.2 MeV)
prompt
~ns
γ (511 keV)
e^+
γ (511 keV)



诱导蒙卡成功后，我们得到

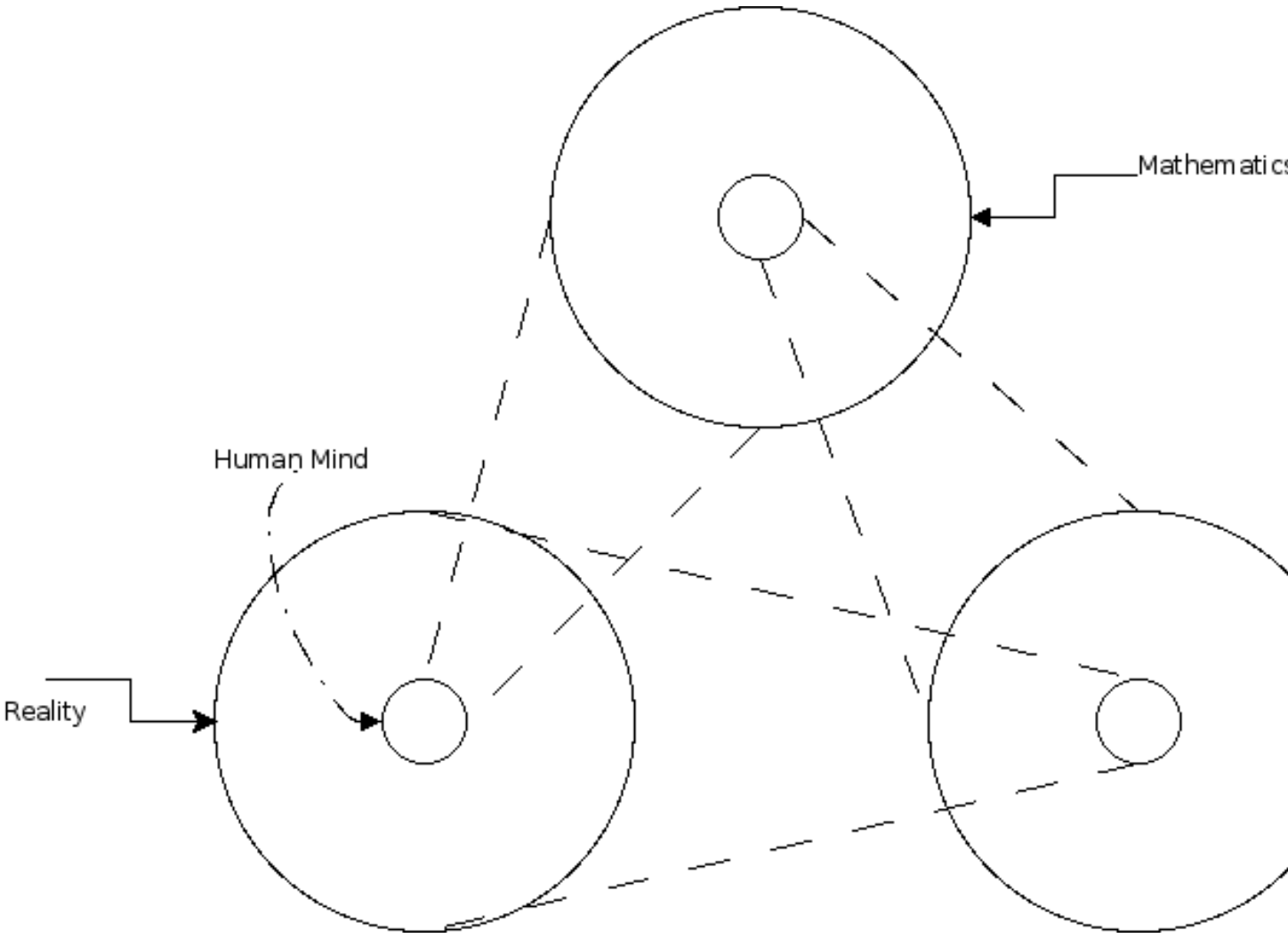
- 一系列随机数种子，编码了 e^+ 在探测器中的详细演化历程。
- 等价于在液闪中成像！

欢迎

期待大家选择 *Gamma* 粒子物理大作业

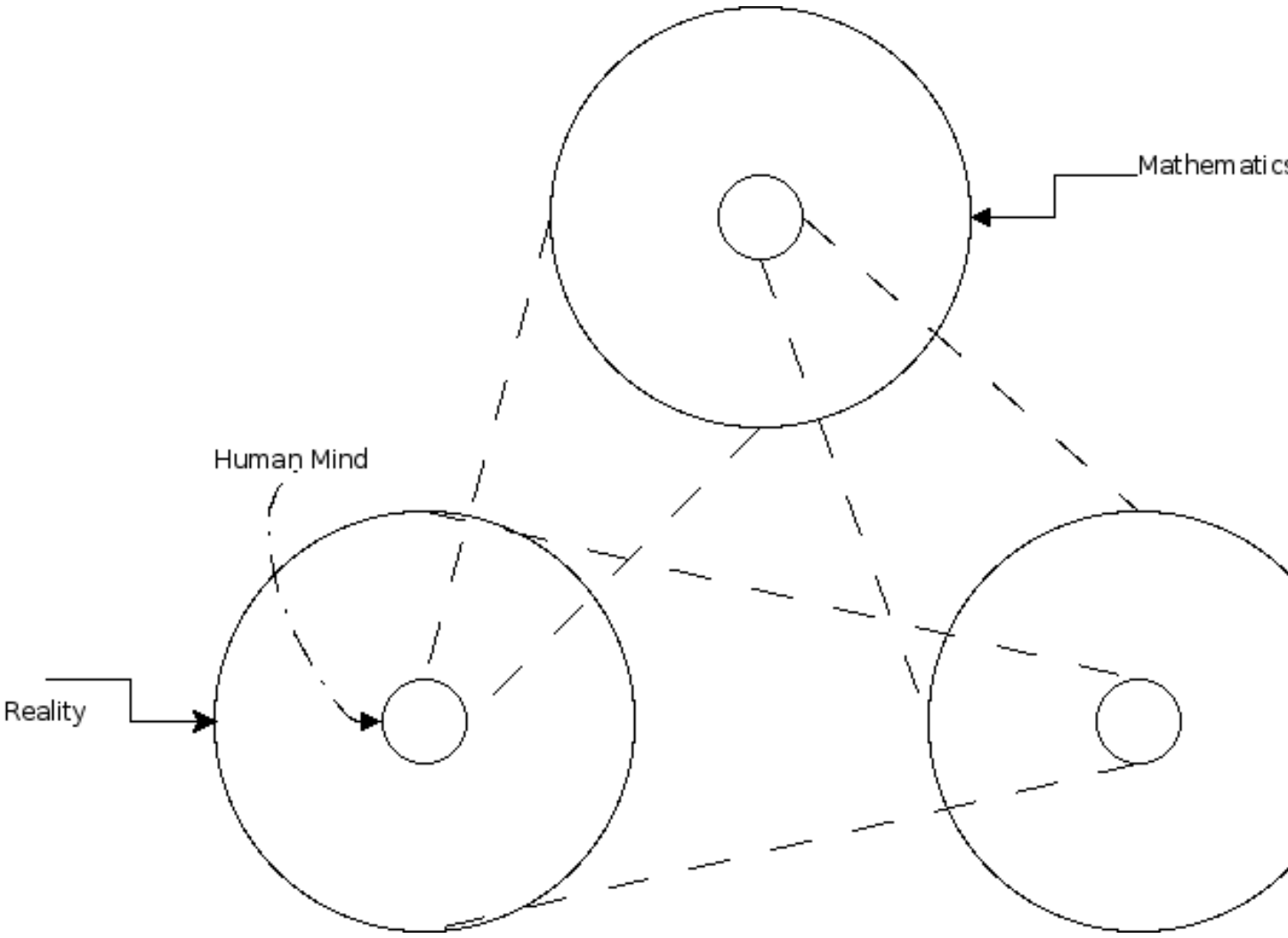
8.8 实验物理

柏拉图世界观



- 数学世界独立存在，由它推演出物理世界

实验物理



- 实验仪器是从物理世界到人类的一条信息通道
  - 实验仪器的分辨率有限，我们在进行有损通信

简单的物理实验

Albert Einstein (1926)

Whether you can observe a thing or not depends on the theory which you use. It is the theory which decides what can be observed.

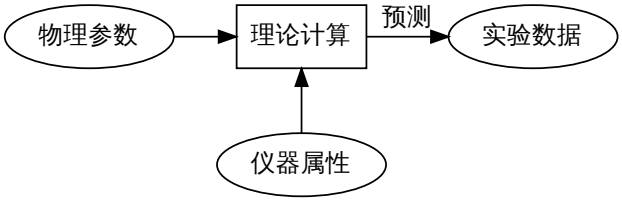
贝叶斯公式

$$p(\theta_T|D,T) = \frac{p(D|\theta_T,T)p(\theta_T|T)}{\int p(D|\theta_T,T)p(\theta_T|T)\mathrm{d}\theta_T}$$

- D 是实验数据，T 是理论， θ_T 是理论参数。

复杂的物理实验

- 理论计算预测时，不仅有理论参数 θ_T 还有仪器参数 θ_E。

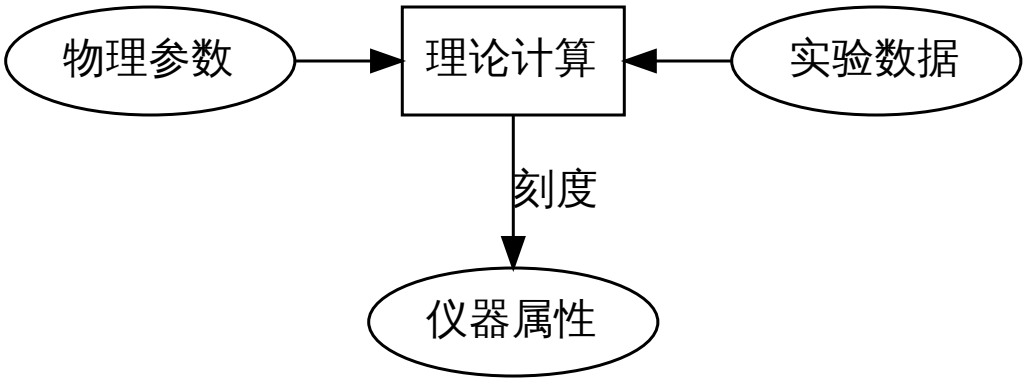


- 仪器变得复杂，需要自己标定仪器属性，才能给出合理的预测。

$$p(D|\theta_T,\theta_E,T)$$

刻度：自行实验确定仪器属性

- 使用已知的 γ 放射源，例如 $^{68}\mathrm{Ge}$



液闪的

- 吸收长度
- 淬灭因子
- 散射角分布
- 色散关系

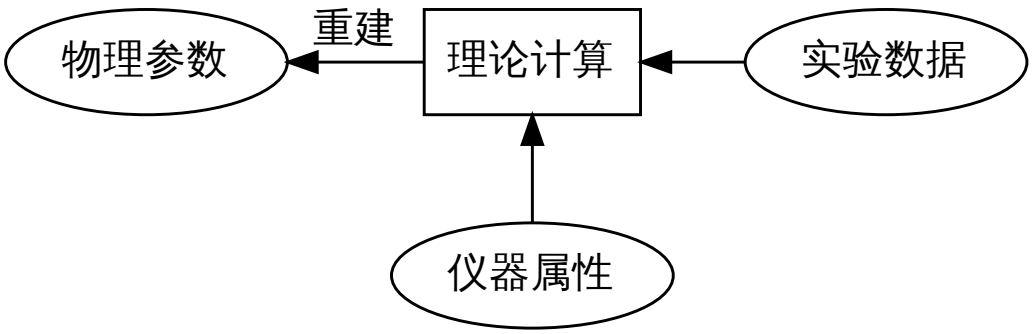
PMT 的

- 量子效率
- 时间特性

反过来使用贝叶斯

$$p(\theta_E|D,\theta_T,T)=\frac{p(D|\theta_T,\theta_E,T)p(\theta_E)}{\int p(D|\theta_T,\theta_E,T)p(\theta_E)\mathrm{d}\theta_E}$$

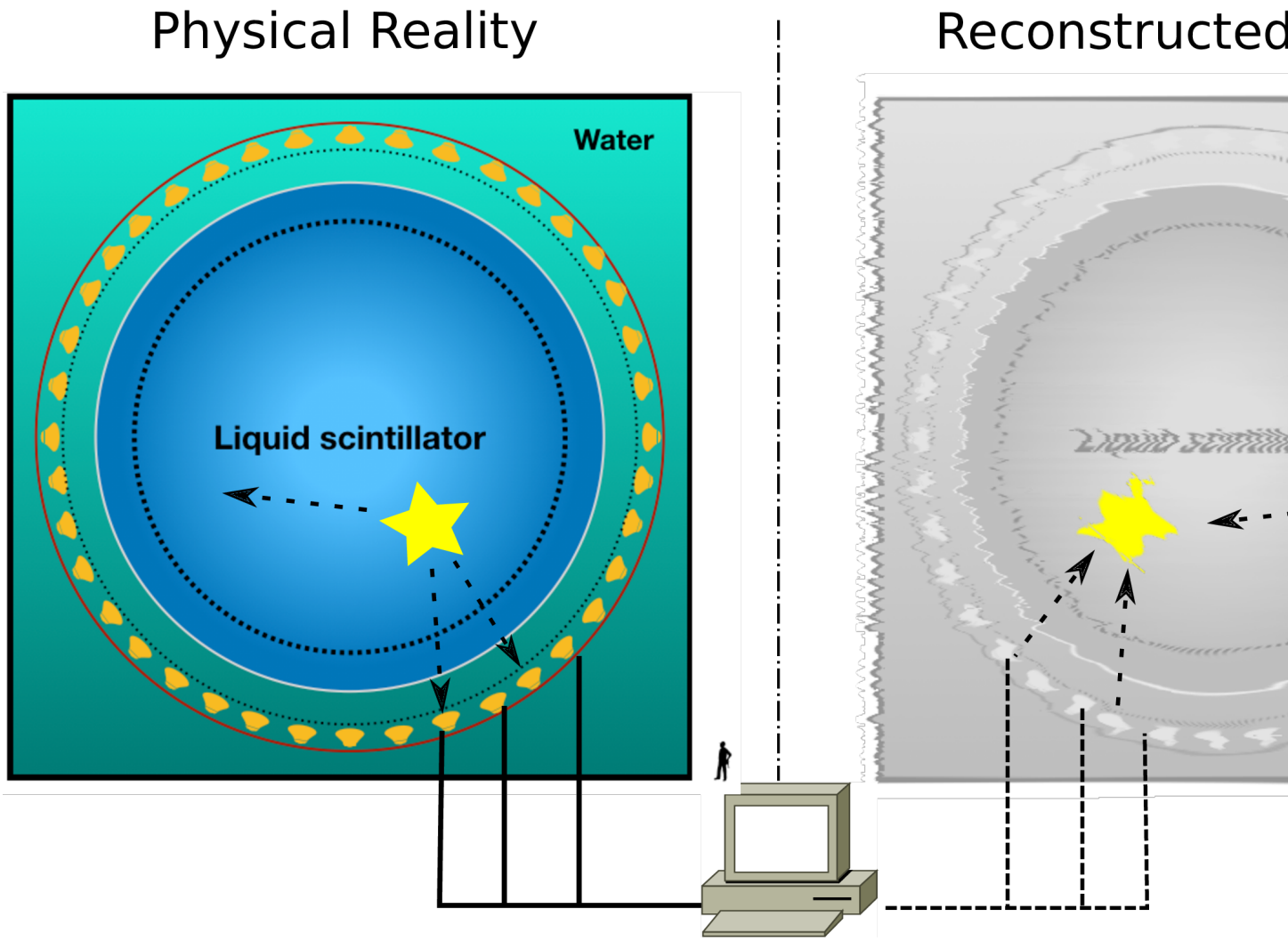
重建：推断理论中的参数



- γ 的能量，位置
- ν_e 的方向
- μ 的时空径迹
- 中微子质量顺序
- θ_{13} 振荡参数

$$p(\theta_T|\theta_E,D,T)=\frac{p(D|\theta_T,\theta_E,T)p(\theta_T|\theta_E,T)}{\int p(D|\theta_T,\theta_E,T)p(\theta_T|\theta_E,T)\mathrm{d}\theta_T}$$

举个例子



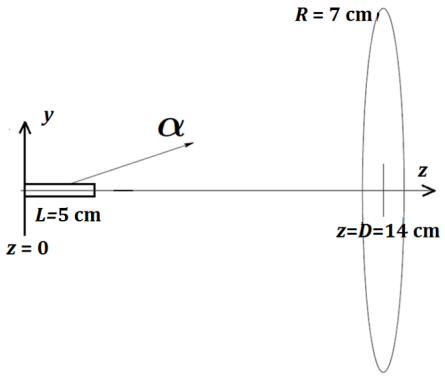
- 建立模型解释物理世界发生的现象，通过统计方法反推本质。

8.9 蒙特卡罗

引例：探测效率估计

实验仪器往往很复杂

- 有一个长度为 $L = 5\,\mathrm{cm}$ 的细长棒状的 α 放射源，棒的左端位于原点，棒体与 z 轴平行。放射源在棒中均匀分布，发射的 α 粒子在空间中的角分布各向同性。
- 在 z 轴正向距离原点 $D = 14\,\mathrm{cm}$ 处放置了一个圆盘状的探测器用来记录 α 粒子，圆盘半径 $R = 7\,\mathrm{cm}$，轴线与 z 轴重合。



一个大积分

- α 被观察到与否，有多种因素影响
  - 棒表面的哪里出射
  - 向哪个方向出射
  - ~~飞行过程中有没有被碰撞，有没有被电磁场影响~~
  - 有没有撞到圆盘上，撞上之后有没有信号
  - 信号是不是足够大被仪器记录
- 所有因素做笛卡尔积
- ……只为了最终得到一个数字： α 的探测效率 $\in [0,1]$。

理论引子：量子论

- 微观粒子可能的路径按照作用量叠加起来，给出量子态的概率振幅。

$$\int_{\vec{s}(t)\in S} \exp\left[-i\int \mathcal{L}(\vec{s},\dot{\vec{s}})\mathrm{d}t\right]\mathrm{d}\vec{s}$$

- 连续统无穷维积分，无严格的数学测度定义。
- ……只为了最终得到一个数字：观测结果出现的概率

计算积分

设 $X_1, X_2, \cdots X_n$ 是独立同分布的随机变量序列，且 $X_n \sim U[a,b](b>a)$ ，$f(x)$ 是 $[a,b]$ 上的连续函数。

- 提供了一种计算定积分的方法
- 用计算机产生服从 $[a,b]$ 上的均匀分布的随机数 $\{X_i\}$，然后得随机序列 $\{f(X_i)\}$，则：

$$\int_a^b f(x)\mathrm{d}x \approx \frac{b-a}{n}\sum_{i=1}^{n} f(X_i)$$

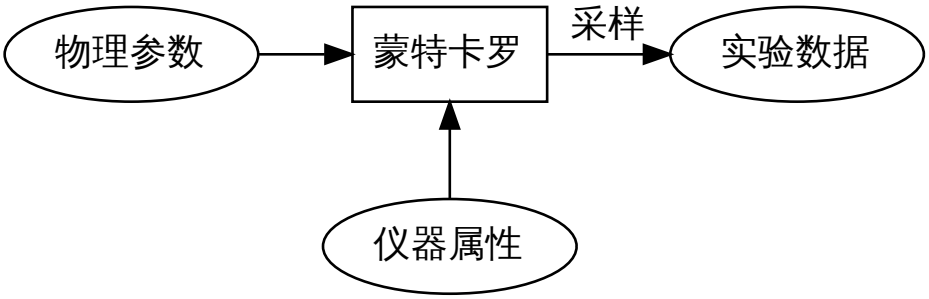
为什么随机数可以胜任?

- 自然小信息 $\rightarrow$ 大信息复杂过程 $\rightarrow$ 人类小信息
- 自然小信息 $\rightarrow$ 无所不用其极的方法 $\rightarrow$ 人类小信息

理论计算被计算机蒙特卡罗模拟替代

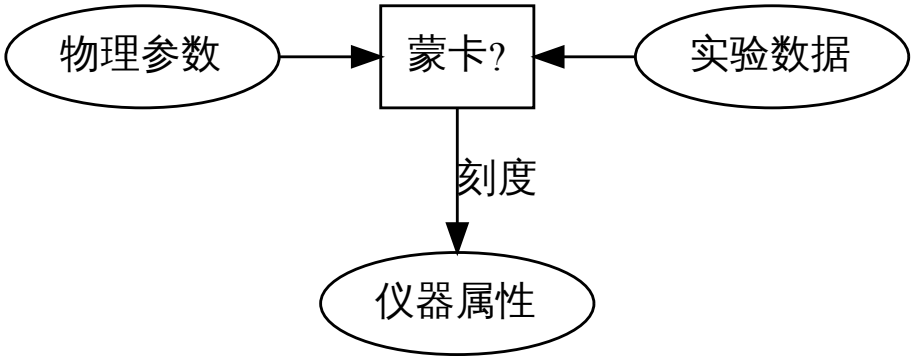
- 没有解析形式，人类无法有效运算。
- 使用程序替代理论模型，模型由程序描述。

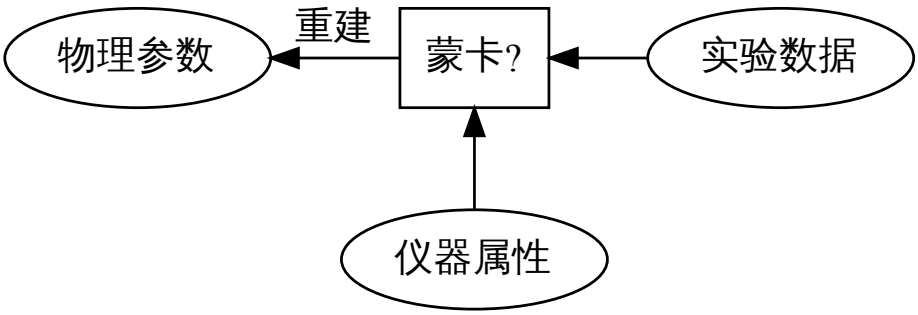
狭义的“蒙卡”



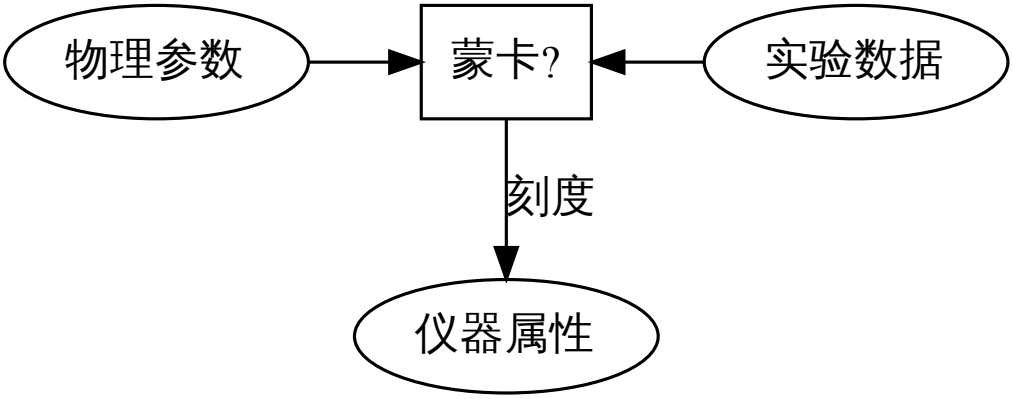
- 代价：用采样代替了预测
- 只能“正向”前进，无法像解析式一样“掉头”。

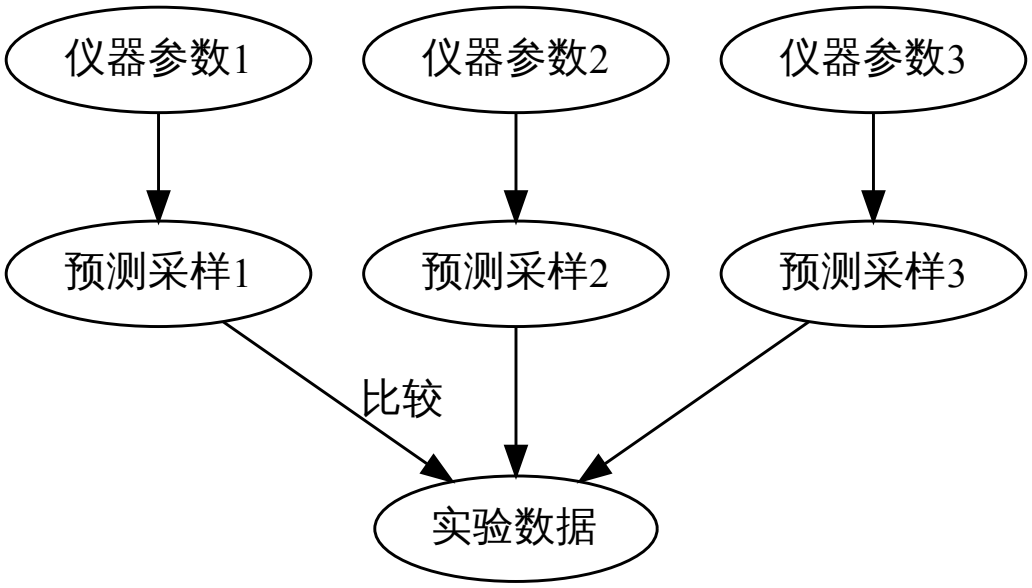
$$p(D|\theta_T, \theta_E, T)$$





探测器刻度的分析方案

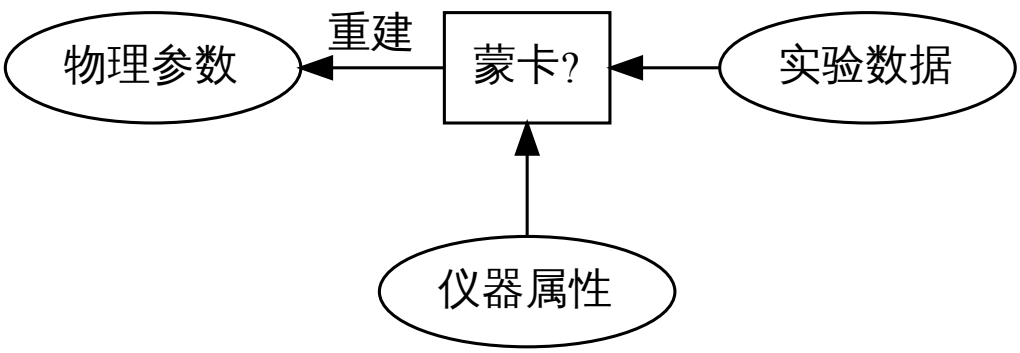


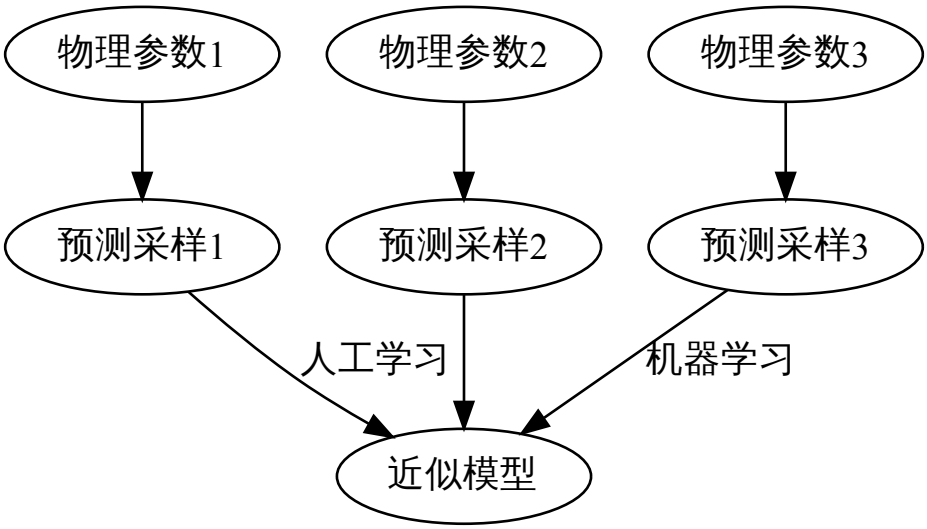


逆向蒙卡

- 散弹枪形式，穷举所有可能的仪器参数，分别用正向蒙卡预测
- 选用与实验相符的仪器参数，实现了“逆向蒙卡”。
- 好用，只是慢，但是值得

近似模型的事例重建





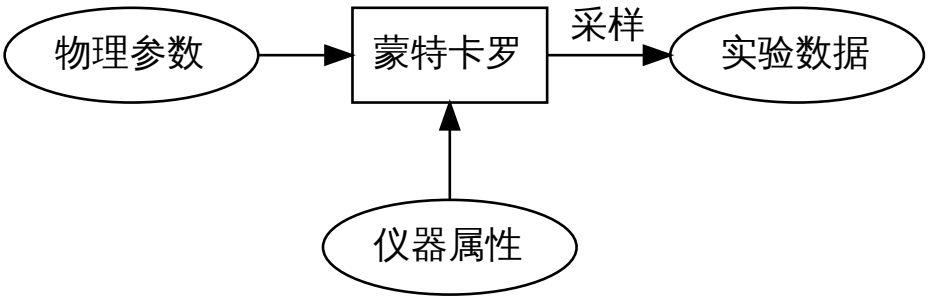
近似模型

- 人为寻找规律，人为概括，得到一个近似的 $p(D|\theta_T, \theta_E, T)$
  - 应用贝叶斯公式求解
- 机器寻找规律，得到一个 $(D, \theta_E, T) \to \theta_T$ 的映射，不再具有数理统计性质
  - 误差分析必须另外移植缝合

8.10　大作业

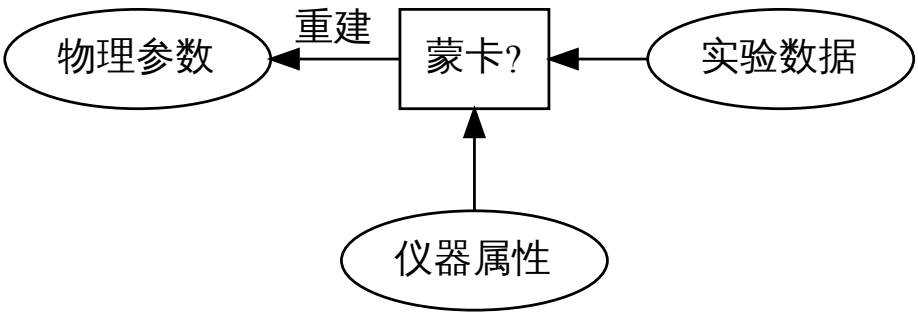
大作业安排

正向第一阶段



- 2024-07-11 – 2024-07-31
- 模拟实验测量

逆向第二阶段



- 2024-07-28 – 2024-08-17
- 分析数据
- 测量模型参数
- 黑盒分数按排名

实验测量的模拟

输入 物理对象的信息，随机因素的概率分布

输出 多阶段，最终输出为模拟的实验测量原始数据

可视化 中间结果用图表描述

采分 各阶段的实现，最终输出的正确性，作业报告

分组

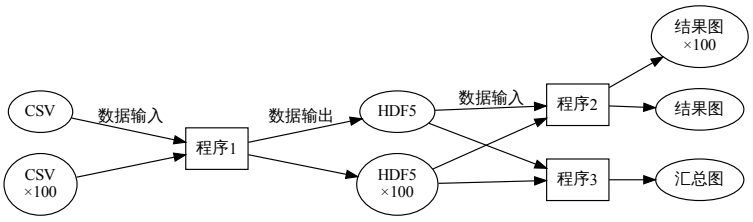
- 同学们先联络好，组队信息在网络学堂提交
- 每队至多三人
  - 单人队：大作业得分 $\times 1.03$
  - 三人队：每人大作业得分 = 队伍得分 $\times 0.95$
  - 不同队伍间请勿直接交换代码
- 大作业带有较高难度的附加任务，至多在（黑盒）满分基础上加 10%。

大作业

- 物理学是实验科学
- 实验研究的两个方向：
  1. 演绎，模拟
  2. 分析

自定义大作业

批量处理海量数据



命令行探索之后，要将数据处理方法自动化

- 实验要调用很多命令和程序

  - 重复运行，控制变量：以不同条件多次测量，探索规律
- 处理很多数据，有很多中间结果，依赖关系复杂
- 程序有更新怎么办？数据有更新怎么办?

数据流水线的构造目标

复现 原则的要求
要记录下来以什么样的顺序和参数运行什么命令，执行什么程序。

思路和要点

- 把流程系统化成输入、输出与过程三要素。
  - 而向数据编程，data-driven programming
- 系统表达输入数据、输出数据和中间结果的依赖关系，
  - 成为“可执行的说明文档”
- 高效执行，包括并行处理和整合超级计算机等。
- 错误恢复
  - 修正错误后，可以从最后一步正确的数据开始继续执行。

Make 是最佳工具

- make 工具已经有 40 多年的历史，最初用来管理 C 语言程序的编译。
  - 根据依赖关系决定命令执行顺序
- GNU make 是 GNU 运动中，对 make 进行的扩展，更适合管理数据
  - 版本 $\geq$ 4.3 支持 多目标规则

```
make --version
```

```
GNU Make 4.3
Built for x86_64-pc-linux-gnu
Copyright (C) 1988-2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

作用

1. 实现 复现 要求
2. 管理程序运行，在超级计算机上运行
3. 从错误中恢复

Make 要素：Makefile 对流程的定义

- 文档

```
info make
man make
```

- GNU Make 恰好可描述数据流水线的三要素：输入、输出与程序。
- 基本结构

```
target: source
        program source target # 必须 TAB 起始，记录如何做
```

```
输出目标：输入源
    执行的命令 $^ $@
```

  - `$^` 代表输入源
  - `$@` 代表输出目标

- `make -n` 用来测试即将执行的命令。
- 基本语法单元：清晰写明输入数据，输出数据和计算方法

参考书

John Graham Cumming, The GNU Make Book

8.11 备用分隔

8.12 Python 环境

Jupyter

- Jupyter 从 IPython 的网页界面发展而来
- 安装使用

1. `apt install jupyter`
2. 配置文件
3. 安装 jupyter kernel
4. 本地网络测试

Jupyter 的特点

- 最强大的优势:
  - 程序与文本混合编排，literate programming，鼓励了科学结果的复现。
  - 文章和程序既然在描述同一件事，可以自然地写在一起。体现“一次”原则。
- 其它优势:
  - 图文混排
  - 支持 Python 之外的其它语言环境
- 弱点:
  - 网页上打字，没有强大的编辑器。(Emacs IPython Notebook，填补了空白)
  - 要进行高强度大规模的运算，需要与其它工具配合。不够成熟。
  - 代码的复用需要额外步骤。

8.13 SciPy

SciPy: Scientific Python

- SciPy 在 NumPy 的基础上提供的数值计算算法

scipy.cluster Vector quantization / Kmeans

scipy.constants Physical and mathematical constants

scipy.fftpack Fourier transform

scipy.integrate Integration routines

scipy.interpolate Interpolation

scipy.io Data input and output

scipy.linalg Linear algebra routines

scipy.ndimage n-dimensional image package

scipy.odr Orthogonal distance regression

scipy.optimize Optimization

scipy.signal Signal processing

scipy.sparse Sparse matrices

scipy.spatial Spatial data structures and algorithms

scipy.special Any special mathematical functions

scipy.stats Statistics

SciPy 模块举例 `scipy.integrate`

- 数值计算积分

$$\int_{-\pi}^{\pi}(1+\cos\theta)\mathrm{d}\theta$$

课堂练习：积分

- 几种方法举例
  1. 手动矩形法
  2. `scipy` 调库法
  3. 蒙特卡罗法

SciPy 模块举例 `scipy.optimize`

第九章　GNU 命令行

9.1　复习与提示

软件准备

GNU 环境 课程与作业环境

sed 文本流编辑器

gawk GNU awk 语言，数据驱动程序语言

- Alfred Aho, Peter Weinberger, and Brian Kernighan

coreutils GNU 核心命令

man-db 帮助文档阅读器

info 层级帮助文档阅读器

curl 网络请求和下载器

bc 计算器

file 查看文件类型

bash-doc bash 详细教程

```
apt update
apt install gawk file sed man-db info wget curl bc bash-doc neofetch
```

复习

Git

- 队友分工合作，过去与未来的合作

Python

- 程序结构，数据类型，数组张量，画图

技能学习的特点

- 归纳地学习黑客技术
- 先“不求甚解”再择最重要的“寻根究底”，在两种学习模式间自由跳跃。
  - 在麦克斯韦方程出现之前，已经有光学
  - 不是只有做完实验才能吃饭

当我们使用 Python 的 REPL 环境时，它本身也是一个程序，由我们从外壳（shell）执行了。这一部分的目标是从 Python 出发，“向外”探索，分析 Python 所运行的环境，深入操纵计算机系统。

学习的风格是是归纳地学习黑客技术。“黑客技术”带有强烈的文化色彩：当黑客使用一件工作或见到一个物品，他都非常有兴趣与激情去了解它本质的内在机制。在了解它之后去改造它，让它更好地服务人的生活。黑客精神和物理学相通。物理学家理解世界后试着改造世界，让生活变得更美好。

在前两部分的 Python 内容中，这种精神已经有了体现。同学们在配置环境时，遇到了 bug 要可复现地描述清晰，探索解决办法，解决问题后如何分享。大家在调试程序时，就在面对一个工具，你只有彻底理解它，才可能调试好，让工具按照人的意愿工作。我们已经践行了黑客精神。

技术在科学领域之外。技术与科学最重要的不同，在于技术要面对纷繁复杂的现实世界，复杂到人的脑力无法彻底理解。如果再让人类重新构建一遍电气社会和计算机系统，那很可能因为人和环境不相同而构建出来完全不一样的体系。也许它们在抽象层面本质一样，但是形式上却很不同。技术是历史路径和人类经验依赖的，它没有唯一解。

当然经验的总结也能经过高度概括发展成科学。科学可以源于技术的积累，只是科学可被表述成非历史路径依赖的。以经典力学为例，它经过了几轮革命性的重塑，当下我们对经典力学的描述已经有一个普适的方式，至于历史上力学如何描述已经不是那么重要了。它已经成为科学，可以从简单的假设为内核出发演绎，这个内核简洁明了自洽。

对物理专业的学生而言，前两年的课程都是前人科学思想的浓缩总结，容易形成寻根究底的学习习惯。大家学习了麦克斯韦方程，但是没有学习麦克斯韦方程出现之前的手磨透镜的光学技巧。即使科学比技术更本质，但不能因为科学存在而轻视技术。如果不掌握“形式”的技术，就难以迈出学习的第一步。

在下面的技术学习中，你尽可能跟着我一起把的例子都自己运行一遍，在理解机理前，你也会发现它有确定的运行条件和运行结果。当你再次使用它，经验慢慢聚集到一起，逐渐形成你的技能树。你从收集经验的碎片开始，经过不断练习在某一天发现它们连接起来形成了技能树，而不是从一个树根直接生长到树叶。积累碎片的过程是混乱痛苦的，我尽可能的梳理出工具的逻辑，有条理地给同学们呈现出来。但它只是辅助，学习的“寻根究底”行为要有一个边界，大家可以探讨技术后面的“本质”。但同时，当你发现不理解某物，却可以运行的时候，请不要有心理负担坦然使用它。

Python 程序的支撑

- 程序由 Python 书写，或者 R、C++ 等等任何最佳工具
- 透明数据的格式和可复现的结果图。

如何把它们组织起来?

- 如何管理 GB、TB 乃至 PB 级的输入输出数据?
- 如何高效地在不同组输入数据上重复运行亿万次?
- 四大基本原则在此依然适用。

掌握了 Python 语言后，可以写数据处理和物理规律推断工具了。在综合运用和组织工具，完成一件复杂的任务如大作业的过程中，如果每个子任务都写一个程序解决，很快就会好多个程序。这些程序之间如何协调？下面我们学习协调 Python 脚本运行的最佳工具，而不是用 Python 脚本管理 Python。

如图中的例子，在数据处理流程中，程序 1 和程序 2 由 Python 写成。这个流程在实验中要被执行千万遍，输入为几百万个 CSV 文件。为了高效地执行 Python 程序，让步骤一个不落地完成，我们要学习新的工具。虽然我们在这一部分跳出 Python，出发点已变，但是已经有的四大基本原则依旧适用：在任何时候都要保证工作可复现，中间结果透明，不管数据处理流程多么复杂都要确保“一次”，每个工具都有特定的适用范围。

9.2 命令行

GNU 命令行界面在配置环境时大家已有经验。每个人在刚刚接触命令行界面都有不同的感受，或者觉得很酷，或迷茫，或无助，抑或觉得复古。这种感觉总结起来，是“文化冲击”。好像出国留学，来到陌生国度。你发现需要换纸币电子支付不普及，你发现城市里没有出租车，你发现大家直接用手抓着饭吃，你发现竟然人类社会的邻里关系和同事关系可以如此不同。在异国他乡会有诸多不适，开始我们不断问“怎么是这样?”，慢慢开始思考“这是为什么?”。当我们进入命令行，配置环境时，都要进入一个纯黑的界面—虽然不只是配色，“纯黑”是个比喻—大家所感受到的不适，正是文化冲击。当今世界，资本越来越希望用表面的、绚烂的东西抓住我们的注意力，从而借投放广告等牟利，对我们的思想产生影响，例如无限接续的短视频。纯黑的界面恰恰相反，它很拽，你不跟他说，他也不理你。你让它干什么它就什么，你没让它干它就什么也不干，从来不会主动来找你推送。这是巨大的文化差异。

你马上会进入魔法世界，慢慢从麻瓜变成魔法师。命令界面更本质地贴近于计算机系统，2024 年的人类社会，几乎所有日常生活都已被计算机支配，除非是在冥想，每时每刻都无法离开计算机了。，当然啊，当然我们我们口袋里的这些东西这也都是。计算机程序本质上都是 0 和 1，即一串字符。如果掌握字符的与计算机交互的界面——一切离散对象都能映射到自然数，0 和 1 也能表达成自然数，一切的字符通信本质上还是 0 和 1 —就在计算机支配的世界里，掌握了计算机的基本通信方式，直接和计算机对话，就等于我们从麻瓜变成了魔法师。

好，那么我我们回过头来说，我们为什么这个课程里边我们要学习这个呢？那么当然是要做。做这个实验物理啊，实验物理的这个数据处理，尤其是大规模的数据处理。首先呢，我们要保证啊，这个通过命令行来保证这个负限性，也就是我们写出了一个 python 程序，大家已经很好的贯彻了我们四个原则，这个 python 程序写的非常优美。呃，写的写的非常的非常具有可复现性，然后非常好。

GNU 命令行界面：神秘的黑色魔法世界

- 我们已经接触的命令：`ls`, `cd`, `rm`, `git`, `less`, `man`, `cat`, `pwd`

为什么学习命令行

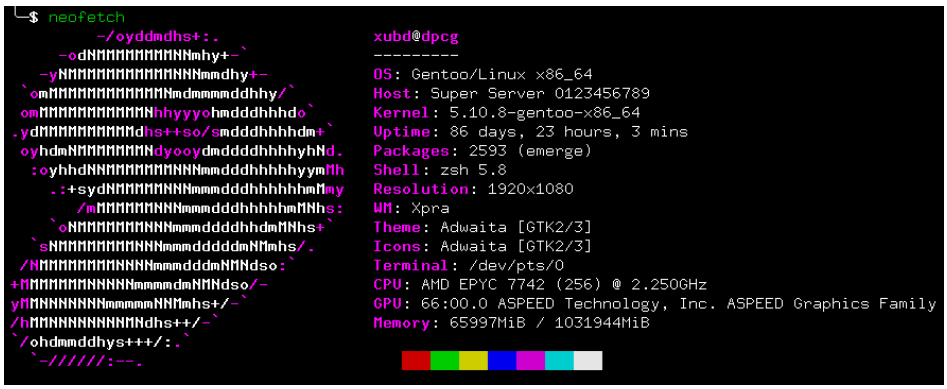
- 复现 原则的保证
  - Python 程序，有明确的功能，定义了输入和输出。
  - 需要高效且可复现地整合协调各类程序，处理数据。
- 体现 透明 原则
  - 数据的中间结果，可以用命令行工具查看 `cat`, `less`, `h5dump` …
  - 注：对应的图形工具也存在，可互相搭配。
- 体现 最佳工具 原则
  - GNU 命令行目前是文件和数据处理流程管理的最佳工具
  - 人操纵计算机最高效的方式：快捷键

参考书

Jeroen Janssens, Data Science at the Command Line

两周之后，你可能会成为：

ゆき Installs Gentoo



数字时代：如何更本质地与世界相连?

- 我们购买设备，使用设备。
  - 如果可以理解设备，改造设备、修理设备和创造设备，生活会有什么变化?
- 程序和命令行是掌控自己数字世界的重要一步
  - 如果你不掌控自己的世界，别人就会代替你成为主人
  - 菜单、窗口是他们设计好给我们用的，程序、命令是自己组合给自己用的。

手机例子

- Gentoo on Android

命令行迫使我们思考

对任何人下达明确的指令，都是困难的。对机器也是。除非通过思考，我们无法让指令明确
而命令行会非常诚实，任何不明确的指令都会被惩罚（报错）

9.3 命令行的特点

命令行的易用性

- 命令行是天然的 REPL
- 命令行有最丰富的文件操作工具
- 介于程序设计语言与图形界面之间
- 是计算机系统的“母语”，用户与开发者的重叠更大
  - 科学计算中，不是任何时候都有现成的工具可用，经常需要自己开发
    - 对于科学探索来说，地表最强的产品经理也没用用
  - 依赖命令行可以缩小开发者与用户的鸿沟

命令行的整合性和扩展性

- 所有的语言工具都可以从命令行调用
  - 新的语言写成的命令行工具不断涌现，许多工具非常适合用于数据处理
- 命令行是天然的“胶水”，整合各类语言协同工作
  - 鼓励小巧精悍工具的组合，避免又长又难理解的马拉松程序出现
  - 不论流行的语言 Perl、Python、Scala 如何变化，命令环境基本不变

命令行的自动性

- 使用的命令可以轻易组成脚本，重复使用
  - 图形界面不易录制操作
- 命令的传递方式是字符，字符可以被清晰地表述处理，因此可充分自动。

命令行的普适性

- GNU/Linux, 各种 Unix，macOS，Microsoft Windows WSL.
- 世界顶级的超级计算机，云计算主机，智能手机，物联网设备都可用同一套命令行工具操作
- 已有约 50 年历史，一定会继续存在
  - 字符的编码标准起源于电报机时代，一个多世纪以来不变。
  - 当今的计算机“终端”“控制台”“命令行”“外壳”（名字太多，统一不了）仍在模拟一个电报机（teleport）。

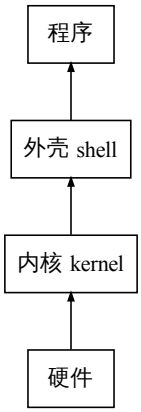
9.4 外壳

外壳 shell

- 外壳 shell 是相对于操作系统内核 kernel 而言，与人类交互的部分
- 有命令行 (Command Line Interface, CLI) 和图形 (Graphical User Interface, GUI) 两种
- Python 在 shell 之上运行

实际上 Python 也可以成为外壳环境

- Python 与 `bash` 平等，都是“用户态”程序 userspace
- `bash` GNU Bourne-Again SHell, 目前使用最为广泛，方便进行文件操作和调用命令
  - `man bash` 可阅读在线文档
  - `info bash` 获得详细教程



命令的类型

- 可执行程序，例如 `ls`

```
type ls
```

```
ls is hashed (/bin/ls)
```

- 脚本，例如 Python 程序

```
type vitables
file $(realpath /usr/bin/vitables)
```

```
vitables is /usr/bin/vitables
/usr/bin/vitables: Python script, ASCII text executable
```

- Shell 内建命令，例如 `cd`

```
type cd
type type
```

```
cd is a shell builtin
type is a shell builtin
```

命令的类型（二）

- Shell 函数

```
function hello { :; }
type hello
```

```
hello is a function
hello ()
{
    :
}
```

- 别名 alias

```
alias vt='vitables -m r'
type vt
```

```
vt is aliased to `vitables -m r'
```

9.5 标准输入输出

概念

- 标准输入 standard input `stdin` (file descriptor 0)
- 标准输出 standard input `stdout` (fd 1)
- 标准报错 standard error `stderr` (fd 2)

管道

- 把前一个程序的标准输出和后一个标准输入连接起来
- 无限串联，每个命令各司其职

复习 Python 的标准输入和输出

- `input()` 标准输入
- `print()` 标准输出
- `print(..., file=sys.stderr)` 标准报错

输出

```
echo "I am using the command line!"
```

```
I am using the command line!
```

管道注入 `wc`

```
echo "I am using the command line!" | wc -c
```

```
29
```

管道用于筛选

```
seq 5
```

```
1
2
3
4
5
```

```
seq 30 | grep 7
```

```
7
17
27
```

```
seq 100 | grep 7 | wc -c
```

```
56
```

保存管道的中间结果

- tee

```
seq 100 | grep 7 | tee number-seven.txt | wc -c
```

```
56
```

```
cat number-seven.txt
```

```
7
17
27
37
47
57
67
70
71
72
73
74
75
76
77
78
79
87
97
```

重定向

- 标准输入和输出都可以被重定向到文件

```
seq 100 > s100
cat s100 | head
```

```
1
2
3
4
5
6
7
8
9
10
```

```
wc -l < s100
```

```
100
```

自由管道 FIFO named pipe

```
mkfifo filter.pipe
seq 100 > filter.pipe &
grep 7 < filter.pipe
```

```
[1] 42881
7
17
27
37
47
57
67
70
71
72
73
74
75
76
77
78
79
87
97
[1]+  Done                    seq 100 > filter.pipe
```

任务控制

- bash 环境中 & 代表置于后台运行。

```
sleep 1m &
sleep 2m &
jobs
```

```
[1] 43068
[2] 43069
[1]-  Running                 sleep 1m &
[2]+  Running                 sleep 2m &
```

- `fg` 将任务拉至前台
- `bg` 将任务转至后台
- `kill` 将任务终结
- `^z` （Ctrl-z），挂起前台任务
- `^c` 终结前台任务

计算器的四种写法

- `awk` 派，使用 `awk` 语言

```
cat s100 | awk '{total+=$NF} END {print total}'
```

```
5050
```

- `Python` 派？Python 无法写成一行
- `paste` 派

```
cat s100 | paste -s -d +
cat s100 | paste -s -d + | bc
```

```
1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17+18+19+20+21+22+23+24+25+26+27+28+29+30+31+32+33+34+35+36+37+38+39+40+41+42+43+44+45+46+47+48+49+50+51+52+53+
5050
```

帮助

- manual page `manpage`

```
man cat
man paste
```

  - Emacs `M-x woman`

- info page

```
info cat
info paste
```

  - Emacs `C-h i`

- 命令 `--help -h`

帮助（二）

```
uname --help
```

```
Usage: uname [OPTION]...
Print certain system information.  With no OPTION, same as -s.

  -a, --all                print all information, in the following order,
                             except omit -p and -i if unknown:
  -s, --kernel-name        print the kernel name
  -n, --nodename           print the network node hostname
  -r, --kernel-release     print the kernel release
  -v, --kernel-version     print the kernel version
  -m, --machine            print the machine hardware name
  -p, --processor          print the processor type (non-portable)
  -i, --hardware-platform  print the hardware platform (non-portable)
  -o, --operating-system   print the operating system
      --help     display this help and exit
      --version  output version information and exit

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation <https://www.gnu.org/software/coreutils/uname>
or available locally via: info '(coreutils) uname invocation'
```

9.6 命令巡礼

本机信息

```
hostname
uname -a
```

```
dpcg
Linux dpcg 5.10.8-gentoo-x86_64 #1 SMP Sun Aug 15 14:46:49 CST 2021 x86_64 AMD EPYC 7742 64-Core Processor AuthenticAMD GNU/Linux
```

```
id
date
```

```
uid=1003(xubd) gid=1003(xubd) groups=1003(xubd),10(wheel),103(vboxusers),250(portage),999(xpra),1001(astrotools)
Mon Jul 25 10:53:06 AM CST 2022
```

本机信息

```
uptime
dmesg | tail
```

```
 10:53:35 up 86 days, 23:32,  7 users,  load average: 1.05, 0.59, 0.36
[6943834.870848] nfs: server jpd not responding, timed out
[6943847.158500] nfs: server jpd not responding, timed out
[6943852.790312] nfs: server jpd not responding, timed out
[6943871.733748] nfs: server jpd not responding, timed out
[6943888.645236] nfs: server jpd not responding, timed out
[6943894.261019] nfs: server jpd not responding, timed out
[6943900.404864] nfs: server jpd not responding, timed out
[6943906.420662] nfs: server jpd not responding, timed out
[7345744.319918] xpra[197935]: segfault at 1 ip 00007f687e955f89 sp 00007fff68363050 error 4 in libgdk-3.so.0.2404.26[7f687e901000+5d000]
[7345744.319934] Code: 00 00 48 89 44 24 08 31 c0 e8 53 be fc ff 48 85 c0 74 1d 0f b6 50 4c 66 81 e2 fd 00 66 83 fa 01 75 0e 48 8b 50 18 48 8b 42 28 <f6> 40
```

日常

- 创建空文件

```
touch nf
ls -l nf
```

```
-rw-r--r-- 1 xubd xubd 0 Jul 25 11:01 nf
```

- 输出信息

```
echo "Hi!"
```

```
Hi!
```

cut 选择列

```
sudo dmesg | cut -c-15 | tail
```

```
[370615.920192]
[370616.151878]
[370616.389111]
[370619.345380]
[370619.345382]
[370619.365730]
[370619.365751]
[476022.060361]
[476109.862666]
[476204.332049]
```

```
ls -1 */* | cut -d/ -f 1 | uniq
```

```
__pycache__
_minted-e1
_minted-e2
_minted-e3
_minted-e4
_minted-e5
_minted-e6
_minted-e7
_minted-e8
_minted-e9
```

```
_minted-ea
_minted-eb
_minted-ec
_minted-ed
_minted-ee
_minted-ef
_minted-p2-ein
_minted-p2
dataframe-practice
ein-images
fig
img
lecture
lm-examples
notebooks
out
upload
```

文件信息

- 寻找 dev/waveform-compress 路径之下所有 R 程序文件

```
find dev/waveform-compress -name "*.R"
```

```
dev/waveform-compress/compress_comp.R
dev/waveform-compress/compress_core.R
dev/waveform-compress/compress_one.R
dev/waveform-compress/getH.R
dev/waveform-compress/wdlib.R
```

- 识别文件类型

```
file dev/waveform-compress/compress_one.R
```

```
dev/waveform-compress/compress_one.R: a /usr/bin/env Rscript script, ASCII text executable
```

9.7 通配符

定义

- 通配符可以按规则匹配，用于构造简单匹配模式
  - * 匹配任意多个任意字符
  - ? 匹配一个任意字符

```
cd physics_data
ls *.html
```

```
Command-Line.slides.html    Python-Basics.slides.html
Data-Formats.slides.html    Python-Constructs.slides.html
Data-Frame-RPy.slides.html  Python-Functions.slides.html
Data-Frame.slides.html     Python-Modules.slides.html
Pandas-RDB.html     Python-SQLite.slides.html
Pandas-RDB.slides.html     Shell-Scripts.slides.html
```

```
echo *.html
```

```
Command-Line.slides.html Data-Formats.slides.html Data-Frame-RPy.slides.html Data-Frame.slides.html Pandas-RDB.html Pandas-
RDB.slides.html Python-Basics.slides.html Python-Constructs.slides.html Python-Functions.slides.html Python-Modules.slides.html Python-
SQLite.slides.html Shell-Scripts.slides.html
```

例子

```
ls Python-*.html
```

```
Python-Basics.slides.html      Python-Modules.slides.html
Python-Constructs.slides.html  Python-SQLite.slides.html
Python-Functions.slides.html
```

```
ls *.?pynb
```

```
Command-Line.ipynb     Pandas-RDB.ipynb
Data-Formats.ipynb     Python-Basics.ipynb
DataFrame.ipynb     Python-Constructs_Functions.ipynb
Data-Frame-RPy.ipynb     Python-Functions.ipynb
GLM.ipynb     Python-IO_Survival.ipynb
GLM-Python.ipynb     Python-SQLite.ipynb
LIGO-intro.ipynb     Relational-Algebra.ipynb
LOSC_Event_tutorial.ipynb  Relational-SQL.ipynb
Make-Pipeline.ipynb     Shell-Practice.ipynb
Merge-GroupBy.ipynb     Shell-Scripts.ipynb
NumPy.ipynb     Visualization.ipynb
NumPy-SciPy.ipynb
```

9.8 bash 的运行模式

交互模式

- ~~REPL~~

脚本模式

- 把命令集合起来，逐条执行
- 一边文件名以 `.sh` 结尾

```
echo "cat s100 | paste -s -d + | bc" > add-up-100.sh
bash add-up-100.sh
```

```
5050
```

让脚本可执行

- 在第一行填加脚本解释器 `#!/bin/bash` ，告知内核本脚本要由 `/bin/bash` 解释执行。“#!”也称为“shebang”，意思是“sharp”（#）和“bang”（!）。

```
sed '1i#!/bin/bash' -i add-up-100.sh
cat add-up-100.sh
```

```
#!/bin/bash
cat s100 | paste -s -d + | bc
```

- 赋予可执行权限

```
chmod +x add-up-100.sh
./add-up-100.sh
```

```
5050
```

脚本程序

- 世界上本没有脚本，打的命令多了，聚在一起，就形成了脚本
- 脚本可以当作命令使用

例子：=bash= 脚本

```
cat arguments.sh
```

```
#!/bin/sh

echo The first argument is $1
echo The second argument is $2
echo all the arguments are $@
```

```
chmod +x arguments.sh
./arguments.sh 1 输入 2 参数
```

```
The first argument is 1输入
The second argument is 2参数
all the arguments are 1输入 2参数
```

Shebang 和脚本命令

- `#!` 是一对有特殊含义的字符，它处在第一行开头在被执行时，内核会寻找其后的解释器来运行其后的脚本。
- `chmod` 改变文件的权限，分别为 r 读 w 写 x 执行。
- `chmod +x arguments.sh` 给 `arguments.sh` 可执行的权限。
- 执行时，系统使用 `#!/bin/sh` 指定的 `/bin/sh` 执行程序。
  - `./arguments.sh` 等价于 `/bin/sh arguments.sh` 。
  - 注意 `/bin/sh` 很可能不是 `bash` 。需要 `bash` 时，直接使用 `/bin/bash`

```
realpath /bin/sh # dash 是不同的 shell，功能少速度快
```

```
/usr/bin/dash
```

- 如果可执行的脚本在 PATH 指定的路径里，就可以被 shell 找到，例如 egrep 。

```
file $(which egrep)
```

```
/bin/egrep: POSIX shell script, ASCII text executable
```

脚本的参数

- 脚本调用时的参数，在脚本中使用 ‘$1‘ ‘$2‘
  - 相当于 Python 的 ‘sys.argv[1]‘, ‘sys.argv[2]‘
- 所有的参数是 ‘$@‘

```
cat /bin/egrep
```

```
#!/bin/sh
exec /bin/grep -E "$@"
```

- bash 脚本例子

```
file /usr/bin/* | grep "Bourne-Again shell" | head
```

```
/usr/bin/AddOrReplaceReadGroups:                Bourne-Again shell script, ASCII text executable
/usr/bin/ant:                                   Bourne-Again shell script, ASCII text executable
/usr/bin/antlr:                                 Bourne-Again shell script, ASCII text executable
/usr/bin/antlr3:                                Bourne-Again shell script, ASCII text executable
/usr/bin/any2djvu:                              Bourne-Again shell script, ASCII text executable
/usr/bin/apache-rat:                            Bourne-Again shell script, ASCII text executable
/usr/bin/BamIndexStats:                         Bourne-Again shell script, ASCII text executable
/usr/bin/BamToBfq:                              Bourne-Again shell script, ASCII text executable
/usr/bin/beeline:                               Bourne-Again shell script, ASCII text executable
/usr/bin/binutils-config:                       Bourne-Again shell script, ASCII text executable
grep: write error: Broken pipe
```

9.9 bash 变量

bash 变量

- 在 bash 中，一切皆字符串
- 变量需要额外的 $ 来指定， $a 和 ${a} 代表变量 a 的值
  - 类比 make ，使用 $(a) 代表变量取值
- 赋值号“=”的前后都不能有空格，与 Python 的格式建议相反！

```
a=1
echo $a
b=我是谁
echo ${a}${b}
```

```
1
1我是谁
```

```
echo "${a}${b}，我从哪里来" # 双引号中变量是被替换的
```

```
1我是谁，我从哪里来
```

变量的引用

- 无引号时，空格起作用
- 单引号字符串中的字符保持不变
- 双引号时，字符串中的变量被替换

```
echo '${a}${b}，我从哪里来' # 单引号保持不变
```

```
${a}${b}，我从哪里来
```

引用命令的执行结果

- 把命令运行的标准输出赋给变量
- 与 make 类似，换行分隔的列表被转成由空格分隔

```
n_list=$(seq 9)
echo ${n_list}
```

```
1 2 3 4 5 6 7 8 9
```

```
n233=$(seq 10000 | egrep ^23+$)
echo ${n233}
echo ${#n233} # 字符个数
```

```
23 233 2333
11
```

bash 数组

- 虽然取名为数组 array ，但与 Python 的 列表更接近。
  - 可以存储任意元素。

```
busket=()
busket+=( 香蕉 弥猴桃 牛油果 )
busket+=( 西瓜 荔枝 )
echo ${busket} # 按普通变量取值，只有第一个元素
echo ${busket[@]}
echo ${busket[3]}
```

```
香蕉
香蕉 弥猴桃 牛油果 西瓜 荔枝
西瓜
```

```
echo ${#busket[@]} # 元素个数
```

```
5
```

bash 字典

- 又称关联数组 associative array ， 类比 Python 字典

```
declare -A cargo # 声明 cargo 是字典
cargo[banana]=3
cargo[apple]=4
echo ${cargo[banana]}
echo ${!cargo[@]} # 取词
echo ${cargo[@]} # 取值
```

```
3
apple banana
4 3
```

bash 算术

- bash 一等公民是字符串，算术运算用于辅助和便利
  - 比用管道 和 $(...) 简洁

```
echo $((1+3)) $((3*5))
echo $(echo 1+3 | bc) $(echo 3*5 | bc)
```

```
4 15
4 15
```

- 但是计算能力有限，只能做简单运算

```
echo $((2**100))
echo $(echo "2^100" | bc)
```

```
0
1267650600228229401496703205376
```

9.10 bash 程序结构

选择结构

```
if [ 3 -gt 2 ]; then
    echo "3>2"
else
    echo "3<=2"
fi
```

```
3>2
```

判断语句

- [...] 是一个程序，正式名字是 test
- 真假判断来自该程序执行的返回值。
  - 0 为真：执行成功
  - 非 0 为假：执行失败
  - 注意与 Python 正好相反！

文档

- man [、 info test

真假判断

- $? 或 ${?} 变量值是前一条命令的返回值

```
[ 1 = 2 ] # 假，返回非 0
echo $?
[ 2 != 3 ] # 真，返回 0
echo $?
```

```
1
0
```

- [INTEGER1 -eq INTEGER2] 相等
- [INTEGER1 -ge INTEGER2] 大于等于
- [INTEGER1 -lt INTEGER2] 小于
- [INTEGER1 -ne INTEGER2] 不等
- [! EXPRESSION] 取否
- [EXPRESSION1 -a EXPRESSION2] 取和
- [EXPRESSION1 -o EXPRESSION2] 取或

bash 内建真假判断

- bash 提供了 [[...]] 的内建命令，兼容 [...] 语法，有更多功能
- 内建命令，调用速度更快
- 命令层次的逻辑运算
  - command1 && command2 取和
  - command1 || command2 取或
  - ! command 取否
- 逻辑运算替代选择结构，以下两者等价

```
if [ 3 -gt 2 ]; then
    echo "3>2"
else
    echo "3<=2"
fi
# 逻辑运算替代选择结构，可读性更强
[[ 3 -gt 2 ]] && echo "3>2" || echo "3<=2"
```

```
3>2
3>2
```

循环结构

- for 循环，计算 Fabonacci 数
- 从 seq 10 结果中，取得 10 个元素，用变量 i 遍历
  - 类比： Python for 循环，迭代器

```
x=1
y=1
for i in $(seq 10); do
    s=$(($x+$y))
    x=${y}
    y=${s}
done

echo ${y}
```

```
144
```

while 循环

- 与 seq 1000 | egrep ^23+$ 等价

```
seq 1000 | while read line
do
    echo $line | egrep -q '^23+$' && echo $line
done
```

```
23
233
```

until 循环

- 与 while 循环类似，条件相反

```
seq 1000 | until ! read line
do
    echo $line | egrep -q '^23+$' && echo $line
done
```

循环控制命令

- break 终止循环、 continue 跳入下一轮循环

```
seq 1000 | until ! read line
do
    if echo $line | egrep -q '^23+$'; then
        echo $line
        break
    fi
done
```

9.11 bash 函数

函数定义

- bash 函数是五种命令之一

```
function greet() {
    echo "Hello, $1"
}

greet 大佬
```

```
Hello, 大佬
```

函数递归

- 递归计算 Fabonacci 数
- `$(command)` 从 `command` 的标准输出取得结果
- `return` 给返回值，只能用 `$?` 捕捉或者用于逻辑运算
  - 与 Python 函数完全不同，因为命令成功与失败的判断更重要

```
function fib() {
    if [ $1 -le 2 ]; then
        echo 1
    else
        echo $(($(fib $(($1-1))) + $(fib $(($1-2)))))
    fi
}

fib 11
```

```
89
```

bash 的代码块与子进程

- `{...}` 代表一个代码块，把多行代码汇总成一个单元。
- `(...)` 使用复刻 (fork) 一个子 `bash` 运行代码。
  - 子 `bash` 中的变量设置不影响外层 `bash` 。

    ```
    s=1
    ( s=2; )
    echo $s
    ```

    ```
    1
    ```

  - 类比： `$(seq 100)` 代表复刻后运行 `seq` 提取标准输出。

9.12 bash 调试

bash 调试

交互法 在 `bash` 环境试验获得想要的语句

屏幕输出法 用 `echo ... >&2` 输出到标准错误 `stderr`

额外调试信息输出法 `set -xv` 开启调试输出

- `set +xv` 关闭调试输出
- `bash -xv script.sh` 以调试模式执行脚本

bash(1)

- `-v` Print shell input lines as they are read.
- `-x` Print commands and their arguments as they are executed.

9.13 总结

使用 `bash` 的原因

- `bash` 的列表、字典，形似 Python，计算能力不如 `bc` ，为什么值得使用?
- 最佳工具：使用 `bash` 处理文件时，有时需要列表、字典和算术
  - 但不需要太多。需要更专业的操作时，意味着更高级的工具胜任
  - “信手拈来”的境界是创新的温床，重大的全新突破都在有了足够积累后不经意间出现
    - 不经意的前提是：放松、随意
  - 不让大脑等待双手是最佳工具原则的最高等级
    - 人机交互，打字，扔掉鼠标
    - 避免不必要的环境切换

9.14 备用

9.15 娱乐命令

`sl`

```
sl
```

```
bash: sl: command not found
```

```
apt install sl
```

matrix 可操作动画

- 安装

```
apt install cmatrix
```

- 文档

```
man cmatrix
```

ASCII Art

```
apt install sysvbanner bsdgames
```

- worm 贪食蛇
- worms 模拟蚯蚓

```
banner physics
```

```
######   #     #  #     #   #####   ###   #####    #####
#     #  #     #   #   #   #     #   #   #     #  #     #
#     #  #     #    # #    #         #   #        #
######   #######     #     #####    #   #         #####
#        #     #     #          #   #   #              #
#        #     #     #    #     #   #   #     #  #     #
#        #     #     #     #####   ###   #####    #####
```

ASCII Art: cowsay

```
apt install cowsay-off
```

```
cowsay -e xx -T W "parity?"
```

```
 ---------
< parity? >
 ---------
        \   ^__^
         \  (xx)\_______
            (__)\       )\/\
             W ||----w |
                ||     ||
```

cowsay II

```
cowsay -f meow "parity?"
```

```
 ---------
< parity? >
 ---------
  \
   \ ,   _ ___.--'''`--''//-,-_--_.
      \`"' ` || \\ \ \\/ / // / ,-\\`,_
     /'`  \ \ || Y  | \|/ / // / - |__ `-,
    /@"\  ` \ `\ |  | ||/ // | \/  \  `-._`-,_.,
   /  _.-. `.-\,___/\ _/|_/_\_\/|_/ |     `-._._)
   `-'``/  /  |  // \__/\__  /  \__/ \
        `-'  /-\/  | -|   \__ \   |-' |
          __/\ / _/ \/ __,-'   ) ,' _|'
         (((__/(((_.' ((___..-'((__,'
```

- 其它动物

```
ls /usr/share/cowsay/cows
```

管道练习

```
apt install fortune-zh lolcat
```

```
fortune | cowsay | lolcat -a
```

```
 ------------------------------------------------------------------
/ 谷神不死，是谓玄牝。 玄牝之门，是谓天地根。 绵绵若存，用之不勤。  \
\ --《道德经》                                                     /
 ------------------------------------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

ASCII Art Anime

```
apt install bb
bb
```

ASCII Art color Anime

```
apt install caca-utils mpv
cacafire
```

```
mpv "http://hep.tsinghua.edu.cn/~orv/teaching/Yuki_Installs_Gentoo.mp4" -vo caca
```

再来一打

- 小玩具类

```
apt-cache depends games-toys
```

- 所有类

```
apt-cache depends games-all
```

- 欢迎大家以 merge request 的形式补充更多的游戏
  - ~~尤其是能在校园网内联网的!~~

第十章　GNU Make 数据生产线

10.1　复习与提示

软件准备：带 Scheme 语言扩展的 GNU Make

- Debian

```
# Debian
apt install make-guile make-doc guile-3.0-doc
```

命令行

- 命令行操作系统外壳 shell 的一种，介于内核 kernel 与用户程序之间。
- 最佳工具：调度用户程序、管理文件。
- 研发门槛低：输入命令的过程中，就随手写了新程序。
- `bash` GNU Bourne-Again SHell 提供最流行的强大命令行环境
  - 五类命令：二进制程序、可执行脚本、函数、内建命令、别名
  - 标准输入输出、管道
  - 脚本 `#!/usr/bin/env bash` 开头，加上若干命令
  - 变量 `$A`, `${A}`
  - 引用

转译	单引号	双引号
空格	x	x
$	x	o

10.2　任务流水线

学堂路车神

- 判断题：学神还是菜鸡?
- 因害怕关闭机盖程序中断

当事人

- “骑车带电脑是因为程序没跑完，害怕关闭机盖后导致程序中断，三四个小时重新来。”
- “我没有也从未试图在骑车的过程中做出敲击键盘，编辑程序，阅读程序这类高难度行为，只是托着电脑避免机盖关闭。”
- “至于为什么没想到其他方法保持程序运行，是因为我确实不知道，实在是很菜了……”

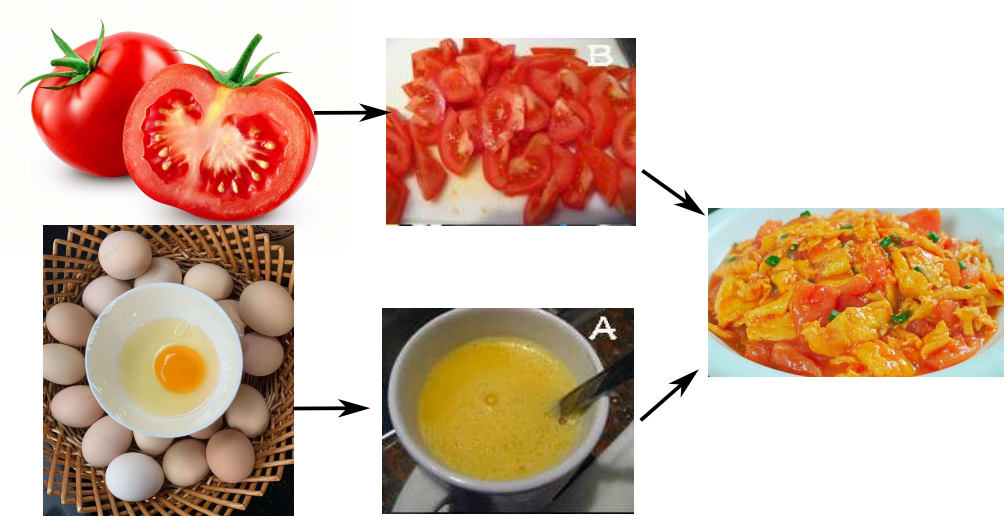
数据流水线的构造目标

复现原则的要求
要记录下来以什么样的顺序和参数运行什么命令，执行什么程序。

思路和要点

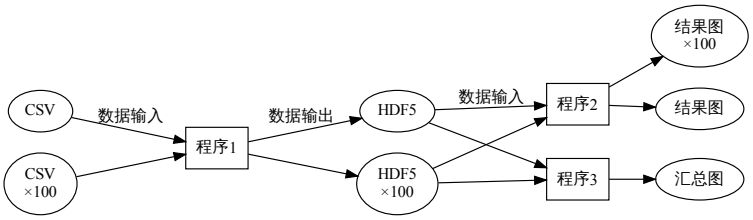
- 把流程系统化成输入、输出与过程三要素。
  - 而向数据编程，data-driven programming
- 系统表达输入数据、输出数据和中间结果的依赖关系，
  - 成为“可执行的说明文档”
- 高效执行，包括并行处理和整合超级计算机等。
- 错误恢复
  - 修正错误后，可以从最后一步正确的数据开始继续执行。

西红柿炒蛋如何做?



- 任务的执行有前后的依赖顺序。

批量处理海量数据



命令行探索之后，要将数据处理方法自动化

- 实验要调用很多命令和程序
  - 重复运行，控制变量：以不同条件多次测量，探索规律
- 处理很多数据，有很多中间结果，依赖关系复杂
- 程序有更新怎么办？数据有更新怎么办?

10.3 GNU Make

Make 是最佳工具

- `make` 工具已经有 40 多年的历史，最初用来管理 C 语言程序的编译。
  - 根据依赖关系决定命令执行顺序
- `GNU make` 是 `GNU` 运动中，对 `make` 进行的扩展，更适合管理数据
  - 与 `BSD make` 有区别，在 macOS 环境里请配置 GNU 环境，使用 `gmake` 。
  - 版本 $\geq$ 4.3 支持 多目标规则

```
make --version
```

```
GNU Make 4.3
Built for x86_64-pc-linux-gnu
Copyright (C) 1988-2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

作用

1. 实现 复现 要求
2. 管理程序运行，在超级计算机上运行

3. 从错误中恢复

当今的超级计算机，大多由分布式的集群构成。集群是多台独立的，可互相高速通信的计算机构成。每台计算机相对于集群，是一个“节点”。人类操作集群，在一个“登录节点”上交互实现。通过登录节点，把任务提交给调度器，调度器根据情况，如节点的计算饱和度和任务需求等，把任务分配给计算节点，达到协调把千万个节点协同运算的效果。调度器能接受多名用户的任务提交，从而实现计算资源的共享。

Make 要素

- 文档

```
info make
man make
```

- 基本语法单元：清晰写明输入数据，输出数据和计算方法

```
target: source
	program source target # 必须 TAB 起始，记录如何做
```

- target 和 source 都应对应于当前路径下的文件名。
  - 当 source 存在时，可以通过运行 program source target 生成文件 target

参考书

- John Graham Cumming, The GNU Make Book
- 陈皓，跟我一起写 Makefile

10.4 变量

变量取值

```
a:=1 # eager evaluation (= 是 lazy evaluation)
$(info $(a))
```

- 调用时使用 $( )， $(info TEXT) 输出 TEXT
- 一次 原则，避免重复
  - 特殊变量： $^ 输入数据、 $@ 输出数据。
  - 目标与源都以文件表示，引用 $^ $@ 可有效减少重复。

```
target: source
        program source target #如何做
        program $^ $@ #如何做
```

```
target: source1 source2 source3
        program $< --reference $(word 2,$^) \
                --location $(word 3,$^) -o $@
```

Make 的函数

- 函数调用 `$(func argument1,argument2,...)`
  - word 函数 `$(word N,TEXT)`

    ```
    $(word 2,make.c use.c high.c) # => use.c
    ```

  - patsubst 函数 `$(patsubst PATTERN,REPLACEMENT,TEXT)`

    ```
    $(patsubst %.c,%.f,make.c use.c high.c) # => make.f use.f high.f
    ```

  - 更多的函数数在 `info make` 指南中查询（Emacs 下为 `C-h i` ）

10.5 例子

生成含有 1 到 100 数字的文件

- 创建 `Makefile` 文件

```
numbers.csv:
        seq 100 > $@
```

- 到 `Makefile` 所在的文件夹中执行

```
make numbers.csv
```

```
seq 100 > numbers.csv
```

```
cat numbers.csv | wc -l
```

```
100
```

不做无谓重复

```
numbers.csv:
	seq 100 > $@
```

再执行时， make 会判断是否还有必要

```
make numbers.csv
```

```
make: 'numbers.csv' is up to date.
```

判断准则

1. numbers.csv 已经存在
2. numbers.csv 比它依赖的文件都新。此例中无依赖文件，存在即是最新。

加入求和规则

```
total: numbers.csv
	paste -s -d + < $^ | bc > $@
```

```
make total
```

```
paste -s -d + < numbers.csv | bc > total
```

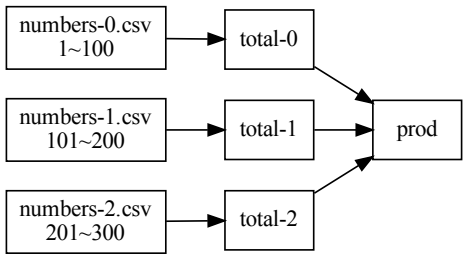
- 如果第一步 numbers.csv 不存在， make 会自动生成它

```
rm total numbers.csv
make total
```

```
seq 100 > numbers.csv
paste -s -d + < numbers.csv | bc > total
```

- 一次原则：只必要地写两处 numbers.csv 防止更新时造成不一致

进阶：暗含循环



```
numbers-0.csv:
	seq 100 > $@
numbers-1.csv:
	seq 101 200 > $@
numbers-2.csv:
	seq 201 300 > $@
total-0: numbers-0.csv
	paste -s -d + < $^ | bc > $@
total-1: numbers-1.csv
	paste -s -d + < $^ | bc > $@
total-2: numbers-2.csv
	paste -s -d + < $^ | bc > $@
prod: total-0 total-1 total-2
	paste -d '*' $^ | bc > $@
```

- 召唤一次原则!
- 循环程序结构是共通的。

10.6 通配符与替换

一般匹配关系 pattern

- 处理 *.h5 文件，生成对应文件名的 s/.h5/.png 图形。

```
filelist:=x.h5 y.h5 z.h5
.PHONY: all
all: $(filelist:.h5=.png)

%.png: %.h5
	./plot-celestial.py $^ -o $@
```

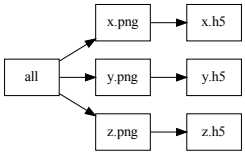
说明

- % 是 GNU Make 的通配符，用于替代任意字符串
- $(filelist:%.h5=%.png) 是 $(patsubst P,R,TEXT) 的简写。
  - 意为在 $(filelist) 中把符合 P (%.h5) 的部分替换成 R (%.png)
  - 有共同前缀时，可进一步简写为

    ```
    $(filelist:.h5=.png)
    ```
- 在执行规则中， %.png: %.h5 定义任意 png 都由相应的 h5 文件生成。

all 是一个总括目标

```
filelist:=x.h5 y.h5 z.h5
.PHONY: all
all: $(filelist:%.h5=%.png)
```



特殊目标

- .PHONY 代表无对应文件的目标
  - 否则如果存在一个名为 all 那么 make all 就不会执行
  - GNU Make 默认执行第一个目标，可以把它定义为无对应文件的 all
- .DELETE_ON_ERROR 如果出错就把坏文件删掉
- .SECONDARY 保留中间结果

```
.DELETE_ON_ERROR:
.SECONDARY:
```

新工具：guile

- make-guile 在 GNU make 的基础上嵌入了 GNU guile [gʌɪl] 解释器支持 scheme 语言。
- scheme 语言是 LISP 语言的一支，是历史悠久的人工智能语言。
  - LISP 与 fortran 是最早的高级编程语言，都出现在 1960 年代。
- 算术练习

```
(+ 1 1)
```

```
2
```

```
(* 100 (+ 1 1))
```

```
200
```

- 手册 info guile

函数式编程：把目标描述清楚

- Makefile 是函数式编程 Functional Programming 的语言。
- 不再关注“执行什么操作”，而是关注“输入到输出的映射”。
  - 类比：Python 的 map() ， 迭代器

```
# 生成 numbers-N.csv，包含 N*100+1 到 (N+1)*100 的数字
numbers-%.csv:
	seq $(guile (+ 1 (* $* 100))) $(guile (* 100 (+ $* 1))) > $@
# 分别求和
total-%: numbers-%.csv
	paste -s -d + < $^ | bc > $@
# 求积
prod: total-0 total-1 total-2
	paste -d* $^ | bc > $@
```

- 循环从 `prod: total-0 total-1 total-2` 构造，由数据驱动

数据驱动暗含循环

```
make prod
```

```
seq 1 100 > numbers-0.csv
paste -s -d + < numbers-0.csv | bc > total-0
seq 101 200 > numbers-1.csv
paste -s -d + < numbers-1.csv | bc > total-1
seq 201 300 > numbers-2.csv
paste -s -d + < numbers-2.csv | bc > total-2
cat total-0 total-1 total-2 | paste -s -d '*' | bc > prod
rm numbers-0.csv numbers-2.csv numbers-1.csv
```

- 可以自动补上失去的文件，只执行需要的部分

```
rm total-2
make prod
```

```
seq 201 300 > numbers-2.csv
paste -s -d + < numbers-2.csv | bc > total-2
cat total-0 total-1 total-2 | paste -s -d '*' | bc > prod
rm numbers-2.csv
```

可以用 shell 命令替换 guile

```
# 调用 shell 命令方案
numbers-%.csv:
	seq $(shell echo "$* * 100 + 1" | bc ) $(shell echo "($* + 1) * 100" | bc ) > $@
```

但是不如 guile 简明

```
numbers-%.csv:
	seq $(guile (+ 1 (* $* 100))) $(guile (* 100 (+ $* 1))) > $@
```

10.7 Make 机理

调试

- Makefile 的调试器？ `remake` 可以试验。
- 一般调试使用 `$(info )` 查看中间变量和结果
- 不要把文件写得太长，太长时分成小文件，分成小单元调试
- 多写多 bug 少写少 bug，不写无 bug
- 推论：以最简洁的代码实现目的和四个原则
- 把不同功能分放在不同文件中

```
include names.mk
include process.mk

.PHONY: all
all: step1 step2
```

深入理解 make 的执行机理：两个阶段

```
file=numbers
$(file)-%.csv:
	seq $(shell echo "$* * 100 + 1" | bc ) $(guile (* 100 (+ $* 1))) > $@
```

替换阶段

把所有的 `$(...)` 都进行替换：变量换成它的值，函数执行后换成返回值。

当执行 ‘make numbers-1 numbers-2‘ 时，

```
file=numbers
numbers-1.csv: # %=1, $*=1, $@=numbers-1.csv
	seq 101 200 > numbers-1.csv
numbers-2.csv: # %=2, $*=2, $@=numbers-2.csv
	seq 201 300 > numbers-2.csv
```

深入理解 make 的执行机理：第二阶段

```
file=numbers
$(file)-%.csv:
	seq $(shell echo "$* * 100 + 1" | bc ) $(guile (* 100 (+ $* 1))) > $@
```

执行阶段

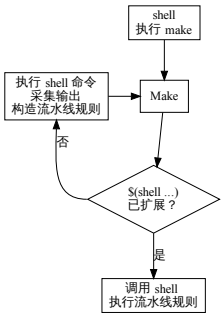
根据每一个要生成的目标， `numbers-1.csv numbers-2.csv` ，先生成它的依赖，再执行对应的命令

```
seq 101 200 > numbers-1.csv
seq 201 300 > numbers-2.csv
```

Make 与 shell 的调用关系

```
numbers-%.csv:
	seq $(shell echo "$* * 100 + 1" | bc ) $(shell echo "($* + 1) * 100" | bc ) > $@
```

- make 在 shell 中执行，make 通过 shell 执行流水线规则和取值



赋值

- “:=”是 eager evaluation，在被调用时，就把运算做完。
- “=”是 lazy evaluation，在被使用时，才做运算。
  - 不同时节运算，可能有不同变量环境，可到不同的效果

10.8 简化语句

继续深化落实一次原则

- Makefile 里可以调用 shell 命令，将其标准输出作为值。

```
sequence:=$(shell seq -w 00 99)
file_list:=$(shell find . -name "*.h5")
```

- 构造 `total-0 total-1 total-2`

```
seqs:=$(shell seq 0 2)
totals:=$(seqs:%=total-%)
# 另一种方法
totals:=$(addprefix total-,$(seqs))
```

内嵌 scheme

- 最佳工具原则： `$(shell )` 和 `$(guile )` 哪个简洁用哪个。 `iota` [ʌɪˈəʊtə]

```
seqs:=$(guile (iota 3))
```

再次尝试

- 直接来 10 个

```
seqs:=$(guile (iota 10))
totals:=$(seqs:%=total-%)

# 生成 numbers-N.csv, 包含 N*100+1 到 (N+1)*100 的数字
numbers-%.csv:
        seq $(guile (+ 1 (* $* 100))) $(guile (* 100 (+ $* 1))) > $@
# 分别求和
total-%: numbers-%.csv
        paste -s -d + < $^ | bc > $@
# 求积
prod: $(totals)
        paste -d '*' $^ | bc > $@
```

运行

```
make prod
```

```
seq 301 400 > numbers-3.csv
paste -s -d + < numbers-3.csv | bc > total-3
seq 401 500 > numbers-4.csv
paste -s -d + < numbers-4.csv | bc > total-4
seq 501 600 > numbers-5.csv
paste -s -d + < numbers-5.csv | bc > total-5
seq 601 700 > numbers-6.csv
paste -s -d + < numbers-6.csv | bc > total-6
seq 701 800 > numbers-7.csv
paste -s -d + < numbers-7.csv | bc > total-7
seq 801 900 > numbers-8.csv
paste -s -d + < numbers-8.csv | bc > total-8
seq 901 1000 > numbers-9.csv
paste -s -d + < numbers-9.csv | bc > total-9
cat total-0 total-1 total-2 total-3 total-4 total-5 total-6 total-7 total-8 total-9 | paste -s -d '*' | bc > prod
rm numbers-8.csv numbers-3.csv numbers-7.csv numbers-5.csv numbers-6.csv numbers-9.csv numbers-4.csv
```

10.9 Makefile 脚本

选择结构

```
a:=$(if CONDITION,THEN-PART[,ELSE-PART])
```

- 如果 CONDITION 成立，则取 THEN-PART。类比 Python

```
a = THEN-PART if CONDITION else ELSE-PART
```

- `$(or COND1,COND2)` `$(and COND1,COND2)` 用于逻辑运算

循环结构与自定义函数

- 数据本身可以构造一重循环，当循环多于一重时，使用 `$(foreach )`

```
# radius list
rl:=$(shell seq -w 0.10 0.01 1.00)
profile:=$(foreach d,x y z,$(rl:%=1t_+%_$(d).h5))
$(info $(profile))
```

```
1t_+0.10_x.h5 1t_+0.11_x.h5 1t_+0.12_x.h5 1t_+0.13_x.h5 ...
```

- Makefile 模版（起到“函数”复用的作用）由 `define … endef` 定义，使用 `$(eval )` 执行。

```
define video
$(1)/%.avi: $(1)/%.mp4
	ffmpeg -i $$^ $$@

endef

$(eval $(foreach d,up down transverse,$(call video,$(d))))
```

更简洁的笛卡尔积：bash

```
echo {w..z}{1,2,5,9}
```

```
w1 w2 w5 w9 x1 x2 x5 x9 y1 y2 y5 y9 z1 z2 z5 z9
```

lecture 仓库的 Makefile

- `https://git.tsinghua.edu.cn/physics-data/lecture/`
  - 仓库使用的 `Makefile` 描述如何生成课件和讲义。
  - 尝试阅读 `Makefile` 给出文件的依赖关系。

JUNO 真实世界例子

- 用于数百 TB 蒙特卡罗数据生成的例子

```
# 1=model, 2=imh, 3=dist, 4=iter
define SN-tpl

output+=data/det/$(1)/$(2)/$(3)/ith/$(4).root

data/$(1)/$(2)/$(3)/%.root:
	./genSN.sh $(1) $(2) $(3) $$@
```

```
data/det/%/$(4).root: data/%.root
        ./exeDet.sh $$^ $$@ $(4)
endef

$(eval $(foreach i,$(imod),$(foreach j,$(imh),\
        $(foreach k,$(dist),$(call SN-tpl,$(i),$(j),$(k),0))\
        $(foreach l,$(ip),$(call SN-tpl,$(i),$(j),10,$(l))))))

all: $(foreach i,$(ith),$(subst ith,$(i),$(output)))

# Delete partial files when the processes are killed.
.DELETE_ON_ERROR:
# Keep intermediate files around
.SECONDARY:
```

10.10 进阶使用

并行计算

- make -j
- 把 SHELL 换成超算任务调度器 → 超算并行计算

命令行指定变量

```
make -j SHELL=/bin/bash
```

联合目标

When 'make' builds any one of the grouped targets, it understands that all the other targets in the group are also created as a result of the invocation of the recipe. Furthermore, if only some of the grouped targets are out of date or missing 'make' will realize that running the recipe will update all of the targets.

As an example, this rule defines a grouped target:

```
foo bar biz &: baz boz
        echo $^ > foo
        echo $^ > bar
        echo $^ > biz
```

During the execution of a grouped target's recipe, the automatic variable '$@' is set to the name of the particular target in the group which triggered the rule. Caution must be used if relying on this variable in the recipe of a grouped target rule.

第十一章　正则表达式

11.1　复习与提示

大数据方法的两大支柱数值运算和字符串处理。处理数据时常用 Python 程序，它胜任数值运算。但在在命令行中控制程序和管理数据，都是整理文件，其本质是在处理文件名，即字符串处理。命令和程序本身，都是计算机与人通过符合语法的字符串交流，把要做的事表达清楚，编译器、解释器是字符的语义和语法处理器。写程序与命令，和《写作与沟通》有着类似的目的。

本节课我们学习处理文字的微语言，正则表达式。相比 Python、Bash、Scheme，正则表达式的语法很少，没有程序结构，不是通用编程语言，图灵不完全。正则表达式的重要性在于它嵌入到了所有通用语言中。Python、Bash、Scheme 都内嵌正则表达式，使用成为普适的字符串处理工具。掌握了它之后，用命令行说话会变得简洁，使用管理文件得心应手。

软件准备

pcre2-utils Perl 正则表达式工具

libpcre2-dev Perl 正则表达式文档和开发工具

libregexp-debugger-perl 正则表达式调试器

```
# Debian
apt install pcre2-utils libpcre2-dev libregexp-debugger-perl
```

文件列表素材

```
wget "http://hep.tsinghua.edu.cn/~orv/pd/usr_share_doc_list.txt"
```

```
--2022-07-27 11:59:50--  http://hep.tsinghua.edu.cn/~orv/pd/usr_share_doc_list.txt
Resolving hep.tsinghua.edu.cn... 101.6.6.219, 2402:f000:1:416:101:6:6:219
Connecting to hep.tsinghua.edu.cn|101.6.6.219|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 128653 (126K) [text/plain]
Saving to: ‘usr_share_doc_list.txt’
```

Debian 系统配置文件素材

```
wget "http://hep.tsinghua.edu.cn/~orv/pd/sources.list"
```

```
    --2022-07-27 12:21:55--  http://hep.tsinghua.edu.cn/~orv/pd/sources.list
    Resolving hep.tsinghua.edu.cn... 101.6.6.219, 2402:f000:1:416:101:6:6:219
    Connecting to hep.tsinghua.edu.cn|101.6.6.219|:80... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 1784 (1.7K)
    Saving to: ‘sources.list’


sources.list          0%[                    ]       0  --.-KB/s
sources.list        100%[===================>]   1.74K  --.-KB/s    in 0s

    2022-07-27 12:21:55 (200 MB/s) - ‘sources.list’ saved [1784/1784]
```

HDF5 素材

```
wget "http://hep.tsinghua.edu.cn/~orv/pd/ser.h5"
```

```
    --2022-07-27 13:06:55--  http://hep.tsinghua.edu.cn/~orv/pd/ser.h5
    Resolving hep.tsinghua.edu.cn... 101.6.6.219, 2402:f000:1:416:101:6:6:219
    Connecting to hep.tsinghua.edu.cn|101.6.6.219|:80... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 3648 (3.6K)
    Saving to: ‘ser.h5’


ser.h5                0%[                    ]       0  --.-KB/s
ser.h5              100%[===================>]   3.56K  --.-KB/s    in 0s

    2022-07-27 13:06:55 (301 MB/s) - ‘ser.h5’ saved [3648/3648]
```

`make`

- 数据流水线构建最佳工具。
  - 程序写得尽可能短，特定场景下 Python 之外的工具更能胜任。
  - 不要一切都用“我人生中掌握的第一门程序语言”实现。
- 函数式编程：一切都是函数，从输入到输出的映射。
  - 数据驱动编程：循环是暗含的，只关注组成单元的处理。
- `make` 处理的替换阶段与执行阶段。
- `guile` 的 scheme 语言在 `make` 中的调用。

```
target: source
        program source target #如何做
        program $^ $@ #如何做
```

字符搜索匹配

```
seq 50 | grep 7
```

```
7
17
27
37
47
```

如何匹配更复杂的规律?

- 正则表达式
- 更高级的文件管理
- 文件内容识别

大家熟悉了使用管道和 grep 命令，实现了文字处理。此处的例子把 1 到 50 里面所有带 7 的数字输出。但是我们面对海量数据文件时，会产生更复杂的过滤需求，例如“7 之前必须是 1 或 4”，“7 且它不在开头和结尾”，“7 但不能与另一个 7 挨在一起”，乃至人类语言难以描述但逻辑上又可定义的需求。正则表达式正是完成此类任务的最佳工具。

正则表达式设计之初，是为了讨论计算机的基本原理所构造的数学工具：到底什么是计算机，计算机有几种形式，图灵机为什么是通用计算机。以及那些机器与图灵机不等价，它们的功能是图灵机的什么子集，它们对应代数系统与图灵机的相比有什么区别？它恰好可以处理字符串，即文件名和文本文件的内容。

高级匹配

- 所有以 3 结尾的两位数

```
seq 50 | grep .3
```

```
13
23
33
43
```

- 23、233、2333、……

```
seq 10000 | grep -E ^23+$
```

```
23
233
2333
```

比如，我只要以 3 结尾的两位数，即把 3 排除只要 13、23、33 和 43。如果过滤所有带 3 的数

```
seq 50 | grep 3
```

```
3
13
23
30
31
32
33
34
35
36
37
38
39
43
```

但我想要以 3 结尾的，这个句点 "." 是正则表达式的通配符，匹配某个字符。因此它必须返回某字符与 3 ，就把 31、32 去除了。

下一个例子，我们想拿到以 2 开头，后面全都是 3 的数，如 23、233、2333。这可以通过在 1 到 10000 中使用正则表达式筛选。其含义是第一个数字必须是 2，后只接一个以上的 3 而无其它数。

这两个例子尚可用人类语言表达，更复杂的正则表达式将超越人类语言的交流效率。人类语言比较冗余，形式语言威力大。

11.2 正则表达式

Regular Expressions

- 多数情况下文本处理的 最佳工具
- 优秀工具的典范：
  1. 有严格的数学模型 – 用户和开发者之间沟通容易
  2. 有标准或约定俗成的语法 – 知识迁移
  3. 有广泛的实现 – 学好正则表达式，走遍天下都不怕
- 实验物理用户：
  - 只要能写出正确的表达式，就可以使用高性能的正则表达式引擎。

参考书

- Jeffrey Friedl, Mastering Regular Expressions
- Michael Sipser, Introduction to the Theory of Computation

正则表达式的形式定义

设 Σ 是字母表（不可分符号集），A 和 B 是字母表 Σ 上的有限字符串的集合，我们定义：

并（Union） $A \cup B = \{x \mid x \in A \text{ 或 } x \in B\}$

连接（Concatenation） $A.B = \{x.y \mid x \in A \text{ 且 } y \in B\}$

Kleene 星号（Kleene star） $A^* = \bigcup_{k\geq 0} A^k = \bigcup_{k\geq 0}\{x_1x_2\ldots x_k \mid i \leq k \to x_i \in A\}$ 例如

$$\{a, ab\}^* = \{\varepsilon, a, ab, aab, aba, aa, abab, aaa, aaab, \ldots\}$$

R 是正则表达式 (Regular Expression，RE)，若 R 通过以下归纳定义得到：

- $\{a\}, a \in \Sigma$，
- $\{\varepsilon\}$，即空字符串（长度为 0）的集合，
- $\emptyset$ （空集）
- 若 R_1, R_2 是 RE，则 $R_1 \cup R_2$ 亦然 （对 $\cup$ 运算封闭），
- 若 R_1, R_2 是 RE，则 $R_1.R_2$ 亦然（对 . 运算封闭），
- 若 R_1 是 RE，则 $(R_1)^*$ 亦然 （对 $*$ 运算封闭）。

正则表达式的等价表述

以下命题等价：

- R 是正则表达式
- R 是被有限自动机接受的语言
- R 是被 Chomsky 的 3-型文法生成的语言

有限自动机 $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$，其中 Q 是有限的状态集，Σ 是有限的字母表，$\delta : Q \times \Sigma \to \mathcal{P}(Q)$ 是状态转移关系，$q_0 \in Q$ 是初始状态，$F \subset Q$ 是接受状态的集合。

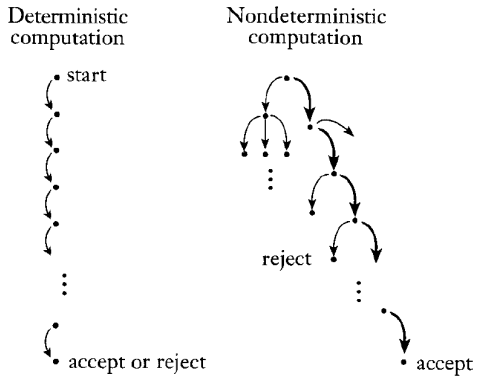
3-型文法满足右线性或左线性

右线性的例子：

$$A \to wB, A \to w, A \to \varepsilon$$

其中 A, B 表示可分符号（non-terminal symbol），$w \in \Sigma$ 是不可分符号（terminal symbol）。

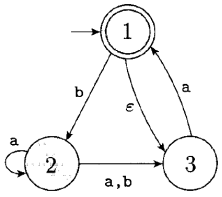
确定与非确定自动机



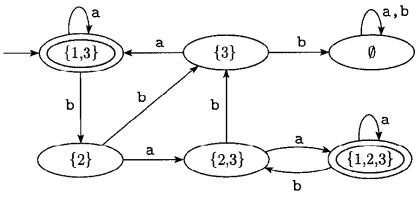
对于任意的非确定型自动机，存在与其等价的确定型自动机。即，任何被非确定型自动机接受的语言，都可以被某一个确定型自动机接受。

例子

- 字母表 $\Sigma = \{a, b\}$ 。非确定型有限自动机 $\mathcal{M}$:



- 一个与 $\mathcal{M}$ 等价的确定型自动机:



11.3 应用实例

语法要素：类算术运算的组合规则

- 所有字符默认与自己匹配
- . 代表任意字符
- ^, $ 开始与结束
- * 任意重复
  - + 至少一次重复

  - `?` 0 或 1 次
  - `{n,m}` 重复 n 至 m （包含）次
- `()` 组合
- `|` 或
- `[]` 字符集
  - `[0-9]`
  - `[a-zA-Z]`
  - `[^abc]`
- `\<`, `\>` 单词的开始和结束处的空白符
- `\b` 单词两边的空白符
- `\B` 非单词两边的空白符， `\b` 的补集

正则表达式流派

基础正则表达式 Basic RE

- `? + { | ( )` 是字符本身，需要特殊含义得加上 `\` ，例如 `\? \+` 等

扩展正则表达式 Extended RE

- GNU 环境里与基础正则表达式功能完全一致，只是语法约定相反， `\? \+` 等代表 “?” 和 “+” 本身， `? +` 等有特殊含义。

Perl 正则表达式 Perl-Compatible RE

- 与 Perl 5 语言正则表达式有同样功能
- 超出有限状态自动机的功能：递归、条件、引用等
  - 严谨来讲已经不再是数学意义上的正则表达式，但更加实用

`grep`

- 基本功能：检验输入的第一行是否与正则表达式匹配，若是则输出该行。
- `egrep` 等价于 `grep -E` ，扩展正则表达式

```
file $(which egrep)
```

```
/bin/egrep: POSIX shell script, ASCII text executable
```

```
cat /bin/egrep
```

```
#!/bin/sh
exec /bin/grep -E "$@"
```

文档

- man grep
- info grep
- 源于 ed 的语法 global/regular expression/print → g/re/p

试验

- ^, $ 开始与结束
- + 至少一次重复
- ? 0 或 1 次
- () 组合

```
seq 10000 | egrep ^2?3+$
```

```
3
23
33
233
333
2333
3333
```

```
seq 10000 | egrep '^(23)*$'
```

```
23
2323
```

试验（二）

```
seq 10000 | egrep '^(23?)+$'
```

```
2
22
23
222
223
232
2222
2223
2232
2322
2323
```

- 正则表达式写下了人类语言已经很难描述匹配的规则。

试验（三）

- | 或
- [] [0-9] 字符集

```
seq 10000 | egrep '^(2(3|4))+$'
```

```
23
24
2323
2324
2423
2424
```

```
seq 10000 | egrep '^(23[3-6])+$'
```

```
233
234
235
236
```

试验（四）

- {n,m} 重复 n 至 m （包含）次

```
seq 1000000 | egrep '^23{2,4}$'
```

```
233
2333
23333
```

更多字符集

- 在 [ ] 之内有效
  - [:alpha:] 字母
  - [:digit:] 数字
  - [:alnum:] 字母或数字
    - \w 与 [[:alnum:]] 同义， \W 与 [^[:alnum:]] 同义
  - [:graph:] 可见的字符，不包括空白
  - [:lower:] 小写字母
  - [:print:] 可见的字符和空白
  - [:punct:] 标点
  - [:space:] 空白符，包括 \t \r \n
    - \s 与 [[:space:]] 同义， \S 与 [^[:space:]] 同义
  - [:upper:] 大写字母
  - [:xdigit:] 十六进制数

11.4 文件名匹配练习

TEX

```
grep tex usr_share_doc_list.txt | head
```

```
drwxr-xr-x  4 root root 4096 Feb  8  2021 auctex
drwxr-xr-x  3 root root 4096 Feb  8  2021 chktex
drwxr-xr-x  4 root root 4096 Feb  8  2021 dot2tex
drwxr-xr-x  2 root root 4096 Feb  8  2021 fonts-texgyre
drwxr-xr-x  2 root root 4096 Feb  8  2021 gettext
drwxr-xr-x  2 root root 4096 Feb  8  2021 gettext-base
drwxr-xr-x  2 root root 4096 Feb  8  2021 latex2rtf
lrwxrwxrwx  1 root root   16 May 11  2016 latex-cjk-chinese -> latex-cjk-common
drwxr-xr-x  5 root root 4096 Apr 14 15:46 latex-cjk-common
drwxr-xr-x  3 root root 4096 Aug  8  2017 latex-mk
```

去掉与 TEX 无关的项

- 去掉 `text` `texgyre` [ˈdʒʌɪɘ, ˈgʌɪɘ] `texinfo`

```
egrep 'tex([^tgi]|$)' usr_share_doc_list.txt | head
```

```
drwxr-xr-x  4 root root 4096 Feb  8  2021 auctex
drwxr-xr-x  3 root root 4096 Feb  8  2021 chktex
drwxr-xr-x  4 root root 4096 Feb  8  2021 dot2tex
drwxr-xr-x  2 root root 4096 Feb  8  2021 latex2rtf
lrwxrwxrwx  1 root root   16 May 11  2016 latex-cjk-chinese -> latex-cjk-common
drwxr-xr-x  5 root root 4096 Apr 14 15:46 latex-cjk-common
drwxr-xr-x  3 root root 4096 Aug  8  2017 latex-mk
drwxr-xr-x  3 root root 4096 Feb  8  2021 latexmk
drwxr-xr-x  2 root root 4096 Feb  8  2021 libpod-latex-perl
drwxr-xr-x  2 root root 4096 Apr 14 15:44 libptexenc1
```

先有 TEX 再有 LATEX

```
egrep 'tex\S*latex' usr_share_doc_list.txt | head
```

```
drwxr-xr-x  2 root root 4096 Apr 14 15:45 texlive-latex-base
drwxr-xr-x  2 root root 4096 Apr 14 15:45 texlive-latex-extra
drwxr-xr-x  2 root root 4096 Apr 14 15:45 texlive-latex-recommended
```

TEX 为单词的起始

```
egrep '\btex' usr_share_doc_list.txt | head
```

```
drwxr-xr-x  2 root root 4096 Feb  8  2021 fonts-texgyre
drwxr-xr-x  2 root root 4096 Feb  8  2021 libdjvulibre-text
drwxr-xr-x  2 root root 4096 Apr 14 15:44 tex-common
drwxr-xr-x  2 root root 4096 Mar 24 11:19 tex-gyre
drwxr-xr-x  3 root root 4096 Feb  8  2021 texinfo
drwxr-xr-x  4 root root 4096 Apr 14 15:46 texlive-base
drwxr-xr-x  2 root root 4096 Apr 14 15:45 texlive-bibtex-extra
drwxr-xr-x  2 root root 4096 Apr 14 15:44 texlive-binaries
drwxr-xr-x 25 root root 4096 Feb  8  2021 texlive-doc
drwxr-xr-x  2 root root 4096 Apr 14 15:46 texlive-extra-utils
```

带有 python3 的

```
egrep '\<python3\>' usr_share_doc_list.txt | head
```

```
drwxr-xr-x  2 root root 4096 Apr 14 15:44 python3
drwxr-xr-x  2 root root 4096 Apr 14 15:44 python3.9
drwxr-xr-x  2 root root 4096 Apr 14 15:44 python3.9-minimal
drwxr-xr-x  2 root root 4096 Feb  8  2021 python3-apt
drwxr-xr-x  2 root root 4096 Feb  8  2021 python3-attr
drwxr-xr-x  2 root root 4096 Feb  8  2021 python3-automat
drwxr-xr-x  2 root root 4096 Feb  8  2021 python3-backcall
drwxr-xr-x  2 root root 4096 Feb  8  2021 python3-bcrypt
drwxr-xr-x  2 root root 4096 Feb  8  2021 python3-cairo
drwxr-xr-x  2 root root 4096 Feb  8  2021 python3-certifi
/bin/grep: write error: Broken pipe
```

恰好出现 3 次的字母数字

```
egrep '\b[[:alnum:]]{3}\b' usr_share_doc_list.txt | head
```

```
drwxr-xr-x  2 root root 4096 Aug  8  2017 abootimg
drwxr-xr-x  2 root root 4096 Feb  8  2021 acl
drwxr-xr-x  2 root root 4096 Jul  2  2019 acpi
drwxr-xr-x  2 root root 4096 Feb  8  2021 adb
drwxr-xr-x  3 root root 4096 Oct 12  2019 adduser
drwxr-xr-x  2 root root 4096 Feb  8  2021 adwaita-icon-theme
drwxr-xr-x  2 root root 4096 Jul  2  2019 alsa-oss
drwxr-xr-x  6 root root 4096 Feb  8  2021 alsa-tools
drwxr-xr-x  8 root root 4096 Feb  8  2021 alsa-tools-gui
drwxr-xr-x  2 root root 4096 Feb  8  2021 alsa-utils
/bin/grep: write error: Broken pipe
```

11.5 流编辑器 sed

sed: stream editor

- `sed` 是 stream editor，在“流”上进行编辑
- 常用的操作是替换，可以使用正则表达式
  - 支持 Basic RE 和 Extended RE `-r`

文档

- `man sed`
- `info sed`
- 名字是 streamed `ed`，前身是 `ed`

替换

```
echo hello | sed 's/lo/do/'
```

```
heldo
```

生成 CSV

- 把 9 处的“,”转为换行
- g 代表全部替换

```
seq -s, -w 00 99 | sed 's/9,/9\n/g'
```

```
00,01,02,03,04,05,06,07,08,09
10,11,12,13,14,15,16,17,18,19
20,21,22,23,24,25,26,27,28,29
30,31,32,33,34,35,36,37,38,39
40,41,42,43,44,45,46,47,48,49
50,51,52,53,54,55,56,57,58,59
60,61,62,63,64,65,66,67,68,69
70,71,72,73,74,75,76,77,78,79
80,81,82,83,84,85,86,87,88,89
90,91,92,93,94,95,96,97,98,99
```

生成 CSV （二）

- 在 4 或 9 处换行
- \1 代表第一个括号中的成分

```
seq -s, -w 00 99 | sed -r 's/([49]),/\1\n/g'
```

```
00,01,02,03,04
05,06,07,08,09
10,11,12,13,14
15,16,17,18,19
20,21,22,23,24
25,26,27,28,29
30,31,32,33,34
35,36,37,38,39
40,41,42,43,44
45,46,47,48,49
50,51,52,53,54
55,56,57,58,59
60,61,62,63,64
65,66,67,68,69
70,71,72,73,74
75,76,77,78,79
80,81,82,83,84
85,86,87,88,89
90,91,92,93,94
95,96,97,98,99
```

使用 TUNA 源

- 多个替换可以用 -e 分开 jammy ['dʒami]

```
sed -e 's,archive.canonical.com/,mirrors.tuna.tsinghua.edu.cn/,' -e 's/trusty/jammy/' sources.list | head
```

```
# deb cdrom:[Ubuntu 14.04 LTS _Trusty Tahr_ - Release amd64 (20140417)]/ jammy main restricted

# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse

## Major bug fix updates produced after the final release of the
## distribution.
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
```

过滤掉无用的注释

- 多个替换也可以用 ; 分隔
- -n 默认不输出， p 来输出

```
sed -n '/^[^#]/{s,archive.canonical.com/,mirrors.tuna.tsinghua.edu.cn/,;s/trusty/jammy/;p}' sources.list | head
```

```
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse
```

合成一个表达式

- 使用匹配引用 \1

```
sed -nr '/^[^#]/{s,archive.canonical.com(.*)trusty,mirrors.tuna.tsinghua.edu.cn\1jammy,;p}' sources.list | head
```

```
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy main restricted universe multiverse
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-updates main restricted universe multiverse
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-backports main restricted universe multiverse
deb http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse
deb-src http://mirrors.tuna.tsinghua.edu.cn/ubuntu/ jammy-security main restricted universe multiverse
```

grep 达到类似效果

```
egrep -v '^(#.*)?$' sources.list
```

```
deb http://archive.canonical.com/ubuntu/ trusty main restricted universe multiverse
deb-src http://archive.canonical.com/ubuntu/ trusty main restricted universe multiverse
deb http://archive.canonical.com/ubuntu/ trusty-updates main restricted universe multiverse
deb-src http://archive.canonical.com/ubuntu/ trusty-updates main restricted universe multiverse
deb http://archive.canonical.com/ubuntu/ trusty-backports main restricted universe multiverse
deb-src http://archive.canonical.com/ubuntu/ trusty-backports main restricted universe multiverse
deb http://archive.canonical.com/ubuntu/ trusty-security main restricted universe multiverse
deb-src http://archive.canonical.com/ubuntu/ trusty-security main restricted universe multiverse
```

- 达成成就：没有任何字母数字的正则表达式

11.6 bash 正则表达式

bash 内建正则判断

- 由 [[EXPR =~ REGEX]] 判断 EXPR 是否能被 REGEX 匹配。
  - 支持扩展正则表达式
  - 匹配结果由 BASH_REMATCH 列表给出
- 比每次调用 egrep 快很多

```
seq 1000 | until ! read line
do
    if [[ $line =~ ^23+$ ]]; then
        echo ${line}
        echo ${BASH_REMATCH[0]}
    fi
done
```

```
23
23
233
233
```

取得匹配结果

```
while read entry; do
    if [[ $entry =~ ^([^#].*)archive.canonical.com(.*)trusty ]]; then
        echo ${BASH_REMATCH[1]} ${BASH_REMATCH[2]}
    fi
done < sources.list
```

```
deb http:// /ubuntu/
deb-src http:// /ubuntu/
deb http:// /ubuntu/
deb-src http:// /ubuntu/
deb http:// /ubuntu/
deb-src http:// /ubuntu/
deb http:// /ubuntu/
deb-src http:// /ubuntu/
```

例子

OSIRIS 的文件整理。

```
/eos/juno/Commissioning_Dryrun/OSIRIS/2024/0311/OSIRISData_hybrid_20240311_101439.dat
```

链接成

```
/junofs/OSIRIS/0311/101439.h5
```

引用常见问题

```
pattern='\.' # 引号被 bash 处理

[[ . =~ $pattern ]] # 成功
[[ . =~ \. ]] # 成功

[[ . =~ "$pattern" ]] # 失败
[[ . =~ '\.' ]] # 失败
```

- \ 同时在 bash 和正则语句意为“除去特殊含义”。需要格外注意
- 引号在变量赋值时使用时，变量的值不含引号。
  - 在 [[ ]] 里写引号时，所有字符都没有特殊含义

11.7 Perl 正则表达式

多行匹配

- grep 的多行匹配功能很弱，应当使用 pcre2grep
- pcre2 吸纳了 Perl 的正则表达式，一部分来自 Python 正则表达式。

```
columns=( "DATASET.*SER"
          "DATATYPE.*F64"
          "DATASET.*SER_stddev"
          "DATATYPE.*F64"
        )

h5dump -A ser.h5| pcre2grep -M "$(echo ${columns[@]} | sed 's/ /(.|\n)*/g')"
echo $?
```

```
   DATASET "SER" {
      DATATYPE  H5T_IEEE_F64LE
      DATASPACE  SIMPLE { ( 100 ) / ( 100 ) }
   }
   DATASET "SER_stddev" {
      DATATYPE  H5T_IEEE_F64LE
0
```

有限状态自动机之外的功能

- 基本正则表达式与有限自动机 finite automata 等价
- 在 FA 之外时，可引入更强大的处理功能

提供类似 sed 的命令语法

- Perl 的 -p 选项

  causes Perl to assume the following loop around your program, which makes it iterate over filename arguments somewhat like sed:

```
while (<>) {
    ...             # your program goes here
} continue {
    print or die "-p destination: $!\n";
}
```

高级构造

- 前趋引用匹配：形似 sed 的替换
- iterative matches \G where previous match ends
- look ahead behind
- atmoic grouping 不再缩短

选择结构

- conditional constructs (140)
  - 使用之前的匹配 (<)?\w+(?(1)>)

递归结构

- 由 pcre 提出，用于匹配形如 $a^n b^n$ 的平衡问题。

例子：使用“回头看”构造

- 如果逗号之前为 4 或 9 ，就把它替换成换行。

```
seq -s, -w 00 49 | perl -p -e 's/(?<=[49]),/\n/g'
```

```
00,01,02,03,04
05,06,07,08,09
10,11,12,13,14
15,16,17,18,19
20,21,22,23,24
25,26,27,28,29
30,31,32,33,34
35,36,37,38,39
40,41,42,43,44
45,46,47,48,49
```

11.8 Python 正则表达式

re 模块

受 Perl 正则表达的影响，语法参数了 Perl 正则表达式。

```
import re

m = re.search('(?<=abc)def', 'abcdef')
m.group(0)
```

```
def
```

11.9 调试

rxrx

- Perl 的正则表达式调试和教学工具

11.10 常见问题

匹配长度

- greedy: 贪婪匹配
- lazy: 最小匹配

	?	+	*
greedy	?	+	*
lazy	??	+?	*?

第十二章　从 sed 到 awk

12.1　复习准备

软件安装

下载文件

```
wget 'http://hep.tsinghua.edu.cn/~orv/pd/ccompiler_opt.py'
```

- Debian

  ```
  apt install gawk-doc feedgnuplot
  ```

正则表达式

- 普适的字符串处理工具
  - `^`, `$` 开始与结束
  - `+` 至少一次重复
  - `?` 0 或 1 次
  - `()` 组合
  - `|` 或
- 重要工具 `grep` 行匹配 `sed` 替换
- 进阶： Perl 正则表达式
  - 多行匹配

正则表达式的写法

1. 思考清楚我们需要匹配什么文字，这些文字在文本里都可能以什么形式出现
2. 写出一个正则表达式
3. 试验正则表达式，改进
   - 哪些是我想要匹配却没匹配的

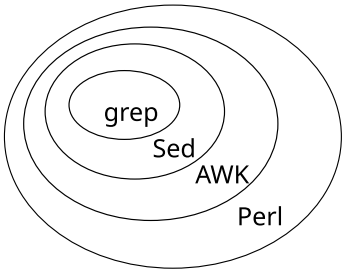
- 哪些是我不想匹配但却匹配了的

这几乎适用于所有程序的写作。

几乎是所有科学实验的思路

1. 思考清楚我们在证伪什么论断
2. 设计实验
3. 做实验，看它是否成功证伪了论断
    - 若是，理论被推翻
    - 若不是，理论经受住了考验，思考更强更精细的可证伪假说

先试用功能集小的，再试用大的



对人也一样

在信任一个人能胜任工作之前，先布置一个小任务

1. 打字
2. 小作业
3. 大作业

自动处理文本和字符串的重要性

- 节省大量的重复劳动
    - 找到一种解决问题的途径和高效的工作方法，让人满足
    - 是工作和苦劳的本质区别
    - 程序化地解决问题，让问题本身变得更有趣
- 入门时，程序化解决文本问题花的时间更长
    - 注意不仅要学习如何解决问题，还要学习识别问题

参考资料

- Dougherty and Robbins, Sed & Awk, 1997.
- 陈皓，AWK 简明教程
- `info sed`, `info awk`

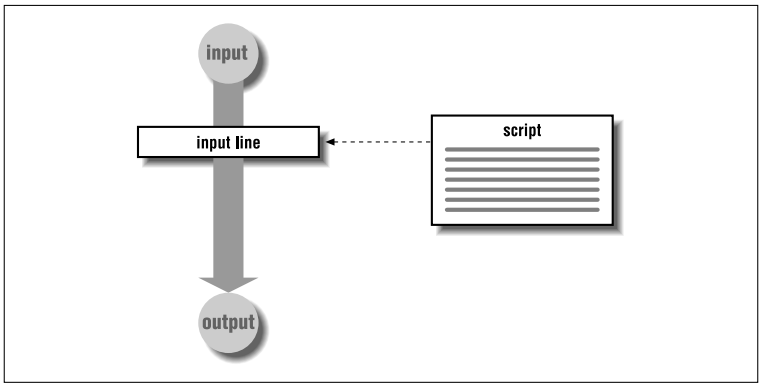
12.2 sed 进阶

命令行语法

```
sed/awk script filename
```

1. 一行写不下时
2. 担心特殊字符在 `shell` 被转义时

数据驱动的编程



匹配变换模式

1. 通过 `/regex/` 匹配要修改的行
2. 通过随后的命令修改之

使用脚本

```
sed -f tuna.sed sources.list
```

- 所有命令先处理第一行，再处理第二行，……

定位行

1. 什么都不写代表所有行
2. `/regex/` 代表满足 regex 的行
3. 1, 2, 3 行号，例如

   ```
   1d
   ```

   为删除第一行。

   `$` 代表最后一行。
4. `line1,line2` 代表从 line1 行到 line2 行，闭区间

   ```
   50,$d
   ```

   把从第 50 行到最后一行都删除
5. 在行的定位后加 ! 代表反义

   ```
   50,$!d
   ```

   与

   ```
   1,49d
   ```

   等价

定位行的嵌套

可以把行的定位嵌套，例如，删除所有从第 1 到 49 行中的空行，并把 lo 换成 do

```
1,49{
  /^\s*$/d
  s/lo/do/
}
```

比较新旧文件的区别

```
sed '1,100s/Aug/ 08/' usr_share_doc_list.txt > usr_share_doc_list.txt.new
diff -u usr_share_doc_list.txt{,.new}
```

```
--- usr_share_doc_list.txt 2021-08-30 11:59:19.000000000 +0800
+++ usr_share_doc_list.txt.new 2023-07-27 12:18:20.209493339 +0800
@@ -1,5 +1,5 @@
 total 8868
-drwxr-xr-x  2 root root 4096 Aug  8  2017 abootimg
+drwxr-xr-x  2 root root 4096  08  8  2017 abootimg
 drwxr-xr-x  2 root root 4096 Feb  8  2021 acl
 drwxr-xr-x  2 root root 4096 Jul  2  2019 acpi
 drwxr-xr-x  2 root root 4096 Feb  8  2021 adb
@@ -92,7 +92,7 @@
 drwxr-xr-x  2 root root 4096 Apr 14 15:44 cl-swank
```

```
 drwxr-xr-x  2 root root 4096 Feb  8  2021 cl-trivial-gray-streams
 drwxr-xr-x  2 root root 4096 Feb  8  2021 cl-unicode
-drwxr-xr-x  2 root root 4096 Aug 30 10:21 cmatrix
+drwxr-xr-x  2 root root 4096  08 30 10:21 cmatrix
 drwxr-xr-x  2 root root 4096 Feb  8  2021 coreutils
 drwxr-xr-x  3 root root 4096 Feb  8  2021 cowsay
 drwxr-xr-x  2 root root 4096 Feb  8  2021 cowsay-off
```

sed 的典型使用场景

一文多改 一个文件上投入多个改动

一脚多文 一个脚本用在多个文件上

高级 grep 在查找关键词之前，先对待找的文本进行清洗

管道过滤 直接对着标准输入过滤，给到标准输出

数据清洗

数据清理：把数据整理成方便的格式

- 在物理学之外的其它“大数据”学科，90% 的工作是数据清洗。

例子：课件中的代码块

`https://git.tsinghua.edu.cn/physics-data/lecture/-/blob/master/Makefile`

```
e%.tex: p%.tex
	sed -e 's/{sqlite3}/{sqlite3}/' -e 's/python/python/' \
		-e 's/bash/bash/' \
		-e '/{minted}/s/\[\]/\[bgcolor=lightgray,fontsize=\\scriptsize\]/' \
		-e 's/\{\scriptsize\begin{verbatim}/{\\scriptsize\\begin{verbatim}/' \
		-e 's/end{verbatim}}/end{verbatim}}/' < $< > $@
```

多行匹配

- 部分 `pcre2grep -M` 的功能。

```
  echo "This is your cat
  my cat's name is betty
This is your dog
  my dog's name is frank" | sed 'N;s/\n/,/'
```

```
This is your cat,  my cat's name is betty
This is your dog,  my dog's name is frank
```

Hold space

为 sed 自动机配了一个队列。

g 将 hold space 中的内容拷贝到 pattern space 中，原来 pattern space 里的内容清除

G 将 hold space 中的内容 append 到 pattern space 后

h 将 pattern space 中的内容拷贝到 hold space 中，原来的 hold space 里的内容被清除

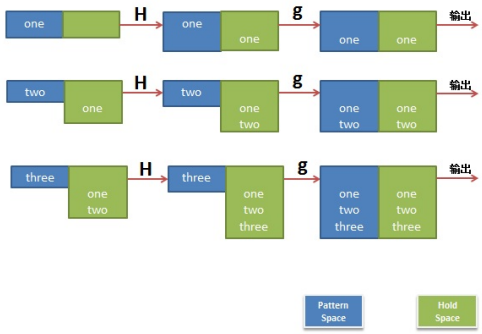
H 将 pattern space 中的内容 append 到 hold space 后

x 交换 pattern space 和 hold space 的内容

收集-积累-输出

```
echo "one
two
three" | sed "H;g"
```

```
one

one
two

one
two
three
```



10 乘 10 矩阵

- 执行 N 9 次。重复过多，不如使用 python 辅助构建字符串。

```
seq -w 00 99 | sed $(python3 -c "print('N;'*9)")'s/\n/,/g'
```

反序

```
seq 5 | sed '1!G;h;$!d'
```

```
5
4
3
2
1
```

```
seq 5 | tac
```

```
5
4
3
2
1
```

12.3 ed 编辑

行编辑器

- 在显示器大行其道之前
  - 计算机终端的主要输出方式是打印机，当时最佳的方式行编辑

试验

```
ed departments.csv
```

- `d`
- `/regex/` 找到下一个满足 regex 的行
  - `g/regex/` 找到所有满足 regex 的行
- `s/pattern/replacement/`
- `g/propagating/p`
- `echo 'g/propagating/p' | ed /tmp/texts.log`

sed 与 ed 的区别

- ed 是交互的，默认作用于一行，用 `/regex/` 扩展编辑范围
- sed 是批处理的，默认作用于每一行，用 `/regex/` 限制编辑范围

awk 与 sed 的区别

- 不再使用 ed 的单字母命令，改用 C 语言风格的语法，例如`/regex/{ print }`

ed 的命令影响了 ex

ex 是 vi 的前身，后者发展成 VIM (Vi IMproved)， VIM 的开发停滞，开发者正在转向 neovim

12.4 awk 特点

sed 到 awk

有些简单的问题可用 sed 解决，复杂一些的问题适合 awk 解决

sed 适合解决

1. 批量修改文本
2. 把同一个修改应用到多个文件上
3. 把一种格式的文本转化成另一种

sed 的操作（最常用的是查找-替换），更接近于编辑器

awk 是模式匹配语言

- 用 awk 描述数据中的规律，从而取得对应的信息
- 单行的 awk 程序可以从命令行直接输入
- 把文本看成是一直数据仓库，例如 CSV
  - 在文本处理的同时，集成算术运算

sed 与 awk 的共同点

- 都是数据流驱动的编程语言但与 Make 语言不同，不是函数式语言
- 大量使用正则表达式
- 可以快速从 bash 调用
- awk 起源于 grep 和 sed，而后两者都脱胎于 `ed` 编辑器

学习 sed 和 awk 的步骤

☒ 调用 sed 和 awk

☒ 使用正则表达式

☒ 从 bash 调用时的引用

即用 \ 和 ' “废掉他人武功”

☐ 书写 sed 和 awk 的脚本

学习新语言的原因

- ~~实验物理的大数据方法~~
  - 28 天速成七种语言：Python, Bash, Make, Regex, Awk, R, SQL
- 经济性：有了 Python 基础，学习其它语言门槛大大降低，不再需要额外精力
  - 第一门语言难度 100%，思维广度 +100%
  - 第二门语言难度 25%，思维广度 +100%
  - 第三门语言难度 1%，思维广度 +100%
  - 回报越来越大，learn x in y mintes。

12.5 awk 基础

运行 awk

```
awk '{print $1}' list
```

计算 1 到 100 的和

```
seq 100 | awk '{total+=$NF} END {print total}'
```

```
5050
```

正式入门

```
echo | awk '{ print "Hello, world!" }'
```

```
Hello, world!
```

```
echo "Hello, world!" | awk '{ print }'
```

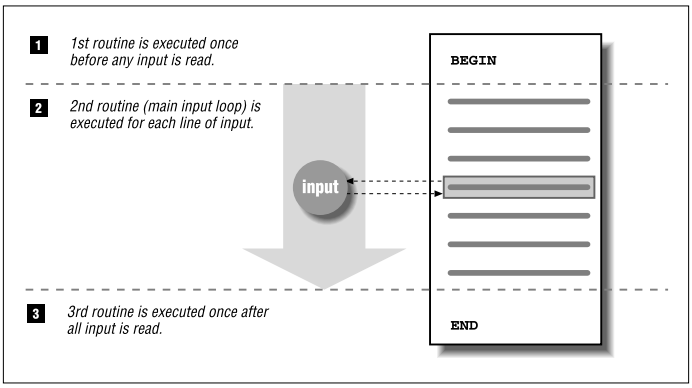
```
Hello, world!
```

```
awk 'BEGIN { print "Hello world!" }'
```

```
Hello world!
```

awk 的运行模式

- 数据驱动的编程风格。



内建的变量

变量	含义
`$0`	当前记录（这个变量中存放着整个行的内容）
`$1~$n`	当前记录的第 n 个字段，字段间由 FS 分隔
`FS`	输入字段分隔符 默认是空格或 Tab
`NF`	当前记录中的字段个数，就是有多少列
`NR`	已经读出的记录数，就是行号，从 1 开始。 如果有多个文件话，这个值也是不断累加。
`FNR`	当前记录数，与 NR 不同的是，这个值会是各个文件自己的行号
`RS`	输入的记录分隔符， 默认为换行符
`OFS`	输出字段分隔符， 默认也是空格
`ORS`	输出的记录分隔符，默认为换行符
`FILENAME`	当前输入文件的名字

- `-F` 可以改变 `FS` 的值，例如 `awk -F,`

替换文本内容

- info: gawk → sample programs → clones
  - cut
  - egrep
  - id
  - split
  - tee
  - uniq
  - wc

课堂练习： wc

- wc -l

```
seq 3 100 | awk '{lines++} END {print lines}'
```

```
98
```

- wc -c

```
seq 3 100 | awk '{chars+=length($0)+1} END {print chars}'
```

```
288
```

```
seq 3 100 | wc -c
```

```
288
```

课堂练习： uniq

- uniq -c

```
echo "a
a
b
c
c
c" | awk -f uniq.awk
```

```
2 a
1 b
3 c
```

课堂练习： egrep

- egrep

```
seq 10000 | awk '/^23+$/ { print }'
```

```
23
233
2333
```

sed 与 awk 的联合

使用管道

```
seq 9 | sed 'H;g' |\
    awk -v RS='' '{for(i=1;i<=NF;i++)
        printf("%dx%d=%d%s", i, NR, i*NR, i==NR?"\n":"\t")}'
```

```
1x1=1
1x2=2 2x2=4
1x3=3 2x3=6 3x3=9
1x4=4 2x4=8 3x4=12 4x4=16
1x5=5 2x5=10 3x5=15 4x5=20 5x5=25
1x6=6 2x6=12 3x6=18 4x6=24 5x6=30 6x6=36
1x7=7 2x7=14 3x7=21 4x7=28 5x7=35 6x7=42 7x7=49
1x8=8 2x8=16 3x8=24 4x8=32 5x8=40 6x8=48 7x8=56 8x8=64
1x9=9 2x9=18 3x9=27 4x9=36 5x9=45 6x9=54 7x9=63 8x9=72 9x9=81
```

乘法表

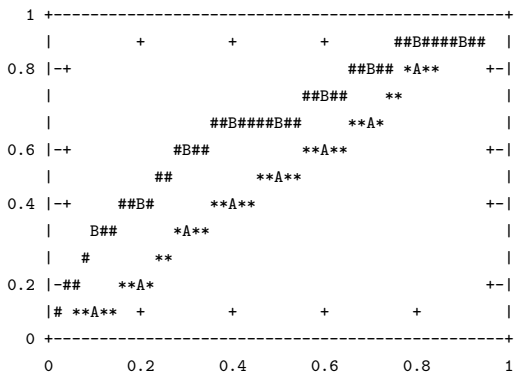
```
seq 9 | sed 'H;g;s/\n/ /g' | awk '{for(i=1;i<=NF;i++)
        printf("%dx%d=%d%s", i, NR, i*NR, i==NR?"\n":"\t")}'
```

```
1x1=1
1x2=2 2x2=4
1x3=3 2x3=6 3x3=9
1x4=4 2x4=8 3x4=12 4x4=16
1x5=5 2x5=10 3x5=15 4x5=20 5x5=25
1x6=6 2x6=12 3x6=18 4x6=24 5x6=30 6x6=36
1x7=7 2x7=14 3x7=21 4x7=28 5x7=35 6x7=42 7x7=49
1x8=8 2x8=16 3x8=24 4x8=32 5x8=40 6x8=48 7x8=56 8x8=64
1x9=9 2x9=18 3x9=27 4x9=36 5x9=45 6x9=54 7x9=63 8x9=72 9x9=81
```

12.6 feedgnuplot

在命令行制图

```
seq 0 0.1 1 | awk '{print $1, $1, sqrt($1)}' | \
    feedgnuplot --domain --lines --points --terminal 'dumb 60,16' --unset grid --exit
```



12.7 awk 练习

统计选课人数

- 使用字典

去掉无用的编译选项

```
{
    GENTOO_ENABLE=1;
    if (match($0, /flags="([^"=]*)"/, cflags)) {
        split(cflags[1], fields);
        for (i in fields) {
            if (match(fields[i], /-m([[:graph:]]*)/, inst)) {
                if (!index(enabled_flags, inst[1])) {
                    GENTOO_ENABLE=0;
                }}}}
    if (!GENTOO_ENABLE) { sub(cflags[1], "-mGENTOO_DISABLE"); }
    print;
}
```

https://gitweb.gentoo.org/repo/gentoo.git/commit/?id=ea2987349ece6f07da01e1c2f3e9808455fa394c

12.8 后备

回顾

科学数据处理的原则

复现 透明 一次 最佳工具

版本控制

Git 与队友分工协作，与明天的自己协作

数据流水线

GNU Make 管理数据的依赖与转换，实现错误恢复和并行计算

命令环境

GNU 环境中强大的小工具组合，开发与使用相融合

计算语言

Python 语法友好，工具丰富，统领 C/C++/Fortran 库

第十三章　关系代数与 SQL

13.1　复习准备

下载课堂练习数据并安装软件

```
wget "http://hep.tsinghua.edu.cn/~orv/pd/dataframe-practice-r1.tar.gz"
tar -xf dataframe-practice-r1.tar.gz
```

- 安装 SQLite3 和查看器，以及 CSV 查看器

  ```
  apt install sqlite3 sqlitebrowser csvkit
  ```

  通过 `https://sqlitebrowser.org/dl/` 介绍的其它方式安装 sqlitebrowser

本周提要

第一周 一切工作都是差分

第二周 一切计算都是数组

第三周 一切行为都是命令

- 程序运行与文件管理：
  1. 外壳 shell: `bash`
  2. 流水线: `make` （函数式编程、描述性编程）
  3. 字符处理：正则表达式 （描述性编程）

第四周 一切数据都是表格

- 学习如何有序组织数据，尽一切努力让统计分析和数据洞见更易进行
  - 关系代数
  - 学习新语言 SQL
- 应用意义
  - 直接应用到分布式 SQL 引擎
- 1981 年和 2014 年的两届图灵奖工作

13.2 关系代数

引例：表格 vs 数组

`eid` 指事例编号 event ID， `ch` 指读数通道 `channel` 。

eid	ch	wave
0	0	(0, 2, 3)
0	2	(0, 0, 1)
1	2	(3, 2, 0)
1	3	(0, 3, 1)
...		

- `eid` 的取值可达几百万 N_{E}， `ch` 的取值是 0 到 29 共 30 个。
- 表格定义：把坐标 (0, 0)、(0, 2)、(1, 2)、(1, 3) 写在列上
- 长度略小于 $N_{\mathrm{E}} \times 30$ 的表格

ch \ eid	0	1
0	(0, 2, 3)	x
2	(0, 0, 1)	(3, 2, 0)
3	x	(0, 3, 1)

- 如果行与列的标号不连续，则另行开辟 `eid` 和 `ch` 的数组，建立 0, 2, 3 与标号 0, 1, 2 的对应关系。
- 形状为 $(N_{\mathrm{E}}, 30)$ 的二维数组

概念

eid	ch	wave
0	0	(0, 2, 3)
0	2	(0, 0, 1)
1	2	(3, 2, 0)
1	3	(0, 3, 1)
...		

- 一切都是表，可以表达一切数据
  - 未必真实存在，可以是逻辑的表
- 集合运算，交、并、差 $\cap, \cup, \setminus$
  - 集合元素是表格的行。
- 线性运算，笛卡尔积、投影、选取 $\otimes, \Pi, \sigma$
  - 表格扩大，列方向与行方向缩小

- 关系运算，连接 $\bowtie$
  - 非关系代数基本运算，可由 $\otimes\sigma$ 组合得到，极常用
- 拓展运算，GroupBy（分组） $\mathcal{G}$
  - 非关系代表运算，不可由基本运算实现，极常用

我们经常遇到表格数据

eid	ch	wave
0	0	(0, 2, 3)
0	3	(0, 0, 1)
1	4	(0, 3, 1)

`eid` 指事例编号 event ID， `ch` 指读数通道 `channel`。 `eid` 的取值可达几百万，=ch= 的取值是 0 到 29 共 30 个。那么是写成表格方便，还是写成 3 维的数组方便呢？写成表格，是否违反了一次原则？

两者都有好处，高维的数组适合静态的、整齐的和一一对应的场景，或者它们的近似情形。但是现实的实验里，有这些条件不满足的情况。数据在更多时候是动态的、参差不齐的和稀疏的。此时表格是更好的数组组织形式。表格由标号列和取值列组成，标号列对应数组的下标，而取值列对应数组的值。当值不存在时，可以省去整个标号，即 0 行；相反当一组标号对应很多值时，可悉数增加大于 1 行。这把数组从下标到值的单射关系，扩展成了自定义域变动的多值“函数”，大大增强了表现力。

这样的做的代价是列上有重复，但是对压缩算法来说，重复造成的空间浪费完全可控。

困惑：以代数理论组织数据够图灵奖?

- Edgar F. Codd，1981 年图灵奖得主

  Relational database: a practical foundation for productivity.

- Michael Stonebraker 2014 年图灵奖得主

  For fundamental contributions to the concepts and practices underlying modern database systems.

参考书

Hellerstein, Joseph M. and Michael Stonebraker. Readings in Database Systems.

把表格的概念通过数学公理的构建，论证其有效性并加以推广，是 E.~F.~Codd 在 1981 年图灵奖的划时代工作。高度概括起来就是一句话：一切都是关系，关系就是表格。

在数据格式中， CSV 和 HDF5 都可以表达表格。 CSV 能表达二维以内的数组， HDF5 能表示高维数组。关系代数要讨论数据以什么形式组织的存储。

正则表达式是一个很好的计算机子体系的范例，它有公理体系，使得不同学科人群有天然的共同出发点，从而促进了分工合作。计算机专家能设计出更强大的执行引擎，而用户则定制具体的使用方式。价值或者体现在用户以商品和服务形式购入引擎，或者体现在创新的引擎被公开发表。关系代数正是提供了类似的体系。

NumPy 把这个概念继续推广，允许复合数据类型本身是数组。

反思：如此简明的理论为何影响深远?

1. 数据组织在物理学家眼里，是一件微不足道的“小事”：如何组织都无所谓，能用就行
2. “自然”地采用树状结构：一个高能量的粒子产生很多“次数粒子”，次数粒子诱导出光子，一个光子被放大成 1 千万个电子
3. 争论：什么样的“树”最好?
   - 身分 vs. 能力：出身星系 vs. 天体种类
   - 身分 vs. 能力：母粒子 vs. 粒子种类
4. 每个人有自己关注的物理问题，因此争证持续，或者走向分裂
5. “都是那些个二货坚持用奇怪的数据格式”
6. 反思时，发现自己在“小事”上花了太多精力，非常空虚

研究这个问题是为了不再研究这个问题

- 按照简明的理论组织数据，从此之后忘掉它。
- 简明的理论有严格的数学体系
- 与正则表达式一样，可以有完美的社会分工
  - 使用者以关系代数的思想和语言描绘出自己要做什么
  - 研发者以天才的程序技巧实现关系代数的基本运算，优化复合运算

工业领域的某些反弹

- 2000 年左右，Web 界开始了去“关系数据库”的运动，开发 NoSQL 服务。
  - SQL 是 structured query language，关系数据库的基本语言
  - 关系数据库是以关系代数为基础的数据管理程序，有严格的数据规范
- NoSQL 运动的支持者认为，世界丰富多样，不应该受关系代数的限制。
  - Web 2.0 被作为商业概念炒作，从业人员四处革命
  - 关系代数让一切都成为了表格，大大限制了我们的想象力，剥夺了我们自由设计数据格式的权力
  - 于是大家开浪起来, 进行了一轮新的探索, 在这个过程中创造了一批卓越的分布式的数据系统
  - Google 的 MapReduce 也在此运动中，改变了整个大数据处理的格局。
- 但是，随着时间的推移，这些 NoSQL 的方案逐渐成熟，慢慢发展出关系代数的各种运算
  - 关系代数系统被重新发明一次，与分布式系统结合起来，成为数据时代的基石

13.3 SQL 语言

关系代数的实现

- SQL：是 structured query language 的缩写
  - 关系代数的天然语言，语法简明
  - 描述型的语言，描述把什么样的数据取出来
  - 也可以写数据，但一般是一次写，多次读
- 关系数据库 Relational Database，使用 SQL 语言描述
  - MariaDB (MySQL 的后继)
  - PostgreSQL
  - SQLite
- 统计软件分支，比如 S 语言中的 SAS/SPSS 中的 Dataset 或 DataFrame
  - GNU R 以及对应的 R 语言，是一个 S 语言的后继，其中 DataFrame 是语言的核心
  - Pandas 受 GNU R 的影响，目标是在 Python 的语言环境中实现 DataFrame 及其基本操作
- MapReduce 分布式大数据算法也受到关系代数的影响，一般都会使用关系代数作为平台高级接口
  - Hadoop 生态圈, Spark, etc.

所谓关系，即同一主体的多个参数之间的关系，即表格中的一行。在 NumPy 里对应复合数据类型，这些类型绑定在一起形成整体，使得表格在 NumPy 是一个逐行添加的一维数组。

SQL 描述性编程：定义清楚“我要什么”。

`CREATE` 创建表格

`INSERT` 插入行

`SELECT` 取得内容

SQL 关键字的大写约定

- 为了醒目地展现 SQLite 的关键字，我们约定它们全大写。

csvsql

来自 csvkit 的工具。

13.4 基本操作

SQL 基本操作

```
CREATE TABLE A(ID integer, name text);
INSERT INTO A VALUES(1, 'Wang');
INSERT INTO A VALUES(2, 'Li');
```

```
SELECT * FROM A;
```

ID	name
1	Wang
2	Li

Python 的 SQLite 模块

- 把 SQL 完整语句内嵌到函数调用中。

```
import sqlite3

c = sqlite3.connect("ra-python.db")
cur = c.cursor()
cur.execute("CREATE TABLE A(ID integer, name text)")
cur.execute("INSERT INTO A VALUES (1, 'Wang')")
cur.execute("INSERT INTO A VALUES (2, 'Li')")
cur.execute("SELECT * FROM A")
print(cur.fetchall()) # 读取 SQL 返回值
c.close() # 关闭
```

```
[(1, 'Wang'), (2, 'Li')]
```

- 计算机的本质：把一系列指令交给另一个工具执行。

Python 循环减少重复

```
import sqlite3

c = sqlite3.connect("ra-python.db")
cur = c.cursor()
cur.execute("CREATE TABLE A(ID integer, name text)")
for ID, name in ((1, 'Wang'), (2, 'Li')):
    cur.execute(f"INSERT INTO A VALUES ({ID}, {name})")
cur.execute("SELECT * FROM A")
print(cur.fetchall()) # 读取 SQL 返回值
c.close() # 关闭
```

13.5 集合与线性运算

集合操作

并 UNION

差 EXCEPT

交 INTERSECT

```
SELECT column_name(s) FROM table_name1
UNION
SELECT column_name(s) FROM table_name2;
```

笛卡尔积 SELECT * FROM A,B

投影 SELECT name FROM A

选择 SELECT name FROM A WHERE ID==1

再建新的表格

```
CREATE TABLE B(ID integer, name text);
INSERT INTO B VALUES(2, 'Li');
INSERT INTO B VALUES(3, 'Zhang');
```

```
SELECT * FROM B;
```

ID	name
2	Li
3	Zhang

并运算

```
SELECT * FROM B UNION
SELECT * FROM A;
```

ID	name
1	Wang
2	Li
3	Zhang

交运算

```
SELECT * FROM B INTERSECT
SELECT * FROM A;
```

ID	name
2	Li

笛卡尔积运算

```
SELECT * FROM A, B;
```

ID	name	ID	name
1	Wang	2	Li
1	Wang	3	Zhang
2	Li	2	Li
2	Li	3	Zhang

差运算

```
SELECT * FROM A
EXCEPT
SELECT * FROM B;
```

ID	name
1	Wang

投影

```
SELECT name FROM A;
```

name
Wang
Li

选择：where 代表选择条件

```
SELECT * FROM A WHERE ID==1;
```

ID	name
1	Wang

13.6 连接运算

关系代数的特色

```
SELECT * FROM A JOIN B ON A.ID=B.ID;
```

ID	name	ID	name
2	Li	2	Li

- 用笛卡尔积和选择

```
SELECT * FROM A, B WHERE A.ID==B.ID;
```

ID	name	ID	name
2	Li	2	Li

左连接

- 左连接，无条件保留左边

```
SELECT * FROM A LEFT JOIN B ON A.ID=B.ID;
```

ID	name	ID	name
1	Wang		
2	Li	2	Li

- SQLite 不支持右连接

```
SELECT * FROM B LEFT JOIN A ON A.ID=B.ID;
```

ID	name	ID	name
2	Li	2	Li
3	Zhang		

13.7 分组概括

GroupBy

```
INSERT INTO A VALUES(4, 'Li');
SELECT * FROM A;
```

ID	name
1	Wang
2	Li
4	Li

```
SELECT name,count(*) AS '人数' FROM A GROUP BY name;
```

name	人数
Li	2
Wang	1

13.8 课堂练习

成绩录入

天才少年爱迪生上课时做了一个梦，梦见自己成为了一门课的助教，协助老师向注册中心系统录成绩。但是少年爱迪生在梦境位面中的超能力减半，无法调用他强大的编程战斗值，所以他找到了你。

查看 CSV

```
csvlook dataframe-practice/students.csv | head
```

```
| 学号 | 姓名 | 性别 | 班级   |          联系方式 |
| -- | -- | -- | ---- | -------------- |
|  1 | AB | 女  | 物理71 | 50,626,922,811 |
|  2 | AC | 女  | 工物61 |  6,533,879,773 |
|  3 | AD | 男  | 物理71 | 43,865,400,582 |
|  4 | AE | 男  | 工物71 | 58,581,462,691 |
|  5 | AF | 女  | 物理72 | 45,017,508,911 |
|  6 | AG | 男  | 工物72 | 26,000,243,873 |
|  7 | AH | 男  | 物理71 |  4,295,424,729 |
|  8 | AI | 男  | 工物83 | 51,193,285,308 |
```

SQLite 读入 CSV

```
.mode csv
.import dataframe-practice/students.csv students
SELECT * FROM students LIMIT 5;
```

学号	姓名	性别	班级	联系方式
1	AB	女	物理 71	50626922811
2	AC	女	工物 61	6533879773
3	AD	男	物理 71	43865400582
4	AE	男	工物 71	58581462691
5	AF	女	物理 72	45017508911

- 也可直接使用 `dataframe-practice/people.db` 读入数据。

查看已经有的表格

```
.tables
```

A B students

- 任务：读入 `scores.csv` `classes.csv`

少年爱迪生想

- 比较物理系和工物系的男女比例
- 算出大家的总评成绩：
  - 小作业权相等，总体占 65% 的成绩
  - 大作业占 30% 的成绩
  - 划分出不同的分数段，给出某分数段同学的手机号
- 画出各班平均小作业成绩的变化曲线

物理系和工物系的男女比例

```
SELECT classes.院系, students.性别, count(*) AS 人数
FROM students JOIN classes
ON students.班级 = classes.班级
GROUP BY classes.院系, students.性别;
```

院系	性别	人数
工物	女	4
工物	男	18
物理	女	10
物理	男	25

选出特定成绩段的学生查询手机号

```
SELECT 联系方式,姓名,
(`curve.fitting`+`gpa.calculator`)/2*0.65 + `大作业`*0.3 AS total
FROM students JOIN scores ON students.学号 = scores.学号
WHERE total < 60;
```

联系方式	姓名	total
26000243873	AG	22.75
11373628062	AK	45.65
11391622912	AM	38.35
9693622361	AR	55.575
73349261755	AY	53.705
99263903250	BE	53.945
99555170920	BJ	58.66
26880267330	BS	36.665
77213592969	BV	47.65

13.9 长表与宽表

引例：小作业所有加和

```
.schema scores
```

```
CREATE TABLE `scores` (
  `学号` INTEGER,
  `self.intro` INTEGER,
  `a.b` INTEGER,
  `rank.guesser` INTEGER,
  `hdf5` INTEGER,
  `prime` INTEGER,
  `heart.curve` INTEGER,
  `gpa.calculator` REAL,
  `curve.fitting` INTEGER,
  `大作业` REAL
);
```

```
SELECT 联系方式, 姓名,
(`self.intro`+`a.b`+`rank.guesser`+hdf5+prime
+`heart.curve`+`curve.fitting`+`gpa.calculator`)/8*0.65
+ `大作业`*0.3 AS total
FROM students JOIN scores ON students.学号 = scores.学号
WHERE total < 60;
```

联系方式	姓名	total
26000243873	AG	42.575
11391622912	AM	58.0125

- 可以用双引号 " 代替斜引号 `

太丑了

- 四周 16 次作业全都写上吗?
- 不能写循环吗?
- 不能自动完成吗，一次原则在哪里?
- 问题何在?
  - 表格列的对称性。

长表

```
SELECT * FROM longscores LIMIT 5
```

学号	作业	分数	大作业
1	1	100.0	0
2	1	100.0	0
3	1	100.0	0
4	1	100.0	0
5	1	100.0	0

- “作业”一列给出作业编号
- “大作业”一列 0 代表非，1 代表是

再计算一次总评

- 用 `longscores` 分组大大简化

```
SELECT 学号, avg(分数) FROM longscores GROUP BY 学号 LIMIT 5
```

学号	avg(分数)
1	104.411111111111
2	92.3333333333333
3	94.2222222222222
4	104.033333333333
5	99.8333333333333

加上权重

- 0 到 0.65 权重， 1 到 0.3 权重
- `HAVING` 与 groupby 连用，对 group 进行过滤。

```
SELECT 学号, sum(权重*分数) AS 总评 FROM
(SELECT 学号, CASE 大作业
  WHEN 0 THEN 0.65
  WHEN 1 THEN 0.3
  END AS 权重, avg(分数) AS 分数
  FROM longscores GROUP BY 学号, 大作业)
GROUP BY 学号
HAVING 总评 > 105
```

学号	总评
49	105.16375
55	105.70625

储存起来用于下一步使用

- CREATE VIEW 把中间结果存成虚拟表格

```
CREATE VIEW final_scores AS
SELECT 学号, sum(权重*分数) AS 总评 FROM
(SELECT 学号, CASE 大作业
  WHEN 0 THEN 0.65
  WHEN 1 THEN 0.3
  END AS 权重, avg(分数) AS 分数
  FROM longscores GROUP BY 学号, 大作业)
GROUP BY 学号
```

```
SELECT * FROM final_scores LIMIT 5
```

学号	总评
1	104.50375
2	87.20625
3	94.05625
4	103.265
5	98.26875

算出大家的总评成绩

```
SELECT students.学号, 总评, 联系方式
FROM final_scores JOIN students
ON final_scores.学号 = students.学号
WHERE 总评 > 105
```

学号	总评	联系方式
49	105.16375	81920572715
55	105.70625	24891470639

总评成线关联班级

```
SELECT total.班级,院系,姓名,总评,联系方式 FROM classes JOIN
(SELECT 姓名,班级,总评,联系方式 FROM final_score
JOIN students on final_score.学号 = students.学号 WHERE 总评 < 60) AS total
ON total.班级 = classes.班级
```

班级	院系	姓名	总评	联系方式
工物 72	工物	AG	42.575	26000243873
物理 71	物理	AM	58.0125	11391622912

成绩按班级计算

- ☒ 比较物理系和工物系的男女比例
- ☒ 算出大家的总评成绩：
  - ☒ 小作业权相等，总体占 65% 的成绩
  - ☒ 大作业占 30% 的成绩
  - ☒ 划分出不同的分数段，给出某分数段同学的手机号
- ☐ 画出各班平均小作业成绩的变化曲线

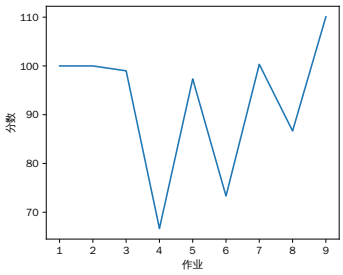
取得班级作业平均数据

```
SELECT 班级, 作业, avg(分数) AS 分数
FROM longscores JOIN students ON longscores.学号 = students.学号
GROUP BY 班级, 作业 LIMIT 10
```

班级	作业	分数
基科 71	1	100.0
基科 71	2	100.0
基科 71	3	99.0
基科 71	4	66.6666666666667
基科 71	5	97.3333333333333
基科 71	6	73.3333333333333
基科 71	7	100.333333333333
基科 71	8	86.6666666666667
基科 71	9	110.066666666667
基科 72	1	100.0

绘图

```
from matplotlib import pyplot as plt
import pandas as pd
import sqlite3

c = sqlite3.connect("dataframe-practice/people.db")
class_score = pd.read_sql_query("""
select 班级, 作业, avg(分数) as 分数
from longscores join students
on longscores.学号 = students.学号
group by 班级, 作业
""", c)

one_line = class_score.query("班级 == '基科 71'")
plt.plot(one_line["作业"], one_line["分数"])
plt.xlabel("作业")
plt.ylabel("分数")
```



长表与宽表总结

- 长表，为了恢复列之间的对称性，对计算机系统更友好
- 宽表，某些时候人类易于理解。
  - 不同的列有不同的属性，例如权重
  - 相同属性的列在重复，违反一次原则

第十四章　DataFrame

14.1　复习与提示

安装软件 R 的 ggplot2 与 dbplyr

Debian

```
apt install r-cran-ggplot2 r-cran-dbplyr r-cran-dplyr r-cran-reshape2 r-cran-rsqlite
```

最佳工具

- Dataframe 流派的关系代数系统 全局最佳工具是 GNU R
  - Python 只是小圈子的最佳工具
- Grammar of graphics 的全局最佳工具是 R ggplot2
  - seaborn 只是眼前的妥协，未来 Python 小圈子的最佳工具可能是 plotnine

关系代数复习

- 集合运算：$\cap, \cup, \setminus$
  - SQL `INTERSECT` `UNION` `EXCEPT`
- 线性运算：$\otimes, \Pi, \sigma$
  - SQL `SELECT` `WHERE`
- 关系运算：连接 $\bowtie$
  - SQL `JOIN`
- 拓展运算：GroupBy（分组） $\mathcal{G}$
  - SQL `GROUP BY`

[66%] 少年爱迪生想

☒ 比较物理系和工物系的男女比例

☒ 算出大家的总评成绩:

- ☒ 小作业权相等，总体占 65% 的成绩
- ☒ 大作业占 30% 的成绩
- ☒ 划分出不同的分数段，给出某分数段同学的手机号

☐ 画出各班平均小作业成绩的变化曲线

```
SELECT 学号, sum(权重*分数) AS 总评 FROM
(SELECT 学号, CASE 大作业
  WHEN 0 THEN 0.65
  WHEN 1 THEN 0.3
  END AS 权重, avg(分数) AS 分数
  FROM longscores GROUP BY 学号, 大作业)
GROUP BY 学号
HAVING 总评 > 105
```

学号	总评
49	105.16375
55	105.70625

14.2 DataFrame

DataFrame

由 R 语言提出的数据表格形式：与 SQL 一样都脱胎于关系代数思想

SQL 的区别

- SQL 是描述性语言，与函数式编程更接近。
  - 生成的表格不能随意改动。改动只是通过创建新表格实现。
- DataFrame 是动态结构
  - 可以更灵活地更改;
  - 但过于灵活可能带来调试上的麻烦。
- SQL 可以处理大于内存容量的数据， DataFrame 必须借助 Spark 等大数据平台才能实现。

GNU R 语言

- R 是著名的统计学语言，应用广泛
- 起源于 1970 年代的 S 语言
  - 在 fortran 库的基础上交互地处理数据进行统计分析
    调用 fortran 库提供交互环境是 S, Matlab 和 Numpy 的共同起源
  - 提供交互式的绘图工具，以便于快速迭代
  - 受 LISP 语言的影响
  - DataFrame 概念影响了几乎所有统计工具
    例如 S-PLUS, SAS, SPSS 应用广泛
- GNU R 重新使用 Scheme 实现了 S 语言，并以自由软件形式发布
  - 自由软件和开源运动是为科学研究提供了可 复现 的工具

为什么使用 R

- 大部分统计学算法都有 R 语言的实现
  站在数理统计学天才们的肩膀上。
- 非常适合快速试验并找到统计方法上的 最佳工具

`dplyr` 数据整理的利器，简洁的语法表达关系代数

`dbplyr` 与 SQL 浑然连成一体，统一 DataFrame 与 SQL

`ggplot2` 科学制图利器，易用性远好于 matplotlib

数据类型

呼叫传送门

numeric 数值型：分成整型、浮点型和复数型

character 字符型

factor 枚举型

```
# 获得帮助
help(help)
help(numeric)
```

数值型

- 数组
- 数组的索引
- 字符型
- 字符串操作
- 布尔型
- 枚举型

程序结构

选择结构

循环结构

函数

各类概率分布

- 正态分布

DataFrame 用法

- 简单制图
- 线性回归
- 枚举类型

DataFrame 里的关系代数基本操作

- 集合基本运算：交、并、差
- 线性代数运算：笛卡尔积、投影、选择
- 连接
- GroupBy, SortBy

14.3 关系代数制图

图形语法 Grammar of Graphics

- 在表格与图形要素之间建立映射关系
- 表格对称的同等层次的列，都可以映射到各类图形要素中
  1. x 轴坐标
  2. y 轴坐标
  3. 颜色
  4. 点形状（圈、三角、方块等）
  5. 线形状（实线、虚线、点划线等）
  6. 点的大小
  7. 线的粗细
  8. 透明度
  9. 子图位置
- 在绘图时，通过调整映射关系来找到最佳的展现方式

参考书

- Leland Wilkinson, The Grammar of Graphics （沉痛缅怀 Wilkinson 教授）
- Hadley Wickham, A Layered Grammar of Graphics

ggplot

呼叫传送门 `Merge-GroupBy.slides.html`

```
require(ggplot2)
data(mtcars)
p <- ggplot(mtcars, aes(x=wt, y=hp)) + geom_point()
print(p)
```

```
p <- p + aes(color=am)
print(p)
```

ggplot 枚举型制图

```
mtcars$am <- factor(mtcars$am, levels=c(0, 1), labels=c("automatic", "manual"))
p <- p %+% mtcars
print(p)
```

14.4 宽表到长表

表格的等价变换

呼叫传送门 Merge-GroupBy.slides.html

- R 和 Pandas DataFrame 可以将长表与宽表相互转换
  - SQLite 不具备此功能
- melt 函数

melt

- id_vars 指定作为索引的列
- value_vars 指定作为数值的列
- var_name 指定长表中组变量的列名
- value_name 指定长表中值的列名

统计班级平均分

[100%] 少年爱迪生想

☒ 比较物理系和工物系的男女比例

☒ 算出大家的总评成绩：

- ☒ 小作业权相等，总体占 65% 的成绩
- ☒ 大作业占 30% 的成绩
- ☒ 划分出不同的分数段，给出某分数段同学的手机号

☒ 画出各班平均小作业成绩的变化曲线

14.5 R 版总评计算

课堂练习

14.6 dbplyr

SQL 的 DataFrame 接口

```
library(dplyr)
library(dbplyr)
con <- DBI::dbConnect(RSQLite::SQLite(), dbname = "dataframe-practice/people.db")
classes <- tbl(con, "classes")

classes %>% filter(院系==' 物理')
```

基科 71 物理
物理 72 物理
物理 71 物理
基科 72 物理

查看 SQL 语句

```
sql_render(classes %>% filter(院系=='物理'))
```

```
<SQL> SELECT *
FROM `classes`
WHERE (`院系` = '物理')
```

14.7 后备资料

关系代数

- 关系：$\{(r,s)|r \in R, s \in S\}$
- 关系代数：在集合基础上定义关系运算的封闭系统
  - 封闭系统：运算作用于一个或多个关系上来生成一个关系
- 围绕关系代数设计的关系数据库是存储海量数据的标准
  - 代表：Structure Query Language (SQL) 语言
- 关系代数的思想具有一般性： 管理、添加和分析数据

直观理解：一切都是表格

Event	Channel	Time	Weight
0	0	1	1.1
0	0	1.1	1.15
0	2	1.2	1.3
1	3	0.8	0.9

Event	Channel	Wave
0	0	[0,0.1,...,0]
0	2	[0,0.2,...,0]

基本动机

- 关系代数设计师 Todd，图灵奖工作
- 数据都应该自我描述
  - 即使数据的存储形式变了，对程序进行操作的程序也不应该改变
  - 反例：链表

  - 反例：随意写成的 Excel 表格

- 方便扩展到大规模的数据库中

实用价值

- 引擎优化与应用分工
- 引擎：自动 out-of-core computing (超出内存的运算)
- 引擎：自动并行计算

第十五章　回归分析

15.1　复习提示

数据集下载

- `lm-examples.tar.gz` 线性回归样例集

```
wget http://hep.tsinghua.edu.cn/~orv/pd/lm-examples.tar.gz
tar -xf lm-examples.tar.gz
```

安装 rhdf5

```
# Debian
apt install r-bioc-rhdf5
# Gentoo @ macOS
emaint sync -r R_Overlay
emerge -vt sci-BIOC/rhdf5
```

DataFrame

- DataFrame （数据表格） 与 SQL 一样，是关系代数的具体实现
  - DataFrame 更动态，可联用更多丰富的语言功能
- GNU R 语言是统计学领域的编程语言
  - 提供了跨领域的与统计学高效的对话方式

ggplot2

- 把数据表格的列映射到图中的要素
  - 例如 x 轴、 y 轴、颜色等

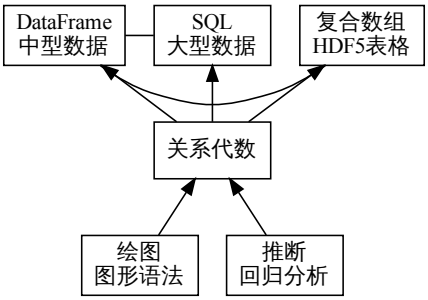
```
ggplot(mtcars, aes(x=wt, y=hp)) + geom_point()
```

dplyr 与 dbplyr

- 关系代数的基本操作对应到命令

```
students %>% inner_join(longscores) %>%
  group_by(班级，作业) %>% summarise(平均分=mean(分数))
```

关系代数工具组



- 以关系代数的数据形式为核心
  - 实现列的对称性
- 工具设计变得简单、工具使用变得简单
  - 一行代码绘图
  - 一行代码回归
  - 类比：一行命令实现一个功能
  - 一次原则：只输入必要信息
  - 保证数据的透明：更易于理解
- 人类更能专注于高层次的概念与问题，不被细节湮没

推论：数据分析最佳工具策略

- 原始数据尽快等价变换成关系代数形式：站在统计学与图形学巨人的肩膀上

15.2 线性回归

概念

- 复习：线性回归，最小二乘法

Y	x
2	1
5.5	3
6.5	4
9	7

- $Y = a + bx + \epsilon$
- (Y, x) 是关系，组成表格
- 表格 DataFrame 之上自然进行回归分析

读入例子：书的重量

```
books = read.csv("lm-examples/books.csv",
                 row.names=1)
books
```

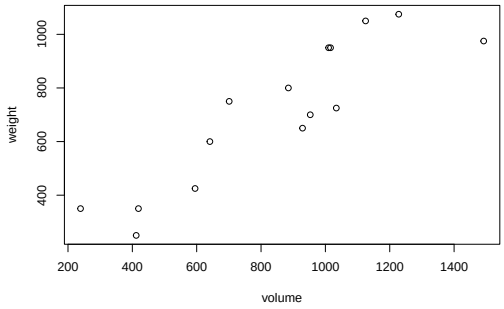
```
   volume area weight cover
1     885  382    800    hb
2    1016  468    950    hb
3    1125  387   1050    hb
4     239  371    350    hb
5     701  371    750    hb
6     641  367    600    hb
7    1228  396   1075    hb
8     412    0    250    pb
9     953    0    700    pb
10    929    0    650    pb
11   1492    0    975    pb
12    419    0    350    pb
13   1010    0    950    pb
14    595    0    425    pb
15   1034    0    725    pb
```


```
   volume area weight cover
1     885  382    800    hb
2    1016  468    950    hb
3    1125  387   1050    hb
4     239  371    350    hb
5     701  371    750    hb
6     641  367    600    hb
7    1228  396   1075    hb
8     412    0    250    pb
9     953    0    700    pb
10    929    0    650    pb
11   1492    0    975    pb
12    419    0    350    pb
13   1010    0    950    pb
14    595    0    425    pb
15   1034    0    725    pb
```


- `hb` 代表 hard back，硬质纸壳封面封底
- `pb` 代表 paper back，软纸封面封底
- `area` 代表封面的面积
- `volume` 代表书的页数

绘制重量与页数的关系

```
plot(weight ~ volume, books)
```



线性回归

```
lm.books = lm(weight ~ volume, books)
summary(lm.books)
```

```
Call:
lm(formula = weight ~ volume, data = books)

Residuals:
    Min      1Q  Median      3Q     Max
-189.97 -109.86   38.08  109.73  145.57

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 107.67931   88.37758   1.218    0.245
volume        0.70864    0.09746   7.271 6.26e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 123.9 on 13 degrees of freedom
Multiple R-squared:  0.8026,Adjusted R-squared:  0.7875
F-statistic: 52.87 on 1 and 13 DF,  p-value: 6.262e-06
```

拟合结果解读

- $Y = a + bx + \epsilon$
  - $\hat{a} = 108$ 不显著
  - $\hat{b} = 0.71$ 显著
- 回归方程为

  $Y = 0.71x + \epsilon$
- 显著度
  - $\hat{a}$ 和 $\hat{b}$ 都服从 t-分布

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 107.67931   88.37758   1.218    0.245
volume        0.70864    0.09746   7.271 6.26e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

F-检验

$$H_0 : b = 0, H_1 : b \neq 0$$

$$\underbrace{\sum_{i=1}^{n}(y_i - \overline{y})^2}_{S_T, \nu_T = n-1} = \underbrace{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}_{S_E : \nu_E = n-p} + \underbrace{\sum_{i=1}^{n}(\overline{y} - \hat{y}_i)^2}_{S_M : \nu_M = p-1} - \underbrace{\sum_{i=1}^{n}(y_i - \hat{y}_i)(\overline{y} - \hat{y}_i)}_{0}$$

S_M 模型变差。进行线性回归的预测值与 Y 样本均值的差异

p 回归参数个数，一元线数回归取 2

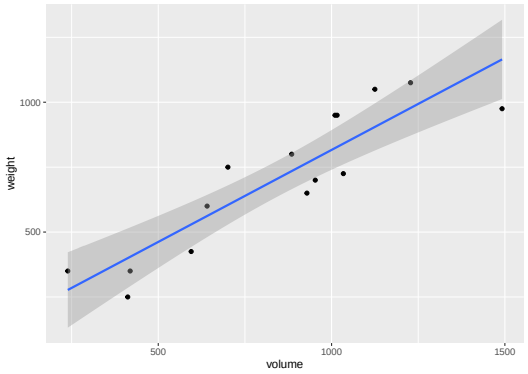
$$\frac{S_M/1}{S_E/(n-2)} = F \sim F(1, n-2)$$

$$R^2 = \frac{S_M}{S_T}$$

```
Multiple R-squared:  0.8026,Adjusted R-squared:  0.7875
F-statistic: 52.87 on 1 and 13 DF,  p-value: 6.262e-06
```

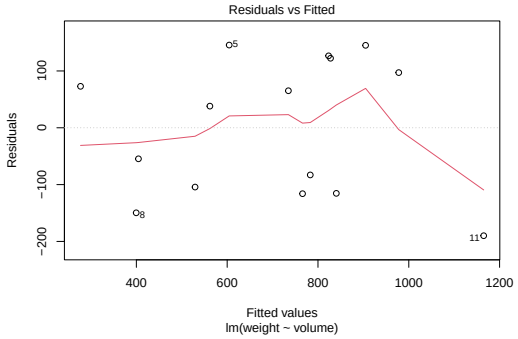
拟合效果图

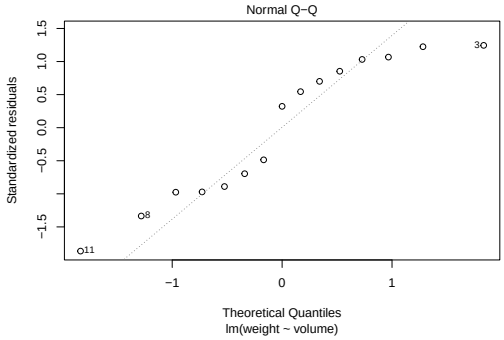
```
library(ggplot2)
p = ggplot(books, aes(x=volume, y=weight)) + geom_point()
print(p + geom_smooth(method="lm"))
```



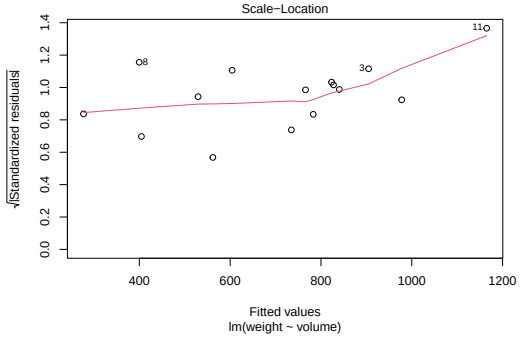
拟合优度的制图判定

```
plot(lm.books)
```

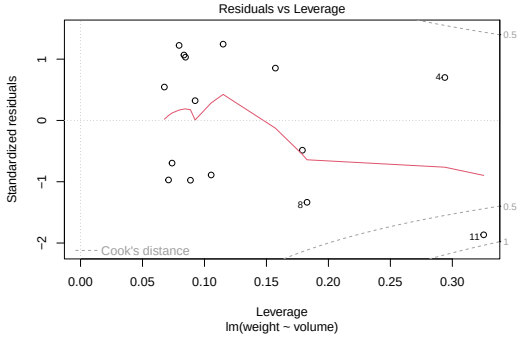




平移缩放关系



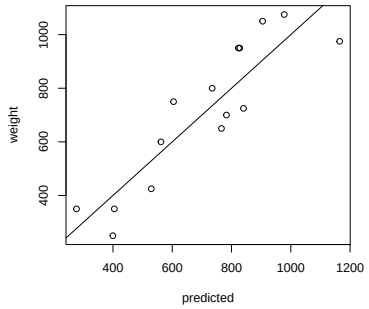
标准化的残差图



Cook scaling-location 理论残差图

回归预测

```
books$predicted = predict(lm.books)
plot(weight~predicted, books)
abline(0, 1)
```



模型矩阵

- 细致描述变量的依赖关系，扩展公式的表达。使用 `model.matrix` 厘清内部机理

```
model.matrix(weight ~ volume, books)
```

```
   (Intercept) volume
1            1    885
2            1   1016
3            1   1125
4            1    239
5            1    701
6            1    641
7            1   1228
8            1    412
9            1    953
10           1    929
11           1   1492
12           1    419
13           1   1010
14           1    595
15           1   1034
attr(,"assign")
[1] 0 1
```

双变量回归情形

```
model.matrix(weight ~ volume + cover, books)
```

```
   (Intercept) volume coverpb
1            1    885       0
2            1   1016       0
```

```
3            1   1125       0
4            1    239       0
5            1    701       0
6            1    641       0
7            1   1228       0
8            1    412       1
9            1    953       1
10           1    929       1
11           1   1492       1
12           1    419       1
13           1   1010       1
14           1    595       1
15           1   1034       1
attr(,"assign")
[1] 0 1 2
attr(,"contrasts")
attr(,"contrasts")$cover
[1] "contr.treatment"
```

去除截距情形

```
model.matrix(weight ~ volume + cover - 1, books)
```

```
   volume coverhb coverpb
1     885       1       0
2    1016       1       0
3    1125       1       0
4     239       1       0
5     701       1       0
6     641       1       0
7    1228       1       0
8     412       0       1
9     953       0       1
10    929       0       1
11   1492       0       1
12    419       0       1
13   1010       0       1
14    595       0       1
15   1034       0       1
attr(,"assign")
[1] 1 2 2
attr(,"contrasts")
attr(,"contrasts")$cover
[1] "contr.treatment"
```

去除截距拟合

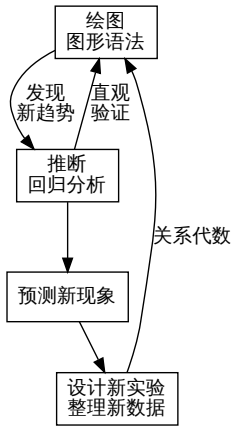
```
lm3.books = lm(weight ~ volume + cover - 1, books)
summary(lm3.books)
```

```
Call:
lm(formula = weight ~ volume + cover - 1, data = books)

Residuals:
    Min      1Q  Median      3Q     Max
-110.10  -32.32  -16.10   28.93  210.95

Coefficients:
```

```
         Estimate Std. Error t value Pr(>|t|)
volume    0.71795    0.06153  11.669  6.6e-08 ***
coverhb 197.96284   59.19274   3.344  0.00584 **
coverpb  13.91557   59.45408   0.234  0.81889
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 78.2 on 12 degrees of freedom
Multiple R-squared:  0.9914,Adjusted R-squared:  0.9892
F-statistic: 459.5 on 3 and 12 DF,  p-value: 1.207e-12
```

15.3 探索性分析

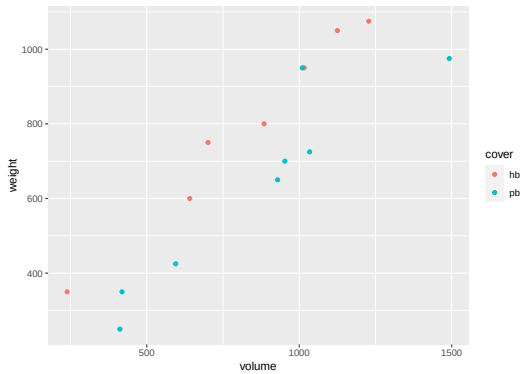
探索性分析



填加自变量

- 书的包装有什么影响？画图探索

```
print(p + aes(color=cover))
```



多线性回归

- 线性回归可以推广到多个变量

还可以包含离散变量

- 每个离散变量，实际上对应多个自由参数，个数等于离散变量的取值数

```
books$cover = as.factor(books$cover)
books[c("weight", "volume", "cover")]
```

```
   weight volume cover
1     800    885    hb
2     950   1016    hb
3    1050   1125    hb
4     350    239    hb
5     750    701    hb
6     600    641    hb
7    1075   1228    hb
8     250    412    pb
9     700    953    pb
10    650    929    pb
11    975   1492    pb
12    350    419    pb
13    950   1010    pb
14    425    595    pb
15    725   1034    pb
```

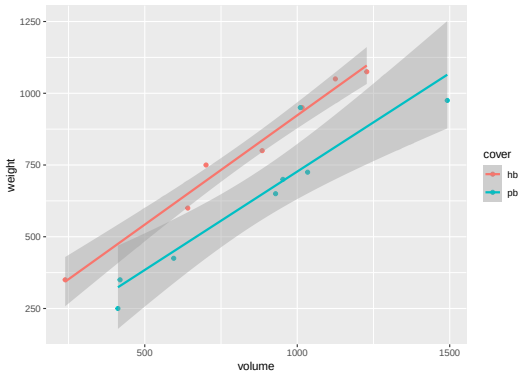
多线性回归

```
lm2.books = lm(weight ~ volume + cover, books)
summary(lm2.books)
```

```
Call:
lm(formula = weight ~ volume + cover, data = books)

Residuals:
    Min      1Q  Median      3Q     Max
-110.10  -32.32  -16.10   28.93  210.95

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  197.96284   59.19274   3.344 0.005841 **
volume         0.71795    0.06153  11.669  6.6e-08 ***
coverpb     -184.04727   40.49420  -4.545 0.000672 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 78.2 on 12 degrees of freedom
Multiple R-squared:  0.9275,Adjusted R-squared:  0.9154
F-statistic: 76.73 on 2 and 12 DF,  p-value: 1.455e-07
```

多线性回归结果图

```
print(p + aes(color=cover) + geom_smooth(method="lm"))
```



15.4 模型选择

Akaike's An Information Criterion

- 赤池信息判据
  1. 应当用多少个变量描述模型?
  2. 变量越多，拟合越精确
  3. 变量越少，模型越简明

$$\mathrm{AIC} = 2n - 2\log \mathcal{L}$$

其中 n 是变量个数，$\mathcal{L}$ 是拟合似然函数 (likelihood)

Von Neumann's elephant

With four parameters I can fit an elephant, and with five I can make him wiggle his trunk.

R 下的 AIC

```
AIC(lm.books) # weight ~ volume
AIC(lm2.books) # weight ~ volume + cover
AIC(lm3.books) # weight ~ volume + cover - 1
```

```
[1] 191.016
[1] 177.9986
[1] 177.9986
```

- 可见 lm2 与 lm3 等价
  - 因为两者的 `model.matrix` 可用通过线性变换等价互换

改进模型的灵感

- 回归结果说明 paper back （纸皮）不应该有截距，只有 hard back （硬皮） 才有截距。
  - 使用公式已经无法表达这个需求

```
Coefficients:
         Estimate Std. Error t value Pr(>|t|)
volume    0.71795    0.06153  11.669  6.6e-08 ***
coverhb 197.96284   59.19274   3.344  0.00584 **
coverpb  13.91557   59.45408   0.234  0.81889
```

使用 `model.matrix` 改进模型

- 直接修改 `model.matrix`

课堂练习

- 分析：有否必要把 `area` 加入回归?

试验使用 `area`

```
lm5.books = lm(weight ~ volume + area - 1, books)
summary(lm5.books) # AIC = 175.9741
```

```
Call:
lm(formula = weight ~ volume + area - 1, data = books)

Residuals:
    Min      1Q  Median      3Q     Max
-112.53  -28.73  -10.52   24.62  213.80

Coefficients:
       Estimate Std. Error t value Pr(>|t|)
volume  0.72891    0.02767  26.344 1.15e-12 ***
area    0.48087    0.09344   5.146 0.000188 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 75.07 on 13 degrees of freedom
Multiple R-squared:  0.9914,Adjusted R-squared:  0.9901
F-statistic: 747.9 on 2 and 13 DF,  p-value: 3.799e-14
```

15.5 广义线性回归

概念

- 线性回归中，假设了残差服从正态分布
- $f(E[Y|X]) = bX$
  - Y 的期望经过连接函数 $f(y)$ 与 X 是线性关系
- 可以在保持高效计算的前提下，把这两个条件放宽，可以大大扩展线性模型的适用范围

种类

- 泊松回归
- 二项回归 (生存回归, cloglog 回归)
- 伽马回归
- Tweedie 回归

课堂任务

- 尝试对 `r10500.h5` 进行泊松回归
- 辅助文件 `geo.csv` , `pmtdata.csv`

```
PE = pd.read_hdf("lm-examples/r10500.h5", "PETruth")
PEs = PE.groupby(["EventID", "ChannelID"]).count().reset_index()
PMT = pd.read_csv("lm-examples/pmtdata.csv", header=0, delimiter=" ",
                  names=["ChannelID", "type", "QE"])
# 不能种类的 PMT , 预期的信号计数是否一致
type_count = pd.merge(PEs, PMT)
y, X = patsy.dmatrices("PETime ~ type - 1", type_count)
pois_res = sm.GLM(y, X, sm.families.Poisson()).fit()
```

泊松回归拟合结果

```
print(pois_res.summary())
```

```
                 Generalized Linear Model Regression Results
==============================================================================
Dep. Variable:                 PETime   No. Observations:               549654
Model:                            GLM   Df Residuals:                   549650
Model Family:                 Poisson   Df Model:                            3
Link Function:                    Log   Scale:                          1.0000
Method:                          IRLS   Log-Likelihood:            -3.9571e+06
Date:                Wed, 03 Aug 2022   Deviance:                   5.7277e+06
Time:                        17:12:27   Pearson chi2:                 7.26e+06
No. Iterations:                     7   Pseudo R-squ. (CS):              1.000
Covariance Type:            nonrobust
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
type[HZC]          0.3697      0.002    197.609      0.000       0.366       0.373
type[Hamamatsu]    3.3682      0.001   5738.410      0.000       3.367       3.369
type[HighQENNVT]   3.3916      0.000   8503.298      0.000       3.391       3.392
type[NNVT]         3.2986      0.001   3460.998      0.000       3.297       3.300
==============================================================================
```

15.6 总结

模型的选择

1. 关系代数让回归分析变得极其直观，让我们专注于问题的本质
2. 模型的选择极其重要，应当使用客观标准
   - AIC 是最简单直接的客观标准
   - 找到“最佳模型”的过程充满曲折，是实验“研究”的主体过程
3. 广义线性回归，把误差分布从高斯替换为其它指数族分布
   - 把连接函数从恒等替换为非线性函数
   - 几乎可以解决所有日常工作中的非线性问题，惊喜！

15.7 无脑回归

可用 sklearn

- Debian

```
apt install python3-sklearn-pandas
```

- macOS @ Gentoo

```
pip3 install sklearn-pandas
```

支持向量机

```
### SVM regression
from sklearn.svm import SVR
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
regr = make_pipeline(StandardScaler(), SVR(C=1.0, epsilon=0.2))
regr.fit(X, y)
```

决策树

```
### GBoost
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, random_state=0)
reg = GradientBoostingRegressor(random_state=0)
reg.fit(X_train, y_train)
reg.predict(X_test[1:2])
reg.score(X_test, y_test)
```

神经网络

```
### 神经网络
from sklearn.neural_network import MLPRegressor
regr = MLPRegressor(random_state=1, max_iter=500).fit(X_train, y_train)
regr.predict(X_test[:2])
regr.score(X_test, y_test)
```

第十六章　大作业与未来方向

16.1　复习

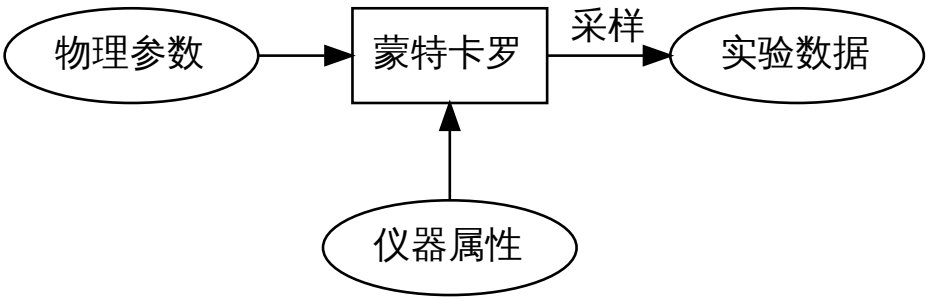
广义线性回归

1. 关系代数让回归分析变得极其直观，让我们专注于问题的本质
2. 模型的选择极其重要，应当使用客观标准
    - AIC 是最简单直接的客观标准
    - 找到“最佳模型”的过程充满曲折，是实验“研究”的主体过程
3. 广义线性回归，把误差分布从高斯替换为其它指数族分布
    - 把连接函数从恒等替换为非线性函数
    - 几乎可以解决所有日常工作中的非线性问题，惊喜！

16.2　大作业

大作业安排

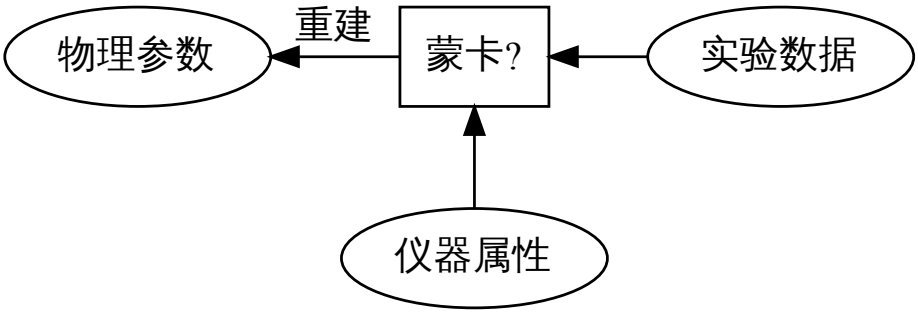
正向第一阶段



- 2023-07-21 – 2023-08-10

- 模拟实验测量

逆向第二阶段



- 2023-08-03 – 2023-08-24
- 分析实验数据
- 测量物理模型参数
- 发现物理规律
- 黑盒分数按排名
  - 在不同大作业之间归一化

实验测量的分析

输入 （模拟的）实验测量原始数据

- 如果不是模拟的，则无法评分
- 但是助教会尽可能把它做得和真的一样

输出 物理对象的信息

采分 与助教手中的模拟输入相比

第二阶段分组

- 同学们先联络好，组队信息在网络学堂提交
  - 到 `https://physics-data.meow.plus` 注册账号，一并提交

  - 可能需要重新组队

- 每队至多三人
  - 单人队：大作业得分 $\times 1.03$
  - 三人队：每人大作业得分 = 队伍得分 $\times 0.95$
  - 不同队伍间请勿直接交换代码
- 如果大作业结果含有学术突破，总评保送 A+。

`https://physics-data.meow.plus` 的使用

1. 注册账号：注意邮箱的“垃圾箱”等，如果收到不到注册邮件尝试更换邮箱
   - 遇到困难开 issue 提问
2. 找到大作业的位置

   注意平台上还有很多其它的作业，不要走错了
   - `https://physics-data.meow.plus/challenges/pd2023-gamma`
   - `https://physics-data.meow.plus/challenges/pd2023-muon`
   - `https://physics-data.meow.plus/challenges/pd2023-spectroscopy`
3. 前往 ”submissions” 选项卡，点击 ”CREATE SUBMISSION”
   - 上传你的解答文件。

大作业的可复现性

复现 原则的要求

提交到 `https://physics-data.meow.plus` 的结果必须可复现，否则无效。

思路和要点

- 组队完成后，将获得

  `https://git.tsinghua.edu.cn/physics-data/2023/project_2`

  之下的仓库一份，使用 GNU Make 构建整个分析流程，连同报告、程序整理到仓库中。
  - 注意小组分工中 Git 使用的规范
  - 善于使用 Git `branch`, Gitlab `merge request` 等团队协作功能
- 把流程系统化成输入、输出与过程三要素。
  - 而向数据编程，data-driven programming
- 系统表达输入数据、输出数据和中间结果的依赖关系，
  - 成为“可执行的说明文档”

具体说明

gamma 刘学伟

- 粒子物理方向：台山中微子实验的成像原理探索
- Ghost Hunter 2023 课赛结合

muontagging 刘明昊

- 宇宙射线缪子的测量与屏蔽

spectroscopy 吴致颉

16.3 总复习

黑客技能

科学数据处理的原则
黑客的审美： 复现 透明 一次 最佳工具

版本控制
Git 与队友分工协作，与明天的自己协作
Git 是“搬砖工地安全帽”，无头盔禁止上岗

关系代数
数据表示成关系，数据的操作表示成关系代数运算

数据格式
透明 CSV, HDF5, JSON, 数据库 SQL

黑客技能（二）

数据流水线
GNU Make 管理数据的依赖与转换，实现错误恢复和并行计算
实现数据层次的 Python/R/Bash/Scheme/SQL 多语言融合

正则表达式
描述字符串的微型语言，数学模型

命令环境
POSIX 环境中强大的小工具组合，开发与使用相融合

计算语言
Python 语法友好，工具丰富，统领 C/C++/Fortran/R/SQL 库

16.4 现实世界的大数据方法

永远留在古老的计算环境

- SuperK 质子衰变和中微子实验，XMASS 暗物质实验
- 问题
  1. 数据处理技术发展停滞，违反“最佳工具”原则
  2. 新成员需要花精力学习旧技术，人力浪费
- 解决方案：test-driven development

自制 Python 驱动的批量处理流水线

- 症状问题
  1. 使用 Python 调用大量 shell 命令，程序可读性差，违反“最佳工具”原则

```
for name in args.name:
  if name not in productions:
    print('Unknown production: ' + name)
    continue
  for script in productions[name]:
    parg = arguments + ' --name %s ' % name
    print('python %s %s' % (script, parg))
    os.system('python %s %s' % (script, parg))
```

  2. 流水线验证 flag 文件是否存在来确定是否成功执行，误判多，难以 debug
- 解决方案：使用 GNU Make 构建流水线
  - `make` 默认使用 `/bin/sh` 执行命令。 `SHELL=/bin/sh`
  - 把 `SHELL` 换成提交任务给超级计算机集群的脚本

```
SHELL=lsf
export MAKE_TARGET=$@
export MAKE_SOURCE=$^
```

`make` 对接集群的脚本

```
#!/home/jinping/gentoo/bin/bash -e
# Platform LSF wrapper to be used as GNU Make shell.

# GNU Make convention for the first argument.
[[ ${1} = '-c' ]] && shift

for j in ${MAKE_SOURCE}; do
    [[ -z $(bjobs -J ${j}) ]] && continue
    DEP+=" && done(${j})" # 把 make 中的依赖关系传递给调度系统
done
```

```
cat << EOF > ${MAKE_TARGET}.sh
#!/home/jinping/gentoo/bin/bash
$@
EOF

chmod +x ${MAKE_TARGET}.sh

bsub -q normal"${DEP}" -J ${MAKE_TARGET} -o ${MAKE_TARGET}.log ${MAKE_TARGET}.sh
```

16.5 讲义

建设状况

- https://git.tsinghua.edu.cn/physics-data/lecture

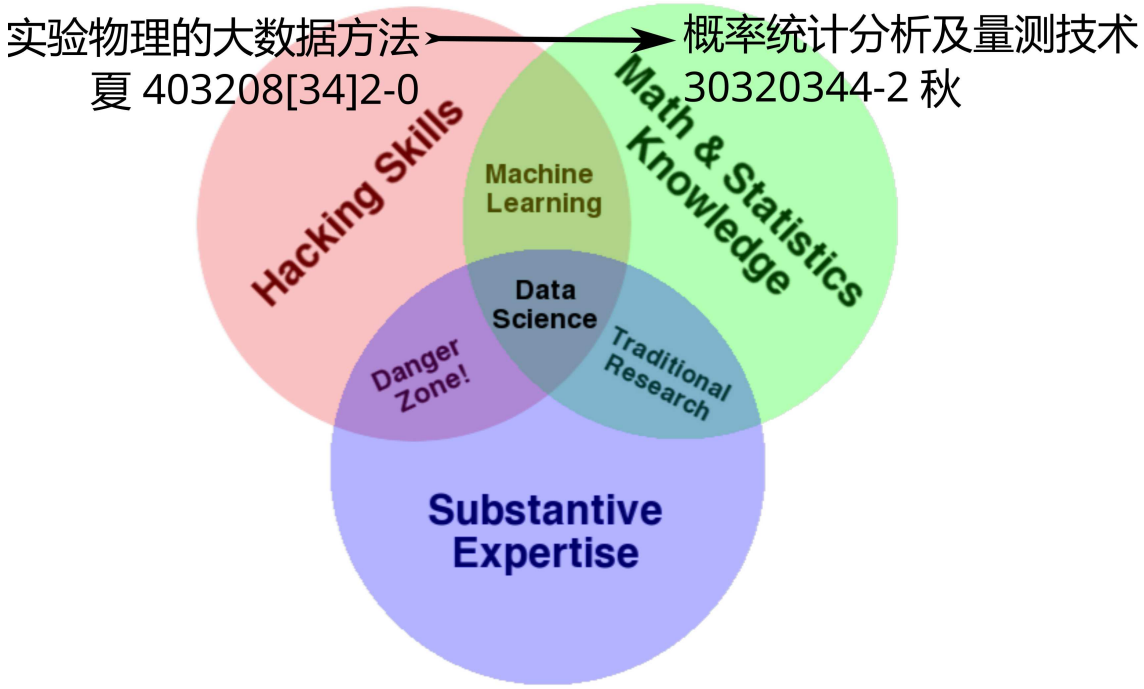
```
head -q -n1 lecture/l*.org
```

```
#+TITLE: 第一讲 实验物理的大数据方法总论 DONE
#+TITLE: 第二讲 Python基础 DONE
#+TITLE: 第三讲 复合类型与函数 DONE
#+TITLE: 第四讲 Python 模块 DONE
#+TITLE: 第五讲 数组 DONE
#+Title: 第六讲 数据格式 DONE
#+Title: 第七讲 数据绘图 TODO
#+Title: 第八讲 蒙特卡罗方法与大作业 TODO
#+Title: 第九讲 GNU 命令行 TODO
#+Title: 第十讲 GNU Make 数据生产线 TODO
#+Title: 第十一讲 正则表达式 TODO
#+Title: 第十二讲 bash 脚本 TODO
#+Title: 第十三讲 关系代数 TODO
#+Title: 第十四讲 DataFrame 表格数据结构 TODO
#+Title: 第十五讲 关系代数与回归分析 TODO
#+Title: 第十六讲 现实案例与未来方向 TODO
```

- 给讲义仓库提 issue 和 merge request ， 可获得伍至捌分。
- 一次原则的应用：把讲议与课件写到一起
  - 把课堂口述内容通过 speech-to-text 引擎合成文字

16.6 下一步学习

数据时代的物理技能



概率是逻辑的扩展 – Cox 定理

- Laplace: probability theory is nothing but common sense reduced to calculations.
- R. T. Cox, E. T. Jaynes，两位对统计学有重大贡献的物理学家
- Logical interpretation of probability
  1. Divisibility and comparability – The plausibility of a proposition is a real number and is dependent on information we have related to the proposition.
  2. Common sense – Plausibilities should vary sensibly with the assessment of plausibilities in the model.
  3. Consistency – If the plausibility of a proposition can be derived in many ways, all the results must be equal.
- 概率论的唯一性: Any system for plausible reasoning that satisfies certain qualitative requirements intended to ensure consistency with classical deductive logic and correspondence with common-sense reasoning is isomorphic to probability theory.

  参考: Van Horn, K.S., 2003. Constructing a logic of plausible inference: a guide to Cox' s theorem. International Journal of Approximate Reasoning 34, 3 – 24.

以本课程为起点

- 函数式编程: 一切都是函数
  - 无状态，从而容易从错误中恢复

- MapReduce：分布式大数据系统的开始
  - 在之上建立了关系代数：
  - Apache Hive，SparkQL，etc.
- 机器学习：与回归分析有同样的程序接口
  - 算法上更一般，需要更多的“调参”

竞赛

- gamma 大作业 → Ghost Hunter 2023 中微子数据分析排位赛
  - 概率统计分析及量测技术课赛结合

技术问题：TUNA 协会

- 清华大学学生开源软件与网络技术协会
- TUNA 主页 `https://tuna.moe/`
- TUNA 技术群，黑客（广义）技术问题探讨
- 为课程提供了
  - Gentoo 的镜像支持
    `https://mirrors.tuna.tsinghua.edu.cn/gentoo/`
  - Debian 的镜像支持
    `https://mirrors.tuna.tsinghua.edu.cn/debian/`
- 为大家提供了
  - 清华大学学位论文 LaTeX 模版 `thuthesis`

暑期实习

如果你觉得配环境是一个非常快乐的事，并且经常帮助同学配环境。
计算环境的可复现性是而容易被忽视，但也最重要的环节。

- Google Summer of Code：Google
  - Gentoo、Debian 操作系统相关项目
- 开源之夏活动：中科院软件所、华为、TUNA
  - Gentoo、Debian 操作系统相关项目
- 脱去资本枷锁，为人类数字化的自由而战 (bs，有点中二

SRT 与大创：Scheme

如果你认同实验物理与形式逻辑是文明的两大支柱，并喜欢课程的内容，可以考虑与我继续探索：

- 分析力学的 Scheme 描述：
  - Structure and Interpretation of Classical Mechanics, MIT 本科课程
  - 分析力学完成后，计划拓展至量子力学和电动力学
- 自动推理机：
  - 利用第一阶段大作业成果，直接生成第二阶段大作业的解答程序
  - 语言为 Anglican ，为 Clojure 语言的一种
  - Clojure 语言是 Scheme 语言在 Java 平台上的实现

致谢

顾问 杨鲁懿教授、高飞教授、陈晟祺、陈嘉杰

助教 王宇逸、刘晓义、刘学伟、刘明昊、盘笛、徐闯、陶嘉燊、孙迅、吴致颉、刘逸祺

小助教 李卓航、温欣洋、胡楚坤、陈诗洋、许威、徐一恺、胡宇阳、卢一鸣、宋明卓

科协 物理系科协环境配置培训，担任助教；工物系科协提供算力支持；计算系科协提供 saiblo 平台，第二阶段大作业平台，担任助教。

克服困难上课的同学们 谢谢！