R 语言在 CGED-Q JSL 中的运用

陈俊

2025-06 V2.0

本教程得到香港研究资助局优配研究金 16602621（Campbell PI）以及香港卓越学科领域计划 AoE/B-704/22-R (Chen PI) 的资助

本教程严正声明：本教程考虑到简化代码、串联制图流程、直观展示分析研究结果的重要性，所以在制作 GIS 图像章节使用了 CHGIS 制作的 1911 年中国历史地图，本教程制作的所有地图仅供展示制图代码结果以及个人学习制图代码使用，严禁单独转发、盗用、误用本教程制作的历史地图。本教程是开源的公益教程，因此不具备实时监控能力，对于任何因不完整转发、盗用、误用本教程制作的所有历史地图而产生的问题，本教程概不负责，亦不负任何法律责任。

前言

R 语言被广泛运用于生物学、医学、统计学、地理学、社会科学等领域，其功能强大、性能突出，深受统计软件使用者的青睐，是目前最受欢迎的数据分析软件之一。目前针对 R 语言的教程主要有：《R 语言实战（第 3 版）》、《R 语言数据分析与可视化：从入门到精通》、《R 语言入门与实践》，涵盖了 R 语言的基础知识、主要模块、数据可视化技巧等等，堪称全面。

相较于其它分析软件，R 语言至少具有以下三个优势：第一，免费开源。R 语言是一款开源的数据分析软件，任何使用者可以免费在互联网上下载 R 语言和 RStudio，相较于其他需要付费购买的分析软件，下载和使用 R 语言不需要付出经济代价。第二，外扩功能包众多。R 语言的一些使用者秉承着共享精神，开发出一些功能强大的功能包来扩展 R 语言的功能，使得 R 语言能够适应不同学科的需求，目前 R 语言已经有超过 4000 个外扩功能包，足以见其功能的强大。第三，不断更新。虽然 R 语言是一款免费的开源软件，但 R 语言的开发者、R 语言扩展包的使用者仍然在不断更新这一软件，不同于其他软件，R 语言的更新是多方进行的，能够在很大程度上确保新版本的适用性。

不过，R 语言对于初学者并不太友好，原因在于：第一，学习和运用 R 语言需要相当的数理基础。因为 R 语言没有 Excel、SPSS 等统计分析软件的交互式界面，使用者需要编写代码来运行 R 语言，尽管有不少开发者在 R 语言中开发了一些创建交互式界面的包，但受限于 R 语言的内核，这些包也没有被广泛地利用起来。第二，学习和运用 R 语言需要付出大量的时间成本。初学者不仅要耗费大量的时间学习其运行逻辑，还需要在理论学习和实践操作之间不断徘徊摸索。此外，初学者在运行环境的调整、运行结果的校对和运行报错的回溯中均需要耗费相当的时间。总之，R 语言功能繁多，且操作有一定的门槛，导致其在历史学等人文学科中运用并不太广泛。

目前还没有专门的教程讲解 R 语言如何在历史学当中运用。显然，以 R 语言为代表的计算机语言与传统历史学的联系并不太紧密，但是历史数据库的诞生为历史学引入计算机语言作为学术工具提供了契机。历史数据库建设方兴未艾，大量史料的存在为历史数据库建设奠定了基调。目前，学界在历史数据库建设领域已经取得了一些成果，比如哈佛大学牵头的“中国历代人物传记数据库”（CBDB）、香港科技大学李中清-康文林团队建设的“中国历史官员量化数据库——清代 缙绅录”（China Government Employee Database-Qing JSL，简称 CGED-Q JSL 或缙绅录数据库）。R 语言强大的字符串处理模块在历史数据库分析中具有先天的优势，但目前并没有专门讲解 R 语言如何分析历史数据库的教程，对于同时对定量分析、历史数据库感兴趣但又苦于没有学习门路的学人而言，实乃一个遗憾。

本教程主要运用以 R 语言为内核的 RStudio 对缙绅录公开版数据库（CGED-Q JSL Public Release 1760-1798 & 1850-1864 & 1900-1912）进行分析，主要包含 Rstudio 的界面简介、变量创建、数据转换、制作图表、数据集链接、制作 GIS 图像等内容，是一个针对 R 语言初学者的基础性教程。缙绅录数据库是李中清-康文林团队的研究成果之一，也是目前国内知名度较高的历史数据库之一。缙绅录数据库具备高度结构化的特性，利用字符串作为存储方式，其记载量大、连续性强、可信度高，有上百万条记录，在历史数据库中具有典型性。故本教程选择缙绅录数据库作为分析目标，希望为 R 语言分析历史数据库提供一些启发。

我同时对数据分析和历史数据感兴趣，通过课余时间自学了一些数据分析软件，但都只学会了一些皮毛。就

我使用数据分析软件的经验来看，STATA 虽功能全面，集成度高，但价格昂贵，对于没有研究经费支持的学人而言不堪重负；SPSS 虽有交互式操作界面，但其在处理历史数据库方面不具特色，也缺少可视化功能；而各类 GIS 软件是历史数据可视化的重要软件，但需要重新导入、编辑、整合大体量的数据库，颇为麻烦。R 语言拥有强大的字符串处理功能，又同时拥有令人称道的可视化功能，是处理历史数据库最为合适的软件之一。我结合自己学习 R 语言时的经验编写了这一部适合初学者的教程，教程中有很多实例是我在实践过程中遇到的案例。我希望此教程能为历史数据库使用者提供一个思路，让使用者们能够受到一些启发，这也是本教程的价值所在。

我在担任康文林教授（Prof. Cameron Campbell，香港科技大学人文社会科学学院署理院长兼讲座教授、华中师范大学历史文化学院“长江学者”特聘教授）“大数据历史的理论与方法”这一课程的助教时编写了一部教学 PPT 和 R 语言代码包。康文林教授是我的硕士指导老师和 STATA 数据课老师，是他鼓励我从事历史数据库分析这一方向，并支持我将这一套教学 PPT 和 R 语言代码包整合成一部教程，也提供了很多具有开创性的指导意见，为本教程的编写提供了巨大的帮助。日本一桥大学的倪志宏副教授（Prof. Matthew Noellert）为本教程的更新和完善提供了许多宝贵的修改建议。此外，香港科技大学的韦圣彬和侯玥然同学、上海交通大学的吴艺贝博士、华中师范大学的高帅奇博士在本教程的修改过程中也提供了许多建议，谨在此向各位师友表示衷心地感谢！

本教程是我根据数据库处理经验并参考部分网络资源（已标明出处）制作而成的，如果有使用者发现有任何未注明引用的地方，请联系作者删除。另外，本教程是一个完全公益的教学手册，服务于广大历史爱好者、历史数据库使用者、数据分析软件初学者，使用者可以在互联网中免费下载，请不要利用本教程进行任何获利的活动，如需转载，请联系作者。本教程是一个主要针对 R 语言初学者的教程，亦是一次初步的尝试。经过两年的课堂实践，我们发现本教程对于零基础的初学者有着一定的帮助，而对于那些本就精通和擅长 R 语言的学人，我们更希望该教程能够提供一些思路上的启发。由于我编程水平有限，在技术层面和组织层面都存在着问题，恳请广大读者和使用者批评指正，我将会不断增补、更新此教程。

陈俊 于清水湾

2025 年 6 月（2023 年 5 月初版）

目　录

前言 **I**

图目录 **VI**

表目录 **VII**

第一章 R 以及 RStudio 的安装与界面 **1**

1.1 R 和 RStudio 的关系 1

1.2 R 以及 RStudio 的下载 1

1.3 RStudio 的界面 1

1.4 R 语言的自学资源 3

第二章 数据库的管理 **5**

2.1 缙绅录数据库及下载方式 5

2.1.1 利用缙绅录数据库开展的相关研究 5

2.1.2 缙绅录数据库的下载方式 6

2.2 在 RStudio 中导入数据库 6

2.3 在 RStudio 中查看、保存数据 7

第三章 字符串处理 **9**

3.1 在数据库中创建一个新变量 9

3.1.1 基础运算符 9

3.1.2 删除数据库中的缺失值 10

3.1.3 转换变量的类型 11

3.1.4 创建新变量 11

3.2 逻辑表达式 12

3.3 转换变量 13

3.3.1 将字符型变量转换为数值型变量 13

3.3.2 将数值型变量转换为字符型变量 14

3.3.3 将离散型变量转换为数值型变量 15

3.4 字符串处理的相关函数 17

3.4.1 串联字符 17

3.4.2 提取和判断字符 18

3.4.3 替换字符 19

第四章 制表 **20**

4.1 简单制表 20

4.2 整理变量 20
4.3 制作可以导出的表格 24
4.3.1 利用 flextable 包导出二联表和三联表 24
4.3.2 利用 table1 包导出表格 33
4.4 指定条件制表 34

第五章 直方图、散点图、折线图 **37**
5.1.1 基础直方图 37
5.1.2 分条直方图 38
5.1.3 堆积直方图 41
5.1.4 利用非数值型变量和离散型变量制作直方图 43
5.2 散点图 44
5.3 折线图 47

第六章 数据集的匹配 **50**
6.1 筛选列与行 50
6.1.1 筛选列 50
6.1.2 筛选行 51
6.2 数据集的匹配 51
6.2.1 匹配以整理变量 51
6.2.2 如何创建一个副数据集 55
6.3 连接两个数据集 57

第七章 数据集的内外连接 **59**
7.1 数据集内部记录的连接 59
7.1.2 利用个人编号分析官员任职年限 60
7.2.3 如何统计人数而非记载数 61
7.3 两个数据集的连接 64

第八章 制作 GIS 图像 **68**
8.1 导入 GIS 底图 68
8.2 构建地理坐标 70
8.3 绘制 GIS 图像 72
8.3.1 绘制点密度图 72
8.3.2 绘制核密度图 74
8.3.3 绘制热力图 76
8.3.4 添加指北针、比例尺和制作、导出组合图 79
8.4 制作局部 GIS 图像 81

8.5 利用离散数据制作分级统计图 84

第九章 总结 **87**

图目录

插图

图 1　缙绅录数据库中官员的地理来源 38
图 2　1900-1906 年兵部官员的满汉比例分条直方图 40
图 3　1900-1906 年兵部官员的满汉比例堆积直方图 42
图 4　NBA 球员的场均得分分条直方图 43
图 5　1900-1906 年兵部官员的品级与数量 46
图 6　NBA 球员的篮板助攻比散点图 47
图 7　1900-1906 年兵部官员满汉比例折线图 49
图 8　1900-1912 文官的出身结构 55
图 9　1906 年文官的地理来源统计图 63
图 10　1911 年秋全国文官任职地分布每个点代表一名文官 74
图 11　1911 年秋全国文官任职地分布核密度图 76
图 12　1911 年秋全国文官任职地分布热力图 78
图 13　1911 年秋全国文官任职地分布组合图 81
图 14　1911 年秋华北地区文官任职地分布点密度图 83
图 15　1911 年秋全国文官省籍分布图 86

表目录

表格

表 1　R 语言中的基础运算符 10
表 2　文官的籍贯和满汉比例（flextable） 27
表 3　清代知县的满汉比例和地理来源 31

第一章 R 以及 RStudio 的安装与界面

1.1 R 和 RStudio 的关系

R 语言 1995 年由新西兰奥克兰大学的 Ross Ihaka 以及 Robert Gentleman 开发，因两位开发者英文名的首字母都是 `R`，所以被称为 R 语言。R 语言最初主要被运用到生物学研究领域，由于 R 语言用户和程序包开发者数量的增长，R 语言逐渐被运用到医学、经济学、统计学、地理学、社会学领域。目前，R 语言拥有 4000 多个外扩程序包，已经成为了最受欢迎的数据分析软件之一。

RStudio 是 R 语言的一个集成开发环境（Integrated Development Environment）。简单来说，RStudio 是 R 语言的简便操作系统，在 RStudio 中，能够同时开展编码、分析、记录、管理、展示、优化等多个项目。总之，RStudio 更加便利，也更加适合没有编程基础的人使用。需要注意的是，RStudio 是不能单独运行的，需要使用者提前下载好 R 语言。本手册是在使用 RStudio 的基础上对缙绅录数据库进行分析所开发出的基础教程。

1.2 R 以及 RStudio 的下载

R 语言的下载地址众多，我们推荐的是清华大学的镜像网站https://mirrors.tuna.tsinghua.edu.cn/CRAN/，请使用者根据电脑系统选择需要下载的类型。这里以 Windows 系统为例，首先在浏览器中输入网址，然后点击 `Download R for Windows`，再点击 `install R for first time`，最后点击 `Download R-x.x.x for Windows`[1]，并打开下载好的 R 语言安装程序根据提示进行安装。另外，我们建议使用者在非系统盘建立一个以 `R` 为名称的新文件夹来存储下载好的 R 语言程序和其他相关文件，且文件路径中不要出现中文字符。

下载好 R 语言之后，进入https://posit.co/download/rstudio-desktop/#download下载 RStudio，请使用者根据电脑系统选择需要安装的类型，这里仍然以 Windows 系统为例，在浏览器输入网址后，选择 `RStudio-yyyy.mm.dd-xxx.exe`[2]，并打开下载好的 RStudio 安装程序根据提示进行安装。我们建议使用者将 RStudio 和 R 语言存放在同一个文件夹下。

1.3 RStudio 的界面

RStudio 的操作界面可以分为 4 个区域和一个功能栏。下面逐一讲解四个区域以及功能栏的内容。

1. 区域一。区域一位于 Rstudio 界面的左上角，是 RStudio 的命令编写区和数据查看区。下载后刚开始运行 Rstudio 时没有区域一，可依次点击 `File>New File>R script` 。下面介绍几个常用的按钮：

    1. 主区域：区域一中间的空白区域可以编写代码，可随时增减、修改代码，添加注释，也可以打开已有的 R Script 文件。同时，该区域也是数据框的内容查看区。

[1]这里的 `R-x.x.x` 是指目前 R 语言的最新版本，R 语言会不断更新版本，请使用者留意并下载最新版本的 R 语言程序。

[2]这里的 `RStudio-yyyy.mm.dd-xxx` 是指 RStudio 的最新版本，与 R 语言一样，RStudio 也会不断更新版本，请使用者留意并下载最新版本的 RStudio。

2. `Run`：选中代码，点击 Run 运行。

3. `Source on save` 左边的保存按钮：将编写的代码保存为脚本文件，类似于 STATA 的 do 文件。

2. 区域二。区域二位于 Rstudio 界面的右上角，是 RStudio 的数据窗口。主要功能包括：

    1. `Environment`：展示导入的数据集（库）。

    2. `History`：运行过的所有代码都保存在历史区中。

    3. `Connections`：可用于连接外部数据集。

3. 区域三。区域三位于 Rstudio 界面的左下角，是 RStudio 的控制区。主要功能如下：

    1. `Console`：控制台，可在此输入代码。

    2. `Terminal`：将代码发送到终端。

    3. `Jobs`：可以查看安装包的工作进度。

4. 区域四。区域四位于 Rstudio 界面的右下角，是 RStudio 的输出窗口。主要功能如下：

    1. `Files`：保存文件的区域。

    2. `Plots`：输出区域。

    3. `Packages`：R 的扩展包展示区，点击即可自动加载运行扩展包，还设置了搜索功能，可供搜索没有被展示出来的扩展包。
    4. `Help`：帮助显示区。

    5. `Viewer`：查看区。

5. 功能栏。功能栏位于 Rstudio 界面的左上角（MAC OS 系统位于屏幕左上顶部），是 RStudio 的设置区。主要功能如下：

    1. `File`：新建、打开、保存文件等功能。

2. `Edit`：编辑功能。

3. `Code`：优化编码体验，如添加注释等。

4. `View`：查看界面内容、定位光标等。

5. `Plots`：查看图片、保存图片。

6. `Session`：新建工作界面、重启 R 等。

7. `Build`：创建包。

8. `Debug`：错误代码回溯功能。

9. `Profile`：预设代码功能，预设代码可在 R 启动时自动运行。

10. `Tools`：工具、安装包、界面设置等功能。

11. `Help`：帮助功能。

1.4 R 语言的自学资源

R 语言在国外数据科学领域的运用十分广泛，许多大学开设有专门的 R 语言数据分析课程，比如世界知名大学约翰霍普金斯大学、杜克大学等都开设了 R 语言编程课程，这些课程都有网络公开课版本，使用者可以自行在网络上下载并学习。

此外，R 语言自学书籍丰富多样，涵盖范围广。我们在这里主要推荐的是《R 语言实战》。[3]该书作者罗伯特 I. 卡巴科弗（Robert Ira Kabacoff）是一名公认的统计编程专家，创建并维护着 R 语言学习网站 Quick-R, 且拥有心理学的学士和博士学位，他在多元统计方法、数据可视化、预测分析和心理测量学方面拥有 30 多年的经验。《R 语言实战》是公认的学习 R 语言的基础教材，适合从零开始学习 R 语言的使用者，其由浅入深，从最基础的向量、矩阵开始，逐步扩展到回归分析、统计建模等部分，最后落脚于 R 语言最强大的可视化功能上，整体逻辑性强、全面且细致，推荐 R 语言初学者使用。

R 语言中名声在外的 `ggplot2` 包，是 R 语言强大的数据可视化功能的重要体现。学习 R 语言 `ggplot2` 包的书籍包括：《R 语言入门与实践》、《R 语言数据分析与可视化：从入门到精通》、《R 语言数据可视化之美》、《R 语言数据可视化实战》等等。另外，北京大学数学科学学院李东风老师将自己多年统计分析课程的讲义整理成了

[3]目前最新版的《R 语言实战》是第 3 版，请使用者留意该书的版本变化。

一套全面的教程——《R 语言教程》，这套教程并未公开出版，供学生内部使用，有兴趣者可登录北大数院官网学习观摩。在这里，还想简单介绍一下 `tidyverse` 项目，这是一个包括了数据科学的一个集合工具项目，用于数据提取，数据清理，数据类型定义，数据处理，数据建模，函数化编程，数据可视化，包括了下面的包：

1. `ggplot2` 数据可视化；
2. `dplyr` 数据处理；
3. `tidyr` 数据清理；
4. `readr` 数据提取；
5. `purrr` 函数化编程；
6. `tibble` 数据类型定义。

这一项目同样也是 R 语言中使用频率最高的项目之一，它不仅包括了可视化功能，还包括了字符串处理功能，堪称全面。目前，国内许多视频软件平台、R 语言教程、代码分享互助网站有很多关于这个项目及其子项目的介绍和使用范例，大家可自行在网络上查找资源，此处不再赘述。[4]

针对 R 语言的专业软件除了前述的 `Quick-R`，还有 `RWeekly` 网站。此外，国外著名的数据编程共享网站 `Github` 上有许多 R 语言编程专家设有专栏介绍 R 语言各种函数、包的用法，有些博主还在线解答疑问。国内类似的网站 `CSDN` 中也有不少针对 R 语言编程问题的专栏。此外，国内的在线视频网站 `Bilibili` 有整套的 R 语言在线学习课程。

我们建议使用者从多个来源学习 R 语言，这对使用者后续提高代码纠错能力、扩展编程知识运用广度是有帮助的。

[4] `tidyverse` 项目的地址https://github.com/tidyverse/tidyverse。

第二章 数据库的管理

2.1 缙绅录数据库及下载方式

李康研究团队是由香港科技大学人文社会科学学院李中清、康文林教授领导的研究团队。团队从 1980 年代开始构建量化数据库，并进行相关的量化历史研究，主要包括中国多代人口量化数据库、中国大学生量化数据库、清代缙绅录量化数据库等。《缙绅录》是开列官职和在任官员的详细名单，从清代中后期开始按季出版，有大量版本存世。团队在建的《中国历史官员量化数据库（清代）》以清华大学图书馆出版的《清代缙绅录集成》为基础，同时以哈佛燕京图书馆、哥伦比亚大学图书馆、上海图书馆等所藏《缙绅录》版本作为补充。[5]

2.1.1 利用缙绅录数据库开展的相关研究

缙绅录数据库体量庞大，涵盖历史信息众多，具有极高的史料价值。目前，学界已有众多利用缙绅录数据库的研究。陈必佳、任玉雪等学者《清代缙绅录量化数据库与官员群体研究》，初次向大众介绍了缙绅录数据库及其在官员研究上的巨大潜力；之后，陈必佳回顾了缙绅录数据库的建设过程，探讨了缙绅录记载的准确性和史料价值；近来，陈必佳还从方法论的角度探讨了数字人文研究的前景，并考察了缙绅录数据库在官员仕途研究上的可行性。[6] 以上作品均是从缙绅录数据库本身出发从整体的视角探讨缙绅录数据库的过去、未来和前景，目前学界也有不少针对缙绅录数据库的专题文章：康文林利用 1900-1912 年的缙绅录数据库分析了科举停废 对士人文官群体的影响，从微观大数据的视角重新审视了科举停废这一近代史上的重大议题；[7] 薛勤、康文林两位学者则聚焦于清末改革，以清末官制改革为切入点研究了吏部这一重要人事部门的职员结构变动，从而回顾清末官制改革的成效；[8] 胡存璐、胡恒等学者则是将关注点下移到地方行政官员——知州群体上，并结合清代的政区分等制度研究清代知州的选任模式和特点，重点分析了地理因素对官员选任的影响。[9]与之相似的是，胡恒、陈必佳等学者以清代知府群体为例，利用空间量化分析的方法从历史地理学的角度考察清代知府的选任特点。[10]

不难看出，针对缙绅录数据库的研究是从多维度同时展开的，且大都发表在历史学的重要期刊上，足见其在历史数据库中的典型性。缙绅录数据库不仅具有巨大的史料价值，可以为其他历史数据库的建设提供范例；它还拥有巨大的研究潜力，通过对缙绅录数据库的研究可以为清史、近代史上的重要议题提供一些证据，推进史学发展。

[5]李康团队的官网是https://shss.hkust.edu.hk/lee-campbell-group/。

[6]参见任玉雪, 陈必佳等:《清代缙绅录量化数据库与官员群体研究》,《清史研究》2016 年第 4 期; 陈必佳:《“中国历史官员量化数据库-清代 ” 的建设过程，现状与前景》，付海晏主编:《大数据与中国历史研究》（第 3 辑），北京：社会科学文献出版社，2021 年，第 31-44 页；陈必佳:《再论记载的准确性及其史料价值》,《清史研究》2019 年第 4 期；陈必佳:《“数字人文 ” 与清代官员仕途研究》,《史学月刊》2023 年第 4 期。

[7]参见康文林:《清末科举停废对士人文官群体的影响——基于微观大数据的宏观新视角》,《社会科学辑刊》，2020 年第 4 期。

[8]薛勤、康文林:《清季改革视阈下吏部官员群体的人事递嬗与结构变迁 (1898—1911) ——以《缙绅录》数据库为中心》,《社会科学研究》2022 年第 2 期。

[9]胡存璐、胡恒、陈必佳、康文林:《清代州的政区分等与知州选任的量化分析》,《数字人文研究》2021 年第 1 期。

[10]胡恒、陈必佳、康文林:《清代知府選任的空間與量化分析——以政區分等、《縉紳錄》數據庫為中心》,《新亞學報》Vol.37，August。

2.1.2 缙绅录数据库的下载方式

缙绅录数据库的部分内容已经向社会公开，下载网址包括香港科技大学数据空间、哈佛大学数据储存库（Harvard Dataverse Lee-Campbell Group Dataverse）以及中国人民大学清史数据共享平台。

香港科技大学数据空间下载方式为：进入香港科技大学数据空间，根据下载目录选择最新版本的缙绅录数据库，已公开的缙绅录数据库包含三个年代（1760-1798；1850-1864；1900-1912），我们推荐使用者在港科大数据空间下载最新版本 dta 版本的缙绅录数据库，我们会不定期更新数据库的版本，请使用者留意。

中国人民大学的下载方式为：浏览器搜索中国人民大学清史数据共享平台，点击资源目录，再点击中国历史官员量化数据库——清代，缙绅录，浏览器跳转至数据下载页面（下载页面中也可以选择香港科技大学的下载地址），下载页面中包括数据库（xlsx 格式、dta 格式、tab 格式）、用户指南、重要声明、数据库处理便捷代码集等资源。我们建议使用者选择 `CGED-Q Public Release 1760-1798 1 July 2024 STATA.tab`（右边下载按钮下拉后选择 `Stata 14 Binary Original File Format` ）、`CGED-Q Public Release 1850-1864 19 Apr 2022` 和 `CGED-Q Public Release 1900-1912 22 May 2022.dta`，这三个分别是缙绅录数据库乾隆后期部分、太平天国部分和清末部分，使用 dta 格式的目的是方便在 RStudio 中导入和分析。另外，数据库整理的便捷代码集我们也建议下载保存，方便后续数据库的分析，同时，我们强烈建议使用者在使用数据库之前阅读用户指南和重要声明。[11]

2.2 在 RStudio 中导入数据库

在 RStudio 中导入数据库主要有三种方法：

第一种是利用 RStudio 的 Import 功能。Import 导入的具体步骤为：先要保证联网，因为 RStudio 下载资源包需要转到镜像网站下载，没联网无法开启 Import 功能。之后下拉区域 2 中 `Environment` 的 `Import dataset` 选项，选择需要导入的文件类型。然后点击浏览，选择文件，预览无误后，点击 Import 即可，数据库较大，可能需要稍等几分钟。或者点击功能栏 `File` 下的 `Import dataset`，效果是一样的。Import 导入快捷方便，推荐导入缙绅录 dta 版本。

第二种方法是利用代码导入。具体的代码如下：

```
library(haven)


JSL1900_1912 <- read_dta("/Users/jc/Documents/2-数据库-缙绅录数据库/原始文件/缙绅录公开版（新）
 ↪  /1900-1912 19 Apr 2022.dta")
# 更换你自己的文件储存路径
```

[11]任玉雪，陈必佳，郝小雯，康文林，李中清：《中国历史官员量化数据库——清代缙绅录 1900-1912 时段公开版用户指南》。因为数据库在不断优化，所以研究团队也在不断更新用户指南，建议使用者留意并下载最新版本的用户指南。

这种方法的思路和 Import 导入是一样，不同点是 Import 是自动运行函数，而代码需要自己调整路径。请使用者注意根据文件储存位置调整 `read_dta()` 括号中的内容，同时请记得添加引号。
第三种方法是复制粘贴。首先将需要的内容复制到剪切板，然后输入代码：

```
data <- read.table("clipboard", head = T)
```

即可完成剪切板内容的导入。但复制粘贴这种方法只适合于较小的数据，对于大型的数据库来说，Import 和代码导入是最实用高效的导入数据的方式。

2.3 在 RStudio 中查看、保存数据

查看数据。导入之后，在区域一看到数据库的内容，在区域二可以看到数据库的信息。区域一中可以查看数据库的所有记录和变量，区域二中包含数据的基本信息，总记录数和变量数。
保存数据。RStudio 的保存系统主要可以保存以下几个核心类型：

1. 以.Rproj 为后缀的文件，这是 R 语言的主项目文件，保存了运行 R 语言的元数据信息；
2. 以.R 为后缀的文件，这是 R 的脚本文件 R Script，可编写并运行代码；
3. 以.Rhistory 为后缀的文件，这是根据区域二的 history 保存的文件，有 R 语言的会话历史；
4. 以.Rmd 为后缀的文件，这是 R Markdown 文件，混合了 Markdown 文本和 R 代码块，可用于创建可重复生成的研究报告、论文和演示文稿；
5. 以.RData 为后缀的文件，这是区域二中 Enviroment 的数据文件，可以单独保存，亦可以导入到 RStudio 中。

结合以上内容，保存数据工作的重点就较为清晰了：

第一步，点击功能栏 File 下拉项 Save All 保存所有脚本文件（.R、.Rmd 等）。我们强烈建议使用者对载有重要代码信息的.R、.Rmd 文件进行备份，以免重要代码文件丢失。但请注意 Save All 并不会保存 Enviroment 中的对象（比如数据框、变量等）。

第二步，要保存工作空间（即区域二中 Enviroment 的数据文件），可以使用 save.image() 函数指定路径进行保存，或者通过 RStudio 的菜单 Session -> Save Workspace As... 保存为.RData 文件。同时，还可以在 Project Options > General 中可配置退出时是否自动保存工作空间.RData 和命令历史.Rhistory。我们强烈建议使用者对.RData 文件进行备份，以免重要数据文件丢失。

只要我们将脚本文件和数据空间进行及时保存和备份，即便遇到 R 语言出错或者电脑宕机等情况也能从容应对，只需要导入相应脚本文件和数据空间文件便能再次回到原来的进度中。保存数据工作异常重要，请确保在 RStudio 中的工作得到系统化保存，避免数据丢失，并保持项目的可重复性和可维护性。

在 Rstudio 中打开 R 语言代码包（文末附有下载地址），有的使用者可能遇到出现乱码的问题，具体的解决方法是：依次点击 `File` — `Reopen with encoding` — `UTF-8` 即可。

第三章 字符串处理

3.1 在数据库中创建一个新变量

3.1.1 基础运算符

下面是 RStudio 和 R 中的基础运算符：

```
# 安装和读取相关的包
options(repos = c(CRAN = "https://cloud.r-project.org/"))
# 设置全局 CRAN 镜像，使用清华大学镜像源
install.packages("bookdown")
install.packages("officer")
install.packages("flextable")
install.packages("fontBitstreamVera")
install.packages("dplyr")
library(bookdown)
library(officer)
library(flextable)
library(dplyr)

# 生成数据框
Basic_symbols <- data.frame(基础运算符 = c("+",
    "-", "*", "/"), 含义_1 = c(" 加 ", " 减 ", " 乘 ",
    " 除 "), 判定运算符 = c("<", ">", "<=", ">="),
    含义_2 = c(" 小于 ", " 大于 ", " 小于等于 ",
        " 大于等于 "), 逻辑运算符 = c("==",
        "!=", "|", "&"), 含义_3 = c(" 严格等于 ",
        " 不等于 ", " 或者 ", " 和 "))

print(Basic_symbols)  # 在区域三中查看运算符
```

```
# 导出表格
flextable(Basic_symbols) %>%
    set_caption(caption = "R 语言中常用的运算符 ",
```

```
    autonum = run_autonum(seq_id = "tab"))
```

表 1: R 语言中的基础运算符

基础运算符	含义 _1	判定运算符	含义 _2	逻辑运算符	含义 _3
+	加	<	小于	==	严格等于
-	减	>	大于	!=	不等于
*	乘	<=	小于等于	|	或者
/	除	>=	大于等于	&	和

在分析过程中，比较常用的是 ==、!、|、&, 这些运算符被称之为逻辑运算符。

3.1.2 删除数据库中的缺失值

数据库中存在缺失值是正常的现象，缺失值的存在会影响分析结果。因此，在分析之前，我们需要剔除掉部分缺失值，在缙绅录数据库中，缺失值存在的形式是官员的名为空白。我们利用以下代码清除掉缙绅录数据库中姓名为空白的纪录。代码中，kongbaiming 是新变量的名称，可自行拟取，<-表示等于，which() 函数能够返回指定值在逻辑向量中的位置。简单来说，就是利用 which() 函数指定条件，RStudio 就能够返回满足条件的记录所在的行数，这里利用 which() 函数找出名字是空白的记录，JSL1900_1912 是数据库的名称，$ 是引用符号，表示引用该数据库的其中一个变量，名是被引用变量的名称，== 是严格等于，空白是判定条件，如果是字符，要加上引号，如果是数值则不用。

```
kongbaiming <- which(JSL1900_1912$名 == " 空白 ")
JSL1900_1912_clean <- JSL1900_1912[-kongbaiming, ]
```

JSL1900_1912_clean 是一个新的数据库的名称，为了便于理解，我们在取名时用了完整的名称，导致整个数据库的名字比较长，使用者可根据自身需要简化新数据库的名称，而 JSL1900_1912[-kongbaiming,] 表示在原数据库中删除 kongbaiming 的记录。

剔除缺失值是一个基础的环节，比较重要，所以我们放到创建新变量之前来讲，之后我们的数据分析基本上都会在已经剔除缺失值的 JSL1900_1912_clean 中进行。当然，此处剔除缺失值我们仅举了一个例子，即名是空白的记录条数，如果使用者仔细观察缙绅录数据库，会发现还有一些变量上记载的是涂黑或 塗黑，这也是缺失值表现形式的一种，使用者可以利用以上方法剔除掉其他的缺失值，提高数据分析的准确性。

3.1.3 转换变量的类型

目前大多数历史数据库都是字符串类型，即便有一些记录看起来是数值，但其存储方式仍然可能是字符串。因此，在进行变量之间的运算之前，需要将字符串类型的数据库转换为数值型，不然运算时 RStudio 会报错。将字符串类型转换为数值类型的函数是 `as.numeric()` 函数，这一函数的格式为：

```
data$x <- as.numeric(data$x)
```

在该函数格式中，`data` 为数据库或数据集，`$` 为引用符号，这条代码的意思是将 `data` 中的 `x` 变量转化为数值类型。下面，我们根据这一代码的格式编写一个可以转换缙绅录数据库变量类型的实例代码：

```
# 将`JSL1900_1912_clean`数据库中的变量“阳历年份”，
# 利用`as.numeric()`函数转换变量类型，进而生成一个新变量“阳历年份 numeric”
JSL1900_1912_clean$阳历年份 numeric <- as.numeric(JSL1900_1912_clean$阳历年份)

# 利用`typeof()`函数返回变量的属性，检验变量类型转换是否成功
typeof(JSL1900_1912_clean$阳历年份 numeric)

# 将“阳历年份”变量换成了“季节号”
# “季节号”转化为数值类型后，1 代表春季、2 代表夏季、3 代表秋季、4 代表冬季
JSL1900_1912_clean$季节号 numeric <- as.numeric(JSL1900_1912_clean$季节号)

# 利用`typeof()`函数返回变量的属性，检验变量类型转换是否成功
typeof(JSL1900_1912_clean$季节号 numeric)
```

3.1.4 创建新变量

在完成了剔除缺失值、转换变量类型的工作后，我们可以在数据库中随意建立两个以上变量的联系，这种联系通过添加运算符实现。我们现在需要在缙绅录数据库中建立一个新变量“年份季节 ”，用以展示缙绅录数据库中各版本的年代、季节，可以通过以下代码实现：

```
JSL1900_1912_clean$年份季节 <- (JSL1900_1912_clean$阳历年份 numeric +
    ((JSL1900_1912_clean$季节号 numeric/4) - 0.25))
```

以上第 1 条代码看似比较长，实则是数据库的名字取得比较长，逻辑非常简单。其思路是：

```
data$x <- (data$y+((data$z/4)-0.25))
```

在 JSL1900_1912_clean 数据库中，用“阳历年份 numeric”的数值加上“季号份 numeric”数值的 1/4，再减去 0.25，最后生成新的数值来表达缙绅录版本的年代和季节，比如，1906 年夏季的缙绅录，它的“年份季节 ”便是（1906+2/4）-0.25，即是 1906.25，那为什么要在最后减去 0.25，而不是直接用“阳历年份 ”加上“季节号 ”的 1/4 呢？因为如果不减去 0.25，数值进位会让冬季版本被归类为次年春季版本。比如，1907 年冬季的版本，如果不减去 0.25，就变成 1907+4/4，即 1908.00，而 1908.00 是表示 1908 年春季的版本。

利用 table() 函数查看变量

```
table(JSL1900_1912_clean$年份季节)
# `table()`函数能够返回某个变量的记录数。
# 运行这条代码后，RStudio 会在区域三显示出每个年份季节的记录数
```

在创建新变量的过程中，我们利用运算符建立了两个变量之间的联系，这也是量化的一种思路。当然，我们所利用的两个变量，其本身就是数字，所以很容易利用运算符建立联系，当两个变量是字符型变量时，它们便不适合进行运算，这时就需要利用到其他函数创建联系，这个我们在后文会讲到。

3.2 逻辑表达式

逻辑表达式是利用逻辑运算符和关系表达式所编写的代码。逻辑表达式所产生的是一个逻辑值，主要用于逻辑判定。在历史数据库的分析中，我们经常会用到逻辑表达式来判定一个记录的类型。比如，在历史人口数据库中，判定一个人是否拥有孩子就可以用到逻辑表达式，当某个人的记录上，后代数量这一栏中只要标注了非 0 的数量，无论这个数字有多大，其是否拥有孩子这一栏上都会标注上是；反之，则会标注上否。因此，逻辑表达式在历史数据库中运用得十分广泛，下面我们介绍一个简单的逻辑表达式 ifelse() 函数：

```
ifelse(test,yes,no)
```

test 即为 R 需要执行的条件，通常是某个变量大于小于或等于某个值，简单来说就是一个判定条件。如果满足这个判定条件，RStudio 则返回 yes，yes 可以更改为 True 或者 1 等等。如果不满足判定条件，则返回 no，no 可以替换为 False 或者 0 等等。

比如，在缙绅录数据库中，我们编写一个代码来判定缙绅录的版本年份是否在 1900 年之后：

```
ifelse(年份 < 1900, T, F)
```

在清代官员的研究中，有一个比较重要的议题是官员的满汉比例。我们可以通过 ifelse() 函数来判定一个官员是否为“旗人 ”，进而深入研究清代官员满汉比例的变化，具体代码如下：

```
JSL1900_1912_clean$qiren <- ifelse((JSL1900_1912_clean$旗分 !=
    "" | JSL1900_1912_clean$身份二 != "" | JSL1900_1912_clean$姓 ==
    ""), 1, 0)
```

该代码结合了缙绅录数据库中的三个变量来判定官员的民族属性，第一，“旗分 ”，部分旗人的旗分有记载，比如“汉军正蓝 ”等等；第二，“身份二 ”，少部分旗人有身份标识，如宗室、觉罗等等；第三，“姓 ”，缙绅录数据库中，基本上所有旗人都没有姓氏记载，三个关系表达式之间用 | （或者）连接，三个条件任意满足其中一个即可判定为 1（是），反之则返回 0（否）。

```
# 若`qiren`变量等于 1 的话，就改变其值为“旗”
JSL1900_1912_clean$qiren[JSL1900_1912_clean$qiren ==
    1] <- " 旗 "

# 若`qiren`变量等于 0 的话，就改变其值为“民”
JSL1900_1912_clean$qiren[JSL1900_1912_clean$qiren ==
    0] <- " 民 "

# 在区域三返回逻辑值是`1`（旗人）和`0`（民人）的记录数
table(JSL1900_1912_clean$qiren)
```

```
##
##     旗     民
## 154873 484559
```

3.3 转换变量

3.3.1 将字符型变量转换为数值型变量

前面我们介绍了一种方法，利用 as.numeric() 函数将字符型变量转换为数值型。但这种方法的实质是改变变量的形式，让其能够参与计算。另外，这是一种纵向的处理方式，即让该纵列变量下的所有记录全部转换为数值类型。那么，如何让指定的字符串记录转换为数值型呢？以下代码是一个例子：

```
# 将缙绅录数据库中“出身一”变量为“状元”的记录，转换为数值 1
JSL1900_1912_clean$出身一 [JSL1900_1912_clean$出身一 ==
    " 状元 "] <- 1
```

```
# 将“出身一”变量为“榜眼”的记录，转换为数值 2
JSL1900_1912_clean$出身一 [JSL1900_1912_clean$出身一 ==
    " 榜眼 "] <- 2

# 将“出身一”变量为 1 的记录，转换为数值“状元”
JSL1900_1912_clean$出身一 [JSL1900_1912_clean$出身一 ==
    1] <- " 状元 "

# 将“出身一”变量为 2 的记录，转换为数值“榜眼”
JSL1900_1912_clean$出身一 [JSL1900_1912_clean$出身一 ==
    2] <- " 榜眼 "
```

将“状元 ”变量转化为 1 的作用在于：如果要建立一个官员升迁的模型，需要展示出身对官员升迁的影响，那转换后的数值 1 则比原记录“状元 ”更加适合引入到量化模型中。

3.3.2 将数值型变量转换为字符型变量

同样的逻辑，将数值型变量转换为字符型，代码示例如下：

```
# 将“季节”变量中的数值转换为字符，
# `1`转变为春、`2`转变为夏、`3`转变为秋、`4`转变为冬
JSL1900_1912_clean$季节 [JSL1900_1912_clean$季节 ==
    1] <- " 春 "
JSL1900_1912_clean$季节 [JSL1900_1912_clean$季节 ==
    2] <- " 夏 "
JSL1900_1912_clean$季节 [JSL1900_1912_clean$季节 ==
    3] <- " 秋 "
JSL1900_1912_clean$季节 [JSL1900_1912_clean$季节 ==
    4] <- " 冬 "
```

```
# 利用`table()`函数查看春、夏、秋、冬的数量
table(JSL1900_1912_clean$季节)
```

```
##
```

```
##     冬     夏     春     秋
## 152530 150320 183830 152752
```

3.3.3 将离散型变量转换为数值型变量

可以看出，前两种转换变量的方式都有很大的相似之处，代码的格式也基本是同一种类型。这是因为缙绅录的数值都是连续型变量，具有规律性。但当一个数据集是离散型变量，该如何转换?

下面是根据 NBA 官网的数据制作的一个球员生涯各项数据表格。以下代码将在这个表格的基础上进行。

```
NBA <- data.frame(Player = c("Kobe", "Jordan", "James",
    "Garnett", "Duncan", "Kidd", "Anthony", "Pippen",
    "Curry"), Points = c(33643, 32292, 34087, 26071,
    26496, 17529, 26314, 18940, 16419), Rebounds = c(7074,
    6672, 9352, 14662, 15091, 8725, 7251, 7494, 3158),
    Assists = c(6306, 5633, 9298, 5445, 4225, 12091,
        3244, 6135, 4621), Games = c(1346, 1072, 1258,
        1462, 1392, 1391, 1114, 1178, 699), Points_per_game = c(25,
        30.1, 27.1, 17.8, 19, 12.6, 23.6, 16.1, 23.5))
```

```
NBA  # 查看该表格内容
```

```
##    Player Points Rebounds Assists Games Points_per_game
## 1    Kobe  33643     7074    6306  1346            25.0
## 2  Jordan  32292     6672    5633  1072            30.1
## 3   James  34087     9352    9298  1258            27.1
## 4 Garnett  26071    14662    5445  1462            17.8
## 5  Duncan  26496    15091    4225  1392            19.0
## 6    Kidd  17529     8725   12091  1391            12.6
## 7 Anthony  26314     7251    3244  1114            23.6
## 8  Pippen  18940     7494    6135  1178            16.1
## 9   Curry  16419     3158    4621   699            23.5
```

现在，我们根据球员的生涯总得分将球员进行分类，生涯总得分突破 1 万分的球员被归类为“一万分先生 ”，突破 2 万分被归类为“两万分先生 ”，突破 3 万分被归类为“三万分先生 ”。可通过如下代码实现:

```
NBA$Points_origin <- NBA$Points
# 保留原始数据列

NBA$Points[NBA$Points >= 30000] <- 3
NBA$Points[NBA$Points >= 20000 & NBA$Points < 30000] <- 2
NBA$Points[NBA$Points >= 10000 & NBA$Points < 20000] <- 1
NBA$Points <- factor(NBA$Points, levels = c(1, 2, 3),
    labels = c(" 一万分先生 ", " 两万分先生 ",
        " 三万分先生 "))
```

离散型变量转变为分类字符变量，其核心思想是首先转变将数值分类，再转换为因子，最后将因子转换为分类字符变量。factor() 函数作为一个转换因子的函数，在其中起到了中转站的作用，其格式为：

```
factor(x, levels, labels)
```

x 代表需要定义的变量，levels 代表值，labels 代表标签

反之，如果要将已经归类好的字符型变量转换回数值型变量，代码如下：

```
NBA$Points_numerical <- factor(NBA$Points, levels = c(" 一万分先生 ",
    " 两万分先生 ", " 三万分先生 "), labels = c(1,
    2, 3))
```

同理，根据球员的生涯总助攻数将球员进行分类，生涯总助攻突破 5000 的球员被归类为“助攻专家 ”，突破 9000 被归类为“助攻机器 ”，突破 15000 被归类为“助攻王 ”。可通过如下代码实现：

```
NBA$Assists_origin <- NBA$Assists
# 保留原始数据列

NBA$Assists[NBA$Assists <= 5000] <- 1
NBA$Assists[NBA$Assists > 5000 & NBA$Assists <= 9000] <- 2
NBA$Assists[NBA$Assists > 9000 & NBA$Assists <= 15000] <- 3
NBA$Assists <- factor(NBA$Assists, levels = c(1, 2,
    3), labels = c(" 助攻专家 ", " 助攻机器 ",
    " 助攻王 "))
```

导出 NBA 表格：

```r
library(flextable)
library(dplyr)
NBA_ft <- flextable(NBA)
NBA_ft <- flextable(NBA) %>%
    add_header_row(
    values = "NBA 球星核心数据统计表 ",
    # 将表头设置为标题，确保表格导出时标题可以被保留
    colwidths = ncol_keys(NBA_ft),
    top = TRUE) %>%  # 添加到表格顶部
  bold(i = 1, part = "header") %>%
  # 第一行表头加粗
  fontsize(i = 1, size = 14, part = "header") %>%
  # 表头增大字号
  bold(part = "header")  %>%
  # 表头加粗
  border(i = 1, part = "header",
         border.top = fp_border(width = 0, color = "transparent")) %>%
  # 移除第一行表头的上边框
  autofit() %>%
  add_footer_lines(" 数据来源:< http://www.stat-nba.com/ > 截至 2023 年 ") %>%
  align(align = "center", part = "all")

# 保存表格
save_as_image(
  NBA_ft, path = "/Users/jc/Documents/R export/image/NBA_ft.png")
```

```r
NBA_ft # 显示表格
```

使用者可根据自己的研究方向在相关的离散数值型数据库中运用以上方法对数据进行分层。

3.4 字符串处理的相关函数

3.4.1 串联字符

串联字符的代码为 `paste()` 函数，其格式：

```
paste("x", "y", sep=,collapse=)
```

x 代表需要串联的一个变量，y 代表需要串联的另一个变量，sep 代表串联完成后字符内的连接符，collapse 代表字符串间的连接符，一般用令它等于默认值 NULL。
现在我们将缙绅录数据库中官员的姓和名连接起来（缙绅录数据库照录原文，两者在数据库中是分开的），代码示例如下：

```
JSL1900_1912_clean$xingming <- paste(JSL1900_1912_clean$姓,
    JSL1900_1912_clean$名, sep = "", collapse = NULL)
```

除了姓和名之外，使用者还可以尝试将官员的“籍贯省”和“籍贯县”连接起来，比如“湖南”与“湘乡”连接为“湖南湘乡”，因代码逻辑一致，此处就不一一列举。

3.4.2 提取和判断字符

R 中的提取函数为 str_extract() 函数，用以提取需要但不能确定位置的记录。格式为：

```
str_extract(x, "y")
```

x 代表变量，y 代表需要提取的字符。
现在，用该函数提取出官职“員外郎”的官员，实例代码：

```
install.packages("stringr")
library(stringr)
str_extract(JSL1900_1912_clean$官职一, " 員外郎 ")
```

R 中的判断函数为 str_detect() 函数，用以判断某变量是否含有特定字符。格式为：

```
str_detect(x, "y")
```

x 代表变量，y 代表需要判断的字符。现在，用该函数判断官员的官职是否为“員外郎”，实例代码：

```
str_detect(JSL1900_1912_clean$官职一, " 員外郎 ")
```

提取字符函数的返回值只有需要提取的值和 NA，NA 不便处理；而判断字符函数返回值是 ture 和 false，是逻辑值。因此，在寻找特定字符时，常用判断函数。

熟悉 str_detect() 函数之后，就可以利用该函数结合 ifelse() 函数进行官职查找和判定。比如查找数据库中的“总督”群体，先利用 str_detect() 函数判断官职是否为“总督”；随后，给满足“总督”条件的记录赋值 1，否则赋值 0。这样，便可以对总督这一群体进行单独研究。实例代码如下：

```
JSL1900_1912_clean$zongdu <- ifelse(str_detect(JSL1900_1912_clean$官职一,
    " 總督 "), 1, 0)
```

同理，寻找出“知县”群体、“候补官员”群体、“额外官员”群体：

```
JSL1900_1912_clean$zhixian <- ifelse(str_detect(JSL1900_1912_clean$官职一,
    " 知縣 "), 1, 0)
JSL1900_1912_clean$houbu <- ifelse(str_detect(JSL1900_1912_clean$官职一,
    " 候補 "), 1, 0)
JSL1900_1912_clean$ewai <- ifelse(str_detect(JSL1900_1912_clean$官职一,
    " 額外 "), 1, 0)
```

运行以上代码后，我们就可以在区域一看到数据库增加了 zongdu、zhixian、houbu、ewai 四个变量，纵向变量下只有 1 和 0 的逻辑值，显示为 1 则表明某官员的官职为对应变量名称，比如，一官员的 zhixian 变量为 1，那表明该官员官职为知县。锁定了特定官员群体之后，便可进行更深入的分析。

3.4.3 替换字符

替换函数 gsub() 函数，用于替换字符，其格式为：

```
gsub("y", "z", x)
```

其中，y 代表需要替换的字符，z 代表替换成什么，x 代表变量。我们将旗分变量中的“鑲藍”字符，替换为“镶蓝”字符，示例如下：

```
gsub(" 鑲藍 ", " 镶蓝 ", JSL1900_1912_clean$旗分)
```

替换字符主要用于替换数据库中的异体字，因字符串数据库中繁体字、异体字、简体字互不相通，所以需要用替换函数进行统一。使用者可根据研究需要仿照代码示例替换掉字符中的异体字，总体逻辑一致，此处不一一列举。

第四章 制表

4.1 简单制表

简单制表函数 table（），格式为：

```
table(x, exclude = , useNA = ,)
```

x 为变量名，可以输入两个变量，用逗号（英文状态）并列。

exclude 为排除某一个类型的值，比如空白、缺失值、塗黑等特定字符；如果等于 NULL，就是不排除任何值；如果是 NA（缺失值），就是排除 NA 的值；如果等于特定字符，需要加上引号再输入特定字符。useNA 为选择是否统计 NA 的频数，如果等于 no 则为不统计；如果选择 ifany 或者 always 则为统计。

```
table(JSL1900_1912_clean$籍贯省, JSL1900_1912_clean$qiren,
    exclude = NULL, useNA = "always")
```

以上代码分别是一个变量制表、两个变量制表、两个变量加上一个参数制表的代码，使用者可尝试复现以上代码，并利用其它变量进行简单制表。

4.2 整理变量

在制作更加复杂的表格之前，需要对变量进行整理，因为大型的历史数据库是以字符串类型存储的，经常会有两种字符组合表达的意思相同，但却被归类为两种不同的类型。比如地名中，“京师 ”和“顺天 ”可归类为一个类型；“甘肃 ”和“甘肃 ”两者是繁体和简体的区别，应当归类为同一种类型。诸如此类情况，在历史数据库中比较多，我们需要利用 factor() 函数对变量进行简单地整理，factor() 函数在之前的转换变量环节也有涉及，它的格式为：

```
factor(x, levels = c(...) , labels = c(...))
```

x 为需要整理的变量。

levels 括号中为该变量中需要整理的字符，可以同时整理多个字符。用英文状态下引号引用，用逗号分隔。

labels 括号中为整理后的标签。根据 levels 整理出的变量，需要与 levels 中的变量一一对应，用英文状态下引号引用，逗号分隔。

现在，我们以缙绅录数据库中的“籍贯省 ”变量为例，“籍贯省 ”变量中由于记载版本的差异，出现很多异体字、繁体字混用的情况，我们将“籍贯省 ”统一整理为简体字；同时，“籍贯省 ”变量中，许多类型对应的记录数较少，我们需要化零为整，方便后续研究。示例代码如下：

```r
# 查看原始变量的状况
table(JSL1900_1912_clean$籍贯省)

# 生成一个新的变量，利用`factor()`函数对不同类型进行一一对应地整理。
JSL1900_1912_clean$jiguansheng_sort <- factor(JSL1900_1912_clean$籍贯省,
    levels = c("", "?? ", "? 西 ", " 奧國 ", " 比國 ",
        " 丹國 ", " 德國 ", " 俄國 ", " 法國 ", " 韓國 ",
        " 和國 ", " 美國 ", " 腦威國 ", " 葡國 ",
        " 日本國 ", " 瑞典國 ", " 義國 ", " 英國 ",
        " 安徽 ", " 奉天 ", " 福 ", " 福建 ", " 甘肅 ",
        " 广西 ", " 廣 ", " 廣? ", " 廣東 ", " 廣東駐防 ",
        " 廣西 ", " 貴州 ", " 漢軍 ", " 河南 ", " 黑龍江 ",
        " 湖北 ", " 湖南 ", " 吉林 ", " 江 ", " 江南 ",
        " 江蘇 ", " 江西 ", " 滿洲 ", " 蒙古 ", " 南 ",
        " 山东 ", " 山東 ", " 山西 ", " 陝西 ", " 顺天 ",
        " 順天 ", " 順天府 ", " 四川 ", " 天津 ",
        " 西 ", " 新疆 ", " 雲 ", " 雲南 ", " 浙江 ",
        " 直隸 "), labels = c("", " 不详 ", " 不详 ",
        " 外国 ", " 外国 ", " 外国 ", " 外国 ", " 外国 ",
        " 外国 ", " 外国 ", " 外国 ", " 外国 ", " 外国 ",
        " 外国 ", " 外国 ", " 外国 ", " 外国 ", " 外国 ",
        " 安徽 ", " 奉天 ", " 福建 ", " 福建 ", " 甘肅 ",
        " 广西 ", " 不详 ", " 不详 ", " 广东 ", " 广东 ",
        " 广西 ", " 贵州 ", " 不详 ", " 河南 ", " 黑龙江 ",
        " 湖北 ", " 湖南 ", " 吉林 ", " 不详 ", " 江南 ",
        " 江苏 ", " 江西 ", " 不详 ", " 不详 ", " 不详 ",
        " 山东 ", " 山东 ", " 山西 ", " 陕西 ", " 顺天 ",
        " 顺天 ", " 顺天 ", " 四川 ", " 天津 ", " 不详 ",
        " 新疆 ", " 云南 ", " 云南 ", " 浙江 ", " 直隶 "))
```

```r
options(width = 60)
matrix(runif(100), ncol = 20)
# 查看变量整理后的状况
table(JSL1900_1912_clean$jiguansheng_sort, JSL1900_1912_clean$qiren,
    exclude = NULL, useNA = "always")
```

这是一种比较基础的方法，需要手动识别，一一对应，适用于类型比较少的变量，比如“籍贯省”，整理出来差不多 30 余种类型；相反，“出身”这种变量，table() 函数运行之后区域三会出现数百种类型，就不太适合 factor() 函数整理。我们后续会介绍变量整理的简便方法（数据库整理），也会用到之前建议大家下载的数据库整理的便捷代码。变量整理完成后，区域三的表格效果将会变得更加简洁：

现在，我们利用同样方法整理“贡生”，“贡生”出现在“出身一”变量中，也就是说，我们仅仅对“出身一”中的一种种类进行整理。《清史稿 选举五》记载：贡生出身具体有恩贡、岁贡、优贡、拔贡、副贡以及廪贡、增贡、附贡等，贡生以恩、拔、副、岁、优为正途，其余皆为异途。根据《清史稿》，我们编写代码区分正途和异途贡生：

```
# 查看原始变量“出身一”中各类贡生的状况
table(JSL1900_1912_clean$出身一, JSL1900_1912_clean$qiren)

# 生成一个新的变量，利用`factor()`函数对不同类型进行一一对应地整理。
JSL1900_1912_clean$gongsheng <- factor(JSL1900_1912_clean$出身一,
    levels = c(" 挨貢 ", " 拔貢 ", " 拔貢生 ", " 撥貢 ",
        " 恩? ", " 恩貢 ", " 附貢 ", " 附貢生 ",
        " 副貢 ", " 副貢 ", " 副貢生 ", " 廩貢 ",
        " 廩貢 ", " 廩貢生 ", " 歲貢 ", " 歲貢 ",
        " 歲貢生 ", " 優貢 ", " 優貢生 ", " 增貢 ",
        " 增貢 ", " 增貢生 "), labels = c(" 正途贡生 ",
        " 正途贡生 ", " 正途贡生 ", " 正途贡生 ",
        " 正途贡生 ", " 正途贡生 ", " 异途贡生 ",
        " 异途贡生 ", " 正途贡生 ", " 正途贡生 ",
        " 正途贡生 ", " 异途贡生 ", " 异途贡生 ",
        " 异途贡生 ", " 正途贡生 ", " 正途贡生 ",
        " 正途贡生 ", " 正途贡生 ", " 正途贡生 ",
        " 异途贡生 ", " 异途贡生 ", " 异途贡生 "))
```

```
# 查看变量整理后的状况
table(JSL1900_1912_clean$gongsheng)
```

```
##
## 正途贡生 异途贡生
##    36400    33928
```

接着，我们整理“鼎甲”，“鼎甲”是清代科举名次前三者的统称。实例代码如下：

```
JSL1900_1912_clean$dingjia <- factor(JSL1900_1912_clean$出身一,
    levels = c(" 状元 ", " 榜眼 ", " 探花 "), labels = c(" 鼎甲 ",
        " 鼎甲 ", " 鼎甲 "))
```

确定“贡生”和“鼎甲”后，我们可以利用制表函数考察贡生群体和鼎甲群体的民族结构和地理来源：

```
# 展示“鼎甲”群体中的满汉比例
table(JSL1900_1912_clean$dingjia, JSL1900_1912_clean$qiren,
    exclude = NULL, useNA = "always")
```

```
##
##             旗      民    <NA>
##    鼎甲      76     786       0
##    <NA> 154797 483773       0
```

```
# 展示出身为正途贡生和异途贡生官员的满汉比例
table(JSL1900_1912_clean$gongsheng, JSL1900_1912_clean$qiren,
    exclude = NULL, useNA = "always")
```

```
##
##                  旗      民    <NA>
##    正途贡生     956   35444       0
##    异途贡生     326   33602       0
##    <NA>      153591 415513       0
```

```
# 展示出身为正途贡生和异途贡生官员的地理来源
table(JSL1900_1912_clean$jiguansheng_sort, JSL1900_1912_clean$gongsheng,
    exclude = NULL, useNA = "always")
```

```
##
##           正途贡生 异途贡生    <NA>
##              23545    26197 226418
##    不详          1        0     24
##    外国          0        0    693
##    安徽        999      497  24821
```

```
##    奉天          283        34    3729
##    福建          464       246   11653
##    甘肃          208        49    3059
##    广西          327       166    6352
##    广东          466       299   15155
##    贵州          526       252    7236
##    河南          716       446   14184
##    黑龙江         16         0       6
##    湖北          713       585   15679
##    湖南          781       482   24159
##    吉林           32        17     253
##    江南            1         0     103
##    江苏         1088       510   38062
##    江西          598       542   17168
##    山东          735       588   18498
##    山西          380       299    7052
##    陕西          582       207    6792
##    顺天          820       672   30444
##    四川          836       400   20747
##    天津            0         0       5
##    新疆            0         2       2
##    云南          402       180    5975
##    浙江         1025       541   50181
##    直隶          856       717   20654
##    <NA>            0         0       0
```

变量整理是制表一个基础的环节，使用者还可以仿照代码展示不同群体之间的联系，比如出身“贡生 ”官员群体取得“鼎甲 ”功名的具体数量；“鼎甲 ”功名获得者的地理分布等等。

4.3 制作可以导出的表格

4.3.1 利用 flextable 包导出二联表和三联表

在撰写论文的过程中，我们常常需要列出比较专业、简洁的表。前面介绍的 table() 函数会在区域三返回观测值，但需要复制粘贴，并重新制作表格样式，比较麻烦。使用 flextable 包可以导出专业的表格，以下代码介绍了如何利用缙绅录数据库和 flextable 包导出二联表格。

制图之前，需要用 `label()` 函数给前面创造的几个变量贴标签，确保表格导出时更美观。注意使用 `label()` 函数前请安装或加载 `dplyr` 包。同时将 `zhixian` 变量的判定值定义为“知县 ”和“非知县 ”，以准备数据。

还需要将所需数据转换为数据框，`as.data.frame()` 函数可以将 `table()` 函数制作的频数统计表格转换为数据框，列和行均会成为改造后数据框的变量，如果直接使用 `as.data.frame()` 函数，可能会得到一个不符合预期的结果，因此使用 `as.data.frame.matrix()` 函数，`as.data.frame.matrix()` 函数能够将 `table()` 的频数统计表格直接转换为矩阵数据框。

```
install.packages("table1")
library(table1)

label(JSL1900_1912_clean$jiguansheng_sort) <-" 籍贯省 "
label(JSL1900_1912_clean$qiren) <- " 旗人 "
label(JSL1900_1912_clean$gongsheng) <- " 贡生 "
label(JSL1900_1912_clean$zhixian) <- " 知县 "
JSL1900_1912_clean$zhixian[JSL1900_1912_clean$zhixian == 1] <- " 知县 "
JSL1900_1912_clean$zhixian[JSL1900_1912_clean$zhixian == 0] <- " 非知县 "

library(tibble)
Two_way_table_ft <- as.data.frame.matrix(
  table(JSL1900_1912_clean$jiguansheng_sort,JSL1900_1912_clean$qiren)) %>%
  rownames_to_column(var = " 省籍 ")
# `rownames_to_column()`函数可以插入行名作为数据框的第一列，
# 确保`fletable`包生成表格时显示行名

# 利用`flextable`包将生成的数据框转换为专业的数据表格
ft_1 <- flextable(Two_way_table_ft)
ft_1 <- flextable(Two_way_table_ft) %>%
  add_header_row(
    values = " 缙绅录数据库中官员的籍贯和满汉比例 ",
    # 将表头设置为标题，确保表格导出时标题可以被保留
    colwidths = ncol_keys(ft_1),
    top = TRUE) %>%
    # 添加到表格顶部
  bold(i = 1, part = "header") %>%
  # 第一行表头加粗
  fontsize(i = 1, size = 14, part = "header") %>%
```

```
  # 表头增大字号
  bold(part = "header")  %>%
  # 表头加粗
  border(i = 1, part = "header",
         border.top = fp_border(width = 0, color = "transparent")) %>%
  # 移除第一行表头的上边框
  add_footer_lines(" 数据来源: CGED-Q JSL Public Release 1900-1912") %>%
  # 注明数据来源
  autofit() %>%
  # 自动调整列宽
  align(align = "center", part = "all") %>%
  # 全部居中
  font(fontname = "Songti SC", part = "all")
# 如果是 MAC OS 系统，使用中文宋体

# 导出为 html 格式
save_as_html(
  ft_1,
  path =
    "/Users/jc/Documents/R export/table/缙绅录数据库中官员的籍贯和满汉比例.html"
  )
# 设置为你自己的路径

# 导出为图片
save_as_image(
  ft_1,
  path =
    "/Users/jc/Documents/R export/image/缙绅录数据库中官员的籍贯和满汉比例.png",
  zoom = 4)
# 设置为你自己的路径

# 导出为 docx 文件
save_as_docx(
  ft_1,
  path =
    "/Users/jc/Documents/R export/office/缙绅录数据库中官员的籍贯和满汉比例.docx"
```

```
  )
# 设置为你自己的路径

# 导出为 pptx 文件
save_as_pptx(
  ft_1,
  path =
    "/Users/jc/Documents/R export/office/缙绅录数据库中官员的籍贯和满汉比例.pptx"
  )
# 设置为你自己的路径
```

```
ft_1
```

表 2: 文官的籍贯和满汉比例（flextable）

缙绅录数据库中官员的籍贯和满汉比例

省籍	旗	民
	153,810	122,350
不详	2	23
外国	689	4
安徽	52	26,265
奉天	0	4,046
福建	11	12,352
甘肃	10	3,306
广西	5	6,840
广东	33	15,887
贵州	1	8,013
河南	32	15,314

表 2: 文官的籍贯和满汉比例（flextable）

缙绅录数据库中官员的籍贯和满汉比例

省籍	旗	民
黑龙江	0	22
湖北	29	16,948
湖南	5	25,417
吉林	0	302
江南	0	104
江苏	13	39,647
江西	9	18,299
山东	3	19,818
山西	3	7,728
陕西	17	7,564
顺天	59	31,877
四川	45	21,938
天津	0	5
新疆	0	4
云南	16	6,541
浙江	21	51,726
直隶	8	22,219

数据来源：CGED-Q JSL Public Release 1900-1912

以上是二联表格的制作和导出过程，但在缙绅录数据库的分析当中，我们有可能对多个变量进行操作。多变量表格的表现形式之一便是三联嵌套表，三联嵌套表通常以一个变量作为行变量，两个变量作为列变量，且列变

量一嵌套于列变量二之上。而这其中，嵌套关系的处理最为关键。flextable 包主要的功能是对表格进行优化，并不带有统计功能，所以我们在利用 flextable 包制作三联嵌套表时，需要对列变量的嵌套关系进行计算。在制作二联表时，我们运用了 jiguansheng_sort 和 qiren 两个变量，现在我们再引入一个 zhixian 变量制作三联表

首先需要准备数据，将 jiguansheng_sort 作为行名，qiren 和 zhixian 作为列变量，同时使 zhixian 嵌套于 qiren 之上，由于 qiren 和 zhixian 都是 0，1 值，所以嵌套格式下一共有 4 种组合，总体制表逻辑是先计算四种组合对应的统计数，再用 flextable 包制成三联表，示例如下：

```
Three_way_table_ft <- JSL1900_1912_clean %>%
  group_by(jiguansheng_sort) %>%
  # 按`jiguansheng_sort`对数据进行分组
  summarise(
    n_ 非知县 _ 民 = sum(qiren == " 民 " & zhixian == " 非知县 "),
    n_ 非知县 _ 旗 = sum(qiren == " 旗 " & zhixian == " 非知县 "),
    n_ 知县 _ 民 = sum(qiren == " 民 " & zhixian == " 知县 "),
    n_ 知县 _ 旗 = sum(qiren == " 旗 " & zhixian == " 知县 "),
    total = n()  # 计算四种组合对应的频数，并添加一列计算各行的频数之和
  ) %>%
  mutate( # 现在将百分比引入表格，使单元格的格式变为数量（百分比），
          # 百分比的计算公式为单元格统计数/总样本量
    `非知县 _ 民` = sprintf("%d (%.1f%%)",
                      n_ 非知县 _ 民, n_ 非知县 _ 民 / 639432 * 100),
    `非知县 _ 旗` = sprintf("%d (%.1f%%)",
                      n_ 非知县 _ 旗, n_ 非知县 _ 旗 / 639432 * 100),
    `知县 _ 民` = sprintf("%d (%.1f%%)",
                     n_ 知县 _ 民, n_ 知县 _ 民 / 639432 * 100),
    `知县 _ 旗` = sprintf("%d (%.1f%%)",
                     n_ 知县 _ 旗, n_ 知县 _ 旗 / 639432 * 100)
  ) %>%
  bind_rows( # 添加一个总计行，展示各列的频数之和
    summarise(.,
              jiguansheng_sort = " 总计 ",
              `非知县 _ 民` = as.character(sum(n_ 非知县 _ 民)),
              `非知县 _ 旗` = as.character(sum(n_ 非知县 _ 旗)),
              `知县 _ 民` = as.character(sum(n_ 知县 _ 民)),
              `知县 _ 旗` = as.character(sum(n_ 知县 _ 旗)),
```

```
            total = sum(total))
  ) %>%
  select(
    jiguansheng_sort, `非知县 _ 民`, `非知县 _ 旗`, `知县 _ 民`, `知县 _ 旗`, total
    ) # 精简表格

# 使用`flextable`包制成三联表
ft_2 <- flextable(Three_way_table_ft)
ft_2 <- flextable(Three_way_table_ft) %>%
  set_header_labels( # 设置各列的标签
    jiguansheng_sort = " 省籍 ",
    `非知县 _ 民` = " 民人 ",
    `非知县 _ 旗` = " 旗人 ",
    `知县 _ 民` = " 民人 ",
    `知县 _ 旗` = " 旗人 ",
    total = " 样本量 "
  ) %>%
  add_header_row( # 加入一行作为嵌套上层行，并设置名称，单元格宽度
    values = c("", " 非知县 ", " 知县 ", ""),
    colwidths = c(1, 2, 2, 1)
  ) %>%
  add_header_row( # 在表头加入一行作为标题行，并设置标题
    values = " 知县的地理来源与满汉比例 ",
    colwidths = ncol_keys(ft_2),
    top = TRUE) %>%
  bold(i = 1, part = "header") %>% # 第一行表头加粗
  bold(part = "header")  %>% # 表头加粗
  fontsize(i = 1, size = 14, part = "header") %>%
  border(i = 1, part = "header", # 设置表头上框线宽度为 0 并设置为透明
         border.top = fp_border(width = 0, color = "transparent")) %>%
  add_footer_lines(" 注：1）省籍第一行的空白代表无对应省籍记载
                   2）单元格百分比计算公式为单元格内统计数/总样本量
                   3）本表统计的是缙绅录数据库的记载数
                   4）数据来源：CGED-Q JSL Public Release 1900-1912") %>%
  autofit() %>% # 自动调整列宽
  align(align = "center", part = "all") %>% # 全部居中
```

```r
  font(fontname = "Songti SC", part = "all") # 如果是 MAC OS 系统，使用中文宋体

# 导出表格
save_as_image(
  ft_2,
  path = "/Users/jc/Documents/R export/image/知县的地理来源和满汉比例.png",
  zoom = 4)
```

```r
ft_2
```

表 3: 清代知县的满汉比例和地理来源

知县的地理来源与满汉比例

	非知县		知县		
省籍	民人	旗人	民人	旗人	样本量
	122125 (19.1%)	150437 (23.5%)	225 (0.0%)	3373 (0.5%)	276,160
不详	21 (0.0%)	2 (0.0%)	2 (0.0%)	0 (0.0%)	25
外国	4 (0.0%)	689 (0.1%)	0 (0.0%)	0 (0.0%)	693
安徽	22396 (3.5%)	52 (0.0%)	3869 (0.6%)	0 (0.0%)	26,317
奉天	3478 (0.5%)	0 (0.0%)	568 (0.1%)	0 (0.0%)	4,046
福建	9767 (1.5%)	11 (0.0%)	2585 (0.4%)	0 (0.0%)	12,363
甘肃	2364 (0.4%)	7 (0.0%)	942 (0.1%)	3 (0.0%)	3,316
广西	4901 (0.8%)	5 (0.0%)	1939 (0.3%)	0 (0.0%)	6,845
广东	14444 (2.3%)	33 (0.0%)	1443 (0.2%)	0 (0.0%)	15,920
贵州	6104 (1.0%)	1 (0.0%)	1909 (0.3%)	0 (0.0%)	8,014
河南	12718 (2.0%)	31 (0.0%)	2596 (0.4%)	1 (0.0%)	15,346
黑龙江	22 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	22

表 3: 清代知县的满汉比例和地理来源

知县的地理来源与满汉比例

	非知县		知县		
省籍	民人	旗人	民人	旗人	样本量
湖北	13835 (2.2%)	28 (0.0%)	3113 (0.5%)	1 (0.0%)	16,977
湖南	21828 (3.4%)	5 (0.0%)	3589 (0.6%)	0 (0.0%)	25,422
吉林	254 (0.0%)	0 (0.0%)	48 (0.0%)	0 (0.0%)	302
江南	103 (0.0%)	0 (0.0%)	1 (0.0%)	0 (0.0%)	104
江苏	34688 (5.4%)	13 (0.0%)	4959 (0.8%)	0 (0.0%)	39,660
江西	15161 (2.4%)	9 (0.0%)	3138 (0.5%)	0 (0.0%)	18,308
山东	17317 (2.7%)	3 (0.0%)	2501 (0.4%)	0 (0.0%)	19,821
山西	6455 (1.0%)	1 (0.0%)	1273 (0.2%)	2 (0.0%)	7,731
陕西	6172 (1.0%)	17 (0.0%)	1392 (0.2%)	0 (0.0%)	7,581
顺天	29746 (4.7%)	57 (0.0%)	2131 (0.3%)	2 (0.0%)	31,936
四川	18691 (2.9%)	44 (0.0%)	3247 (0.5%)	1 (0.0%)	21,983
天津	5 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	5
新疆	4 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	4
云南	5008 (0.8%)	16 (0.0%)	1533 (0.2%)	0 (0.0%)	6,557
浙江	45808 (7.2%)	17 (0.0%)	5918 (0.9%)	4 (0.0%)	51,747
直隶	19734 (3.1%)	8 (0.0%)	2485 (0.4%)	0 (0.0%)	22,227
总计	433153	151486	51406	3387	639,432

注：1）省籍第一行的空白代表无对应省籍记载

2）单元格百分比计算公式为单元格内统计数/总样本量

3）本表统计的是缙绅录数据库的记载数

4）数据来源：CGED-Q JSL Public Release 1900-1912

三联表的制作相对复杂，因为要涉及到对两个变量的嵌套组合，变量的类型越多，嵌套的组合也就越多，计算量也就越大。但是在缙绅录数据库这一类以字符为主的数据库中，几乎没有离散数据，更多的是逻辑判定值，因此，其总的计算量其实并不大。但数据统计的环节尤其重要，务必要保证数据的精确性。

4.3.2 利用 table1 包导出表格

除了 flextable 包以外，table1() 也可以制作出适应学术规范的表格，并展示在区域四，以供导出。table1() 的格式为：

```
table1(x, data, overall = , rowlabelhead = ,)
```

x 为需要制表的变量，如果是多个变量，用 + 号连接。| 号用来分层，即 | 号之后通常是列变量。* 号用于嵌套，嵌套两个逻辑值。比较复杂，后面可结合代码理解。

data 为所引用的数据集。

overall 为总计。需要在等号后添加内容，比如 overall、total 等。如果令其等于 F，则不显示总计的数值。

rowlabelhead 是抬头行标题，对应第一列变量，可以等于任意字符，需要添加引号。

现在我们尝试利用该函数制作可以导出的二联表格，实例代码如下：

```
library(table1)
table1(~jiguansheng_sort | qiren, data = JSL1900_1912_clean,
    overall = " 总计 ")
# 注意，前面的`~`符号不可省略，`|`用来分层，即`籍贯省`为行，`qiren`为列，两者可以颠倒顺序
```

```
library(table1)
table1(~jiguansheng_sort + gongsheng | qiren, data = JSL1900_1912_clean,
    overall = " 总计 ")
# 使用`+`号在行添加一个变量`gongsheng`，使`jiguansheng_sort`和`gongsheng`同时为行，`qiren`为列
```

table1() 包不仅可以在行上增加多个变量，也能够实现嵌套组合，进而制作三联表，示例代码如下：

```
library(table1)
table1(~jiguansheng_sort + gongsheng + dingjia | zhixian *
    qiren, data = JSL1900_1912_clean, overall = " 总计 ")
```

jiguansheng_sort、gongsheng 和 dingjia 为行，qiren 和 zhixian 将为列，且两者是嵌套关系，zhixian 嵌套在 qiren 之上，如需导出，可以直接在区域四 Viewer 区域点击 Export 导出为图片或者 html 格式。

在 table1() 包中，~ 号不可省略，| 号用来区分行变量和列变量，通常在 | 号之前，是表格中的行，| 号之后是表格中的列。+ 号用来连接多个行变量；* 号用来嵌套列变量。请注意符号的运用。

flextable 包和 table1() 包均能制作和导出表格，但两者又具备各自的优势和局限性，flextable 包的优势和局限性如下：

1. 优势 1. 样式定制灵活：支持单元格合并、边框控制、背景色、字体样式等精细化操作 2. 多格式输出：直接导出为 Word、PDF 等，无需手动调整格式 3. 支持复杂表格结构：支持嵌套表头、跨页表格、脚注等学术论文所需的高级功能 4. 适用范围广：可与 dplyr 等数据处理包无缝集成，适用于各类制表场景
2. 局限性 1. 统计功能弱：需手动计算统计量，再整合到表格中 2. 学习曲线陡峭：复杂样式需熟悉各类函数的利用 3. 无内置统计检验，依赖外部包生成统计结果

table1 包的优势和局限性如下：

1. 优势 1. 自动化统计描述：一键生成分层统计表，自动计算均值、标准差、频数、百分比等值，支持连续变量和分类变量 2. 分层嵌套展示：通过公式语法实现多级分组 3. 三线表规范输出：默认生成符合学术期刊要求的三线表格式，减少手动调整
2. 局限性 1. 样式定制有限：依赖自定义函数 render 修改表格布局，自定义函数的学习和运用较为复杂 2. 输出格式单一：主要支持 HTML，Word 需复制粘贴，可能丢失格式 3. 扩展性不足：无法生成非统计类表格

综上所述，两个包各自具有优势和局限性，使用者可根据运用场景进行切换。且两者并不是相互独立的关系，可以互补使用，兼顾效率和美观。

4.4 指定条件制表

在实际的制表中，我们有可能不需要整个数据库的数据，只需要某一部门或某一年份的数据即可。这种指定条件的数据分析需要用到 subset() 函数，格式如下：

```
subset(x, subset, select =c(...),)
```

x 为数据集名称，subset 为逻辑条件，即从数据集中选取数据的条件，必须是逻辑表达式，满足符号要求，可以用 &、| 号进行多条件筛选；select 为指定从相关的列中选择数据。现在，我们尝试提取满足单一条件的数据，示例代码如下：

```
# 清理数据
JSL1900_1912_clean$贡生 <- "0"
JSL1900_1912_clean$贡生 [JSL1900_1912_clean$gongsheng ==
    " 正途贡生 "] <- " 正途贡生 "
JSL1900_1912_clean$贡生 [JSL1900_1912_clean$gongsheng ==
    " 异途贡生 "] <- " 异途贡生 "
JSL1900_1912_clean$贡生 [JSL1900_1912_clean$贡生 ==
    "0"] <- " 非贡生 "

# 提取指定年份的数据
JSL1900_1912_only1910 <- subset(JSL1900_1912_clean,
    阳历年份 numeric == 1910)

# 用`table1()`函数在新创建的数据集基础上制表

table1(~jiguansheng_sort | qiren * 贡生, data = JSL1900_1912_only1910,
    overall = "total", rowlabelhead = "")
table1(~jiguansheng_sort + 贡生 + dingjia | qiren *
    季节, data = JSL1900_1912_only1910, overall = "total",
    rowlabelhead = "")

# 提取指定变量的数据
JSL1900_1912_only 顺天 <- subset(JSL1900_1912_clean,
    jiguansheng_sort == " 顺天 ")

# 制表检验提取数据结果
table1(~贡生 | qiren, data = JSL1900_1912_only 顺天,
    overall = "total", rowlabelhead = "")
```

subset() 函数还可以用于剔除缺失值，可剔除指定列中的缺失值，参考格式为

```
data<-subset(data,column_name!="NA")
```

以上讲述的指定单一条件，即要么年份是 1910，要么官员任职地区是顺天。如果需要指定多个条件，比如需要研究某个年份某个地区的官员群体，就需要用到 &，示例如下：

```
# 创建一个只有 1910 年顺天地区记录的数据集
JSL1900_1912_ 顺天 1910 <- subset(JSL1900_1912_clean,
    jiguansheng_sort == " 顺天 " & 阳历年份 numeric ==
        1910)

table1(~机构一 | qiren, data = JSL1900_1912_ 顺天 1910,
    overall = "total", rowlabelhead = "")

# 创建一个只有 1910 年顺天地区现任候补官员记录的数据集
JSL1900_1912_ 顺天 1910 候补 <- subset(JSL1900_1912_clean,
    jiguansheng_sort == " 顺天 " & 阳历年份 numeric ==
        1910 & houbu == 1)
```

```
table1(~机构一 | qiren, data = JSL1900_1912_ 顺天 1910 候补,
    overall = "total", rowlabelhead = "")  # 展示表格
```

在实际分析中，我们还会遇到条件多选一的情况，数据集设置多个条件只要满足其中一个即可的情况，这时需要用到 | 符号。

创建一个只有 1908 年之后在西南三省任职官员记录的数据集，西南三省包括云、贵、川三省。因此，在这三省任意一个省份任职的官员都可被纳入到新数据集中，代码中，我们用 |（或者）符号进行连接。示例如下：

```
JSL1900_1912_ 西南 after1908 <- subset(JSL1900_1912_clean,
    (jiguansheng_sort == " 四川 " | jiguansheng_sort ==
        " 贵州 " | jiguansheng_sort == " 云南 ") &
        (阳历年份 numeric > 1908))
```

```
table1(~籍贯省 | qiren * 季节, data = JSL1900_1912_ 西南 after1908,
    overall = "total", rowlabelhead = "")  # 展示表格
```

以上是本章制表的内容，这些都是比较基础的用法，如果有更高需求的使用者，可以在 RStudio 中键入`?flextable` 或者 `help("table1")` 查看帮助或者学习新的参数。

5.1.1 基础直方图

制作直方图推荐的程序包是 ggplot2 包，这是 R 语言中制图功能比较强大的包，使用者可以在区域四的 Packages 里面找到 ggplot2，也可以使用代码运行 ggpolt2 包：

```
install.packages("ggplot2")
library(ggplot2)
help("ggplot2")
```

制图包安装完成后，就可进行制图。ggpolt2 包制图分为两个步骤，第一步是输入 ggpolt() 函数针对数据集建立图层，再使用 geom_bar() 函数调整直方图的样式和内容，两个函数的格式如下：

1. g <- ggplot(data,mapping = aes(x, y, fill, color,...),...)
2. g + geom_bar(mapping,data,stat,position,just,width,,...)

首先是 ggpolt() 函数，其中的 data 为数据集。mapping = aes() 为建立图层，这里的 aes() 函数中必须包含必要的 X 轴变量，和 Y 轴变量。fill 为直方图图形的填充变量。color 为直方图颜色的填充变量。

geom_bar() 函数中的 mapping（图层）、data（数据集）都已经在前一条代码中确定，所以此处可以直接令其等于默认值 NULL。stat 参数有三种类型：其一是 count, 表示计算频数；其二是 identity, 表示按照 y 的值进行计数；其三是 bin, 表示对连续变量进行统计转换。Position 参数也有三种类型：其一是 stack, 堆积；其二是 fill, 且每个条形的最大值为 1；其三是 dodge, 分条展示。Just 表示条形距离中心的距离。Width 表示条形的宽度。以上是制作直方图函数中比较常用的几个参数，设置这几个参数后，便可以尝试制作简单的直方图，示例代码如下：

```
P_BarChart <- ggplot(JSL1900_1912_only1910, aes(jiguansheng_sort)) +
    geom_bar(color = "red", fill = "white", just = 0.5,
        width = 0.8) + theme(text = element_text(family = "STKaiti",
    size = 9))
```

这里用到的数据集 JSL1900_1912_only1910 是上一章指定条件制表时创建的数据集，现在我们用其来做图。图层里面选取的是“籍贯 ”这一变量作为横轴，纵轴默认使用 ”籍贯 ”的频数。theme() 函数可以更改图形的样式。边框颜色使用红色，填充设置为白色，条状图形居中，条状宽度 0.8cm。

```
P_BarChart  # 显示图片
```

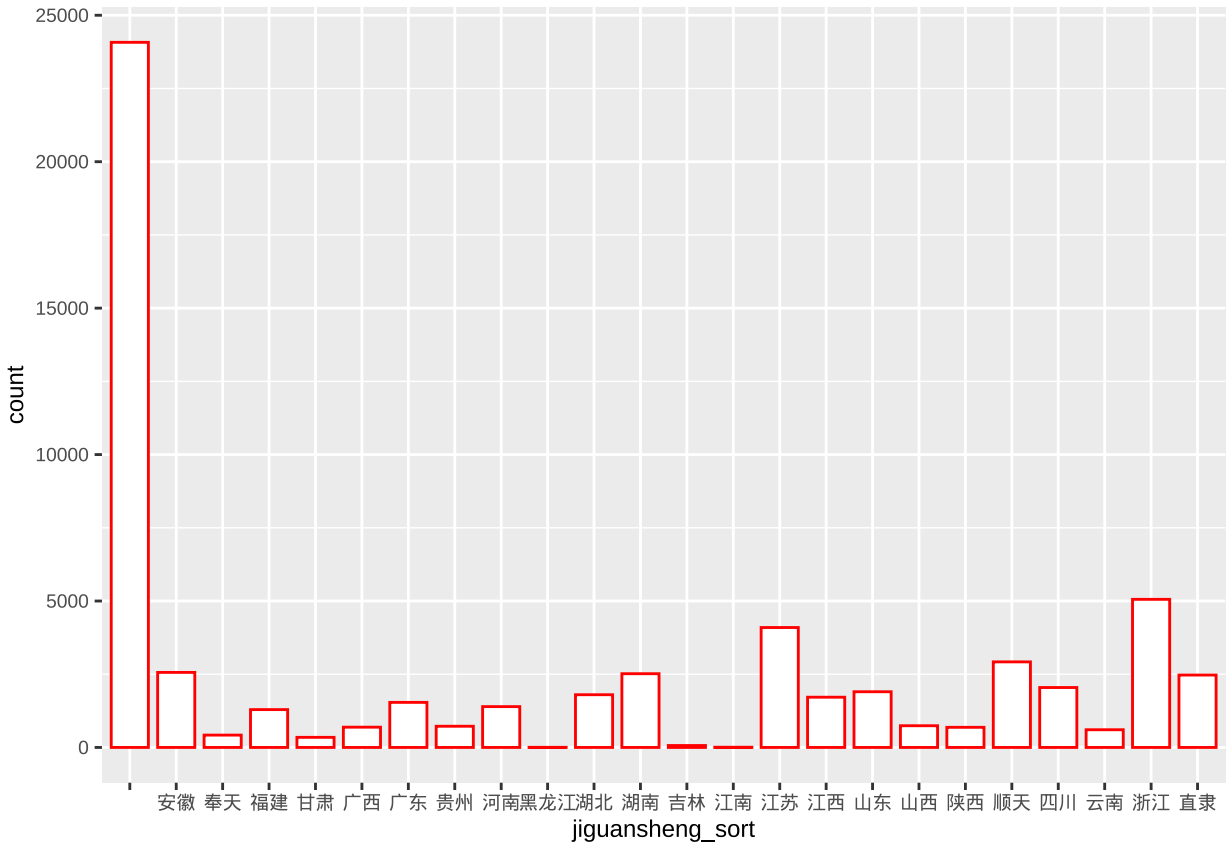

图 1: 缙绅录数据库中官员的地理来源

可以看出，这只是一个初步的图形，存在许多问题：首先，横轴字符较为拥挤，看不清楚；其次，横轴和纵轴的标题没有更改，且整个图形没有标题；最后，条形最高的一项只有引号，即没有内容（缙绅录当中，在籍贯所在省任职的官员，其籍贯一项留空不录，这是缙绅录的编纂体例导致的），这一项所含信息较多，如果单纯地将其处理为缺失值，会影响分析结果。简单直方图存在许多问题，需要进一步调整制图函数的参数，优化图形效果。

5.1.2 分条直方图

进阶直方图是在简单直方图的基础上控制更多函数的参数所制作出的更美观、更准确的图形。进阶直方图分为多种类型：分条直方图、堆积直方图。这一节主要讲分条直方图，在制作分条直方图前，我们需要先建立一个数据集再制图。

```
# 指定条件提取只在兵部衙门任职的官员，以该条件为基础生成数据集，
JSL1900_1912_bingbu <- subset(JSL1900_1912_clean, 机构一 == " 兵部衙門 ")
```

```r
# 套用兵部衙门数据集，查看 1900-1906 兵部衙门的满汉比例
P_BarChart_Dodge <- ggplot(JSL1900_1912_bingbu,aes(阳历年份 numeric, fill = qiren)) +
  geom_bar(position = "dodge",just= 0.5,width = 0.5)+
  labs(x = " 年份 ", y = " 记载数 ",title = "1900-1906 年兵部的满汉比例 ") + # 可设置横轴、纵轴、整个图形的标题
  guides(fill=guide_legend(title = " 旗人 ")) + # 可设置图例的标题。观察图 5.2 的图例，图例的标题已将按照代码的要求被设置成 " 旗人 "
  scale_fill_manual(values = c("cyan","pink")) + # 可设置条形的颜色，图中，民人和旗人颜色分别为青色和粉红色
  theme(text = element_text(family = "STKaiti",size = 9), # 能处理图例颜色、图例样式、网格间距、网格颜色等等修饰图形的细节，这一参数可以设置的细节众多
    plot.title = element_text(size = 15,hjust = 0.5,color = "orange"),
        axis.text.x = element_text(size = 9,angle = 30,color = "black", margin = margin(t = .5, unit = "cm")),
        axis.text.y = element_text(size = 9),
        axis.title.x = element_text(size = 10,hjust = 0.5,vjust = 0),
        axis.title.y = element_text(size = 10),
        legend.position = "bottom",
        legend.title = element_text(size = 12,hjust = 0.5,color = "orange" ),
        legend.background = element_rect(color = "black",fill = "grey90", linewidth = 1),
        legend.text = element_text(size = 10,color = "blue"),
        legend.key = element_rect(linewidth = 2,color = "purple"),
        panel.background = element_rect(color = "grey50"),
        panel.grid = element_line(color="grey50",linewidth = 0.1)) +  # Mac 系统制图后如无法显示中文字体，请在此添加这条代码 "text = element_text(family = "STKaiti",size = 9),", windows 系统请忽略后同。
  scale_x_continuous(expand = c(0.1,0.1),breaks = seq(1900,1906,1)) + # 可调整 X 轴与 y 轴的最大最小值、间隔等等
  scale_y_continuous(limits = c(0,1000),breaks = seq(0,1000,100)) +
  geom_text(aes(label = ..count..,family = "serif"), stat = "count",size = 3.5, # 可设置直方图数值标签的统计方式、大小、间隔、位置等
            vjust = -0.8,
            hjust = 0.5,
            color = "red2",
            position = position_dodge(0.5))
```

```
P_BarChart_Dodge  # 显示图片
```

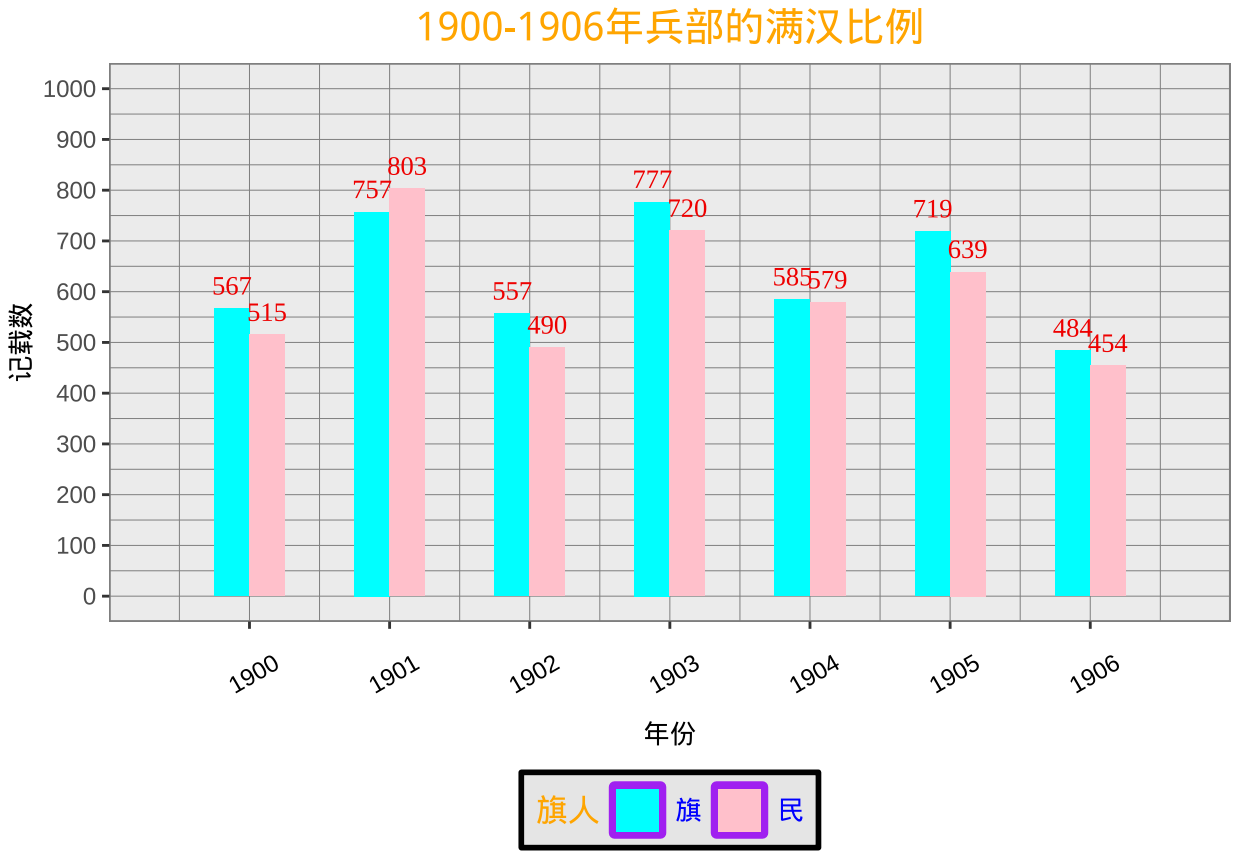

图 2: 1900-1906 年兵部官员的满汉比例分条直方图

以上代码是直方图的初步图形，我们在整个代码中加入一个修饰函数 `theme()`，使用方法就是在在原始代码的后面加入加上 `+` 号，将整个 `theme()` 函数添加到代码中。这一函数能处理图例颜色、图例样式、网格间距、网格颜色等等修饰图形的细节。

`theme()` 函数有非常多参数，代码中提到的只是几个比较常用的。值得注意的是，`theme()` 函数中的参数必须与 `element_text()` 函数合用，`element()` 系列函数有 4 个：`element_text()`、`element_line()`、`element_rect()`、`element_blank()`。使用者必须根据参数的不同而调整 `element()` 函数的使用，也可利用 help 功能查看具体用法。

使用者可根据自己的喜好设置图形的细节。在之后的图形制作讲解中，为了使代码看起来不冗长，我们不再标注 `theme()` 函数的具体设置数值，如果有意学习 `theme()` 函数的使用者，可以在控制台键入`?theme`，或者登录 `ggplot2` 官网，学习 `theme()` 的语法。

5.1.3 堆积直方图

堆积直方图相较于分条直方图更适合研究比例问题。堆积直方图和分条直方图的代码大体相似，但在一些参数的设置上必须要细致。示例如下：

```
P_BarChart_Fill <- ggplot(JSL1900_1912_bingbu,aes(阳历年份, fill = qiren)) +
  geom_bar(position = "fill",color="black",just= 0.5,width = 0.5)+
  labs(x = " 年份 ", y = " 百分比 ",title = "1900-1906 年兵部的满汉比例 ")+
  guides(fill=guide_legend(title = " 旗人 "))+
  scale_fill_manual(values = c("cyan","pink"))+
  theme(text = element_text(family = "STKaiti",size = 9),
        plot.title = element_text(size = 15,hjust = 0.5,color = "orange"),
        axis.text.x = element_text(size = 9,angle = 30,color = "black", margin = margin(t
        ↪  = .5, unit = "cm")),
        axis.text.y = element_text(size = 9,angle = -30, margin = margin(r = 0.2, unit =
        ↪  "cm")),
        axis.title.x = element_text(size = 10,hjust = 0.5, margin =
        ↪  margin(0.2,0.1,0.2,0.1,"cm")),
        axis.title.y = element_text(size = 10,angle = 90,vjust = 0.5, margin =
        ↪  margin(0.1,0.5,0.1,0.5,"cm")),
        legend.position = "bottom",
        legend.title = element_text(size = 12,hjust = 0.5,color = "orange" ),
        legend.background = element_rect(color = "black",fill = "grey90", linewidth = 1),
        legend.text = element_text(size = 10,color = "blue"),
        legend.key = element_rect(size = 2,color = "purple"),
        panel.background = element_rect(color = "grey50"),
        panel.grid = element_line(color="white",linewidth = 0.5))+
  scale_y_continuous(labels = scales::percent_format(accuracy = 1))+
  geom_text(aes(label=paste0(sprintf("%1.1f",
                             ..count.. / tapply(..count.., ..x.., sum)
                             [as.character(..x..)]*100),
                             "%"),## 引用自：https://stackoom.com/question/43QCB
              family = "serif"),
          stat = "count",
          size = 5,
          vjust = -0.8,
```

```
          hjust = 0.5,
          color = "gray10",
          position = position_fill(0.5))
```

```
P_BarChart_Fill  # 显示图片
```

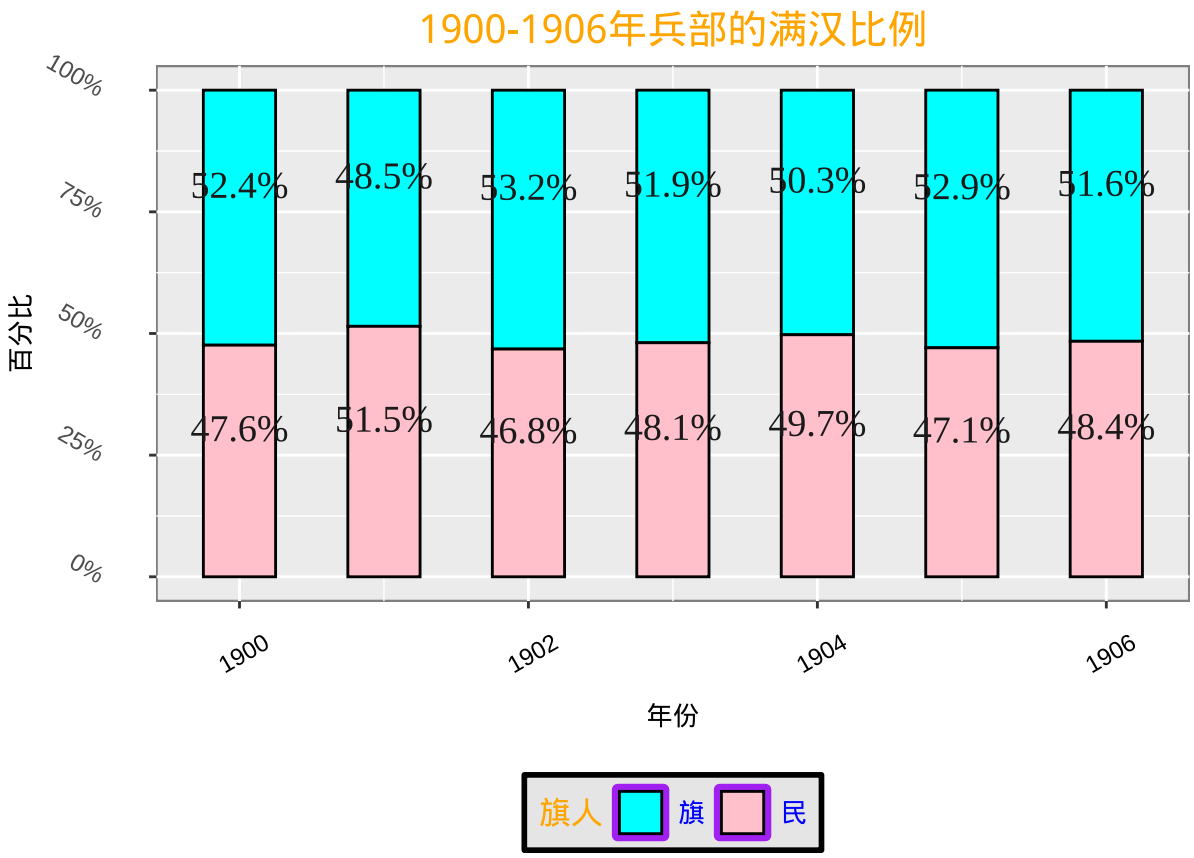

图 3: 1900-1906 年兵部官员的满汉比例堆积直方图

对比分条直方图，堆积直方图主要有两个地方不同：第一个是 `position` 变量变成了 `fill`；第二个是 `geom_text()` 的 `label` 出现了嵌套函数，如果直接套用分条堆积图的代码，会导致每个条形比例之后不是 100%，所以在这里我们运用了 `paste0()` 的嵌套函数。

制作堆积直方图的代码较分条直方图较为复杂，使用者可根据需要直接套用代码。以上为进阶直方图制作的具体流程，即在简单直方图函数上加上主题修饰函数 `theme()` 以及值标签制作函数 `geom_text()`、另外可根据制图需求选择加上调整横、纵坐标间隔的 `scale()` 函数等。

5.1.4 利用非数值型变量和离散型变量制作直方图

由于缙绅录数据库中缺少离散型变量，所以这里利用的数据集是第三章第三节提到的 NBA 数据集。其实离散型变量整体的代码与前面的相似，不过需要更改一些参数：geom_bar() 中需要添加一个 y 变量，同时，stat 参数需要更改为 identity，其默认为 count，因为在这里我们添加了 y 变量，要用 y 的值来作为计数标准，所以选择 identity。此外，geom_text() 中的 aes() 里的 label 参数要更改为与 y 轴一样的变量，代码示例如下：

```
P_NBA_BarChart <- ggplot(NBA, aes(Player, Points_per_game)) +
    geom_bar(color = "black", fill = "white", stat = "identity",
        just = 0.5, width = 1) + xlab("Players") +
    ylab("Points_per_game") + theme(axis.text.x = element_text(size = 8),
    axis.text.y = element_text(size = 8), axis.title.x = element_text(size = 13),
    axis.title.y = element_text(size = 13)) + geom_text(aes(label = Points_per_game),
    vjust = -0.8, hjust = 0.5, color = "red")
```

```
P_NBA_BarChart  # 显示图片
```

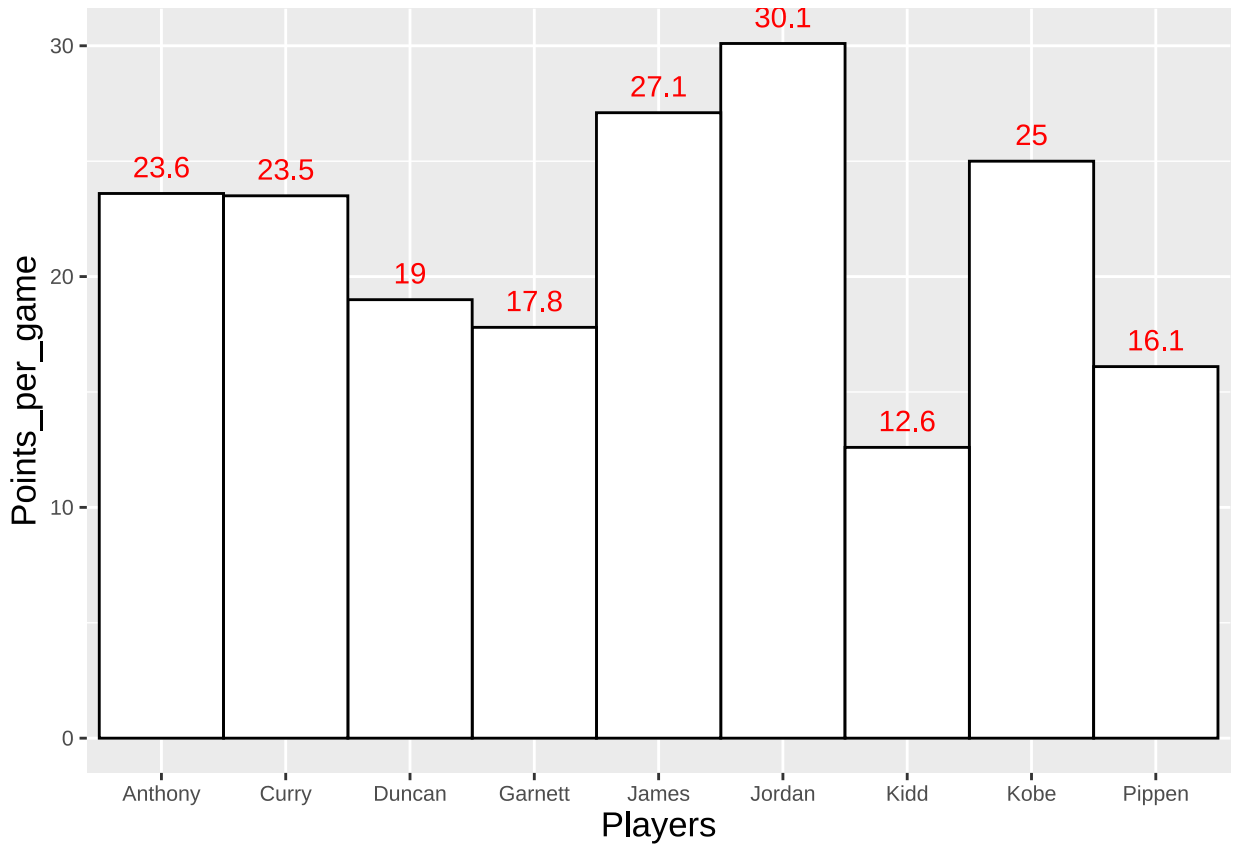

图 4: NBA 球员的场均得分分条直方图

离散型数据制作直方图等图形相较于连续型数据较为方便，且离散型数据库的数量要远大于连续型数据库。因此，学习离散型数据库的制图方法也比较重要。使用者可根据自身需要结合代码和网络资源学习离散型数据制作图形的进阶方法。

5.2 散点图

散点图的制作流程和直方图类似，在建立图层的基础上套用 geom_point() 函数即可。在制作散点图之前，我们需要制作一个数据集来展现不同品级官员的满汉比例，以此来观察品级高低与官员满汉比例的关系。在进行制图之前，需要准备数据，创建一个数据集。这个数据集中包含“尚书 ”（从一品）、“左侍郎 ”（二品）、“右侍郎 ”（二品）、“郎中 ”（五品）、“員外郎 ”（从五品）、“主事 ”（六品）、“七品京官 ”（七品）等 7 个官职的任职官员。整体的思路是：首先创造一个空变量“官职品级 ”；然后利用 str_detect() 判定函数，找出官职是“尚书 ”的记录；再在“官职品级 ”变量中填入“尚书 ”（只在“尚书 ”官职判定为 TRUE 的行记录上），并利用相同思路，填入其他官职名称；最后利用 factor 函数以“官职品级 ”为基础创造新变量“品级 ”并分层。

代码如下：

```
JSL1900_1912_bingbu$官职品级 <- ""
尚书 <- str_detect(JSL1900_1912_bingbu$官职一," 尚書 ")
JSL1900_1912_bingbu$官职品级 [尚书 == TRUE] <- " 尚书 "
左侍郎 <- str_detect(JSL1900_1912_bingbu$官职一," 左侍郎 ")
JSL1900_1912_bingbu$官职品级 [左侍郎 == TRUE] <- " 左侍郎 "
右侍郎 <- str_detect(JSL1900_1912_bingbu$官职一," 右侍郎 ")
JSL1900_1912_bingbu$官职品级 [右侍郎 == TRUE] <- " 右侍郎 "
主事 <- str_detect(JSL1900_1912_bingbu$官职一," 主事 ")
JSL1900_1912_bingbu$官职品级 [主事 == TRUE] <- " 主事 "
員外郎 <- str_detect(JSL1900_1912_bingbu$官职一," 員外郎 ")
JSL1900_1912_bingbu$官职品级 [員外郎 == TRUE] <- " 員外郎 "
郎中 <- str_detect(JSL1900_1912_bingbu$官职一," 郎中 ")
JSL1900_1912_bingbu$官职品级 [郎中 == TRUE] <- " 郎中 "
七品京官 <- str_detect(JSL1900_1912_bingbu$官职一," 七品 ")
JSL1900_1912_bingbu$官职品级 [七品京官 == TRUE] <- " 七品京官 "
JSL1900_1912_bingbu$品级 <- factor(JSL1900_1912_bingbu$官职品级,
                                levels = c(" 尚书 "," 左侍郎 "," 右侍郎 "," 郎中 "," 員外郎
                                 ↪  "," 主事 "," 七品京官 "),
                                labels = c(1.5,2,2,5,5.5,6,7))

# 制作散点图
```

```
P_ScatterChart <- ggplot(data=subset(JSL1900_1912_bingbu, !is.na(品级)), # 利用`!is.na()`函
↪  数剔除缺失值
        aes(阳历年份, 品级,shape = qiren,colour = qiren)) +
  geom_point(position = "jitter") + # 利用`jitter`添加抖动，防止点重合
  labs(title = "1900-1906 年兵部官员的品级及数量 ")+
  guides(shape=guide_legend(title = " 旗人 "),colour=guide_legend(title = " 旗人 "))+
  scale_colour_manual(values = c("gold","black"))+
  theme(text = element_text(family = "STKaiti",size = 9),
        plot.title = element_text(size = 15,hjust = 0.5,color = "orange"),
        axis.text.x = element_text(size = 9,angle = 30,color = "black", margin = margin(t
        ↪  = .5, unit = "cm")),
        axis.text.y = element_text(size = 9, margin = margin(t = .5, unit = "cm")),
        axis.title.x = element_text(size = 10,hjust = 0.5,vjust = 0),
        axis.title.y = element_text(size = 10),
        legend.position = "bottom",
        legend.title = element_text(size = 12,hjust = 0.5,color = "orange" ),
        legend.background = element_rect(color = "black",fill = "grey90", linewidth = 1),
        legend.text = element_text(size = 10,color = "blue"),
        legend.key = element_rect(size = 1,color = "purple"),
        panel.background = element_rect(color = "grey50"),
        panel.grid = element_line(color="white",linewidth = 0.5))
```

相较于制作直方图，散点图有以下改动：第一，加入了`!is.na()` 函数剔除掉“品级 ”变量的缺失值，如果不剔除缺失值，RStudio 会报错；第二，`shape` 参数可以按所指定变量的不同类型区分形状；第三，`colour` 可以按所指定变量的不同类型区分颜色；第四，`postition` 参数改为了 `jitter`, 可以让散点抖动，由于设置的品级是连续型数值，如果不添加抖动，同一品级的所有观测值会集中在一个点上，会造成混淆，所以必须添加散点抖动。

```
P_ScatterChart  # 显示图片
```

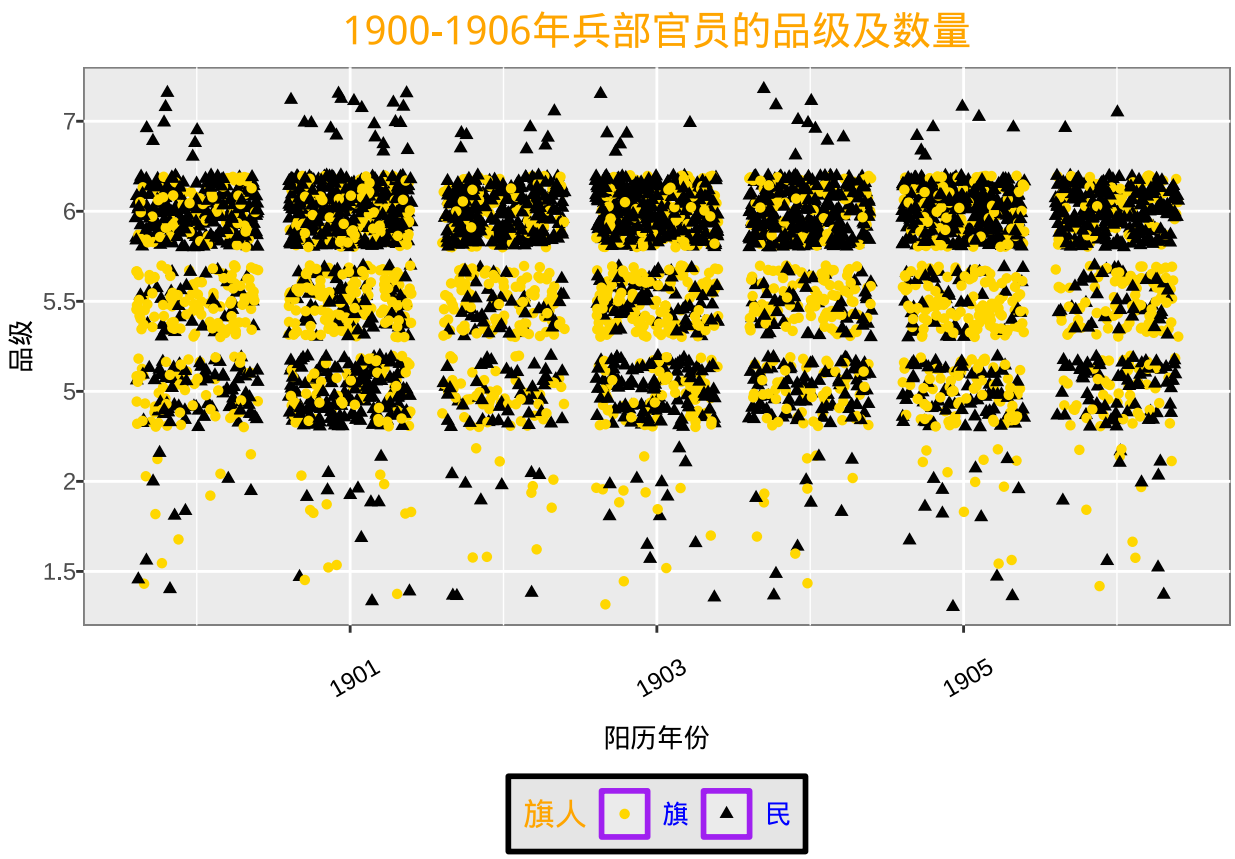

图 5: 1900-1906 年兵部官员的品级与数量

从上图中可以看出，在中下层官员中，担任兵部员外郎的旗人数量比较多；担任兵部郎中、兵部主事的民人数量较多；没有旗人担任七品京官。高层官员中，满汉比例几乎持平。这是一张比较清晰的散点图，使用者可以套用代码研究其它部门官职品级和满汉比例的关系。

离散数据制作散点图较为简单，省去了准备数据的环节，只需要指定 x、y 值，我们还是用 NBA 数据集来举例子，示例代码如下：

```
P_NBA_ScatterChart <- ggplot(NBA, aes(Rebounds, Assists_origin,
    colour = Player)) + geom_point() + theme(text = element_text(family = "STKaiti",
    size = 9)) + geom_text(aes(label = Player), size = 5,
    vjust = 1.5, hjust = 0.5)
```

```
P_NBA_ScatterChart  # 显示图片
```

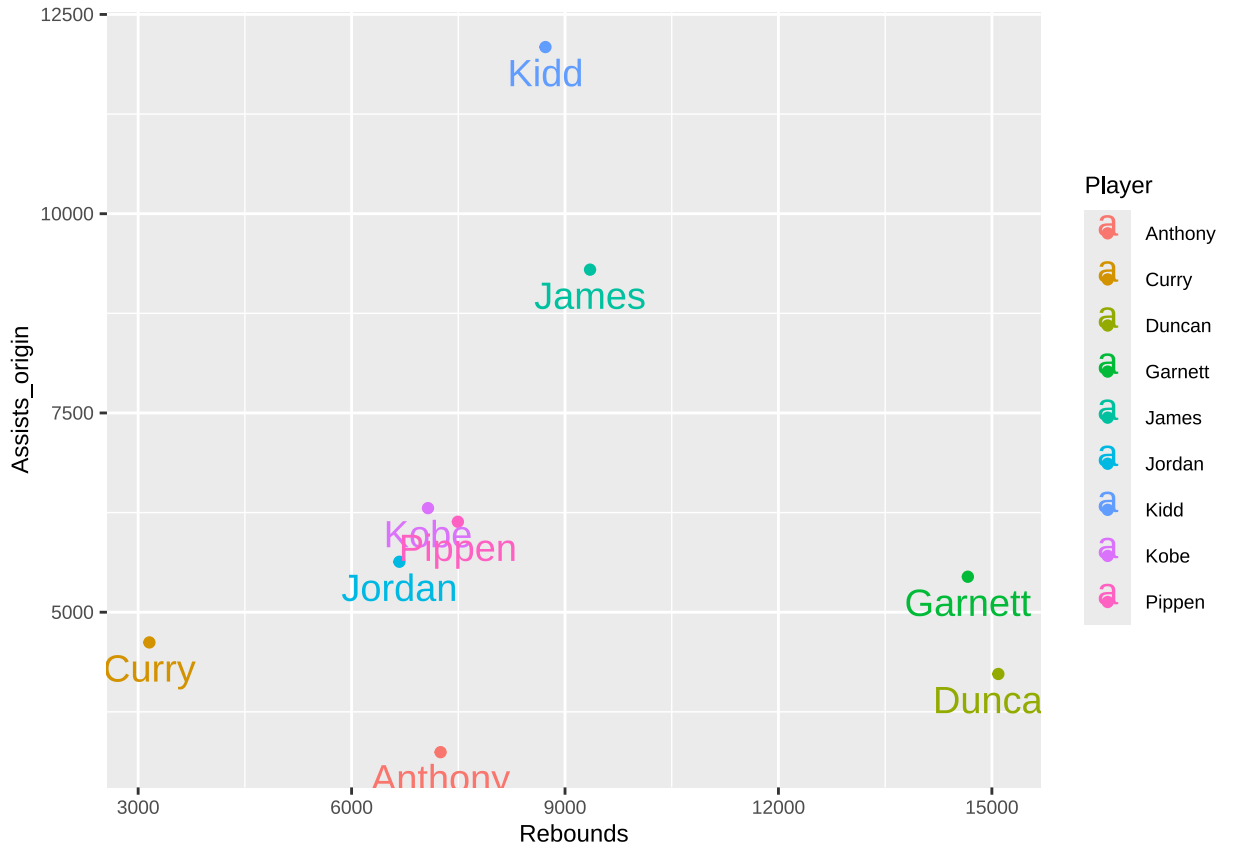

图 6: NBA 球员的篮板助攻比散点图

这个图形反映了 NBA 球员助攻篮板比的分布，越靠近右上区域的球员助攻篮板比的比例越大。使用者可根据以上代码在离散型数据集上尝试制作散点图。

5.3 折线图

折线图的制作逻辑是：先创造散点图，形成点状图层；再制作折线图，将点状图层相连接，即形成折线图。折线图对数据的要求更高，x、y 轴任意一轴需要是连续型变量，而另外一轴必须是离散型变量或连续型变量。JSL 中缺少相应的离散变量，但可以根据数据计算出新的离散型变量，比如满汉比例。

现在，我们尝试使用折线图来展现 1900-1906 年之间兵部衙门官员满汉比例的变化。首先利用已知的 1900-1906 年兵部官员的满汉比例制作一个数据集，创造 `year`、`percentage`、`shenfen` 三列数据，并将这三条列数据整合为一个数据集。请注意，`shenfen` 与 `percentage` 一一是对应的，方便后面制图。

```
year <- c("1900","1901","1902","1903","1904","1905","1906")
percentage <- c(52.4,48.5,53.2,51.9,50.3,52.9,51.6,47.6,51.5,46.8,48.1,49.7,47.1,48.4)
shenfen <- c(" 旗 "," 旗 "," 旗 "," 旗 "," 旗 "," 旗 "," 旗 "," 民 "," 民 "," 民 "," 民 ","
↪  民 "," 民 "," 民 ")
```

```r
bingbu1900_1906manhanbili <- data.frame(year,percentage,shenfen)

# 数据集完成后，画出折线图
P_LineChart <- ggplot(bingbu1900_1906manhanbili,
            aes(x = year, y = percentage,group = shenfen,
                colour = shenfen, shape= shenfen)) +
# x、y 轴有明确的指向，"group" 表示按变量类型对散点进行分组，
# 如不分组，就会默认为全部点都在同一个组内，但这样就不能进行线段比较，
# 另外，按照 "shape" 区分颜色和点的类型，防止点重合
  geom_point(size=2.5,stroke=0.5) +
  geom_line() +
  scale_color_manual(values = c("gold","black")) +
  scale_shape_manual(values = c(1,5)) +
# `geom_line()`函数可以在散点图层制作完毕后，加入线段进行连接；
# `scale_shape_manual()`可更改点的样式
  labs(x = "Year",y= "Percentage",title = "1900-1906 兵部衙门满汉比例折线图 ") +
  theme(text = element_text(family = "STKaiti",size = 12),
        legend.title = element_blank(),
        legend.position.inside = c(0.2,0.9),
        legend.direction = "horizontal",
        legend.background = element_rect(color="grey90",fill="grey90"),
        axis.text = element_text(color = "black",family = "serif"),
        axis.title = element_text(family = "serif",size = 18,color = "black"),
        plot.title = element_text(size = 15,hjust = 0.5,color = "orange"),
        axis.text.x = element_text(size = 9,angle = 30,color = "black", margin = margin(t
        ↪  = .5, unit = "cm")),
        axis.text.y = element_text(size = 9, margin = margin(t = .5, unit = "cm")),
        axis.title.x = element_text(size = 10,hjust = 0.5,vjust = 0),
        axis.title.y = element_text(size = 10,angle = 90,vjust = 0.5, margin =
        ↪  margin(0.1,0.5,0.1,0.5,"cm")),
        panel.background = element_rect(colour = "white",fill = "grey90"),
        legend.key = element_rect(color = "grey90",fill="grey90"))+
  scale_y_continuous(limits = c(45,55),breaks = c(46,48,50,52,54), labels =
   ↪  c("46%","48%","50%","52%","54%"))+
  geom_text(aes(label=paste0(percentage,"%")),
# `geom_text()`函数中的 "label=paste0(percentage,"%"" 用到了粘贴函数`paste0()`,
```

```
# 将百分比与% 符号粘贴在一起
            family = "serif",
            stat = "identity",
            size = 4,
            vjust = 1.5,
            hjust = 0.5,
            color = "red2",
            position = position_dodge(0.1))
```

```
P_LineChart   # 显示图片
```

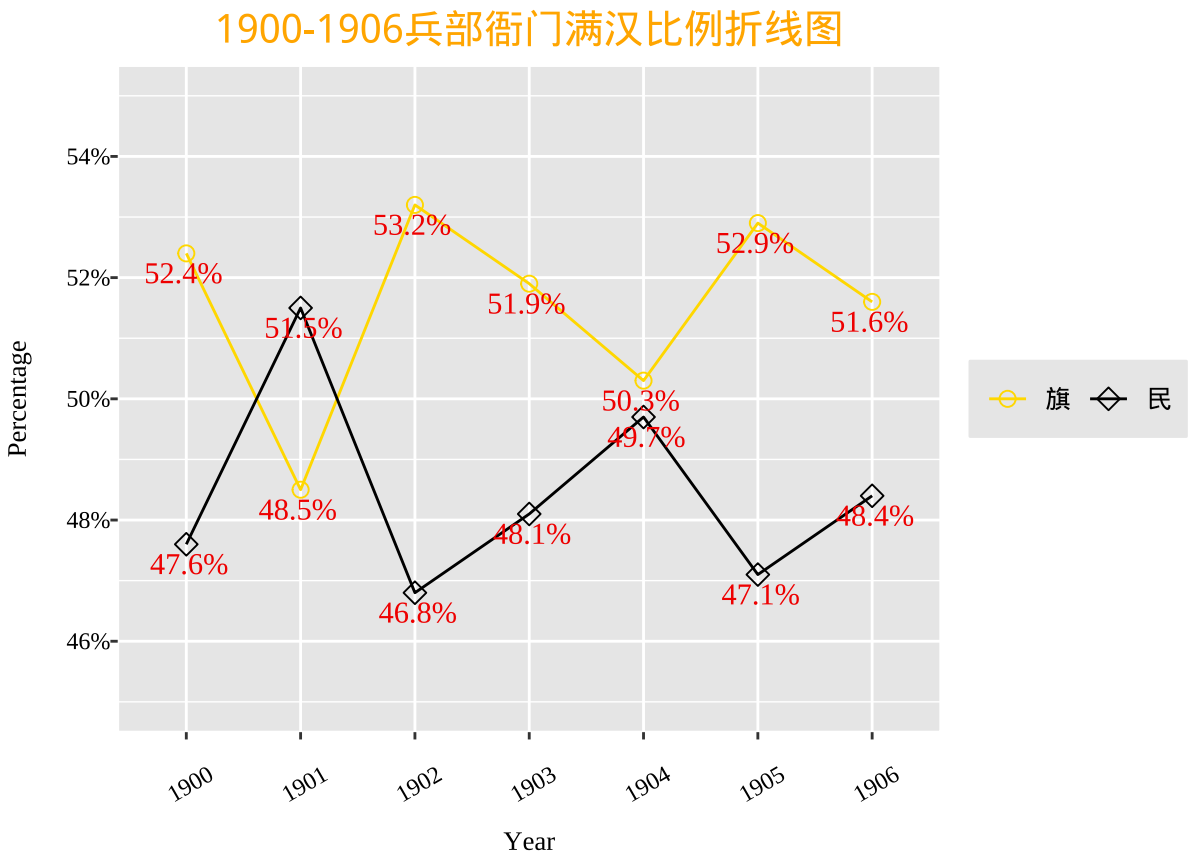

图 7: 1900-1906 年兵部官员满汉比例折线图

以上即为三大基本图形的简单制作流程。R 中的 ggpolt2 包制作图形的方式如出一辙：首先建立图层，再在图层上加上需要建立的图形函数 `geom_bar()`、`geom_point()`、`geom_line()`、最后对建立的图形进行修饰并加上值标签。如使用者有更多的制图需求，可以查看 `ggplot2` 包的帮助。

第六章 数据集的匹配

6.1 筛选列与行

6.1.1 筛选列

从数据库中筛选列的函数是 select()，可以挑选出指定条件的列变量。select() 函数是 base R（一个函数聚合项目）的一个筛选函数，其实在 tidyverse 项目中也有相关的筛选函数，但 base R 的筛选函数更加简短、更加快捷、便利理解，适合初学者使用，因此，我们这里使用的是 base R 的初阶筛选函数。select() 的格式为：

```
select(.data,...)
```

data 为数据集，…为需要选择的列，可以套用参数，该函数常用的参数有：last_col()：表示选择最后一列。

starts_with()：表示以什么开头的列。

ends_with()：表示以什么结尾的列。

contains()：表示某列是否包含什么内容。

缙绅录数据库中有 50 余个变量，但有时候为了分析方便，我们往往只需要其中一部分变量，比如姓名、官职、籍贯、出身、任职地等重要变量，这时就需要用到 select() 函数。示例如下：

```
JSL1900_1912_clean %>%
    select(阳历年份，地区，机构一，官职一，
        姓，名，籍贯省，籍贯县，旗分，出身一) ->
    JSL1900_1912_simplify
```

%>% 为管道符，可以将某一个数据集的行或列转移到另一个数据集，在第四章第三节介绍导出表格时用到过。利用管道符和 select() 函数，可以截取某一个数据集的部分变量生成新的数据集。“阳历年份…出身一 ” 为需要转移的列变量的名称，注意，这里不需要再指定数据集，因为管道符的有一个作用类似于 attach() 函数，运行 attach() 函数后，在之后的分析中就无需每次都指定代码集，RStudio 会默认使用 attach() 函数指定的代码集，但很多使用者在数据集分析之后会忘记取消 attach() 函数中指定的代码集，在其他分析中还是套用原数据集，影响分析结果，故不推荐使用。JSL1900_1912_simplify 为转移的/生成的数据集的名称。

以上代码运行之后，我们便创建了一个只有 10 个列变量的数据集，方便后面的分析。

6.1.2 筛选行

筛选行的 `filter()` 函数，可以以挑选出指定条件的列，其用法与 `select()` 函数一样，格式为：

```
filter(.data,...)
```

代码示例为：

```
JSL1900_1912_clean %>%
    filter((阳历年份 numeric >= 1900) & (阳历年份 numeric <=
        1906) & (机构一 == " 翰林院衙門 ")) ->
    JSL1900_1906_hanlinyuan
```

“阳历年份 numeric...” 是筛选的条件，需要使用者注意连接符号的运用。

无论是筛选行还是筛选列，都需要注意`%>%` 管道符的使用，可以利用管道符将一个数据集中符合条件的列转移到另一个数据集。

到这里，我们已经学习了两种按条件生成数据集的方法，一种是 `subset()` 函数，一种是 dplyr 包下的管道符与 `filter()` 函数连用，可以根据自己的需求和对函数的理解来选择适合自己的方法。

6.2 数据集的匹配

6.2.1 匹配以整理变量

在第四章第二节我们运用到 `factor()` 函数来整理变量，使其更加简洁。但这种方法只适用于“籍贯省 ” 这种记录类型比较少的变量，而“出身一 ” 这一变量的记录类型会达到数百个，利用 `factor()` 一一输入对应无疑加大了工作量。面对这种记录类型多的变量，我们在整理时需要用到匹配函数。在 `dplyr` 包中，匹配函数是 `merge()`，格式为：

```
merge(x, y, ...)
```

`x` 为主数据集，`y` 为副数据集，...为参数。常用的参数有：

`by = ...` 表示选择主数据集和副数据集连接的列，当遇到主数据集和副数据集标题名称不同时，可以将 by 参数拆分为 by.x 和 by.y 参数，令 by.x 和 by.y 分别等于名称不同但内容有交集的公共列。

`all = ...` 表示全连接，可选 TRUE 或 FALSE。当只需要保存主数据集的所有列而不需要副数据集的列时，令 all.x = TRUE 并且 all.y = False，即为左连接；当只需要保存副数据集的所有列而不需要主数据集的列

时，令 all.x = False 并且 all.y = TRUE，即为右连接；当只需要主数据集和副数据集的交集，即令 all = False，即为求交集，这也是 R 的默认选项。

`sort` 表示 `by` 指定的列（即公共列）是否要排序。

`suffixes` 表示指定除 by 外相同列名的后缀。

`incomparables` 表示指定 by 中哪些单元不进行合并。

现在，我们按照 `merge()` 函数的格式尝试整理“出身一”变量。导入匹配所需要的副数据集，这里我们用到的是整理“出身一”、“籍贯省”的副数据集。该数据集在第一章“缙绅录的下载”中讲到，请使用者在香港科技大学数据空间、Harvard Dataverse，或者中国人民大学“清史数据共享平台”下载。请注意，`read_dta()` 函数括号中的路径是编写者电脑文件的路径，请使用者根据自己电脑文件的路径进行修改，示例如下：

```
library(dplyr)
JSL1900_1912_clean <- tibble::as_tibble(JSL1900_1912_clean)  # 转换数据集的类型
head(JSL1900_1912_clean, 10)  # 查看该数据集的前十行

library(haven)
Chushen_Recodes_1_ <- read_dta("/Users/jc/Documents/2-数据库-缙绅录数据库/原始文件/缙绅录公开
 ↪  版（新）/CGED-Q JSL Public Release Chushen Recodes.dta")
library(haven)
jiguansheng_Recodes <- read_dta("/Users/jc/Documents/2-数据库-缙绅录数据库/原始文件/缙绅录公
 ↪  开版（新）/CGED-Q JSL Public Release Province of Origin 籍贯省 Recodes.dta")  # 更改为你
 ↪  自己的储存路径

JSL1900_1912_clean_1 <- JSL1900_1912[-kongbaiming,
    ]
JSL1900_1912_clean_2 <- merge(JSL1900_1912_clean_1,
    Chushen_Recodes_1_, by = " 出身一 ", all.x = TRUE)
```

以上代码中，`JSL1900_1912_clean_1` 为剔除掉空白名的新数据集，同时保留 `JSL1900_1912_clean` 这个数据集，因为我们在后面还会用到，所以在新数据 `JSL1900_1912_clean_1` 这个数据集上匹配。代码运行之后，我们就创建了一个自动整理好“出身一”变量的数据集 `JSL1900_1912_clean _2`。

连接环节中，主数据集是 `JSL1900_1912_clean_1`，副数据集是 `Chushen_Recodes_1`，公共列为“出身一”，连接方式为左连接。简单来说，即主数据集和副数据集根据它们相同的列“出身一”所匹配而生成一个新数据集。

那么，如何检验这个数据集是否匹配成功？检验代码需要用到我们之前学过的一些知识：

```
# 在新的数据集中生成 qiren 变量，
# 这个代码在'创建新变量 (1)'中出现过，其含义是利用 ifelse 函数判别旗人
JSL1900_1912_clean_2$qiren <- ifelse((JSL1900_1912_clean$旗分 !=
    "" | JSL1900_1912_clean$身份二 != "" | JSL1900_1912_clean$姓 ==
    ""), 1, 0)

table(JSL1900_1912_clean_2$出身一, JSL1900_1912_clean_2$qiren)
# 原始的'出身一'变量，比较杂乱
table(JSL1900_1912_clean_2$chushen_category, JSL1900_1912_clean_2$qiren)
# 匹配之后生成的新的变量，较为统一

library(table1)
JSL1900_1912_clean_2$qiren[JSL1900_1912_clean_2$qiren ==
    1] <- " 旗 "
JSL1900_1912_clean_2$qiren[JSL1900_1912_clean_2$qiren ==
    0] <- " 民 "
```

```
table1(~chushen_category | qiren, data = JSL1900_1912_clean_2,
    overall = "total")
```

匹配完“出身一”之后，我们可以继续匹配“籍贯省”。需要注意的是，merge() 函数一次只能匹配一个副数据集，所以我们需要在“出身一”匹配成功的基础上创建一个新的数据集来匹配“籍贯省”。

```
JSL1900_1912_clean_3 <- merge(JSL1900_1912_clean_2,
    jiguansheng_Recodes, by = " 籍贯省 ", all = TRUE)
table(JSL1900_1912_clean_3$籍贯省, JSL1900_1912_clean_3$qiren)
table(JSL1900_1912_clean_3$籍贯省 _clean, JSL1900_1912_clean_3$qiren)
```

检验方式与之前相同：

```
JSL1900_1912_clean_3 <- filter(JSL1900_1912_clean_3,
    !is.na(阳历年份) & !is.na(chushen_category) &
        !is.na(籍贯省 _clean))
# 利用 filter() 函数清理掉含有 NA 的行，
```

```r
JSL1900_1912_clean_3$qiren[JSL1900_1912_clean_3$qiren ==
    1] <- " 旗 "
JSL1900_1912_clean_3$qiren[JSL1900_1912_clean_3$qiren ==
    0] <- " 民 "
```

```r
table1(~籍贯省 _clean | qiren, data = JSL1900_1912_clean_3,
    overall = "total")
```

```r
# 下面利用整理好的变量来绘图,
# 利用之前学习过的 ggplot() 函数来制作一个简单的直方图
library(ggplot2)
P_BarChart_Chushen <- ggplot(JSL1900_1912_clean_3,
    aes(阳历年份, fill = chushen_category, )) +
    geom_bar(position = "fill", just = 0.5, width = 0.5) +
    labs(x = " 年份 ", y = " 记载数 ", title = "1900-1912 年文官出身结构 ") +
    guides(fill = guide_legend(title = " 出身 ")) +
    theme(text = element_text(family = "STKaiti", size = 12),
        plot.title = element_text(size = 15, hjust = 0.5,
            color = "orange"), axis.text.x = element_text(size = 9,
            angle = 30, color = "black", margin = margin(t = 0.5,
                unit = "cm")), axis.text.y = element_text(size = 9),
        axis.title.x = element_text(size = 10, hjust = 0.5,
            vjust = 0), axis.title.y = element_text(size = 10),
        legend.position = "bottom", legend.title = element_text(size = 12,
            hjust = 0.5, color = "orange"), legend.background = element_rect(color =
             ↪  "black",
            fill = "grey90", linewidth = 1), legend.text = element_text(size = 10,
            color = "blue"), legend.key = element_rect(color = "purple"),
        panel.background = element_rect(color = "grey50"),
        panel.grid = element_line(color = "grey50",
            linewidth = 0.1))
```

```
P_BarChart_Chushen  # 显示图片
```

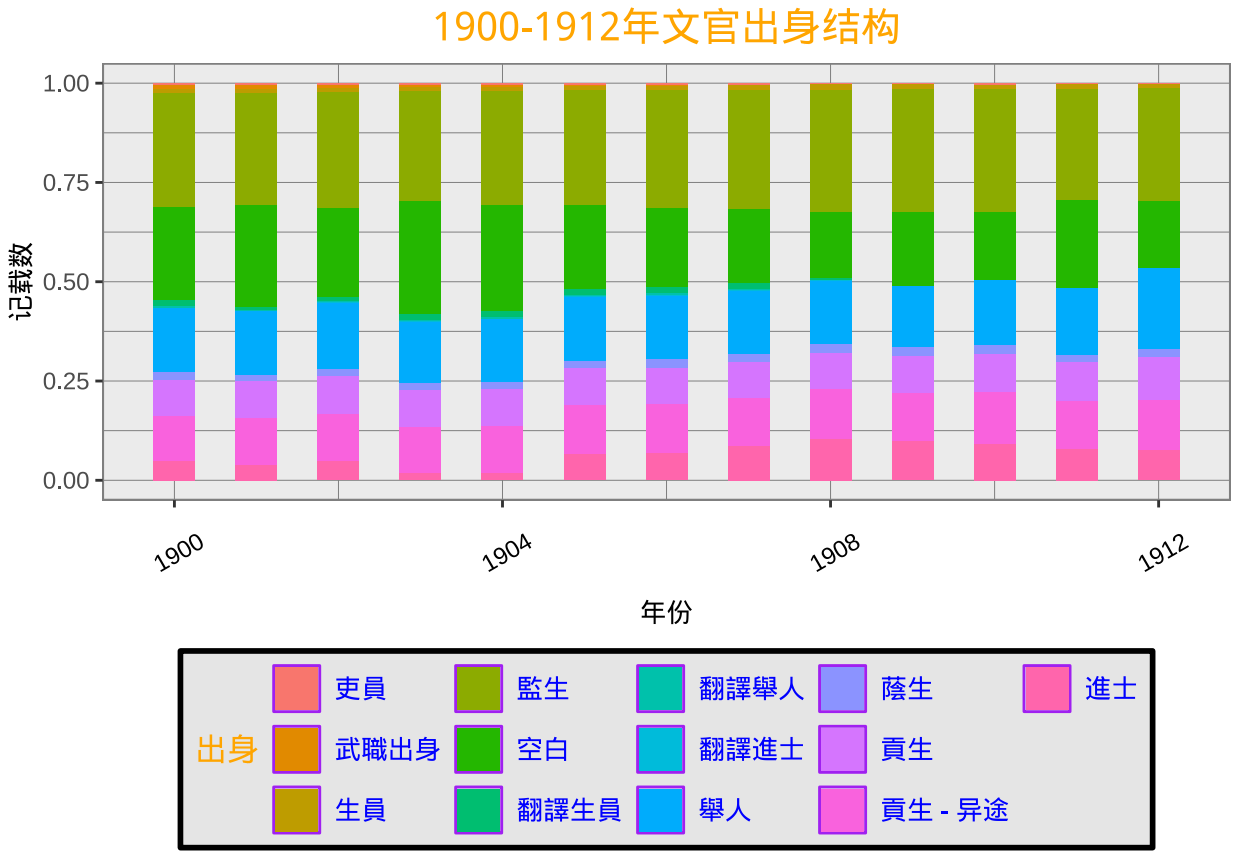

图 8: 1900-1912 文官的出身结构

上图较为清晰地展示了 1900-1912 年文官出身结构的变化，可以看出，清末十年文官的出身结构比较稳定，文官主要来源为正途贡生、异途贡生、监生、进士、举人。

`merge()` 函数可用来整理数据库中记录类型比较多的变量，并且利用制表和制图函数还可以检验匹配效果。使用者还可以尝试整理“任职地 ”来制图制表，发现缙绅录中的诸多信息。

6.2.2 如何创建一个副数据集

以上我们匹配数据集的代码来自李康团队制作的代码，比较便利。但如果使用者需要利用到“出身一 ”、“籍贯省 ”之外的其他变量进行研究，该如何自行制作一个副数据集来整理变量呢？具体步骤如下：

1. 用 `table()` 函数将某个变量的所有数据显示出来
2. 复制 `table()` 的表格到 Excel 中
3. 利用 excel 进行分列，提出主要的数据信息
4. 对数据信息进行编辑，生成新的一列，整理的信息与原数据信息一一对应

5. 保存整理好的文件，导入到 RStudio 中

6. 利用 `merge()` 函数进行匹配

下面以“铨选方式”的整理为例：

```
table(JSL1900_1912_clean_3$铨选方式,
      JSL1900_1912_clean_3$qiren)
# 将`table()`的表格复制到 Excel 中进行
# 对应地编辑，并保存为数据集

install.packages("readxl")
library(readxl)
quanxuanfangshi_sort <- read_excel("/Users/jc/Documents/5-教学-R 语言在缙绅录数据库中的运用
 ↪  v1.1.6/R 语言代码和数据集/quanxuanfangshi_sort.xlsx") # 更改为你自己的储存路径
JSL1900_1912_clean_4 <- merge(JSL1900_1912_clean_3,
                                          quanxuanfangshi_sort,
                                          by = " 铨选方式 ",
                                          all= TRUE)
table(JSL1900_1912_clean_4$铨选方式 _sort,
      JSL1900_1912_clean_4$qiren)
JSL1900_1912_clean_4$qiren[JSL1900_1912_clean_4$qiren == 1] <- " 旗 "
JSL1900_1912_clean_4$qiren[JSL1900_1912_clean_4$qiren == 0] <- " 民 "
table1(~铨选方式 _sort | qiren,
       data = subset(JSL1900_1912_clean_4, !is.na(qiren)), #
       overall = "total")
```

由于 JSL1900_1912_clean_4 数据集的 qiren 变量存在缺失值，故需要利用 subset() 函数在不改变原数据的情况下剔除掉 qiren 变量中的缺失值进行制表，不然 RStudio 会报错。当遇到 table1() 制表后 RStudio 返回列变量存在缺失值的错误指令时，第一种方法是利用 data$var[is.na(var)] <- "text" 代码，即更改数据中缺失值为指定内容，但是这种方法需要更改原始数据，故不推荐。第二种方法是在制表前，利用 filter() 函数筛除缺失值，具体请参照 6.2.1 中“检验籍贯省变量是否匹配成功”代码块的第一条代码。第三种方法即是利用 subset() 函数在不改变原数据的情况下忽略掉 qiren 变量中的缺失值进行制表，这种方法简单快捷且保留了原始数据，故推荐用这种方法。

```
table1(~铨选方式 _sort | qiren, data = subset(JSL1900_1912_clean_4,
    !is.na(qiren)), overall = "total")
```

观察上图可以发现，经过整理后的“铨选方式 ”变量变得更加简洁，更有利于分析。使用者可根据研究需要自行创建副数据集来整理变量。

6.3 连接两个数据集

在缙绅录的分析中，我们进行两个数据集的匹配，以追踪官员的仕途。比如，我们确定了 1900 年秋天在任的官员，但同时又想知道这些官员在 1901 年的秋天是否还在任，该怎么处理呢？我们需要用到连接代码 `inner_join()`，格式如下：

```
inner_join(x, y, by = , na_matches = ,)
```

x 为主数据集，y 为副数据集，by 选定两个数据集中相同的列，与 merge 函数用法相似，如果选择多个列进行连接，则采用 by = c("","","",...) 的形式，na_matches 表示是否匹配含 NA 的行，通常选择 never。下面我们尝试创建连接只有 1900 年秋记录和只有 1901 年秋记录的数据集：

```
# 准备数据
JSL1900_1912_clean %>%
    filter((年份季节 == 1900.5)) %>%
    select(阳历年份, 地区, 机构一, 官职一,
        姓, 名, 字号, 籍贯省, 籍贯县, 旗分,
        出身一, 年份季节) -> JSL1900fall
JSL1900_1912_clean %>%
    filter((年份季节 == 1901.5)) %>%
    select(阳历年份, 地区, 机构一, 官职一,
        姓, 名, 字号, 籍贯省, 籍贯县, 旗分,
        出身一, 年份季节) -> JSL1901fall

# 尝试用`inner_join()`函数将两个数据集连接起来，看看有多少官员 1900 秋-1901 秋均在任
JSL1900_1901_inner_join <- inner_join(JSL1900fall,
    JSL1901fall, by = c(" 名 ", " 姓 ", " 字号 ", " 籍贯省 ",
        " 籍贯县 ", " 出身一 ", " 旗分 "), keep = FALSE,
    na_matches = "never")
```

by=c（...）表示公共列，这里我们用了集合的方式选出了 7 个公共列，这意味着两个数据集的行都需要满足 7 个条件才可进行连接；

keep 表示是否保留合并之后副数据集的连接列，选择 TRUE 可以检验连接是否正常，选择 FALSE 可以简化数据集，仅凭个人意愿选择；

`na_matches = never` 表示不匹配含有 NA 的行。新数据集 `JSL1900_1901_inner_join` 即为两个数据集连接后产生的新数据集，检验方式非常简单，连接后的数据集数量必须小于等于原两个数据集的和。使用者可根据研究需要连接匹配数据集。

第七章 数据集的内外连接

7.1 数据集内部记录的连接

在古代官员研究领域，官员仕途是一个热点话题。目前比较有专门针对清代官员仕途、履历的档案汇编《清代官员履历档案全编》，这套书 1997 年由华东师范大学出版社出版。但这套书具有工具书的性质，需要利用人名检索，操作起来颇为复杂。缙绅录数据库其实也有研究官员仕途的潜力，通过对官员个人信息进行识别，锁定官员并赋予其独一无二的个人编号，通过检索个人编号追踪官员的任职记录，锁定官员任职记录并在《清代官员履历档案全编》中查找相关官员的履历，可以完整而清晰地展现官员的仕途。以上通过创造官员个人编号追踪其任职记录的方法称之为数据集的内连接。

那么，如何在缙绅录数据库中创建内连接呢？步骤如下：

```
library(tidyverse)
JSL1906 <- subset(JSL1900_1912_clean, 阳历年份 == "1906")
JSL1906$RecordNumber <- 1:nrow(JSL1906)
JSL1906 <- arrange(JSL1906,xingming, 身份二, 旗分, 出身一, 年份季节,RecordNumber)
JSL1906 <- JSL1906 %>%
  group_by(xingming, 旗分) %>%
  mutate(PersonID = ifelse(row_number() == 1, 1, 0))
# 这是一个复合函数，利用管道符进行连接。此复合函数分为三步
# 首先，将数据集按照 "xingming" 和 " 旗分 " 两个变量的顺序进行分组，
# 分组是确定不同官员编号的关键；
# 然后，利用`mutate()`函数添加新的变量；
# 最后，`mutate()`里用到了嵌套函数`ifelse()`，
# 如果某一行它是该分组的第一行，那么 R 给它赋值 1，否则为 0

JSL1906$PersonID_cumsum <- cumsum(JSL1906$PersonID)
```

逐步分析以上代码。第一步，创建分析所用数据集。读取 tidyverse 包，利用 subset() 函数创建一个数据集，只提取 1906 年的数据。第二步，创建一个升序变量。“RecordNumber” 是一个新升序变量，1:nrow 函数的意思是按照行的顺序生成一个从 1 到 n 的序号变量。第三步，按数据库中的特定变量进行排序。排序函数为 arrange()，它可以将行按指定列的顺序来排序。在这一条代码中，xingming...RecordNumber 是排序的条件，这里我们选择的首要条件是 xingming，然后 R 会根据姓名进行排序，如果有多条件，那么 R 会根据条件的先后进行排序，即先考虑 xingming，再考虑身份二。把身份二和旗分放在第二和第三是因为有很多旗人是同名的，所以我们利用身份二和旗分变量进行二次甄别。同时不考虑 by_group，令其等于默认，即 FALSE。第四步，按数

据库中的特定变量进行分组。分组函数为 group_by()，可以将行按指定条件进行分组。按照 group_by() 函数的格式，编写如下代码创建一个 PersonID（个人编号）的新变量。第五步，对升序变量进行累加。cumsum() 函数，可以对指定行、分组求累加值。分组和赋值完成后，PersonID 里只有逻辑值，这里需要用到 cunsum() 累加函数给每一个 PersonID 一个独一无二的编号。

以上代码用到一些核心的函数，分别是 arrange()、group_by()、mutate()、cumsum() 函数，它们的基本语法为：

arrange(.data, ...,by_group) 排序函数.data 为数据集，...为指定的条件（选择列为条件），by_group 表示是否首先按照分组的变量进行排序，可选 TRUE 或者 FALSE

group_by(.data, ...) 分组函数.data 为数据集，...为指定的条件

mutate(.data, ...) 变量生成、转换以及删除函数.data 为数据集，...要创建或修改的列。mutate() 函数是 dplyr 包下的核心函数之一，使用者可以通过在控制台键入?mutate 获取该函数的所有参数和用法

cumsum(data$var) 累加函数 data 为数据集，var 为变量

现在，每个官员都拥有了一个独一无二的编号 PersonID_cumsum。我们可以利用官员编号来分析该官员的官职迁转、任职年限甚至是仕途发展。

7.1.2 利用个人编号分析官员任职年限

下面，我们利用官员个人编号分析官员的任职年限，并检验前述代码的效果。代码如下：

```
# 基于管道符的组合代码函数
JSL1906 <- JSL1906 %>%
    group_by(PersonID_cumsum) %>%
    mutate(YearsServed = abs((年份季节 - 年份季节 [1]) +
        0.25))

# 制表检验
JSL1906_winter <- subset(JSL1906, 季节号 == 4)
library(table1)
JSL1906_winter$YearsServed[JSL1906_winter$YearsServed ==
    0.25] <- "0.25 年 "
JSL1906_winter$YearsServed[JSL1906_winter$YearsServed ==
```

```
    0.5] <- "0.5 年 "
JSL1906_winter$YearsServed[JSL1906_winter$YearsServed ==
    0.75] <- "0.75 年 "
JSL1906_winter$YearsServed[JSL1906_winter$YearsServed ==
    1] <- "1 年 "
```

上述组合代码中，第一步是将数据集按照 PersonID_cumsum 变量的顺序进行分组。第二步是利用 `mutate()` 函数添加新的变量 `YearsServed`，以当前行的年份季节减去每个分组第一行的“年份季节”，再加上 0.25，为什么要加 0.25？因为我们在创造“年份季节”这个变量（见 3.1.4）的时候为了避免值进 1 引起混淆减去了一个 0.25，这个时候要加回来。

```
table1(~jiguansheng_sort | YearsServed, data = JSL1906_winter,
    overall = "total")
```

从上图我们可以看出，在籍贯省任职的官员任期稳定性最高，同时，在浙江、顺天、江苏任职的官员任期稳定性比较高，在所有任职达到一年的官员中（因为我们只取了 1 年的数据），浙江占比最高，为 8.3%。顺天占比第二，为 6.1%。当然，我们这里只用了一年的数据，在很大层面上会引起歧义。所以，我们建议大家在更大的年限跨度上去做尝试。

7.2.3 如何统计人数而非记载数

我们还可以利用 `PersonID` 将统计模式从记载数转变为人数。用人数来统计有什么好处？假如一个数据集有 50w 万条记录，但可能同一个人就有 10 条记录，所以整个数据集可能只有 5 万人。当我们用记载数来统计整个数据库的时候，有很多项目都会出现重复记录的情况，因为一个官员出现了多次。当我们用人数来做统计对象时，满汉比例、出身构成、官员社会和地理来源都会变得非常精确，不会出现重复计算的情况。代码示例如下：

```
# 此处用到`filter()`函数，筛选出 PersonID ==
# 1 的行。
# 利用'PersonID'来删除同一个人的多条记录，
# 只保留第一条记录，这样就能保证留下来的所有记录是每个人的唯一一条记录
JSL1906_ 人数 <- JSL1906 %>%
    filter(PersonID == 1)

# 制图检验
library(ggplot2)
```

```
P_BarChart_Jiguansheng <- ggplot(JSL1906_ 人数, aes(jiguansheng_sort)) +
    geom_bar(just = 0.5, width = 0.5) + labs(x = " 籍贯 ",
    y = " 人数 ", title = "1906 年文官的地理来源（人数）") +
    theme(text = element_text(family = "STKaiti", size = 12),
        plot.title = element_text(size = 15, hjust = 0.5,
            color = "orange"), axis.text.x = element_text(size = 9,
            angle = 30, color = "black", margin = margin(t = 0.5,
                unit = "cm")), axis.text.y = element_text(size = 9),
        axis.title.x = element_text(size = 10, hjust = 0.5,
            vjust = 0), axis.title.y = element_text(size = 10),
        panel.background = element_rect(color = "grey50"),
        panel.grid = element_line(color = "grey50",
            linewidth = 0.1)) + geom_text(aes(label = ..count..,
    family = "serif"), stat = "count", size = 3.5,
    vjust = -0.8, hjust = 0.5, color = "red2")
```

```
P_BarChart_Jiguansheng  # 显示图形
```

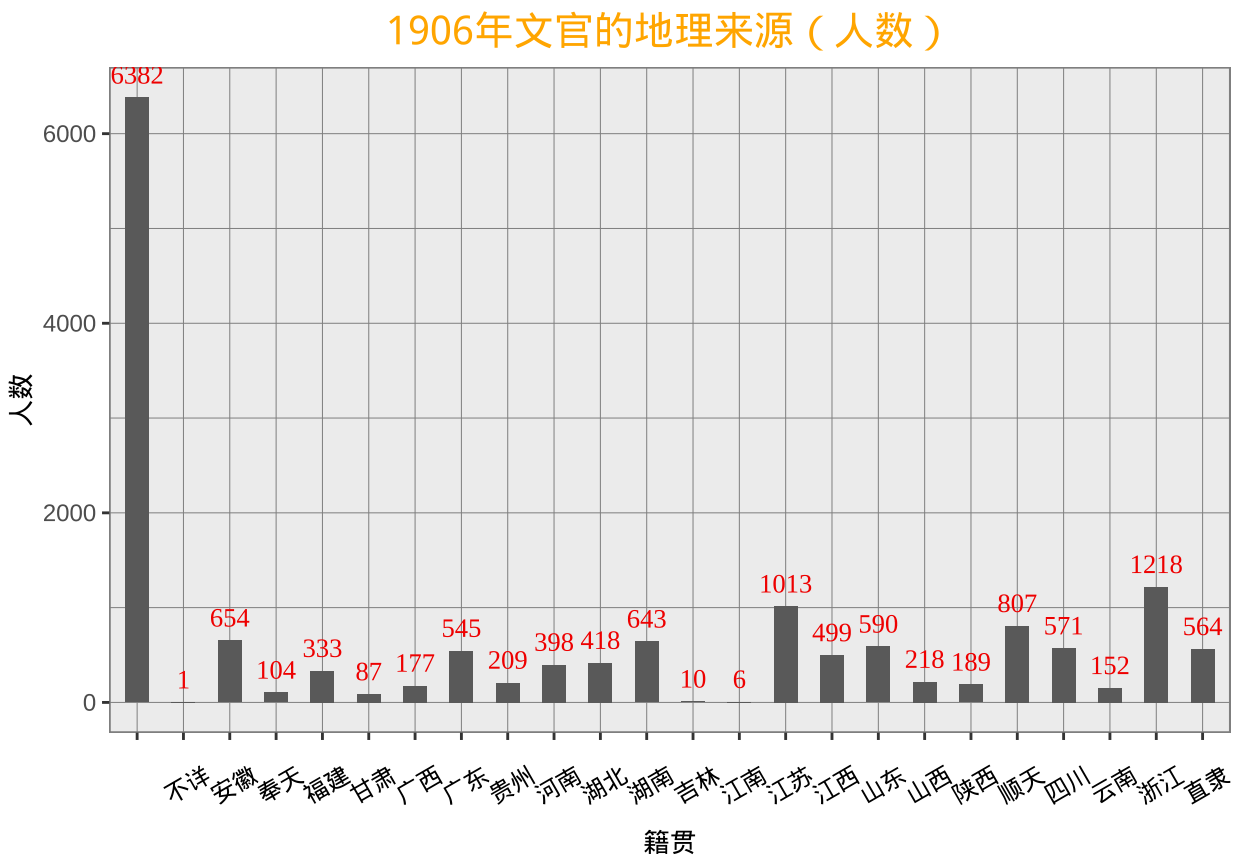

图 9: 1906 年文官的地理来源统计图

原始的数据集 JSL1906 大约有 5 万条记录，而经过删除个人重复记录后的数据集 JSL1906_ 人数则约有 1.5 万条记录，展示了 1.5 万名不同官员的个人信息，如果不删除重复记录而直接对记载数做统计的话会导致结果精确性大打折扣。

从上图中可以看出，1906 年文官大多数都是 qiren（因旗人籍贯省记载为 " 空白 "）和籍贯为京师的人（在本籍地任职的官员，其籍贯省也为空白），这表现出清代选官 " 首崇满洲 " 的思想。除此之外，文官多来自浙江、江苏、广东等沿海城市，这也表明在文化发达的沿海省份，产生行政管理者的概率更高。利用人数可以从整个大数据集出发考察具体真实的满汉比例，出身构成和官员社会地理来源。

在部分情况下保留官员的第一条任职记录和最后一条记录几乎没有太大的差别，但是官员任职过程中可能更换字号和出身信息，也有可能随着官职的迁转其铨选方式也会发生改变，这些情况下保留官员最后一条任职记录再进行分析的准确性会更高一点。同时，由于创建官员个人编号时使用了组内排序方法，故研究官员的任期也需要保留最后一条任职记录。保留官员最后一条任职记录的代码如下：

```
JSL1906_last_record <- JSL1906 %>%
    group_by(xingming) %>%
    filter(row_number() == n())
# 只保留官员的最后一次任职记录
```

```
# 可以求得官员职业生涯总的任职年限
```

```
table1(~jiguansheng_sort | YearsServed, data = JSL1906_last_record,
    overall = "total")
# 通过制表来检验保留官员最后一次任职记录的效果
```

是否保留官员的最后一条任职记录需要示情况而定。如使用者不确定，可根据官员的第一条任职记录和最后一条任职记录生成不同数据集，进而进行分析和比较研究。考虑到缙绅录的编纂体例随着时间的推移而愈加完善，这意味着越后期的版本信息更加完整，因此我们推荐使用者在利用人数进行分析研究时保留官员的最后一条任职记录。

7.3 两个数据集的连接

在实际的数据集分析中，我们时常会遇到两个数据集相匹配的问题。我们前面介绍的数据集匹配 `merge()` 函数采用的是精准匹配方法，且我们所用到的数据集是从整个大的缙绅录数据库中截取出来的，两者的结构、变量设置都是一样的，所以匹配起来比较容易。但在实际的匹配时，我们时常会遇到两个不同类型、不同结构的数据库匹配，这种情况下，公共列的选取就比较困难。比如一个数据库是缙绅录，另一个数据库是人口数据库，人口数据库的内容是人的年龄、亲属数量、税务等等，这与缙绅录数据库的设置完全不一致。现在，我们需要将两个数据库连接起来，考察官员的亲属关系和个人生活，就只能利用到 `xingming` 这个变量去进行匹配，将两个数据库连接起来，再对数据进行慢慢甄别、检验。

现在我们仍然以缙绅录数据库为例，讲解数据库外连接的步骤。首先需要截取缙绅录数据库中 1906 年四川知县的数据、1910 年四川知县的数据，以考察四年间四川知县的变动情况。请注意，如果我们要考察 1906 到 1910 年间四川地区知县是否一直在任的情况，我们需要统计的是人数而非记载数。因此，在准备数据阶段，我们不仅要截取指定年份的数据，还要创造 `PersonID` 变量，再保留官员的最后一条任职记录。如果不删除重复任职记录，那么一个数据会被连接多次，最后导致生成的数据库样本量远大于被连接的两个数据集。以下代码创建 `PersonID` 和保留最后一条任职记录在前节 (7.2.3) 出现过，只是更换了数据集名称，所以不再一一解释。示例代码如下：

```
JSL1900_1912_clean %>%
    filter((阳历年份 == "1906" & 地区 == " 四川省 " &
        zhixian == " 知县 ")) %>%
    select(阳历年份, 地区, 机构一, 机构二,
        官职一, xingming, 字号, 籍贯省, 籍贯县,
        身份二, 旗分, 出身一, 年份季节) ->
    JSL1906_sichuan_zhixian
```

```
JSL1906_sichuan_zhixian$RecordNumber <- 1:nrow(JSL1906_sichuan_zhixian)
JSL1906_sichuan_zhixian <- arrange(JSL1906_sichuan_zhixian,
    xingming, 身份二, 旗分, 出身一, 年份季节,
    RecordNumber)
JSL1906_sichuan_zhixian <- JSL1906_sichuan_zhixian %>%
    group_by(xingming, 旗分) %>%
    mutate(PersonID = ifelse(row_number() == 1, 1,
        0))
JSL1906_sichuan_zhixian$PersonID_cumsum <- cumsum(JSL1906_sichuan_zhixian$PersonID)
JSL1906_sichuan_zhixian <- JSL1906_sichuan_zhixian %>%
    group_by(PersonID_cumsum) %>%
    mutate(YearsServed = abs((年份季节 - 年份季节 [1]) +
        0.25))
JSL1906_sichuan_zhixian <- JSL1906_sichuan_zhixian %>%
    group_by(xingming) %>%
    filter(row_number() == n())

JSL1900_1912_clean %>%
    filter((阳历年份 == "1910" & 地区 == " 四川省 " &
        zhixian == " 知县 ")) %>%
    select(阳历年份, 地区, 机构一, 机构二,
        官职一, xingming, 字号, 籍贯省, 籍贯县,
        身份二, 旗分, 出身一, 年份季节) ->
    JSL1910_sichuan_zhixian
JSL1910_sichuan_zhixian$RecordNumber <- 1:nrow(JSL1910_sichuan_zhixian)
JSL1910_sichuan_zhixian <- arrange(JSL1910_sichuan_zhixian,
    xingming, 身份二, 旗分, 出身一, 年份季节,
    RecordNumber)
JSL1910_sichuan_zhixian <- JSL1910_sichuan_zhixian %>%
    group_by(xingming, 旗分) %>%
    mutate(PersonID = ifelse(row_number() == 1, 1,
        0))
JSL1910_sichuan_zhixian$PersonID_cumsum <- cumsum(JSL1910_sichuan_zhixian$PersonID)
JSL1910_sichuan_zhixian <- JSL1910_sichuan_zhixian %>%
    group_by(PersonID_cumsum) %>%
    mutate(YearsServed = abs((年份季节 - 年份季节 [1]) +
```

```
        0.25))
JSL1910_sichuan_zhixian <- JSL1910_sichuan_zhixian %>%
    group_by(xingming) %>%
    filter(row_number() == n())
```

fuzzy_join() 函数，可对两个数据集进行匹配，格式为：

```
fuzzy_join( x, y, by = NULL, match_fun = NULL, multi_by = NULL, multi_match_fun =
NULL, index_match_fun = NULL, mode = "inner", ...)
```

x 为主数据集，y 为副数据集，by = 为根据某个变量进行模糊匹配。

match_fun = 给定两列的向量化函数，返回 true 或 FALSE 以判断它们是否匹配。可以是中指定的每对列的函数列表（如果是命名列表，则使用 x 中的名称）。如果只给定一个函数，则在所有列对上使用它。

multi_by = 要联接的列，其中所有列将用于一起测试匹配项。

multi_match_fun = 用于测试匹配项的参数，同时对每个数据帧中的所有列执行。

index_match_fun = 用于匹配表的参数。

mode 匹配的模式，可以选左连接，右连接，外连接，内连接，全连接，需输入对应名称。

根据 fuzzy_join() 函数编写代码：

```
# 安装和读取包
install.packages("fuzzyjoin")
library(fuzzyjoin)

# 利用`fuzzy_join()`函数匹配两个数据集
JSL1906_1910_zhixian <- stringdist_inner_join(JSL1906_sichuan_zhixian,
    JSL1910_sichuan_zhixian, by = "xingming", method = "dl",
    max_dist = 1, distance_col = NULL)
```

stringdist_inner_join 参数是 fuzzy_join 包中一个较为直接、简单的函数，其用法和 merge() 函数相似：by = 参数根据 xingming 列进行模糊匹配；method 表示计算距离的算法，这里选择的是 dl 算法，可选择的算法包括详见 help；max_dist 表示链接的最大距离；distance_col 表示是否创建包含两个数据集不同点的一个新变量，默认是 NULL。

这样我们就创建了一个新的数据集 JSL1906_1910_zhixian，里面是四年间长期担任知县的官员，可以基于这个数据集探究为何这些官员能够长期担任知县。

R 语言在寻找这些满足条件的行的时候，采用内置的距离算法，计算出与公共列距离最近的样本量。距离算法只考虑计算出的距离，而不考虑其它因素，其计算过程中只以给定的信息为基准，在最大程度上找到与之相关的样本。可以看出，我们之前学习的 `merge()` 和 `fuzzy_join()` 函数逻辑是相同的。

但是，`fuzzy_join()` 这一类连接方法还存在着一些局限性，比如不能对不同字体（繁体字、异体字）展开有效的甄别，从而影响数据库链接的效率和准确度。我们建议使用者在进行连接之前，先利用字符替换函数将数据库中的繁体字、异体字替换为简体字，然后对数据库的字符变量进行清理，尽量让两个数据库的变量格式达成一致，最后再进行连接。

清代典籍中带有大量的繁体字、异体字，在数据库分析中时常会遇到两个官员名字看起来完全一样，但分析软件依然无法识别，究其原因是两个官员名字字体结构上存在肉眼难以发现的笔画差别，导致匹配失败。针对这一问题，有部分学者提出了可行的解决方案，其中康文林、陈必佳两位学者的方法值得借鉴，他们以缙绅录数据库为例，列表展示了缙绅录数据库中最常见的姓和名，并利用数据库外连接的经验，展示了姓和名中异体字配对的累计百分比，进而减少了异体字甄别的时间，提高了分析效率。如果有使用者希望对历史数据库的匹配和链接进行更为深入地考察，可以参考康、陈两位学者的文章。[12]

[12]康文林，陈必佳:《中国历史官员量化数据库——清代（CGED-Q）的人名匹配与官员记录连接》，付海晏主编:《大数据与中国历史研究》（第 5 辑），北京：社会科学文献出版社，2024 年，第 35-72 页。

第八章 制作 GIS 图像

8.1 导入 GIS 底图

GIS 图像较表格而言更加直观，且图像能展现出表格难以显现的地理分布特征，在历史学分析中运用较为广泛。缙绅录数据库中的“籍贯省”和“地区”两个变量包含有地理信息，且由于缙绅录数据库体量庞大，以 GIS 可视化图像能够更加清晰地展现其分布特征，有助于理解清代官员任职地和籍贯分布的时空特征。

目前主流的 GIS 平台是 ArcGIS 和 QGIS，前者为付费软件，专业性强，平台响应快，其引领的数据格式已成为 GIS 行业标准，但操作复杂、经济成本和学习成本高昂；后者为开源软件，轻便高效、兼容性强，但专业性较弱，更新依赖社区维护。

在 R 语言中运用 `ggplot` 包亦能够制作 GIS 图像，制图的核心逻辑为“导入底图——匹配坐标——构建图层”。

哈佛大学和复旦大学历史地理研究中心开发的 CHGIS（The China Historical Geographic Information System），从 2002 年开始陆续公开了六个开源版本的中国古代聚居区和行政区划数字地理信息数据库。这套数据库分为时间连续和时间截面两类，截面数据中包含有 1820 年、1911 年的数据，其时间范围与缙绅录数据库重合。CHGIS 中 1820 年的地理信息数据提供了省、府的地理边界和省、府、县、镇的地理点数据，还提供了河流、湖泊、海岸线等水文数据，还包含有太湖的区域性地理数据，堪称全面；1911 年的数据提供了省、府、县的地理边界数据和省、府、县、镇的地理点数据，与 1900-1912 年缙绅录数据库提供的的地理信息紧密相关，因此，本章以 CHGIS 提供的 1911 年的地理信息数据作为底图制作 GIS 图像。

CHGIS 的地理信息数据供学术研究者免费使用，使用者可以先行至 CHGIS 官网或复旦大学中国历史地理研究中心下载数据。

请使用者注意，使用地图时必须遵守相关规范，如需要制作公开发表的地图，请参照自然资源部的相关规范文件或者使用自然资源部推出的标准地图服务。

本教程严正声明：本教程考虑到简化代码、串联制图流程、直观展示分析研究结果的重要性，所以使用了 CHGIS 制作的 1911 年中国历史地图，本教程制作的所有地图仅供展示制图代码结果以及个人学习制图代码使用，严禁单独转发、盗用、误用本教程制作的历史地图。本教程是开源的公益教程，因此不具备实时监控能力，对于任何因不完整转发、盗用、误用本教程制作的所有历史地图而产生的问题，本教程概不负责，亦不负任何法律责任。

导入 GIS 底图的步骤如下：

```
# 安装地理信息处理包
install.packages("sf")
install.packages("ggpointdensity")
```

```r
install.packages("ggspatial")
install.packages("ggnewscale")
install.packages("viridis")
install.packages("scales")

# 读取包，请注意，如果已经安装了包，就不需要再次安装，只需要读取即可
library(sf)
library(ggplot2)
library(dplyr)
library(stringr)
library(ggpointdensity)
library(ggspatial)
library(ggnewscale)
library(viridis)
library(scales)

# 导入文件
chgis1911_county_pts <- "/Users/jc/Documents/6-分析-GIS 底
 ↪  图/CHGIS/CHCIS1911county_gbk_shp_pts/v6_1911_cnty_pts_gbk.shp"
chgis1911_prov_pgn <- "/Users/jc/Documents/6-分析-GIS 底
 ↪  图/CHGIS/CHCIS1911prov_gbk_shp_pgn/v6_1911_prov_pgn_gbk.shp"
chgis1911_prov_pts <- "/Users/jc/Documents/6-分析-GIS 底
 ↪  图/CHGIS/CHCIS1911prov_gbk_shp_pts/v6_1911_prov_pts_gbk.shp"
chgis1911_pref_pgn <- "/Users/jc/Documents/6-分析-GIS 底
 ↪  图/CHGIS/CHCIS1911prefecture_gbk_shp_pgn/v6_1911_pref_pgn_gbk.shp"
chgis1911_county_pgn <- "/Users/jc/Documents/6-分析-GIS 底
 ↪  图/CHGIS/CHCIS1911county_gbk_shp_pgn/v6_1911_cnty_pgn_gbk.shp"

# 转换地理数据
chgis1911_county_pts <- st_read(chgis1911_county_pts,
    options = "ENCODING=GBK")
chgis1911_prov_pgn <- st_read(chgis1911_prov_pgn, options = "ENCODING=GBK")
chgis1911_prov_pts <- st_read(chgis1911_prov_pts, options = "ENCODING=GBK")
chgis1911_pref_pgn <- st_read(chgis1911_pref_pgn, options = "ENCODING=GBK")
chgis1911_county_pgn <- st_read(chgis1911_county_pgn,
    options = "ENCODING=GBK")
```

在读取 CHGIS V6 shapefile 文件时，以上代码导入的分别是 1911 年县级点、省级边界、省级点、府级边界、县级边界。请更改以下代码引号中内容为文件在你自己电脑中的存储路径。另请读取.shp 为后缀的文件，同时.shx 以及.dbf 等其他后缀文件需要和.shp 在同一个文件夹之下，以便 R 调取。后缀为 pts 的数据集为地理点数据，后缀为 pgn 的数据集为地理边界数据，前者用于绘制坐标点，常用于和各类数据进行匹配以赋予每条记录坐标点；后者用于绘制边界，常用于生成底图。两类文件作用不同，请明确区分两类文件的用法，否则会导致制图不成功。

导入完成后，可以在区域三 Environment 下拉的 Values 中查看到导入的数据，但这个时候，导入的数据并非是一个数据框，无法参与运算，因此，需要将对应数据转换为数据框。将导入数据转换为数据集，指定 GBK 编码格式，否则可能会出现乱码。如文件本身编码格式为 GBK，则需要指定为 GBK 格式；如本身是 utf-8，则需要指定为 utf-8 格式

数据导入和转换之后，即能在区域三 Environment 下拉的 Data 中查看到对应的数据集，紧接着利用导入的数据和缙绅录数据库构建记录的地理坐标。

8.2 构建地理坐标

缙绅录数据库中“籍贯省”和“地区”两个变量提供了省、府、州、县的名称，但并不涉及地理坐标。查看前述生成的 GIS 数据集可以发现，CHGIS 中 1911 年的地理数据提供省、府、县的地理点和地理边界以及名称。因此，可以利用 merge() 函数以区域名称为公共列匹配两个数据集，从而赋予缙绅录数据库地理坐标信息。这样，便可以得到一个不仅包含官员的个人信息，还包含官员任职地/籍贯地理信息的新数据集，进而以新数据集为基础制图。

从缙绅录数据库中提取出 1911 年秋的数据与 1911 年地理信息数据匹配，由于两者年份相同，所以能够较为准确的反映 1911 年清代文官的地理分布状况。请注意，由于单独一季的数据每个人仅有一条记录，所以此处省去了个人删除重复数据的步骤，如果要使用多季的纪录，可参考 7.2.3 节的步骤删除重复记录，再匹配数据。

```
# 从 1900-1912 公开版缙绅录数据库中提取并简化 1911 年秋的数据
JSL1900_1912_clean %>%
  filter(年份季节 == 1911.5) %>%
  select(阳历年份, 地区, 机构一, 机构二, 机构三, 官职一, 姓, 名, 籍贯省, 籍贯县, 旗分, 出身
   ↪  一,qiren) ->
  JSL1911fall

# 匹配
JSL1911fall_2 <- merge(JSL1911fall,
```

```
                    chgis1911_county_pts,
                    by.x = " 机构二 ", by.y = "NAME_FT")
```

在匹配环节运用的是 CHGIS1911 县级点和缙绅录数据库 1911 秋数据集，公共列采用的是任职地的县级名称，任职地的县级名称在缙绅录数据库中为“机构二 ”变量，在 CHGIS1911 年县级点数据集中为 NAME_FT，两者数据内容和结构一致，具有匹配意义，因此作为匹配工作的公共列。

查看新生成的 JSL1911fall_2 可以发现约有 5000 多条记录匹配成功，这个数据集内包含有两个被匹配数据集的所有列，每条记录不仅有官员个人信息，还有官员任职地的地理信息，可作为制图的数据集。但请注意，缙绅录数据库的“机构二 ”变量与 CHGIS1911 县级点数据的数据结构一致，均为“某某县 ”，所以不需要处理便可以直接匹配。当遇到结构不一致的数据时，比如缙绅录数据库的“籍贯县 ”变量，标识为某某地，没有“县 ”字，这时如果直接进行匹配是不会有匹配成功的纪录，因为 merge() 函数采用的是精确匹配的方法。遇到这种情况，可以为缙绅录数据库的“籍贯县 ”变量增加一个“县 ”字便能解决。

R 可为指定变量增减指定字符，以“省 ”为例，缙绅录数据库 ”籍贯省 ”变量中省份不带“省 ”字，而 CHGIS1911 地理边界/地理点数据的文件中省份也不带“省 ”字，所以两者可以直接匹配，而缙绅录数据库中“地区 ”变量的省份带有“省 ”字，而 CHGIS 中没有，遇到这种情况，可以用以下代码增加或去掉“省 ”字，以确保匹配成功。以下两条代码以“籍贯省 ”为例增加或去掉“省 ”字，请根据实际情况使用请使用者根据实际情况使用：

```
JSL1911fall <- JSL1911fall %>%
    mutate(籍贯省 = ifelse(籍贯省 != "", paste0(籍贯省,
        " 省 "), 籍贯省))
table(JSL1911fall$籍贯省)

# 注意：如果不小心多次运行了以上代码，
# 可能会重复在变量上增加省字，可以用以下代码移除
JSL1911fall <- JSL1911fall %>%
    mutate(籍贯省 = str_remove(籍贯省, " 省 $"))
```

使用者利用增减字符的方法促使数据标准化。但数据标准化的步骤并不仅此一步，数据清理也是数据标准化的一部分，前述 4.2 节和 6.2.1 节分别整理了“籍贯省 ”和“出身一 ”变量，但缙绅录数据库的“机构一 ”、“机构二 ”、“机构三 ”和“籍贯县 ”等地理信息变量并未整理，导致数据的匹配率仅有 37%。如果将上述变量结合数据清理方法进行精细化处理，其匹配率会得到显著提升。但由于本教程主要介绍方法，加入冗长的整理代码会影响制图步骤的延续性，故略去了数据清理环节。使用者可根据自身需求整理上述的地理信息变量，以提高匹配率。

8.3 绘制 GIS 图像

8.3.1 绘制点密度图

绘制点密度图的步骤是：首先绘制底图，再利用点密度函数将点添加到底图图层之上，最后设置标题、图例和调整图形的主题。关键步骤在于前两步。

第一步是绘制底图。例如，现在需要绘制 1911 秋全国文官任职地分布图，首先需要绘制全国省级边界，可以直接利用 CHGIS 提供的 1911 年全国省级边界底图，也就是 8.1 节导入并转换的省级边界底图 `chgis1911_prov_pgn`，考虑到全国省级边界比较大，可以再进一步加入 1911 年全国府级边界底图 `chgis1911_pref_pgn`，来探讨文官在省份内部的分布状况。在选取地理边界底图时，尤其注意要和地理点数据区分开来。在 CHGIS 中，地理边界数据后缀为 `pgn`，常用于绘制边界，生成底图；地理点数据后缀为 `pts`，常用于和各类数据进行匹配以赋予坐标点。若在数据导入环节没有注意到两者的区别进而产生混淆，可以在数据集内部查看 `geometry` 变量判断，例如，`chgis1911_pref_pgn` 数据集中 `geometry` 变量内容的格式为 `MULTIPOLYGON(x, ...)`；而 `chgis1911_county_pts` 数据集的 `geometry` 变量内容的格式为 `POINT(x, y)`，使用者可以结合两者进行区分。

在绘制底图的过程中，需要将府级边界设置为第一层图层，省级边界设置为第二层图层，如果颠倒过来，府级边界底图会完全覆盖省级边界图形，导致省级边界无法完全显现，这是因为府级边界中包含了省级边界，县底图与府底图的关系同理。根据地理信息数据的这个特点，使用者需要先绘制次一级行政区划底图，再绘制高一级行政区划底图。

第二步是绘制点。前述环节给每条记录均构建了坐标，这意味着每条记录均有自己在坐标轴上的点。在 `JSL1911fall_2` 这个数据集中，每条记录点坐标的获取是依靠 `geometry` 变量，查看 `JSL1911fall_2` 数据集中的 `geometry` 变量，它的格式是 `POINT(x, y)`，可以发现括号中前者代表 `x` 轴位置（经度），后者代表 `y` 轴位置（纬度），这时便可以利用 `sf` 包中的 `st_coordinates()` 函数从 `geometry` 变量提取坐标信息。之所以要提取坐标而不是直接套用 `geometry` 变量进行制图，是因为 `geometry` 是一个空间点变量，而点密度制图函数 `geom_pointdensity()` 不支持直接输入空间几何对象，它需要明确的 `x` 和 `y` 数值坐标，需要通过 `st_coordinates()` 提取坐标后，才能传递给图层编辑函数 `aes(x=..., y=...)`。

第三步是设置标题、图例和调整图形的主题。主要用到 `labs()` 设置标题和 XY 轴名称，`theme()` 函数调整图形主题，这些函数在第五章亦提到过，所以不再赘述。

```
# 将数据集转换为 sf 对象以便制图
JSL1911fall_2 <- st_as_sf(JSL1911fall_2)
chgis1911_pref_pgn <- st_as_sf(chgis1911_pref_pgn)
chgis1911_prov_pgn <- st_as_sf(chgis1911_prov_pgn)

# 绘制 "1911 年秋全国文官任职地分布 " 点密度图
```

```r
GIS_PointDensity <- ggplot() +
  geom_sf( data = chgis1911_pref_pgn,          # 先设置府级边界作为第一层底图
           mapping = aes( geometry = geometry ),  # 地理边界数据可以在图层编辑函数`aes()`直接套用`geometry`
           color = "grey60",  # 使用浅灰色
           size = 1,          # 细线条
           alpha = 0.5,       # 半透明
           fill = NA ) +      # 无填充
  geom_sf( data = chgis1911_prov_pgn,          # 再设置省级边界作为第二层底图，防止府级边界覆盖省级边界
           mapping = aes( geometry = geometry ),  # 地理边界数据可以在图层编辑函数`aes()`直接套用`geometry`
           color = "black",    # 深色突出
           size = 1.5,         # 粗线条
           fill = NA) +        # 无填充
  geom_pointdensity(data = JSL1911fall_2,     # 绘制点，利用 8.2 节构建好地理坐标的数据集
                    mapping = aes( x = st_coordinates(geometry)[,1],    # 地理点数据需要利用`st_coordinates()`函数从`geometry`变量提取坐标信息，[,1] 意为提取第一列作为经度
                                   y = st_coordinates(geometry)[,2] ),  # 地理点数据需要利用`st_coordinates()`函数从`geometry`变量提取坐标信息，[,2] 意为提取第二列作为纬度
                    color = "#f47720",
                    size = 0.5, # 点大小
                    alpha = 0.6, # 点透明度
                    adjust = 0.8) +  # 控制带宽，显示更精细的局部细节，该值小于 1 适用于观察高密度区域的微小聚集
  labs(title = "1911 年秋全国文官任职地分布（每个点代表一名文官）",
       x = " 经度 ", y = " 纬度 ") +
  theme(text = element_text(family = "STKaiti",size = 12),  # 如果是 MAC OS 系统，请保留该代码，否则会出现字符显示不全的问题
        panel.background = element_rect(fill = "white"),  # 设置面板背景颜色
        plot.background = element_rect(fill = "white"),      # 设置图片背景颜色
        plot.title = element_text(hjust = 0.5),              # 设置图片标题居中
        panel.grid.major = element_line(color = "#88c4e8", linewidth = 0.25),  # 主要网格线颜色和大小
```

```
    )
```

```
GIS_PointDensity  # 查看图片
```

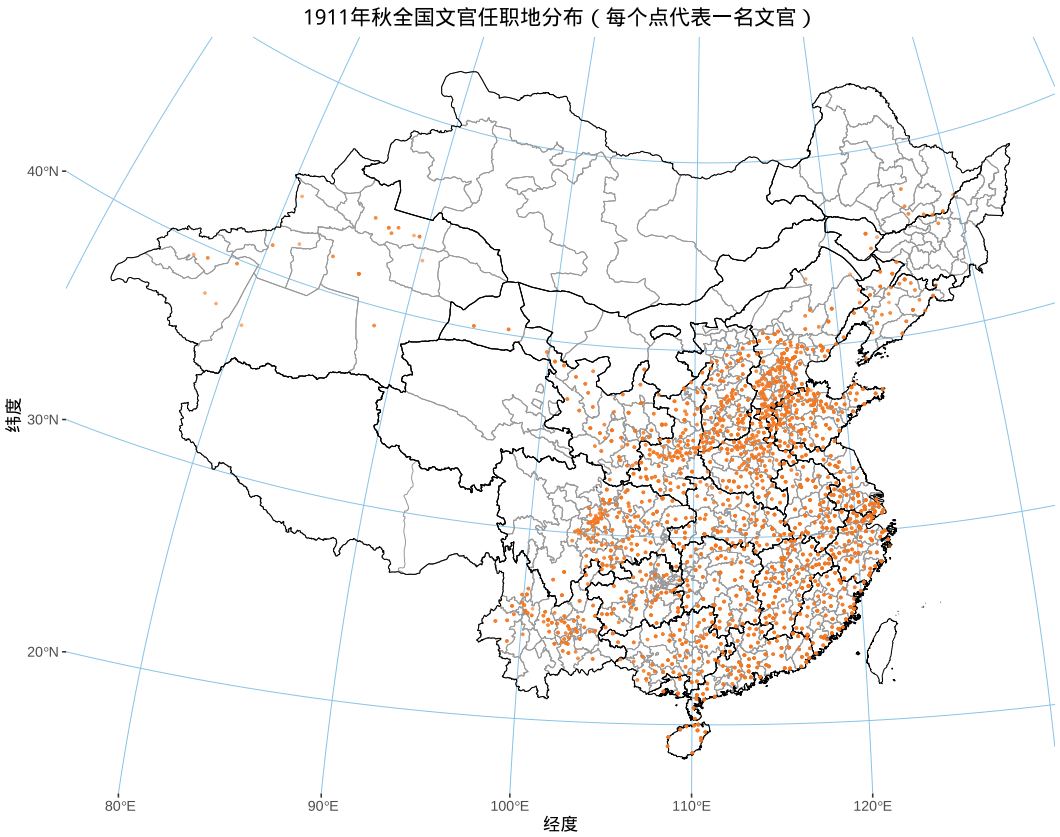

图 **10:** 1911 年秋全国文官任职地分布每个点代表一名文官

观察上图可以发现，1911 年秋文官任职地分布较为密集的区域为华北地区和江南地区，表明清政府在这些区域设置了较多的文官。但这仅是一部分的匹配结果，比如青海、蒙古、西藏等地区的官员没有显示出来，这是由于 `JSL1911fall` 和 `chgis1911_county_pts` 两个数据集地名变量的不一致导致的结果，如果进行完全的精细化匹配，便能够得到更完整的图像以考察清代文官任职地分布状况。

由于缙绅录数据库的数据量够大，所以点密度图在很大程度上已经能反映分布的趋势。如果遇到只需要观测数据集中趋势而不需要显示点的分布的情况，可以使用核密度图。

8.3.2 绘制核密度图

绘制核密度图的步骤与绘制点密度图的步骤基本一致，但需要将点密度制图函数 `geom_pointdensity()` 更换为核密度制图函数 `stat_density_2d()`，然后调整 `stat_density_2d()` 函数的一些参数。

```r
GIS_KDE <- ggplot() +
  geom_sf( data = chgis1911_pref_pgn,
           mapping = aes( geometry = geometry ),
           color = "grey60",
           size = 1,
           alpha = 0.5,
           fill = NA ) +
  geom_sf( data = chgis1911_prov_pgn,
           mapping = aes( geometry = geometry ),
           color = "black",
           size = 1.5,
           fill = NA) +
  stat_density_2d(
    data = JSL1911fall_2, # `stat_density_2d()`函数根据点分布自动计算核密度
    mapping = aes(x = st_coordinates(geometry)[,1],
                  y = st_coordinates(geometry)[,2],
                  fill = after_stat(level)), # 填充使用自动计算的 "level"
    color = NA,
    geom = "polygon",   # 绘制填充多边形
    contour = TRUE,     # 开启等高线
    alpha = 0.5) + # 设置透明度
  scale_fill_gradientn(  # 添加颜色渐变
    colours = c("grey80", "cyan", "green", "yellow", "red"),
    name = " 密度值 ",
    trans = "identity") +
  labs(title = "1911 年秋全国文官任职地分布核密度图 ",
       x = " 经度 ", y = " 纬度 ") +
  theme(text = element_text(family = "STKaiti",size = 12),
        panel.background = element_rect(fill = "white"),
        plot.background = element_rect(fill = "white"),
        plot.title = element_text(hjust = 0.5),
        panel.grid.major = element_line(color = "#88c4e8"
                                        , linewidth = 0.25))
```

```
GIS_KDE  # 显示图片
```

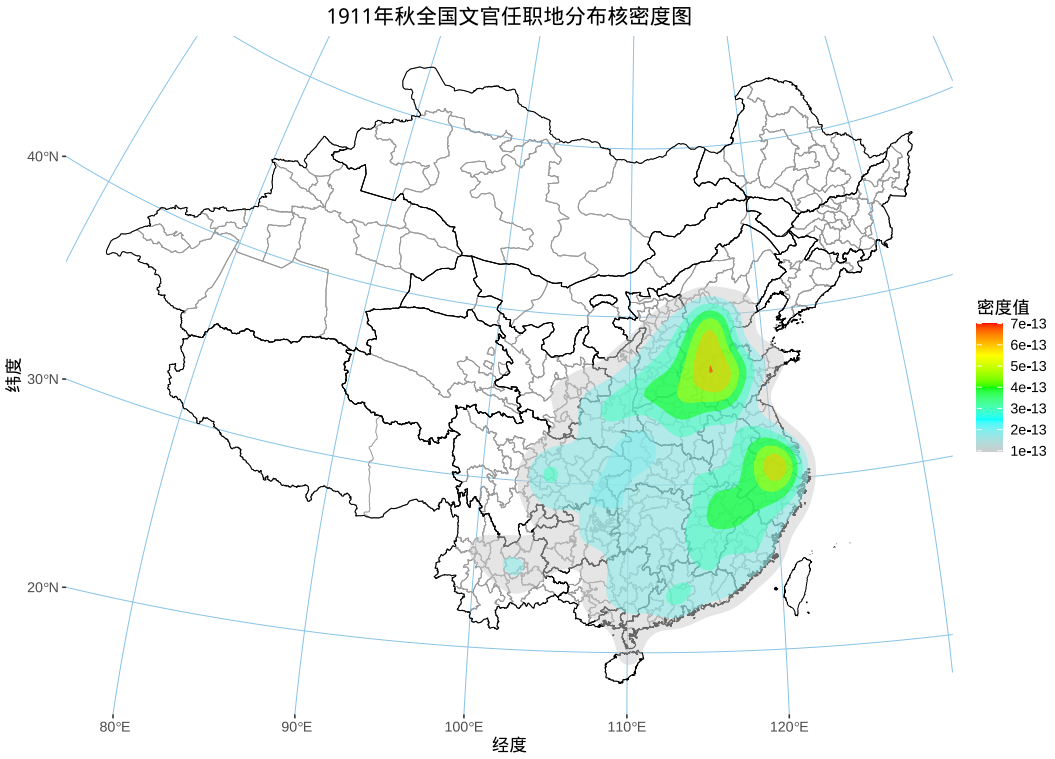

图 **11:** 1911 年秋全国文官任职地分布核密度图

根据以上核密度图可以发现，清政府在直隶山东一带、江浙地区设置了较多的文官，全国文官密度最高的区域在山东西部。结合黄河的走向可以发现，清末文官的分布与黄河下流沿岸地区有所重合，这或许能够说明清政府一直很重视黄河治理。核密度图较点密度图而言更容易观察到分布的集中趋势。

8.3.3 绘制热力图

热力图能够将空间划分为网格计算密度，进而通过色阶展示密度值，它的视觉效果更加直观，尤其适合高密度点数据可视化。热力图的绘图步骤与核密度图相似，只是要将 `stat_density_2d()` 函数的 `geom` 参数修改为 `raster`，将其转换为栅格热力形式。同时，需要将图层函数 `aes()` 中的 `fill` 参数改填充使用自动计算的 `density`，即密度值而非层级。同时需要加入 `coord_sf()` 函数限定地图显示范围，以防出现空白区域过多的情况。以下是代码示例：

```
GIS_Heatmap <- ggplot() +
  geom_sf( data = chgis1911_pref_pgn,
```

```
          mapping = aes( geometry = geometry ),
          color = "grey40",
          size = 1,
          alpha = 0.5,
          fill = NA ) +
  geom_sf( data = chgis1911_prov_pgn,
          mapping = aes( geometry = geometry ),
          color = "black",
          size = 1.5,
          fill = NA) +
  stat_density_2d(
    data = JSL1911fall_2,
    mapping = aes(x = st_coordinates(geometry)[,1],
                  y = st_coordinates(geometry)[,2],
                  fill = after_stat(density)),
    geom = "raster",   # 使用栅格热力图
    contour = FALSE,     # 关闭等高线
    alpha = 0.8      # 设置透明度
    ) +
  coord_sf(
      xlim = range(st_coordinates(JSL1911fall_2)[,1]),
      ylim = range(st_coordinates(JSL1911fall_2)[,2]),
      expand = FALSE) +  # 严格使用 xlim/ylim 指定的范围，不留任何边距
  scale_fill_gradientn(  # 添加颜色渐变
    colours = c("white", "cyan", "green", "yellow", "red"), # 指定五种颜色渐变
    name = " 密度值 ", # 图例标题
    trans = "identity") +
# 指定颜色渐变的变换，可以对颜色值进行对数变换、平方根变换等
  labs(title = "1911 年秋全国文官任职地分布热力图 ",
       x = " 经度 ",
       y = " 纬度 ") +
  theme(text = element_text(family = "STKaiti",size = 12),
        panel.background = element_rect(fill = "white"),
        plot.background = element_rect(fill = "white"),
        plot.title = element_text(hjust = 0.5),
        panel.grid.major = element_line(color = "black",
```

```
                                    linewidth = 0.25))
```

上述代码中，`coord_sf()` 是 `ggplot2` 专门用于地理空间数据的坐标系函数，可用于设置地图显示范围。`range()` 函数可以计算这些 X、Y 坐标的最小值和最大值，可以设置地图的水平方向显示范围，使其刚好包含所有数据点。加入 `coord_sf()` 和 `range()` 函数可以解决热力图显示太多空白区域的问题。

```
GIS_Heatmap  # 显示图片
```

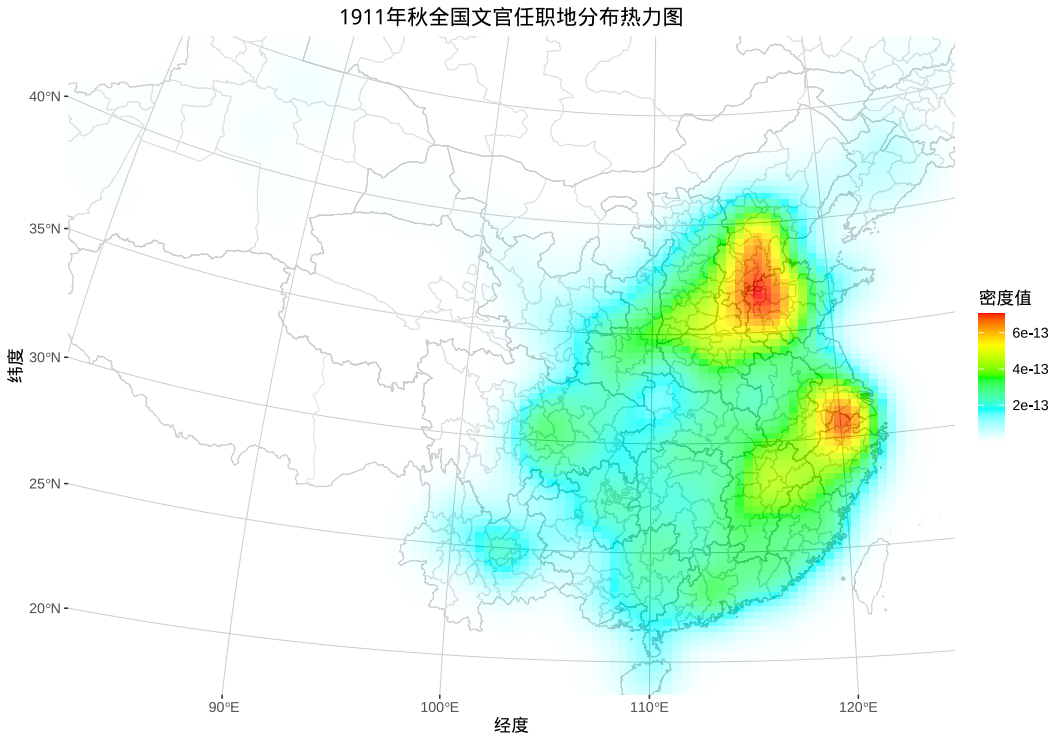

图 **12:** 1911 年秋全国文官任职地分布热力图

观察上图可以发现，清代文官任职地分布的高密度区域位于湖北地区和江南地区，且热力图与底图的契合度更高，能够更直观地看到高密度区域所在的府级行政单位，但其也容易出现分辨率不高的问题，需结合实际情况谨慎使用。

结合上述点密度图、核密度图和热力图的实际例子，可以归纳这三种图形在缙绅录数据库等历史数据库分析中的优势和适用场景。点密度图计算效率高，保留原始数据点，适用于较小规模数据集（在大规模数据集亦能展示分布趋势），展示微观结构；热力图解读直观性强，与底图契合度高，适用于大规模数据集，展示宏观趋势；而核密度图处于以上二者之间，尤其适用于展示空间结构，分析单中心或多中心的分布模式。使用者可结合自身数据库的提点和具体的场景选择合适的展示形式。

8.3.4 添加指北针、比例尺和制作、导出组合图

一幅完整的 GIS 图像除了底图和图层外，还需要添加指北针、比例尺和图像说明，帮助读者快速理解图形的方位、大小和数据来源。此外，还可以制作点密度图和核密度图的组合图。示例如下：

```
GIS_PointKDE <- ggplot() +
  geom_sf( data = chgis1911_pref_pgn,
           mapping = aes( geometry = geometry ),
           color = "grey60",
           size = 0.5,
           alpha = 1,
           fill = NA ) +
  geom_sf( data = chgis1911_prov_pgn,
           mapping = aes( geometry = geometry ),
           color = "black",
           size = 1.5,
           fill = NA) +
  geom_pointdensity(data = JSL1911fall_2,
                mapping = aes( x = st_coordinates(geometry)[,1],
                               y = st_coordinates(geometry)[,2] ),
                color = "red",
                size = 0.5,
                alpha = 0.6,
                adjust = 0.8) +
  stat_density_2d(data = JSL1911fall_2,
                  aes(x = st_coordinates(geometry)[, 1],
                      y = st_coordinates(geometry)[, 2],
                      fill = after_stat(level)),  # 使用计算后的层级填充
                  color = NA,
                  geom = "polygon",
                  contour = TRUE,
                  alpha = 0.5) +
  scale_fill_gradientn(
    colours = c("grey80", "cyan", "green", "yellow", "red"),
    name = " 核密度 ",
    trans = "identity") +
```

```
  theme(text = element_text(family = "STKaiti",size = 12),
        panel.background = element_rect(fill = "white"),
        plot.title = element_text(hjust = 0.5),
        plot.background = element_rect(fill = "white"),
        panel.grid.major = element_line(color = "#88c4e8", linewidth = 0.25),) +
  labs(title = "1911 年秋全国文官任职地分布组合图（每个点代表一名文官）",
       x = " 经度 ", y = " 纬度 ")  +
  annotation_scale( # 添加比例尺
    location = "bl",  # 底部左 left
    style = "ticks",  # 刻度样式，输入 "?annotation_scale" 以查看其他样式
    width_hint = 0.1  # 比例尺宽度
  ) +
  annotation_north_arrow( # 添加指北针
    location = "tr",  # 顶部右 right
    style = north_arrow_orienteering (text_size = 5),
  # 指北针样式，输入 "?annotation_north_arrow" 以查看其他样式
    height = unit(0.75, "cm"),
    width = unit(0.75, "cm")
  ) +
  annotate( # 添加数据说明
    "text", x = Inf, y = -Inf,
    label = c("Created By RStudio ggplot2
              Data sources: CHGIS & CGED-Q JSL"),
    hjust = 1.1, vjust = -0.5,
    size = 3, color = "black", family = "Songti SC"
  )

# 导出图形
ggsave("/Users/jc/Documents/R export/image/1911 年全国文官任职地分布组合图.png",
       plot = GIS_PointKDE,  # 设置你自己的文件保存路径
       width = 12, height = 10,
       dpi = 600,  # 高分辨率
       bg = "white")  # 设置背景色
```

```
GIS_PointKDE  # 查看图形
```

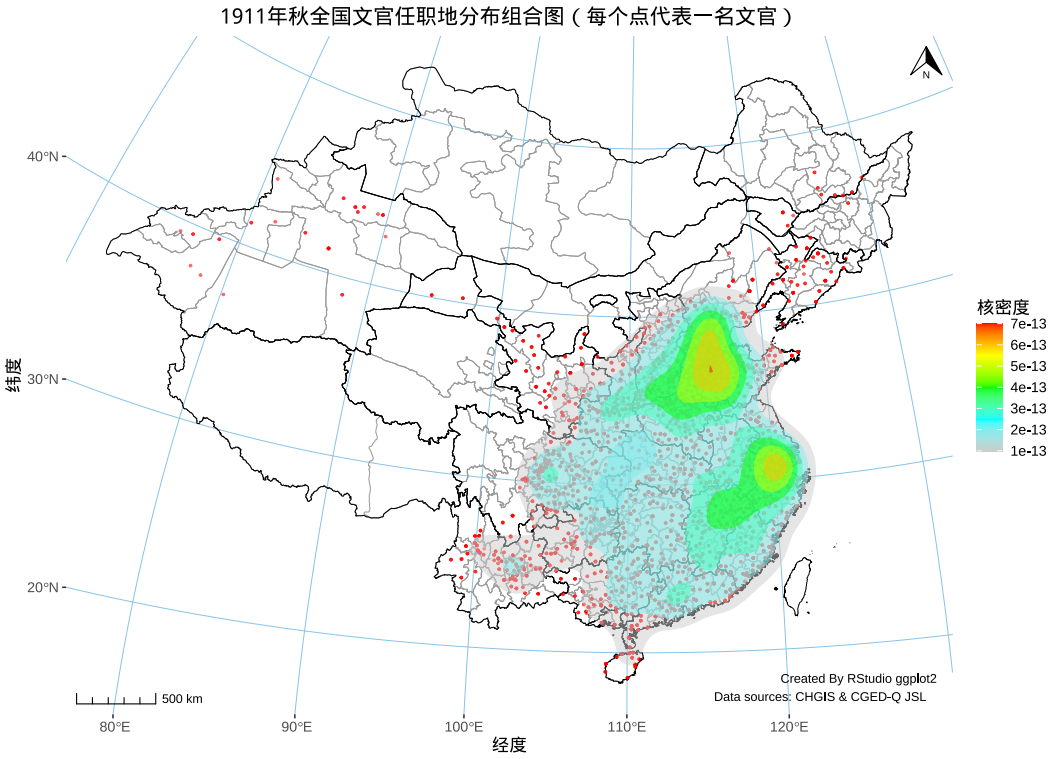

图 **13:** 1911 年秋全国文官任职地分布组合图

观察上图可以发现，系统在左下角添加了比例尺，在右上角添加了指北针，在右下角添加了数据说明，同时还将核密度图层置于点密度图层之上，形成了一幅相对完整的图形。

完成图片的制作后，可以使用 `ggsave()` 函数按指定文件路径导出图形，将其运用到研究之中。

8.4 制作局部 GIS 图像

在区域性的研究中，通常用到的是局部 GIS 图像而非完整 GIS 图像。如何制作一个局部 GIS 图像？只需筛选出底图中的指定要素，再筛选数据集中的指定要素，再用筛选之后的底图和数据集来制图，制图步骤与全局 GIS 图像制图步骤一致，正确选取绘制底图和点图层的数据集后可直接复制前节使用的标题、主题、指北针、比例尺、数据说明代码。下面以华北地区（另加陕西）为例，制作清代华北地区文官任职地分布的局部 GIS 图像。示例代码如下：

```r
# 提取 shapefile 中的局部文件
chgis1911_prov_pgn %>%
    filter(NAME_CH == " 直隶 " | NAME_CH == " 山西 " |
        NAME_CH == " 山东 " | NAME_CH == " 河南 " |
        NAME_CH == " 陕西 ") -> chgis1911_prov_pgn_huabei
chgis1911_pref_pgn %>%
    filter(LEV1_CH == " 直隶 " | LEV1_CH == " 山西 " |
        LEV1_CH == " 山东 " | LEV1_CH == " 河南 " |
        LEV1_CH == " 陕西 ") -> chgis1911_pref_pgn_huabei

# 提取件缙绅录数据库中的局部文件
JSL1911fall_2 %>%
    filter(LEV1_CH == " 直隶 " | LEV1_CH == " 山西 " |
        LEV1_CH == " 山东 " | LEV1_CH == " 河南 " |
        LEV1_CH == " 陕西 ") -> JSL1911fall_2_huabei

# 制作 GIS 图像
GIS_Regional <- ggplot() + geom_sf(data = chgis1911_pref_pgn_huabei,
    mapping = aes(geometry = geometry), color = "#CCCCCC",
    size = 1, alpha = 0.5, fill = NA) + geom_sf(data = chgis1911_prov_pgn_huabei,
    mapping = aes(geometry = geometry), color = "black",
    size = 2, fill = NA) + geom_pointdensity(data = JSL1911fall_2_huabei,
    mapping = aes(x = st_coordinates(geometry)[, 1],
        y = st_coordinates(geometry)[, 2]), color = "#f47720",
    size = 0.5, alpha = 0.6, adjust = 0.8) + labs(title = "1911 年秋华北地区文官任职地分布点
    ↪ 密度图（每个点代表一名文官）",
    x = " 经度 ", y = " 纬度 ") + theme(text = element_text(family = "STKaiti",
    size = 12), panel.background = element_rect(fill = "white"),
    plot.title = element_text(hjust = 0.5), plot.background = element_rect(fill =
    ↪ "white"),
    panel.grid.major = element_line(color = "#88c4e8",
        linewidth = 0.25), ) + annotation_scale(location = "bl",
    style = "ticks", width_hint = 0.1) + annotation_north_arrow(location = "tr",
    style = north_arrow_orienteering(text_size = 5),
    height = unit(0.75, "cm"), width = unit(0.75, "cm")) +
    annotate("text", x = Inf, y = -Inf, label = c("Created By RStudio ggplot2
```

```
            Data sources: CHGIS & CGED-Q JSL"),
        hjust = 1.1, vjust = -0.5, size = 3, color = "black",
        family = "Songti SC")
```

```
GIS_Regional  # 显示图片
```

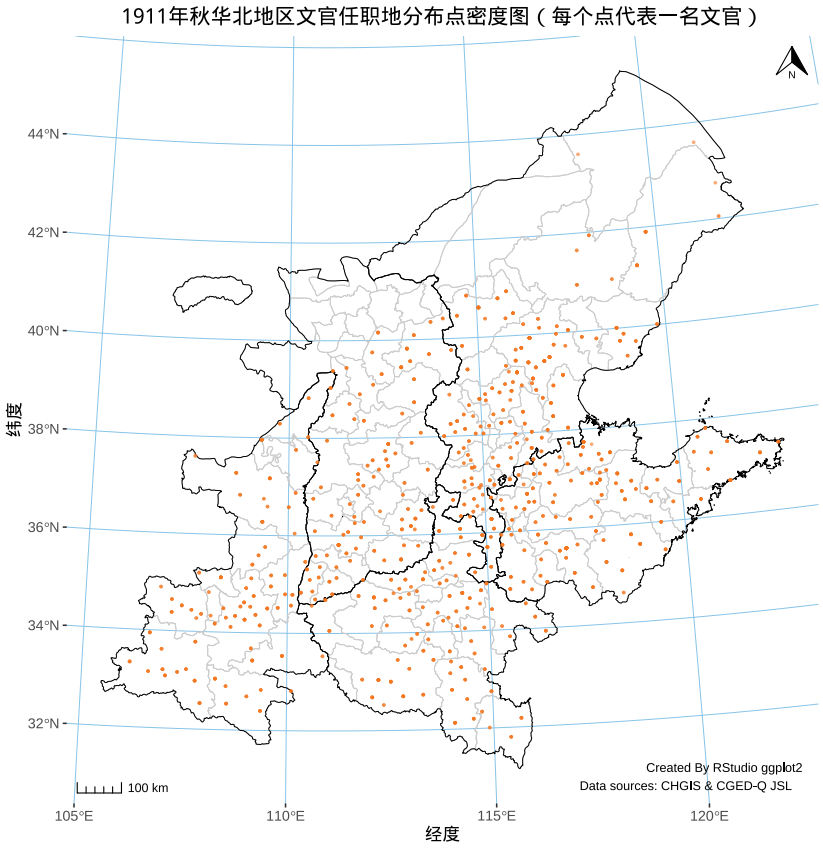

图 14: 1911 年秋华北地区文官任职地分布点密度图

上图为 1911 年华北地区（另加陕西）局部 GIS 图像，包含直隶、山东、河南、山西、陕西五省，图中一块飞地为清光绪二十九年设立的五原直隶厅，隶山西省，在今内蒙境内。从上图可以看出，华北地区（另加陕西）文官的分布似与省级边界有关联，陕西山西交界地区、山西河南交界地区、直隶山东河南三省交界地区官员密度较高。如果考虑黄河的流向，会发现华北地区（另加陕西）文官分布似与黄河走向有紧密关联，尤其在下游河道地区，官员的密度较高，这似乎表明清政府十分重视黄河的治理问题。

尤其需要说明的是，图中部分地区官员分布没有显示出来，如京城应分布有大量京官，这是由于在代码编写阶段为保持制图代码的简洁性和连贯性，略去了对被匹配数据集变量格式的整理环节，使用者可根据自身研究需求结合 4.2 节和 6.2.1 节整理变量的内容对局部或者全局数据作进一步精细化处理，进而提高数据的匹配率，生成更佳完整的 GIS 图像。

8.5 利用离散数据制作分级统计图

缙绅录数据库中没有对应的离散数据，可以转换表格生成的频数，比如可以利用 table() 函数统计文官省籍的分布状况，利用 as.data.frame() 函数将其转换为数据框，并将其与 CHGIS 省级边界数据进行匹配，合并为一个既带有离散数据又带有地理坐标的数据集，利用该数据集便能制作分级统计图。示例如下：

```
# 将表格数据转换为数据框
JSL1911fall 籍贯省 <- as.data.frame(table(JSL1911fall$籍贯省))
colnames(JSL1911fall 籍贯省) <- c(" 省份 ", " 文官数量 ") # 修改列名

# 匹配 shapefile 文件以赋予地理边界信息
JSL1911fall 籍贯省 _2 <- merge(JSL1911fall 籍贯省,
                        chgis1911_prov_pgn,
                        by.x = " 省份 ", by.y = "NAME_FT")

# 制图之前生成一个新的变量对各省产生的文官数量进行分级，设置分级间隔、标签等，以便制图
JSL1911fall 籍贯省 _2 <- JSL1911fall 籍贯省 _2 %>%
  mutate(文官数量分级 = cut(文官数量, # 对数据集中的变量切分
       breaks = c(0,300,600,900,1200,1500), # 指定分级间隔
       include.lowest = TRUE,
       labels = c("<300","300-600","601-900","901-1200",">1200"))) # 设置各级别标签

# 将制图所需的文件转换为 sf 对象，再转换为带有坐标系统的文件
JSL1911fall 籍贯省 _2 <- st_as_sf(JSL1911fall 籍贯省 _2)
JSL1911fall 籍贯省 _2 <- st_transform(JSL1911fall 籍贯省 _2)

# 制作 GIS 图像完整图像（带有图例、指北针、比例尺和图像说明）
GIS_Choropleth <- ggplot(JSL1911fall 籍贯省 _2,aes(geometry = geometry)) +
  geom_sf( mapping = aes(fill = 文官数量分级)) + # 填充采用前述生成的分级变量
  scale_fill_manual(name = " 文官数量 ", # 图例名称
                    values = c(  "#E0F0F5", "#2A6E96", "#9C6E4A", "#C45C3D","#9D2235" )) +
  theme(text = element_text(family = "STKaiti",size = 12),
        panel.background = element_rect(fill = "white"),  # 设置面板背景颜色
        plot.title = element_text(hjust = 0.5),
        plot.background = element_rect(fill = "white"),
        panel.grid.major = element_line(color = "#88c4e8", linewidth = 0.25),) +
```

```
  labs(title = "1911 年秋全国文官省籍分布图 ",
       x = " 经度 ", y = " 纬度 ")  +
  annotation_scale(
    location = "bl",
    style = "ticks",
    width_hint = 0.1
  ) +
  annotation_north_arrow(
    location = "tr",
    style = north_arrow_orienteering (text_size = 5),
    height = unit(0.75, "cm"),
    width = unit(0.75, "cm")
  ) +
  annotate(
    "text", x = Inf, y = -Inf,
    label = c("Created By RStudio ggplot2
              Data sources: CHGIS & CGED-Q JSL"),
    hjust = 1.1, vjust = -0.5,
    size = 3, color = "black", family = "STKaiti"
  )
```

```
GIS_Choropleth  # 显示图片
```

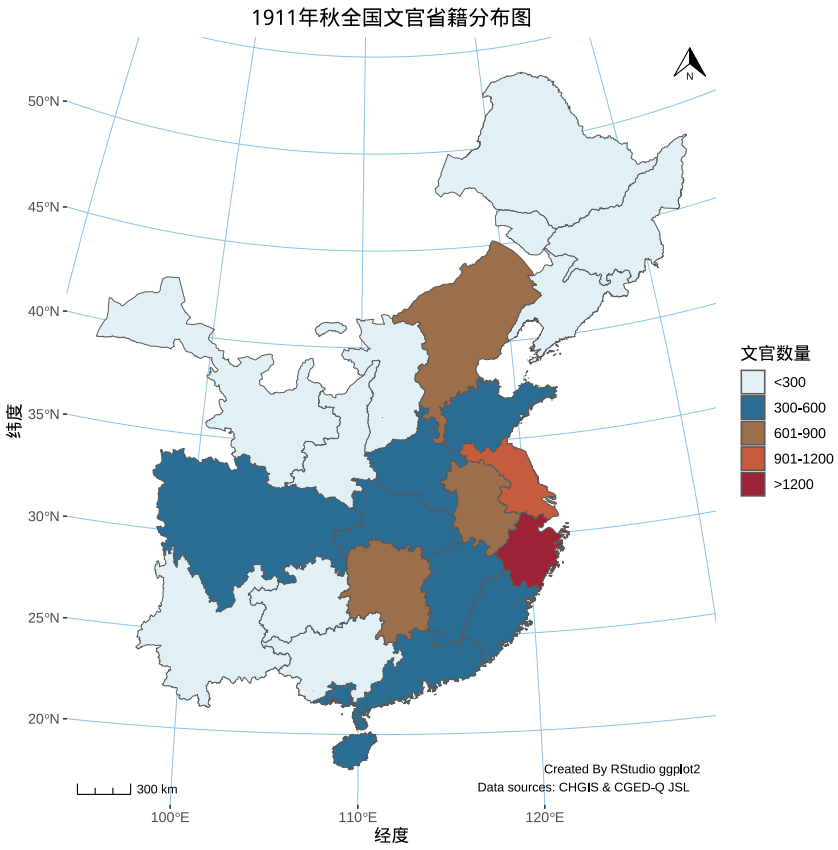

图 15: 1911 年秋全国文官省籍分布图

从上图中可以看出，浙江、江苏、安徽、直隶和湖南产生了较多的文官，这与清代江南地区经济、文化、教育较为发达有关。需要注意的是，图中未显示区域为数据不全或匹配不成功，使用者若需要更精细的匹配结果，可参考 4.2 节和 6.2.1 节整理变量的内容对“籍贯省 ”变量进行进一步整理，使其与 CHGIS 数据的匹配度更高。

上述内容为点密度图、核密度图、热力图、组合图、局部地图和分级统计图的制作方法，使用者可根据研究内容和场景选择合适的可视化图形，但需要注意的是在制图之前要对缙绅录数据库的制图关键变量进行精细化处理，以满足底图数据的格式，提高匹配率和可信度。

第九章 总结

相对于其它分析软件，R 语言对初学者的数学基础和计算机基础要求更高。R 语言本质上是一个语言编程的环境，而不是一个分析软件。利用 R 语言，是在创造一种联系，而用分析软件，只是在分析数据。这就是计算机编程语言和计算机软件的本质区别之一。

本教程从零开始介绍缙绅录数据库在 R 语言中的运用，从一些最基本的操作，逐步延伸到比较高级的操作，目前介绍到的函数有：

1. 导入和读取文件 导入数据集 `read_excel()`、`read_dta()` 等、保存文件 `save()` 等

2. 创建新变量 转换变量类型 `as.numeric()`、创建新变量 `$` 符号的运用、逻辑表达式 `ifelse()`、数值与字符变量互换 `[]` 符号的运用、串联字符 `paste()`、提取判断字符 `str_extract()` 和 `str_detect()`、替换字符 `gsub()` 等

3. 制表 简单制表 `table()`、整理变量 `factor()`、转换为数据框函数 `as.data.frame()` 以及其转置 `as.data.frame.matrix()`、制作可以导出的表格 `flextable()`、`table1()`、指定条件制表 `subset()` 以及 & 和 | 符号的运用，还包括 `flextable` 包制表时的诸多函数，请详见 4.3.1

4. 直方图、散点图和折线图 简单直方图 `ggplot()`、`geom_bar()`、进阶直方图 `labs()`、`guides()`、`scale_fill_manual()`、`theme()`、`scale_x_continuous()`、`scale_y_continuous()`、`geom_text()`、散点图 `geom_point()`、折线图 `geom_line()` 等

5. 数据集的匹配 管道符`%>%` 的运用、`select()`、筛选行 `filter()`、匹配数据集以整理变量 `merge()`、连接两个数据集 `inner_join()` 等

6. 创建官员个人编号 排序 `arrange()`、分组 `group_by()`、生成新变量 `mutate()`、累加 `cumsum()`、匹配函数 `stringdist_inner_join()` 等

7. 制作 GIS 图像 将数据集转换为适用于制作地图的格式 `st_read()` 和 `st_as_sf()`、制作底图 `geom_sf()`、点密度图 `geom_pointdensity()`、核密度图热力图 `stat_density_2d()`、添加颜色渐变函数 `scale_fill_gradientn()`、设置地图显示范围 `coord_sf()`、添加比例尺 `annotation_scale()`、添加指北针 `annotation_north_arrow()`、添加图像说明 `annotate()`、导出图形 `ggsave()` 等

以上为本教程的总结。