R 语言

在历史数据库分析中的运用

——以《缙绅录》数据库为中心的基础教程

陈 俊

华中师范大学

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

2023 年 9 月

V 1.1.5

R for the Analysis of Historical Databases：A Training Guide Based on the CGED-Q JSL

Chen Jun
Central China Normal University

The preparation of the User Guide received support from Hong Kong Research Grants Council General Research Fund 16602621 (Campbell PI) and Areas of Excellence AoE/B-704/22-R (Chen PI).

September 2023

V 1.1.5

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、优势明显，深受统计软件使用者的青睐。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持着“取之于斯、用之于斯”的共享精神，开发出一些功能强大的包来扩展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。

但是，R语言也具有一些局限性：第一，学习和使用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，R语言的运行结果受电脑环境的影响较大，R语言的代码在不同硬件、运行环境下有时会产生不同的运行效果，因此，R语言的代码并不能直接套用，需要结合电脑环境调整代码的细节，同时，运行报错后回溯报错原因也需要相当的时间。

R语言目前在国外已经成为最受欢迎的数据分析软件之一，但在国内普及度还不够高。目前针对R语言的教程主要有：《R语言实战（第2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。

不过，目前还没有专门的教程讲解R语言在历史学当中的运用。在历史学的发展中，历史数据库的建设成为了一个新兴的方向，大量历史数据的存在为历史数据库的发展奠定了基调。目前，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），历史数据库建设方兴未艾，在未来也会成为历史学发展的主要方向之一。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R语言、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”数据库（JSL database 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、数据集链接等内容。“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。同时，“缙绅录”数据库具备高度结构化的特点，利用字符串作为存储方式，其体量庞大，有数十万条记录，在历史数据库中具有典型性。此外，研究生两年的时间中，我在缙绅录数据库的分析中积累了一些经验，编写代码后R语言报错通常都能找到缘由，也能够及时解决。鉴于缙绅录数据库的知名度和我对缙绅录的熟悉程度，我选择缙绅录数据库作为分析目标，希望为历史数据库分析提供一些范例。

我本科学习经济学，硕士阶段转向历史学，因而同时对数据分析和历史数据库感兴趣，并通过不断自学，游走于各个数据分析软件之间，但都只学会了一些皮毛。就我使用数据分析软件的经验，STATA虽功能全面，集成度高，但价格昂贵，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；而各种GIS软件是历史数据可视化的必学软件，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。我最初接触到R语言，完全处于迷茫的状态，代码格式不会变通，运算符不会套用，时常为了一个运行报错而纠结很长时间，不断地在网上检索解决方法并补充基础知识，没有他人指导，走了许多弯路。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。

我是华中师范大学中国近代史研究所一名研二的学生（cjccnu@163.com），我在担任康文林老师（Prof.Cameron Campbell，香港科技大学教授、华中师范大学特聘教授）“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 语言程辑包。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，并鼓励我将这一套教学 PPT 和 R 语言程辑包整合成一部教程，同时提供了很多开创性的指导意见，为本教程的编写提供了巨大的帮助。同时，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同学、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。本教程是一个初步的尝试，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。

在 Rstudio 中打开 R 语言程辑包（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。

陈俊 于桂子山

2023 年 9 月

目录

前言.....i
第 1 章　R 以及 RStudio 的安装与界面.....1
1.1　R 和 RStudio 的关系.....1
1.2　R、RStudio 的下载.....1
1.3　RStudio 的界面.....2
1.4　R 语言的自学资源.....6
第 2 章　数据库的管理.....9
2.1 《缙绅录》数据库及下载方式.....9
2.1.1 利用《缙绅录》数据库的相关研究.....9
2.1.2 《缙绅录》数据库的下载方式.....10
2.2 在 RStudio 中导入数据库.....11
2.3 在 RStudio 中查看、编辑、保存数据库.....12
第 3 章　字符串处理.....14
3.1 在数据库中创建一个新变量.....14
3.1.1 运算符.....14
3.1.2 删除数据库中的缺失值.....14
3.1.3 转换变量的类型.....15
3.1.4 创建新变量.....16
3.2 逻辑表达式.....17
3.3 转换变量.....18
3.3.1 将字符型变量转换为数值型变量.....18
3.3.2 将数值型变量转换为字符型变量.....19
3.3.3 将离散型变量转换为数值型变量.....20
3.4 字符串处理的相关函数.....22
3.4.1 串联字符.....22
3.4.2 提取和判断字符.....22
3.4.3 替换字符.....24
第 4 章　制表.....25
4.1 简单制表.....25
4.2 整理变量.....25
4.3 制作可以导出的表格.....29
4.4 指定条件制表.....34
第 5 章　直方图、散点图、折线图.....38
5.1 直方图.....38
5.1.1 简单直方图.....38
5.1.2 分条直方图.....40
5.1.3 堆积直方图.....45
5.1.4 利用非数值型变量和离散型变量制作直方图.....47
5.2 散点图.....48
5.3 折线图.....51
第 6 章　数据集的匹配.....54

6.1 筛选列与行 54
6.1.1 筛选列 54
6.1.2 筛选行 55
6.2 数据集的匹配 55
6.2.1 匹配以整理变量 55
6.2.2 如何创建一个副数据集 61
6.3 连接两个数据集 63
第 7 章 数据集的内外链接 65
7.1 数据集内部记录的链接 65
7.1.1 创建官员的个人编号 65
7.1.2 利用个人编号分析官员任职年限 67
7.2.3 如何统计人数而非记载数 69
7.2 数据集与外部数据库的链接（模糊匹配） 71
第 8 章 总结 75
附录 76
本教程使用到的代码合集 76
R 语言教学 PPT 和程辑包下载网址 83

表目录

表 3.1 基础运算符 14
表 3.2 NBA 球星得分、篮板、助攻、比赛场次、场均得分表（离散数据库示例） 20
表 6.1 不同民族属性官员的铨选方式差异 62

图目录

图 1.1 RStudio 的操作界面 2
图 1.2 RStudio 的区域一 3
图 1.3 RStudio 的区域二 4
图 1.4 RStudio 的区域三 4
图 1.5 RStudio 的区域四 5
图 1.6 RStudio 的功能栏 6
图 2.1 区域一、区域二中的数据库信息 12
图 2.2 编辑数据库界面 13
图 4.1 缙绅录数据库中官员的籍贯和满汉比例 31
图 4.2 所有官员以及贡生群体的满汉比例和籍贯 32
图 4.3 出身为贡生、“鼎甲”的知县的满汉比例和地理来源 33
图 4.4 贡生群体的满汉比例与地理来源 36
图 5.1 简单直方图制图效果 39
图 5.2 兵部官员满汉比例代码运行效果 41
图 5.3 兵部官员满汉比例（加入修饰函数 theme（）后） 43
图 5.4 图例修饰函数 theme（）中各参数的用法 44
图 5.5 图里修饰函数 theme（）参数用法图解 45
图 5.6 兵部满汉比例（堆积直方图） 46
图 5.7 “NBA”数据集相关代码运行效果 48
图 5.8 1900-1906 兵部官员的品级和数量（散点图） 50
图 5.9 1900-1906 兵部官员的品级和数量（折线图） 53

图 6.1 缙绅录数据中官员的出身构成和满汉比例 58
图 6.2 1900-1912 年文官出身结构变化 61
图 7.1 利用个人编号来查看官员任职年限 69
图 7.2 1900-1906 年兵部的地理来源（人数） 70

第1章 R 以及 RStudio 的安装与界面

1.1 R 和 RStudio 的关系

R 语言（The R Programming Language），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。

R 语言能够取得如此广泛的学科基础，离不开共享精神（the spirit of sharing）的支撑：R 语言一经问世便是开源的，R 语言官网上的第一句话便是“R is a free software”，不同于许多价格昂贵的统计分析软件，R 语言开发的源代码甚至能在互联网上找到。也正是因为如此，R 语言的一些使用者秉承着 R 语言的共享精神，开发出许多功能强大的程序包以扩展 R 语言的功能，供社会大众使用。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 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、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（79 megabytes，64 bit）”[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，且文件路径中尽量不要出现中文字符。

下载好 R 语言之后，进入 RStudio Desktop - Posit 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

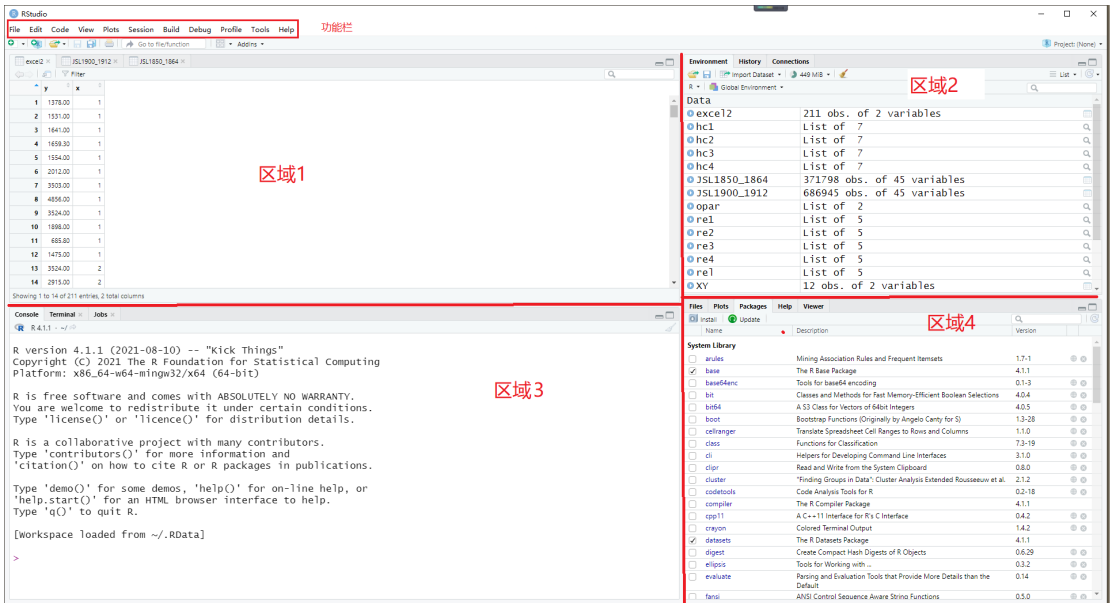


图 1.1 RStudio 的操作界面

如图 1 所示，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 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。

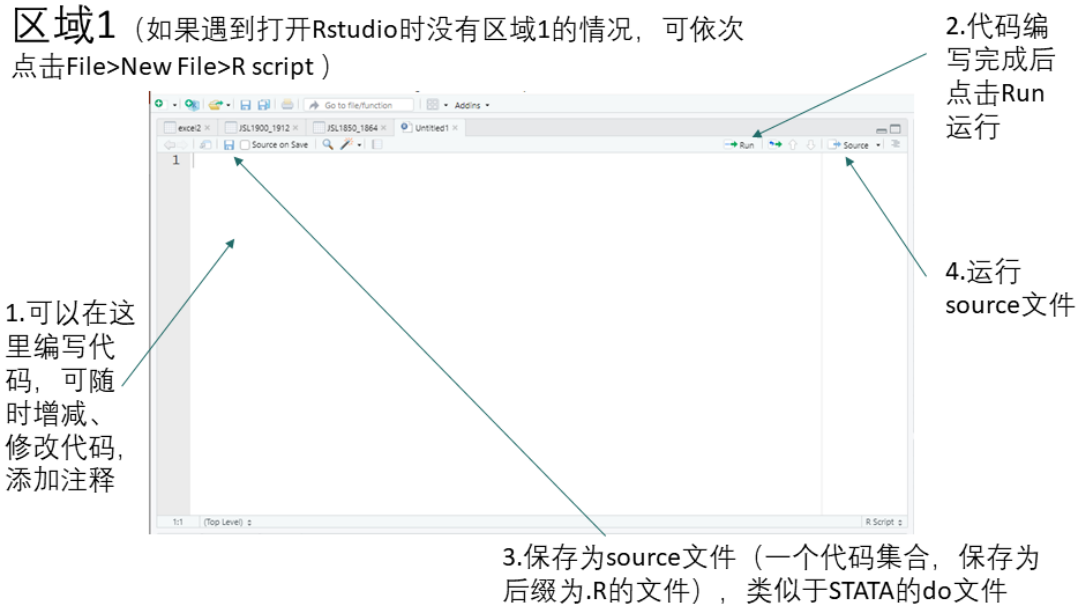


图 1.2 RStudio 的区域一

区域一是 RStudio 的命令编写区，如果遇到打开 Rstudio 时没有区域 1 的情况，可依次点击 File>New File>R script。下面介绍几个常用的按钮：

“空白区域”：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。

“Run”：选中代码，点击 Run 运行。

“Source on save”左边的保存按钮：将编写的代码保存为 source 文件（一个代码集合，保存为后缀为.R 的文件），类似于 STATA 的 do 文件。

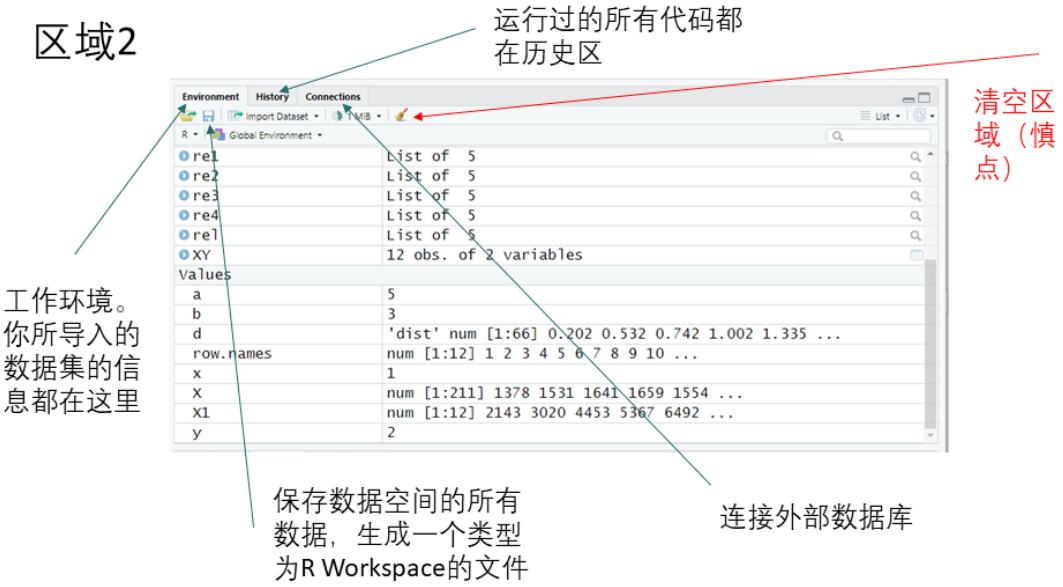


图 1.3 RStudio 的区域二

区域二是数据窗口，主要功能如下：

“Environment”：展示导入的数据集（库）。

“History”：运行过的所有代码都保存在历史区中

“Connections”：可用于连接外部数据集，在基础环节用得很少。

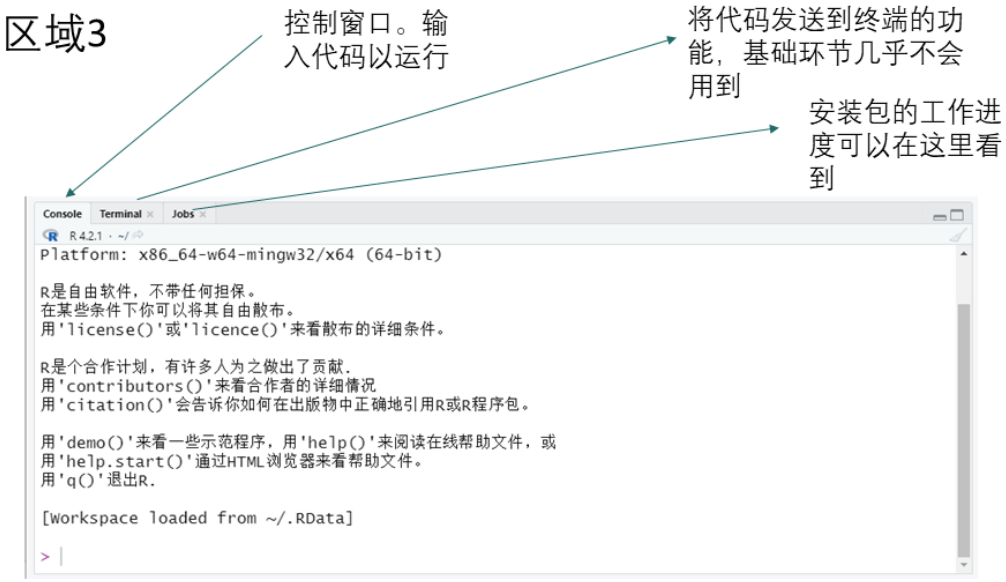


图 1.4 RStudio 的区域三

区域三是控制台，主要功能如下：

“Console”：控制区，可在此输入代码。

“Terminal”：将代码发送到终端，基础环节用得比较少。

“Jobs”：可以查看安装包的工作进度。

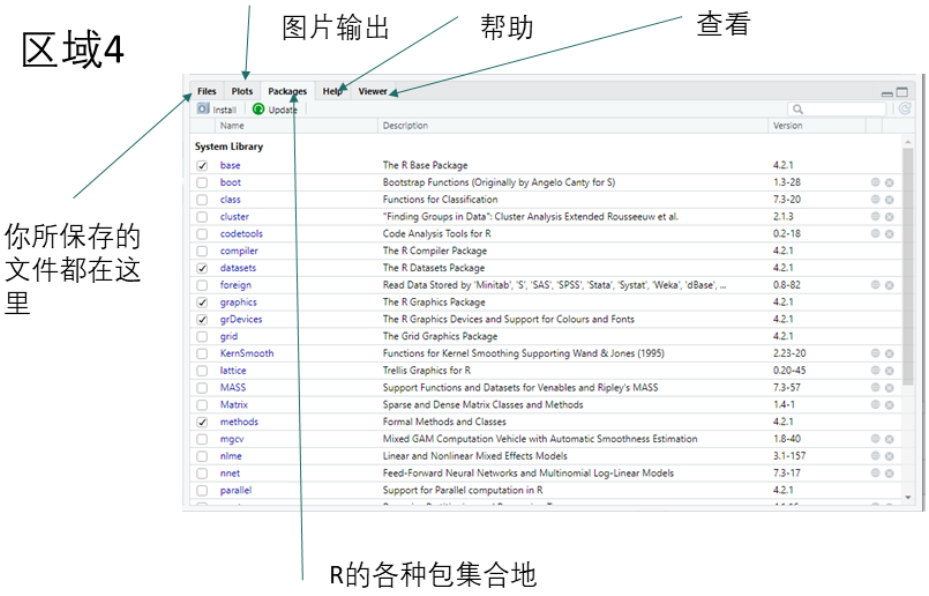


图 1.5 RStudio 的区域四

区域四是输出窗口，主要功能如下：

“Files”：保存文件的区域。

“Plots”：图片输出区域。

“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。

“Help”：帮助显示区。

“Viewer”：查看区。

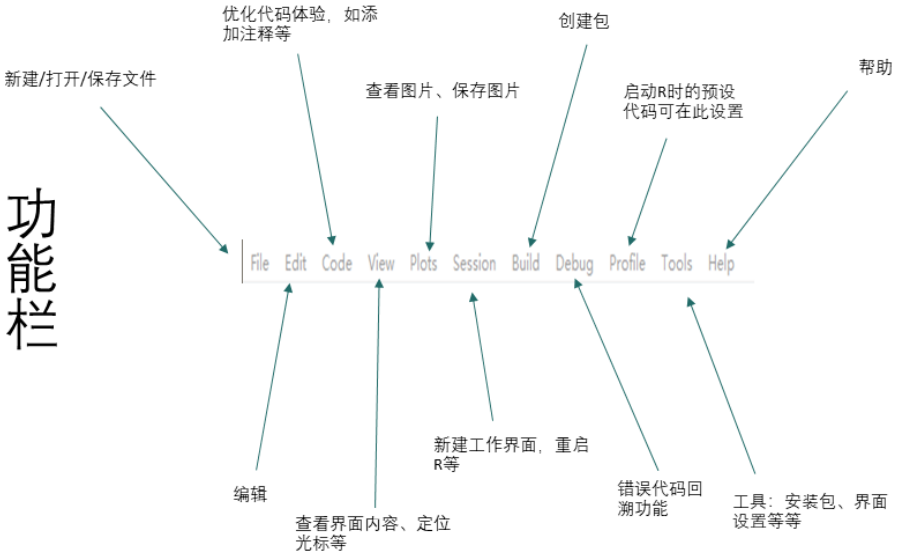


图 1.6 RStudio 的功能栏

功能栏的主要按钮如下：

“File”：新建、打开、保存文件等功能。

“Edit”：编辑功能

“Code”：优化编码体验，如添加注释等。

“View”：查看界面内容、定位光标等。

“Plots”：查看图片、保存图片。

“Session”：新建工作界面、重启 R 等。

“Build”：创建包。

“Debug”：错误代码回溯功能。

“Profile”：预设代码功能，预设代码可在 R 启动时自动运行。

“Tools”：工具、安装包、界面设置等功能。

“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》，[①]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，目前任教于美国著名的私立文理学院——维思大学（又译卫斯理大学）。[②]《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。[③]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： ggplot2--数据可视化、dplyr--数据处理、tidyr--数据清理、readr--数据提取、purrr--函数化编程、tibble--数据类型定义。[④]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。

① 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。
② https://www.wesleyan.edu/academics/faculty/rkabacoff/profile.html.
③ https://www.zhihu.com/tardis/zm/art/366605041?source_id=1005.
④ tidyverse项目的地址：https://github.com/tidyverse/tidyverse.

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。

第2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、中国四清阶级成分数据库，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。[①]

2.1.1 利用《缙绅录》数据库的相关研究

《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。[②]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；[③]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

① 引用自 http://vis.cse.ust.hk/searchjsl/. 李康团队的官网是 https://shss.hkust.edu.hk/lee-campbell-group/.
② 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<缙绅录>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。
③ 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。

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；[①]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。[②]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。[③]

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。

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”[④]，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》数据库包含两个年代（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以及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。[⑤]

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”[⑥]，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”，

① 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。
② 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。
③ 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。
④ 香港科技大学数据空间网站 https://dataspace.hkust.edu.hk/dataset.xhtml?persistentId=doi:10.14711/dataset/E9GKRS. 同时，哈佛大学Dataverse网站上也有数据库及其延伸数据集可供下载 https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/GMQWVZ。
⑤ 目前的最新版本为2023年7月17日更新，后续研究团队会继续更新数据库，请使用者留意。
⑥ 中国人民大学清史数据共享平台：http://47.103.109.28/DownloadFile/DLFile。该下

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、数据库处理的便捷代码集等资源。我们建议使用者选择“CGED-Q Public Release 1900-1912 27 Jul 2021.dta（DTA 格式）”和“CGED-Q Public Release 1850-1864 20 Dec 2021（DTA 格式）”，这两个分别是缙绅录数据库清末部分和缙绅录数据库太平天国部分，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。[①]

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：

```
1. library(haven)

2. dataset <- read_dta("E:/R/JSL1900-1912.dta")
```

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。

载地址截止到 2023 年 9 月，在后续可能会发生变化，请使用者留意。

① 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》。最新版用户指南下载地址：https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/GMQWVZ。同时，也可在在中国人民大学清史数据共享平台上下载，请参照 2.1.2 节。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，目前用户指南的最新版本为 2022 年 4 月版，我们建议使用者留意并下载最新版本的用户指南。

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：

```
3.  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、编辑、保存数据库

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。

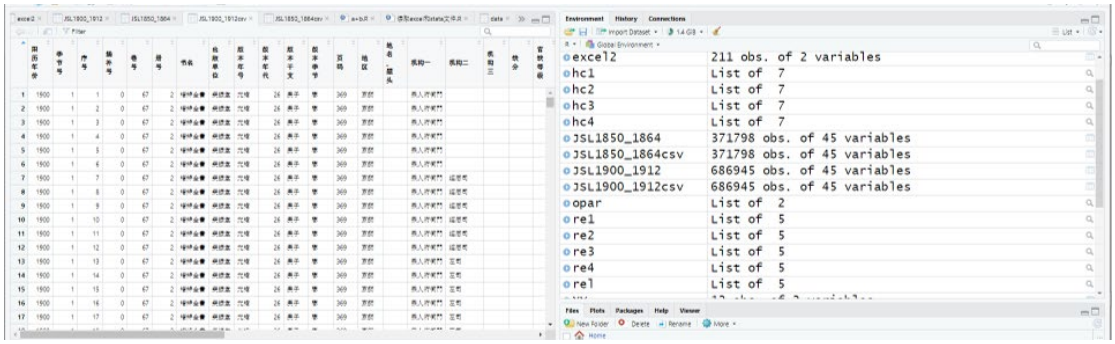
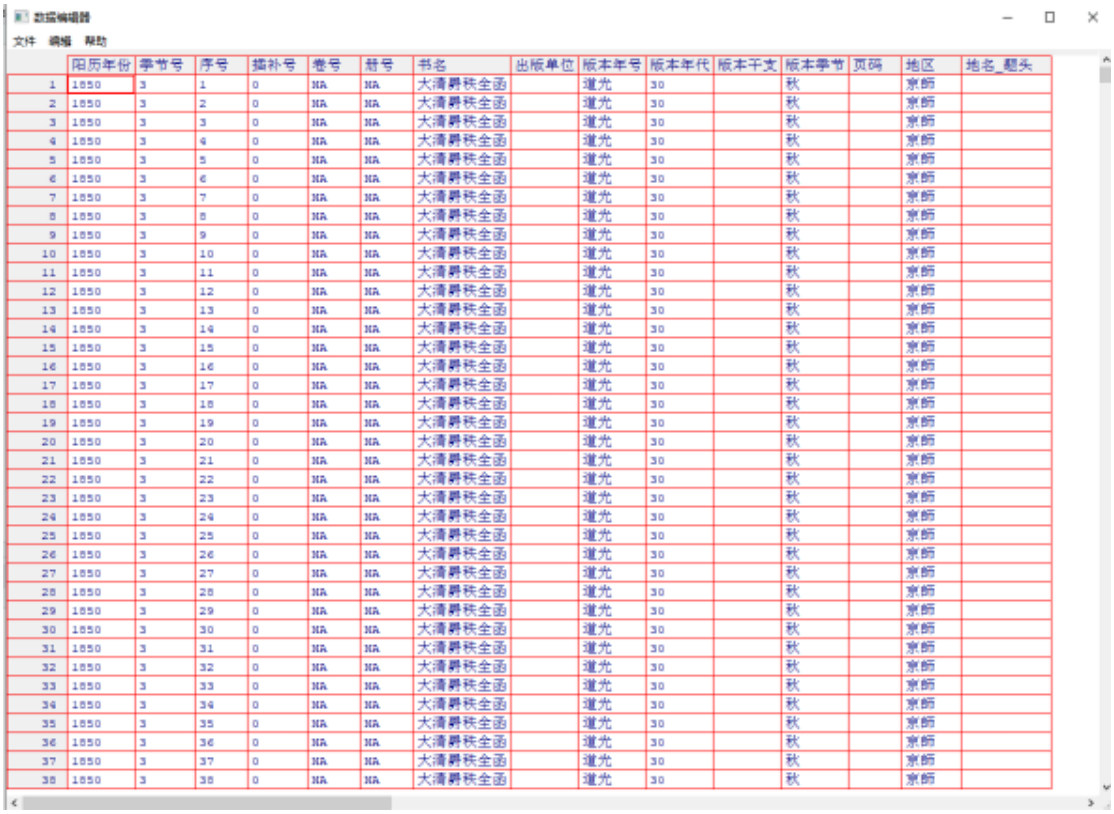


图 2.1 区域一、区域二中的数据库信息

编辑数据。编辑数据可以用到如下函数：

```
4.  edit(JSL1850_1864csv)
```

请注意，edit（）函数里面的内容是需要编辑的数据集，请使用者根据需要编辑的数据库名称更改函数括号中的内容。效果如下：


数据编辑器

文件 编辑 帮助

	阳历年份	季节号	序号	插补号	卷号	册号	书名	出版单位	版本年号	版本年代	版本干支	版本季节	页码	地区	地名_题头
1	1850	3	1	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
2	1850	3	2	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
3	1850	3	3	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
4	1850	3	4	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
5	1850	3	5	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
6	1850	3	6	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
7	1850	3	7	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
8	1850	3	8	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
9	1850	3	9	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
10	1850	3	10	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
11	1850	3	11	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
12	1850	3	12	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
13	1850	3	13	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
14	1850	3	14	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
15	1850	3	15	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
16	1850	3	16	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
17	1850	3	17	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
18	1850	3	18	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
19	1850	3	19	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
20	1850	3	20	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
21	1850	3	21	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
22	1850	3	22	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
23	1850	3	23	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
24	1850	3	24	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
25	1850	3	25	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
26	1850	3	26	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
27	1850	3	27	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
28	1850	3	28	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
29	1850	3	29	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
30	1850	3	30	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
31	1850	3	31	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
32	1850	3	32	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
33	1850	3	33	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
34	1850	3	34	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
35	1850	3	35	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
36	1850	3	36	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
37	1850	3	37	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	
38	1850	3	38	0	NA	NA	大清鹤秩全函		道光	30		秋		京師	

图 2.2 编辑数据库界面

保存数据。保存文件，点击 File>save all 即可。区域一的文件点击保存按钮保存为.R 为后缀的文件；区域二也要点保存按钮生成类型为 R Workspace 的文件。我们强烈建议使用者及时保存文件以免丢失。

第3章 字符串处理

3.1 在数据库中创建一个新变量

3.1.1 运算符

下表是RStudio和R中的基础运算符：

表 3.1 基础运算符

+	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

在历史数据库分析的过程中，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“空白”：

```
5. kongbaiming <- which(JSL1900_1912$名=="空白")

6. JSL1900_1912_clean <- JSL1900_1912[-kongbaiming,]
```

第5条代码的思路是：创建一个新变量名字叫做“kongbaiming”，然后利用which（）函数找出名字是“空白”的记录，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。“kongbaiming”是新变量的名称，可自行拟取；“<-”表示等于；“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。

第6条代码的思路是:创建一个新数据库,条件是原数据库删除掉“空白名”的记录，之后的分析都是基于新数据库来进行的。

“JSL1900_1912_delete_kongbaiming”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;“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_delete_kongbaiming”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，我们建议使用者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)
```

“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：

```
JSL1900_1912_clean$阳历年份 numeric <-
as.numeric(JSL1900_1912_clean$阳历年份)
typeof(JSL1900_1912_clean$阳历年份 numeric)
JSL1900_1912_clean$季节号 numeric <-
as.numeric(JSL1900_1912_clean$季节号)
```

```
10. typeof(JSL1900_1912_clean$季节号 numeric)
```

第 7 条代码的思路是：将“JSL1900_1912_delete_kongbaiming”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”。

第 8 条代码利用了 typeof（）函数，typeof（）函数能够返回变量的属性，可以检验变量类型转换是否成功。

第 9 条代码思路和第一条代码类似，只是将“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。

第 10 条代码思路和第二条一致。

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：

```
11.  JSL1900_1912_clean$年份季节 <- ( JSL1900_1912_clean$阳历年
     份 numeric + ( ( JSL1900_1912_clean$季节号 numeric/4 )-0.25) )
12. table(JSL1900_1912_clean$年份季节)
```

第 11 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：用“JSL1900_1912_delete_kongbaiming”数据库中的“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，那每个年代冬季的版本会被归类到下一年中。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。显然，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，这会导致后面的分析出现很大的问题。第 11 条代码可以简单理解为：

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

第 12 条代码运用了 table()函数，它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建立的表达式，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年份会否是 1900 年之后的：

```
13. 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：

```
14. JSL1900_1912_clean$qiren <- ifelse((JSL1900_1912_clean$旗
    分!="" | JSL1900_1912_clean$身份二!="" | JSL1900_1912_clean$姓
    =="",1,0)
15. table(JSL1900_1912_clean$qiren)
```

第 14 条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“满洲正黄”、“汉军正蓝”等等；“身份二”，少部分旗人有身份标识，如“宗室”、“觉罗”等等；“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”。

第 15 条代码运行之后，可在区域三返回逻辑值是“1”（旗人）和“0”（民人）的记录数。

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存储方式，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：

```
16. JSL1900_1912_clean$出身一[JSL1900_1912_clean$出身一 == "状
    元" ] <- 1
17. JSL1900_1912_clean$出身一[JSL1900_1912_clean$出身一 == "榜
    眼" ] <- 2
```

```
18. JSL1900_1912_clean$出身一[JSL1900_1912_clean$出身一 == 1 ]
    <- "狀元"
19. JSL1900_1912_clean$出身一[JSL1900_1912_clean$出身一 == 2 ]
    <- "榜眼"
```

运行第 16-17 条代码可以让缙绅录数据库中“出身一”变量为“狀元”的记录，转换数值 1，“出身一”变量为“榜眼”的记录，转换数值 2；运行后第 18-19 条代码可以使变量变回原样。将“狀元”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“狀元”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：

```
20. JSL1900_1912_clean$qiren[JSL1900_1912_clean$qiren == 1] <- "
    旗"
21. JSL1900_1912_clean$qiren[JSL1900_1912_clean$qiren == 0] <- "
    民"
22. table(JSL1900_1912_clean$qiren)
23. JSL1900_1912_clean$季节[JSL1900_1912_clean$季节 == 1 ] <- "春
    "
24. JSL1900_1912_clean$季节[JSL1900_1912_clean$季节 == 2 ] <- "
    夏"
25. JSL1900_1912_clean$季节[JSL1900_1912_clean$季节 == 3 ] <- "
    秋"
```

```
26. JSL1900_1912_clean$季节[JSL1900_1912_clean$季节 == 4 ] <- "
    冬"

27. table(JSL1900_1912_clean$季节)
```

运行第 20-21 条代码，如果“qiren”变量=1 的话，就改变其值为“旗”；如果“qiren”变量=0 的话，就改变其值为“民”。第 22 条代码利用 table()函数查看“旗”和“民”的数量。第 23-26 条代码将“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。第 27 条代码利用 table()函数查看“春”、“夏”、“秋”、“冬”的数量。

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。但当一个数据库是离散型变量，而且充满随机性的时候，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。[①]以下代码将在这个表格的基础上进行。

表 3.2 NBA 球星得分、篮板、助攻、比赛场次、场均得分表（离散数据库示例）

Player	Point	Backboard	Assistant	Game	point per game
Kobe	33643	7047	6306	1346	25.0
Jordan	32292	6672	5633	1072	30.1
James	34087	9352	9298	1258	27.1
Garnett	26071	14662	5445	1462	17.8
Duncan	26496	15091	4225	1392	19.0
Kidd	17529	8725	12091	1391	12.6
Anthony	26314	7251	3244	1114	23.6
Pippen	18940	7494	6135	1178	16.1
Curry	16419	3158	4621	699	23.5

① 数据来源：http://www.stat-nba.com/.

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$score[NBA$score >= 30000] <- 3
NBA$score[NBA$score >= 20000 & NBA$score < 30000] <- 2
NBA$score[NBA$score >= 10000 & NBA$score < 20000] <- 1
NBA$score <- factor(NBA$score, 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，如果要将将字符型变量转换回数值型变量，代码如下：

```
NBA$score_origin <- factor(NBA$score, 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$assistant[NBA$assistant <= 5000] <- 1
NBA$assistant[NBA$assistant > 5000 & NBA$assistant <=
    9000] <- 2
```

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

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，用得不多，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：

```
37. JSL1900_1912_clean$xinming <-paste(JSL1900_1912_clean$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：

```
38. 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：

```
39. 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：

```
40. 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：

```
41. JSL1900_1912_clean$zhixian <-
      ifelse( str_detect(JSL1900_1912_clean$官职一,"知縣"),1,0)
42. JSL1900_1912_clean$houbu <-
```

```
    ifelse( str_detect(JSL1900_1912_clean$官职一,"候補"),1,0)
43. JSL1900_1912_clean$ewai <-
    ifelse( str_detect(JSL1900_1912_clean$官职一,"額外"),1,0)
```

运行第 41-43 条代码后，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：

```
44. 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4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）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，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；如果等于“ifany”或者“always”则为统计。

```
45. table(JSL1900_1912_clean$阳历年份)

46. table(JSL1900_1912_clean$籍贯省, JSL1900_1912_clean$qiren)

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

第 45-47 条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”可归类为一个地方；“甘肃”和“甘肅”两者是繁体和简体的区别，应当归类为同一个地方。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“”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“”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：

```
48. table(JSL1900_1912_clean$籍贯省)
49. JSL1900_1912_clean$jiguansheng_sort <-
    factor(JSL1900_1912_clean$籍贯省, levels = c("","? ？ ","? 西","
    奥國","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","甘肃","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"))
```

第 48-59 条代码的思路是首先用 table（）函数查看“籍贯省”这一变量的类型，再生成一个新的变量 jiguansheng_sort，利用 factor()函数对不同类型进行整理。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（数据库连接），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：

```
50. table(JSL1900_1912_clean$jiguansheng_sort,
    JSL1900_1912_clean$qiren, exclude= NULL, useNA= "always")
```

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：

```
51. table(JSL1900_1912_clean$出身一, JSL1900_1912_clean$qiren)
52. 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","附贡","附貢生","副贡","副貢","副貢生","
    廩贡","廩貢","廩貢生","崴貢","歲貢","歲貢生","優貢","優貢生","增贡
    ","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","异途贡生","异途
贡生","异途贡生"))
53. table(JSL1900_1912_clean$gongsheng)
```

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：

```
54. 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：

```
55. table(JSL1900_1912_clean$dingjia, JSL1900_1912_clean$qiren,
    exclude= NULL, useNA= "always")
56. table(JSL1900_1912_clean$gongsheng,
    JSL1900_1912_clean$qiren, exclude= NULL, useNA= "always")
57. table(JSL1900_1912_clean$jiguansheng_sort,
    JSL1900_1912_clean$gongsheng, exclude= NULL, useNA=
    "always")
```

第 55 条代码运行之后可以展示“鼎甲”群体中的满汉比例；第 56 条代码运行之后可以展示出身为正途贡生、异途贡生官员分别的满汉比例；第 57 条代码可以展示出身为正途贡生和异途贡生官员的地理来源。总而言之，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

在撰写论文的过程中，我们常常需要列出比较专业、简洁的表格，遵守学术规范。前面介绍的 table（）函数会在区域三返回观测值，需要复制粘贴，并重新制作表格样式，比较麻烦。而 table1（）这个函数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（）函数的格式为：

```
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。需要添加“”。“”内可以输入任意字符，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“”。

现在我们尝试利用该函数制作可以导出的表格，实例代码如下：

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

59. library(table1)

60. help(table1)

61. label(JSL1900_1912_clean$jiguansheng_sort) <-"籍贯省"

62. label(JSL1900_1912_clean$qiren) <- "旗人"

63. label(JSL1900_1912_clean$gongsheng) <- "贡生"
```

```
64. label(JSL1900_1912_clean$zhixian) <- "知县"
65. table1(~jiguansheng_sort|qiren, data = JSL1900_1912_clean,
    overall = "total")
66. table1(~jiguansheng_sort+gongsheng|qiren, data =
    JSL1900_1912_clean, overall = "total")
67. JSL1900_1912_clean$zhixian[JSL1900_1912_clean$zhixian == 1]
    <- "知县"
68. JSL1900_1912_clean$zhixian[JSL1900_1912_clean$zhixian == 0]
    <- "非知县"
69. table1(~jiguansheng_sort+gongsheng+dingjia|zhixian*qiren,
    data = JSL1900_1912_clean, overall = "Overall")
```

第 58-59 条代码是安装并运行 table1（）所在的包。

第 60 条代码是启动 table1（）函数的帮助。

第 61-64 条代码是为籍贯省、旗人、贡生、知县四个变量添加标签，以便在表格中显示，label（）函数的格式为 label（data$x）<- “y”，x 是数据集中需要贴标签的变量，y 是标签，可以是字符，也可以是数值。

第 65 条代码制作可以导出的表格，前面的“~”符号不可省略。“|”用来分层，即“籍贯省”为行，“旗人”为列，两者可以颠倒顺序。

	民 (N=484559)	旗 (N=154873)	total (N=639432)
籍贯省			
	122350 (25.2%)	153810 (99.3%)	276160 (43.2%)
不详	23 (0.0%)	2 (0.0%)	25 (0.0%)
外国	4 (0.0%)	689 (0.4%)	693 (0.1%)
安徽	26265 (5.4%)	52 (0.0%)	26317 (4.1%)
奉天	4046 (0.8%)	0 (0%)	4046 (0.6%)
福建	12352 (2.5%)	11 (0.0%)	12363 (1.9%)
甘肃	3306 (0.7%)	10 (0.0%)	3316 (0.5%)
广西	6840 (1.4%)	5 (0.0%)	6845 (1.1%)
广东	15887 (3.3%)	33 (0.0%)	15920 (2.5%)
贵州	8013 (1.7%)	1 (0.0%)	8014 (1.3%)
河南	15314 (3.2%)	32 (0.0%)	15346 (2.4%)
黑龙江	22 (0.0%)	0 (0%)	22 (0.0%)
湖北	16948 (3.5%)	29 (0.0%)	16977 (2.7%)
湖南	25417 (5.2%)	5 (0.0%)	25422 (4.0%)
吉林	302 (0.1%)	0 (0%)	302 (0.0%)
江南	104 (0.0%)	0 (0%)	104 (0.0%)
江苏	39647 (8.2%)	13 (0.0%)	39660 (6.2%)
江西	18299 (3.8%)	9 (0.0%)	18308 (2.9%)
山东	19818 (4.1%)	3 (0.0%)	19821 (3.1%)
山西	7728 (1.6%)	3 (0.0%)	7731 (1.2%)
陕西	7564 (1.6%)	17 (0.0%)	7581 (1.2%)
顺天	31877 (6.6%)	59 (0.0%)	31936 (5.0%)
四川	21938 (4.5%)	45 (0.0%)	21983 (3.4%)
天津	5 (0.0%)	0 (0%)	5 (0.0%)
新疆	4 (0.0%)	0 (0%)	4 (0.0%)
云南	6541 (1.3%)	16 (0.0%)	6557 (1.0%)
浙江	51726 (10.7%)	21 (0.0%)	51747 (8.1%)
直隶	22219 (4.6%)	8 (0.0%)	22227 (3.5%)

图 4.1 缙绅录数据库中官员的籍贯和满汉比例

第 66 条代码增加了一个“+”符号和一个变量“贡生”。这表明“籍贯省”和“贡生”同时为行，“旗人”为列。

	民 (N=484559)	旗 (N=154873)	total (N=639432)
籍贯省			
	122350 (25.2%)	153810 (99.3%)	276160 (43.2%)
不详	23 (0.0%)	2 (0.0%)	25 (0.0%)
外国	4 (0.0%)	689 (0.4%)	693 (0.1%)
安徽	26265 (5.4%)	52 (0.0%)	26317 (4.1%)
奉天	4046 (0.8%)	0 (0%)	4046 (0.6%)
福建	12352 (2.5%)	11 (0.0%)	12363 (1.9%)
甘肃	3306 (0.7%)	10 (0.0%)	3316 (0.5%)
广西	6840 (1.4%)	5 (0.0%)	6845 (1.1%)
广东	15887 (3.3%)	33 (0.0%)	15920 (2.5%)
贵州	8013 (1.7%)	1 (0.0%)	8014 (1.3%)
河南	15314 (3.2%)	32 (0.0%)	15346 (2.4%)
黑龙江	22 (0.0%)	0 (0%)	22 (0.0%)
湖北	16948 (3.5%)	29 (0.0%)	16977 (2.7%)
湖南	25417 (5.2%)	5 (0.0%)	25422 (4.0%)
吉林	302 (0.1%)	0 (0%)	302 (0.0%)
江南	104 (0.0%)	0 (0%)	104 (0.0%)
江苏	39647 (8.2%)	13 (0.0%)	39660 (6.2%)
江西	18299 (3.8%)	9 (0.0%)	18308 (2.9%)
山东	19818 (4.1%)	3 (0.0%)	19821 (3.1%)
山西	7728 (1.6%)	3 (0.0%)	7731 (1.2%)
陕西	7564 (1.6%)	17 (0.0%)	7581 (1.2%)
顺天	31877 (6.6%)	59 (0.0%)	31936 (5.0%)
四川	21938 (4.5%)	45 (0.0%)	21983 (3.4%)
天津	5 (0.0%)	0 (0%)	5 (0.0%)
新疆	4 (0.0%)	0 (0%)	4 (0.0%)
云南	6541 (1.3%)	16 (0.0%)	6557 (1.0%)
浙江	51726 (10.7%)	21 (0.0%)	51747 (8.1%)
直隶	22219 (4.6%)	8 (0.0%)	22227 (3.5%)
贡生			
正途贡生	35444 (7.3%)	956 (0.6%)	36400 (5.7%)
异途贡生	33602 (6.9%)	326 (0.2%)	33928 (5.3%)
Missing	415513 (85.8%)	153591 (99.2%)	569104 (89.0%)

图 4.2 所有官员以及贡生群体的满汉比例和籍贯

第 67-68 条代码运用到了数值型变量转换为字符型的代码，这在前文第三章第二节讲过。运行这两条代码的目的是后面需要用到“知县”这个变量来制作表格，提前转换为字符型便于理解。

第 69 条代码增加了“*”符号和两个变量“贡生”、“鼎甲”。“籍贯省”、“贡生”和“鼎甲”为行，“旗人”和“知县”将为列，且两者是嵌套关系，“知县”嵌套在“旗人”之上。

	非知县		知县		Overall	
	民 (N=433153)	旗 (N=151486)	民 (N=51406)	旗 (N=3387)	民 (N=484559)	旗 (N=154873)
籍贯省						
	122125 (28.2%)	150437 (99.3%)	225 (0.4%)	3373 (99.6%)	122350 (25.2%)	153810 (99.3%)
不详	21 (0.0%)	2 (0.0%)	2 (0.0%)	0 (0%)	23 (0.0%)	2 (0.0%)
外国	4 (0.0%)	689 (0.5%)	0 (0%)	0 (0%)	4 (0.0%)	689 (0.4%)
安徽	22396 (5.2%)	52 (0.0%)	3869 (7.5%)	0 (0%)	26265 (5.4%)	52 (0.0%)
奉天	3478 (0.8%)	0 (0%)	568 (1.1%)	0 (0%)	4046 (0.8%)	0 (0%)
福建	9767 (2.3%)	11 (0.0%)	2585 (5.0%)	0 (0%)	12352 (2.5%)	11 (0.0%)
甘肃	2364 (0.5%)	7 (0.0%)	942 (1.8%)	3 (0.1%)	3306 (0.7%)	10 (0.0%)
广西	4901 (1.1%)	5 (0.0%)	1939 (3.8%)	0 (0%)	6840 (1.4%)	5 (0.0%)
广东	14444 (3.3%)	33 (0.0%)	1443 (2.8%)	0 (0%)	15887 (3.3%)	33 (0.0%)
贵州	6104 (1.4%)	1 (0.0%)	1909 (3.7%)	0 (0%)	8013 (1.7%)	1 (0.0%)
河南	12718 (2.9%)	31 (0.0%)	2596 (5.1%)	1 (0.0%)	15314 (3.2%)	32 (0.0%)
黑龙江	22 (0.0%)	0 (0%)	0 (0%)	0 (0%)	22 (0.0%)	0 (0%)
湖北	13835 (3.2%)	28 (0.0%)	3113 (6.1%)	1 (0.0%)	16948 (3.5%)	29 (0.0%)
湖南	21828 (5.0%)	5 (0.0%)	3589 (7.0%)	0 (0%)	25417 (5.2%)	5 (0.0%)
吉林	254 (0.1%)	0 (0%)	48 (0.1%)	0 (0%)	302 (0.1%)	0 (0%)
江南	103 (0.0%)	0 (0%)	1 (0.0%)	0 (0%)	104 (0.0%)	0 (0%)
江苏	34688 (8.0%)	13 (0.0%)	4959 (9.6%)	0 (0%)	39647 (8.2%)	13 (0.0%)
江西	15161 (3.5%)	9 (0.0%)	3138 (6.1%)	0 (0%)	18299 (3.8%)	9 (0.0%)
山东	17317 (4.0%)	3 (0.0%)	2501 (4.9%)	0 (0%)	19818 (4.1%)	3 (0.0%)
山西	6455 (1.5%)	1 (0.0%)	1273 (2.5%)	2 (0.1%)	7728 (1.6%)	3 (0.0%)
陕西	6172 (1.4%)	17 (0.0%)	1392 (2.7%)	0 (0%)	7564 (1.6%)	17 (0.0%)
顺天	29746 (6.9%)	57 (0.0%)	2131 (4.1%)	2 (0.1%)	31877 (6.6%)	59 (0.0%)
四川	18691 (4.3%)	44 (0.0%)	3247 (6.3%)	1 (0.0%)	21938 (4.5%)	45 (0.0%)
天津	5 (0.0%)	0 (0%)	0 (0%)	0 (0%)	5 (0.0%)	0 (0%)
新疆	4 (0.0%)	0 (0%)	0 (0%)	0 (0%)	4 (0.0%)	0 (0%)
云南	5008 (1.2%)	16 (0.0%)	1533 (3.0%)	0 (0%)	6541 (1.3%)	16 (0.0%)
浙江	45808 (10.6%)	17 (0.0%)	5918 (11.5%)	4 (0.1%)	51726 (10.7%)	21 (0.0%)
直隶	19734 (4.6%)	8 (0.0%)	2485 (4.8%)	0 (0%)	22219 (4.6%)	8 (0.0%)
贡生						
正途贡生	32747 (7.6%)	869 (0.6%)	2697 (5.2%)	87 (2.6%)	35444 (7.3%)	956 (0.6%)
异途贡生	31805 (7.3%)	301 (0.2%)	1797 (3.5%)	25 (0.7%)	33602 (6.9%)	326 (0.2%)
Missing	368601 (85.1%)	150316 (99.2%)	46912 (91.3%)	3275 (96.7%)	415513 (85.8%)	153591 (99.2%)
dingjia						
鼎甲	786 (0.2%)	76 (0.1%)	0 (0%)	0 (0%)	786 (0.2%)	76 (0.0%)
Missing	432367 (99.8%)	151410 (99.9%)	51406 (100%)	3387 (100%)	483773 (99.8%)	154797 (100.0%)

图 4.3 出身为贡生、“鼎甲”的知县的满汉比例和地理来源

观察三条代码的符号，有相同之处，也有不同之处。“~”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。

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，我们只需要缙绅录 1910 年的数据，来分析 1910 年出身为贡生的官员群体之满汉比例和地理来源，示例代码如下：

```
70. JSL1900_1912_clean$贡生<-"0"
71. JSL1900_1912_clean$贡生[JSL1900_1912_clean$gongsheng == "
    正途贡生"] <- "正途贡生"
72. JSL1900_1912_clean$贡生[JSL1900_1912_clean$gongsheng == "
    异途贡生"] <- "异途贡生"
73. JSL1900_1912_clean$贡生[JSL1900_1912_clean$贡生  == "0"] <- "
    非贡生"
74. JSL1900_1912_only1910 <-subset(JSL1900_1912_clean,阳历年份
    numeric == 1910)
75. table1(~jiguansheng_sort|qiren*贡生, data =
    JSL1900_1912_only1910, overall = "total",rowlabelhead = "")
```

第 70 条代码运行后，会创建一个新的变量“贡生”（这里我们用中文字符表示，因为之前我们已经创建了一个拼音的“gongsheng”），令其所有观测值为“0”。

第 71 条代码运行后，“gongsheng”中观测值为“正途贡生”的记录，在新的“贡生”变量中由“0”换成“正途贡生”。

第 72 条代码同理，只是换为了“异途贡生”。

第 73 条代码则是将“贡生”中剩下为“0”的观测值转变为“非贡生”。

那我们为什么要创建一个新的“贡生”变量呢？这是因为 table1（）函数中的表格列变量，不允许有缺失值的存在，而原始的“gongsheng”变量除了“正途贡生”和“异途贡生”外，其余都是缺失值，直接运行 RStudio 会报错。因此，我们在进行制表之前，必须保证表格的列变量没有缺失值。

第 74 条代码运用 subset（）函数创建了一个只有 1910 年记录的数据集。

第 75 条代码在只有 1910 年记录的数据集上制表，已观测 1910 年贡生群体的满汉比例和地理来源，效果如下：

	民			旗			total		
	非贡生 (N=38701)	异途贡生 (N=2732)	正途贡生 (N=3242)	非贡生 (N=14854)	异途贡生 (N=26)	正途贡生 (N=129)	非贡生 (N=53555)	异途贡生 (N=2758)	正途贡生 (N=3371)
籍贯省									
	5702 (14.7%)	1835 (67.2%)	1551 (47.8%)	14833 (99.9%)	26 (100%)	129 (100%)	20535 (38.3%)	1861 (67.5%)	1680 (49.8%)
不详	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
外国	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
安徽	2397 (6.2%)	57 (2.1%)	104 (3.2%)	4 (0.0%)	0 (0%)	0 (0%)	2401 (4.5%)	57 (2.1%)	104 (3.1%)
奉天	374 (1.0%)	11 (0.4%)	37 (1.1%)	0 (0%)	0 (0%)	0 (0%)	374 (0.7%)	11 (0.4%)	37 (1.1%)
福建	1180 (3.0%)	28 (1.0%)	83 (2.6%)	1 (0.0%)	0 (0%)	0 (0%)	1181 (2.2%)	28 (1.0%)	83 (2.5%)
甘肃	307 (0.8%)	12 (0.4%)	26 (0.8%)	0 (0%)	0 (0%)	0 (0%)	307 (0.6%)	12 (0.4%)	26 (0.8%)
广西	629 (1.6%)	13 (0.5%)	50 (1.5%)	0 (0%)	0 (0%)	0 (0%)	629 (1.2%)	13 (0.5%)	50 (1.5%)
广东	1447 (3.7%)	38 (1.4%)	51 (1.6%)	3 (0.0%)	0 (0%)	0 (0%)	1450 (2.7%)	38 (1.4%)	51 (1.5%)
贵州	623 (1.6%)	28 (1.0%)	73 (2.3%)	0 (0%)	0 (0%)	0 (0%)	623 (1.2%)	28 (1.0%)	73 (2.2%)
河南	1285 (3.3%)	36 (1.3%)	73 (2.3%)	0 (0%)	0 (0%)	0 (0%)	1285 (2.4%)	36 (1.3%)	73 (2.2%)

河南	1285 (3.3%)	36 (1.3%)	73 (2.3%)	0 (0%)	0 (0%)	0 (0%)	1285 (2.4%)	36 (1.3%)	73 (2.2%)
黑龙江	0 (0%)	0 (0%)	5 (0.2%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	5 (0.1%)
湖北	1638 (4.2%)	69 (2.5%)	92 (2.8%)	1 (0.0%)	0 (0%)	0 (0%)	1639 (3.1%)	69 (2.5%)	92 (2.7%)
湖南	2372 (6.1%)	68 (2.5%)	78 (2.4%)	0 (0%)	0 (0%)	0 (0%)	2372 (4.4%)	68 (2.5%)	78 (2.3%)
吉林	53 (0.1%)	8 (0.3%)	5 (0.2%)	0 (0%)	0 (0%)	0 (0%)	53 (0.1%)	8 (0.3%)	5 (0.1%)
江南	7 (0.0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	7 (0.0%)	0 (0%)	0 (0%)
江苏	3831 (9.9%)	85 (3.1%)	179 (5.5%)	0 (0%)	0 (0%)	0 (0%)	3831 (7.2%)	85 (3.1%)	179 (5.3%)
江西	1539 (4.0%)	71 (2.6%)	105 (3.2%)	0 (0%)	0 (0%)	0 (0%)	1539 (2.9%)	71 (2.6%)	105 (3.1%)
山东	1704 (4.4%)	78 (2.9%)	120 (3.7%)	0 (0%)	0 (0%)	0 (0%)	1704 (3.2%)	78 (2.8%)	120 (3.6%)
山西	656 (1.7%)	36 (1.3%)	49 (1.5%)	0 (0%)	0 (0%)	0 (0%)	656 (1.2%)	36 (1.3%)	49 (1.5%)
陕西	573 (1.5%)	20 (0.7%)	90 (2.8%)	4 (0.0%)	0 (0%)	0 (0%)	577 (1.1%)	20 (0.7%)	90 (2.7%)
顺天	2827 (7.3%)	31 (1.1%)	60 (1.9%)	4 (0.0%)	0 (0%)	0 (0%)	2831 (5.3%)	31 (1.1%)	60 (1.8%)
四川	1884 (4.9%)	46 (1.7%)	116 (3.6%)	0 (0%)	0 (0%)	0 (0%)	1884 (3.5%)	46 (1.7%)	116 (3.4%)
天津	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
新疆	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
云南	539 (1.4%)	12 (0.4%)	50 (1.5%)	4 (0.0%)	0 (0%)	0 (0%)	543 (1.0%)	12 (0.4%)	50 (1.5%)
浙江	4850 (12.5%)	69 (2.5%)	138 (4.3%)	0 (0%)	0 (0%)	0 (0%)	4850 (9.1%)	69 (2.5%)	138 (4.1%)
直隶	2284 (5.9%)	81 (3.0%)	107 (3.3%)	0 (0%)	0 (0%)	0 (0%)	2284 (4.3%)	81 (2.9%)	107 (3.2%)

图 4.4 贡生群体的满汉比例与地理来源

现在，尝试研究在京师任职且出身为“贡生”的官员之满汉比例，代码如下：

```
JSL1900_1912_only 京师 <-
    subset(JSL1900_1912_clean,jiguansheng_sort == "京师")
table1(~qiren|贡生, data = JSL1900_1912_only 京师, overall =
    "total",rowlabelhead = "")
```

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京师。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：

```
JSL1900_1912_京师 1910 <-
    subset(JSL1900_1912_clean,jiguansheng_sort == "京师"& 阳历年
    份 numeric == 1910)
table1(~籍贯省|qiren*贡生, data = JSL1900_1912_京师 1910,
    overall = "total",rowlabelhead = "")
JSL1900_1912_京师 1910 候补 <-
    subset(JSL1900_1912_clean,jiguansheng_sort == "京师" & 阳历
```

```
    年份 numeric == 1910 & houbu == 1)
81. table1(~籍贯省|qiren*贡生, data = JSL1900_1912_京师 1910 候补,
    overall = "total",rowlabelhead = "")
```

第 78-79 条代码的思路是：首先创建一个只有 1910 年京师地区记录的数据集，再在这个数据集的基础上研究出身为“贡生”官员群体的满汉比例和地理来源。

第 80-81 条代码的思路是：首先创建一个只有 1910 年京师地区现任候补官员记录的数据集，再在这个数据集的基础上研究出身为“贡生”官员群体的满汉比例和地理来源。

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，示例如下

```
82. JSL1900_1912_西南三省 after1908 <- subset(JSL1900_1912_clean,
    (jiguansheng_sort == "四川" | jiguansheng_sort == "贵州" |
    jiguansheng_sort == "云南" ) & (阳历年份 numeric > 1908 ) )
83. table1(~籍贯省|贡生*qiren, data = JSL1900_1912_西南三省
    after1908, overall = "total",rowlabelhead = "")
```

第 82 条代码的思路是：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“|”符号（或者）进行连接。

第 83 条代码的思路是，在新数据集上研究 1908 年之后在西南三省任职且出身为“贡生”的官员之满汉比例和地理来源。

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 help(“table1”)查看帮助或者学习新的参数。

第5章　直方图、散点图、折线图

5.1 直方图

5.1.1 简单直方图

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：

```
84. install.packages("ggplot2")

85. library(ggplot2)

86. 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 ggplot（）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：

```
g <- ggplot(data, mapping = aes(x, y, fill, color,…),…)

g + geom_bar(mapping,data,stat,position,just,width,,…)
```

首先是 ggplot（）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：

```
g1 <- ggplot(JSL1900_1912_only1910, aes(x =
    jiguansheng_sort))
g1+geom_bar(color="red",fill = "white",just= 0.5, width = 0.8)
```

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。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。效果如下：

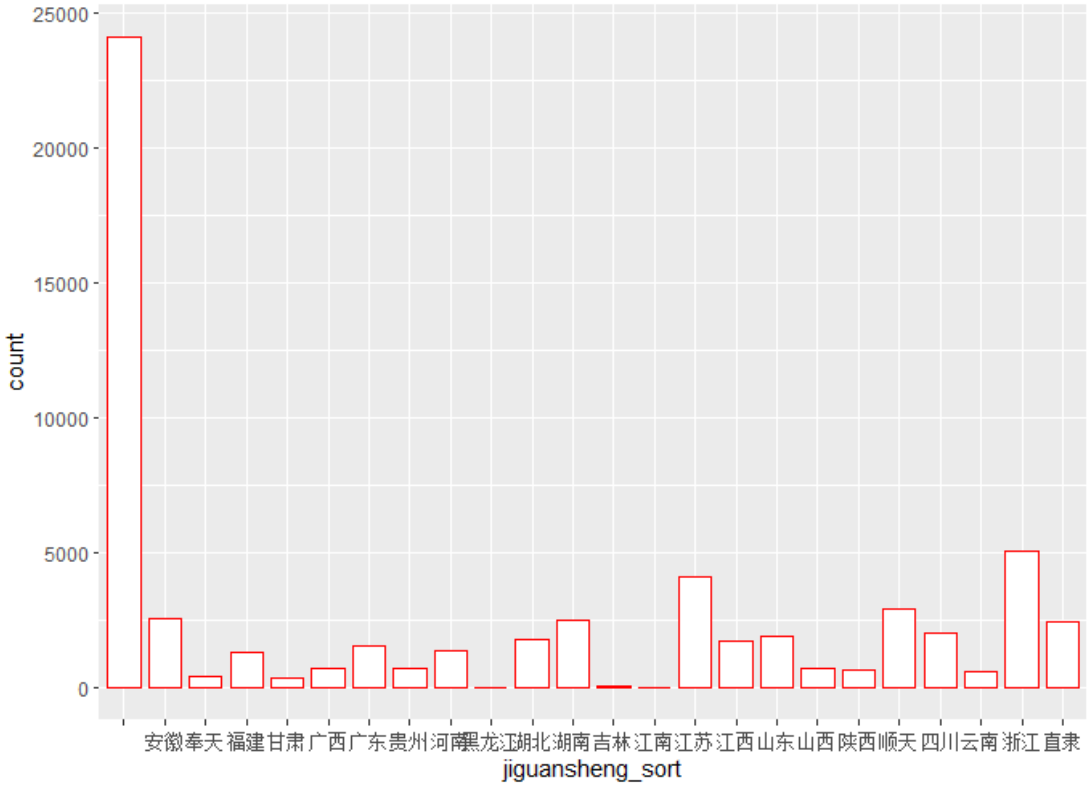


图 5.1 简单直方图制图效果

可以看出，这只是一个初步的图形，存在许多问题：首先，横轴字符太过拥挤，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，条形最高的一项是“”，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，这是缙绅录的体例决定的），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，我们先建立一个数据集：

```
89. JSL1900_1912_bingbu <- subset(JSL1900_1912_clean,机构一 == "
    兵部衙門")
```

这里，我们建立的数据集只记载了在兵部衙门任职的官员，我们以此为基础研究兵部衙门的满汉比例，示例如下：

```
90. g2 <- ggplot(JSL1900_1912_bingbu, aes(x = 阳历年份 numeric,
    fill = qiren))
91. g2+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
    ")+guides(fill=guide_legend(title = "旗人
    "))+scale_fill_manual(values = c("cyan","pink"))+
    scale_x_continuous(expand = c(0.1,0.1), breaks =
    seq(1900,1906,1))+ scale_y_continuous(limits = c(0,1000),
    breaks = seq(0,1000,100))+ geom_text(aes(label = ..count..,
    family = "serif"),stat = "count",size = 3.5, vjust = -0.8, hjust =
    0.5, color = "red2",position = position_dodge(0.5))
```

整体来看，这两条代码量比较大，涉及的参数也比较多，我们先来看看运行效果，结合效果进行分析：

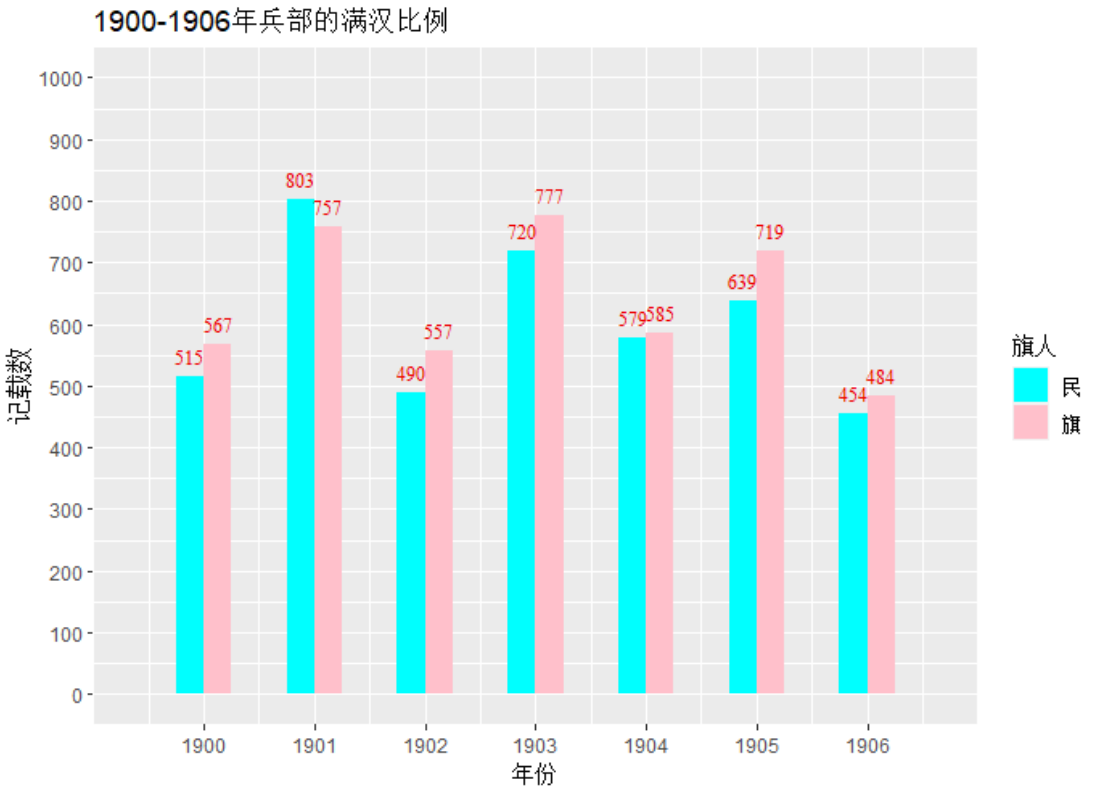


图 5.2 兵部官员满汉比例代码运行效果

第 90 条代码和第 91 条代码的第一行是简单直方图的运行代码。其中，geom_bar()的 position 参数，position 参数一共有 3 种形式：其一是"stack"，堆积；其一是"fill",堆积，且每个条形的最大值为 1；其一是"dodge",分条展示。这里我们选择的是“dodge”,图形效果呈现的是兵部满汉官员数量分条展示。

第 91 条代码的“label”参数，可设置横轴、纵轴、整个图形的标题。

“guides”参数可设置图例的标题。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”参数可设置条形的颜色，图中，民人和旗人颜色分别为青色和粉红色。

“scale_x_continuous”和“scale_y_continuous”参数可调整 X 轴与 y 轴的最大最小值、间隔等等。

“geom_text”参数可设置直方图数值标签的统计方式、大小、间隔、位置等。

以上两条代码是直方图的初步图形，我们还可以在整个代码中加入一个修饰函数 theme（），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，这一参数可以设置的细节众多，比较复杂，它实际上也是一个函数，格式为：

```
theme(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(size = 2, color = “purple”),
      panel.background = element_rect(color = “grey50”),
      panel.grid =element_line(color=“grey50”, linewidth = 0.1))
```

我们尝试在原始代码的后面加入 theme 函数，具体的添加方法是在 theme（）函数之前加上“+”号，将整个 theme（）函数添加到代码中，具体的运行效果如下：

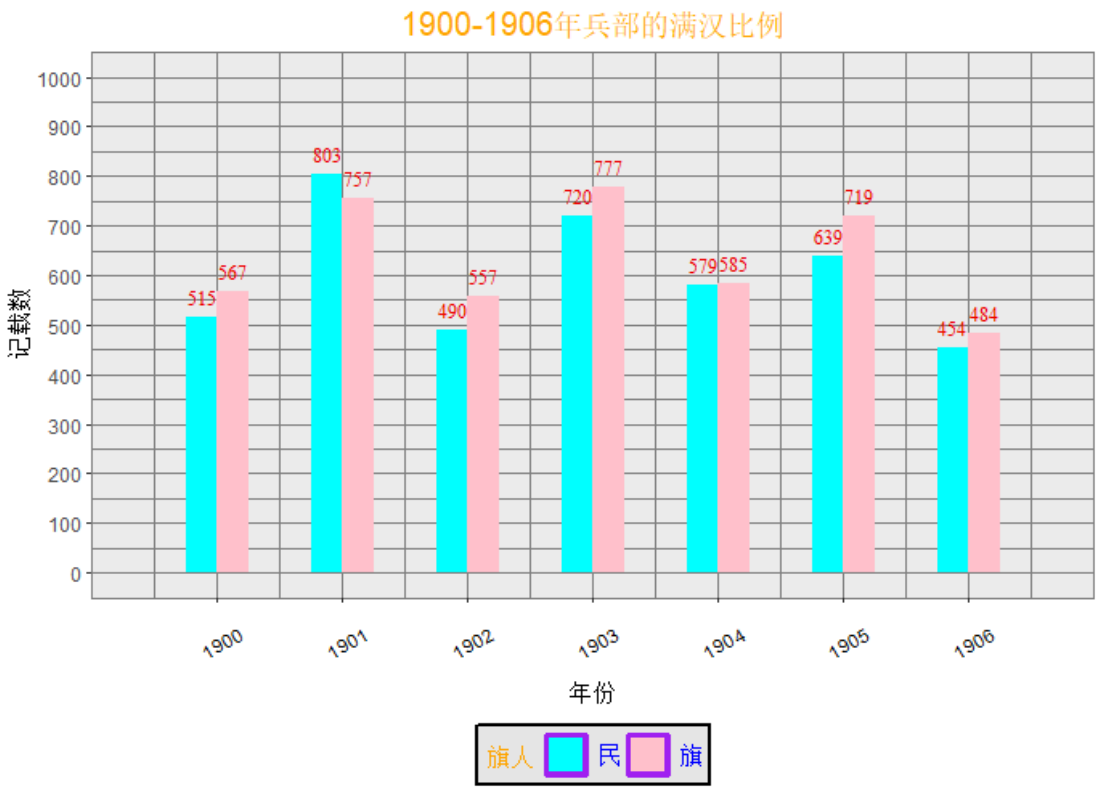


图 5.3 兵部官员满汉比例（加入修饰函数 theme（）后）

可通过对比图 5-2 和图 5-3 可以看出，虽然图 5-3 的整体效果和图 5-2 差不多，但是图 5-3 整体看起来比较美观，比如图标题居中、图例下移，网格线设置成黑色使其更加清晰等等。为了让使用者更加方便地理解函数，我们将标题的颜色、图例的颜色更改为了一些特定的颜色。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，可以在文末附录“代码合集”或者链接中查看具体的设置格式，进行调试。下面这张图详细讲解 theme（）函数中各个参数的作用：

```
theme(plot.title = element_text(size = 15, hjust = 0.5, color = "orange"),
                                                    ------标题大小、位置、颜色
      axis.text.x = element_text(size = 9, angle = 30, color = "black",
                                 margin = margin(t = .5, unit = "cm")),
                                   -------X轴标签的大小、角度、颜色、标签与轴的间距
      axis.text.y = element_text(size = 9),                          ------y轴大小
      axis.title.x = element_text(size = 10, hjust = 0.5, vjust = 0),
                                          ------x轴标题的大小、与底部水平或垂直间距
      axis.title.y = element_text(size = 10),                        ------y轴标题大小
      legend.position = "bottom",                                    ------图例位置（底部）
      legend.title = element_text(size = 12, hjust = 0.5, color = "orange"),
                                                    -------图例标题大小、位置、颜色
      legend.background = element_rect(color = "black", fill = "grey90", linewidth = 1),
                                   -------图例背景的边框颜色、填充颜色、线条宽度
      legend.text = element_text(size = 10, color = "blue"), -------图例文本的大小、颜色
      legend.key = element_rect(size = 2, color = "purple"),-------图例节点的大小、颜色
      panel.background = element_rect(color = "grey50"),            -------大背景的颜色
      panel.grid = element_line(color="grey50", linewidth = 0.1))---网格的颜色和线条大小
```

图 5.4 图例修饰函数 theme（）中各参数的用法

theme（）函数有非常多参数，这里提到的只是几个比较常用的，如果有更多需求，可以在 R 中 help（theme）查看可用的参数，或者登录 ggplot 的官网查询。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。以下图片，比较形象的展示了 theme（）函数中常用参数的用法，可供参考：①

① 图片来源 https:://blog.csdn.net/zty0104/article/details/119646934。

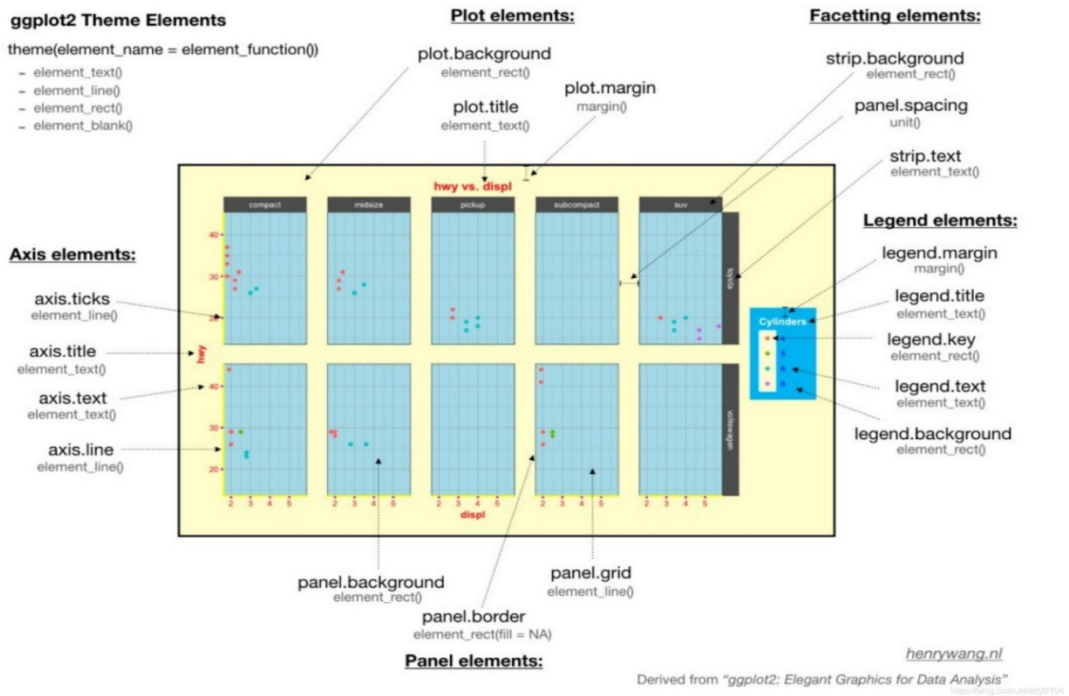


图 5.5 图里修饰函数 theme（）参数用法图解

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：①

```
92. g2 <- ggplot(JSL1900_1912_bingbu, aes(x = 阳历年份, fill =
    qiren))
93. g2+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"))+
    scale_y_continuous(labels = scales::percent_format(accuracy=
    1))+ geom_text(aes(label=paste0(sprintf("%1.1f", ..count.. /
```

① 第 92 条代码中的“sprintf("%1.1f", ..count.. / tapply(..count.., ..x.., sum)[as.character(..x..)]*100),”部分引用自：https://stackoom.com/question/43QCB。

```
tapply(..count.., ..x.., sum) [as.character(..x..)]*100), "%"),
family = "serif"),stat = "count",size = 5, vjust = -0.8, color =
"gray10",position = position_fill(0.5))
```

下面是这两条代码的运行效果：

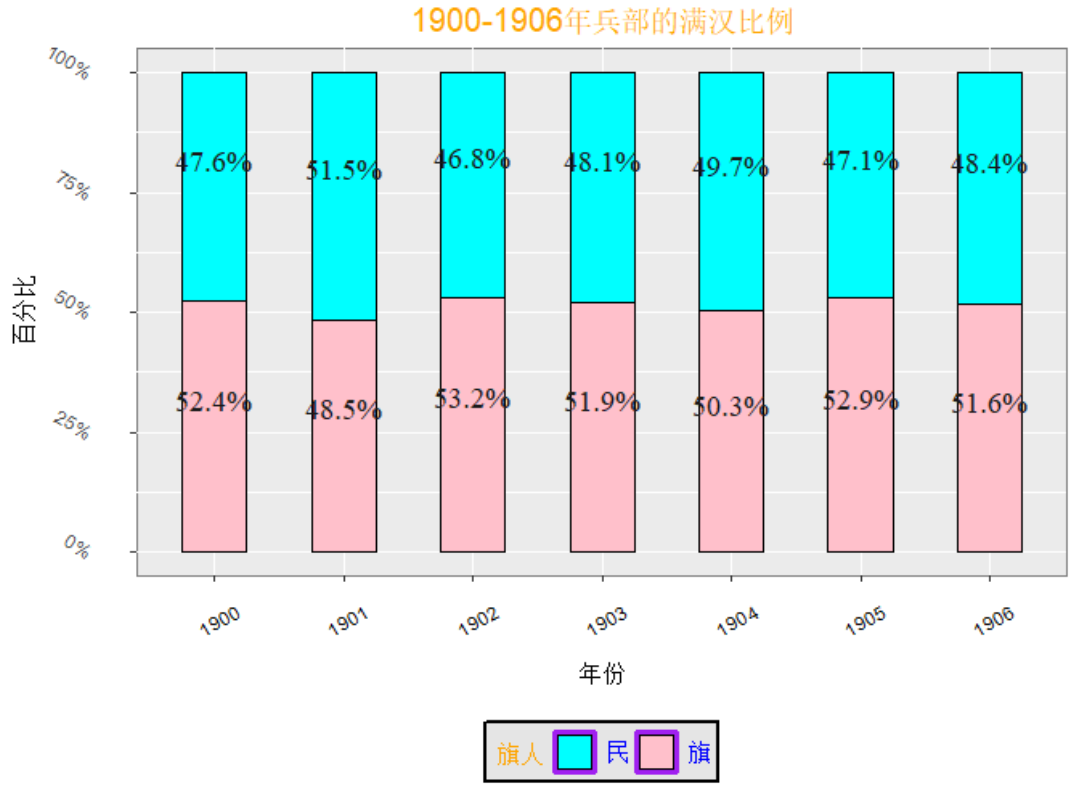


图 5.6 兵部满汉比例（堆积直方图）

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：

```
94. N <- ggplot(NBA,aes(...1,score_per_game))
95. N+geom_bar(color="black", fill="white", stat= "identity",just=
    0.5,width = 1)+ xlab("players")+ ylab("ppg")+
    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=score_per_game), vjust=-0.8, hjust=0.5,
    color="red")
```

运行效果如下：

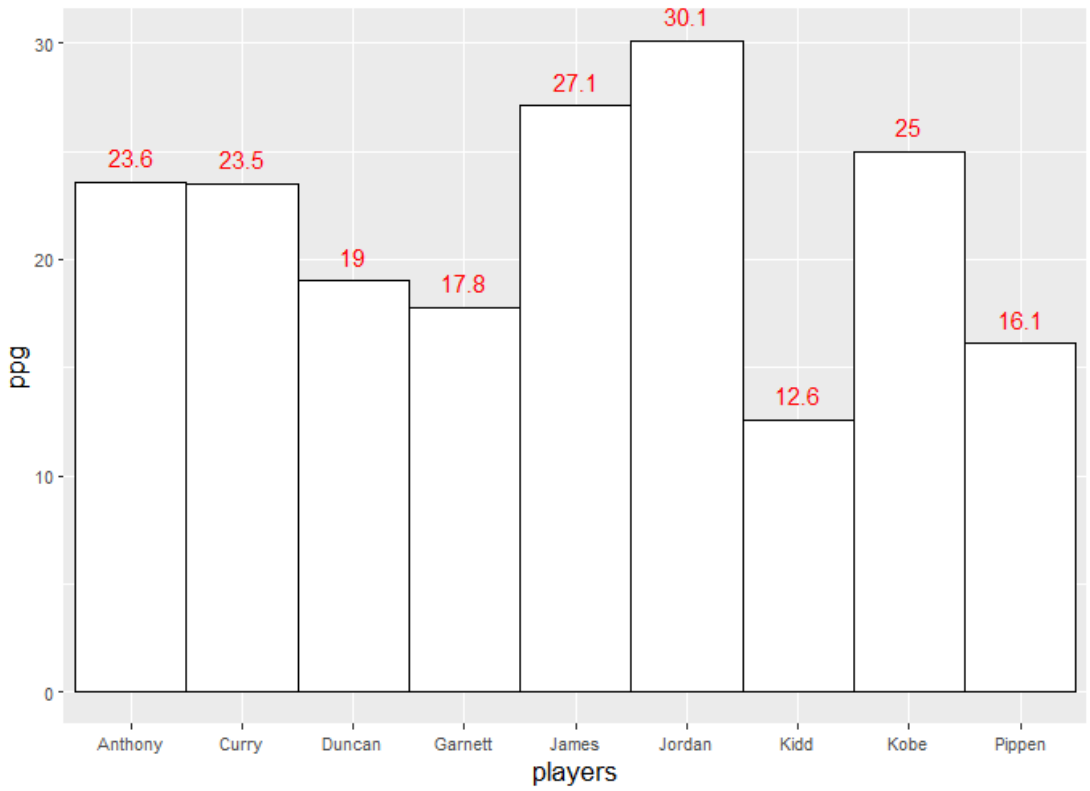


图 5.7 “NBA”数据集相关代码运行效果

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，代码如下：

```
96. JSL1900_1912_bingbu$官职品级 <- ""

97. 尚书 <- str_detect(JSL1900_1912_bingbu$官职一,"尚書")

98. JSL1900_1912_bingbu$官职品级[尚书 == TRUE] <- "尚书"

99. 左侍郎 <- str_detect(JSL1900_1912_bingbu$官职一,"左侍郎")

100. JSL1900_1912_bingbu$官职品级[左侍郎 == TRUE] <- "左侍郎"

101. 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))
```

利用 96-111 条代码，我们创建了一个数据集。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。这 16 条代码的思路是：首先创造一个空变量“官职品级”；然后利用 str_detect（判定函数，第三章第四节）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），并以此类推，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。创建数据集后，便可制作散点图：

```
g3<- ggplot(data=subset(JSL1900_1912_bingbu, !is.na(品级)),
```

```
    aes(x =  阳历年份,品级,shape = qiren,colour = qiren))
113.  g3+geom_point(position = "jitter")+ 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"))
```

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。

运行效果如下：

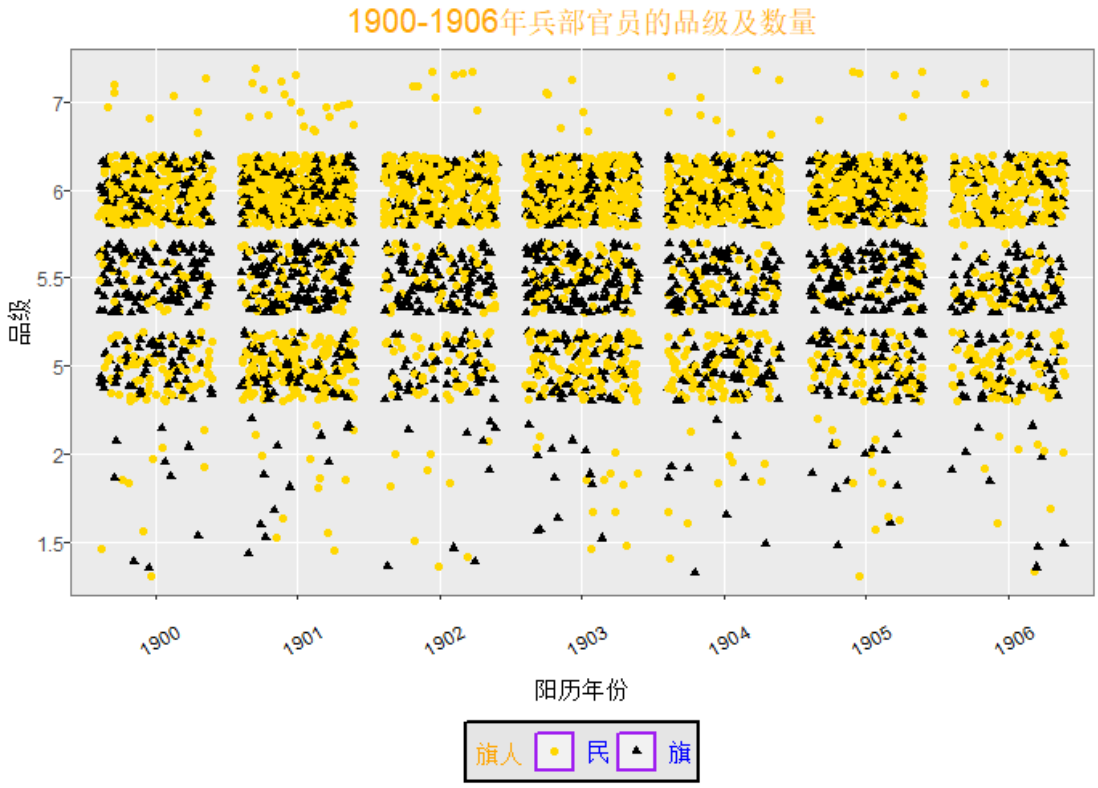


图 5.8 1900-1906 兵部官员的品级和数量（散点图）

从图5.7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。

5.3 折线图

折线图的制作制作逻辑是：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，我们尝试使用直线图来展现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：

```
114.  year <- c("1900","1901","1902","1903","1904","1905","1906")
115.  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)
116.  shenfen <- c("旗","旗","旗","旗","旗","旗","旗","民","民","民","
      民","民","民","民")
117.  兵部1900_1906满汉比例 <-
      data.frame(year,percentage,shenfen)
```

第114-117条代码的思路是根据直方图章节创建“year”、“percentage”、“shenfen”三列数据。第116条数据是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。

数据集建立完成后，画出折线图：

```
118.  g5<- ggplot(兵部1900_1906满汉比例, aes(x = year, y =
```

```
    percentage, group = shenfen, colour = shenfen, shape=
    shenfen))
119.  g5+geom_point(size=2.5,stroke=0.5)+   geom_line()+
    scale_color_manual(values = c('gold','black'))+
    scale_shape_manual(values = c(1,5))+ labs(x="年份",y="百分比
    ",title = "1900-1906 兵部衙门满汉比例折线图
    ")+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,"%")), family =
    "serif", stat = "identity", size = 4, vjust = 1.5, hjust = 0.5, color
    = "red2", position = position_dodge(0.1))
```

对比直方图和散点图，折线图的代码有如下不同点：第一，第 118 代码的 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。第二，第 119 条代码的“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。第三，第 119 条代码最后的 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。

运行效果如下：

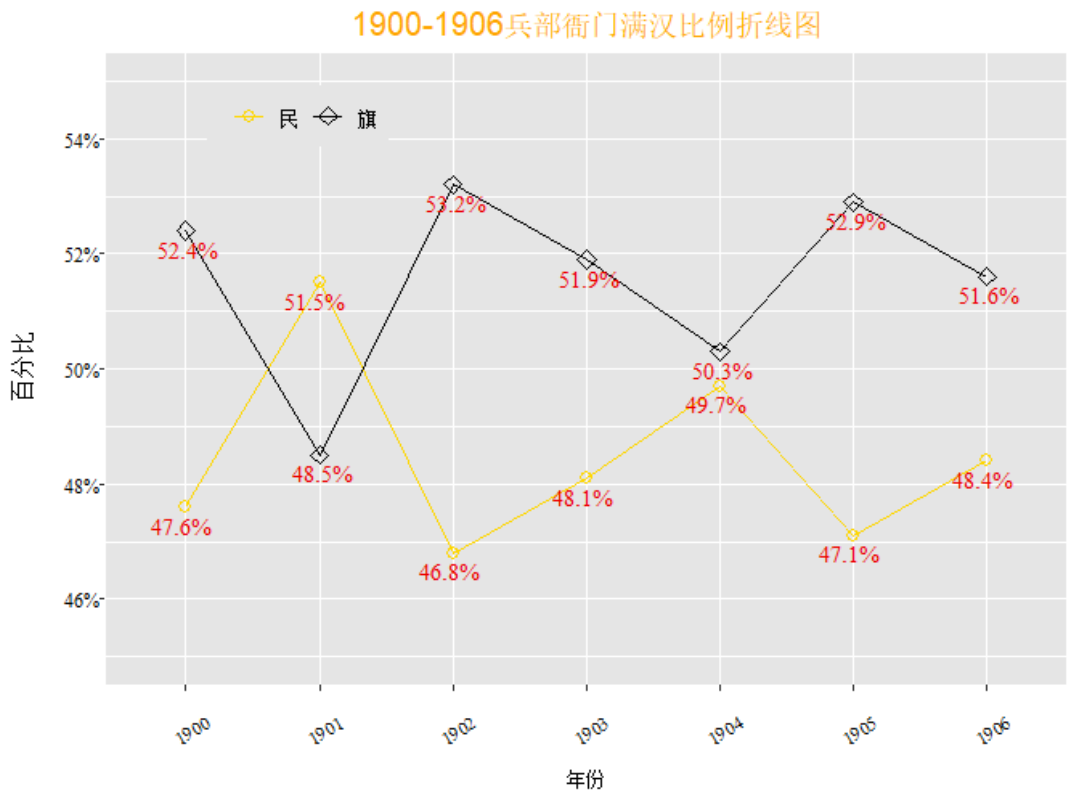


图 5.9 1900-1906 兵部官员的品级和数量（折线图）

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6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：

```
120. 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。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。

这些代码运行之后，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：

```
121.  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：[①]

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：by 指定的列（即公共列）是否要排序。

suffixes：指定除 by 外相同列名的后缀。

incomparables：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：

```
122. library(haven)
123. Chushen_Recodes_1_ <- read_dta("E:/R/Chushen
     Recodes(1).dta")
124. library(haven)
125. jiguansheng_Recodes <- read_dta("E:/R/jiguansheng
     Recodes.dta")
```

以上四条代码是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，请使用者在中国人民大学“清史数据共享平台”下载。请注意，read_data（）函数

① 参考链接： https:// zhuanlan.zhihu.com/p/130993141.

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。

接着，我们尝试整理“出身一”变量：

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

第126条代码是创造一个剔除掉空白名的新数据集，这条代码的逻辑在“创建新变量”章节讲过，此处不再赘述。运行这条代码的目的是保留“JSL1900_1912_delete_kongbaiming”这个数据集，因为我们在后面还会用到，所以在新数据“JSL1900_1912_delete_kongbaiming_1”这个数据集上匹配。

第127条代码的主数据集是“JSL1900_1912_delete_kongbaiming_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，这两条代码的含义是由主数据集和副数据集根据它们相同的列"出身一"所匹配而创建的新数据集

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_delete_kongbaiming_2”。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：

```
128. JSL1900_1912_clean_2$qiren <- ifelse( (JSL1900_1912_clean$旗
     分!="" | JSL1900_1912_clean$身份二!="" | JSL1900_1912_clean$姓
     =="") ,1 ,0 )
129. table(J JSL1900_1912_clean_2$出身一,
     JSL1900_1912_clean_2$qiren)
130. table(JSL1900_1912_clean_2$chushen_category,
     JSL1900_1912_clean_2$qiren)
```

```
131.  library(table1)
132.  table1(~chushen_category|qiren, data =
      JSL1900_1912_clean_2, overall = "total")
```

这三条代码用以检验匹配成果。第 127 条代码是在新数据集上创建“qiren”这一变量，在第三章“逻辑表达式”中讲过。第 128 条代码是用原始的“出身一”变量制表，可以看出非常杂乱。第 129 条代码使用新的“chushen_category”变量制表，可以看出较为统一，这也表明匹配成功，“出身一”变量得到整理。第 130-131 条代码用到的是 table1()函数，主要用来检验匹配的结果，通过 total 总数也可识别是否所有的行都被匹配上了，运行效果如下：

	0 (N=484559)	1 (N=154873)	total (N=639432)
chushen_category			
翻譯進士	735 (0.2%)	71 (0.0%)	806 (0.1%)
翻譯舉人	1076 (0.2%)	108 (0.1%)	1184 (0.2%)
翻譯生員	4852 (1.0%)	404 (0.3%)	5256 (0.8%)
貢生	45322 (9.4%)	12219 (7.9%)	57541 (9.0%)
貢生 - 异途	55618 (11.5%)	19107 (12.3%)	74725 (11.7%)
監生	136783 (28.2%)	43244 (27.9%)	180027 (28.2%)
進士	30663 (6.3%)	9457 (6.1%)	40120 (6.3%)
舉人	72470 (15.0%)	27357 (17.7%)	99827 (15.6%)
空白	100583 (20.8%)	31824 (20.5%)	132407 (20.7%)
吏員	2271 (0.5%)	308 (0.2%)	2579 (0.4%)
生員	6463 (1.3%)	1047 (0.7%)	7510 (1.2%)
武職出身	1680 (0.3%)	81 (0.1%)	1761 (0.3%)
蔭生	9078 (1.9%)	2925 (1.9%)	12003 (1.9%)
Missing	16965 (3.5%)	6721 (4.3%)	23686 (3.7%)

图 6.1 缙绅录数据中官员的出身构成和满汉比例

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，所以我们需要创建一个新的数据集来匹配已经匹配好"出身一"的数据集。

```
133. JSL1900_1912_clean_3 <- merge(JSL1900_1912_clean_2,
     jiguansheng_Recodes, by = "籍贯省",all= TRUE)
```

注意，R 一次只能匹配一个副数据集，因此，整理下一个变量只能在上一个已整理好变量的数据集上进行，我们这里生成的是新的数据集。这条代码“all=TRUE”表示全连接，不过在缙绅录中，左连接和全连接作用是一样的。

检验方式与之前相同：

```
134. table(JSL1900_1912_clean_3$籍贯省,
     JSL1900_1912_clean_3$qiren)
135. table(JSL1900_1912_clean_3$籍贯省_clean,
     JSL1900_1912_clean_3$qiren)
136. table1(~籍贯省_clean|qiren, data = JSL1900_1912_clean_3,
     overall = "total")
```

这里举了两个匹配的例子，大家还可以根据研究需求整理其他的变量，比如"地区"等。下面利用整理好的变量来绘图。

```
137. library(ggplot2)
138. JSL1900_1912_clean_3 <- filter(JSL1900_1912_clean_3, !is.na(阳
     历年份) & !is.na(chushen_category))
```

第 138 条代码是利用 filter()函数清理掉含有 NA 的行，清理掉多余的变量之后图形会更加简洁。下面利用之前学习过的 ggplot()函数来制作一个简单的直方图：

```
g6 <- ggplot(JSL1900_1912_clean_3, aes(阳历年份, fill =
    chushen_category,))
g6+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(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(color = "purple"),
    panel.background = element_rect(color = "grey50"),
    panel.grid = element_line(color="grey50", linewidth = 0.1))
```

代码运行效果如下：

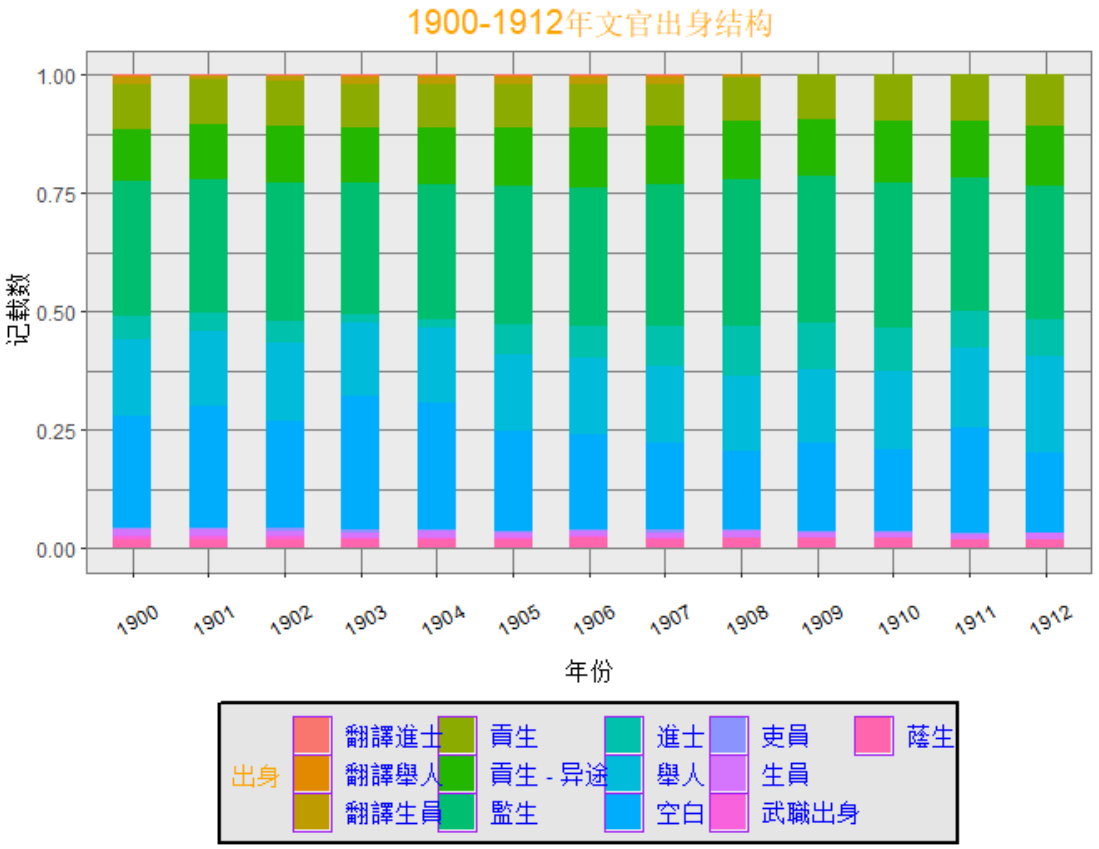


图 6.2 1900-1912 年文官出身结构变化

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 R 中。

（6）利用 merge 函数进行匹配。

下面以“铨选方式”的整理为例：

```
141.  table(JSL1900_1912_clean_3$铨选方式,
      JSL1900_1912_clean_3$qiren)
142.  library(readxl)
143.  quanxuanfangshi_sort <-
      read_excel("E:/R/quanxuanfangshi_sort.xlsx")
144.  JSL1900_1912_clean_4 <- merge(JSL1900_1912_clean_3,
      quanxuanfangshi_sort_1, by = "铨选方式", all= TRUE)
145.  table(JSL1900_1912_clean_4$铨选方式_clean,
      JSL1900_1912_clean_4$qiren)
```

第 141 条代码是用 table（）函数将某个变量的所有记录类型都显示出来，然后复制到 excel 中。第 141-142 条代码中间，需要使用者按照前述步骤在 excel 中自行尝试完成，并保存为数据集。第 142 条代码是运行 readxl（）函数，这一函数能读取 EXCEL 文件。

第 143 条代码即为导入保存的数据集到 RStudio 中。第 144 条代码即是匹配。第 145 条代码是检验匹配成果，具体运行的效果如下：

表 6.1 不同民族属性官员的铨选方式差异

	民人	旗人
补	60947	2498
不详	181	16
调	7636	425
空白	7	775
升	3762	158

授	1557	1037
署	27	0
题	110	2
袭	10	0
选	69405	1511

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”。

下面我们尝试创建只有 1900 年秋记录和只有 1901 年秋记录的数据集：

```
146. 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
147. 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
```

之后我们尝试用 inner_join（）函数将两个数据集连接起来，看看有多少官员 1900 秋-1901 秋在任：

```
148. 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”表示合并之后保留副数据集的连接列吗？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7章　数据集的内外链接

7.1 数据集内部记录的链接

7.1.1 创建官员的个人编号

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链接。

那么，如何在缙绅录数据库中创建内链接呢？步骤如下：

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。

```
149. install.packages("tidyverse")

150. library(tidyverse)

151. JSL1906 <- subset(JSL1900_1912_clean,阳历年份 == "1906")
```

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。

```
152. JSL1906$RecordNumber <- 1:nrow(JSL1906)
```

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，其格式为：

```
arrange(.data,…,by_group)
```

.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。

按照 arrange（）函数的格式，我们按照姓名、身份、旗分、出身、年份、季节、序号变量对整个数据集进行排序，编写代码：

```
153. JSL1906 <- arrange(JSL1906,xingming,身份二,旗分,出身一,年份
     季节,RecordNumber)
```

在这一条代码中，“xinming…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，其格式为：

```
group_by(.data,…)
```

.data 为数据集。

…为指定的条件。

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。

```
154. 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。此复合函数分为两步：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，mutate（），是 dplyr 包下面的一个创建变量的函数。现在利用 mutate（）函数创建一个叫 PersonID 的变量。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。现在数据集中就有了一个只有逻辑值的

PersonID，但还没有结束，我们需要继续用到累加函数 cumsum（）给每个官员一个独一无二的编号。

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，格式为：

```
cumsum(data$var)
```

data 为数据集，var 为变量。

按照 cumsum（）函数的格式，编写如下代码创建一个累加个人编号的新变量：

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

新变量，取名为 PersonID_cumsum。现在，每个官员都拥有了一个独一无二的编号。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：

```
156. JSL1906 <- JSL1906 %>%
     group_by(PersonID_cumsum) %>% mutate(YearsServed =
     abs((年份季节-年份季节[1])+0.25))
```

与第 154 条函数类似，这条函数也是利用管道符建立组合代码函数。分组函数，采用的变量是 PersonID_cumsum。这次利用 mutate（）函数创建一个叫 YearsServed 的变量。Mutate（）里用到了嵌套函数 abs（）,即取绝对值，以防出现有负数的情况。然后用当前行的年份季节减去每个分组的第一个年份季节，再加上 0.25，即可求得每个官员的任职年限。为什么要加 0.25？因为我们在创造年份季节这个变量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。

通过制表来查看效果：

```
157. JSL1906_winter <- subset(JSL1906, 季节号 == 4)

158. library(table1)

159. table1(~jiguansheng_sort|YearsServed, data =

     JSL1906_winter, overall = "total")
```

运行效果如下：

	0.25 (N=776)	0.5 (N=610)	0.75 (N=1566)	1 (N=8889)	total (N=11841)
籍贯省					
	324 (41.8%)	217 (35.6%)	308 (19.7%)	4465 (50.2%)	5314 (44.9%)
不详	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
外国	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
安徽	35 (4.5%)	30 (4.9%)	80 (5.1%)	351 (3.9%)	496 (4.2%)
奉天	7 (0.9%)	1 (0.2%)	14 (0.9%)	47 (0.5%)	69 (0.6%)
福建	10 (1.3%)	15 (2.5%)	68 (4.3%)	98 (1.1%)	191 (1.6%)
甘肃	1 (0.1%)	6 (1.0%)	22 (1.4%)	31 (0.3%)	60 (0.5%)
广西	6 (0.8%)	13 (2.1%)	41 (2.6%)	59 (0.7%)	119 (1.0%)
广东	16 (2.1%)	11 (1.8%)	50 (3.2%)	121 (1.4%)	198 (1.7%)
贵州	15 (1.9%)	11 (1.8%)	43 (2.7%)	94 (1.1%)	163 (1.4%)
河南	9 (1.2%)	13 (2.1%)	81 (5.2%)	155 (1.7%)	258 (2.2%)
黑龙江	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
湖北	14 (1.8%)	16 (2.6%)	58 (3.7%)	206 (2.3%)	294 (2.5%)
湖南	52 (6.7%)	26 (4.3%)	51 (3.3%)	330 (3.7%)	459 (3.9%)
吉林	2 (0.3%)	1 (0.2%)	1 (0.1%)	0 (0%)	4 (0.0%)
江南	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
江苏	41 (5.3%)	38 (6.2%)	141 (9.0%)	496 (5.6%)	716 (6.0%)
江西	22 (2.8%)	17 (2.8%)	82 (5.2%)	218 (2.5%)	339 (2.9%)
山东	24 (3.1%)	25 (4.1%)	90 (5.7%)	207 (2.3%)	346 (2.9%)
山西	11 (1.4%)	18 (3.0%)	35 (2.2%)	65 (0.7%)	129 (1.1%)
陕西	4 (0.5%)	9 (1.5%)	40 (2.6%)	77 (0.9%)	130 (1.1%)
顺天	32 (4.1%)	24 (3.9%)	57 (3.6%)	544 (6.1%)	657 (5.5%)
四川	45 (5.8%)	27 (4.4%)	51 (3.3%)	308 (3.5%)	431 (3.6%)

天津	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
新疆	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0 (0%)
云南	7 (0.9%)	7 (1.1%)	33 (2.1%)	66 (0.7%)	113 (1.0%)
浙江	62 (8.0%)	59 (9.7%)	116 (7.4%)	734 (8.3%)	971 (8.2%)
直隶	37 (4.8%)	26 (4.3%)	104 (6.6%)	217 (2.4%)	384 (3.2%)

图 7.1 利用个人编号来查看官员任职年限

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，比如，在任期 0.25 年这一列中，浙江也是最高的，达到 8.0%。当然，这不是说浙江即是任期稳定性最高的省份，又是任期稳定性最低的省份，而是我们此处只有 1906 年的数据，而 1906 年以前的数据没有被囊括，引起浙江官员任期时间既稳定又不稳定的歧义。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 500000 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 50000 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：

```
160. JSL1906_人数 <- JSL1906 %>% filter( PersonID == 1 )
```

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。

现在，我们利用删除掉官员重复任职记录的数据集来制图并查看运行效果。

```
161. library(ggpolt2)
```

```
g7 <- ggplot(JSL1900_1912_bingbu, aes(jiguansheng_sort ))
g7+geom_bar(just= 0.5, width = 0.5)+ labs(x = "籍贯", y = "人数", title = "1900-1906 年兵部的地理来源（人数）
")+geom_text(aes(label = ..count.., family = "serif"), stat = "count", size = 3.5, vjust = -0.8, hjust = 0.5, color = "red2")
```

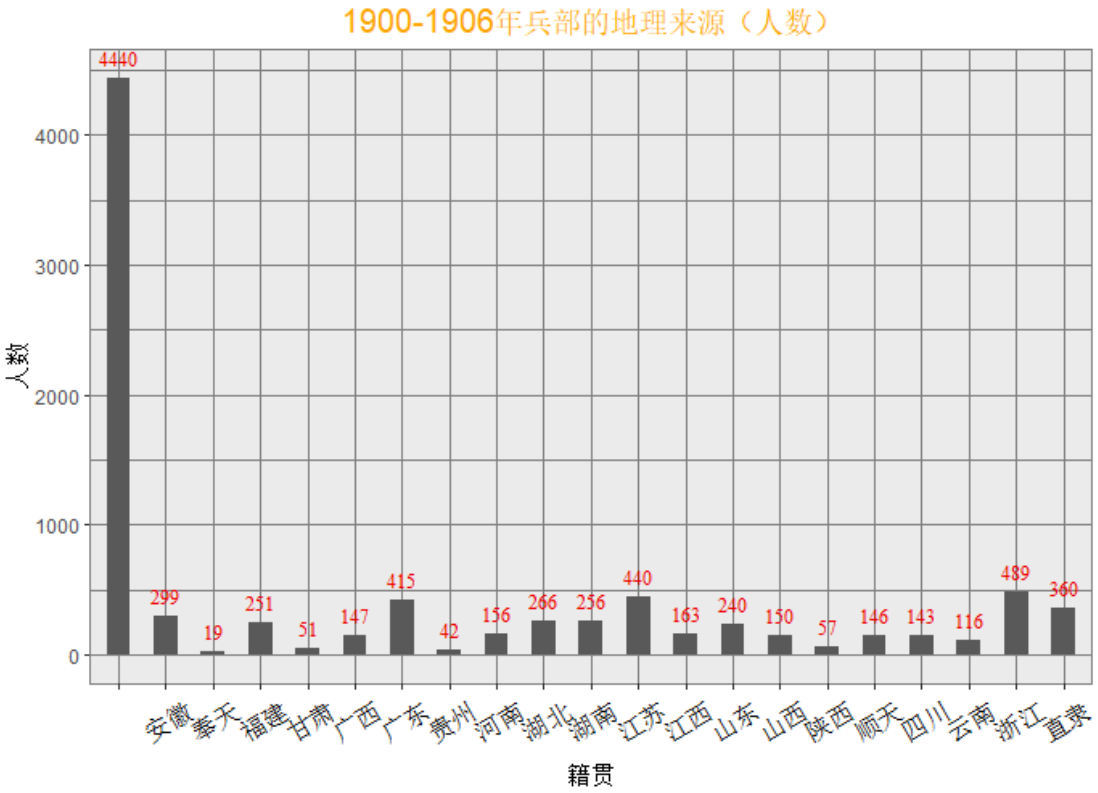


图 7.2 1900-1906 年兵部的地理来源（人数）

从图 7.2 中，可以看出，1906 年兵部官员大多数都是“旗人”（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，兵部官员多来自“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“京官”的概率更高。这个图形，就是用人数来做的。这与我们之前一直用记载数来制表制图有很大不同，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。

7.2 数据集与外部数据库的链接（模糊匹配）

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“名字”这个变量去进行模糊匹配，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，讲解数据库外链接的步骤：

第一步，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。

```
164. JSL1900_1912_clean %>% filter((阳历年份 == "1906" & 地区
     == "四川省" & zhixian == 1)) %>% select(阳历年份,地区,机构一,官
     职一,xingming,字号,籍贯省,籍贯县,旗分,出身一,年份季节) ->
     JSL1906_sichuan_zhixian
165. JSL1900_1912_clean %>% filter((阳历年份 == "1910" & 地区 ==
     "四川省" & zhixian == 1)) %>% select(阳历年份,地区,机构一,官职
     一,xingming,字号,籍贯省,籍贯县,旗分,出身一,年份季节) ->
     JSL1910_sichuan_zhixian
```

第二步，利用模糊匹配函数编写代码，匹配两个数据集。fuzzy_join（）函数，可对两个数据集进行模糊匹配，格式为：[①]

① 参考网址：www.idata8.com/rpackage/fuzzyjoin/fuzzy_join.html.

```
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。

下面，根据这一函数的格式编写代码：

```
166. install.packages("fuzzyjoin")
167. library(fuzzyjoin)
168. JSL1906_1910_zhixian <-
     stringdist_inner_join(JSL1906_sichuan_zhixian,
     JSL1910_sichuan_zhixian, by = "xinming", method = "dl",
     max_dist = 1, distance_col = NULL )
```

“stringdist_inner_join”参数是模糊匹配系列函数中一个较为直接、简单的函数，其用法和 merge()函数相似

“by = ”参数根据“xinming”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。

可以看出，模糊匹配所生成的数据集可能比原始两个数据集都大，因为模糊匹配选择的公共列只有一项，R 会将两个数据集中任何满足这一模糊匹配项的行全都找出来。R 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，其计算过程中只有以给定的少量信息为基准，在最大程度上找到与之相关的样本，因此称为模糊匹配。模糊匹配与精准匹配（我们之前学习的 merge 函数）的匹配结果会有所不同。但是，我们也发现，模糊匹配和精准匹配其逻辑是相似的，甚至可以说是相同的。那么，在哪些情况下可以使用模糊匹配呢？当两个数据集存在较大的结构差异、或者某个数据集信息量大而另一个信息量少、匹配时缺乏足够的精准匹配信息时，就可以采用模糊匹配的方法。

但是，fuzzyjoin 这一类模糊匹配的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模糊匹配之前，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，最后再使用模糊匹配。STATA 也具有类似的模糊匹配方法，它的内核是利用匹配分数来确定匹配程度，这种方法相较于 fuzzyjoin 则更加简便高效。

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”为例，列表展示了“缙绅录”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。[①]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。

① Cameron Campbell and Bijia Chen, “Nominative Linkage of Records of Officials in the China Government Employee Dataset-Qing (CGED-Q), ” *Historical Life Course Studies*, vol.12, no.5 (September 2022), pp.233-259.

第8章 总结

很荣幸能和大家一起学习R语言。到这里，历史数据库分析的基础内容已经全部结束了，相信大家已经学到运用R语言处理数据集的一些技巧。

初学R语言，大家肯定会觉得非常难。的确，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甚至称不上一个分析软件，它实际上是一个语言编程的环境）。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。

我们从0开始一起学习R，从一些最基本的操作，逐步延伸到比较高级的操作，我们到目前学习了：

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

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 () 、 str_detect ())、替换字符（ gsub ())；

3.制表-简单制表（table ())、整理变量（factor ())、制作可以导出的表格（table1 ())、指定条件制表（subset ()以及&和|符号的运用）

4.制图-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_point () +geom_line () ）

5.数据集的匹配-筛选列（ %>% 符号的运用、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 () ）

6.Record linkage-排序（arrange ())、分组（group_by ())、生成新变量（mutate ())、累加（cumsum ())、模糊匹配（ stringdist_inner_join ())、 Probabilistic Record Linkages

以上就是基础教程的总结。希望大家在之后的学习生涯中，不断精进自己的数据分析技巧，能够独自处理数据库的诸多问题。

附录

本教程使用到的代码合集

```
1. library(haven)
2. dataset <- read_dta("E:/R/JSL1900-1912.dta")
3. data <- read.table("clipboard",head=T)
4. edit(JSL1850_1864csv)
5. kongbaiming <- which(JSL1900_1912$名=="空白")
6. JSL1900_1912_clean <- JSL1900_1912[-kongbaiming,]
7. JSL1900_1912_clean$阳历年份 numeric <- as.numeric(JSL1900_1912_clean$阳历年
   份)
8. typeof(JSL1900_1912_clean$阳历年份 numeric)
9. JSL1900_1912_clean$季节号 numeric <- as.numeric(JSL1900_1912_clean$季节号)
10. typeof(JSL1900_1912_clean$季节号 numeric)
11. JSL1900_1912_clean$年份季节 <- ( JSL1900_1912_clean$阳历年份 numeric +
    ( ( JSL1900_1912_clean$季节号 numeric/4 )-0.25) )
12. table(JSL1900_1912_clean$年份季节)
13. ifelse(年份<1900,T,F)
14. JSL1900_1912_clean$qiren <- ifelse((JSL1900_1912_clean$旗分!="" |
    JSL1900_1912_clean$身份二!="" | JSL1900_1912_clean$姓==""),1,0)
15. table(JSL1900_1912_clean$qiren)
16. JSL1900_1912_clean$出身一[JSL1900_1912_clean$出身一 == "狀元" ] <- 1
17. JSL1900_1912_clean$出身一[JSL1900_1912_clean$出身一 == "榜眼" ] <- 2
18. JSL1900_1912_clean$出身一[JSL1900_1912_clean$出身一 == 1 ] <- "狀元"
19. JSL1900_1912_clean$出身一[JSL1900_1912_clean$出身一 == 2 ] <- "榜眼"
20. JSL1900_1912_clean$qiren[JSL1900_1912_clean$qiren == 1] <- "旗"
21. JSL1900_1912_clean$qiren[JSL1900_1912_clean$qiren == 0] <- "民"
22. table(JSL1900_1912_clean$qiren)
23. JSL1900_1912_clean$季节[JSL1900_1912_clean$季节 == 1 ] <- "春"
24. JSL1900_1912_clean$季节[JSL1900_1912_clean$季节 == 2 ] <- "夏"
25. JSL1900_1912_clean$季节[JSL1900_1912_clean$季节 == 3 ] <- "秋"
26. JSL1900_1912_clean$季节[JSL1900_1912_clean$季节 == 4 ] <- "冬"
27. table(JSL1900_1912_clean$季节)
28. NBA$score[NBA$score >= 30000] <- 3
29. NBA$score[NBA$score >= 20000 & NBA$score < 30000] <- 2
30. NBA$score[NBA$score >= 10000 & NBA$score < 20000] <- 1
31. NBA$score <- factor(NBA$score, levels = c(1,2,3), labels = c("一万分先生","两万分
    先生","三万分先生"))
32. NBA$score_origin <- factor(NBA$score, levels = c("一万分先生","两万分先生","三
    万分先生"), labels = c(1,2,3))
33. NBA$assistant[NBA$assistant <= 5000] <- 1
```

```
NBA$assistant[NBA$assistant > 5000 & NBA$assistant <= 9000] <- 2
NBA$assistant[NBA$assistant > 9000 & NBA$assistant <= 15000] <- 3
NBA$assistant == factor(NBA$assistant, levels = c(1,2,3), labels = C("助攻专家","助攻机器","助攻王"))
JSL1900_1912_clean$xinming <-paste(JSL1900_1912_clean$姓, JSL1900_1912_clean$名, sep="",collapse=NULL)
str_extract(JSL1900_1912_clean$官职一 ,"員外郎")
str_detect(JSL1900_1912_clean$官职一 ,"員外郎")
JSL1900_1912_clean$zongdu <- ifelse( str_detect(JSL1900_1912_clean$官职一,"總督"),1,0)
JSL1900_1912_clean$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$官职一,"額外"),1,0)
gsub("鑲藍","镶蓝", JSL1900_1912_clean$旗分)
table(JSL1900_1912_clean$阳历年份)
table(JSL1900_1912_clean$籍贯省, JSL1900_1912_clean$qiren)
table(JSL1900_1912_clean$籍贯省, JSL1900_1912_clean$qiren, exclude= NULL, useNA= "always")
table(JSL1900_1912_clean$籍贯省)
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","甘肃","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"))
table(JSL1900_1912_clean$jiguansheng_sort, JSL1900_1912_clean$qiren, exclude= NULL, useNA= "always")
table(JSL1900_1912_clean$出身一, JSL1900_1912_clean$qiren)
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","附贡","附貢生","副贡","副貢","副貢生","廩贡","廩貢","廩貢生","歲貢","歲貢","歲貢生","優貢","優貢生","增贡","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"))
```

```
table(JSL1900_1912_clean$gongsheng)
JSL1900_1912_clean$dingjia <- factor(JSL1900_1912_clean$出身一, levels = c("状元","榜眼","探花"),labels = c("鼎甲","鼎甲","鼎甲"))
table(JSL1900_1912_clean$dingjia, JSL1900_1912_clean$qiren, exclude= NULL, useNA= "always")
table(JSL1900_1912_clean$gongsheng, JSL1900_1912_clean$qiren, exclude= NULL, useNA= "always")
table(JSL1900_1912_clean$jiguansheng_sort, JSL1900_1912_clean$gongsheng, exclude= NULL, useNA= "always")
install.packages("table1")
library(table1)
help(table1)
label(JSL1900_1912_clean$jiguansheng_sort) <-"籍贯省"
label(JSL1900_1912_clean$qiren) <- "旗人"
label(JSL1900_1912_clean$gongsheng) <- "贡生"
label(JSL1900_1912_clean$zhixian) <- "知县"
table1(~jiguansheng_sort|qiren, data = JSL1900_1912_clean, overall = "total")
table1(~jiguansheng_sort+gongsheng|qiren, data = JSL1900_1912_clean, overall = "total")
JSL1900_1912_clean$zhixian[JSL1900_1912_clean$zhixian == 1] <- "知县"
JSL1900_1912_clean$zhixian[JSL1900_1912_clean$zhixian == 0] <- "非知县"
table1(~jiguansheng_sort+gongsheng+dingjia|zhixian*qiren, data = JSL1900_1912_clean, overall = "Overall")
JSL1900_1912_clean$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"
JSL1900_1912_only1910 <-subset(JSL1900_1912_clean,阳历年份 numeric == 1910)
table1(~jiguansheng_sort|qiren*贡生, data = JSL1900_1912_only1910, overall = "total",rowlabelhead = "")
JSL1900_1912_only 京师  <- subset(JSL1900_1912_clean,jiguansheng_sort == "京师")
table1(~qiren|贡生, data = JSL1900_1912_only 京师, overall = "total",rowlabelhead = "")
JSL1900_1912_京师 1910 <-subset(JSL1900_1912_clean,jiguansheng_sort == "京师"&  阳历年份 numeric == 1910)
table1(~籍贯省|qiren*贡生, data = JSL1900_1912_京师 1910, overall = "total",rowlabelhead = "")
JSL1900_1912_京师 1910 候补  <-subset(JSL1900_1912_clean,jiguansheng_sort == "京师" &  阳历年份 numeric == 1910 & houbu == 1)
table1(~籍贯省|qiren*贡生, data = JSL1900_1912_京师 1910 候补, overall =
```

```
"total",rowlabelhead = "")
JSL1900_1912_西南三省 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|贡生*qiren, data = JSL1900_1912_西南三省 after1908, overall = "total",rowlabelhead = "")
install.packages("ggplot2")
library(ggplot2)
help("ggplot2")
g1 <- ggplot(JSL1900_1912_only1910, aes(x = jiguansheng_sort))
g1+geom_bar(color="red",fill = "white",just= 0.5, width = 0.8)
JSL1900_1912_bingbu <- subset(JSL1900_1912_clean,机构一 == "兵部衙門")
g2 <- ggplot(JSL1900_1912_bingbu, aes(x = 阳历年份 numeric, fill = qiren))
g2+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")+guides(fill=guide_legend(title = "旗人"))+scale_fill_manual(values = c("cyan","pink"))+ theme(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(size = 2, color = "purple"),panel.background = element_rect(color = "grey50"),panel.grid = element_line(color="grey50",linewidth = 0.1))+scale_x_continuous(expand = c(0.1,0.1), breaks = seq(1900,1906,1))+ scale_y_continuous(limits = c(0,1000), breaks = seq(0,1000,100))+ geom_text(aes(label = ..count.., family = "serif"),stat = "count",size = 3.5, vjust = -0.8, hjust = 0.5, color = "red2",position = position_dodge(0.5))
g2 <- ggplot(JSL1900_1912_bingbu, aes(x = 阳历年份, fill = qiren))
g2+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(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), "%"), family = "serif"),stat = "count",size = 5, vjust = -0.8,
color = "gray10",position = position_fill(0.5))
N <- ggplot(NBA,aes(...1,score_per_game))
N+geom_bar(color="black", fill="white", stat= "identity",just= 0.5,width = 1)+
xlab("players")+ ylab("ppg")+ 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=score_per_game), vjust=-0.8,
hjust=0.5, color="red")
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))
g3<- ggplot(data=subset(JSL1900_1912_bingbu, !is.na(品级)), aes(x = 阳历年份,品
级,shape = qiren,colour = qiren))
g3+geom_point(position = "jitter")+ 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(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))
```

```
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")
兵部 1900_1906 满汉比例 <- data.frame(year,percentage,shenfen)
g5<- ggplot(兵部 1900_1906 满汉比例, aes(x = year, y = percentage, group = shenfen, colour = shenfen, shape= shenfen))
g5+geom_point(size=2.5,stroke=0.5)+ geom_line()+ scale_color_manual(values = c('gold','black'))+ scale_shape_manual(values = c(1,5))+ labs(x="年份",y="百分比",title = "1900-1906 兵部衙门满汉比例折线图")+theme(legend.title = element_blank(), legend.text = element_text(family = 'serif'), legend.position = 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,"%")), family = "serif", stat = "identity", size = 4, vjust = 1.5, hjust = 0.5, color = "red2", position = position_dodge(0.1))
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
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
library(haven)
Chushen_Recodes_1_ <- read_dta("E:/R/Chushen Recodes(1).dta")
library(haven)
jiguansheng_Recodes <- read_dta("E:/R/jiguansheng Recodes.dta")
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)
JSL1900_1912_clean_2$qiren <- ifelse( (JSL1900_1912_clean$旗分!="" | JSL1900_1912_clean$身份二!="" | JSL1900_1912_clean$姓=="") ,1 ,0 )
table(J JSL1900_1912_clean_2$出身一, JSL1900_1912_clean_2$qiren)
table(JSL1900_1912_clean_2$chushen_category, JSL1900_1912_clean_2$qiren)
library(table1)
table1(~chushen_category|qiren, data = JSL1900_1912_clean_2, overall = "total")
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$籍貫省_clean, JSL1900_1912_clean_3$qiren)
table1(~籍貫省_clean|qiren, data = JSL1900_1912_clean_3, overall = "total")
library(ggplot2)
JSL1900_1912_clean_3 <- filter(JSL1900_1912_clean_3, !is.na(阳历年份)
    & !is.na(chushen_category))
g6 <- ggplot(JSL1900_1912_clean_3, aes(阳历年份, fill = chushen_category,))
g6+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(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(color = "purple"), panel.background = element_rect(color = "grey50"),
    panel.grid = element_line(color="grey50", linewidth = 0.1))
table(JSL1900_1912_clean_3$铨选方式, JSL1900_1912_clean_3$qiren)
library(readxl)
quanxuanfangshi_sort <- read_excel("E:/R/quanxuanfangshi_sort.xlsx")
JSL1900_1912_clean_4 <- merge(JSL1900_1912_clean_3, quanxuanfangshi_sort_1,
    by = "铨选方式", all= TRUE)
table(JSL1900_1912_clean_4$铨选方式_clean, JSL1900_1912_clean_4$qiren)
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
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")
install.packages("tidyverse")
library(tidyverse)
JSL1906 <- subset(JSL1900_1912_clean,阳历年份  == "1906")
JSL1906$RecordNumber <- 1:nrow(JSL1906)
JSL1906 <- arrange(JSL1906,xingming,身份二,旗分,出身一,年份季
    节,RecordNumber)
JSL1906 <- JSL1906 %>% group_by(xingming,旗分) %>% mutate(PersonID =
    ifelse(row_number() == 1, 1, 0))
JSL1906$PersonID_cumsum <- cumsum(JSL1906$PersonID)
JSL1906 <- JSL1906 %>%   group_by(PersonID_cumsum) %>% mutate(YearsServed =
    abs((年份季节-年份季节[1])+0.25))
JSL1906_winter <- subset(JSL1906,  季节号  == 4)
library(table1)
```

```
table1(~jiguansheng_sort|YearsServed, data = JSL1906_winter, overall = "total")
JSL1906_人数 <- JSL1906 %>% filter( PersonID == 1 )
library(ggpolt2)
g7 <- ggplot(JSL1900_1912_bingbu, aes(jiguansheng_sort ))
g7+geom_bar(just= 0.5, width = 0.5)+ labs(x = "籍贯", y = "人数", title = "1900-1906 年兵部的地理来源（人数）")+ theme(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), 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")
JSL1900_1912_clean %>% filter((阳历年份 == "1906" & 地区 == "四川省" & zhixian == 1)) %>% select(阳历年份,地区,机构一,官职一,xingming,字号,籍贯省,籍贯县,旗分,出身一,年份季节) -> JSL1906_sichuan_zhixian
JSL1900_1912_clean %>% filter((阳历年份 == "1910" & 地区 == "四川省" & zhixian == 1)) %>% select(阳历年份,地区,机构一,官职一,xingming,字号,籍贯省,籍贯县,旗分,出身一,年份季节) -> JSL1910_sichuan_zhixian
install.packages("fuzzyjoin")
library(fuzzyjoin)
JSL1906_1910_zhixian <- stringdist_inner_join(JSL1906_sichuan_zhixian, JSL1910_sichuan_zhixian, by = "xinming", method = "dl", max_dist = 1, distance_col = NULL )
```

R 语言教学 PPT 和程辑包下载网址

https://camerondcampbell.blog/resources-for-users-of-the-cged-q-jinshenlu-public-release/lessons-for-using-r-to-analyze-the-cged-q-jsl-public-release/