

R for bioinformatics

HUST Bioinformatics course series

Wei-Hua Chen (CC BY-NC 4.0)

06 April, 2023

- 1 section 1: R basics**
- 2 Section 2: setting up working environment**
- 3 section 3: dplyr, efficient data wrangler**
- 4 section 4: ggplot2, elegant graphics using R**
- 5 section 5: machine learning**
- 6 section 6: deep learning**
- 7 section 7: 结语**

section 1: R basics

R 语言简史

1993 到 2000 这段时间 R 只在小范围内流传。2000 年之后开始大爆发，用户数量直线上升。除去 R 本身的优秀之外，这种爆发与多个因素有关，比如自由软件的兴起，Linux 的成熟等等；经济危机也促进大家采用免费的自由软件替代统计领域的传统强者如 SPSS、SAS 和 Matlab 等（注：均为收费软件）。

首先，越来越多的学术文章使用 R 作为分析工具。根据来自著名学术搜索引擎 Google Scholar（谷歌学术）的数据，R 的流行趋势有以下两个特点：1) 在学术领域的市场份额逐年增加，且增势迅猛，2) R 是为数不多市场份额增加的统计软件之一。

接下来我们就用 R 把这个趋势画出来！如下面代码所示，所需代码包括 4 个部分：装入所需要的包，读取数据，处理数据和作图。运行这段代码，既专业又美观的图片就生成了！

R 的流行性调查

代码

```
library("ggplot2"); library("reshape2");

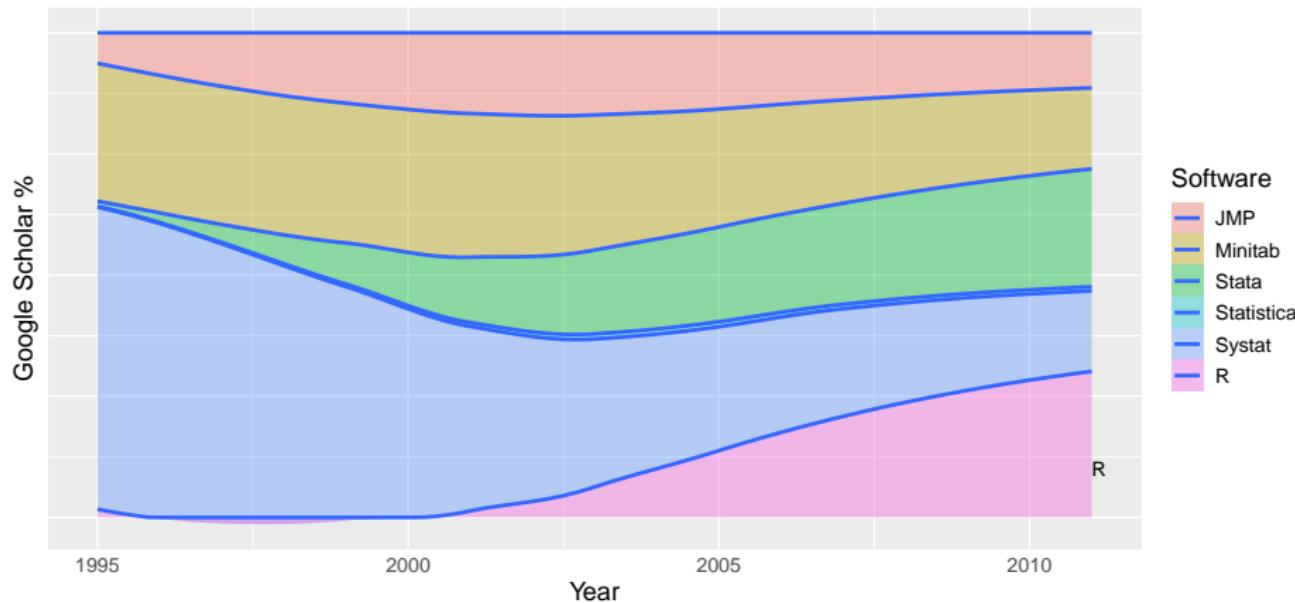
dat <- read.csv(file = "data/talk01/chaper01_preface_scholarly_impact_2012.4.9.csv");

cols.subset <- c("Year", "JMP", "Minitab", "Stata", "Statistica", "Systat", "R");
Subset <- dat[, cols.subset];
ScholarLong <- melt(Subset, id.vars = "Year");
names(ScholarLong) <- c("Year", "Software", "Hits");

plot1 <-
  ggplot(ScholarLong, aes(Year, Hits, group=Software)) + # 准备
  geom_smooth(aes(fill=Software), position="fill", method="loess") + # 画图
  ggtitle("Market share") + # 设置图标标题
  scale_x_continuous("Year") + # 改变 X 轴标题
  scale_y_continuous("Google Scholar %", labels = NULL) +
  theme(axis.ticks = element_blank(), text = element_text(size=14)) +
  guides(fill=guide_legend(title = "Software", reverse = F)) +
  geom_text(data = data.frame(Year = 2011, Software = "R", Hits = 0.10),
            aes(label = Software), hjust = 0, vjust = 0.5);
```

Market share, result

Market share



注：这里移除了市场占有率较大的 SAS 和 SPSS

R 的招聘趋势

其次，统计分析相关工作的招聘信息中要求申请者会用 R 的也越来越多了。根据美国招聘搜索引擎 indeed.com 的数据，自 2005 年（此搜索引擎提供的最早数据）起，需要用到 R 的招聘信息占总体招聘的比例逐年上升，目前仅排在 SAS 和 Matlab 之后，处于第 3 位。而且，除了 Stata 之外，R 是唯一一个占比上升。

同样的，我们用 R 把这个趋势画出来！

R job trends

代码

```
library("ggplot2"); ## 主作图包

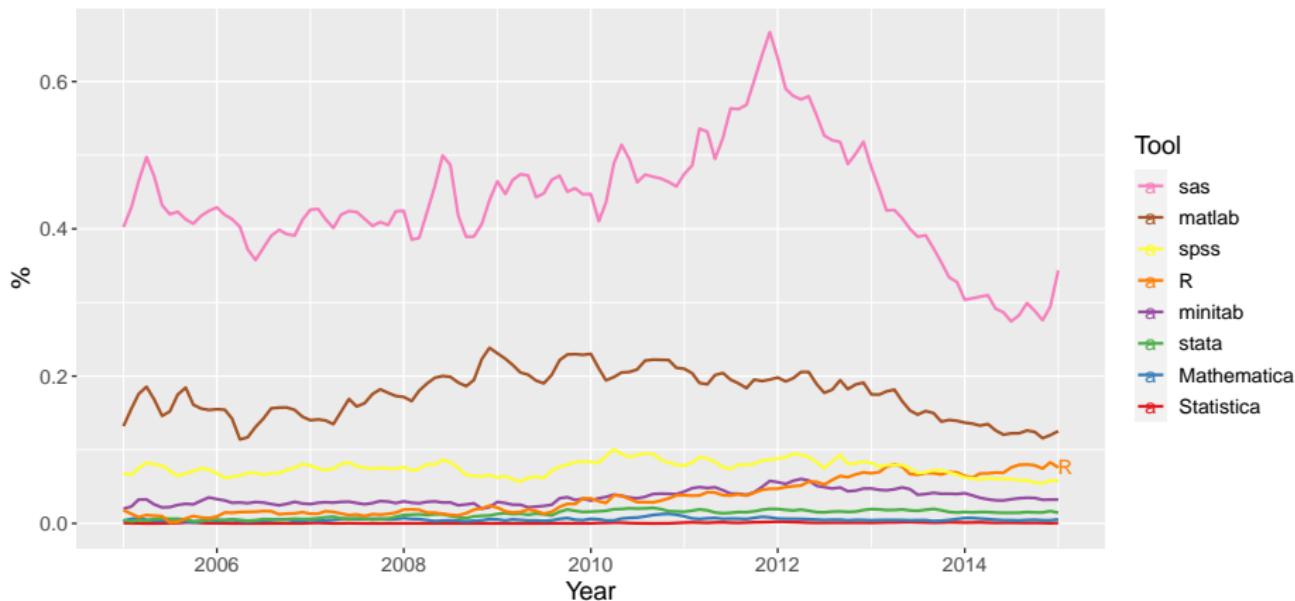
##2. -- 读取数据 --
dat <- read.table(file ="data/talk01/chaper01_preface_indeed_com_stats_2015.txt",
                  header = T, as.is = T);
##3. 处理数据
dat$date <- as.Date(dat$date); ## 把第一列改为日期

# 根据 job 对 software 进行调整
dat <- transform(dat, software = reorder(software, job));

plot2 <-
  ggplot( dat, aes( date, job, group = software, colour = software) ) +
  geom_line( size = 0.8 ) +
  ggtitle("Job trends (data from indeed.com)") + # 设置图标题
  xlab("Year") + ylab("%") +
  # 改变字体大小; 要放在 theme_grey() 后面
  theme( text = element_text(size=14) ) +
  guides(colour=guide_legend( title = "Tool", reverse = TRUE )) +
  scale_colour_brewer(palette="Set1") + # 改变默认颜色
  geom_text(data = dat[dat$date == "2015-01-01" & dat$software %in% c("R"), ],
            aes(label = software), hjust = 0, vjust = 0.5);
```

R job trends, plot

Job trends (data from indeed.com)



Popularity of Programming language 2020

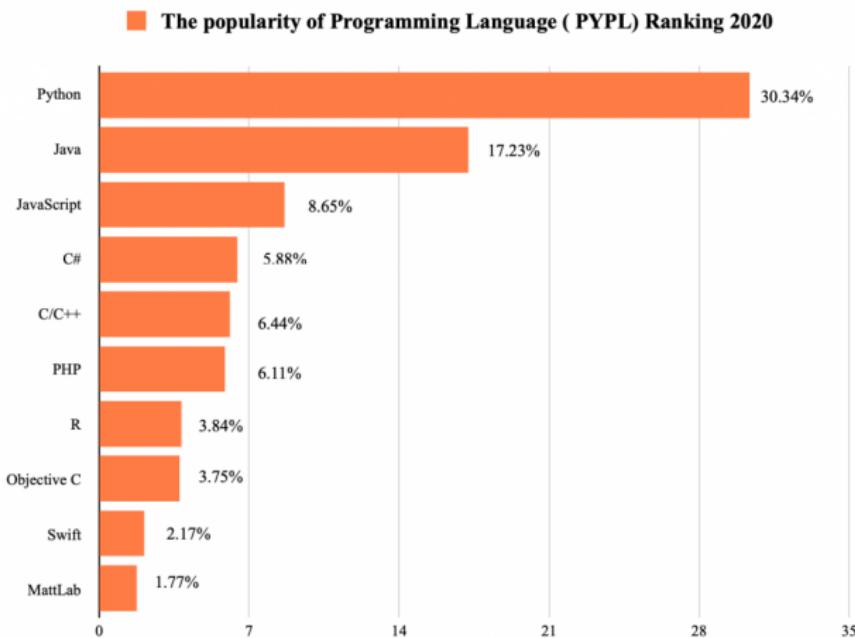


Figure 1: PYPL ranking (<https://pypl.github.io/>)

Programming languages for bioinformatics

Perl 或 Python

- 强大的文本处理能力（包括序列）
- 不错的运行速度（尤其是 Python）
- 强大的生信和统计学扩展包（尤其是 Python）
- 方便的并行计算

R

- 强大的格式数据处理能力（二维表格, dplyr）
- 无以伦比的统计学专业性
- 专业而好看的数据可视化软件（ggplot2）
- 专业的生信扩展包（Bioconductor）
- 超级好用的整合开发环境 IDE（RStudio）

我用过的 programming languages

- - C
- - Perl
- - R
- - PHP
- - Java
- - MySQL
- - HTML
- - Javascript

Evolview ver3.0
 cited 225 times in 2021 (ver2+3),
 166 so far (as of Aug 30, 2022)

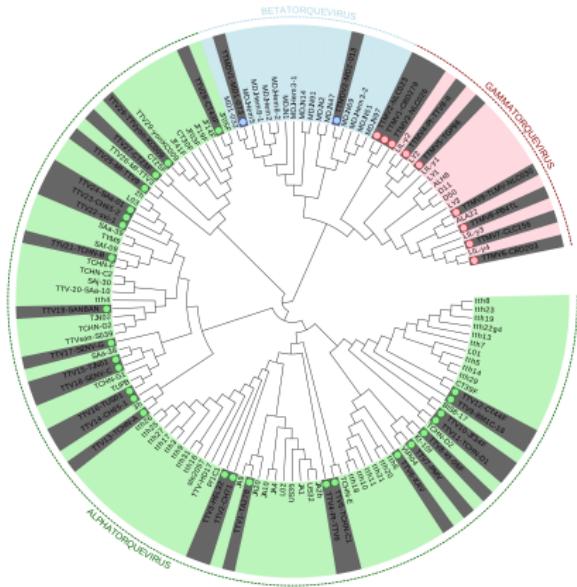


Figure 2: .Evolview showcase 3

网站链接、参考文献和扩展阅读

综上所述，R 已经是最流行的免费统计分析软件，排名仅在几个传统的分析软件之后，而且大有赶超它们的趋势。学好 R，不仅有助于在学术研究领域的发展，对找工作也有不少的帮助。

- R 的官方网站: <http://www.r-project.org>
- R 档案综合网络，即 CRAN(Comprehensive R Archive Network):
<http://cran.r-project.org/>
- ggplot2: <http://ggplot2.org/>
- RStudio: <https://posit.co/>
- 如何从 Google Scholar 抓取引用数据:
<http://librestats.com/2012/04/12/statistical-software-popularity-on-google-scholar/>
- indeed 招聘趋势: www.indeed.com/jobtrends
- R for data science: <https://r4ds.had.co.nz> (必读!!)

Section 2: setting up working environment

Install R

Go to <https://mirrors.tuna.tsinghua.edu.cn/CRAN/> (清华镜像), R supports mainstream operating systems including Linux, Windows and MacOS, please download the corresponding installation files according to the operating system. As shown in the figure below:

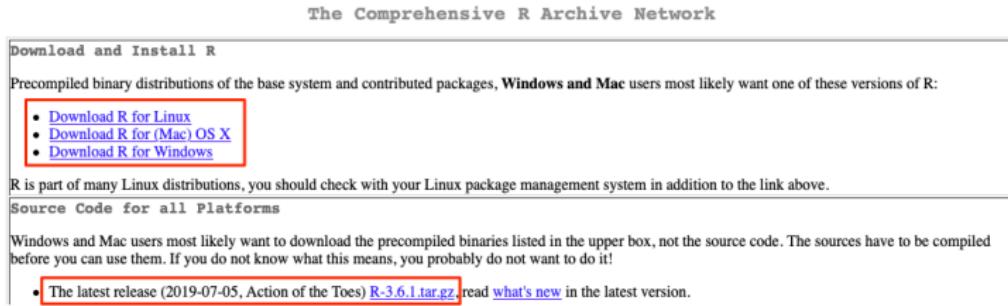


Figure 3: Select the appropriate installation package

New version of Mac OS X needs to be installed

XQuartz (<http://xquartz.macosforge.org/landing/>)。某些还需要用到 Xcode, 可以从 App Store 免费安装。

Install R on Linux

目前大多 Linux 发行版都带有 R，因此可直接使用。从 CRAN 下载文件进行安装稍嫌复杂，要求用户对 Linux 系统有一定的了解，而且需要有管理员权限。建议初级用户在 Linux 高手指导下安装。点击上图中的"Download R for Linux" 后，发行版为 Redhat（红帽）或 Suse 的用户要先阅读网站上提供的 readme 或 readme.html 文件，然后其中的指示进行安装。这里就不再累述了。

r-base-core_3.1.3-1lucid_amd64.deb 或 r-base-core_3.1.2-1lucid0_i386.deb
----- ----- ----- -----
1 2 3 4 5 1 2 3 4 5

Figure 4: R 安装包文件名

R studio

RStudio 可以从 <https://posit.co/download/rstudio-desktop/> 下载，支持等主流的操作系统。

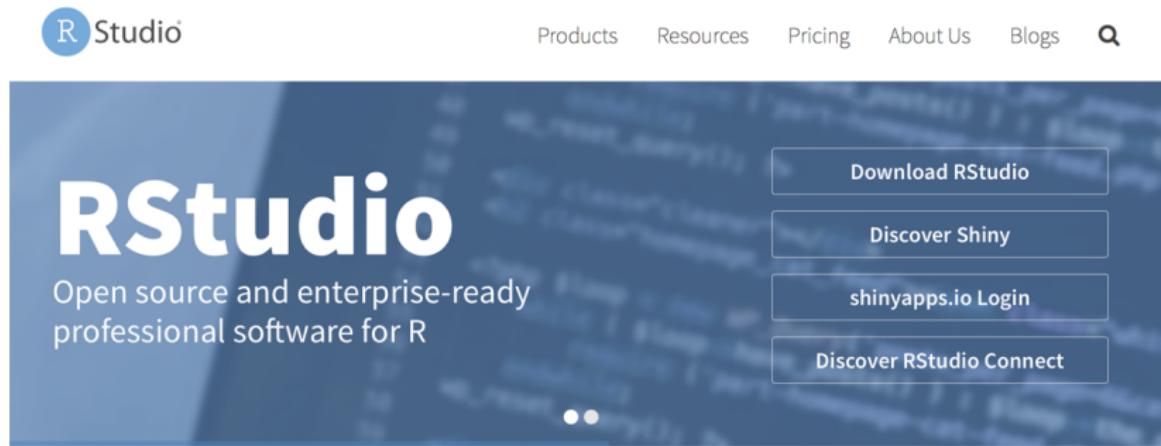


Figure 5: RStudio website main page

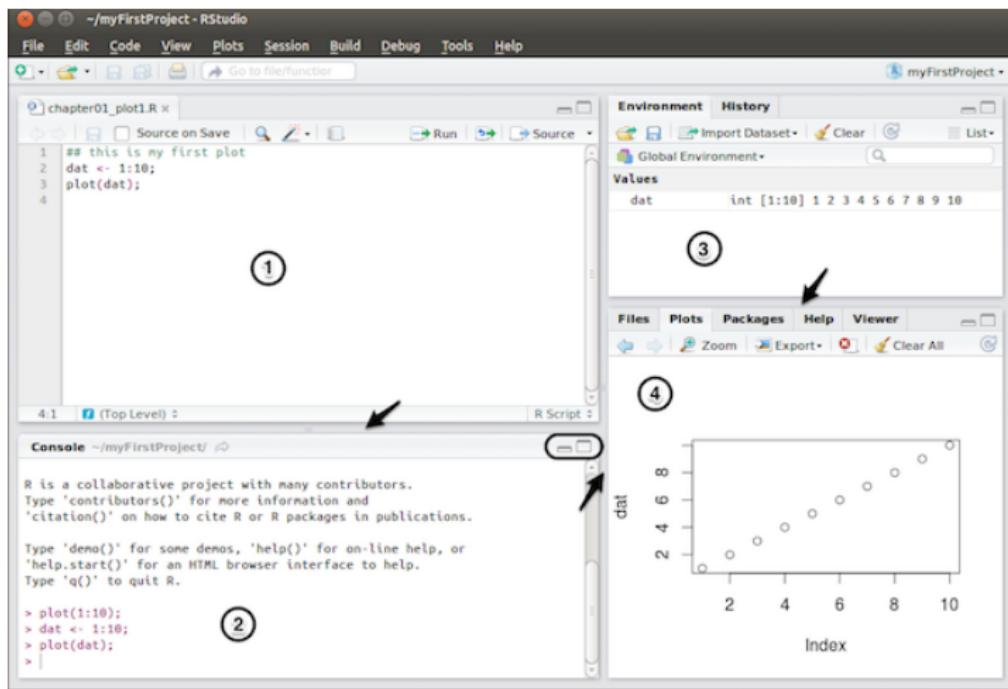
R studio versions

RStudio 有商业和免费版本；也有 server 版

	RStudio Desktop Open Source License	RStudio Desktop Commercial License	RStudio Server Open Source License	RStudio Server Pro Commercial License	RStudio Server Pro + RStudio Connect Commercial License
	FREE DOWNLOAD Learn More	\$995 per year BUY Learn More	FREE DOWNLOAD Learn More	\$9,995 per year DOWNLOAD Learn More	\$29,995 per year TALK Learn More
Integrated Tools for R	●	●	●	●	●
Priority Support		●		●	●
Access via Web Browser			●	●	●
Enterprise Security				●	●
Project Sharing				●	●

R studio, cont.

RStudio 运行时的界面如下图所示，除了顶部的菜单栏工具栏之外，主界面还包括 4 个子窗口：



R studio, cont.

1. 代码编辑器

- 具有代码编辑、语法高亮、代码和变量提示、代码错误检查等功能
- 选中并向 R 控制台（窗口 2）发送并运行代码。用快捷键 Ctrl+Enter (MacOS 下是 Cmd+Enter) 进行代码发送。没有代码选中时，发送光标所在行的代码
- 可同时打开编辑多个文件
- 除 R 代码外，还支持 C++、R MarkDown、HTML 等其它文件的编辑
- 也可用于显示数据

2. R console

- 可在此直接输入各种命令并查看运行结果。支持代码提示

3. 变量列表及代码运行的历史记录

R studio, cont.

4. 其它窗口

- 当前工作目录下的文件列表
- 作图结果
- 可用和已安装的扩展包：在这里可以直接安装新的和升级已有的扩展包
- 帮助

注意，子窗口之间可以通过快捷键 $Ctrl +$ 子窗口编号进行切换。如 $Ctrl + 1$ 可以切换到代码编辑子窗口， $Ctrl + 2$ 则切换到 R 控制台。

其它特点

- 创建、管理 projects

R studio 特点详解

代码提示/自动完成

子窗口 1 和 2 都提供有代码提示功能，即：用户输入 3 个字母时，RStudio 会列出所有前 3 个字母相同的变量或函数名供用户选择；用户可通过键盘的上下键选择，然后用 Enter（回车）选定，非常方便。变量或函数名前面的小图标表示了它们的类型；如果当前高亮的是函数，RStudio 还会显示其部分帮助内容。

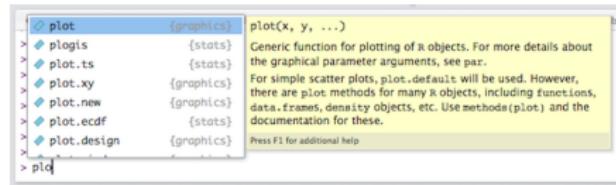


Figure 7: R studio code autocomplete

R studio 特点详解, cont.

查看变量内容

子窗口 3 内会列出所有当前使用的变量、变量的类型以及大小，如下图：

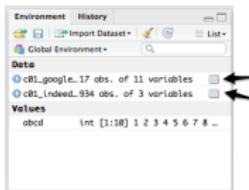


Figure 8: loaded variables

有些简单变量，如数组，RStudio 会直接显示其部分值；对于复杂一些的变量，比如 data.frame（类似于二维表格），则可以点击变量名前边的小三角标识展开其内容。当变量的最右侧出现小网格状图标时（如上图箭头所指位置），点击它们后可以在子窗口 2 内察看。

R studio 特点详解, cont.

导出作图并选择导出格式

RStudio 的第 4 子窗口里集中了许多有用的功能，组织在不同的'Tab'（标签）内。比如作图（plots），不仅可以察看画图的结果，还可以导出当前图像至硬盘，或拷贝至剪贴板；如下图所示。支持导出格式有 png 和 pdf。

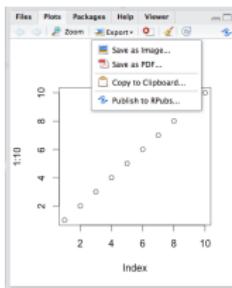


Figure 9: export active plot to various graphical formats

R studio 特点详解, cont.

查看已安装的包

通过第 4 子窗口的“包”(Packages)标签内的工具，用户可以很方便的查看已安装的包：

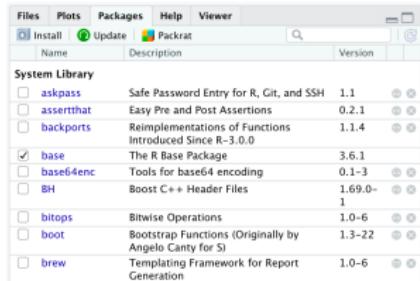


Figure 10: Check installed packages

install new package(s)

同样通过第 4 子窗口的“包”（Packages）标签内的工具，安装新的包：

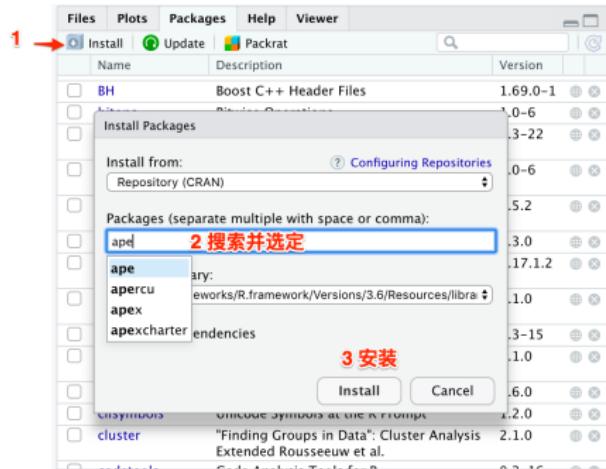


Figure 11: install new package

Packages needed for this study

大部分包都由 RStudio 公司提供；包括：ggplot2, tidyverse, readr, stringr 等。可以用以下命令一次性安装。不过，为方便读者直接从后面章节阅读，在每一次使用新包时，我们会再次进行提示安装方法。

```
install.packages( c("ggplot2", "tidyverse", "readr", "stringr") );
```

也可以单独安装：

```
install.packages( "ggplot2" ); # 安装作图用的 ggplot2  
install.packages( "tidyverse" ); # 数据处理用等
```

第一次运行命令 `install.packages()` 时，系统会提示选择镜像网站；请选择地理位置上距你最近的镜像（比如中国）。

install packages, cont.

You can also choose your CRAN mirror manually (recommended when installing takes a long time):

```
chooseCRANmirror();
```

```
> chooseCRANmirror()  
Secure CRAN mirrors  
  
1: 0-Cloud [https] 2: Algeria [https] 3: Australia (Canberra) [https]  
4: Australia (Melbourne 1) [https] 5: Australia (Melbourne 2) [https] 6: Australia (Perth) [https]  
7: Austria [https] 8: Belgium (Ghent) [https] 9: Brazil (PR) [https]  
10: Brazil (RJ) [https] 11: Brazil (SP 1) [https] 12: Brazil (SP 2) [https]  
13: Bulgaria [https] 14: Chile 1 [https] 15: Chile 2 [https]  
16: China (Hong Kong) [https] 17: China (Guangzhou) [https] 18: China (Lanzhou) [https] ←  
19: China (Shanghai 1) [https] 20: China (Shanghai 2) [https] 21: Colombia (Cali) [https]
```

Figure 12: Choose CRAN mirror

You can also use `chooseBioCmirror()`; to choose mirror for BioConductor packages.

Packages needed for this study, cont.

实际上，以上包属于一个 meta-package，我们只需要安装它就可以了：

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

它是以下包的集合，都由 <https://www.tidyverse.org> 开发：

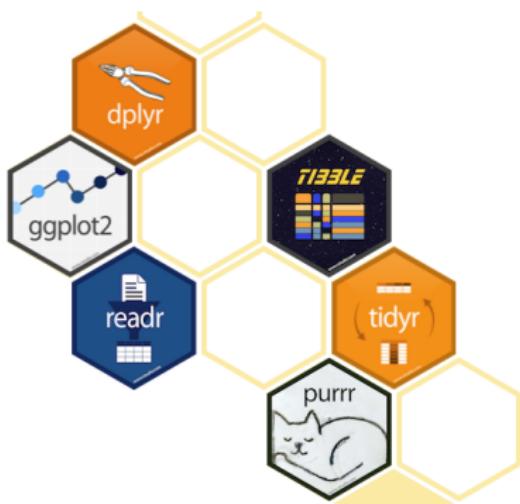


Figure 13: tidyverse: a mega package

R studio server

特点：

- 在服务器上安装，使用服务器的强大计算资源
- 通过网页登录，使用服务器帐号密码（方便，安全）
- 一直运行

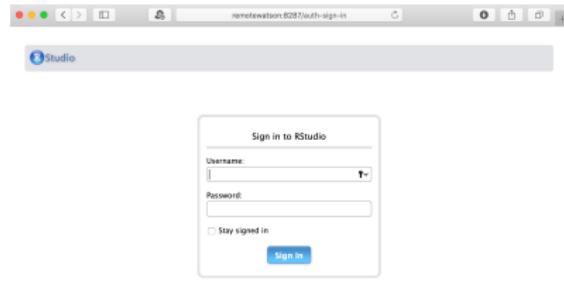


Figure 14: RStudio server web login form (w/ linux account)

R studio server, cont.

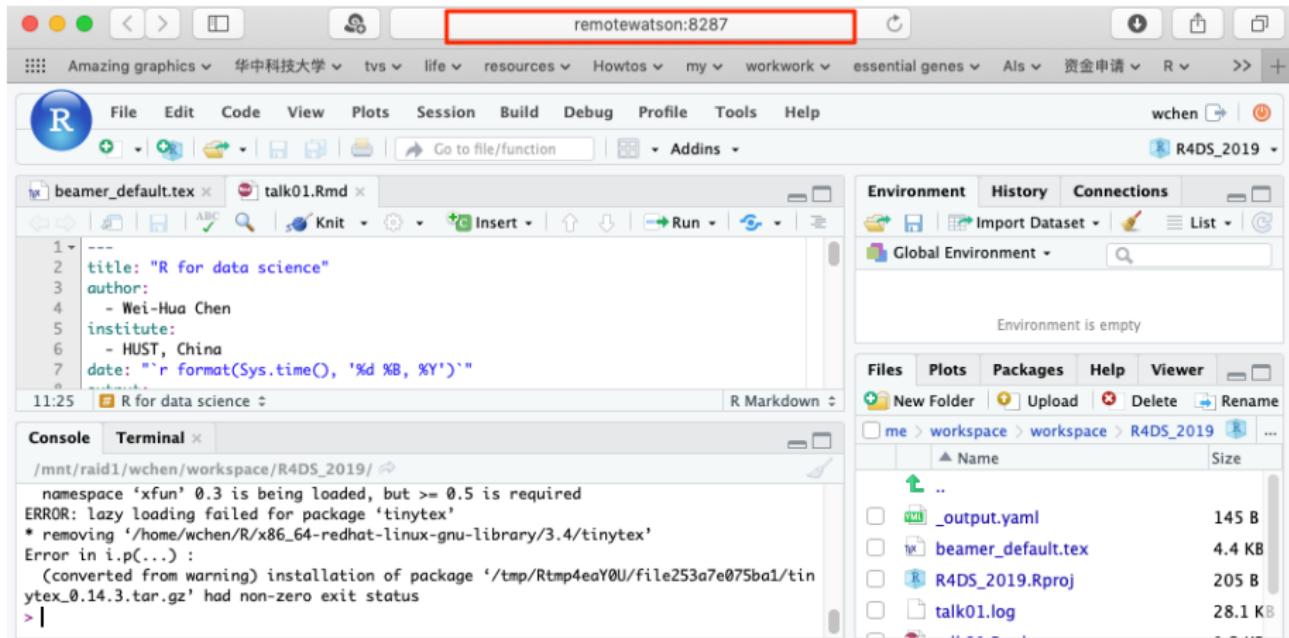


Figure 15: RStudio server 界面（通过浏览器）

R studio packages for data science

all part of (业界良心) tidyverse

- dplyr: 强大且方便的数据处理
- tydvr: 数据转换工具
- readr: 方便的文件 IO
- stringr: 文本处理
- Tibble: 代替 data.frame 的下一代数据存储格式
- purr: (暂时还未用到的包 ~~)

R studio packages for data visualisation

tidyverse

- ggplot2: 专业好用（但学习曲线很陡）的画图工具
 - <http://ggplot2.tidyverse.org>
 - gallery: <http://www.ggplot2-exts.org/gallery/>

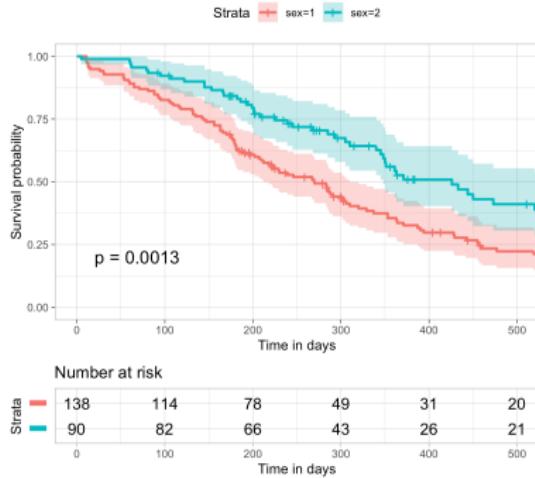
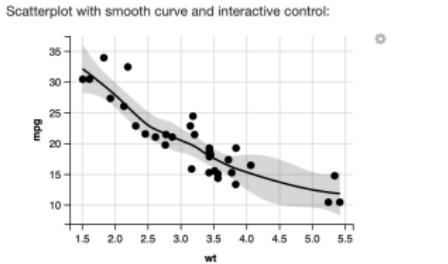
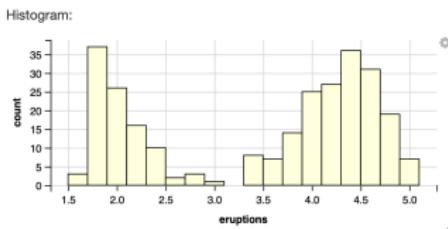


Figure 16: survminer <https://rpkgs.datanovia.com/survminer/index.html>

RStudio packages for data visualisation, cont.

ggvis (currently ver0.4): <http://ggvis.rstudio.com>

- from the **ggplot2** team
- create interactive graphics in RStudio and web browser
- top 50 ggplot2 visualisations



RStudio packages for data visualisation, cont.

Shiny: <http://shiny.rstudio.com/gallery/>

- build professional, interactive visualizations
 - equipped with popular web widgets
 - can be deployed as independent websites

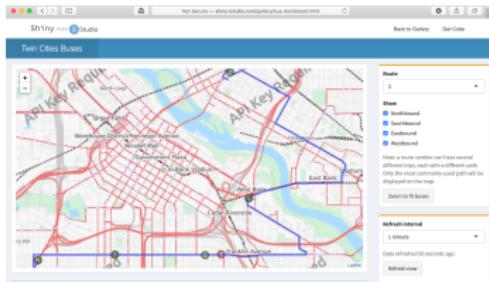


Figure 18: Shiny website example:

<http://shiny.rstudio.com/gallery/bus-dashboard.html>

RStudio packages for data visualisation, cont.

other packages

- rmarkdown : create professional documents
- knitr: convert rmarkdown to pdf, html and more ...

小结

- R 语言历史
- 没有最好的语言，只有最合适的语言
- R 的特色与特长
- IDE、扩展包与资源站点

section 3: dplyr, efficient data wrangler

dplyr 安裝

```
# The easiest way to get dplyr is to install the whole tidyverse:  
install.packages("tidyverse")  
  
# Alternatively, install just dplyr:  
install.packages("dplyr")
```

Development version

```
# install.packages("devtools")  
devtools::install_github("tidyverse/dplyr")
```

Get the cheatsheet at [here](#)

an example of dplyr

get the data ready

```
mouse.tibble <- read_delim( file = "data/talk04/mouse_genes_biomart_sep2018.txt",
                           delim = "\t", quote = "" );
```



```
## Rows: 138532 Columns: 6
## -- Column specification --
## Delimiter: "\t"
## chr (5): Gene stable ID, Transcript stable ID, Protein stable ID, Transcript...
## dbl (1): Transcript length (including UTRs and CDS)
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

查看 mouse.tibble 的内容

```
( ttype.stats <- mouse.tibble %>% count(`Transcript type`) %>% arrange(-n) );  
  
## # A tibble: 48 x 2  
##   `Transcript type`     n  
##   <chr>                 <int>  
## 1 protein_coding        58384  
## 2 retained_intron       21021  
## 3 processed_transcript  15572  
## 4 processed_pseudogene   9425  
## 5 lincRNA                8557  
## 6 nonsense-mediated_decay 6755  
## 7 antisense              4289  
## 8 TEC                     3265  
## 9 unprocessed_pseudogene  2650  
## 10 miRNA                  2265  
## # ... with 38 more rows
```

查看 mouse.tibble 的内容, cont.

```
( chr.stats <- mouse.tibble %>% count(`Chromosome/scaffold name`) %>% arrange(-n) );  
  
## # A tibble: 117 x 2  
##   `Chromosome/scaffold name`     n  
##   <chr>                      <int>  
## 1 7                          12344  
## 2 2                          10877  
## 3 5                          8955  
## 4 11                         8673  
## 5 1                          8553  
## 6 9                          8030  
## 7 6                          7845  
## 8 4                          7573  
## 9 3                          6938  
## 10 10                         6568  
## # ... with 107 more rows
```

分析任务

- ① 将染色体限制在常染色体和 XY 上（去掉未组装的小片段）；处理行
- ② 将基因类型限制在 protein_coding, miRNA 和 lincRNA 这三种；处理行
- ③ 统计每条染色体上不同类型基因（protein_coding, miRNA, lincRNA）的数量
- ④ 按染色体（正）、基因数量（倒）进行排序

用 dplyr 实现

```

dat <- mouse.tibble %>%
  ## 1.

  filter( `Chromosome/scaffold name` %in% c( 1:19, "X", "Y" ) ) %>%
  ## 2.
  filter( `Transcript type` %in% c( "protein_coding", "miRNA", "lincRNA" ) ) %>%

  ## change column name ...
  select( CHR = `Chromosome/scaffold name`, TYPE = `Transcript type`,
         GENE_ID = `Gene stable ID`,
         GENE_LEN = `Transcript length (including UTRs and CDS)` ) %>%

  ## 3.
  group_by( CHR, TYPE ) %>%
  summarise( count = n_distinct( GENE_ID ), mean_len = mean( GENE_LEN ) ) %>%

  ## 4.
  arrange( CHR , desc( count ) );

## `summarise()` has grouped output by 'CHR'. You can override using the `.groups`  

## argument.

```

检查运行结果

```
## # A tibble: 15 x 4
## # Groups:   CHR [5]
##   CHR     TYPE       count  mean_len
##   <chr>   <chr>     <int>    <dbl>
## 1 1      protein_coding 1200    2700.
## 2 1      lincRNA        347     1207.
## 3 1      miRNA         128     98.0
## 4 10     protein_coding 1020    2408.
## 5 10     lincRNA        398     1220.
## 6 10     miRNA          91      89.9
## 7 11     protein_coding 1640    2432.
## 8 11     lincRNA        189     1134.
## 9 11     miRNA          137     87.5
## 10 12    protein_coding  644    2524.
## 11 12    lincRNA        327     1277.
## 12 12    miRNA          146     86.2
## 13 13    protein_coding  831    2380.
## 14 13    lincRNA        428     1251.
## 15 13    miRNA          97      106.
```

这种显示格式通常被称为：**长数据格式!! 又称为数据扁平化**

数据扁平化的优点？

- ① 便于用 dplyr 或 tapply 等进行计算；
- ② 更灵活，用于保存稀疏数据

适合扁平化的数据举例

成绩单

```
library(tidyverse);
grades <- read_tsv( file = "data/talk05/grades.txt" );
head(grades, n=20);
```

```
## # A tibble: 9 x 3
##   name      course    grade
##   <chr>     <chr>     <dbl>
## 1 Zhi Liu   Microbiology 100
## 2 Zhi Liu   English      50
## 3 Zhi Liu   Chinese      69
## 4 Weihua Chen Microbiology 89
## 5 Weihua Chen English      99
## 6 Weihua Chen Bioinformatics 99
## 7 Kang Ning  Bioinformatics 100
## 8 Kang Ning  Chinese      20
## 9 Kang Ning  Chemistry     76
```

灵活性：

- 应对不同学生选择不同课程的情况
- 可随时增加新的课程

长数据变宽

```
grades2 <- grades %>% spread( course, grade );  
grades2;
```

```
## # A tibble: 3 x 6  
##   name      Bioinformatics Chemistry Chinese English Microbiology  
##   <chr>        <dbl>       <dbl>     <dbl>    <dbl>       <dbl>  
## 1 Kang Ning      100        76       20     NA        NA  
## 2 Weihua Chen     99        NA       NA      99        89  
## 3 Zhi Liu        NA        NA       69      50       100
```

可以想像，如果以此为输入，用 R 计算每个人的平均成绩、不及格门数、总学分，将会是很繁琐的一件事（但对其它工具（如 Excel）可能会比较简单）

use gather & dplyr functions

Question: 1. 每个人平均成绩是多少 ? 2. 哪个人的平均成绩最高 ?

```
res <-  
  grades %>%  
  group_by(name) %>%  
  summarise( avg_grades = mean( grade ), courses_count = n() ) %>%  
  arrange( -avg_grades );
```

```
## 显示最终结果  
res;
```

```
## # A tibble: 3 x 3  
##   name      avg_grades courses_count  
##   <chr>        <dbl>        <int>  
## 1 Weihua Chen     95.7         3  
## 2 Zhi Liu        73           3  
## 3 Kang Ning      65.3         3
```

use gather & dplyr functions

问题：每个人 的 最强科目 是什么 ??

```
res2 <-  
  grades %>%  
  arrange( -grade ) %>%  
  group_by(name) %>%  
  summarise( best_course = first( course ),  
            best_grade = first( grade ),  
            avg_grades = mean( grade ),  
            courses = n() ) %>%  
  arrange( -avg_grades );
```

显示最终结果
res;

```
## # A tibble: 3 x 3  
##   name      avg_grades courses_count  
##   <chr>        <dbl>       <int>  
## 1 Weihua Chen     95.7         3  
## 2 Zhi Liu        73           3  
## 3 Kang Ning      65.3         3
```

更多练习，使用 starwars tibble

```
head(starwars);
```

```
## # A tibble: 6 x 14
##   name      height  mass hair_~1 skin_~2 eye_c~3 birth~4 sex   gender homew~5
##   <chr>     <int> <dbl> <chr>   <chr>   <chr>   <dbl> <chr>   <chr>   <chr>
## 1 Luke Skywalker 172    77 blond   fair    blue     19 male   masculin Tatooine
## 2 C-3PO          167    75 <NA>    gold    yellow  112 none   masculin Tatooine
## 3 R2-D2          96     32 <NA>    white   ~ red    33 none   masculin Naboo
## 4 Darth Vader   202   136 none   white   yellow  41.9 male   masculin Tatooine
## 5 Leia Organa   150    49 brown   light   brown   19 female feminin Alderaan
## 6 Owen Lars     178   120 brown,~ light   blue    52 male   masculin Tatooine
## # ... with 4 more variables: species <chr>, films <list>, vehicles <list>,
## #   starships <list>, and abbreviated variable names 1: hair_color,
## #   2: skin_color, 3: eye_color, 4: birth_year, 5: homeworld
```

note 包含 87 行 13 列，星战部分人物的信息，包括身高、体重、肤色等

用 ?starwars 获取更多帮助

dplyr::filter - 行操作

任务：从星战中挑选金发碧眼的人物

```
starwars %>% select( name, ends_with("color"), gender, species ) %>%
  filter( hair_color == "blond" & eye_color == "blue" );
```

```
## # A tibble: 3 x 6
##   name      hair_color skin_color eye_color gender   species
##   <chr>     <chr>     <chr>     <chr>     <chr>     <chr>
## 1 Luke Skywalker blond    fair      blue      masculine Human
## 2 Anakin Skywalker blond    fair      blue      masculine Human
## 3 Finis Valorum  blond    fair      blue      masculine Human
```

小结

- dplyr 等扩展包，提供了基于行，列，等规则数据的快速处理
- 非常高效，语法统一，可读性强
- 灵活性尚可
- 学习曲线尚可

section 4: ggplot2, elegant graphics using R

ggplot2 的四个基本组成

① 图层 (layers)

- `geom_< 图层名 >`

② scale: 控制数据至美学属性的 mapping

- `scale_< 属性 mapping 方式 >, e.g. scale_color_identity()`

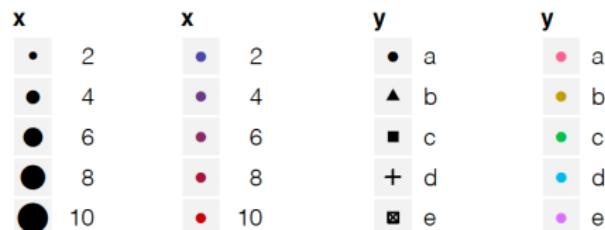


Figure 19: 数据的 4 种 scale 方法

ggplot2 的 scale

- `scale_color_...`
- `scale_shape_...`
- `scale_size_...`
- `scale_fill_...`

与坐标系统联动的函数

- `scale_x_log()`
- `scale_y_log()`

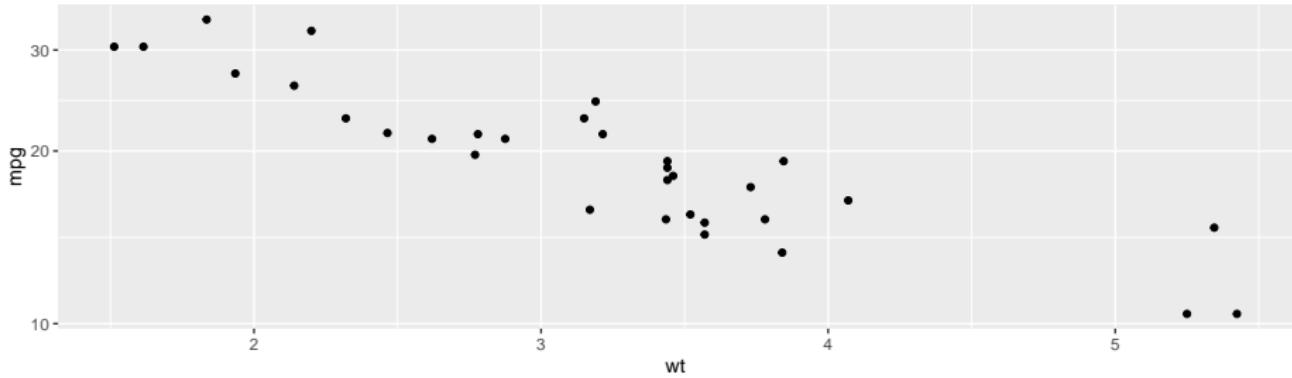
更多内容可以见《ggplot2: elegant graphics for data analysis》一书的第 6 章。

ggplot2 要素 3: 坐标系统

- 正常
- log-transform

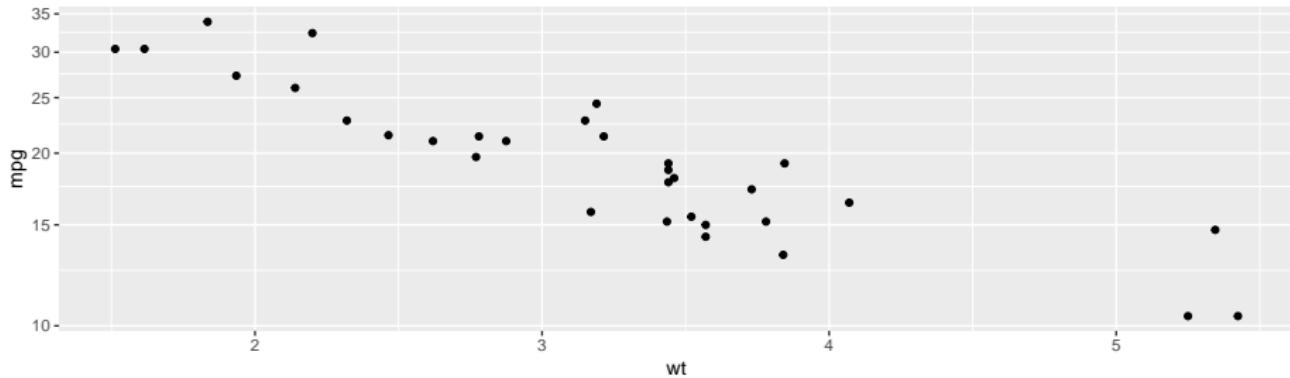
示例：

```
ggplot(mtcars, aes( wt , mpg)) + geom_point() +  
  scale_y_log10()
```



ggplot2 要素 3: 坐标系统, cont.

```
ggplot(mtcars, aes( wt , mpg)) + geom_point() +
  coord_trans( y = "log10" );
```



- * limx, limy: 限制 xy 的显示范围

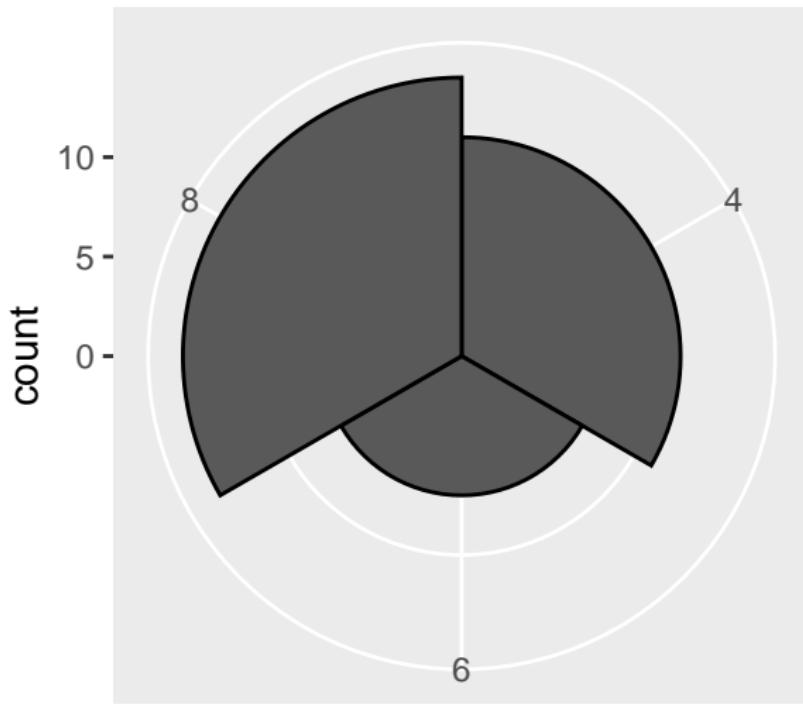
```
## ggplot2 要素 3: 坐标系统, cont.
```

其它函数

- * `coord_flip()` : x, y 轴互换; 竖 bar 变横 bar;
- * `coord_polar()` :

ggplot2 要素 3: 坐标系统, cont.

```
plot1 + coord_polar();
```

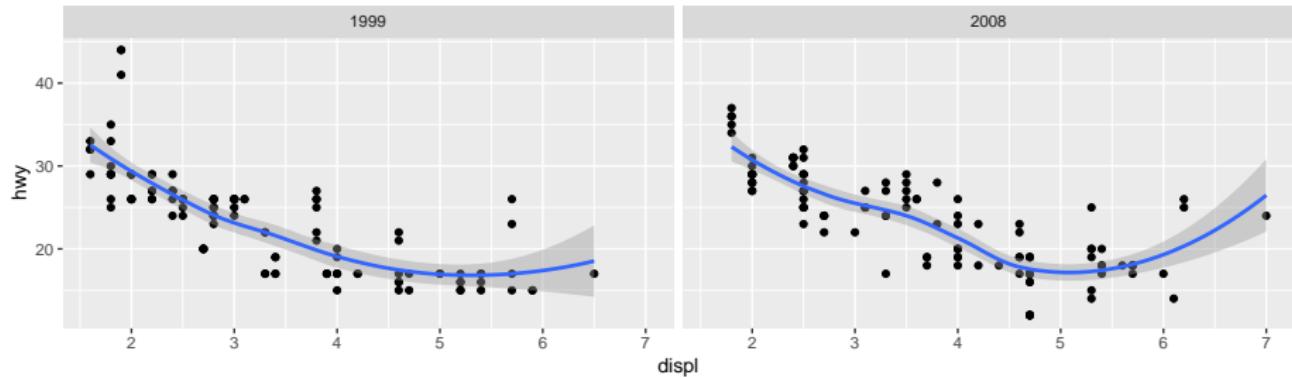


ggplot2 要素 4: faceting ...

```
qplot(displ, hwy, data=mpg, facets = . ~ year) + geom_smooth();
```

Warning: `qplot()` was deprecated in ggplot2 3.4.0.

`geom_smooth()` using method = 'loess' and formula = 'y ~ x'



ggsci: palette for scientific journals!!!

install

```
install.packages("ggsci"); # Install ggsci from CRAN:  
devtools::install_github("nancxstats/ggsci"); # or from github
```

contents

scale_color_<journal> 和 scale_fill_<journal> functions and color palettes

supported journals

- NPG scale_color_npg(), scale_fill_npg()
- AAAS, NEJM, Lancet, JAMA ...

ggsci 举例

```

library("ggsci")
library("ggplot2")
library("gridExtra")
data("diamonds")
p1 <- ggplot(
  subset(diamonds, carat >= 2.2),
  aes(x = table, y = price, colour = cut)
) +
  geom_point(alpha = 0.7) +
  geom_smooth(method = "loess", alpha = 0.05, size = 1, span = 1) +
  theme_bw() + labs( tag = "A" )

p2 <- ggplot(
  subset(diamonds, carat > 2.2 & depth > 55 & depth < 70),
  aes(x = depth, fill = cut)
) +
  geom_histogram(colour = "black", binwidth = 1, position = "dodge") +
  theme_bw() + labs( tag = "B" )

```

要点

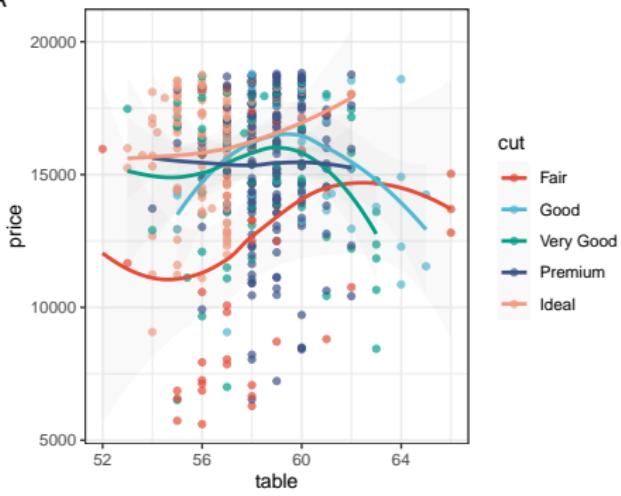
- library(gridExtra)

ggsci 结果, Nature Style !!

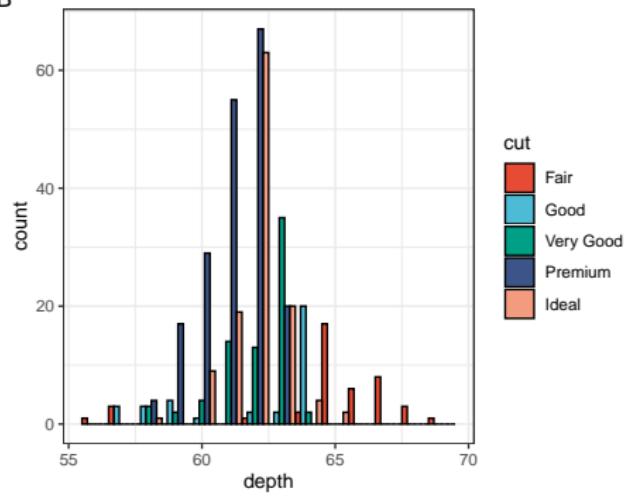
```
p1_npg <- p1 + scale_color_npg()
p2_npg <- p2 + scale_fill_npg()
grid.arrange(p1_npg, p2_npg, ncol = 2)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

A



B



combine multiple plots

Useful packages:

- gridExtra
- cowplot
- grid
- lattice

install or load packages

```
if (!require("gridExtra")){
  install.packages("gridExtra");
}

if (!require("cowplot")){
  install.packages("cowplot");
}

## Loading required package: cowplot
```

```
library( cowplot );
library( gridExtra );
```

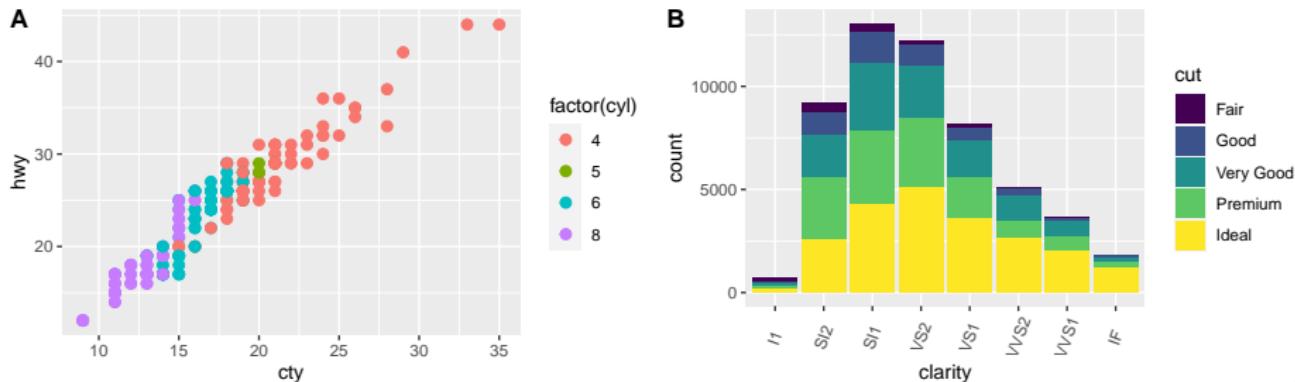
arranging multiple graphs using cowplot

Prepare two plots

```
sp <- ggplot(mpg, aes(x = cty, y = hwy, colour = factor(cyl)))+
  geom_point(size=2.5)
# Bar plot
bp <- ggplot(diamonds, aes(clarity, fill = cut)) +
  geom_bar() +
  theme(axis.text.x = element_text(angle=70, vjust=0.5))
```

Combine the two plots (the scatter plot and the bar plot):

```
cowplot::plot_grid(sp, bp, labels=c("A", "B"), ncol = 2, nrow = 1)
```

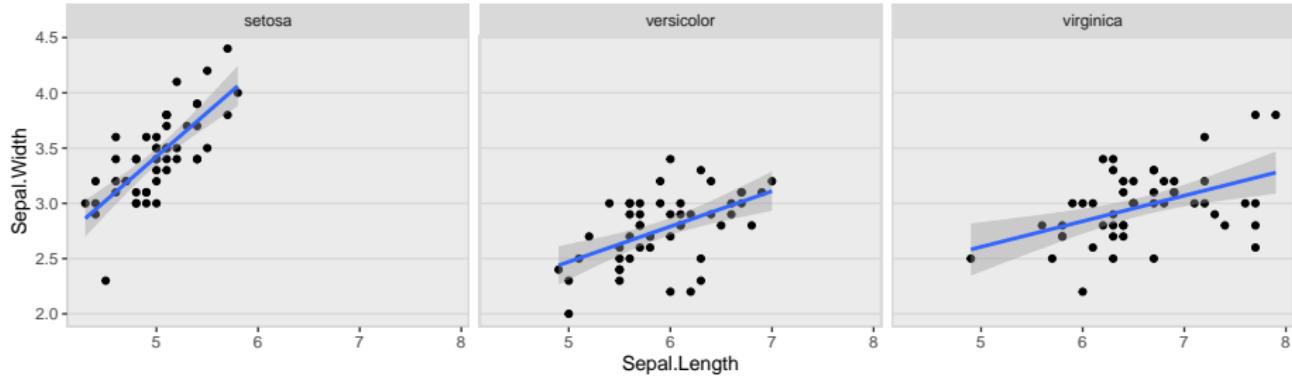


用 draw_plot 调整 graph 的相对大小

先生成一个新的 panel

```
plot.iris <- ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
  geom_point() + facet_grid(. ~ Species) + stat_smooth(method = "lm") +
  background_grid(major = 'y', minor = "none") + # add thin horizontal lines
  panel_border();
plot.iris;

## `geom_smooth()` using formula = 'y ~ x'
```



用 draw_plot 将三个 panel 画在一起

```

plot <-
  ggdraw() +
  draw_plot(plot.iris, x=0, y=.5, width=1, height=.5) +
  draw_plot(sp, 0, 0, .5, .5) +
  draw_plot(bp, .5, 0, .5, .5) +
  draw_plot_label(c("A", "B", "C"), c(0, 0, 0.5), c(1, 0.5, 0.5), size = 15);

## `geom_smooth()` using formula = 'y ~ x'

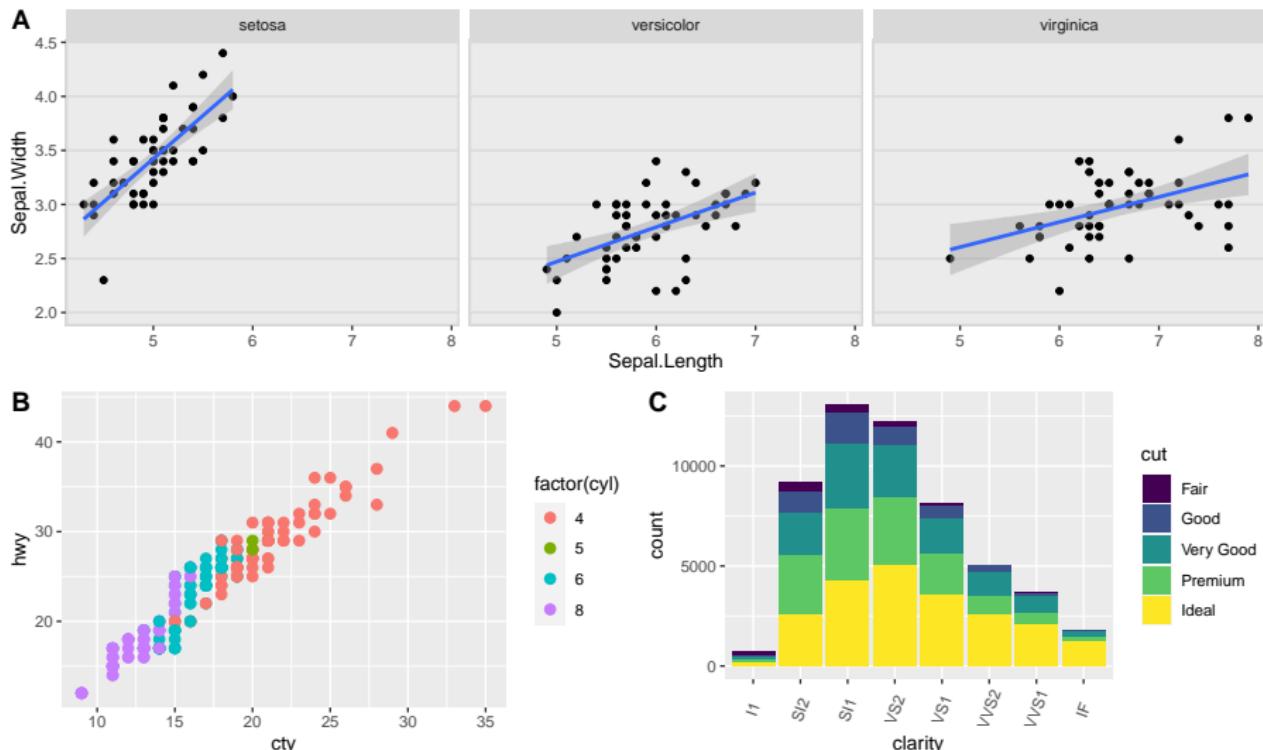
draw_plot(plot, x = 0, y = 0, width = 1, height = 1) 详解:

```

- plot: the plot to place (ggplot2 or a gtable)
- x: The x location of the lower left corner of the plot.
- y: The y location of the lower left corner of the plot.
- width, height: the width and the height of the plot

draw_plot results

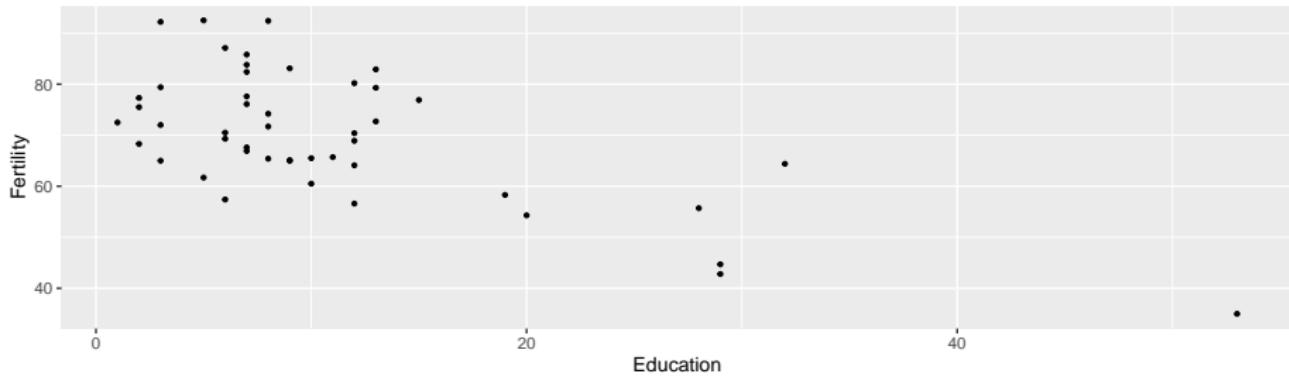
plot



如何在图中加入公式？

显示两组数据间的相关性：

```
## 作图
ggplot( swiss, aes( x = Education, y = Fertility ) ) +
  geom_point( shape = 20 );
```

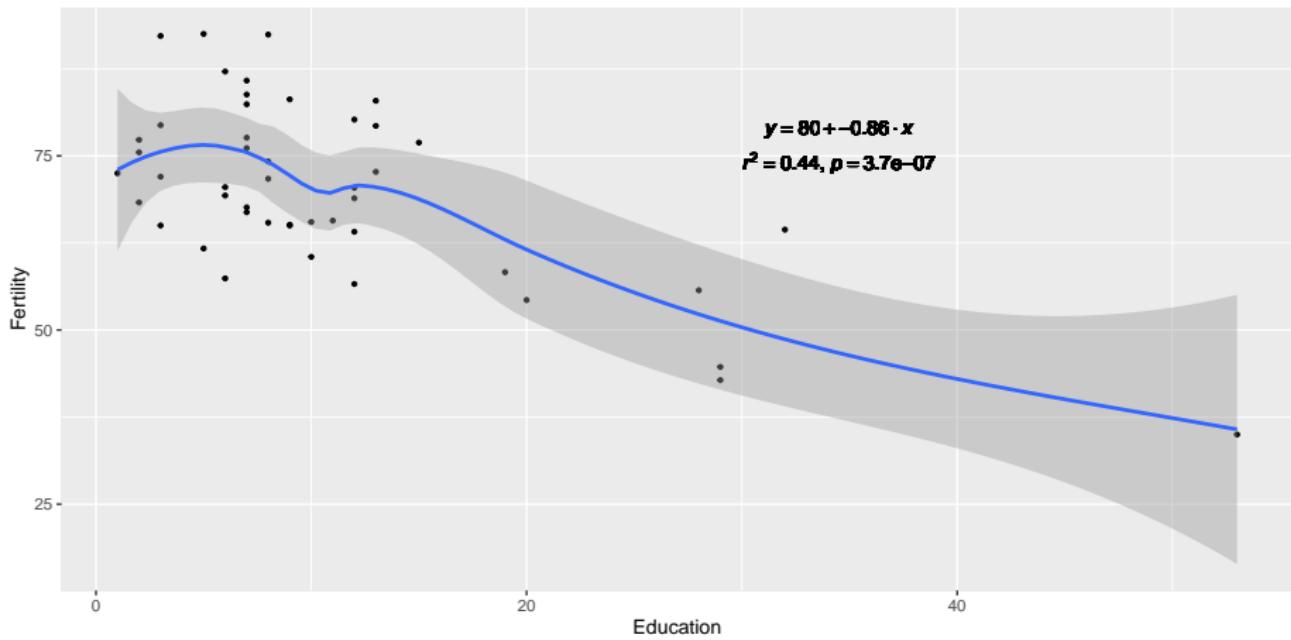


```
## 分析
with( swiss, cor.test( Education, Fertility )$estimate );
```

```
##      cor
## -0.6637889
```

在图中加入公式和统计信息

先展示一下结果



公式详解

`paste(italic(y), " = ", a + b %% italic(x), sep = "")`

$$\underline{y} = 80 + -0.86 \cdot \underline{x}$$

$$\underline{r^2} = 0.44, \underline{p} = 3.7e-07$$

`paste(italic(r)^2, " = ", r2, ", ", italic(p)==pvalue, sep = ""))`

Figure 20: equation explained!

公式详解, cont.

以下代码实现两个任务:

- ① 将两个公式上下放置 `atop(<equation_1> , <equation_2>)`;
- ② 将公式中的某些值替换为数值 `substitute(<equation>, list(...))`

```

## 计算 ...
m = lm(Fertility ~ Education, swiss);
c = cor.test( swiss$Fertility, swiss$Education );

## 生成公式
eq <- substitute( atop( paste( italic(y), " = ", a + b %.% italic(x), sep = ""),
                         paste( italic(r)^2, " = ", r2, ", ", italic(p)==pvalue, sep = "" ) ),
                  list(a = as.vector( format(coef(m)[1], digits = 2) ),
                       b = as.vector( format(coef(m)[2], digits = 2) ),
                       r2 = as.vector( format(summary(m)$r.squared, digits = 2) ),
                       pvalue = as.vector( format( c$p.value , digits = 2) ) )
                );
## 用 as.expression 对公式进行转化
eq <- as.character(as.expression(eq));

```

完整代码

```

## 计算 ...
m = lm(Fertility ~ Education, swiss);
c = cor.test( swiss$Fertility, swiss$Education );

## 生成公式
eq <- substitute( atop( paste( italic(y), " = ", a + b %.% italic(x), sep = ""),
                         paste( italic(r)^2, " = ", r2, ", ", italic(p)==pvalue, sep = "" ) ),
                     list(a = as.vector( format(coef(m)[1], digits = 2) ),
                          b = as.vector( format(coef(m)[2], digits = 2) ),
                          r2 = as.vector( format(summary(m)$r.squared, digits = 2) ),
                          pvalue = as.vector( format( c$p.value , digits = 2) ) )
                   );
## 用 as.expression 对公式进行转化 !!!! 
eq <- as.character(as.expression(eq));

## 作图, 三个图层; 特别是 geom_text 使用自己的 data 和 aes ...
ggplot(swiss, aes( x = Education, y = Fertility ) ) +
  geom_point( shape = 20 ) +
  geom_smooth( se = T ) + ## smooth line ...
  geom_text( data = NULL,
            aes( x = 30, y = 80, label= eq, hjust = 0, vjust = 1), ## hjust, vjust ???
            size = 4, parse = TRUE, inherit.aes=FALSE); ## 注意: parse = TRUE !!!

```

总结, 本节内容

ggplot2 基础

- 优缺点
- 用法
- 基本组成

ggplot2 进阶

- 颜色和色板
- 复杂 layout 的实现
- 公式
- ggplot2 的数据统计逻辑

更多阅读

- Ggplot2: Elegant Graphics for Data Analysis, Book by Hadley Wickham
- ggplot2 gallery provided by RStudio on Github

小结

- ① ggplot2 博大精深，需要一门课去讲
- ② 上手容易，精通难
- ③ 太多记忆点
- ④ 本节内容只涉及了基础中的基础，更多内容，包括进阶技巧和生信相关的扩展包，更多的需要同学们自行探索
- ⑤ 遇到不会的图，先百度/Google，找包和代码

section 5: machine learning

机器学习算法归纳

本节内容来自 CSDN Deep Learning 小舟同学的《机器学习：常见的机器学习算法归纳》一文。

机器学习可分为以下几类

- 1) 回归算法
- 2) 基于实例的算法
- 3) 决策树学习
- 4) 贝叶斯方法
- 5) 基于核的算法
- 6) 聚类算法
- 7) 降低维度算法
- 8) 关联规则学习
- 9) 集成算法
- 10) 人工神经网络

随机森林 – Random forest

本文大部分内容取自“easyai.tech”网站的《随机森林 – Random forest》一文，有修改。

随机森林是一种由决策树构成的集成算法，适用于小样本数据，在很多情况下都能有不错的表现。

随机森林属于 **集成学习** 中的 Bagging (Bootstrap AGgregation 的简称) 方法。

决策树 - decision tree



决策树是一种很简单的算法，他的解释性强，也符合人类的直观思维。这是一种基于 if-then-else 规则的有监督学习算法，上面的图片可以直观的表达决策树的逻辑。

随机森林 – Random Forest | RF

随机森林 – Random Forest



easyai

随机森林是由很多决策树构成的，不同决策树之间没有关联。

当我们进行分类任务时，新的输入样本进入，就让森林中的每一棵决策树分别进行判断和分类，每个决策树会得到一个自己的分类结果，决策树的分类结果中哪一个分类最多，那么随机森林就会把这个结果当做最终的结果

随机森林的 4 个应用方向

随机森林的应用方向



分类



回归



聚类



异常检测

easyai

随机森林实例，数据

使用 Liver cirrhosis 肝硬化数据：

```
summary_tbl <- tibble(
  `Project ID` = c("PRJEB6337-cohort1", "PRJEB6337-cohort2", "PRJNA471972"),
  ID = c("Qin-cohort1", "Qin-cohort2", "Iebba-16S"),
  `Data type` = c("mNGS", "mNGS", "16S"),
  `Nr. controls` = c(83, 31, 14),
  `Nr. cases` = c(98, 25, 23),
  Country = c("China", "China", "Italy")
)
summary_tbl;
```

## # A tibble: 3 x 6	## `Project ID`	ID	## Data type`	## Nr. controls`	## Nr. cases`	Country
	## <chr>	<chr>	<chr>	<dbl>	<dbl>	<chr>
## 1	PRJEB6337-cohort1	Qin-cohort1	mNGS	83	98	China
## 2	PRJEB6337-cohort2	Qin-cohort2	mNGS	31	25	China
## 3	PRJNA471972	Iebba-16S	16S	14	23	Italy

数据和代码目录

“data/talk12/”

文件：

- rf.function.R：自写的函数，将建模相关代码进行了封装；
- working.R：分析代码，利用上面函数对数据进行分析；
- 子目录 data/：文件夹，包括各种所需数据；

装入数据

- 丰度信息
- 样本分组（对照和疾病）

```
## 发现集，用于建模的数据； species 水平
discovery.featdata <-
  read.csv("./data/talk12/data/PRJEB6337.LC.s.discovery.txt",
           sep = "\t", header = T, row.names = 1);

discovery.metadata <-
  read.csv("./data/talk12/data/PRJEB6337.LC.s.discovery.metadata.txt",
           sep = "\t", row.names = 1);
```

用 View() 函数查看数据内容

用 rowSums(discovery.featdata) 查看细菌的丰度数据

装入封装好的函数

注意，需要以下包：

tidyverse, ROCR, randomForestSRC, pROC, caret, progress, tictoc,
doSNOW, ggplot2

```
source("./data/talk12/rf.function.R");
```

封装的函数及其功能

- rf.setdata() : 准备数据
- rf.train(): 用指定的数据构建模型
- rf.evaluate(): 计算模型交叉验证结果
- rf.external.validation(): 外部验证（即：被验证数据的分类结果已知）。
- rf.predict(): 预测（即：未知分类结果的情况下）

1. 准备数据

`rf.setdata()` 有四个参数，分别是：

- `feat.data`：丰度数据 (feature data); `data.frame`, 行为样本, 列为物种 (细菌); 内容为数字; 行名为样本名
- `meta.data`：样本表型数据 (疾病健康); `data.frame`; 行名为样本名; 样本应分为两类;
- `grouping`：指定 `meta.data` 中包含样本分组列的列名;
- `control`：样本分组列中对照组的名称，比如 `Healthy`, `CTR` 等;

返回：一个 `list`

rf.setdata 的使用

```

## 先做过滤; 去掉失败的样本
qc.fail.run<- read.csv("./data/talk12/data/qc.fail.run.csv");

dis.meta <- discovery.metadata[!rownames(discovery.metadata) %in% qc.fail.run$run.id,];
dis.feat <- discovery.featdata[!rownames(discovery.featdata) %in% qc.fail.run$run.id,];

## check data
table(dis.meta$Group);

## 
## Healthy      LC
##    103     124

## set data
dis.mod <- rf.setdata( feat.data = dis.feat, meta.data = dis.meta,
                        grouping = "Group", control = "Healthy");

## All look great.
## -----
## In total, the feature data contains 227 samples, with 399 features.
## Case group: 'LC', w/ 124 samples,
## Control group: 'Healthy', w/ 103 samples,
## -----
## 
## Data size: 1.5 Mb
## Set data : 0.003 sec elapsed

```

2. 构建模型 (train)

rf.train() 函数，其参数为：

- rf: list; rf.setdata 的输出结果；包括了所需数据（样本分组、丰度信息）；
- fold: 折；默认为 10；
- resample: 次数；默认为 10；
- parallel: 是否用并行模式；默认为否；
- num.cores: 使用 CPU 数量；默认为 10；parallel = T 时生效；当超过系统的 CPU 数量时，自动变为 cpu - 1；
- class.weights: 默认为 NULL；可改为 class.weights = T；当样本分组不均衡时使用（比如：健康对照人数是疾病组的 5 倍）；

构建模型 (train) , cont.

```
dis.mod <- rf.train( dis.mod, parallel = T, num.cores = 20 );
```

```
## Training models on 227 samples, with 399 features.  
## Case group: LC, control group: Healthy  
##   - running 10 resample and 10 fold modeling ...  
##   - NOT using class weight option of randomForest classifier ...  
##   - running parallel modeling using 7 cpu cores ...  
## |
```

- 使用了 7 个 CPU (指定 20 个, 但系统只有 8 个);
- 进行了 10 次 * 10 折建模, 共得到 100 个模型
- 没有做 class weight 调整, 因为样本分组相对均衡
 - Case group: 'LC', w/ 124 samples,
 - Control group: 'Healthy', w/ 103 samples

K 折 N 次的实现代码

```
sample.split <- caret::createMultiFolds( as.vector( dis.meta$Group ), times = 10, k = 10 );  
## 检查结果 ...  
class(sample.split);  
  
## [1] "list"  
  
length(sample.split);  
  
## [1] 100  
  
str(sample.split$Fold01.Rep01);  
  
## int [1:204] 1 2 3 4 5 6 7 8 9 10 ...
```

注:

- sample.split 包括了被选中样本的行号；被选中的样本为样本集/train data
- 没有被选中的样本即为测试集/test data；

模型构建代码

```
## assemble data --
dat <- dis.feat;
dat$Group <- as.factor( dis.meta[ rownames(dat), "Group" ] ); ## 这一步很重要!!

## 从 list 中随机取一个出来
split <- sample( sample.split, 1 );

## 生成 train 和 test 数据;
train.data <- dat[ split[[1]] , ];
test.data <- dat[ -split[[1]] , ];

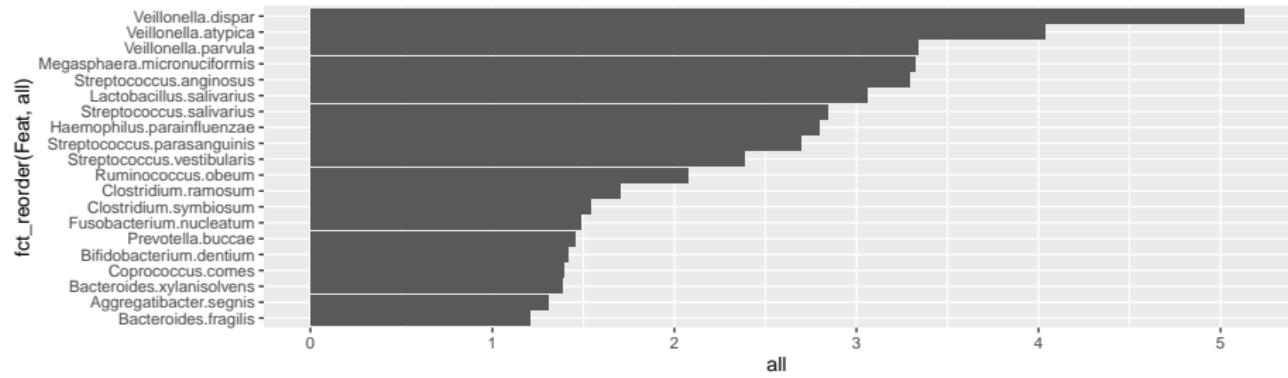
## 建模
model.a <- rfsrc(Group ~ .,train.data, proximity = T, importance = T, ntree = 500);
```

- ntree 指定生成 500 个决策树;

Feature 重要性检验

RandomForest 提供对每一个 feature 重要性的评估，保存在 model.a\$importance data.frame 中。

```
require(tidyverse);
impo <- as.data.frame( model.a$importance * 100 ) %>% arrange( desc(all) ) %>%
  slice_head( n=20 ) %>% rownames_to_column( var = "Feat" ) %>%
  ggplot( aes( x = fct_reorder(Feat, all), y = all ) ) + geom_bar( stat = "identity" ) +
  coord_flip();
impo;
```



预测和结果检验（使用 100 个模型中的一个）

```

## 预测; 结果为 LC 和 Healthy 的可能性;
pred <- predict( model.a, test.data, type = "prob");

## 整理预测结果
res <- as.data.frame( pred$predicted );
res$prediction <- if_else( res$Healthy > res$LC, "Healthy", "LC" );
rownames( res ) <- rownames( test.data );
res$label <- test.data$Group;

## 比较真实和预测值; 得到一个 confusion matrix;
table( res[, c("prediction", "label")] );

##           label
## prediction Healthy LC
##   Healthy      11  2
##     LC          0 10

```

3. 检查建模的整体结果

如上所述，100 个模型就会有 100 个结果；我们接下来把结果总结显示出来；

```
dis.mod.eval <- rf.evaluate( dis.mod );  
  
## Evaluating model performance ...  
##  
## -----  
## Overall performance (AUC): 0.95 (95%CI: 0.92-0.98)  
##  
## Predicted vs. Truth (Group):  
## Control group: Healthy  
## Case group: LC  
##  
##          Predicted  
## Group      Healthy  LC  
##   Healthy      96    7  
##   LC          16 108  
## -----  
## Error rates:  
## Healthy      LC  
##   0.068    0.129  
##  
## Overall error rate: 0.101  
## -----
```

输出的 message

Overall performance (AUC): 0.95 (95%CI: 0.92-0.98)

Predicted vs. Truth (Group):

Control group: Healthy

Case group: LC

Group	Predicted	
	Healthy	LC
Healthy	97	6
LC	17	107

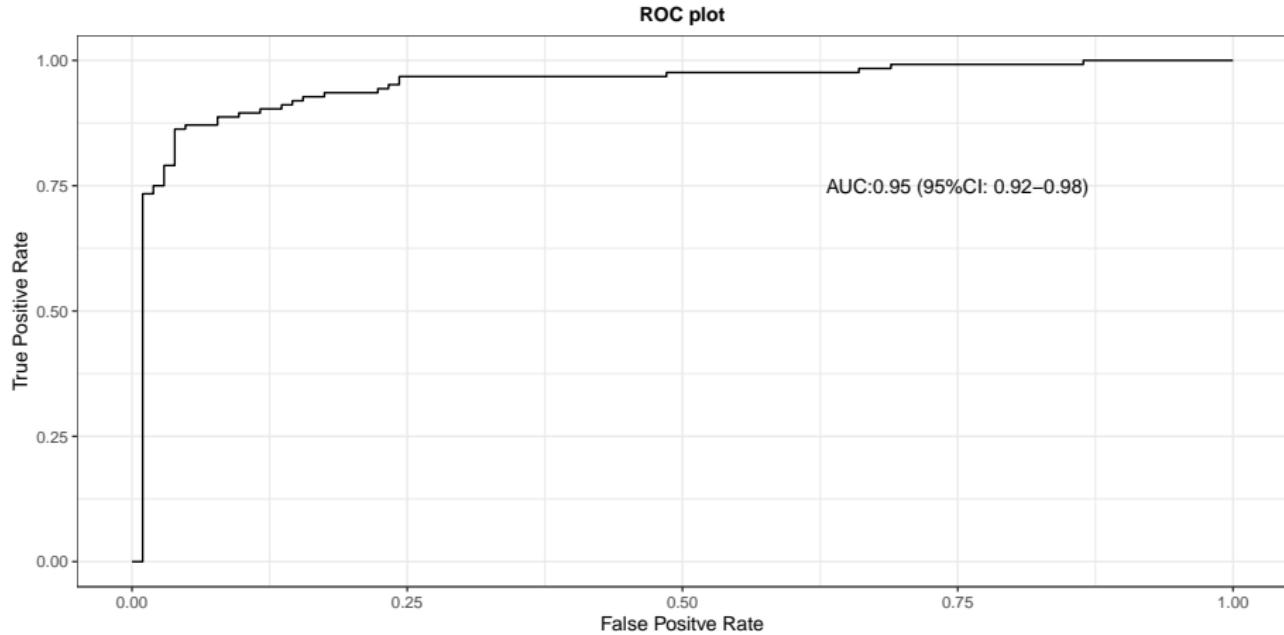
Error rates:

Healthy	LC
0.058	0.137

Overall error rate: 0.101

AUC ?

```
dis.mod.eval$roc.plot;
```



Area Under the Curve (线下面积);

AUC ≥ 0.8 就算是比较好的模型;

Feature 的相对重要性和特征选择

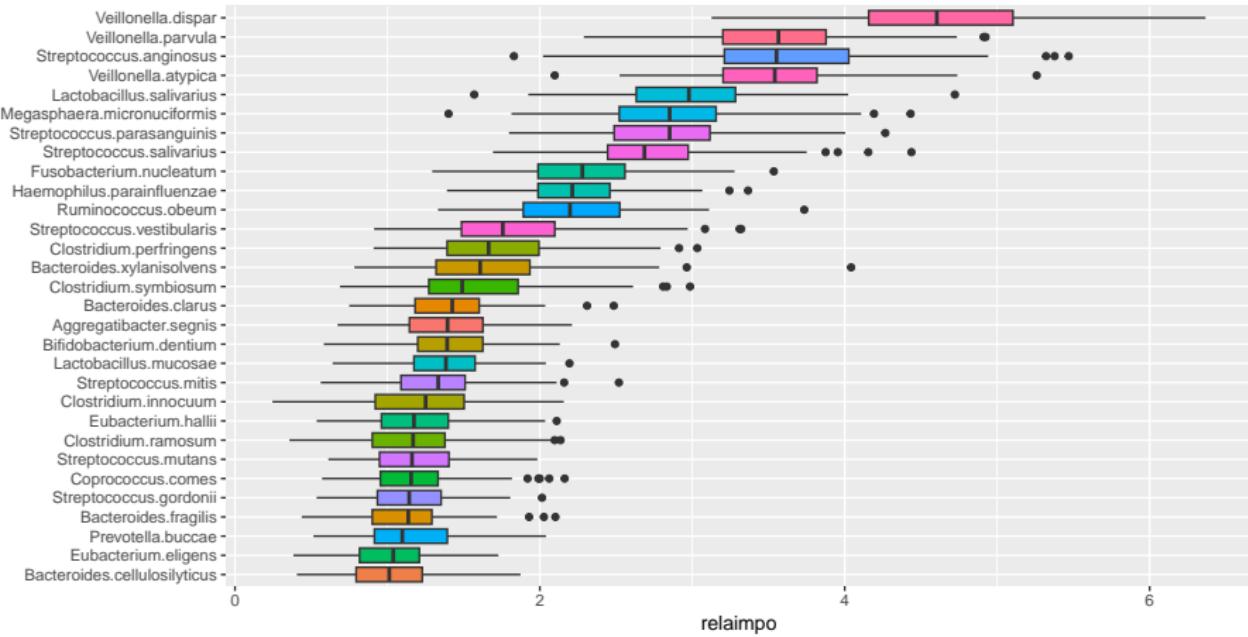
```
## 选择最重要的 30 个 feature ...
dis.mod.selected <- rf.featureselect( dis.mod, top = 30 );

## Get top features ...
## The top parameter is set, get top 30 features ...
## Parameter used to select top features: top,
##   # of top features selected: 30 ,
##   Max importance: 4.663078 ,
##   Min importance: 1.026063 ,
##
## NOTE: a new list will be created with necessary data to run 'rf.train()' function.
## Data size: 1 Mb
## Feature selection : 0.006 sec elapsed
```

显示 feature importance

```
dis.mod.selected$feat.select.importance.plot;
```

reorder(feature, relaimpo, median)



这个数据是怎么来的？每一个 model 都有 importance 这一 data.frame，将这些数据合并后，即可得到。

使用 top four species 重新建模

```

## 选择 重要性排前 4 的 feature
dis.mod.selected2 <- rf.featureselect(dis.mod, top = 4);

## Get top features ...
## The top parameter is set, get top 4 features ...
## Parameter used to select top features: top,
## # of top features selected: 4 ,
## Max importance: 4.663078 ,
## Min importance: 3.53753 ,
##
## NOTE: a new list will be created with necessary data to run 'rf.train()' function.
## Data size: 0.9 Mb
## Feature selection : 0.004 sec elapsed

## 用选择的 feature 重新建模
dis.mod.selected2.mod <- rf.train( dis.mod.selected2, parallel = T, num.cores = 7 );

## Training models on 227 samples, with 4 features.
## Case group: LC, control group: Healthy
## - running 10 resample and 10 fold modeling ...
## - NOT using class weight option of randomForest classifier ...
## - running parallel modeling using 7 cpu cores ...
## |

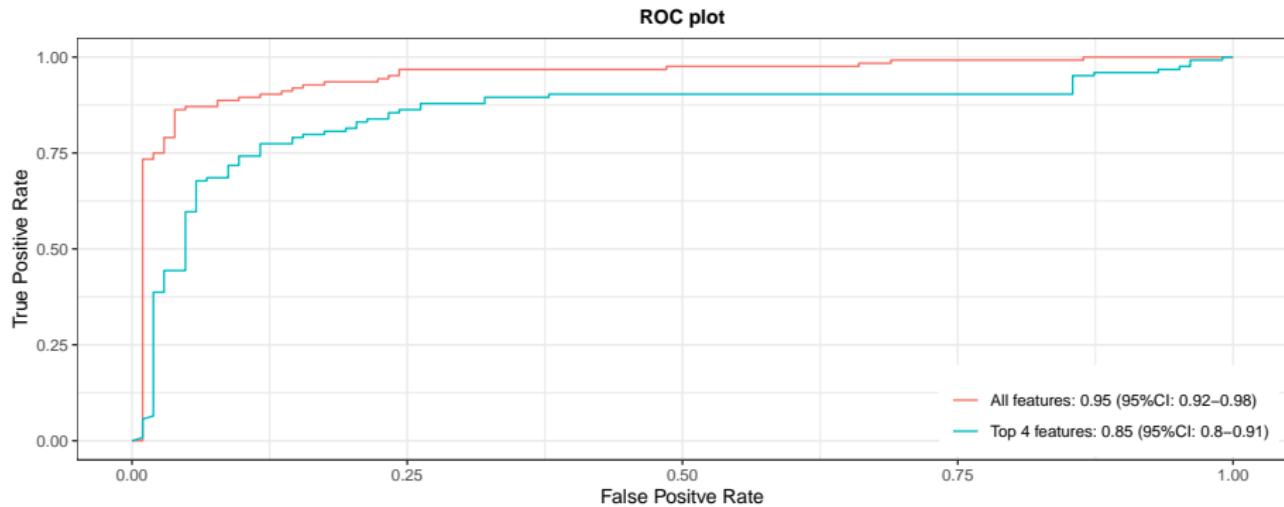
```

检验建模结果

```
dis.mod.selected2.mod.eval <- rf.evaluate( dis.mod.selected2.mod );  
  
## Evaluating model performance ...  
##  
## -----  
## Overall performance (AUC): 0.85 (95%CI: 0.8-0.91)  
##  
## Predicted vs. Truth (Group):  
## Control group: Healthy  
## Case group: LC  
##  
##           Predicted  
## Group      Healthy  LC  
##   Healthy     82  21  
##   LC          22 102  
## -----  
## Error rates:  
## Healthy      LC  
##   0.204    0.177  
##  
## Overall error rate: 0.189  
## -----  
##  
## Data size: 0.7 Mb  
## ---  
## Model performance evaluation : 0.035 sec elapsed
```

比较不同数量特征得到的结果

```
plot <-  
  rf.plot.multiple.roc( dis.mod.eval, dis.mod.selected2.mod.eval,  
                        labels = c("All features", "Top 4 features"));
```



4. 外部验证

外部验证是指用已知表型的另一个数据集，对模型进行检验；

```
## 装入外部验证数据
validation.featdata <-
  read.csv("./data/talk12/data/PRJEB6337.LC.s.validation.txt",
          sep = "\t", header = T, row.names = 1)
validation.metadata<-
  read.csv("./data/talk12/data/PRJEB6337.LC.s.validation.metadata.txt",
          sep = "\t", header = T, row.names = 1)

## 整理外部验证数据；去除分析失败的样本
qc.val.metadata <- validation.metadata[!rownames(validation.metadata) %in% qc.fail.run$run.id,]
qc.val.featdata <- validation.featdata[!rownames(validation.featdata) %in% qc.fail.run$run.id,]
```

外部验证, cont.

使用外部验证函数 rf.external.validation()

```
val.res <-  
  rf.external.validation(dis.mod, qc.val.featdata, qc.val.metadata);  
  
## Warning in rf.setdata(ext.feat.data, ext.meta.data, grouping = grouping, : San check warning  
  
## All look great.  
## -----  
## In total, the feature data contains 75 samples, with 285 features.  
## Case group: 'LC', w/ 36 samples,  
## Control group: 'Healthy', w/ 39 samples,  
## -----  
##  
## Data size: 0.4 Mb  
## Set data : 0.001 sec elapsed  
##  
## ----- WARNING -----  
## # of required features: 399  
## # of features NOT found in your input: 129  
## the predicted results may NOT be reliable due to these missing features!!  
## -----  
  
## `summarise()` has grouped output by 'Sample'. You can override using the  
## `.`groups` argument.
```

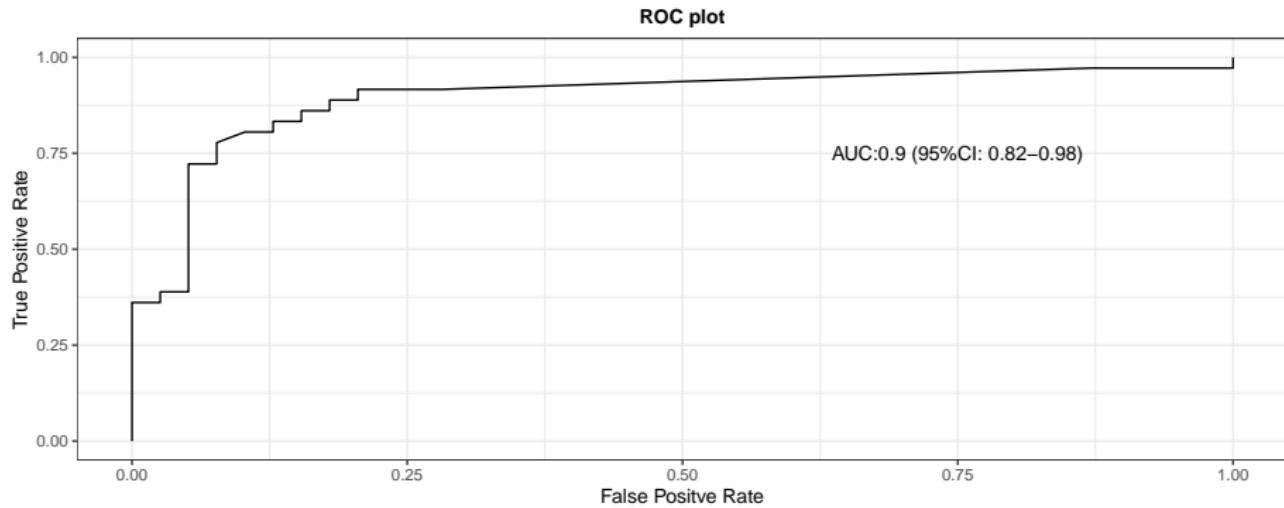
外部验证, cont.

```
val.res2 <-  
  rf.external.validation(dis.mod.selected2.mod, qc.val.featdata, qc.val.metadata);  
  
## Warning in rf.setdata(ext.feat.data, ext.meta.data, grouping = grouping, : San check warning  
  
## All look great.  
## -----  
## In total, the feature data contains 75 samples, with 285 features.  
## Case group: 'LC', w/ 36 samples,  
## Control group: 'Healthy', w/ 39 samples,  
## -----  
##  
## Data size: 0.4 Mb  
## Set data : 0.001 sec elapsed  
  
## `summarise()` has grouped output by 'Sample'. You can override using the  
## `.`groups` argument.  
  
## Prediction using external data : 1.296 sec elapsed  
  
## Setting direction: controls < cases  
  
## -----  
## Overall performance (AUC): 0.9 (95%CI: 0.82-0.98)  
##
```



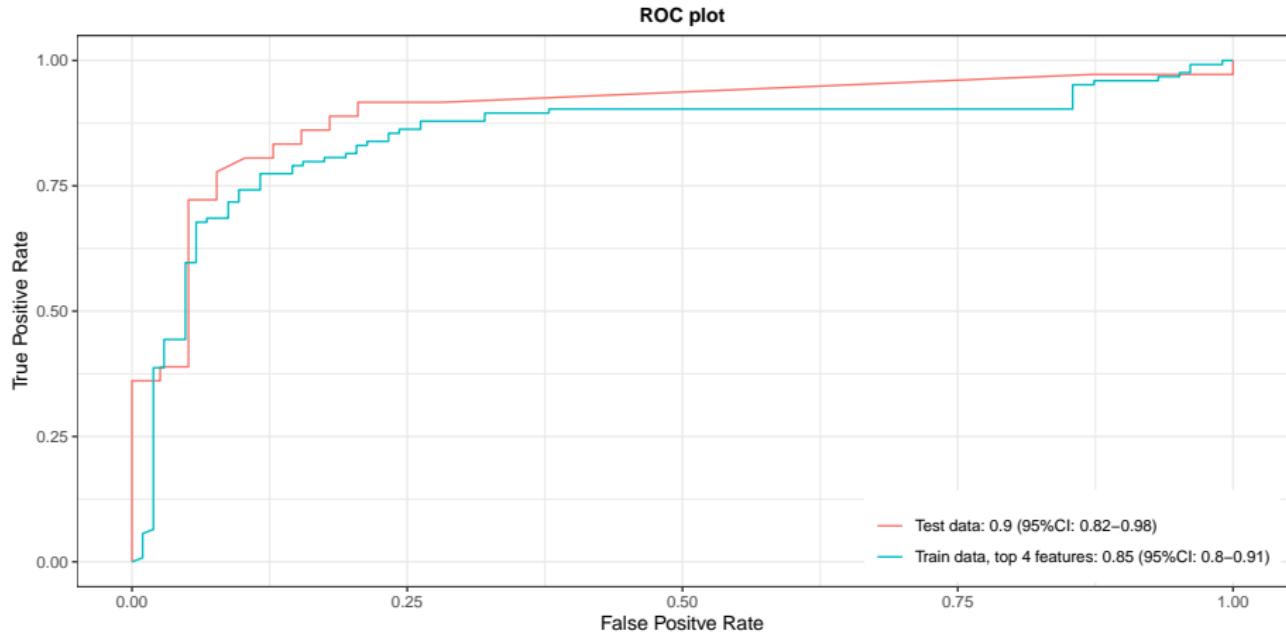
外部验证, cont.

```
val.res2$roc.plot;
```



比较 cross validation 和 external validation 结果

```
res <- rf.plot.multiple.roc( dis.mod.selected2.mod.eval, val.res2,
                            labels = c("Train data, top 4 features", "Test data"));
```

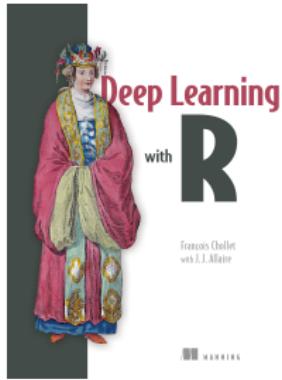


小结

- 学习机器学习的基本知识
- 每一种算法的优缺点
- 随机森林的原理、应用
- 示例

section 6: deep learning

Deep learning with R



contents

示例

- 识别数字（多分类）
- cats vs. dogs （两分类）

技巧

- 提高准确率
- 迁移学习？
- 减少过拟合

section 7: 结语

R 的优点

- 强大的格式数据处理能力 (二维表格, dplyr)
- 无以伦比的统计学专业性
- 专业而好看的数据可视化软件 (ggplot2)
- 专业的生信扩展包 (Bioconductor)
- 超级好用的整合开发环境 IDE (RStudio)

R 的缺点

- 非格式化数据处理能力差
- 运行效率一般
- 并行处理效果一般
- 机器学习效果一般

本课的资源

- all codes are available at Github:

<https://github.com/evolgeniusteam/R-for-bioinformatics>