R 语言编程：基于 tidyverse

第 30 讲 (附录) Excel 任务, 高性能计算, 机器学习

张敬信

2022 年 3 月 30 日

哈尔滨商业大学

附录 C R 实现 Excel 中的 VLOOKUP 与透视表

在用 Office 系列日常办公中，经常需要批量重复地做的一些事情，都值得用 R 或 Python 写成程序代码实现自动化，这样可以一劳永逸为你节省大量的时间，还有就是 R 或 Python 代码所能处理的数据量和处理速度都远远超过 Excel.

C.1 VLOOKUP 查询

Excel 中的 VLOOKUP, Index + Match 是让很多人头痛的话题。

实际上 VLOOKUP 在**数据思维**下来看，无非就是：筛选行、选择列，数据连接(若从较大的表查询)，如果查询完还涉及修改，就再加上修改列。

数据思维，才是解决数据问题的正确思维、简洁思维，一看就懂，一用就会。

比如有如下的 Excel 数据表：

No	销售员	性别	销量	地区
1	王东	男	100	北京
2	小西	男	56	上海
3	小南	女	98	苏州
4	小北	女	66	上海
5	小中	男	87	天津
6	小王	女	99	上海
7	小李	男	20	上海

图 1: Excel 查询示例数据

先加载 tidyverse 包并读入数据：

```
library(tidyverse)
library(readxl)
df = read_xlsx("datas/VLOOKUP 综合.xlsx")
```

先来看各种查询，其实就是构建筛选条件筛选行，想保留哪些列再选择列。

- 单条件查询：根据销售员查找销量

```
df %>%
  filter(销售员 == " 王东") %>%                    # 筛选行
  select(销售员, 销量)                            # 选择列
#> # A tibble: 1 x 2
#>   销售员  销量
#>   <chr>  <dbl>
#> 1 王东     100
```

注：查询多个销售员姓名，还可以用%in% （属于）写查询条件。

根据另一个表（比如 df2）里的销售员姓名查询，把筛选行改成右连接即可：

```
df %>%
  right_join(df2, by = " 销售员") %>%     # 右连接
  select(销售员, 销量)                    # 选择列
```

- 多条件查询：查询销售员在某地区的销量

```
df %>%      # 根据两个条件筛选行
  filter(销售员 == " 王东", 地区 == " 北京") %>%
  select(销售员, 地区, 销量)
#> # A tibble: 1 x 3
#>   销售员 地区   销量
#>   <chr>  <chr> <dbl>
#> 1 王东   北京    100
```

- 多列查询：查询销售员的所有信息

```
df %>%
  filter(销售员 == " 王东")
#> # A tibble: 1 x 6
#>      No 销售员 性别   销量 地区   备注
#>   <dbl> <chr>  <chr> <dbl> <chr>  <lgl>
#> 1     1 王东   男      100 北京   NA
```

- 从右向左查询：查询某销量的销售员

```
df %>%     # 数据思维不用分左右
  filter(销量 == 66) %>%
  select(销量, 销售员)
#> # A tibble: 1 x 2
#>    销量 销售员
#>   <dbl> <chr>
#> 1    66 小北
```

- 使用通配符查询：查询姓名包含”中”

```
df %>%    # 是否检测到" 中" 字，支持正则表达式
  filter(str_detect(销售员, " 中")) %>%
  select(销售员, 销量)
#> # A tibble: 1 x 2
#>   销售员  销量
#>   <chr>  <dbl>
#> 1 小中      87
```

- 划分区间等级[1]：按销量划分等级

```
df %>%
  mutate(销量等级 = case_when(      # 修改列
    销量 < 60  ~ " 不及格",
    销量 < 85  ~ " 及格",
    TRUE         ~ " 优秀"))
#> # A tibble: 7 x 7
#>      No 销售员 性别   销量 地区  备注  销量等级
#>   <dbl> <chr>  <chr> <dbl> <chr> <lgl> <chr>
#> 1     1 王东   男      100 北京  NA    优秀
#> 2     2 小西   男       56 上海  NA    不及格
#> 3     3 小南   女       98 苏州  NA    优秀
#> # ... with 4 more rows
```

[1]对变量重新编码、连续变量离散化

C.2 数据透视表

数据透视表就是透过数据看到汇总的信息，其实就是分组汇总。比如有如下 Excel 数据表：

地区	城市	公司名称	类别名称	产品名称	订购日期	数量	单价	销售额
华北	天津	高上补习	饮料	苹果汁	1996-08-20	45	14.4	648
华东	温州	学仁贸易	饮料	苹果汁	1996-08-30	18	14.4	259
华北	天津	正太实业	饮料	苹果汁	1996-09-30	20	14.4	288
华北	天津	凯旋科技	饮料	苹果汁	1996-11-07	15	14.4	216
华北	天津	就业广兑	饮料	苹果汁	1996-11-14	12	14.4	173
华北	天津	浩天旅行	饮料	苹果汁	1996-12-03	15	14.4	216
华北	北京	留学服务	饮料	苹果汁	1997-01-07	10	14.4	144
华北	天津	池春建设	饮料	苹果汁	1997-01-14	24	14.4	346
华北	张家口	康毅系统	饮料	苹果汁	1997-03-17	15	14.4	216

图 2: Excel 透视表示例数据

想透过数据得到各年份分地区的销售额，这就是按年份、地区分组，对销售额做加和汇总。

- **方法一**：分组汇总 + 长变宽

```
df = read_xlsx("datas/数据透视表.xlsx")
library(lubridate)
pt = df %>%
  group_by(年份 = year(订购日期), 地区) %>%
  summarise(销售额 = sum(销售额))
pt
#> # A tibble: 36 x 3
#> # Groups:   年份 [6]
#>    年份 地区  销售额
#>   <dbl> <chr>  <dbl>
#> 1  1996 东北   16984
#> 2  1996 华北   95935
#> 3  1996 华东   51792
#> # ... with 33 more rows
```

Excel 透视表一般不是这样的整洁长表，而是更具可读性的宽表，再对地区列来个长变宽：

```
pt = pt %>%
  pivot_wider(names_from = 地区, values_from = 销售额)
pt
#> # A tibble: 6 x 7
#> # Groups:   年份 [6]
#>    年份  东北   华北   华东   华南  西北  西南
#>   <dbl> <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl>
#> 1  1996 16984  95935  51792  38761  1883 20950
#> 2  1997 36984 260084 141214 129679  7683 93889
#> 3  1998  2847 157555  99536  87778  2965 94574
#> # ... with 3 more rows
```

Excel 透视表一般还增加按行、按列的加和：

```
library(janitor)
pt %>%
  adorn_totals(where = c("row", "col"))   # 也可以只用一个
#>    年份   东北   华北   华东   华南  西北   西南   Total
#>    1996  16984  95935  51792  38761  1883  20950  226305
#>    1997  36984 260084 141214 129679  7683  93889  669533
#>    1998   2847 157555  99536  87778  2965  94574  445255
#>    1999    861  65997 160557 170917   178   6271  404781
#>    2000    861 110237  28557 181917   178  86271  408021
#>    2001 100461  12625 104357 117102   178 106326  441049
#>   Total 158998 702433 586013 726154 13065 408281 2594944
```

以上是为了展示中间结果，也可以四步管道操作直接到最终透视表。

方法二：tidyquant::pivot_table()

tidyquant 包是做量化金融的包，顺便实现了大部分的 Excel 函数，pivot_table 就是其中之一，它类似于 Excel 中做透视表，选择行分组、列分组的变量或表达式，以及汇总函数。

- 相当于将方法一中的前三步一步到位，只是要增加行和、列和，仍需要接 adorn_totals()：

```
library(tidyquant)
df %>%
  pivot_table(.rows = ~ YEAR(订购日期), .columns = 地区,
              .values = ~ SUM(销售额)) %>%
  adorn_totals(where = c("row", "col"))  # 结果同上（略）
```

方法三：pivottabler 包

pivottabler 包是专为做数据透视表而生，强大到无以复加：可以任意精细定制、导出各种格式。

```
library(pivottabler)
df %>%
  mutate(年份 = year(订购日期)) %>%
  qpvt(rows = " 年份", columns = " 地区",
       calculations = "sum(销售额)")   # 结果同上（略）
```

默认增加行和、列和，相当于参数 totals = c(" 年份", " 地区"), 可以只留一个，或 totals = "" 两个都不留。

数据透视表的进一步定制格式和美化导出可借助 openxlsx 包，或者在 Excel 中再做。

D. R 高性能计算

作为高级语言，R 语言能极大地节省您写代码的时间，不过运行速度要比 C++ 等慢不少，但是 R 也有办法提速，做高性能计算。

本篇介绍几种 R 中最常用的高性能计算的办法，通常是用于处理大数据量的时候。

D.1 并行计算

现在电脑 CPU 都是多核心多线程的，R 默认是只有一个线程工作，其它线程闲着显然是一种浪费。并行计算，就是让多个或全部线程同时工作。

`future` 包为 R 用户提供了并行与分布处理的统一接口，还有些包已经将并行计算隐式地嵌入其中并自动启用，比如 `data.table`, `mlr3` 等。

- future 的基本使用

```
library(future)
availableCores()          # 查看电脑可用的线程数
#> system
#>     12

# 启用多线程, 参数 workers 可设置线程数
plan(multisession)
f <- future({
...                       # 要并行加速的代码
})
value(f)
plan(sequential)          # 回到单线程
```

- 循环迭代的并行加速

本书主张用 purrr 包中的 map_*, walk_*, modify_* 等做循环迭代，对它们做并行加速，只需要加载 furrr 包，启用多线程，再把每个函数名添加前缀 future_ 即可。

```
library(furrr)
library(purrr)
plan(multisession, workers = 6)
future_map_dbl(iris[1:4], mean)
#> Sepal.Length  Sepal.Width Petal.Length  Petal.Width
#>         5.84         3.06         3.76         1.20
```

D.2 运行 C++ 代码

另一种代码提速的办法是将 R 代码改写成 C++ 代码，借助 Rcpp 包运行它，但需要安装 C++ 编译环境：

- Windows 系统安装 Rtools
- Mac 系统安装 Xcode
- Linux 系统：sudo apt-get install r-base-dev

cppFunction() 函数让你可以在 R 中写 C++ 函数：

```
library(Rcpp)
cppFunction('int add(int x, int y) {
  int sum = x + y;
  return sum;
}')
add(1, 2)
#> [1] 3
```

也可以编写标准的 C++ 文件，保存为.cpp，再用 sourceCpp() 在 R 中执行，以下代码来自 (Wickham, 2019):

```
#include <Rcpp.h>
using namespace Rcpp;
// [[Rcpp::export]]
double meanC(NumericVector x) {
  int n = x.size();
  double total = 0;
  for(int i = 0; i < n; ++i) {
    total += x[i];
  }
  return total / n;
}
```

```
/*** R
x <- runif(1e5)
bench::mark(                    # 测速对比
  mean(x),
  meanC(x)
)
*/
```

注意，文件开头都要加上这样两行，在每个自定义函数前加上//[[Rcpp::export]]，如果想要包含 R 代码，按上述注释的方式加入。

```
sourceCpp("test.cpp")     # R 中运行 cpp 文件
```

D.3 操作超过内存的数据集

首先一点：内存能应付的若干 G 级别的数据集，data.table 是最优选择，没有之一。

有时单机电脑需要操作**超过内存但没超过硬盘**的大数据，disk.frame 包提供了简单的解决方案[2]：

- 将超过内存的数据分割成若干块，并将每个块作为独立文件存储在一个文件夹中
- 支持用 dplyr 和 data.table 语法“整体地”操作这些数据块

[2] HarryZhu: disk.frame 和 Spark 的最大区别就是，disk.frame 优先解决的是计算密集型的复杂模型运算任务，让数据跟随计算；而 Spark 因为使用的是 Map-Reduce 模型，计算跟随数据，所以更擅长数据密集型的简单 ETL 运算任务.

```r
library(disk.frame)
# 启用多线程, 参数 workers 可设置线程数
setup_disk.frame()
# 允许大数据集在 session 之间传输
options(future.globals.maxSize = Inf)
## 从 csv 文件创建 disk.frame
# flights = csv_to_disk.frame(
#  infile = "datas/flights.csv",
#  outdir = "temp/tmp_flights.df",
#  nchunks = 6,                # 分为 6 个数据块
#  overwrite = TRUE)
flights = as.disk.frame(nycflights13::flights)
```

创建到硬盘上的分块数据集如下：

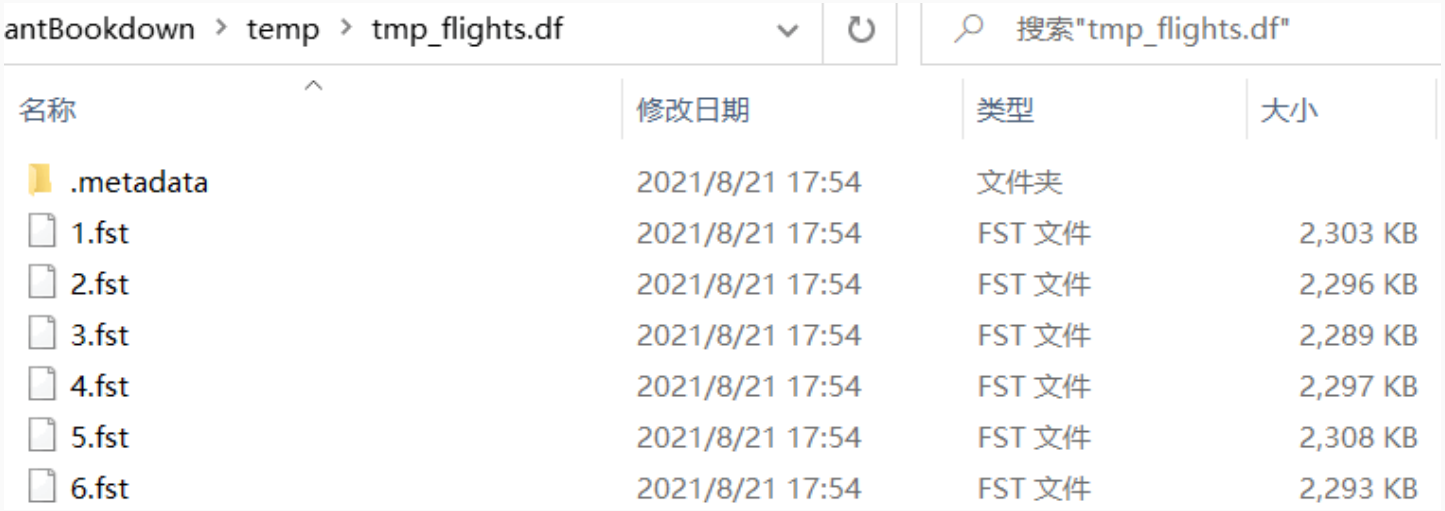

图 3: disk.frame 分割后的数据集

可以像用 `dplyr` 语法操作普通的数据框一样操作 `flights`，只是与连接远程数据库一样，执行的是懒惰计算，经过 `collect()` 后才能真正执行计算：

```r
flights %>%
  filter(month == 5, day == 17,
         carrier %in% c('UA', 'WN', 'AA', 'DL')) %>%
  select(carrier, dep_delay, air_time, distance) %>%
  mutate(air_time_hours = air_time / 60) %>%
  collect() %>%
  arrange(carrier) %>%  # arrange 应该在 collect 之后
  head()
```

```
#>    carrier dep_delay air_time distance air_time_hours
#> 1:      AA        -7      142     1089           2.37
#> 2:      AA        -9      186     1389           3.10
#> 3:      AA        -6      143     1096           2.38
#> 4:      AA        -4      114      733           1.90
#> 5:      AA        -2      146     1085           2.43
#> 6:      AA        -7      119      733           1.98
```

做其它数据操作：分组汇总、数据连接等也是类似的，甚至还可以像对普通数据框构建模型一样地进行统计建模和提取模型结果。

注：真正的 R 中的分布式大数据平台（多台服务器/电脑）操作大数据，是用 `sparklyr` 包连接 `Apache Spark`，参阅 sparklyr from RStudio.

D.4 大型矩阵运算

bigmemory 包提供了三种类型的 big.matrix 对象:

- 内存 big.matrix: 不在多线程间共享，直接使用随机存取内存
- 共享 big.matrix: 使用部分共享内存
- 文件后端 big.matrix: 在进程之间共享，将数据存储在硬盘上，并通过内存映射访问它

在此基础之上，bigstatsr 包提供了更强大易用的 FBM 对象（文件后端大矩阵)，这样的矩阵数据是存储在硬盘上，所以同样能突破内存限制。

bigstatsr 包还提供了强大的 big_apply() 函数：分割（将矩阵分成若干列块）-应用（函数）-合并（结果），且支持并行计算。

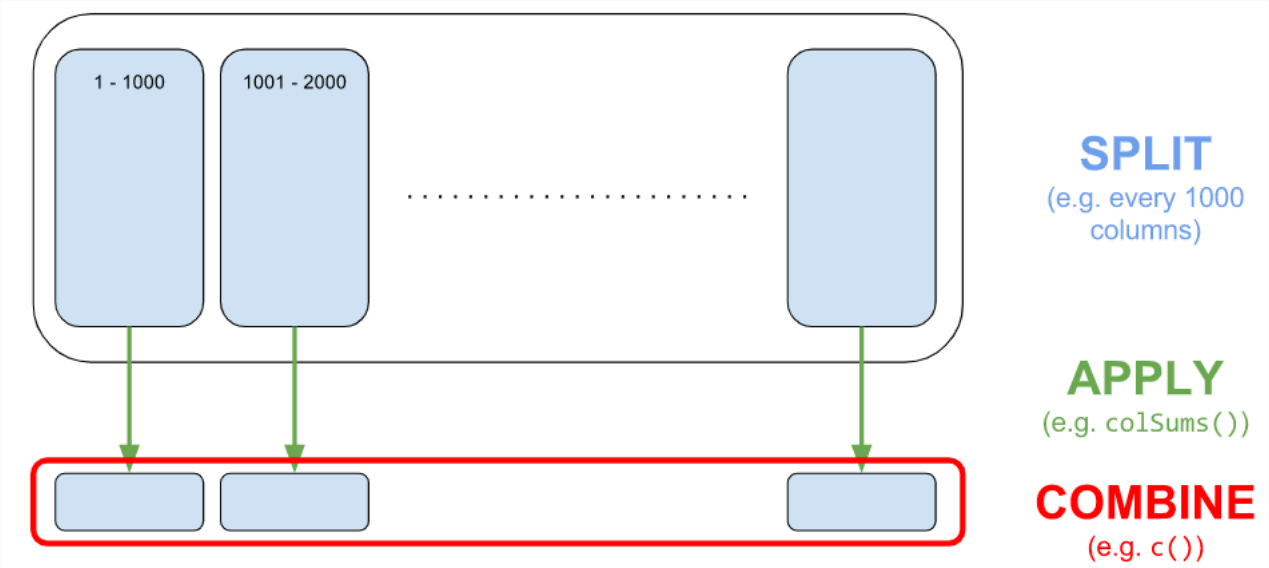

图 4: 大矩阵" 分割-应用-合并" 机制

bigstatsr 包简单使用：创建 1000 列的大矩阵，分割成 500 列一块，计算列和，并打开两个核心并行计算。

```r
library(bigstatsr)
X = FBM(10000, 1000, init = rnorm(10000 * 1000),
        backingfile = "temp/test")
object.size(X)
#> 680 bytes
file.size(X$backingfile)
#> [1] 8e+07
typeof(X)
#> [1] "double"
sums = big_apply(X, a.FUN = function(X, ind) {
  colSums(X[,ind])
}, a.combine = "c", block.size = 500, ncores = 2)
sums[1:5]
#> [1] -82.1 -61.3 -51.7  89.1 212.9
```

另外，bigstatsr 包已经内置了很多操作大矩阵的有用函数（具体使用请参阅包文档），比如：

- 统计建模：big_univLinReg()，big_univLogReg()，big_spLinReg()，big_spLogReg() 等
- SVD(PCA)：big_SVD()，big_randomSVD()
- 矩阵运算：big_cor()，big_cprodMat()，big_prodMat()，big_transpose() 等
- 功能函数：big_colstats()，big_scale()，big_counts()，big_read()，big_write() 等。

E. R 最新机器学习框架

E.1 mlr3verse

mlr3verse 是最新、最先进的 R 机器学习框架，它基于面向对象 R6 语法和 data.table 底层数据流（速度超快），支持 future 并行，支持搭建“图”流学习器，理念非常先进、功能非常强大。

mlr3verse 整合了各种机器学习算法包，实现了统一、整洁的机器学习流程化操作，足以媲美 Python 的 scikit-learn 机器学习库。

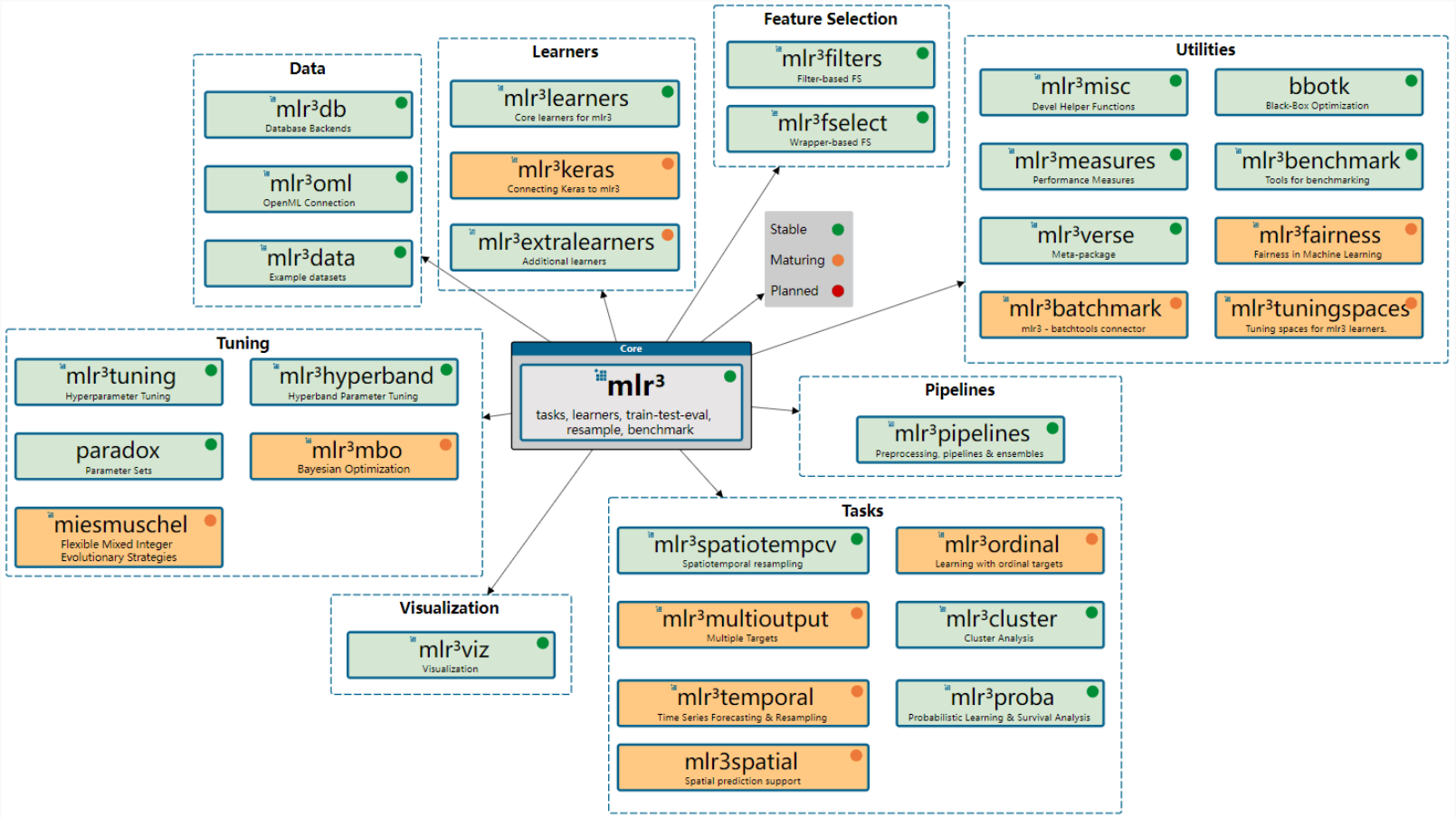

图 5: mlr3verse 机器学习框架生态

mlr3verse 核心包包括:

- mlr3: 机器学习
- mlr3db: 操作后台数据库
- mlr3filters: 特征选择
- mlr3learners: 机器学习学习器
- mlr3pipelines: 特征工程，搭建图流学习器
- mlr3tuning: 超参数调参
- mlr3viz: 可视化
- paradox: 模型解释

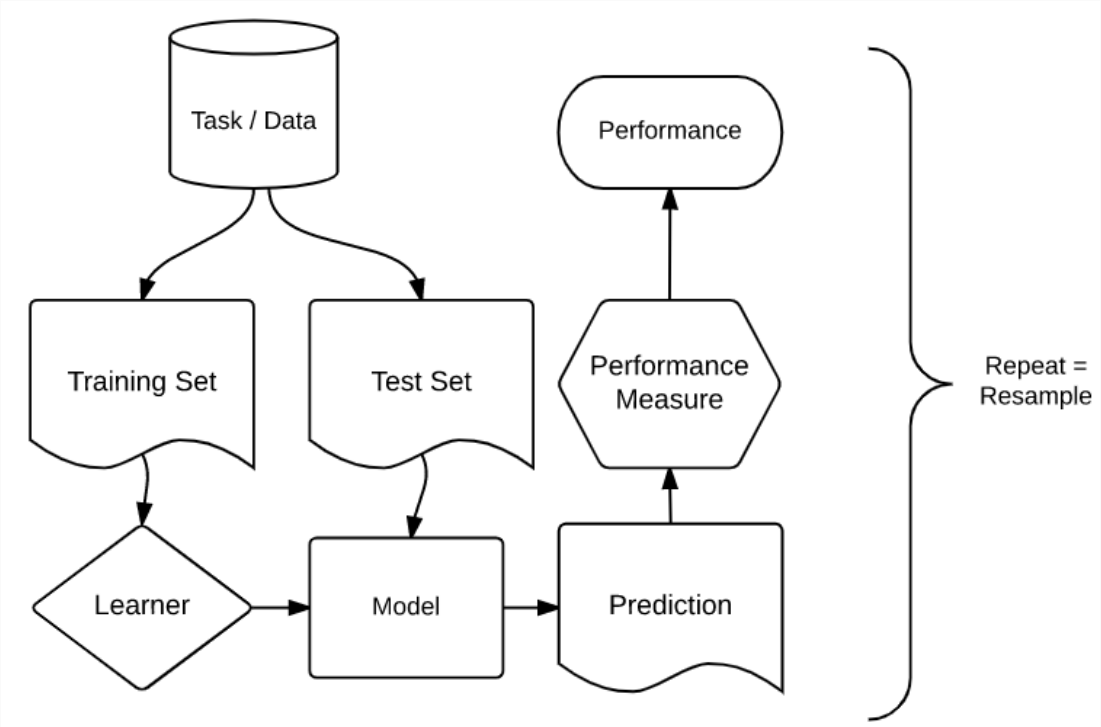

图 6: mlr3verse 机器学习基本工作流

- **任务** (Task)：封装了数据及元信息，如目标变量；
- **学习器** (Learner)：封装了各种机器学习算法，能够训练模型并做预测，大多数学习器都有影响其性能的超参数；
- **度量** (Measure)：基于预测值与真实值的差异计算的数值指标；
- **重抽样** (Resampling)：生成一系列的训练集和测试集；
- **管道** (Pipelines)：机器学习建模工作流，包括特征工程（缺失值处理、特征缩放/变换/降维)、特征选择、图机器学习等。

以对 iris 数据集构建简单的决策树分类模型为例来演示：

```
library(mlr3verse)
## 创建分类任务
task = as_task_classif(iris, target = "Species")
## 选择学习器，并设置两个超参数：最大深度，最小分支节点数
learner = lrn("classif.rpart" , maxdepth = 3, minsplit = 10)
## 划分训练集/测试集
set.seed(123)
split = partition(task, ratio = 0.7)
## 训练模型
learner$train(task, row_ids = split$train)
```

```r
## 模型预测
predictions = learner$predict(task, row_ids = split$test)
## 模型评估
predictions$confusion                          # 混淆矩阵
#>             truth
#> response     setosa versicolor virginica
#>   setosa         15          0         0
#>   versicolor      0         14         0
#>   virginica       0          1        15
predictions$score(msr("classif.acc"))      # 准确率
#> classif.acc
#>       0.978
```

这里只展示最简示例，更多的特征工程、特征选择、超参数调参、集成学习等及实例请参阅 mlr3book 和 mlr3gallery.

tidymodels 是与 tidyverse 一脉相承的 “管道流”R 机器学习框架，提供了统一的统计推断、统计建模、机器学习算法接口。

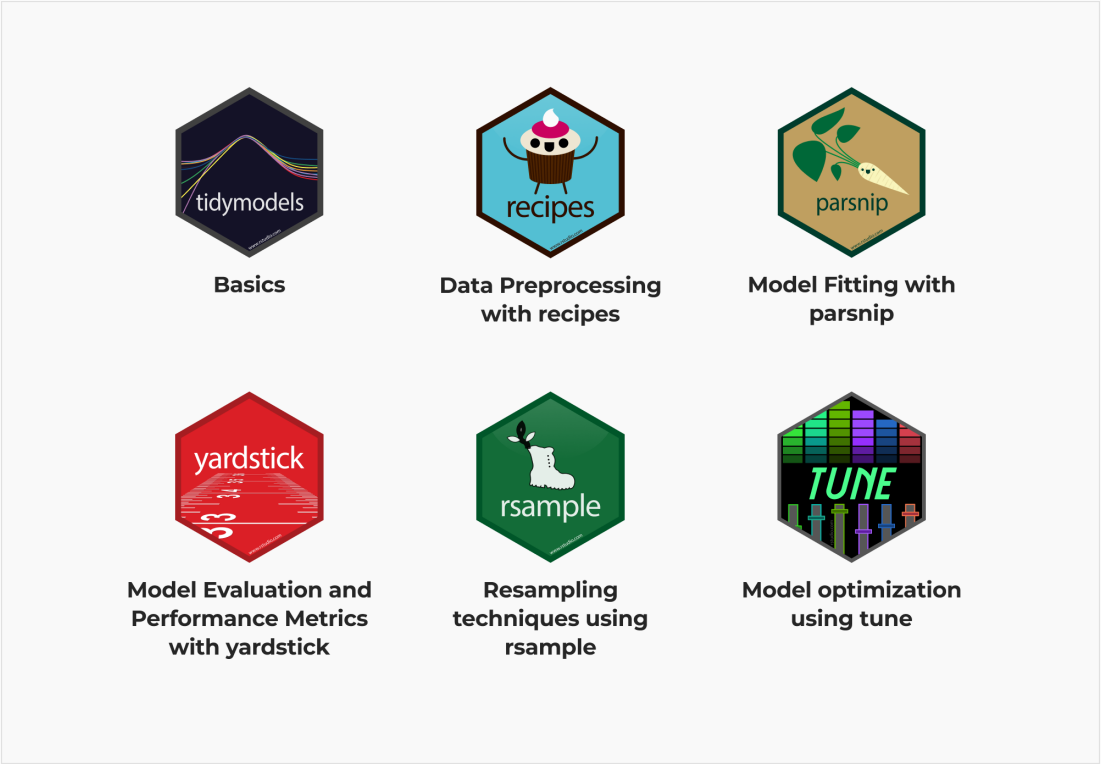

图 7: tidymodels 核心包及其功能

tidymodels 核心包有:

- parsnip: 拟合模型
- recipes: 数据预处理和特征工程
- rsample: 重抽样
- yardstick: 评估模型性能
- dials: 定义调参空间
- tune: 超参数调参
- workflows: 构建建模工作流

同样以对 iris 数据集构建简单的决策树分类模型为例来演示：

```
library(tidymodels)
## 划分训练集/测试集
set.seed(123)
split = initial_split(iris, prop = 0.7, strata = Species)
train = training(split)
test = testing(split)
## 训练模型
model = decision_tree(mode = "classification",
                      tree_depth = 3, min_n = 10) %>%
  set_engine("rpart") %>%    # 来自哪个包或方法
  fit(Species ~ ., data = train)
```

```r
## 模型预测
pred = predict(model, test) %>%
  bind_cols(select(test, Species))
## 模型评估
pred %>%
  conf_mat(truth = Species, .pred_class)     # 混淆矩阵
#>             Truth
#> Prediction   setosa versicolor virginica
#>   setosa         15          0         0
#>   versicolor      0         14         0
#>   virginica       0          1        15
```

```
pred %>%
  accuracy(truth = Species, .pred_class)      # 准确率
#> # A tibble: 1 x 3
#>   .metric  .estimator .estimate
#>   <chr>    <chr>          <dbl>
#> 1 accuracy multiclass     0.978
```

这里只展示最简示例，更多的特征工程、特征选择、超参数调参、集成学习、工作流等及实例请参阅 Tidy Modeling with R 和 tidymodels 官网.

本篇主要参阅 (张敬信, 2022), 以及包文档, 以及包文档，模板感谢 (黄湘云, 2021), (谢益辉, 2021).

参考文献

Wickham, H. (2019). *Advanced R*. Chapman and Hall/CRC, 2 edition. ISBN 978-0815384571.

张敬信 (2022). *R* 语言编程：基于 *tidyverse*. 人民邮电出版社, 北京.

谢益辉 (2021). *rmarkdown: Dynamic Documents for R*.

黄湘云 (2021). *Github: R-Markdown-Template*.