R 语言编程：基于 tidyverse

第 30 讲 (附录)　高性能计算, 机器学习

张敬信

2022 年 12 月 7 日

哈尔滨商业大学

F. 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。

F.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
#>     20

# 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
```

F.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
```

F.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[1]：

- 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

[1] 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 = "data/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：

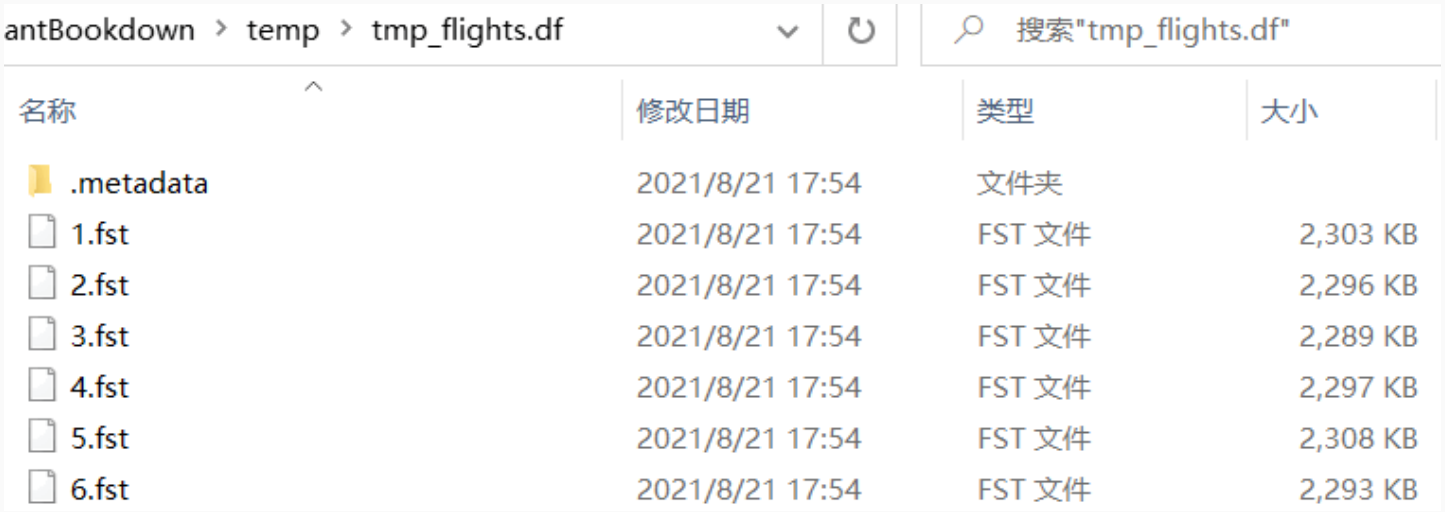

图 1： disk.frame 分割后的数据集

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.

F.4 大型矩阵运算

常规的手写代码做大矩阵运算，使用自带的 RBLAS 速度很慢，可以替换成开源的 OpenBLAS[2]，将轻松提速几十甚至上百倍。

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。

[2]替换方法参阅医学和生信笔记：让你的 R 语言提速 100 倍.

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。

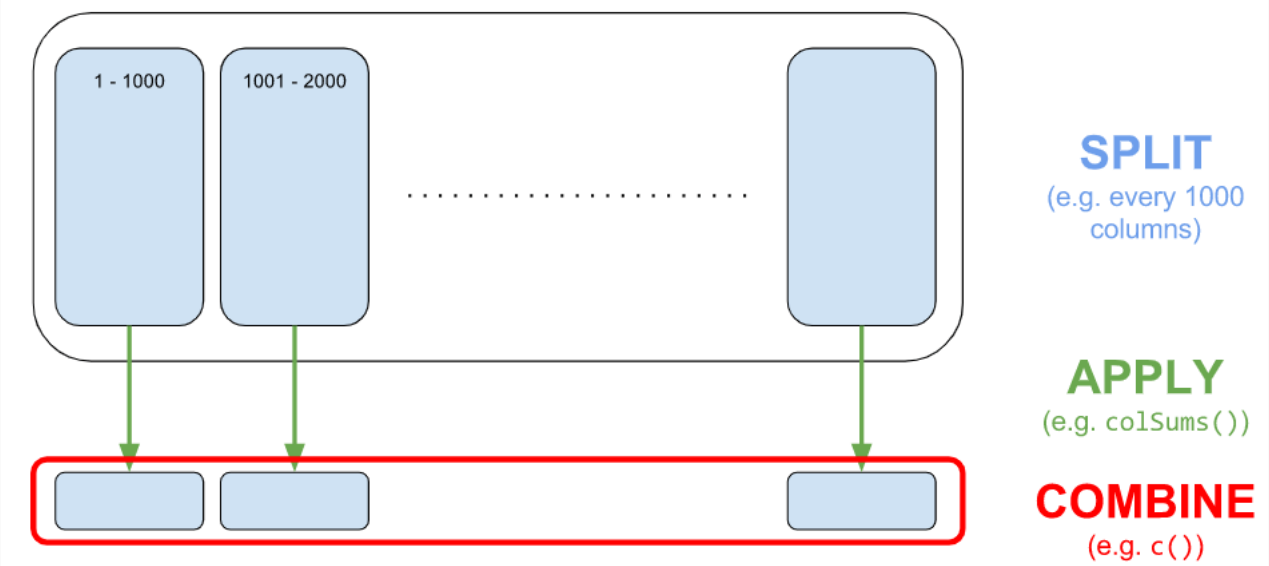

图 2: 大矩阵”分割-应用-合并”机制

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] -36.7 111.7  40.0 179.7 113.2
```

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。

G. R 最新机器学习框架

G.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。

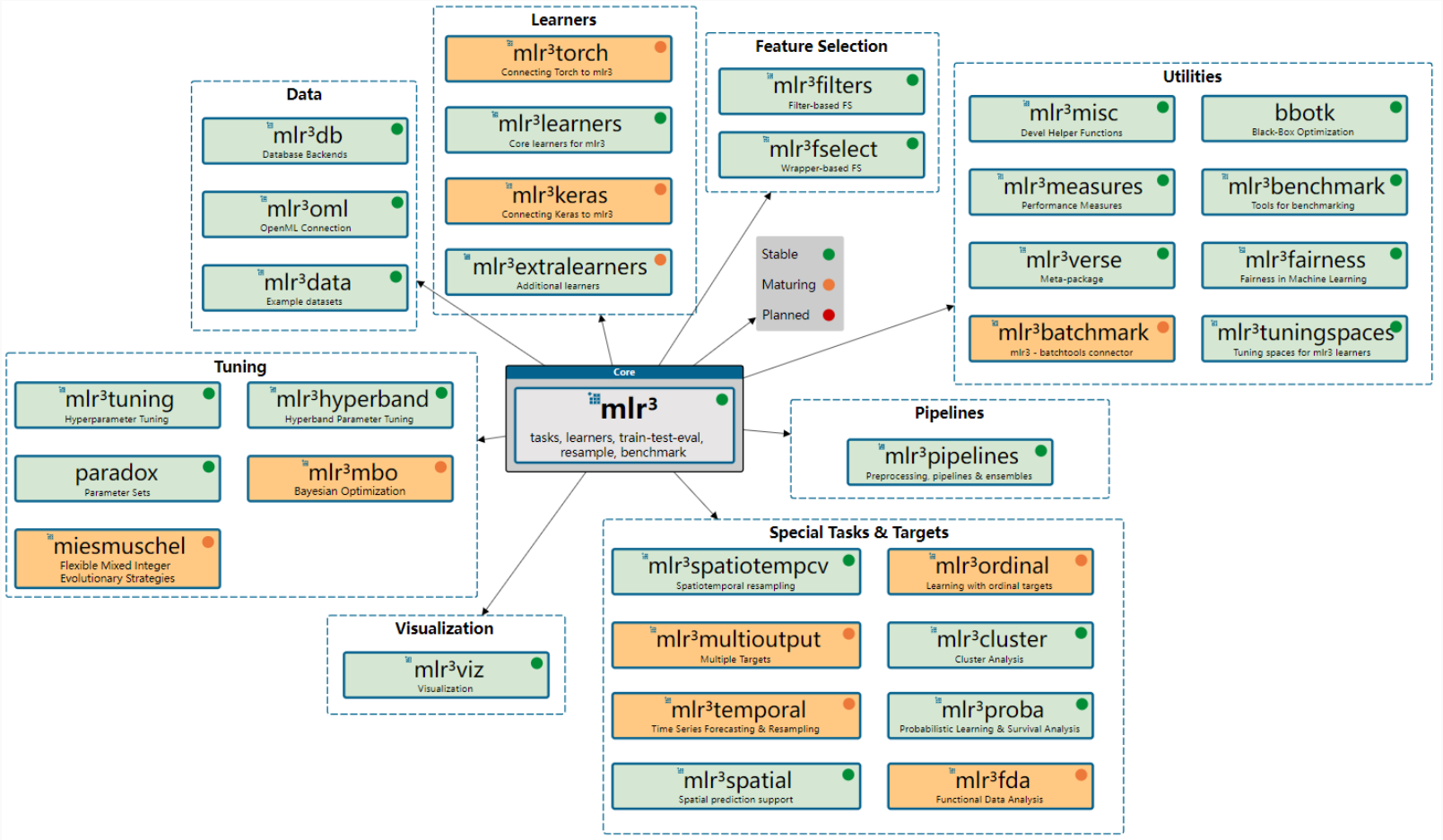

图 3: mlr3verse 机器学习框架生态

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

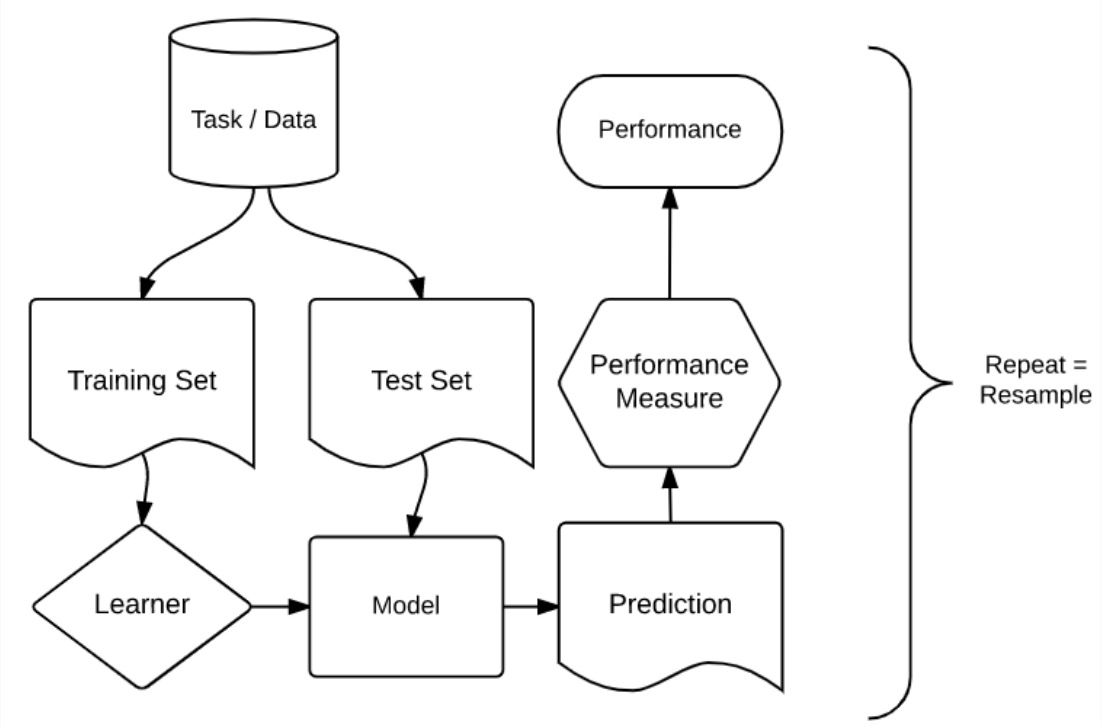

图 4: mlr3verse 机器学习基本工作流

- **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.

G.2 tidymodels

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。

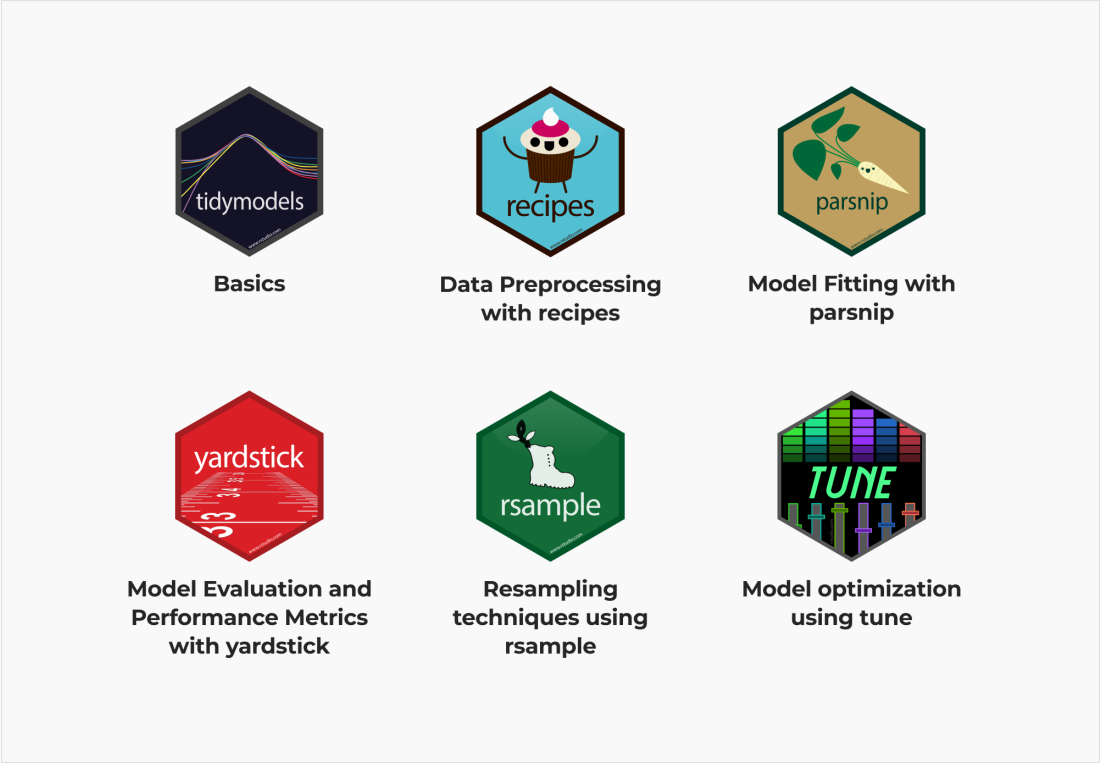

图 5: tidymodels 核心包及其功能

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
```

```r
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)，以及博客文章，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*.