R 语言编程：基于 tidyverse

第 18 讲　统计建模技术

张敬信

2022 年 2 月 15 日

哈尔滨商业大学

- tidyverse 主张以”整洁的”数据框作为输入，但是 lm, nls, t.test, kmeans 等统计模型的输出结果，却是”不整洁的”列表
- broom 包实现将模型输出结果转化为整洁的 tibble，且列名规范一致，方便后续取用
- 再与 tidyr::nest/unnest 以及 purrr::map 连用，非常便于批量建模和批量整合模型结果

一. broom 包整洁模型结果

1. `tidy()`: **模型系数估计及其统计量**
   - 返回结果 `tibble` 的每一行都是具有明确含义的项，如回归模型的一项，一个统计检验，一个聚类；
   - 各列包括：
     - `term`: 回归或模型中要估计的项
     - `estimate`: 参数估计值
     - `statistic`: 检验统计量
     - `p.value`: 检验统计量的 p 值
     - `conf.low`, `conf.high`: `estimate` 的置信区间界
     - `df`: 自由度

2. glance(): 模型诊断信息

- 返回一行的 tibble, 各列是模型诊断信息:
  - r.squared: R^2
  - adj.r.suquared: 根据自由度修正的 R^2
  - sigma: 残差标准差估计值
  - AIC, BIC: 信息准则

3. augment(): 增加预测值列、残差列等

- augment(model, data, newdata): 若 data 参数缺失，则不包含原始数据，若设置 newdata 则只针对新数据
- 返回结果 tibble 的每一行都对应原始数据或新数据的一行
- 新增加的列包括:
    - .fitted: 预测值，与原始数据同量纲
    - .resid: 残差，真实值减去预测值
    - .cluster: 聚类结果

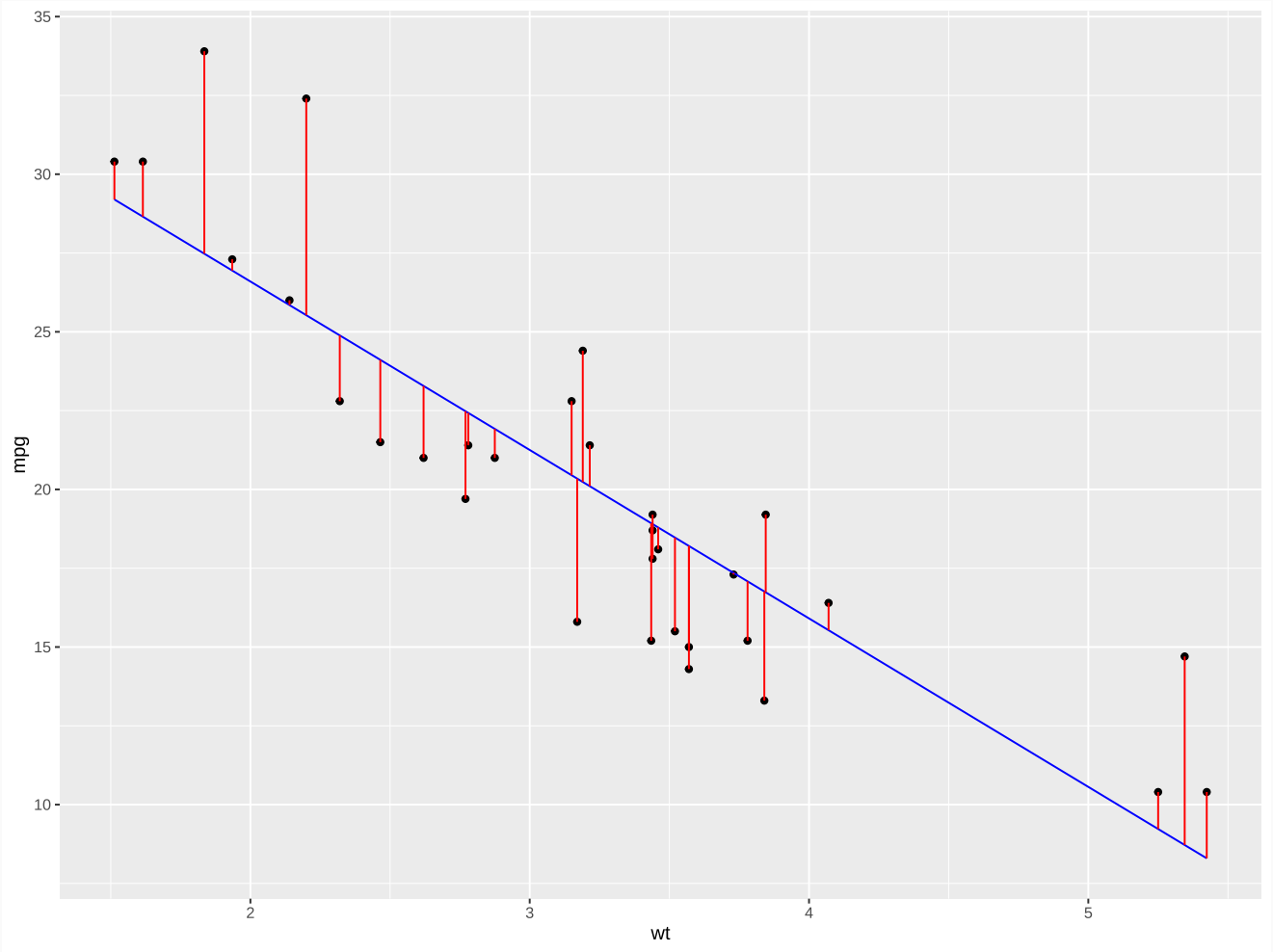
```r
library(broom)
model = lm(mpg ~ wt, data = mtcars)
model %>%
  tidy()
#> # A tibble: 2 x 5
#>   term        estimate std.error statistic  p.value
#>   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
#> 1 (Intercept)    37.3      1.88      19.9  8.24e-19
#> 2 wt             -5.34     0.559     -9.56 1.29e-10
```

```
model %>%
  glance()
#> # A tibble: 1 x 12
#>   r.squared adj.r.squared sigma statistic  p.value    df l
#>       <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>
#> 1     0.753         0.745  3.05      91.4 1.29e-10     1
#> # ... with 3 more variables: deviance <dbl>, df.residual <
```

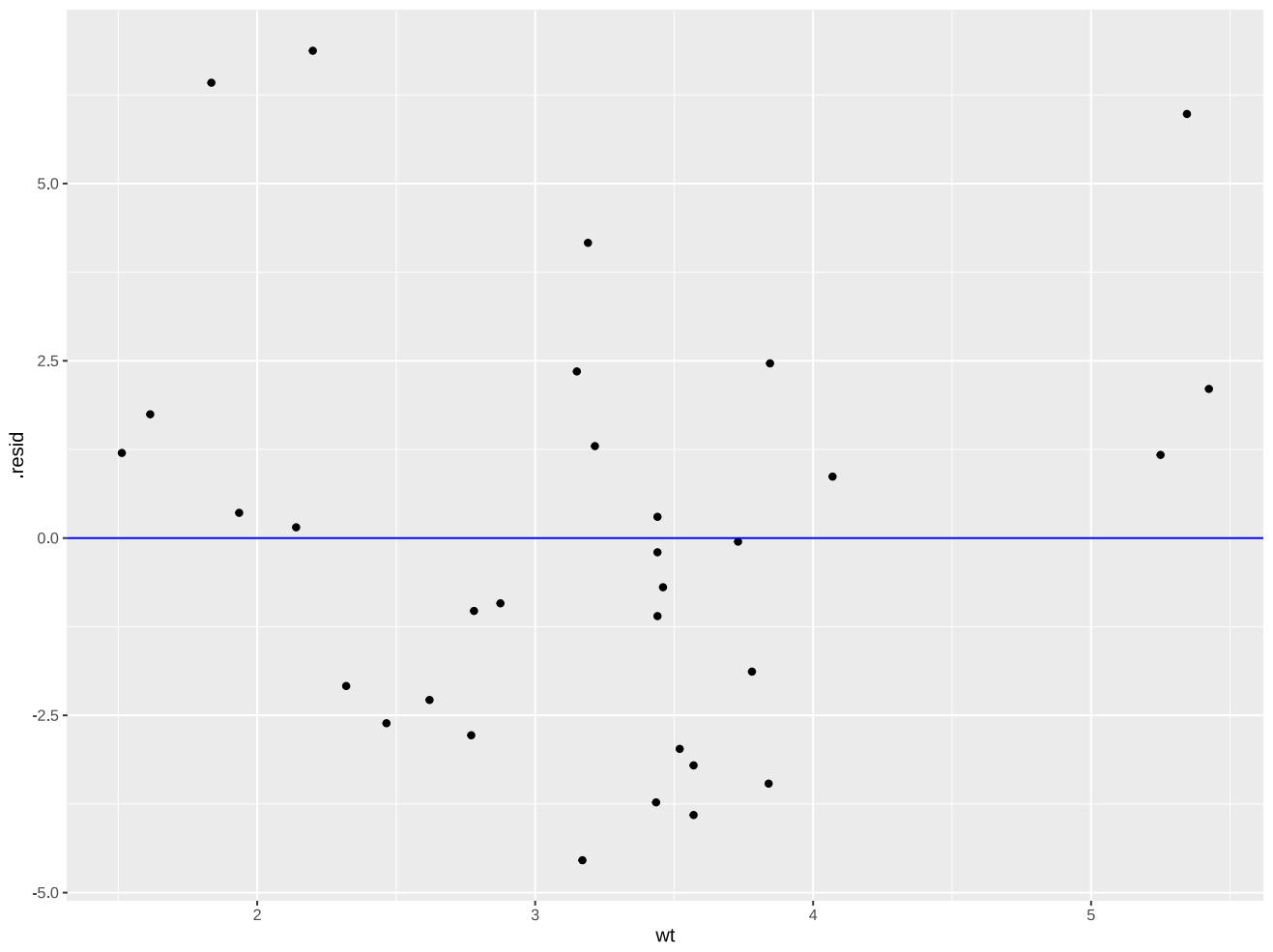
```
model %>%
  augment()
#> # A tibble: 32 x 9
#>   .rownames       mpg    wt .fitted .resid   .hat .sigma .
#>   <chr>         <dbl> <dbl>   <dbl>  <dbl>  <dbl>  <dbl>
#> 1 Mazda RX4      21    2.62    23.3 -2.28  0.0433   3.07 0
#> 2 Mazda RX4 Wag  21    2.88    21.9 -0.920 0.0352   3.09 0
#> 3 Datsun 710     22.8  2.32    24.9 -2.09  0.0584   3.07 0
#> # ... with 29 more rows
```

- 有了这些模型信息，就可以方便地筛选数据或绘图

```
model %>% augment() %>%
  ggplot(aes(x = wt, y = mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted), color = "blue") +
  geom_segment(aes(xend = wt, yend = .fitted),
               color = "red")
```


35
30
25
20
15
10
mpg
2
3
4
5
wt

```r
model %>% augment() %>%
  ggplot(aes(x = wt, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "blue")
```



二. modelr 包辅助建模

modelr 包提供了一系列辅助建模的函数，便于在 tidyverse 框架下辅助建模教学。

1. resample_*(): 重抽样

重抽样，就是反复从数据集中抽取样本形成若干个” 替代” 数据集，用于统计推断或模型性能评估。

常用的重抽样方法有，简单重抽样（留出, Holdout），自助重抽样 (Bootstrap)、交叉验证重抽样 (Cross Validation)、置换重抽样 (Permutation).

- `rsample(data, idx)`: 根据整数向量 `idx` 从数据集 `data` 中重抽样
- `resample_partition(data, p)`: 生成 1 个简单重抽样，即按概率 p 对数据集进行划分，比如划分训练集和测试集
- `resample_bootstrap(data)`: 生成 1 个 bootstrap 重抽样
- `bootstrap(data, n)`: 生成 n 个 bootstrap 重抽样
- `crossv_kfold(data, k)`: 生成 k 折交叉验证重抽样
- `crossv_loo(data)`: 生成留一交叉验证重抽样
- `crossv_mc(data, n, test)`: 按测试集占比 `test`, 生成 n 对蒙特卡罗交叉验证
- `resample_permutation(data, columns)`: 按列 `columns` 生成 1 个置换重抽样
- `permute(data, n, columns)`: 按列 `columns` 生成 n 个置换重抽样

这些重抽样结果：

- 为了避免低效操作数据，都是保存原数据的指针；
- 重抽样数据集都存放在返回结果的列表列，借助 `purrr::map` 函数便于批量建模
- 对每个重抽样数据集，应用 `as.data.frame/as_tibble` 可转化为数据框，可不用转化直接应用于模型函数

另外，`rsample` 包提供了基本工具创建和分析数据集不同类型的重抽样，更适合与机器学习包 `tidymodels` 连用。

2. 模型性能度量函数

- `rmse(model, data)`: 均方根误差
- `mae(model, data)`: 平均绝对误差
- `qae(model, data, probs)`: 分位数绝对误差
- `mape(model, data)`: 平均绝对百分比误差
- `rsae(model, data)`: 绝对误差相对和
- `mse(model, data)`: 均方误差
- `rsquare(model, data)`: R^2

3. **生成模型数据**
   - `seq_range(x, n)`: 根据向量 x 值范围生成等间隔序列
   - `data_grid(data, f1, f2)`: 生成唯一值的所有组合
   - `model_matrix()`: `model.matrix()` 的包装，生成模型（设计）矩阵，特别是用于虚拟变量处理

4. **增加预测值列、残差列**
   - `add_predictions()`
   - `add_residuals()`

```
library(modelr)
ex = resample_partition(mtcars, c(test = 0.3, train = 0.7))
mod = lm(mpg ~ wt, data = ex$train)
rmse(mod, ex$test)
#> [1] 2.51
mod = lm(mpg ~ wt + cyl + vs, data = mtcars)
data_grid(mtcars, wt = seq_range(wt, 10), cyl, vs) %>%
  add_predictions(mod)
#> # A tibble: 60 x 4
#>      wt   cyl    vs  pred
#>   <dbl> <dbl> <dbl> <dbl>
#> 1  1.51     4     0  28.4
#> 2  1.51     4     1  28.9
#> 3  1.51     6     0  25.6
#> # ... with 57 more rows
```

案例：10 折交叉验证

通常将数据集划分为训练集（90%）和测试集（10%），在训练集上训练一个模型，在测试集上评估模型效果。

只这样做一轮的话，模型效果可能具有偶然性，再一个对数据集利用的也不够充分。k 折交叉验证是克服该缺陷的更好做法。

以 10 折交叉验证为例：是将数据集随机分成 10 份，分别以其中 1 份为测试集，其余 9 份为训练集，组成 10 组数据，训练 10 个模型，评估 10 次模型效果，取平均作为最终模型效果。

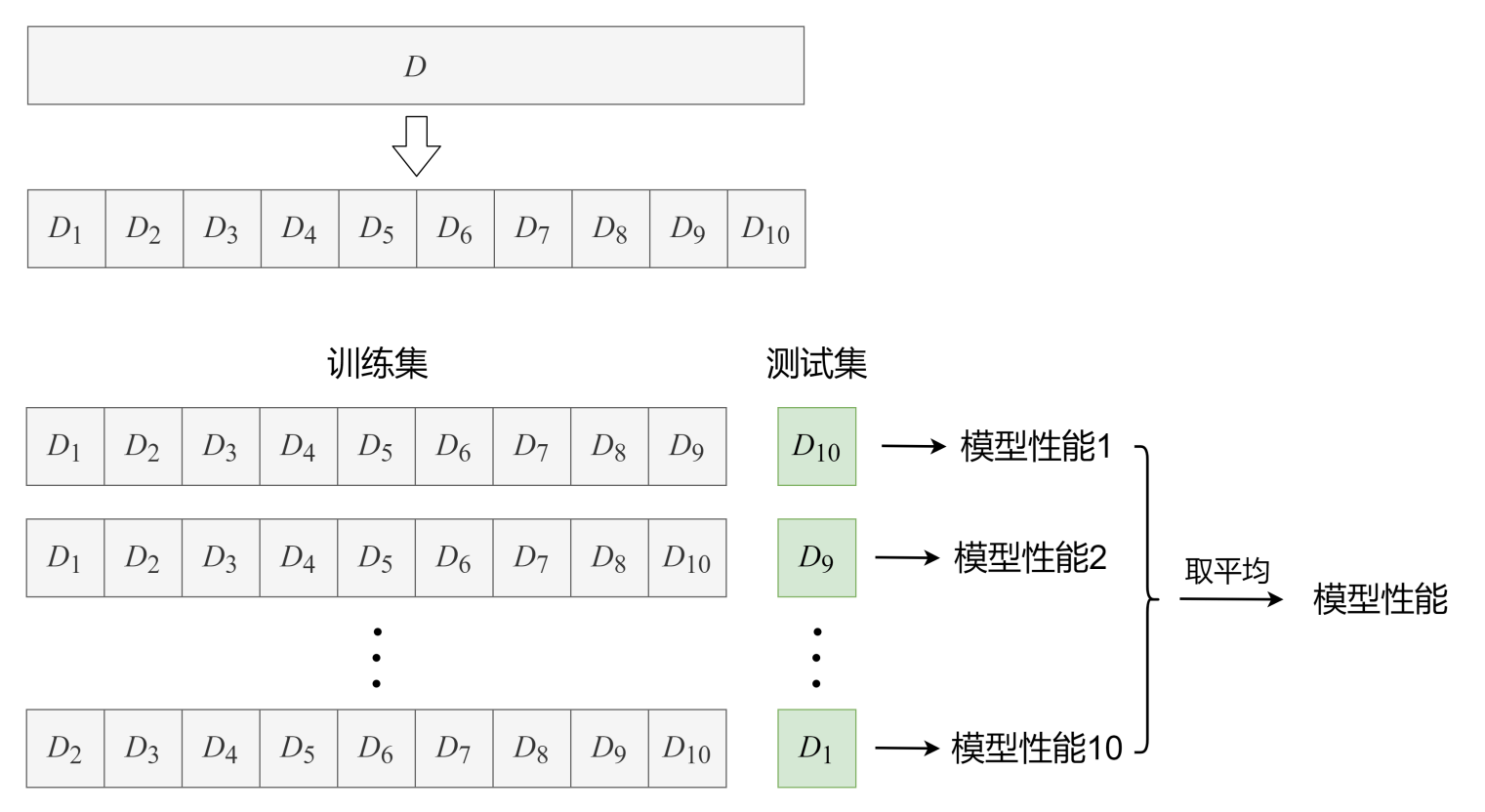


图 1: 10 折交叉验证示意图

- 先用 crossv_kfold() 生成 10 折交叉验证的数据

```
cv10 = crossv_kfold(mtcars, 10)
cv10
#> # A tibble: 10 x 3
#>    train                test                .id
#>    <named list>         <named list>        <chr>
#> 1 <resample [28 x 11]> <resample [4 x 11]> 01
#> 2 <resample [28 x 11]> <resample [4 x 11]> 02
#> 3 <resample [29 x 11]> <resample [3 x 11]> 03
#> # ... with 7 more rows
```

结果为 10 行的嵌套数据框，分别对应交叉组成的 10 组训练集 (train)、测试集 (test) 数据。

- 接着是批量建模，与普通的修改列是一样的：用 `map` 计算新列 + 赋值

```
cv10 %>%
  mutate(models = map(train, ~ lm(mpg ~ wt, data = .)),
         rmse = map2_dbl(models, test, rmse))
#> # A tibble: 10 x 5
#>   train                test                .id   models
#>   <named list>         <named list>        <chr> <named li
#> 1 <resample [28 x 11]> <resample [4 x 11]> 01    <lm>
#> 2 <resample [28 x 11]> <resample [4 x 11]> 02    <lm>
#> 3 <resample [29 x 11]> <resample [3 x 11]> 03    <lm>
#> # ... with 7 more rows
```

注：要计算最终的平均模型效果，对 `rmse` 列做汇总均值即可。

三. 批量建模

对数据做分组，批量地对每个分组建立同样模型，并提取和使用批量的模型结果，这就是批量建模。

批量建模通常是作为探索性数据分析的一种手段，批量建立简单模型以理解复杂的数据集。

批量建模“笨方法”是手动写 `for` 循环实现，再手动提取、合并模型结果。`tidyverse` 中的两种优雅、简洁的做法是：

- 用嵌套数据框 + `purrr::map` 实现
- 用 `dplyr` 包的 `rowwise` 技术，具有异曲同工之妙

- ecostats 数据集，整理自国家统计局网站，包含 2001-2017 年我国 31 个省份的人口、居民消费水平、人均 GDP 等

```
load("datas/ecostats.rda")
ecostats
#> # A tibble: 527 x 7
#>   Region  Year Electricity Investment Consumption Populati
#>   <chr>  <int>       <dbl>      <dbl>       <dbl>      <db
#> 1 安徽    2001        360.       893.        2739       61
#> 2 北京    2001        400.      1513.        9057       13
#> 3 福建    2001        439.      1173.        4770       34
#> # ... with 524 more rows
```

1. 利用嵌套数据框 + purrr::map

嵌套数据框（列表列）

想要对每个省份（数据子集）做重复操作

- 先对数据框用 group_nest() 关于分组变量 Region 做分组嵌套，就得到嵌套数据框，每组数据作为数据框嵌套到列表列 data
- 嵌套数据框的每一行是一个分组，表示一个省份的整个时间跨度内的所有观测，而不是某个单独时间点的观测

```
by_region = ecostats %>%
  group_nest(Region)
by_region
#> # A tibble: 31 x 2
#>   Region                data
#>   <chr>  <list<tibble[,6]>>
#> 1 安徽              [17 x 6]
#> 2 北京              [17 x 6]
#> 3 福建              [17 x 6]
#> # ... with 28 more rows
```

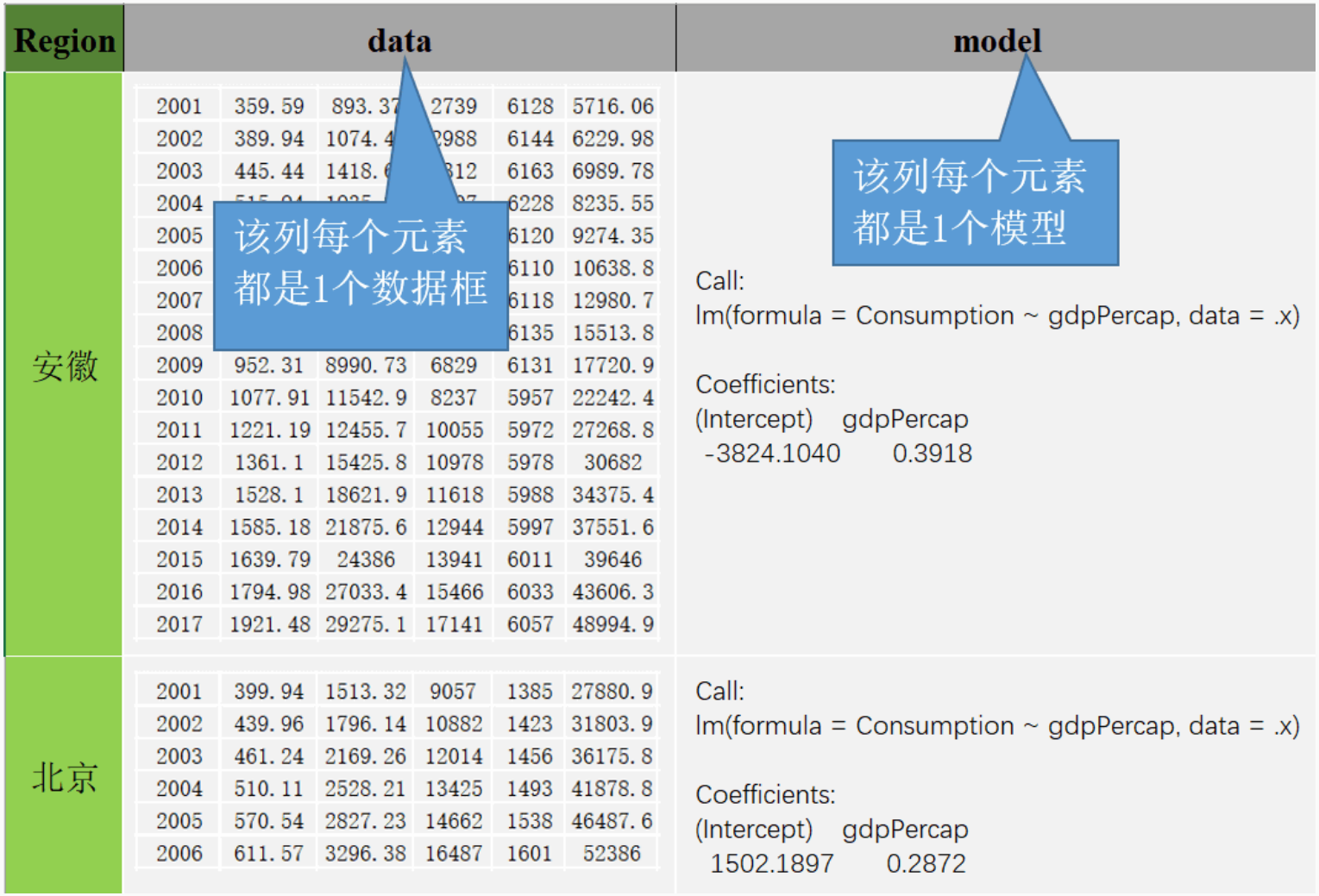

Region
data
model
该列每个元素
都是1个数据框
该列每个元素
都是1个模型
安徽
2001 359.59 893.37 2739 6128 5716.06
2002 389.94 1074.4 2988 6144 6229.98
2003 445.44 1418.6 312 6163 6989.78
2004 6228 8235.55
2005 6120 9274.35
2006 6110 10638.8
2007 6118 12980.7
2008 6135 15513.8
2009 952.31 8990.73 6829 6131 17720.9
2010 1077.91 11542.9 8237 5957 22242.4
2011 1221.19 12455.7 10055 5972 27268.8
2012 1361.1 15425.8 10978 5978 30682
2013 1528.1 18621.9 11618 5988 34375.4
2014 1585.18 21875.6 12944 5997 37551.6
2015 1639.79 24386 13941 6011 39646
2016 1794.98 27033.4 15466 6033 43606.3
2017 1921.48 29275.1 17141 6057 48994.9
Call:
lm(formula = Consumption ~ gdpPercap, data = .x)
Coefficients:
(Intercept) gdpPercap
-3824.1040 0.3918
北京
2001 399.94 1513.32 9057 1385 27880.9
2002 439.96 1796.14 10882 1423 31803.9
2003 461.24 2169.26 12014 1456 36175.8
2004 510.11 2528.21 13425 1493 41878.8
2005 570.54 2827.23 14662 1538 46487.6
2006 611.57 3296.38 16487 1601 52386
Call:
lm(formula = Consumption ~ gdpPercap, data = .x)
Coefficients:
(Intercept) gdpPercap
1502.1897 0.2872

图 2: 嵌套数据框示例

```
by_region$data[[1]]      # 查看列表列的第 1 个元素的内容
#> # A tibble: 17 x 6
#>    Year Electricity Investment Consumption Population gdpP
#>   <int>       <dbl>      <dbl>       <dbl>      <dbl>
#> 1  2001        360.       893.        2739       6128
#> 2  2002        390.      1074.        2988       6144
#> 3  2003        445.      1419.        3312       6163
#> # ... with 14 more rows
```

```
unnest(by_region, data)  # 解除嵌套，还原到原数据
#> # A tibble: 527 x 7
#>   Region  Year Electricity Investment Consumption Populati
#>   <chr>  <int>       <dbl>      <dbl>       <dbl>      <db
#> 1 安徽    2001        360.       893.        2739       61
#> 2 安徽    2002        390.      1074.        2988       61
#> 3 安徽    2003        445.      1419.        3312       61
#> # ... with 524 more rows
```

嵌套数据框与普通数据框一样操作，比如 filter() 筛选行、mutate() 修改列。

(2) 批量建模

- 对嵌套的 data 列，用 mutate() 修改列，增加一列模型列 model，存放每个省份对应的 data 拟合人均消费水平对人均 GDP 的线性回归模型，这就实现了批量建模

```
by_region = by_region %>%
  mutate(model = map(data,
                     ~ lm(Consumption ~ gdpPercap, .x)))
by_region
#> # A tibble: 31 x 3
#>   Region                data model
#>   <chr>  <list<tibble[,6]>> <list>
#> 1 安徽              [17 x 6] <lm>
#> 2 北京              [17 x 6] <lm>
#> 3 福建              [17 x 6] <lm>
#> # ... with 28 more rows
```

- 继续用 mutate() 修改列，借助 map_* 函数从模型列、数据列计算均方根误差、R^2、斜率、p 值：

```
by_region %>%
  mutate(rmse = map2_dbl(model, data, rmse),
         rsq = map2_dbl(model, data, rsquare),
         slope = map_dbl(model, ~ coef(.x)[[2]]),
         pval = map_dbl(model, ~ glance(.x)$p.value))
#> # A tibble: 31 x 7
#>   Region                data model   rmse   rsq slope     p
#>   <chr>  <list<tibble[,6]>> <list> <dbl> <dbl> <dbl>    <d
#> 1 安徽             [17 x 6] <lm>    185. 0.998 0.327 2.36e
#> 2 北京             [17 x 6] <lm>   2005. 0.975 0.392 1.71e
#> 3 福建             [17 x 6] <lm>    415. 0.996 0.287 2.20e
#> # ... with 28 more rows
```

也可以配合 broom 包的函数 tidy(), glance(), augment() 批量、整洁地提取模型结果，这些结果仍是嵌套的列表列，若要完整地显示出来，需要借助 unnest() 解除嵌套。

- 批量提取模型系数估计及其统计量

```
by_region %>%
  mutate(result = map(model, tidy)) %>%
  select(Region, result) %>%
  unnest(result)
#> # A tibble: 62 x 6
#>    Region term         estimate  std.error statistic  p.val
#>    <chr>  <chr>           <dbl>      <dbl>     <dbl>    <db
#> 1 安徽   (Intercept)   942.       89.4         10.5  2.47e-
#> 2 安徽   gdpPercap       0.327     0.00340     96.2  2.36e-
#> 3 北京   (Intercept) -3824.     1301.          -2.94 1.01e-
#> # ... with 59 more rows
```

- 批量提取模型诊断信息

```
by_region %>%
  mutate(result = map(model, glance)) %>%
  select(Region, result) %>%
  unnest(result)
#> # A tibble: 31 x 13
#>   Region r.squared adj.r.squared sigma statistic  p.value
#>   <chr>      <dbl>         <dbl> <dbl>     <dbl>    <dbl>
#> 1 安徽       0.998         0.998  197.     9260. 2.36e-22
#> 2 北京       0.975         0.974 2134.      597. 1.71e-13
#> 3 福建       0.996         0.996  441.     3713. 2.20e-19
#> # ... with 28 more rows, and 4 more variables: BIC <dbl>,
#> #   df.residual <int>, nobs <int>
```

- 批量增加预测值列、残差列等

```
by_region %>%
  mutate(result = map(model, augment)) %>%
  select(Region, result) %>%
  unnest(result)
#> # A tibble: 527 x 9
#>   Region Consumption gdpPercap .fitted .resid  .hat .sigma
#>   <chr>        <dbl>     <dbl>   <dbl>  <dbl> <dbl>  <dbl>
#> 1 安徽          2739     5716.   2811. -72.5  0.140   203.
#> 2 安徽          2988     6230.   2980.   8.49 0.135   204.
#> 3 安徽          3312     6990.   3228.  84.0  0.128   203.
#> # ... with 524 more rows
```

2. 利用 dplyr 包的 rowwise 技术

rowwise 按行方式，可以理解为一种特殊的分组：每一行作为一组。

- 若对 ecostats 数据框用 nest_by() 做嵌套就得到这样 rowwise 化的嵌套数据框

```
by_region = ecostats %>%
  nest_by(Region)
by_region    # 注意多了 Rowwise: Region 信息
#> # A tibble: 31 x 2
#> # Rowwise:  Region
#>   Region                 data
#>   <chr>  <list<tibble[,6]>>
#> 1 安徽             [17 x 6]
#> 2 北京             [17 x 6]
#> 3 福建             [17 x 6]
#> # ... with 28 more rows
```

一个省份的数据占一行，rowwise 化的逻辑，就是按行操作数据，正好适合逐行地对每个嵌套的数据框建模和提取模型信息。

这些操作是与 mutate() 和 summarise() 连用来实现，前者会保持 rowwise 模式，但需要计算结果的行数保持不变；后者相当于对每行结果做汇总，结果行数可变（变多），不再具有 rowwise 模式。

```r
by_region = by_region %>%
  mutate(model = list(lm(Consumption ~ gdpPercap, data)))
by_region
#> # A tibble: 31 x 3
#> # Rowwise:  Region
#>    Region                data model
#>    <chr>  <list<tibble[,6]>> <list>
#> 1 安徽              [17 x 6] <lm>
#> 2 北京              [17 x 6] <lm>
#> 3 福建              [17 x 6] <lm>
#> # ... with 28 more rows
```

- 直接用 mutate() 修改列，从模型列、数据列计算均方根误差、R^2、斜率、p 值

```
by_region %>%
  mutate(rmse = rmse(model, data),
         rsq = rsquare(model, data),
         slope = coef(model)[[2]],
         pval = glance(model)$p.value)
#> # A tibble: 31 x 7
#> # Rowwise:  Region
#>    Region                data model   rmse   rsq slope     p
#>    <chr>  <list<tibble[,6]>> <list> <dbl> <dbl> <dbl>    <d
#> 1 安徽             [17 x 6] <lm>    185. 0.998 0.327 2.36e
#> 2 北京             [17 x 6] <lm>   2005. 0.975 0.392 1.71e
#> 3 福建             [17 x 6] <lm>    415. 0.996 0.287 2.20e
#> # ... with 28 more rows
```

也可以配合 broom 包的 tidy(), glance(), augment() 批量、整洁地提取模型结果。

- 批量提取模型系数估计及其统计量

```
by_region %>%
  summarise(tidy(model))
#> # A tibble: 62 x 6
#> # Groups:   Region [31]
#>   Region term         estimate  std.error statistic  p.val
#>   <chr>  <chr>           <dbl>      <dbl>     <dbl>    <db
#> 1 安徽   (Intercept)   942.       89.4         10.5  2.47e-
#> 2 安徽   gdpPercap       0.327     0.00340     96.2  2.36e-
#> 3 北京   (Intercept) -3824.    1301.          -2.94 1.01e-
#> # ... with 59 more rows
```

- 批量提取模型诊断信息

```
by_region %>%
  summarise(glance(model))
#> # A tibble: 31 x 13
#> # Groups:   Region [31]
#>   Region r.squared adj.r.squared sigma statistic  p.value
#>   <chr>      <dbl>         <dbl> <dbl>     <dbl>    <dbl>
#> 1 安徽       0.998         0.998  197.     9260. 2.36e-22
#> 2 北京       0.975         0.974 2134.      597. 1.71e-13
#> 3 福建       0.996         0.996  441.     3713. 2.20e-19
#> # ... with 28 more rows, and 4 more variables: BIC <dbl>,
#> #   df.residual <int>, nobs <int>
```

- 批量增加预测值列、残差列等

```
by_region %>%
  summarise(augment(model))
#> # A tibble: 527 x 9
#> # Groups:   Region [31]
#>   Region Consumption gdpPercap .fitted .resid  .hat .sigma
#>   <chr>        <dbl>     <dbl>   <dbl>  <dbl> <dbl>  <dbl>
#> 1 安徽          2739     5716.   2811. -72.5  0.140   203.
#> 2 安徽          2988     6230.   2980.   8.49 0.135   204.
#> 3 安徽          3312     6990.   3228.  84.0  0.128   203.
#> # ... with 524 more rows
```

注：rowwise 行化方法的代码更简洁，但速度不如嵌套数据框 + purrr::map 快。

3. (分组) 滚动回归

金融时间序列数据分析中常用到滚动回归，这是滑窗迭代与批量建模的结合：对数据框按时间窗口滑动，在各个滑动窗口批量地构建回归模型并提取模型结果。

借助 `slider` 包很容易实现。这里看一个更进一步的案例：分组滚动回归。

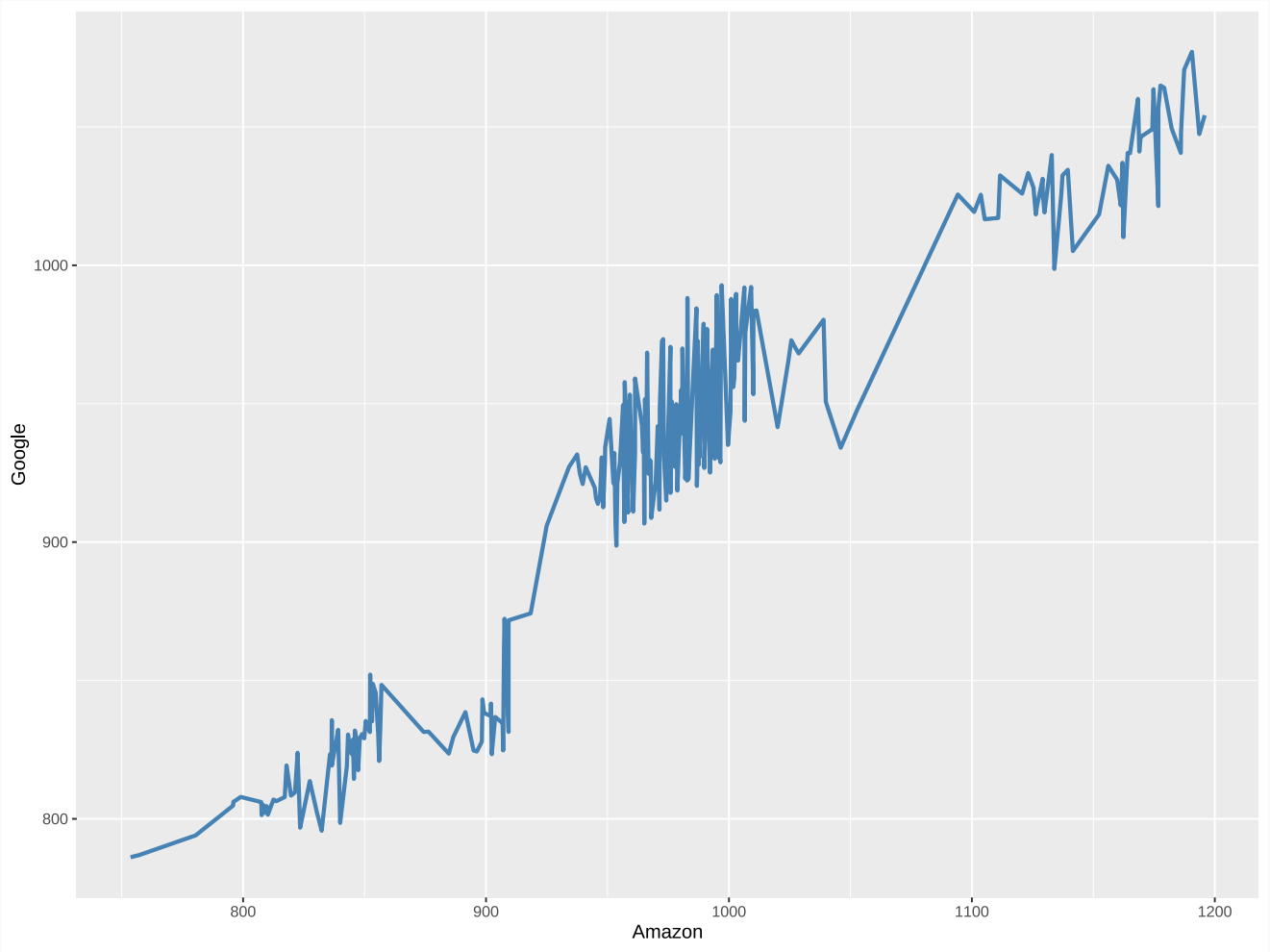
```
library(lubridate)
library(slider)
```

用 stocks 股票数据，它是整洁的长表，但这里要做股票之间的线性回归，先长变宽，再根据日期列计算一个 season 列用于分组：

```
load("datas/stocks.rda")
df = stocks %>%
  pivot_wider(names_from = Stock, values_from = Close) %>%
  mutate(season = quarter(Date))
df
#> # A tibble: 251 x 5
#>   Date       Google Amazon Apple season
#>   <date>      <dbl>  <dbl> <dbl>  <int>
#> 1 2017-01-03   786.   754.  116.      1
#> 2 2017-01-04   787.   757.  116.      1
#> 3 2017-01-05   794.   780.  117.      1
#> # ... with 248 more rows
```

- 绘图可以看出 Amazon 与 Google 股票是大致符合线性关系的

```
df %>%
  ggplot(aes(Amazon, Google)) +
  geom_line(color = "steelblue", size = 1.1)
```



- 因此，考虑对这两支股票做滚动线性回归是合理的，再加入分组操作逻辑：分别对每个季度做 5 步滚动线性回归，当然也离不开 slide() 滑窗迭代。

```
df_roll = df %>%
  group_by(season) %>%
  mutate(models = slide(cur_data(),
                        ~ lm(Google ~ Amazon, .x),
                        .before = 2, .after = 2,
                        .complete = TRUE)) %>%
  ungroup()
```

代码解释

(1) slide() 第 1 个参数 cur_data() 是专门与 group_by() 搭配使用的，代表当前分组的数据框，要对它做滑窗，窗口大小用.before=2, .after=2 控制，.complete=TRUE 表示只留完整窗口，忽略首尾宽度不够的窗口（补 NULL）；

(2) ~ lm(Google ~ Amazon, .x) 是用于每个滑动窗口的函数（purrr 公式写法），每个窗口是个 5 行的数据框，自变量.x 就对应它，在其上按公式 Google ~ Amazon 建立线性回归模型；

(3) 数据框有几行，就有几个滑窗数据框（包括 NULL），就构建几个线性回归模型，所以正好作为一列，赋给 models.

```
df_roll
#> # A tibble: 251 x 6
#>   Date       Google Amazon Apple season models
#>   <date>      <dbl>  <dbl> <dbl>  <int> <list>
#> 1 2017-01-03   786.   754.  116.      1 <NULL>
#> 2 2017-01-04   787.   757.  116.      1 <NULL>
#> 3 2017-01-05   794.   780.  117.      1 <lm>
#> # ... with 248 more rows
```

- 剩下的事情，就是从模型对象构成的列表列，提取想要的模型信息，比如回归系数、残差标准误、R 方等
- 这里采用前文 `map + broom` 包整洁模型结果来提取，注意，需要先把 `models` 列首尾为 `NULL` 的行先过滤掉

```
df_roll %>%
  filter(!map_lgl(models, is.null)) %>%
  mutate(rsq = map_dbl(models, ~ glance(.x)$r.squared),
         sigma  = map_dbl(models, ~ glance(.x)$sigma),
         slope = map_dbl(models, ~ tidy(.x)$estimate[2]))
#> # A tibble: 235 x 9
#>   Date       Google Amazon Apple season models   rsq sigma
#>   <date>      <dbl>  <dbl> <dbl>  <int> <list> <dbl> <dbl>
#> 1 2017-01-05   794.   780.  117.      1 <lm>   0.957 2.41
#> 2 2017-01-06   806.   796.  118.      1 <lm>   0.953 2.22
#> 3 2017-01-09   807.   797.  119.      1 <lm>   0.992 0.569
#> # ... with 232 more rows
```

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

参考文献

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*.