时间序列分析与预测

第四讲

黄嘉平

深圳大学｜中国经济特区研究中心
粤海校区汇文楼办公楼 1510
课程网站 https://huangjp.com/TSAF/

1. 时间序列的分解

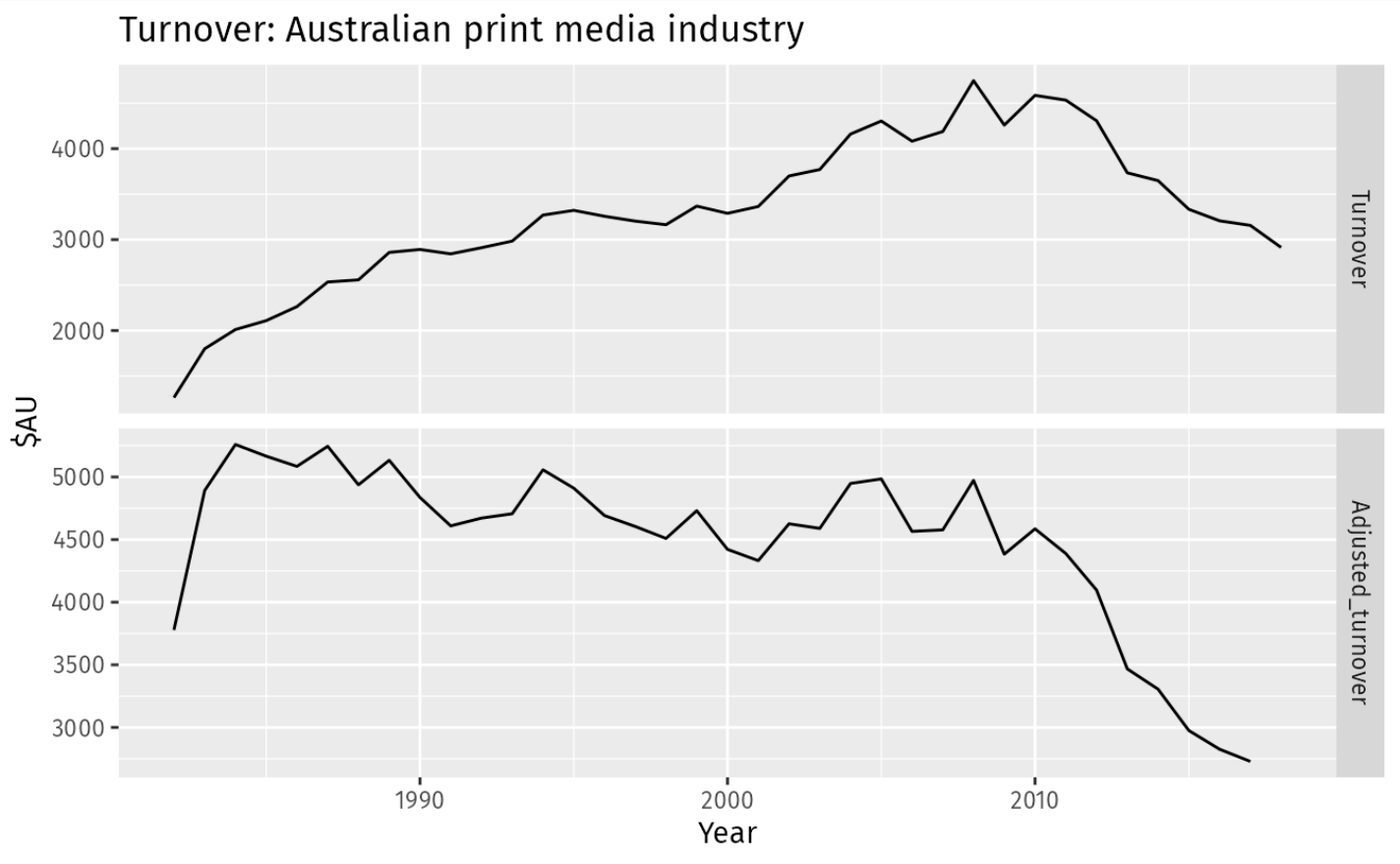
1.1. 数据变换与调整

我们往往需要对历史数据进行**调整**以使其简化，方便分析。调整方法通常分为日历调整、人口调整、通货膨胀调整、数学变换。这样做可以消除波动的来源，使数据特征更加一致，方便后续的建模和预测。

- 日历调整（calender adjustment）：例如月度销售额受到该月份日数的影响，而月度日均销售额则可以消除这种影响。
- 人口调整（population adjustment）：任何受到人口总数变动影响的数据都可以调整为人均值，例如地区内医院的床位数显然会随常住人口的增加而增加。
- 通货膨胀调整（inflation adjustment）：以货币衡量的数据在分析前应调整通货膨胀的影响。例如 GDP 数据分为**现价**和**不变价**，前者是以数据发生时的价格衡量的，后者则是以某个**固定基期**（例如 2000 年）的价值衡量的。

1.1. 数据变换与调整

澳大利亚出版业营业额（澳元），下图用 2010 年的 CPI 作为基准（=100）进行调整



1.1. 数据变换与调整

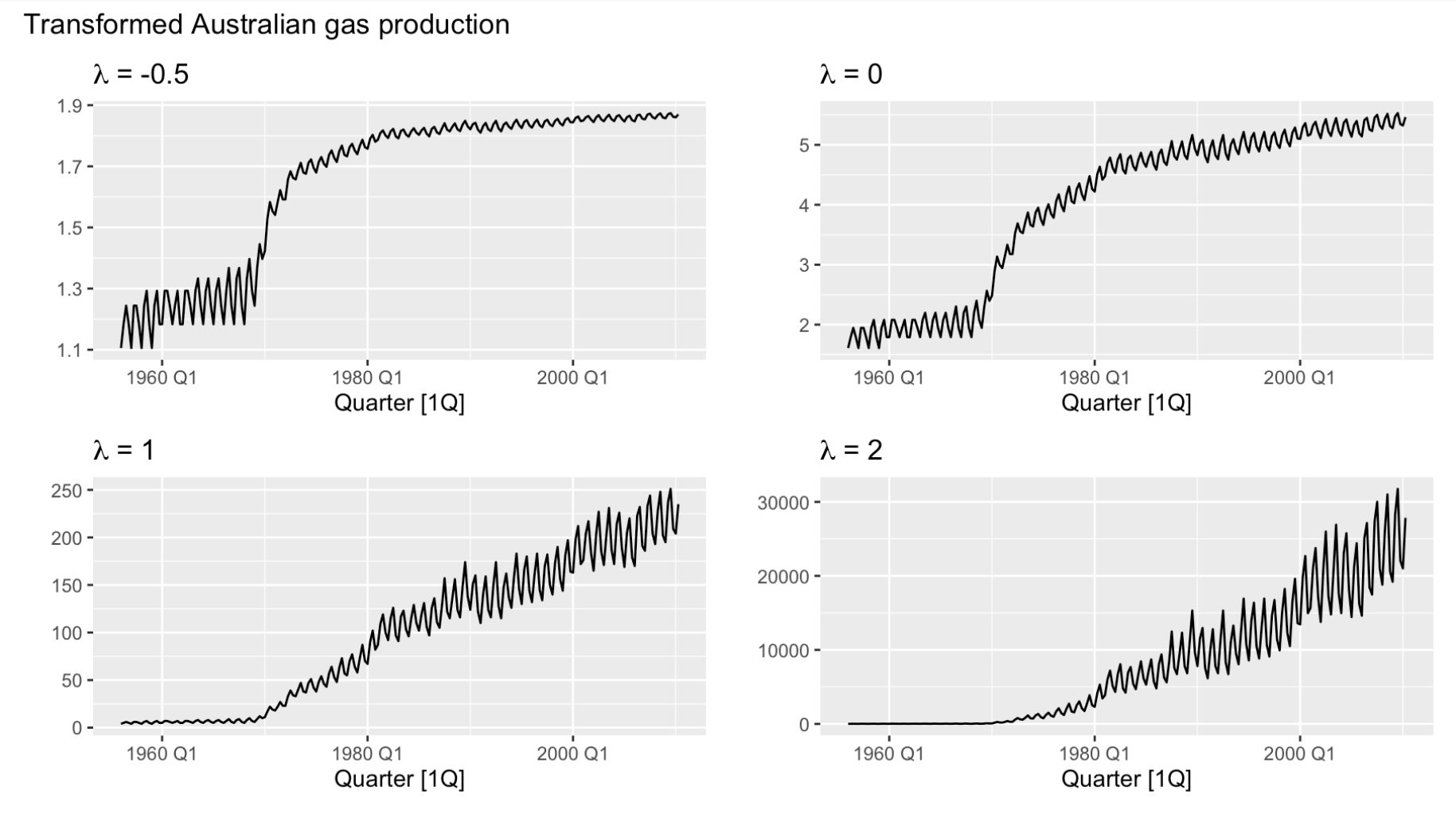
- 数学变换（mathematical transformation）
  - 对数变换：$w_t = \log(y_t)$
  - 指数变换：$w_t = y_t^p$
  - Box-Cox 变换

$$\text{Box \& Cox (1964):}\quad w_t = \begin{cases} \log(y_t) & \text{if } \lambda = 0 \\ (y_t^\lambda - 1)/\lambda & \text{otherwise} \end{cases}$$

$$\text{Bickel \& Doksum (1981):}\quad w_t = \begin{cases} \log(y_t) & \text{if } \lambda = 0 \\ (\text{sign}(y_t)|y_t|^\lambda - 1)/\lambda & \text{otherwise} \end{cases}$$

Box, G. E. P., & Cox, D. R. (1964). An analysis of transformations. *J. Roy. Statist. Soc. B,* 26(2), 211–252.
Bickel, P. J., & Doksum, K. A. (1981). An analysis of transformations revisited. *JASA,* 76(374), 296–311.

1.1. 数据变换与调整


Transformed Australian gas production
λ = -0.5
1.9
1.7
1.5
1.3
1.1
1960 Q1
1980 Q1
2000 Q1
Quarter [1Q]
λ = 0
5
4
3
2
1960 Q1
1980 Q1
2000 Q1
Quarter [1Q]
λ = 1
250
200
150
100
50
0
1960 Q1
1980 Q1
2000 Q1
Quarter [1Q]
λ = 2
30000
20000
10000
0
1960 Q1
1980 Q1
2000 Q1
Quarter [1Q]

1.2. 时间序列的成分

最传统的时间序列分析方法之一是假设数据中包含不同的成分（components），每个成分都会产生独特的变化特征。常见的成分有季节（seasonal）、趋势（trend）、周期（cycle）等。我们也经常将趋势和周期合并为一个成分（trend-cycle）。

如果能将数据按成分分解，不仅能加深我们对数据的理解，也可以增加预测的精确度。

令 S_t 为季节成分，T_t 为趋势周期成分，R_t 为剩余成分，则可以有以下两种模型

- 加法模型：$y_t = S_t + T_t + R_t$
- 乘法模型：$y_t = S_t \times T_t \times R_t$

乘法模型也可以通过对数变换转为加法模型，即

$$y_t = S_t \times T_t \times R_t \qquad \Longleftrightarrow \qquad \log(y_t) = \log(S_t) + \log(T_t) + \log(R_t)$$

1.2. 时间序列的成分

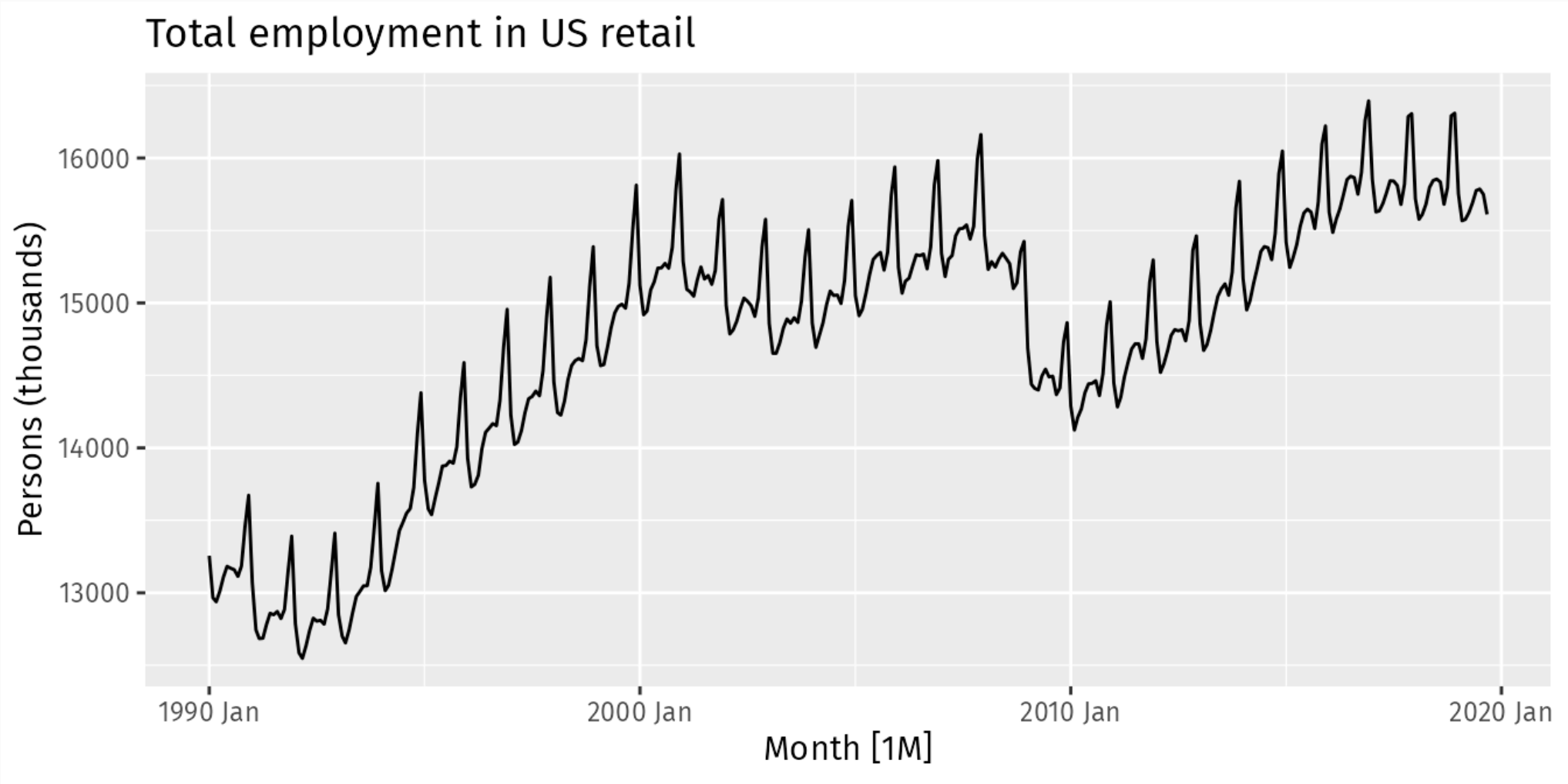
下面的程序从 fpp3 包中包含的数据集 `us_employment` 中提取 1990 年以后的零售业就业人数数据，并保存在 `us_retail_employment` 中。

```
us_retail_employment <- us_employment |>
  filter(year(Month) >= 1990, Title == "Retail Trade") |>
  select(-Series_ID)
```

接下来画出就业人数的时序图。

```
autoplot(us_retail_employment, Employed) +
  labs(y = "Persons (thousands)",
    title = "Total employment in US retail")
```

1.2. 时间序列的成分


Total employment in US retail
Persons (thousands)
16000
15000
14000
13000
1990 Jan
2000 Jan
2010 Jan
2020 Jan
Month [1M]

1.2. 时间序列的成分

下面是用 STL 法（后面会介绍）对 `us_retail_employment` 进行分解的结果。这里用到了 fabletools 包中的 `model()` 函数和 `components()` 函数。

```
dcmp <- us_retail_employment |> model(stl = STL(Employed))
components(dcmp)
```

```
# A dable: 357 x 7 [1M]
# Key:     .model [1]
# :        Employed = trend + season_year + remainder
   .model    Month Employed  trend season_year remainder season_adjust
   <chr>     <mth>    <dbl>  <dbl>       <dbl>     <dbl>          <dbl>
 1 stl    1990 Jan   13256. 13288.      -33.0      0.836         13289.
 2 stl    1990 Feb   12966. 13269.     -258.      -44.6           13224.
 3 stl    1990 Mar   12938. 13250.     -290.      -22.1           13228.
 4 stl    1990 Apr   13012. 13231.     -220.       1.05           13232.
 5 stl    1990 May   13108. 13211.     -114.      11.3            13223.
```

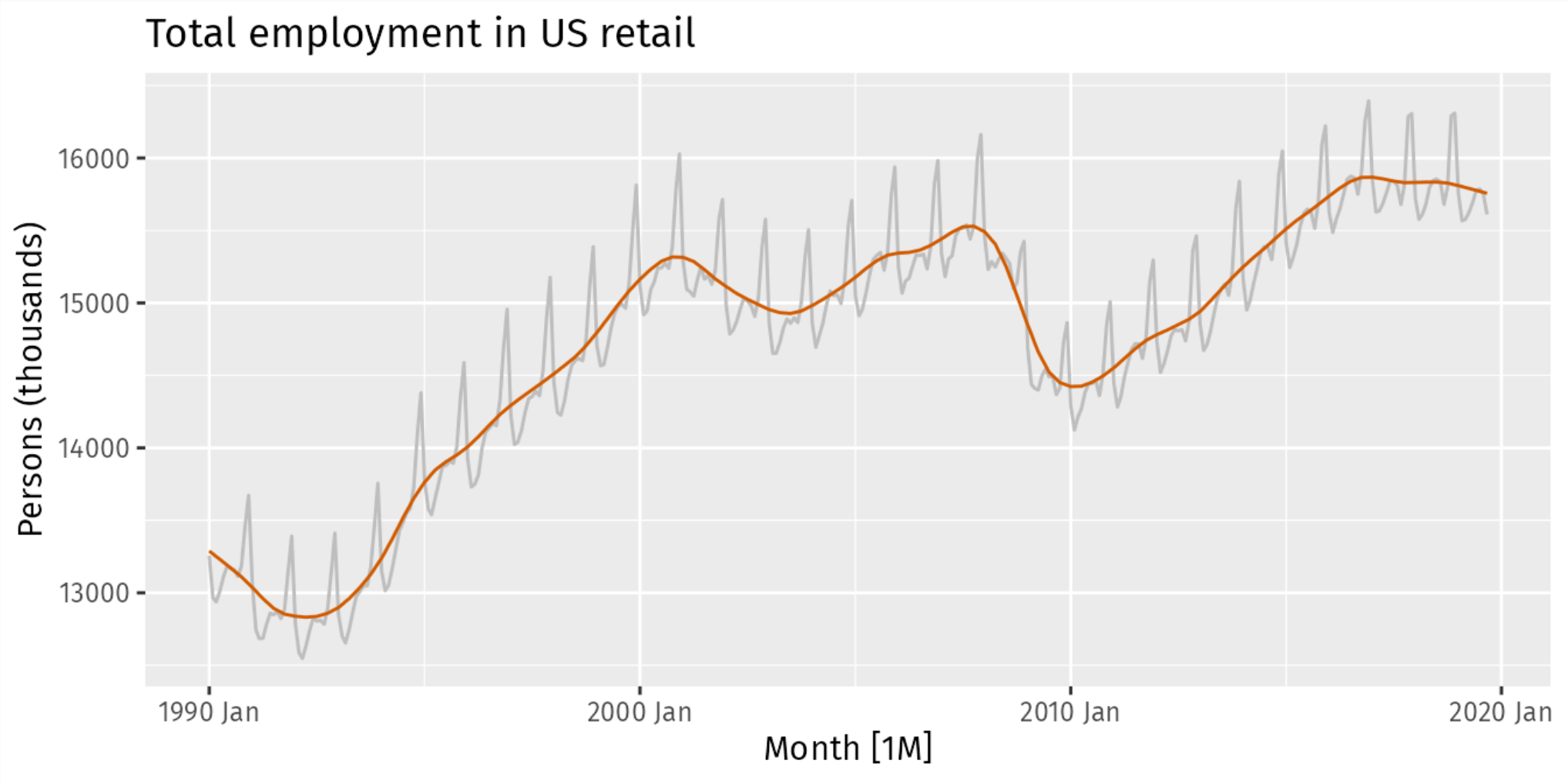
1.2. 时间序列的成分

用 `components()` 输出的结果是一个特殊的数据结构 dable。我们需要把它转换为 tsibble，再用 `autoplot()` 进行绘图。

```
components(dcmp) |>
  as_tsibble() |>
  autoplot(Employed, colour="gray") +
  geom_line(aes(y=trend), colour = "#D55E00") +
  labs(
    y = "Persons (thousands)",
    title = "Total employment in US retail"
  )
```

第四行的 `geom_line()` 将分解后得到的趋势项（变量名为 trend）以折线图的单独绘制，并设定了 RGB 颜色 **#D55E00**。

1.2. 时间序列的成分

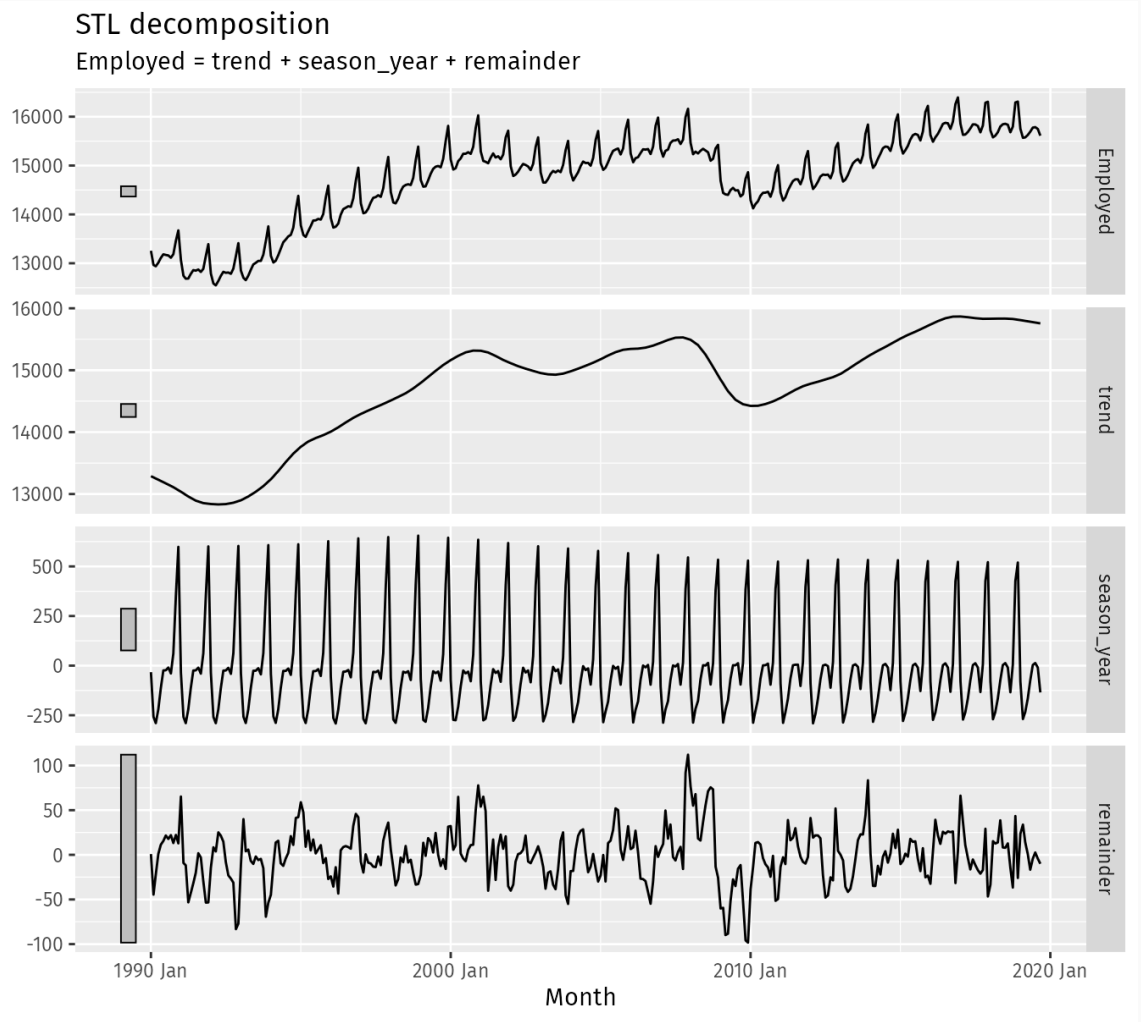

Total employment in US retail
Persons (thousands)
16000
15000
14000
13000
1990 Jan
2000 Jan
2010 Jan
2020 Jan
Month [1M]

1.2. 时间序列的成分

更常见的绘图形式是将原序列和分解后的成分依次绘成时序图，这仅需一个 `autoplot()` 命令即可完成：

```
components(dcmp) |>
  autoplot()
```

注意每个小图左侧的灰柱，它代表相同的取值范围。这样可以修正由于不同成分取值范围不同而造成的视觉误导。



2. 常用的分解方法

2.1. 移动平均法（moving average）

传统的时间序列分解方法（1920s–1950s）首先需要估计趋势周期项，其估计方法称为移动平均法（moving average）。

序列 y_t 的 m **阶移动平均**定义为

$$\hat{T}_t = \frac{1}{m} \sum_{j=-k}^{k} y_{t+j}, \quad m = 2k + 1$$

即在时间 t 估计的趋势周期项是前后 k 期内所有观测值的平均值。我们可以将其记作 m-MA。

右侧的表中展示了澳大利亚的年度出口数据（包含在 `global_economy` 数据集中）和它的 5 阶移动平均。

Year	Exports	5-MA
1960	12.99	
1961	**12.40**	
1962	**13.94**	13.46
1963	**13.01**	**13.50**
1964	**14.94**	13.61
1965	**13.22**	13.40
…	…	…
2012	21.52	20.78
2013	19.99	20.81
2014	21.08	20.37
2015	20.01	20.32
2016	19.25	
2017	21.27	

2.1. 移动平均法（moving average）

下面的程序展示了如何计算移动平均并将结果加入已有的数据中

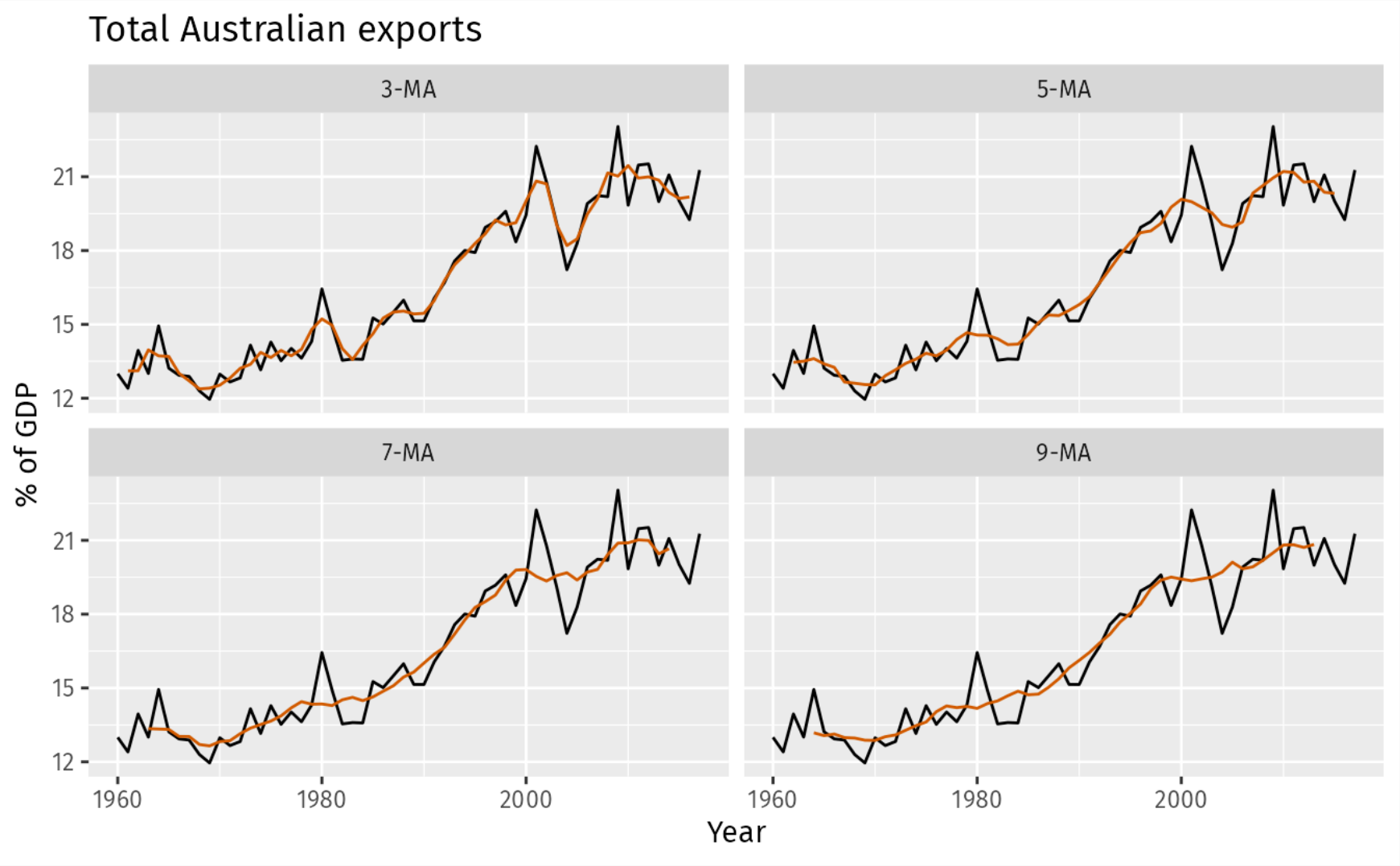
```
aus_exports <- global_economy |>
  filter(Country == "Australia") |>
  mutate(
    `5-MA` = slider::slide_dbl(Exports, mean,
                               .before = 2, .after = 2,
                               .complete = TRUE)
  )
```

这里作者选择了 slider 包中的 `slide_dbl()` 函数。第一个参数是用来计算的时间序列观测值，第二个参数是指定计算方式（这里的 `mean` 是函数名称），第三和第四个参数分别指定每个时间点前后各取几期（这里各取 2 期，因此 $m = 2 \times 2 + 1 = 5$），第五个参数 `.complete` 指定是否需要完整的数据进行计算（默认为 `FALSE`，因此我们需要指定为 `TRUE`）。

2.1. 移动平均法（moving average）

```
aus_exports |>
  autoplot(Exports) +
  geom_line(aes(y = `5-MA`), colour = "#D55E00") +
  labs(y = "% of GDP", title = "Total Australian exports")
```

2.1. 移动平均法（moving average）



2.1. 移动平均法（moving average）

前面介绍的移动平均是对称的，即前后取的期数相同，此时 m 为奇数。如果 m 为偶数，则移动平均为非对称。例如 4-MA 可以写成

$$\frac{1}{4}\sum_{j=-1}^{2} y_{t+j} \quad \text{or} \quad \frac{1}{4}\sum_{j=-2}^{1} y_{t+j}$$

为了使非对称的移动平均变为对称的，我们可以计算**移动平均的移动平均**。例如 4-MA 的 2-MA （也记作 2×4-MA）为

$$\frac{1}{2}\sum_{r=-1}^{0}\left(\frac{1}{4}\sum_{j=-1}^{2} y_{t+j+r}\right) = \frac{1}{2}\sum_{r=0}^{1}\left(\frac{1}{4}\sum_{j=-2}^{1} y_{t+j+r}\right) = \frac{1}{8}y_{t-2} + \frac{1}{4}y_{t-1} + \frac{1}{4}y_t + \frac{1}{4}y_{t+1} + \frac{1}{8}y_{t+2}$$

其他常用的组合包括 3×3-MA, 2×12-MA 等。偶数阶与偶数阶组合或者奇数阶与奇数阶组合可以获得对称的移动平均。

2.1. 移动平均法（moving average）

针对包含季节性特征的数据，用移动平均法估计趋势周期项时要考虑到季节性因素的影响。例如针对季度数据，如果用 5-MA，则 y_{t-2} 和 y_{t+2} 为同一季节，因此这个季节的影响会被放大。此时我们可以采用 2×4-MA，即

$$\hat{T}_t = \frac{1}{8}y_{t-2} + \frac{1}{4}y_{t-1} + \frac{1}{4}y_t + \frac{1}{4}y_{t+1} + \frac{1}{8}y_{t+2}$$

以保证每个季节的权重相同。

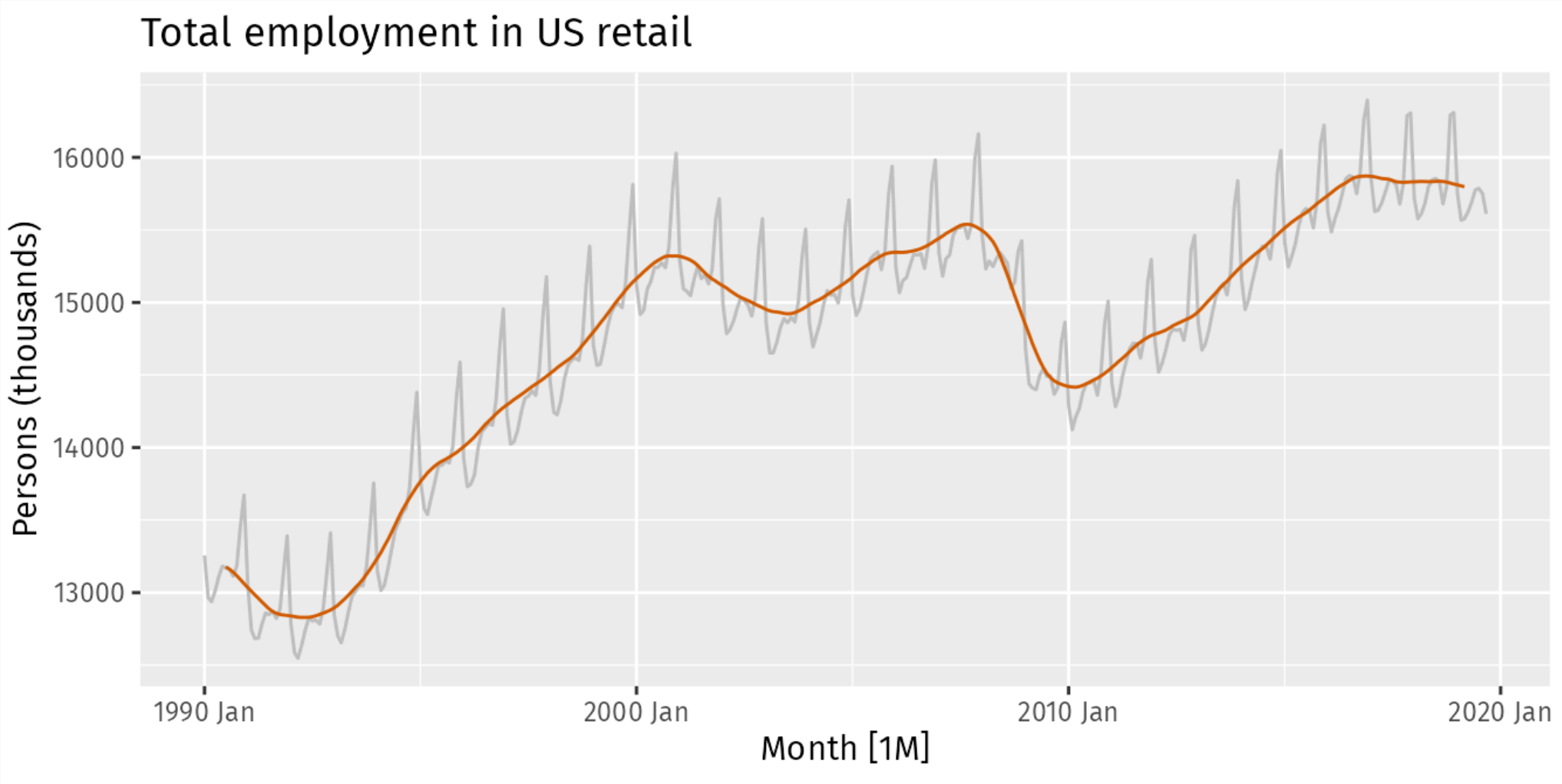
针对月度数据中的年周期季节性，则可以使用 2×12-MA。针对日度数据中的周周期（7 天周期）季节性，简单的 7-MA 就可以胜任。

2.1. 移动平均法（moving average）

以美国零售业就业人数数据为例

```
us_retail_employment_ma <- us_retail_employment |>
  mutate(
    `12-MA` = slider::slide_dbl(Employed, mean,
                                .before = 5, .after = 6,
                                .complete = TRUE),
    `2x12-MA` = slider::slide_dbl(`12-MA`, mean,
                                  .before = 1, .after = 0,
                                  .complete = TRUE)
  )
us_retail_employment_ma |>
  autoplot(Employed, colour = "gray") +
  geom_line(aes(y = `2x12-MA`), colour = "#D55E00") +
  labs(y = "Persons (thousands)", title = "Total employment in US retail")
```

2.1. 移动平均法（moving average）



2.1. 移动平均法（moving average）

加权移动平均

2×4-MA 可以看作在 5-MA 中添加了权重 $\left(\frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}\right)$。一般情况下，令 $(a_{-k}, \ldots, a_k)$ 代表权重向量，$m = 2k + 1$，则加权 m-MA 定义为

$$\hat{T}_t = \sum_{j=-k}^{k} a_j y_{t+j}, \quad \text{s.t.} \quad \sum_{j=-k}^{k} a_j = 1 \text{ and } a_j = a_{-j}$$

由此可见，单纯的移动平均是加权移动平均的一种特殊形式。

2.2. 传统分解法（classic decomposition）

传统分解法起源自 1920 年代，可用于加法模型和乘法模型。传统分解法假设**季节项不随时间变化而变化**。

加法分解：

Step 1. 如果季节性周期 m 为偶数，用 $2\times m$-MA 估计趋势周期项 $\hat{T}_t$；如果 m 为奇数，则用 m-MA 估计 $\hat{T}_t$。

Step 2. 计算去趋势后的数据 $y_t - \hat{T}_t$。

Step 3. 针对每一个季节，从去趋势后的数据中选取该季节所有观测值并取平均，然后将得出的结果按总和为零的方式进行调整，最后将调整后的值按照季节先后排列并逐年重复，即可得到季节项的估计值 $\hat{S}_t$。

Step 4. 剩余项的估计值为 $\hat{R}_t = y_t - \hat{T}_t - \hat{S}_t$。

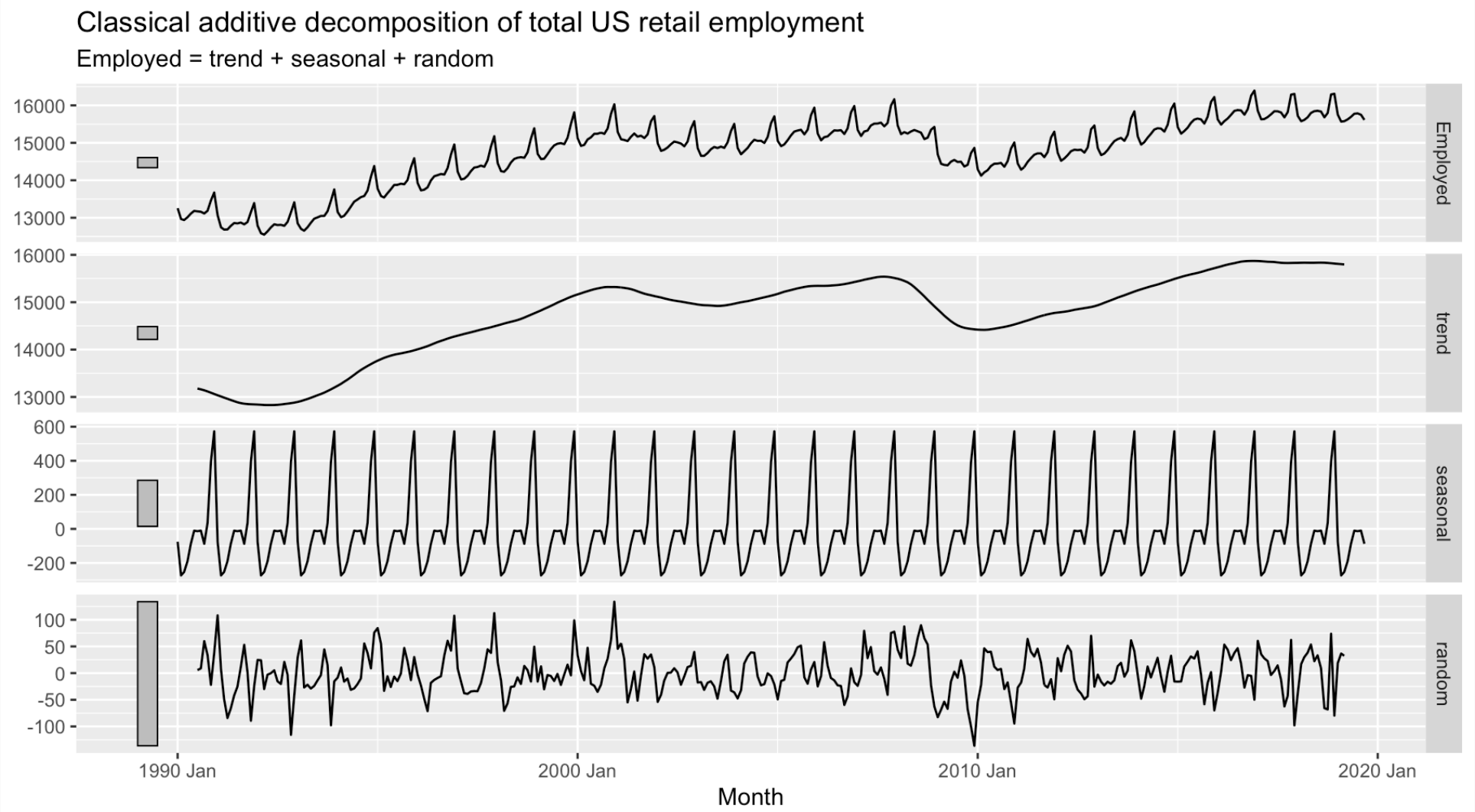
2.2. 传统分解法（classic decomposition）

下面的程序将美国零售业就业人数数据用传统分解法进行了加法分解并绘图。

```
us_retail_employment |>
  model(
    classical_decomposition(Employed, type = "additive")
  ) |>
  components() |>
  autoplot() +
  labs(title = "Classical additive decomposition of total US retail
  employment")
```

注意，`model()` 中的方法采用了 `classical_decomposition`，其参数 `type = "additive"` 指定了加法模型。

2.2. 传统分解法（classic decomposition）



2.2. 传统分解法（classic decomposition）

乘法分解的步骤和加法分解类似，只是将加法模型改为乘法模型，即 Step 2. 中的去趋势数据为 $y_t / \hat{T}_t$，Step 4. 中剩余项的估计值为 $\hat{R}_t = y_t / \left(\hat{T}_t \hat{S}_t\right)$。

传统分解法的问题：

- 趋势周期项最前面和最后面的几个估计值是缺失的。这是因为移动平均法无法计算这些值。同时这也导致了剩余项估计值的缺失。
- 趋势周期项的估计有过度平滑的倾向。
- 季节项不随时间变化而变化的假设有时并不切合实际。例如在 Box-Cox 变换一节展示的澳大利亚天然气产量数据。
- 面对异常值时的稳健性（robustness）较差。

因此，虽然传统分解法依然有广泛应用，但并不推荐大家用。这和宏观经济学中的 Cobb-Douglas 生产函数类似，仅应作为其他模型的对比对象。

2.3. 各国统计部门使用的方法

各国统计部门负责发布该国的官方宏观统计数据。由于宏观数据基本都是月度、季度和年度，统计部门需要针对月度和季度数据设计专门的年周期季节项估计方法。

著名的估计方法包括美国人口普查局（the US Census Bureau）开发的 X-11 和后续版本 X-12-ARIMA，以及西班牙银行（Bank of Spain）开发的 TRAMO/SEATS。现在被广泛应用的方法是结合两种方法的 **X-13ARIMA-SEATS**。我们需要安装 seasonal 工具包才能使用 X-13ARIMA-SEATS。

鉴于西方国家开发的方法无法正确处理我国特有的季节性特征（例如春节、五一、十一等假期带来的影响），国家统计局与南开大学合作开发了国家统计局版季节调整软件 NBS-SA，从 2010 年起开始使用。（👉 国家统计局：什么是季节调整？）

而中国人民银行也于 2005 年开发了 PBC 版 X-12-ARIMA 季节调整软件用于人民银行系统的使用。

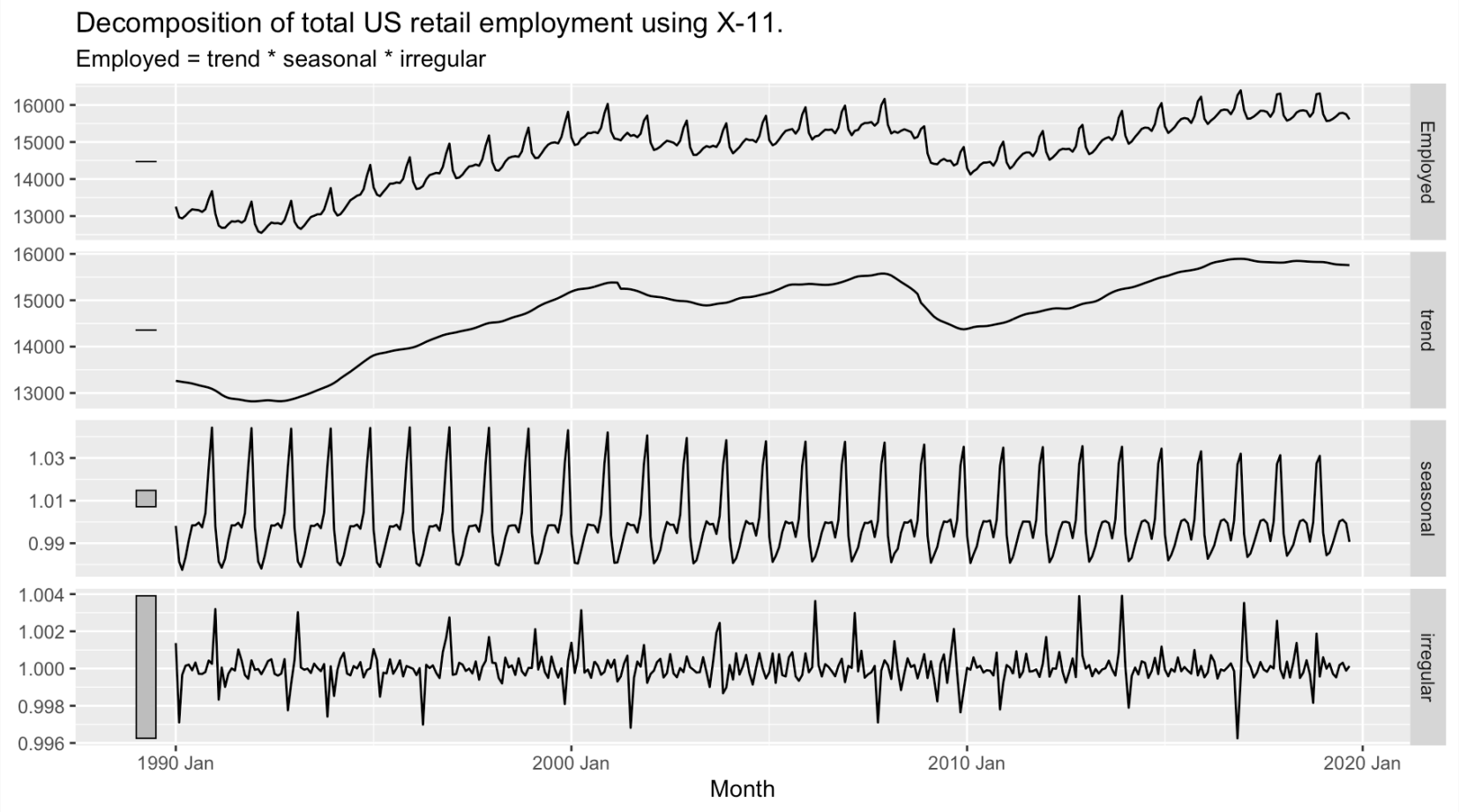
2.3. 各国统计部门使用的方法

X-11 是由美国人口普查局开发，之后由加拿大统计局（Statistics Canada）进一步完善。它改善了传统分解法的缺点，包括可以估计全区间的趋势周期项，以及允许季节项缓慢的变化。同时它还考虑了每月股票市场交易日数、假期、以及其他已知因素对季节性的影响。该方法完全自动（无需手动设定参数）且对异常值稳健。

下面是用 X-11 进行分解的程序，需要事先安装 seasonal 包。

```
library(seasonal)
x11_dcmp <- us_retail_employment |>
  model(x11 = X_13ARIMA_SEATS(Employed ~ x11())) |>
  components()
autoplot(x11_dcmp) +
  labs(title =
       "Decomposition of total US retail employment using X-11.")
```

2.3. 各国统计部门使用的方法

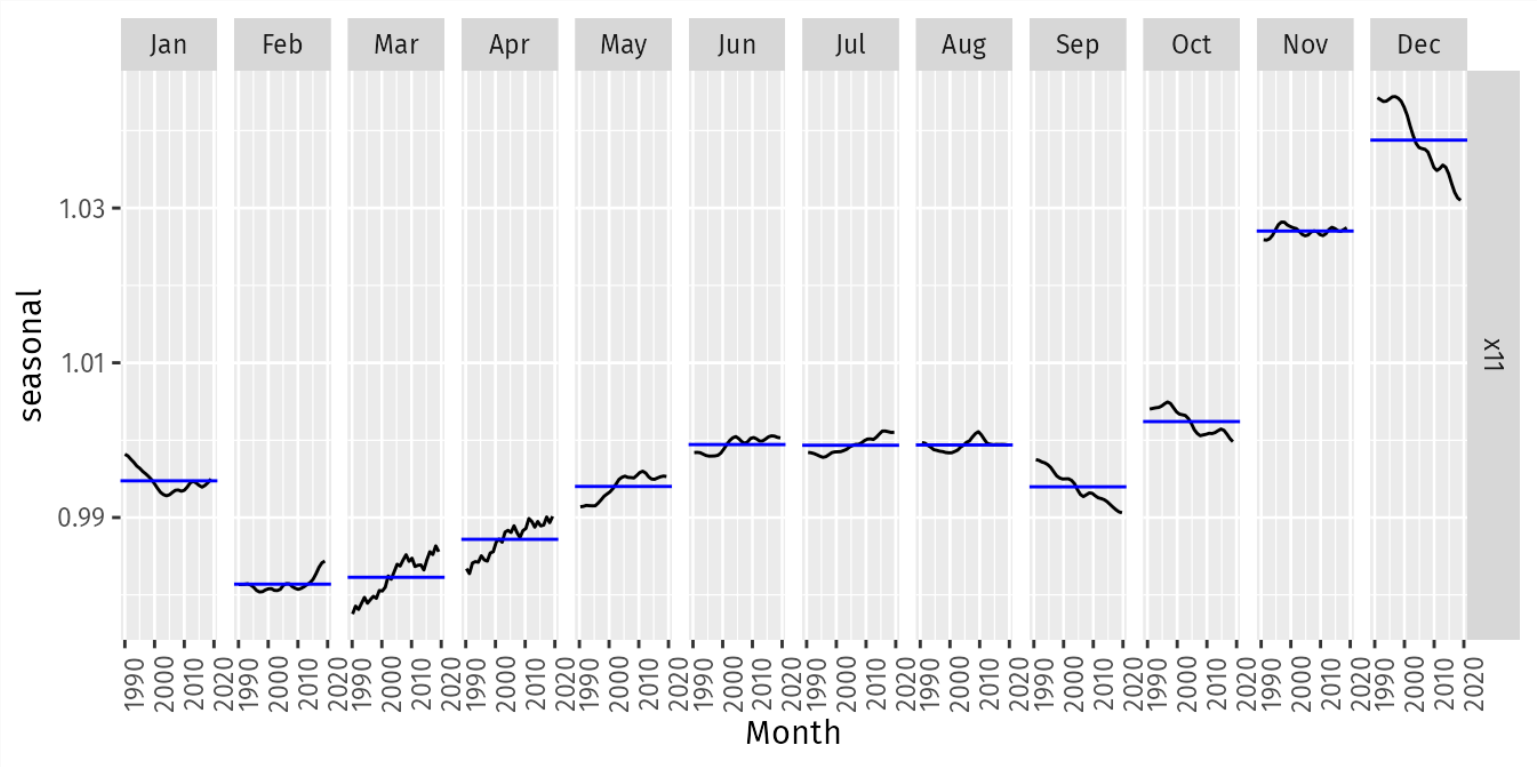


注意：这里算法自动采用了乘法模型（见副标题）。2008 年金融危机带来的就业下降也全部被趋势周期项涵盖（见剩余项）。

2.3. 各国统计部门使用的方法

季节图（seasonal plot）可以帮助我们观察季节项如何随时间变化。

```
x11_dcmp |> gg_subseries(seasonal)
```



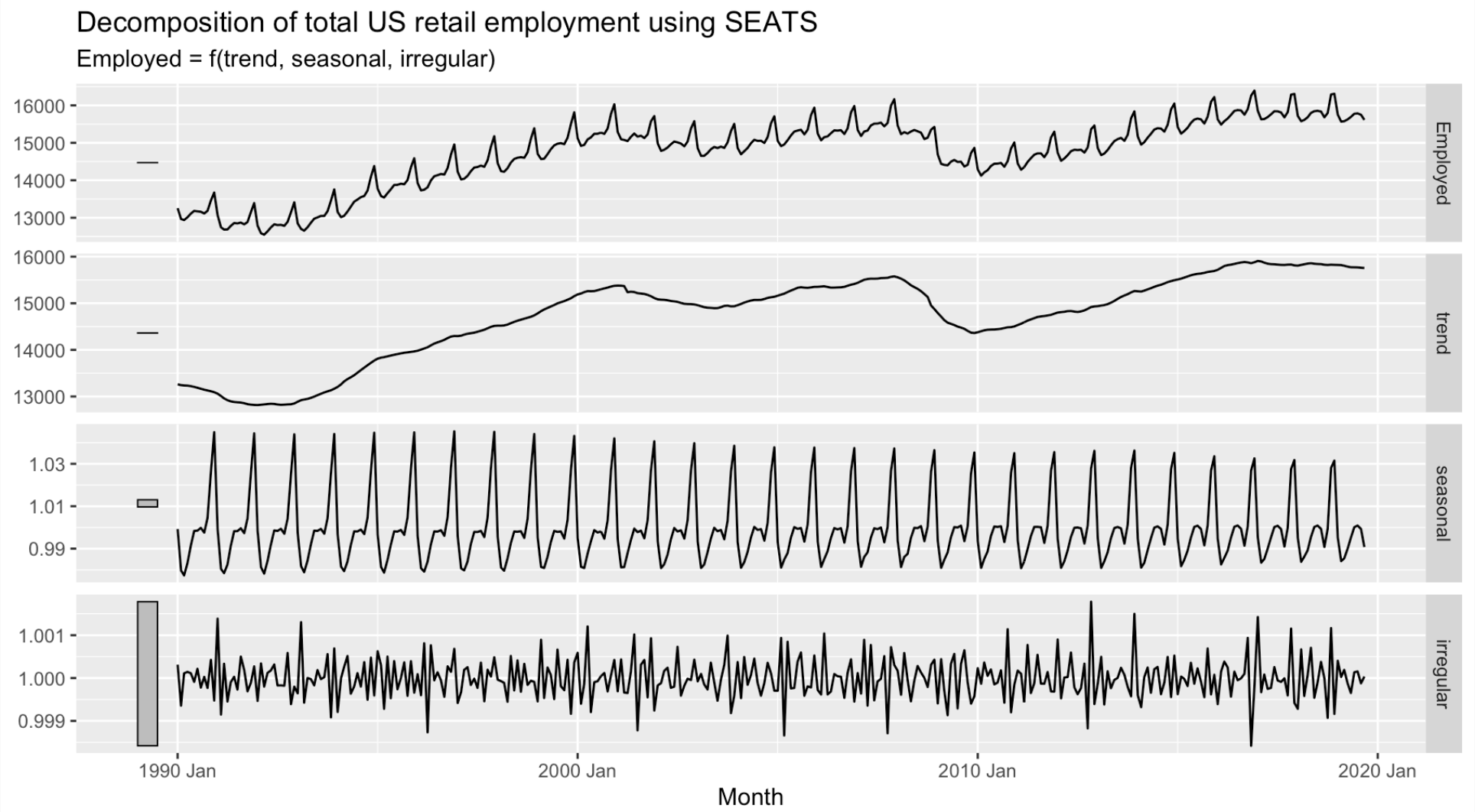
2.3. 各国统计部门使用的方法

SEATS 是 Seasonal Extraction in ARIMA Time Series 的缩写（后面会学习 ARIMA 模型），由西班牙银行开发。此方法也包含在 seasonal 包中。

seasonal 工具包的官方网站是 http://www.seasonal.website/seasonal.html

```
seats_dcmp <- us_retail_employment |>
  model(seats = X_13ARIMA_SEATS(Employed ~ seats())) |>
  components()
autoplot(seats_dcmp) +
  labs(title =
       "Decomposition of total US retail employment using SEATS")
```

2.3. 各国统计部门使用的方法



2.4. STL 分解法

STL 是 Seasonal and Trend decomposition using Loess 的缩写，其中 Loess 是一种平滑方法，可以用来估计两个变量间的非线性关系。

相对于传统分解法以及 X-11 和 SEATS，STL 法有以下优势：

- STL 可以处理任何形式的季节性，并不仅限于月度和季度数据。
- 允许季节项随时间变化，且变化速度可由分析者控制。
- 趋势周期项的平滑程度也可以控制。
- 分析者可以通过设置让异常值不影响趋势周期项和季节项的估计（其影响包含在剩余项中）。

STL 的劣势为：

- 无法自动处理交易日数和月份长度的影响。
- 仅能进行加法分解（但可以通过对数变换或 Box-Cox 变换预先处理数据，使其适用于加法模型）。

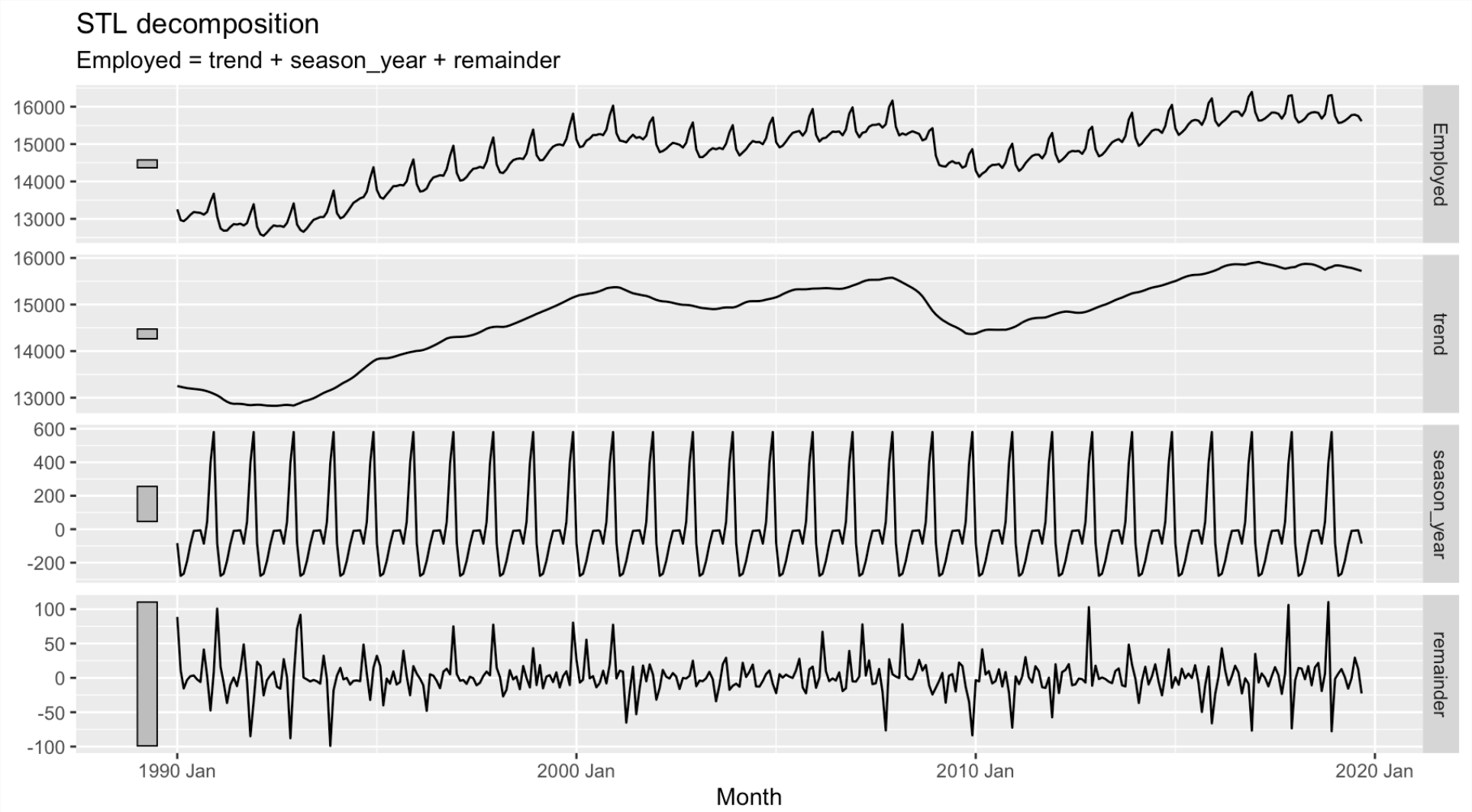
2.4. STL 分解法

下面是利用 STL 进行分解的程序。

```
us_retail_employment |>
  model(
    STL(Employed ~ trend(window = 7) + season(window = "periodic"),
    robust = TRUE)
  ) |>
  components() |>
  autoplot()
```

其中，`STL(Employed ~` 后面的部份是针对模型参数的设定。`trend(window = 7)` 设定了用 Loess 估计趋势周期项时利用的期数（类似 MA 模型中的 m，须为奇数），`season(window = "periodic")` 代表季节项是固定的（类似传统分解法），`robust = TRUE` 指定了使用稳健方法。

2.4. STL 分解法


STL decomposition
Employed = trend + season_year + remainder
16000
15000
14000
13000
Employed
16000
15000
14000
13000
trend
600
400
200
0
-200
season_year
100
50
0
-50
-100
remainder
1990 Jan
2000 Jan
2010 Jan
2020 Jan
Month

2.4. STL 分解法

`STL()` 中的两个参数 `trend(window = ?)` 和 `trend(season = ?)` 分别用来调整趋势周期项和季节项的变化速度，设定的值越小代表每个时间点所参考的前后期数越少，也就是变化的速度越快。如果设定为数字，则都应该是奇数。`season(window = "periodic")` 对应设定期数为无穷大，因此代表固定不变。

如果不设定参数，则 `STL()` 会自动进行判断。针对单一周期的季节项，默认 `season(window = 11)`，同时自动计算 trend window 值。对于月度数据，默认 `trend(window = 21)`。针对多周期季节项，则默认 seasonal window 为 11, 15, 19 等。

自动化显然无法适用于所有数据。作为分析者可以首先尝试自动分解，如果对结果不满意（通常体现在剩余项包含了不应该包含的趋势或周期性），可以再手动调整参数。

3. 课后练习

3. 课后练习

- 学习教科书第 3 章（Time Series Decomposition）中的内容，并尝试在自己的电脑上复现书中的结果。

- 回答下列问题：

  1. 针对美国零售业就业人数数据，尝试用 $2\times n$-MA （n 不是 12 的倍数）估计趋势周期项。观察估计结果是否理想，是否有效剔除了季节性特征。
  2. 利用美国零售业就业人数数据，尝试改变 STL 法中的参数，并观察结果的变化。