

# 時系列解析

氏名：鈴木 朝陽

## 1. 目的

自分で見つけた時系列データに対し、本演習で学んだ手法等を活用して時系列データを分析し、その結果をまとめることが目的である。

## 2. 使用したデータについて

使用したデータは、Kaggle が主催していた Tabular の一つ「Store Item Demand Forecasting Challenge」のデータセットである。データのカラムを以下に示した。

データ名： train.csv

- date…日付 (2013/01/01~2017/12/31)
- store…店舗番号 (1~10)
- item…商品番号 (1~50)
- sales…売上個数

売上個数の分布のグラフを図 2- 1 に示した。

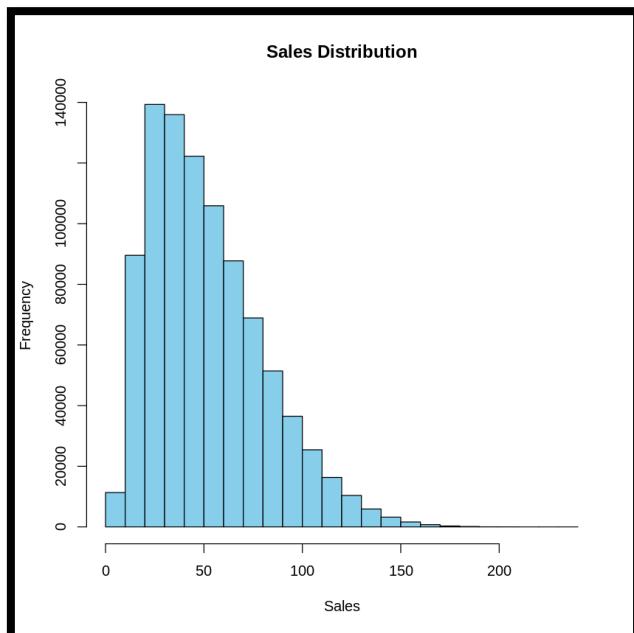


図 2-1 売上個数の分布

図 2-1 より、売上個数の分布は、右に裾が長いグラフであると分かった。

次に Pivot を用い、各店舗×商品の売り上げデータ「df」を作成した。「df」の一部が概要を図 2-2 に示した。

date	store=1&item=1	store=2&item=1	...	store=8&item=50	store=9&item=50	store=10&item=50
<date>	<int>	<int>	<chr>	<int>	<int>	<int>
2013-01-01	13	12	...	45	36	33
2013-01-02	11	16	...	54	44	37
2013-01-03	14	16	...	54	29	46
2013-01-04	13	20	...	52	43	51
2013-01-05	10	16	...	48	53	41
2013-01-06	12	18	...	51	38	41

図 2-2 「df」の一部概要

### 3. 商品番号：1についての可視化

この分析では、特に「item=1」（商品番号：1）について予測モデルを作成した。商品番号：1について可視化していった。横軸に「date」、縦軸に「sales」を取った折れ線グラフを図 3-1 に示した。

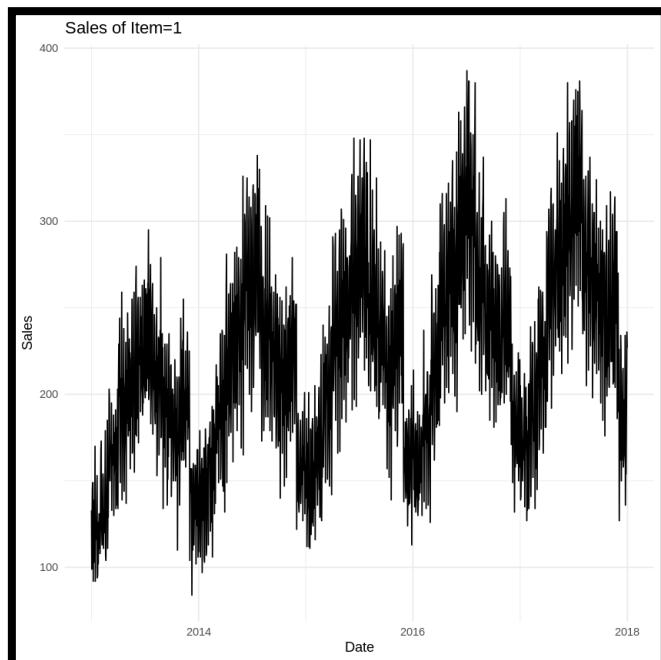


図 3-1 商品番号：1 の売上個数推移

図 3-1 を見ると、1年ごとに周期性が見られた。次に店舗別の商品番号：1の売り上げ個数を箱ひげ図で図 3-2 に示した。

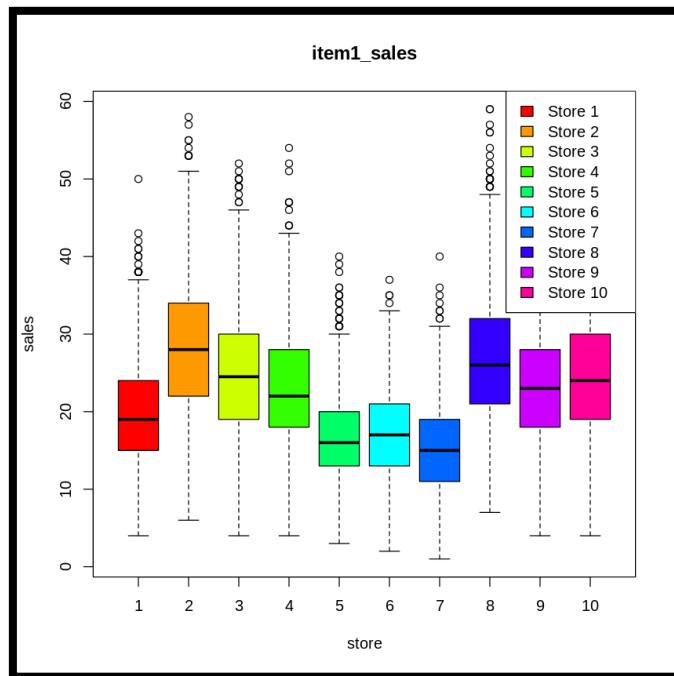


図 3-2 商品番号：1 の店舗別売上

図 3-2 より、売り上げについて全体的に店舗番号：2、8、10 が高く、店舗番号：1、5、6、7 は低い分布であるとわかった。次に商品番号：1 の売り上げを曜日別に箱ひげ図にプロットした図を図 3-3 に示した。

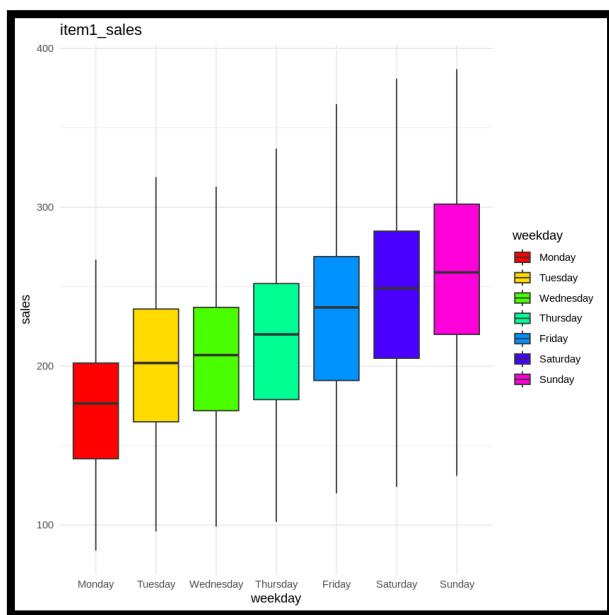


図 3-3 商品番号：1 の曜日別売り上げ

図 3-3 より、商品番号：1 の曜日別の売り上げは、月曜から日曜にかけて分布が高くなっていることが分かった。

R コード：

```
train = read.csv("/content/sample_data/train.csv", header = T)
train$date <- as.Date(train$date)

# ヒストグラムを描画
hist(train$sales,
      breaks = 20,          # 棒の数 (bins に相当)
      col = "skyblue",       # 棒の色
      border = "black",      # 枠線の色
      main = "Sales Distribution", # タイトル
      xlab = "Sales",        # x 軸ラベル
      ylab = "Frequency") # y 軸ラベル

# store と item を組み合わせて列名用のラベルを作成
train <- train %>%
  mutate(store_item = paste0("store=", store, "&item=", item))

# pivot_wider で日付ごとに各 store×item の sales を列にする
df <- train %>%
  pivot_wider(
    id_cols = date,          # 行のインデックスにする列
    names_from = store_item,  # 列名にする列
    values_from = sales      # 値にする列
  )

# 結果確認
show_head_tail_console(df, n = 3, rows = 6)

# item=1 の列だけ抽出
item1_cols <- grep("item=1$", colnames(df), value = TRUE)
df_item1_total <- df %>%
  select(date, all_of(item1_cols)) %>%
  mutate(total_sales = rowSums(across(all_of(item1_cols))))%>%
```

```

mutate(date = as.Date(date),
       weekday = weekdays(date)) # 曜日列を追加

# 折れ線グラフ (全店舗合計)
ggplot(df_item1_total, aes(x = date, y = total_sales)) +
  geom_line(color = "black") +
  labs(title = "Sales of Item=1", x = "Date", y = "Sales") +
  theme_minimal()

# 曜日別箱ひげ図
df_item1_total <- df_item1_total %>%
  mutate(weekday = factor(weekday, levels =
c("Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday")))

ggplot(df_item1_total, aes(x = weekday, y = total_sales, fill = weekday)) +
  geom_boxplot() +
  scale_fill_manual(values = rainbow(7)) +
  labs(title = "item1_sales",
       x = "weekday", y = "sales") +
  theme_minimal()

# item=1 の列だけ抽出
item1_cols <- grep("item=1$", colnames(df), value = TRUE)
df_item1 <- df %>%
  select(date, all_of(item1_cols)) %>%
  pivot_longer(-date, names_to = "store", values_to = "sales")

# store 名を 1~10 に置換
df_item1 <- df_item1 %>%
  mutate(store = factor(1:length(unique(store)))[match(store, unique(store))])

# 箱ひげ図
boxplot(sales ~ store, data = df_item1,
        main = "item1_sales",
        xlab = "store",
        ylab = "sales",

```

```

col = rainbow(10)

# 凡例
legend("topright", legend = paste("Store", 1:10), fill = rainbow(10))

```

#### 4. 商品番号：1 の自己相関・偏自己相関・スペクトル解析

商品番号：1 のデータ名を df\_item1\_total とした。df\_item1\_total を週周期仮定し、時系列オブジェクトに変換したデータを ts\_item1 とした。商品番号：1 の売り上げ個数の自己相関、偏自己相関を図 4-1、4-2 に示した。

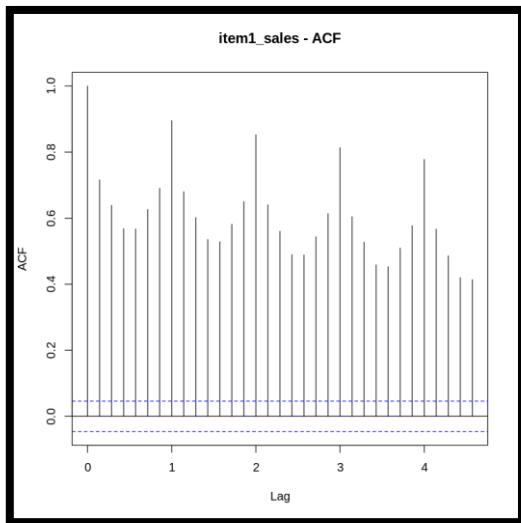


図 4-1 商品番号：1 の自己相関

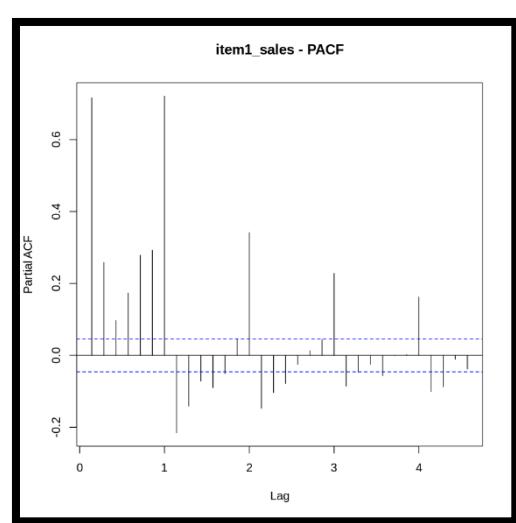


図 4-2 商品番号：1 の偏自己相関

図 4-1、4-2 より、週周期を設定されており、プロットを見ても、約 7 ラグごと（ラグ 0, 1, 2, 3, 4 の単位で見ると、約 7 本ごと）に自己相関が大きなピークを示しているように見えた。したがって、商品番号：1 の売上個数は「週の周期性（季節性）」が強く存在しているといえた。また、すべてのラグにおいて自己相関の値が正であったため、この時系列データに非常に強い正の自己相関が存在しているといえた。したがって、データは非定常的である可能性が高い。

次に、ts\_item1 から平均値を引いた時系列データ sales\_data\_m の折れ線グラフを図 4-3 に示した。

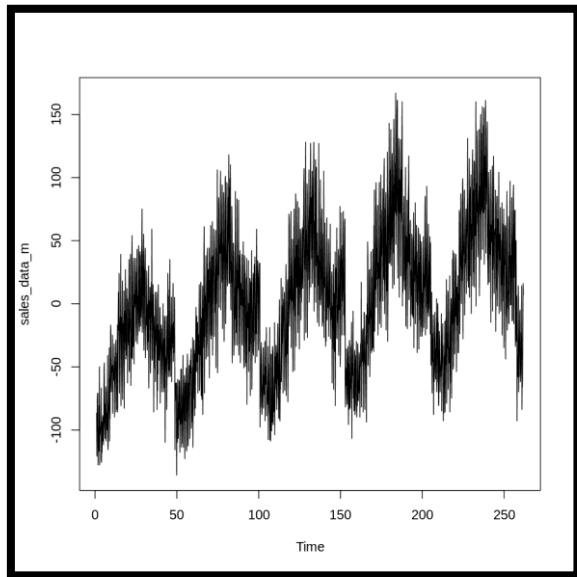


図 4-3 平均値を引いた商品番号：1 の売上個数推移

図 4-3 より、平均値を引くことでデータの水準が 0を中心とするようにした。

`sales_data_m` の自己相関関数・偏自己相関関数を図 4-4、4-5 に示した。

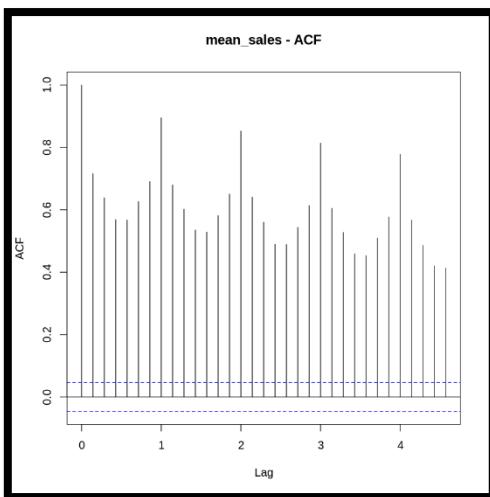


図 4-4 `sales_data_m` の自己相関

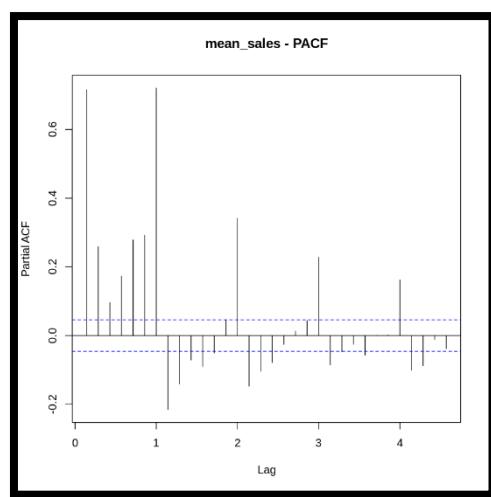


図 4-5 `sales_data_m` の偏自己相関

図 4-4、4-5 より、変わらず約 7 ラグごと（ラグ 0, 1, 2, 3, 4 の単位で見ると、約 7 本ごと）に自己相関が大きなピークを示しているように見え、「週の周期性（季節性）」が強く存在しているといえた。また、すべてのラグにおいて自己相関の値が正であったため、この時系列データに非常に強い正の自己相関が存在しているといえた。したがって、データは非定常的である可能性が高い。

続いて、「周期性（周波数成分）」を検出・評価するため、パワースペクトラムを AR 法を用いて求めた。結果を図 4-6 に示した。

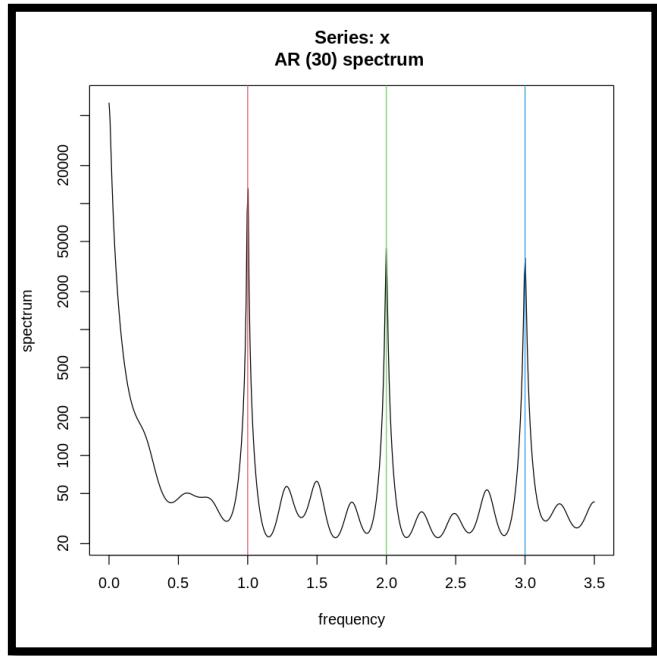


図 4-6 パワースペクトル：AR 法

図 4-6 より、sales\_data\_m は frequency=1,2,3 に対応する非常に強い周期成分が存在していると分かった。続いて、sales\_data\_m の FFT 計算を行い、各周波数のパワースペクトルの自然対数値を求めた結果を図 4-7 に示した。

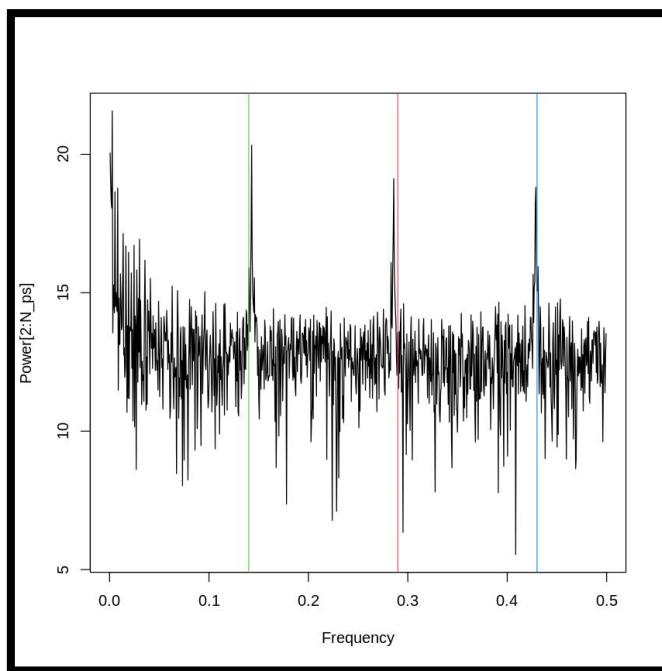


図 4-7 各周波数のパワースペクトルの自然対数値

図 4-7 より、各周波数のパワースペクトルの自然対数値を取った結果、frequency=約

0.15, 0.29, 0.43 でグラフのピークがみられ、データが持つ強い周波数成分を示した。  
frequency=約 0.15 の周期 6.7 は、日次データより約 1 週間に近いと分かった。

R コード：

```
# 時系列オブジェクトに変換（例：日次データ）
ts_item1 <- ts(df_item1$total_sales, frequency = 7) # 週周期を仮定

# 自己相関プロット
acf(ts_item1, main = "item1_sales - ACF")

# 偏自己相関プロット
pacf(ts_item1, main = "item1_sales - PACF")

# 平均値を引いた時系列データの作成
# 売上データの平均値を計算
mean_sales <- mean(ts_item1, na.rm = TRUE)
mean_sales

# 平均値を引いた時系列データ
sales_data_m <- ts_item1 - mean_sales
plot(sales_data_m, type="l")

# 平均値を引いたデータの自己相関関数（ACF）のプロット
acf(sales_data_m, main = "mean_sales - ACF")

# 平均値を引いたデータの偏自己相関関数（PACF）のプロット
pacf(sales_data_m, main = "mean_sales - PACF")

# ペリオドグラム
spectrum(sales_data_m,method="pgram")
abline(v=1,col=2)
abline(v=2,col=4)

# パワースペクトル；AR 法
spectrum(sales_data_m,method="ar")
abline(v=1,col=2)
```

```

abline(v=2,col=3)
abline(v=3,col=4)

# FFT をかける
fft_result <- fft(sales_data_m)
# print(length(sales_data_m))
N_ps <- length(fft_result)/2

Power <- log( Mod(fft_result)^2 )
freq <- (seq(1:(N_ps-1))/N_ps)*0.5
# print(length(Power))
plot(freq, Power[2:N_ps],type='l', xlab='Frequency')
abline(v=0.14,col=3)
abline(v=0.29,col=2)
abline(v=0.43,col=4)

```

## 5. Box-Cox 変換

次に、データの分散を安定化させ、正規分布に近づけるため、Box-Cox 変換を行った。負の値では Box-Cox 変換を行えないため、sales\_data\_m のデータがすべて正の値になるよう平行移動させた sales\_ts\_adjusted を作成した。実行した結果、AIC が最小値となった lambda は 0.70 であった。Box-Cox 変換を行う前後の時系列データのグラフを図 5-1 に示した。

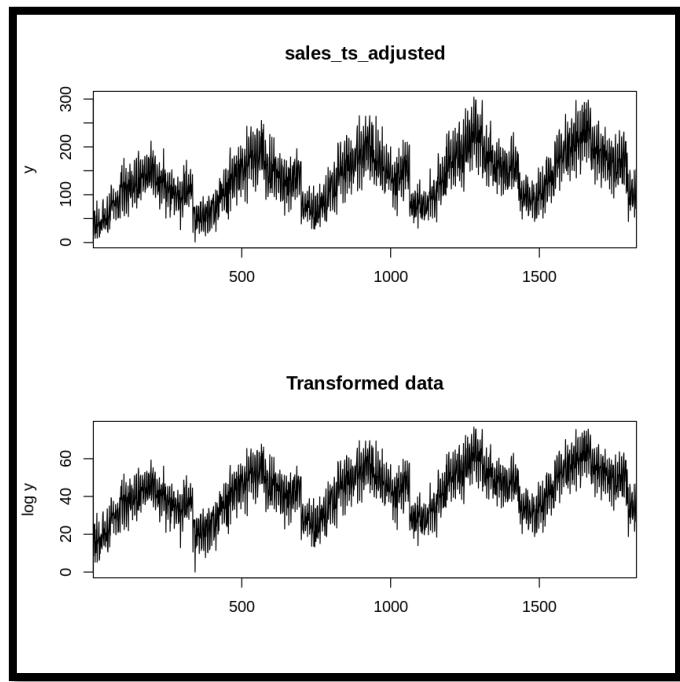


図 5-1 Box-Cox 変換を行う前後の時系列データ（上図：前、下図：後）

図 5-1 より、Box-Cox 変換を行う前の `sales_ts_adjusted` は、時系列の経過とともに変動幅も大きくなっているように見えた。Box-Cox 変換を行った後の `Transformed data` では、変換後もトレンドや季節性は残っているが、変換前のデータと比べて、変動幅がデータレベルによらずほぼ一定になっていた。つまり、Box-Cox 変換（または対数変換）によって、データの分散が安定化（均一化）されたことが分かった。

また、`sales_ts_adjusted` をヒストグラムに描画した図を図 5-2 に示した。最適な `lambda` = 0.7 を用いて、元の調整済みデータ `sales_ts_adjusted` に実際に Box-Cox 変換を適用し、変換後のデータ `M_sales_ts_adjusted` をヒストグラムに描画した図を図 5-3 に示した。

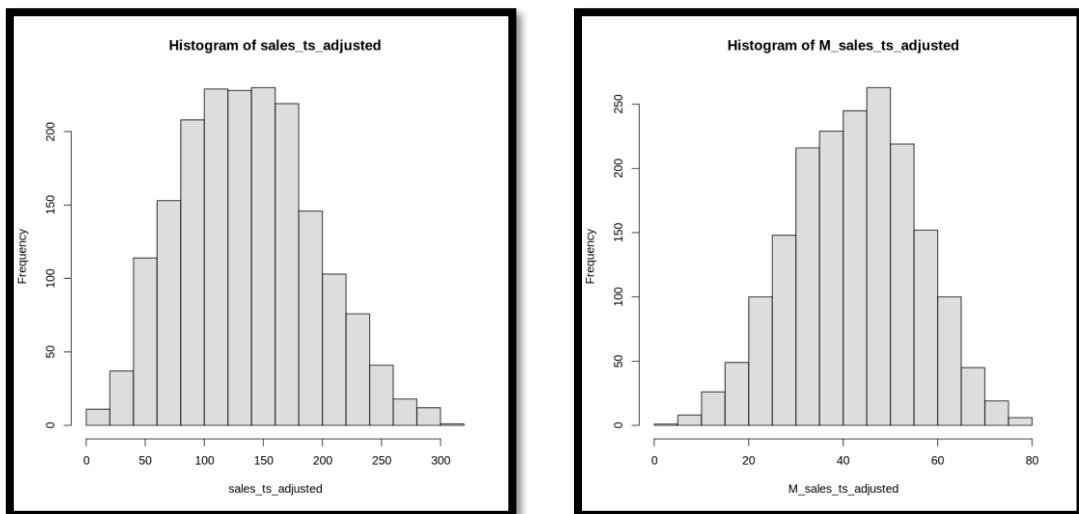


図 5-2 Box-Cox 変換前のヒストグラム　　図 5-3 Box-Cox 変換後のヒストグラム

図 5-2、5-3 より、Box-Cox 変換によってデータが正規分布に近づいたことがわかった。

R コード：

```
# Check for minimum value and add a constant if necessary for Box-Cox
min_sales_data_m = min(sales_data_m, na.rm = TRUE)
if (min_sales_data_m <= 0) {
    # Add a constant slightly larger than the absolute value of the minimum
    sales_data_m_adjusted = sales_data_m + abs(min_sales_data_m) + 1
} else {
    sales_data_m_adjusted = sales_data_m
}

# ts()関数を使って時系列オブジェクトに変換
# データの観測頻度は週単位であるため、frequency は 1
sales_ts_adjusted = ts(sales_data_m_adjusted, frequency = 1)

# ts オブジェクトに変換したデータで boxcox()を実行
boxcox(sales_ts_adjusted)

hist(sales_ts_adjusted)

L_Best = 0.7
M_sales_ts_adjusted =(sales_ts_adjusted^L_Best - 1.0) / L_Best
hist(M_sales_ts_adjusted)
```

## 6. 時系列分解

次に時系列データをトレンド、季節性、残差の 3 つの主要な成分に分解していった。授業では用いていないが、stl()関数を用いて分解し各成分を抽出した。結果を図 6-1 に示した。

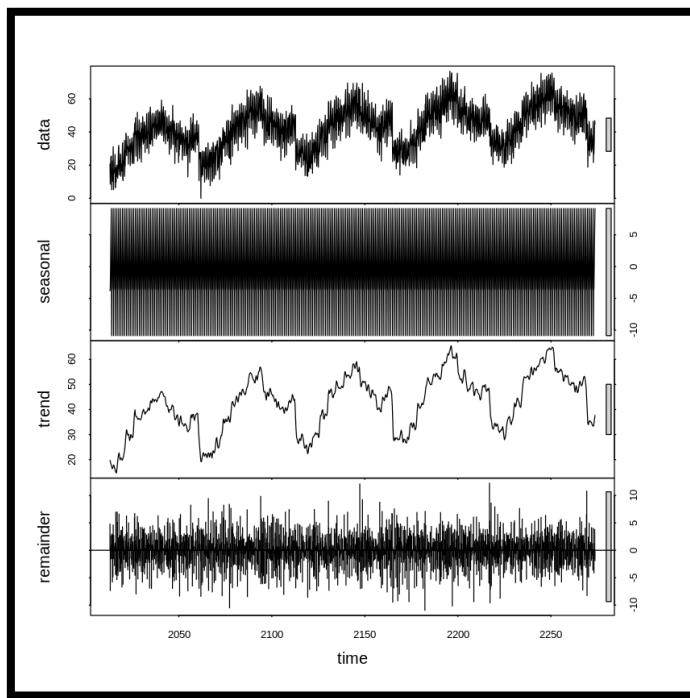


図 6-1 時系列分解

図 6-1 より、seasonal(季節成分)は、振幅が一定で非常に高頻度な振動が継続している。設定した週周期に基づき、曜日ごとの変動パターンが抽出された。振幅がほぼ一定であるため、週周期のパターンが時間とともに大きく変化していないことが分かった。

trend(トレンド成分)は、周期的な変動が取り除かれた、滑らかな長期的な変動を示す。データは、期間を通じて何度も大きく上下に変動しており、特定の一方向への明確な上昇・下降トレンド(線形トレンド)というよりは、数ヶ月～数年単位の緩やかな循環的な変動を持っていることがわかった。

remainder(残差成分)は、トレンドと季節性が取り除かれた残りの変動である。縦軸の変動幅が $\pm 10$ の範囲に収まっており、極端な異常値(スパイク)は少なかった。

抽出した remainder(残差成分)の自己相関関数、偏自己相関関係をプロットした結果を図 6-2、6-3 に示した。

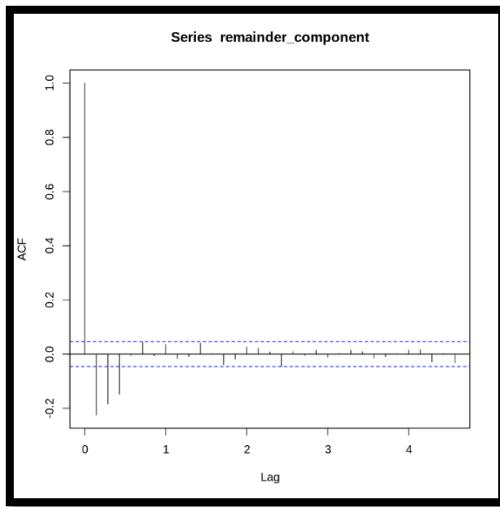


図 6-2 remainder の自己相関関数

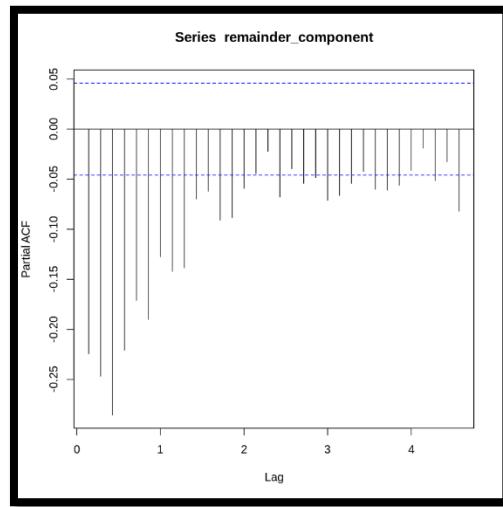


図 6-3 remainder の偏自己相関関数

図 6-2 より、Lag 1 は信頼区間（青い点線）を明確に下回っており、Lag 2 以降は信頼区間に収まっているが、有意な相関は見られなかった。したがって Lag 1 に強い負の相関が残っているため、残差系列はホワイトノイズではないとわかった。この相関は、MA(1) モデルでモデル化される可能性が高い。

図 6-3 より、Lag 1 は信頼区間を明確に下回っており、Lag 2 以降の相関も、Lag 1 よりは小さいものの、Lag 2 と Lag 3 などで信頼区間をわずかに下回っていた。ただし、ラグが増えるにつれて徐々にゼロに近づいていた。したがって、Lag 1 に強い負の相関があり、その後は減衰していることから、残差系列には AR(p) 構造または MA(q) 構造が存在している可能性が高い。

R コード：

```
# 1. Box-Cox 変換後のデータを時系列オブジェクトに変換
# データの頻度 (frequency) を週周期 (7) に設定
# 例：データが 2013 年 1 月 1 日から始まる仮定
# 実際のデータの開始日と頻度に合わせて調整してください
sales_ts <- ts(M_sales_ts_adjusted, start = c(2013, 1), frequency = 7)

# 2. STL 分解の実行
# s.window="periodic" で季節成分を周期的にロバストに推定
stl_decomposition <- stl(sales_ts, s.window = "periodic")

# 3. 分解結果のプロット
plot(stl_decomposition)
```

```

# 4. 各成分の抽出
# トレンド成分
trend_component <- stl_decomposition$time.series[, "trend"]

# 季節成分
seasonal_component <- stl_decomposition$time.series[, "seasonal"]

# 残差成分
remainder_component <- stl_decomposition$time.series[, "remainder"]

acf(remainder_component)

pacf(remainder_component)

```

## 7. SARIMA モデル

これまでの分析から、商品番号：1 の売上個数データには強力な週周期が存在することがスペクトル解析により判明した。時系列分解によってこの周期的な変動が季節成分として明確に分離され、データがトレンド、季節性、残差の三要素を持つことが確認されたため、これらの構造を総合的に扱うことができるモデルが必要である。さらに、季節成分を取り除いた残差系列にもラグ 1 で有意な自己相関 (MA(1)構造) が残っており、これは短期的な非季節性の変動としてモデル化する必要がある。したがって、このデータは周期性を扱う季節項と残差の自己相関を扱う非季節項の両方を備えた季節性 ARIMA (SARIMA) モデルを用いることで、データの持つ全ての相関構造を効率的かつ正確に捉え、最も適切な予測や分析が可能となるため、SARIMA モデルが最適である。

最初のモデルは、時系列分解の結果から残差系列が持つと推定された非季節性の自己相関 (MA(1)) のみを取り込み、季節性は考慮しない SARIMA(0, 0, 1) x (0, 0, 0)\_7 であった。このモデルは、最も単純な構造であったものの、AIC が非常に高く、残差分散も大きかった (aic = 13865.31、残差分散=115.8) ため、検出された週周期の変動を全く捉えられないことが示唆された。

そこで次に、この強い季節性をモデル化するため、SARIMA(0, 0, 1) x (1, 0, 1)\_7 を試みた。このモデルは、季節性の AR(1)項と MA(1)項を加えたことで、AIC と残差分散を大幅に改善 (aic = 11189.73、残差分散=26.51) し、データの適合度は飛躍的に向上した。残差診断を行った結果を図 7-1、7-2 に示した。

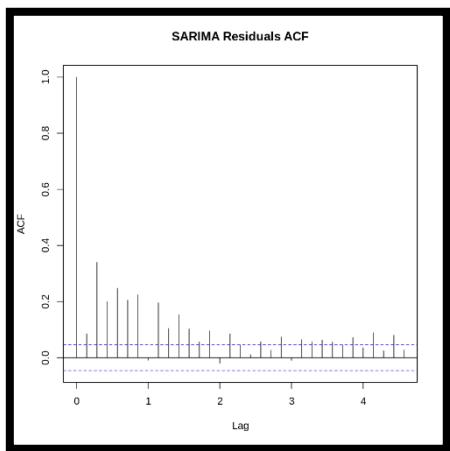


図 7-1 SARIMA2 Residuals ACF

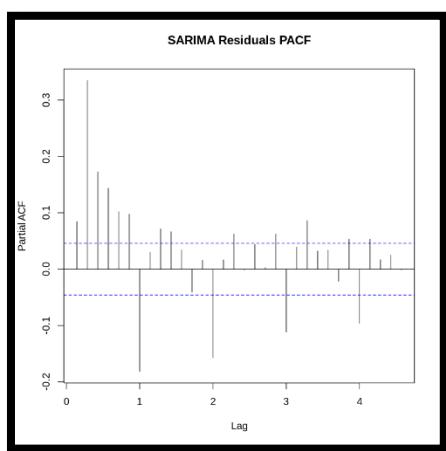


図 7-2 SARIMA2 Residuals PACF

図 7-1、7-2 より、多くのラグで相関が残った。Box-Ljung 検定でも  $p$  値 :  $2.2 \times 10^{-16}$  となり、有意水準よりはるかに小さいため、帰無仮説を棄却した。したがって残差がホワイトノイズではないと強く示された。これは、データの非定常性（トレンドや変動の傾向）が十分に除去されていないことを意味したため、差分操作の必要性が浮上した。

最終的に、データの定常性を確保するため、非季節差分  $d=1$  と季節差分  $D=1$  を導入した SARIMA(0, 1, 1) x (0, 1, 1)\_7 を適用した。このモデルは、前回のモデルから AIC をさらに下げ、残差分散を最小化した ( $aic = 10402.3$ 、残差分散=17.5) ことで、データへの究極的な適合度を示した。残差診断を行った結果を図 7-3、7-4 に示した。

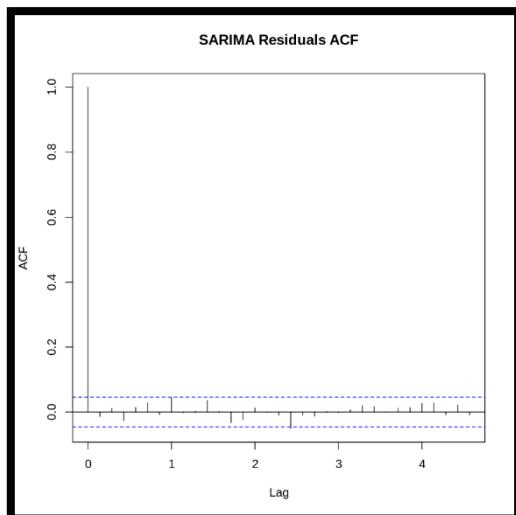


図 7-3 SARIMA3 Residuals ACF

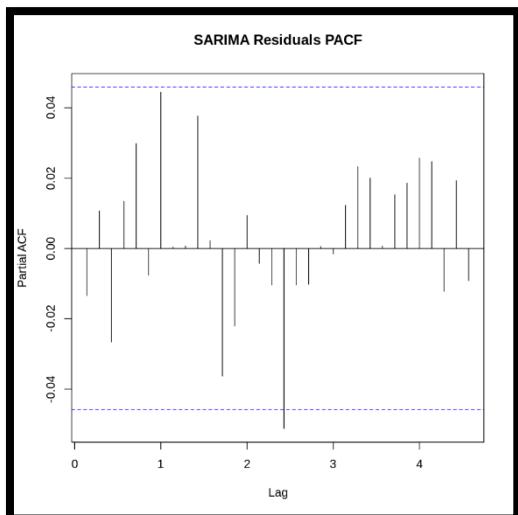


図 7-4 SARIMA3 Residuals PACF

図 7-3、7-4 より、残差診断では ACF と PACF のすべてのラグが信頼区間に収まった。また Box-Ljung 検定の  $p$  値 : 0.513 も大きかったため、残差がホワイトノイズであると確認された。これにより、このモデルがデータの持つすべての自己相関構造（短期的な MA(1)、週周期）を適切に捉えられた最終的な最適モデルであると結論づけられた。

R コード：

```
# Box-Cox 変換後のデータを時系列オブジェクトとして再確認 (frequency=7)
# ts_data_boxcox は Box-Cox 変換後のデータが入っているとする

# -----
# 【モデル 1: 最もシンプルな SARIMA モデルを適用】
# SARIMA(0, 0, 1) x (0, 0, 0)_7
# 季節項は 0 とし、非季節の MA(1)のみを考慮
# -----
model_s0 <- arima(sales_ts,
                    order = c(0, 0, 1),
                    seasonal = list(order = c(0, 0, 0), period = 7))

# 結果の表示 (AIC/BIC、係数、p 値などを確認)
print(model_s0)

# -----
# 【モデル 2: 季節項を考慮した一般的な SARIMA モデルを試す】
# SARIMA(0, 0, 1) x (1, 0, 1)_7
# 季節性の AR(1)と MA(1)を考慮
# -----
model_s1 <- arima(sales_ts,
                    order = c(0, 0, 1),
                    seasonal = list(order = c(1, 0, 1), period = 7))

# 結果の表示と AIC の比較
print(model_s1)

# AIC がより小さいモデルを選択し、以下の診断へ進む

# -----
# 【モデル診断（必ず行うべき工程）】
# -----
```

# 1. 残差の自己相関の確認 (残差がホワイトノイズであるかをチェック)

```
acf(model_s1$residuals, main = "SARIMA Residuals ACF")
```

```
pacf(model_s1$residuals, main = "SARIMA Residuals PACF")

# 2. Ljung-Box 検定（残差に相関が残っていないかを統計的に検定）
# lag = 2 * period = 14 程度を目安に設定
Box.test(model_s1$residuals, lag = 14, type = "Ljung-Box")

# -----
# 【モデル 3: 季節項を考慮した一般的な SARIMA モデルを試す】
# SARIMA(0, 1, 1) x (0, 1, 1)_7
# -----
model_s1 <- arima(sales_ts,
                   order = c(0, 1, 1),
                   seasonal = list(order = c(0, 1, 1), period = 7))

# 結果の表示と AIC の比較
print(model_s1)

# AIC がより小さいモデルを選択し、以下の診断へ進む

# -----
# 【モデル診断（必ず行うべき工程）】
# -----


# 1. 残差の自己相関の確認（残差がホワイトノイズであるかをチェック）
acf(model_s1$residuals, main = "SARIMA Residuals ACF")
pacf(model_s1$residuals, main = "SARIMA Residuals PACF")

# 2. Ljung-Box 検定（残差に相関が残っていないかを統計的に検定）
# lag = 2 * period = 14 程度を目安に設定
Box.test(model_s1$residuals, lag = 14, type = "Ljung-Box")
```