Classwork 1: The Basetable Timeline

Intermediate Predictive Analytics

Practical Exercises in Temporal Data Structuring

Prof. Asc. Endri Raco, Ph.D.

November 2025

Contents

1 Overview **3**
- 1.1 Learning Objectives . . . 3
- 1.2 Instructions . . . 3
- 1.3 Grading Rubric . . . 3

2 Dataset Description **4**
- 2.1 E-Commerce Transaction Data . . . 4
  - 2.1.1 Data Generation . . . 4
  - 2.1.2 Data Dictionary . . . 5
  - 2.1.3 Business Context . . . 5

3 Exercise 1: Data Exploration and Preparation (15 points) **6**
- 3.1 Task 1.1: Load and Inspect Data (5 points) . . . 6
- 3.2 Task 1.2: Data Quality Check (5 points) . . . 6
- 3.3 Task 1.3: Summary Statistics by Product Category (5 points) . . . 7

4 Exercise 2: Timeline Definition and Data Partitioning (20 points) **8**
- 4.1 Task 2.1: Define Timeline Components (5 points) . . . 8
- 4.2 Task 2.2: Partition Transaction Data (10 points) . . . 8
- 4.3 Task 2.3: Visualize Timeline (5 points) . . . 9

5 Exercise 3: Population Construction (25 points) **10**
- 5.1 Task 3.1: Define Eligibility Criteria (5 points) . . . 10
- 5.2 Task 3.2: Implement Population Filters (15 points) . . . 10
- 5.3 Task 3.3: Population Analysis (5 points) . . . 11

6 Exercise 4: Target Variable Definition (20 points) **12**
- 6.1 Task 4.1: Define Binary Target (10 points) . . . 12
- 6.2 Task 4.2: Define Continuous Target (10 points) . . . 12

7 Exercise 5: Complete Basetable Construction (20 points) **14**
- 7.1 Task 5.1: Calculate Predictive Features (15 points) . . . 14
  - 7.1.1 RFM Features . . . 14
  - 7.1.2 Time-Based Features . . . 14
  - 7.1.3 Category Preferences . . . 15
- 7.2 Task 5.2: Construct Final Basetable (5 points) . . . 15
- 7.3 Task 5.3: Validate Basetable (Bonus: 5 points) . . . 15

8 Exercise 6: Reflection and Critical Thinking (Bonus: 10 points) 17
8.1 Question 1: Timeline Integrity . . . 17
8.2 Question 2: Population Definition . . . 17
8.3 Question 3: Target Variable Selection . . . 17
8.4 Question 4: Feature Engineering . . . 17
8.5 Question 5: Practical Application . . . 18

9 Submission Checklist 19

10 Appendix: Useful R Code Snippets 20
10.1 Date Manipulation with lubridate . . . 20
10.2 Set Operations . . . 20
10.3 Common tidyverse Patterns . . . 20
10.4 Handling Missing Values . . . 21

11 Solutions Template 22

1 Overview

1.1 Learning Objectives

By completing this classwork, you will:

1. Construct basetables with proper temporal structure
2. Implement timeline-compliant data partitioning
3. Define and filter populations using set operations
4. Create binary and continuous target variables
5. Apply historical reconstruction techniques
6. Validate temporal integrity in predictive datasets

1.2 Instructions

- **Duration**: 90 minutes
- **Format**: Individual work
- **Submission**: R script (.R) or R Markdown (.Rmd) file
- **Grading**: Based on correctness, code quality, and documentation
- **Resources**: You may use lecture notes, R documentation, and tidyverse references

1.3 Grading Rubric

Component	Points	Description
Exercise 1	15	Data loading and exploration
Exercise 2	20	Timeline definition and data partitioning
Exercise 3	25	Population construction with set operations
Exercise 4	20	Target variable definition
Exercise 5	20	Complete basetable construction
Total	**100**	

2 Dataset Description

2.1 E-Commerce Transaction Data

For this classwork, you will work with a simulated e-commerce transaction dataset containing customer purchase history from an online retail store.

2.1.1 Data Generation

First, generate the dataset using the following code:

```
# Generate e-commerce transaction data
# This simulates 3 years of transaction history for 500 customers

n_customers <- 500
n_transactions <- 5000

# Create transaction data
transactions <- tibble(
  customer_id = sample(1:n_customers, n_transactions, replace = TRUE),
  transaction_date = sample(
    seq(as.Date("2020-01-01"), as.Date("2022-12-31"), by = "day"),
    n_transactions,
    replace = TRUE
  ),
  transaction_amount = round(rgamma(n_transactions, shape = 2, rate = 0.02), 2),
  product_category = sample(
    c("Electronics", "Clothing", "Home", "Books", "Sports"),
    n_transactions,
    replace = TRUE,
    prob = c(0.3, 0.25, 0.2, 0.15, 0.1)
  )
) %>%
  arrange(customer_id, transaction_date) %>%
  mutate(
    transaction_id = row_number(),
    # Add some realistic constraints
    transaction_amount = pmax(5, pmin(500, transaction_amount))
  ) %>%
  select(transaction_id, customer_id, transaction_date,
         transaction_amount, product_category)

# Display first few records
head(transactions, 10) %>%
  kable(caption = "Sample E-Commerce Transaction Data",
        booktabs = TRUE) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 2: Sample E-Commerce Transaction Data

transaction_id	customer_id	transaction_date	transaction_amount	product_category
1	1	2020-03-25	73.86	Electronics
2	1	2020-05-08	204.90	Electronics
3	1	2020-06-19	33.94	Clothing
4	1	2020-09-09	10.24	Sports

5	1	2020-09-21	494.62	Electronics
6	1	2021-01-26	54.33	Electronics
7	1	2021-04-26	98.04	Books
8	1	2021-05-15	231.12	Clothing
9	1	2021-07-22	59.54	Clothing
10	1	2021-10-05	106.04	Books

2.1.2 Data Dictionary

Variable	Type	Description
`transaction_id`	Integer	Unique identifier for each transaction
`customer_id`	Integer	Customer identifier (1-500)
`transaction_date`	Date	Date when transaction occurred
`transaction_amount`	Numeric	Purchase amount in currency units
`product_category`	Character	Product category purchased

2.1.3 Business Context

The e-commerce company wants to predict which customers will make a purchase in response to a promotional email campaign. They plan to:

- Send promotional emails on **July 1st, 2022**
- Measure response (purchase) during a **2-month period** (July-August 2022)
- Use historical purchase behavior to predict likelihood of response

3 Exercise 1: Data Exploration and Preparation (15 points)

3.1 Task 1.1: Load and Inspect Data (5 points)

Load the transaction data (already generated above) and answer the following questions:

```
# Your code here

# Q1: How many unique customers are in the dataset?
n_unique_customers <- ___

# Q2: What is the date range of the transactions?
date_range <- ___

# Q3: What is the average transaction amount?
avg_transaction <- ___

# Q4: How many transactions are there per customer on average?
avg_transactions_per_customer <- ___
```

Questions to answer:

a) How many unique customers made at least one purchase?
b) What is the earliest and latest transaction date?
c) What is the average transaction amount across all transactions?
d) What is the median number of transactions per customer?

Your Answers:

a) _____

b) _____

c) _____

d) _____

3.2 Task 1.2: Data Quality Check (5 points)

Check for data quality issues:

```
# Your code here

# Check for missing values
missing_check <- ___

# Check for duplicate transaction IDs
duplicate_check <- ___

# Check for transactions with amount <= 0
invalid_amount_check <- ___

# Check date ordering within customers
date_ordering_check <- ___
```

Report any data quality issues found:

3.3 Task 1.3: Summary Statistics by Product Category (5 points)

Calculate summary statistics for each product category:

```
# Your code here
# Create a summary table showing:
# - Number of transactions per category
# - Total revenue per category
# - Average transaction amount per category
# - Percentage of total transactions per category

category_summary <- transactions %>%
  group_by(___) %>%
  summarize(
    n_transactions = ___,
    total_revenue = ___,
    avg_amount = ___,
    pct_transactions = ___
  )
```

4 Exercise 2: Timeline Definition and Data Partitioning (20 points)

4.1 Task 2.1: Define Timeline Components (5 points)

Based on the business context, define the key timeline dates:

```
# Define the observation date (email send date)
observation_date <- as.Date("___")

# Define target period start date
target_start <- as.Date("___")

# Define target period end date (2 months after start)
target_end <- as.Date("___")

# Define predictor calculation period
# (we'll use all data before observation date)
predictor_start <- as.Date("___")  # Earliest date in dataset
predictor_end <- observation_date
```

Verify your dates:

```
# Calculate target period length in days
target_period_days <- as.numeric(target_end - target_start)

cat("Observation Date:", format(observation_date, "%B %d, %Y"), "\n")
cat("Target Period:", format(target_start, "%B %d, %Y"),
    "to", format(target_end, "%B %d, %Y"), "\n")
cat("Target Period Length:", target_period_days, "days\n")
```

4.2 Task 2.2: Partition Transaction Data (10 points)

Partition the transaction data into predictor and target periods:

```
# Transactions for calculating predictive features
# (all transactions BEFORE observation date)
transactions_predictors <- transactions %>%
  filter(transaction_date ___ observation_date)

# Transactions in target period
# (transactions DURING target period)
transactions_target <- transactions %>%
  filter(transaction_date ___ target_start &
         transaction_date ___ target_end)

# Verify the partition
cat("Transactions for predictors:", nrow(transactions_predictors), "\n")
cat("Transactions in target period:", nrow(transactions_target), "\n")
cat("Total transactions:", nrow(transactions), "\n")
cat("Sum check:",
    nrow(transactions_predictors) + nrow(transactions_target) ==
    nrow(transactions), "\n")
```

Questions:

a) How many transactions occurred before the observation date?
b) How many transactions occurred during the target period?
c) Does the sum equal the total? If not, why?

Your Answers:

a) _____

b) _____

c) _____

4.3 Task 2.3: Visualize Timeline (5 points)

Create a visualization showing the distribution of transactions over time with timeline markers:

```
# Create timeline visualization
timeline_plot <- transactions %>%
  mutate(month = floor_date(transaction_date, "month")) %>%
  count(month) %>%
  ggplot(aes(x = month, y = n)) +
  geom_line(color = "steelblue", size = 1) +
  geom_point(color = "steelblue", size = 2) +
  geom_vline(xintercept = as.numeric(observation_date),
             color = "red", linetype = "dashed", size = 1) +
  geom_rect(aes(xmin = as.numeric(target_start),
                xmax = as.numeric(target_end),
                ymin = -Inf, ymax = Inf),
            alpha = 0.2, fill = "green") +
  annotate("text", x = observation_date, y = Inf,
           label = "Observation Date", vjust = 1.5, color = "red") +
  annotate("text", x = target_start + (target_end - target_start)/2,
           y = Inf, label = "Target Period", vjust = 1.5, color = "darkgreen") +
  labs(
    title = "Transaction Timeline with Observation and Target Periods",
    x = "Date",
    y = "Number of Transactions",
    caption = "Red line: Observation date | Green area: Target period"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))

print(timeline_plot)
```

5 Exercise 3: Population Construction (25 points)

5.1 Task 3.1: Define Eligibility Criteria (5 points)

For this campaign, define the eligible population as customers who:

1. Made **at least 2 purchases** in the 6 months before the observation date
2. Have **not made a purchase** in the 30 days immediately before the observation date

Explain the business rationale for each criterion:

Criterion 1 Rationale:

Criterion 2 Rationale:

5.2 Task 3.2: Implement Population Filters (15 points)

Implement the eligibility criteria using set operations:

```
# Define the 6-month window before observation
six_months_before <- observation_date - months(6)

# Criterion 1: Customers with >= 2 purchases in last 6 months
customers_criterion1 <- transactions_predictors %>%
  filter(transaction_date >= ___ & transaction_date < ___) %>%
  group_by(customer_id) %>%
  summarize(n_purchases = n(), .groups = "drop") %>%
  filter(n_purchases >= ___) %>%
  pull(customer_id)

cat("Customers meeting criterion 1:", length(customers_criterion1), "\n")

# Criterion 2: Customers with NO purchases in last 30 days
thirty_days_before <- observation_date - days(30)

customers_recent_purchase <- transactions_predictors %>%
  filter(transaction_date >= ___ & transaction_date < ___) %>%
  pull(customer_id) %>%
  unique()

# Customers to INCLUDE (meet criterion 2)
customers_criterion2 <- setdiff(___, ___)

cat("Customers with recent purchases (to exclude):",
    length(customers_recent_purchase), "\n")
cat("Customers meeting criterion 2:", length(customers_criterion2), "\n")

# Final population: Intersection of both criteria
eligible_population <- intersect(___, ___)
```

```r
cat("\nFinal eligible population:", length(eligible_population), "\n")
cat("Percentage of all customers:",
    round(100 * length(eligible_population) / n_customers, 2), "%\n")
```

5.3 Task 3.3: Population Analysis (5 points)

Analyze the characteristics of the eligible population:

```r
# Compare eligible vs. non-eligible customers
population_analysis <- transactions_predictors %>%
  mutate(eligible = customer_id %in% eligible_population) %>%
  group_by(eligible) %>%
  summarize(
    n_customers = n_distinct(customer_id),
    n_transactions = n(),
    total_spend = sum(transaction_amount),
    avg_spend_per_customer = total_spend / n_customers,
    avg_transaction_amount = mean(transaction_amount),
    .groups = "drop"
  )

print(population_analysis)
```

Question: How do eligible customers differ from non-eligible customers in terms of purchasing behavior?

6 Exercise 4: Target Variable Definition (20 points)

6.1 Task 4.1: Define Binary Target (10 points)

Create a binary target variable indicating whether each customer in the eligible population made at least one purchase during the target period:

```
# Identify customers who purchased during target period
customers_purchased_target <- transactions_target %>%
  filter(customer_id %in% eligible_population) %>%
  pull(customer_id) %>%
  unique()

# Create target variable for eligible population
target_binary <- tibble(
  customer_id = eligible_population
) %>%
  mutate(
    target_purchase = if_else(customer_id %in% ___, ___, ___)
  )

# Calculate target statistics
target_summary <- target_binary %>%
  summarize(
    n_customers = n(),
    n_purchased = sum(target_purchase),
    n_not_purchased = sum(target_purchase == 0),
    response_rate = mean(target_purchase),
    .groups = "drop"
  )

print(target_summary)
```

Questions:

a) What is the response rate (percentage of customers who purchased)?
b) Is the target balanced or imbalanced?
c) What implications does this have for modeling?

Your Answers:

a) _____

b) _____

c) _____

_____

6.2 Task 4.2: Define Continuous Target (10 points)

Create an alternative continuous target: total spending during the target period:

```r
# Calculate total spending during target period for eligible customers
target_continuous <- transactions_target %>%
  filter(customer_id %in% eligible_population) %>%
  group_by(customer_id) %>%
  summarize(target_spending = sum(transaction_amount), .groups = "drop") %>%
  right_join(tibble(customer_id = eligible_population), by = "customer_id") %>%
  mutate(target_spending = replace_na(target_spending, 0))

# Summary statistics
spending_summary <- target_continuous %>%
  summarize(
    mean_spending = mean(target_spending),
    median_spending = median(target_spending),
    sd_spending = sd(target_spending),
    min_spending = min(target_spending),
    max_spending = max(target_spending),
    pct_zero = mean(target_spending == 0) * 100
  )

print(spending_summary)

# Visualize distribution
ggplot(target_continuous, aes(x = target_spending)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "white") +
  labs(
    title = "Distribution of Target Period Spending",
    x = "Total Spending ($)",
    y = "Number of Customers"
  ) +
  theme_minimal()
```

7 Exercise 5: Complete Basetable Construction (20 points)

7.1 Task 5.1: Calculate Predictive Features (15 points)

For each customer in the eligible population, calculate the following features using data from **before** the observation date:

7.1.1 RFM Features

```
# Calculate Recency, Frequency, Monetary features
rfm_features <- transactions_predictors %>%
  filter(customer_id %in% eligible_population) %>%
  group_by(customer_id) %>%
  summarize(
    # Recency: days since last purchase before observation date
    recency_days = as.numeric(observation_date - max(transaction_date)),

    # Frequency: total number of purchases
    frequency = n(),

    # Monetary: total spending
    monetary_total = sum(transaction_amount),

    # Additional features
    monetary_avg = mean(transaction_amount),
    monetary_max = max(transaction_amount),
    monetary_min = min(transaction_amount),

    .groups = "drop"
  )

head(rfm_features, 10)
```

7.1.2 Time-Based Features

```
# Calculate features for different time windows
time_features <- transactions_predictors %>%
  filter(customer_id %in% eligible_population) %>%
  mutate(
    days_before_obs = as.numeric(observation_date - transaction_date)
  ) %>%
  group_by(customer_id) %>%
  summarize(
    # Last 3 months
    purchases_3m = sum(days_before_obs <= 90),
    spending_3m = sum(transaction_amount[days_before_obs <= 90]),

    # Last 6 months
    purchases_6m = sum(days_before_obs <= 180),
    spending_6m = sum(transaction_amount[days_before_obs <= 180]),

    # Last 12 months
    purchases_12m = sum(days_before_obs <= 365),
    spending_12m = sum(transaction_amount[days_before_obs <= 365]),
```

```
    .groups = "drop"
  )

head(time_features, 10)
```

7.1.3 Category Preferences

```
# Calculate category-based features
category_features <- transactions_predictors %>%
  filter(customer_id %in% eligible_population) %>%
  group_by(customer_id, product_category) %>%
  summarize(n = n(), .groups = "drop") %>%
  pivot_wider(
    names_from = product_category,
    values_from = n,
    values_fill = 0,
    names_prefix = "category_"
  )

head(category_features, 10)
```

7.2 Task 5.2: Construct Final Basetable (5 points)

Combine all features with the target variable to create the final basetable:

```
# Merge all feature sets
basetable <- eligible_population %>%
  tibble(customer_id = .) %>%
  left_join(rfm_features, by = "customer_id") %>%
  left_join(time_features, by = "customer_id") %>%
  left_join(category_features, by = "customer_id") %>%
  left_join(target_binary, by = "customer_id") %>%
  left_join(target_continuous %>%
              select(customer_id, target_spending),
            by = "customer_id")

# Display structure
cat("Basetable dimensions:", nrow(basetable), "rows x",
    ncol(basetable), "columns\n\n")

cat("Column names:\n")
print(names(basetable))

# Display first few rows
head(basetable, 10) %>%
  kable(caption = "Final Basetable (First 10 Rows)",
        booktabs = TRUE,
        digits = 2) %>%
  kable_styling(latex_options = c("striped", "hold_position", "scale_down"))
```

7.3 Task 5.3: Validate Basetable (Bonus: 5 points)

Perform validation checks on the final basetable:

```r
# 1. Check for missing values
missing_summary <- basetable %>%
  summarize(across(everything(), ~sum(is.na(.)))) %>%
  pivot_longer(everything(), names_to = "variable", values_to = "n_missing") %>%
  filter(n_missing > 0)

if(nrow(missing_summary) > 0) {
  cat("Variables with missing values:\n")
  print(missing_summary)
} else {
  cat("No missing values found.\n")
}

# 2. Check for temporal leakage
# Verify that recency is never negative or zero
temporal_check <- basetable %>%
  filter(recency_days <= 0) %>%
  nrow()

cat("\nCustomers with recency <= 0 (temporal leakage):", temporal_check, "\n")

# 3. Verify feature-target relationship
# Quick correlation check
if("target_purchase" %in% names(basetable)) {
  numeric_features <- basetable %>%
    select(where(is.numeric), -customer_id, -contains("target")) %>%
    names()

  correlations <- basetable %>%
    summarize(across(all_of(numeric_features),
                     ~cor(., target_purchase, use = "complete.obs"))) %>%
    pivot_longer(everything(), names_to = "feature", values_to = "correlation") %>%
    arrange(desc(abs(correlation)))

  cat("\nTop 5 features by correlation with target:\n")
  print(head(correlations, 5))
}

# 4. Summary statistics
cat("\nBasetable Summary Statistics:\n")
summary(basetable %>% select(-customer_id))
```

8 Exercise 6: Reflection and Critical Thinking (Bonus: 10 points)

Answer the following questions based on your work:

8.1 Question 1: Timeline Integrity

Explain why it is critical that all predictive features are calculated using only data from **before** the observation date. What would happen if we violated this rule?

8.2 Question 2: Population Definition

The population was defined using two specific criteria (at least 2 purchases in 6 months, no purchases in last 30 days).

a) What are alternative ways to define the eligible population?
b) How might different population definitions affect model performance?

8.3 Question 3: Target Variable Selection

We created both binary (purchased yes/no) and continuous (total spending) target variables.

a) When would you choose one over the other?
b) What are the implications of each choice for model selection and evaluation?

8.4 Question 4: Feature Engineering

Looking at the features you calculated (RFM, time-based, category):

a) Which features do you expect to be most predictive and why?
b) What additional features could you calculate that might improve predictions?

8.5 Question 5: Practical Application

Imagine you are presenting this basetable to business stakeholders:

a) How would you explain the timeline concept in non-technical terms?
b) What business insights can you derive from the basetable statistics?

9 Submission Checklist

Before submitting your work, ensure you have completed:

- ☐ **Exercise 1**: Data exploration and quality checks
  - ☐ Calculated summary statistics
  - ☐ Checked for data quality issues
  - ☐ Created category summary table
- ☐ **Exercise 2**: Timeline definition and partitioning
  - ☐ Defined all key dates correctly
  - ☐ Partitioned data into predictor and target periods
  - ☐ Created timeline visualization
- ☐ **Exercise 3**: Population construction
  - ☐ Implemented eligibility criteria
  - ☐ Used set operations correctly
  - ☐ Analyzed population characteristics
- ☐ **Exercise 4**: Target variable definition
  - ☐ Created binary target variable
  - ☐ Created continuous target variable
  - ☐ Calculated and interpreted response rates
- ☐ **Exercise 5**: Complete basetable construction
  - ☐ Calculated RFM features
  - ☐ Calculated time-based features
  - ☐ Calculated category features
  - ☐ Merged all components into final basetable
  - ☐ Validated basetable integrity
- ☐ **Exercise 6** (Bonus): Reflection questions
  - ☐ Answered all five questions thoughtfully
- ☐ **Code Quality**:
  - ☐ Code is well-commented
  - ☐ Variable names are descriptive
  - ☐ Code runs without errors
  - ☐ Output is clearly labeled
- ☐ **Documentation**:
  - ☐ All questions are answered
  - ☐ Interpretations are provided where requested
  - ☐ Visualizations include proper labels and titles

10 Appendix: Useful R Code Snippets

10.1 Date Manipulation with lubridate

```r
# Subtract months from a date
six_months_ago <- observation_date - months(6)

# Subtract days from a date
thirty_days_ago <- observation_date - days(30)

# Calculate difference between dates
days_difference <- as.numeric(date2 - date1)

# Extract year, month, day
year(my_date)
month(my_date)
day(my_date)

# Floor date to beginning of period
floor_date(my_date, "month")  # First day of month
floor_date(my_date, "week")   # Beginning of week
```

10.2 Set Operations

```r
# Union: elements in A OR B
union(set_a, set_b)

# Intersection: elements in A AND B
intersect(set_a, set_b)

# Difference: elements in A but NOT in B
setdiff(set_a, set_b)

# Check membership
element %in% set
```

10.3 Common tidyverse Patterns

```r
# Filter rows based on condition
df %>% filter(variable > threshold)

# Select specific columns
df %>% select(col1, col2, col3)

# Create new variables
df %>% mutate(new_var = expression)

# Group and summarize
df %>%
  group_by(group_var) %>%
  summarize(
    mean_value = mean(value),
    count = n()
```

```
  )

# Join two datasets
left_join(df1, df2, by = "key_column")

# Reshape data
pivot_wider(df, names_from = category, values_from = value)
pivot_longer(df, cols = c(col1, col2), names_to = "name", values_to = "value")
```

10.4 Handling Missing Values

```
# Check for missing values
sum(is.na(df$variable))

# Remove rows with any missing values
df %>% drop_na()

# Replace missing values with zero
df %>% mutate(variable = replace_na(variable, 0))

# Replace missing with mean
df %>% mutate(variable = if_else(is.na(variable),
                                 mean(variable, na.rm = TRUE),
                                 variable))
```

11 Solutions Template

This section is for the instructor. Students should not have access to this in the distributed version.

End of Classwork 1

For questions or clarifications, please contact:

Prof. Asc. Endri Raco, Ph.D.
Department of Mathematical Engineering
Polytechnic University of Tirana
Email: endri.raco@fimif.edu.al