ACC 424 Accounting Information System

Accounts Receivable Aging Analysis and Visualization
Python notes for 3.1-3.3

Section 001 MW 9:30 AM – 10:45 AM at Rm. 257 over Jan 13 – May 07
Section 002 MW 11:00 AM – 12:15 AM at Rm. 127 over Jan 13 – May 07

3.1. *Why Accounts Receivable Aging Analysis?*

Accounts Receivable (AR) Aging Analysis provides organizations with critical insights into their receivables management. This analytical approach helps companies understand and optimize their **collection processes** while maintaining healthy **customer relationships**.

At its core, AR Aging Analysis helps evaluate **collection efficiency** by providing a clear picture of payment patterns. Through systematic analysis, companies can identify **slow-paying** customers and implement **targeted collection strategies**. This understanding allows organizations to take **proactive** measures before payment delays significantly impact cash flow.

The analysis also provides a foundation for calculating **expected credit losses**. Organizations can use historical aging patterns to estimate potential **uncollectible** accounts and establish appropriate **allowances** for **doubtful** accounts. This forward-looking approach helps maintain accurate financial statements and comply with accounting standards.

Furthermore, AR Aging Analysis improves **cash flow management**. By tracking the age of receivables, companies can better predict cash inflows and identify trends that might affect working capital. This insight enables more effective treasury management and helps organizations maintain **adequate liquidity**.

Perhaps most importantly, AR Aging Analysis supports **strategic decision-making** regarding **credit** policies and **customer relationships**. Organizations can use aging data to refine **credit terms**, adjust **credit limits**, and develop **customized** collection approaches for different customer segments. This strategic approach helps balance the need for timely collections with maintaining positive customer relationships.

3.2. Key Components of Accounts Receivable Aging Analysis

Accounts Receivable Aging Analysis rests on two primary components: **Time Buckets** and **Analysis Metrics**, which work together to provide a complete picture of **receivables health**.

Time Buckets represent the systematic **categorization** of receivables based on the **elapsed time** from the **invoice date**. This temporal classification starts with Current receivables, which represent amounts not yet due for payment. The analysis then progresses through increasingly concerning time periods: 1-30 days past due captures recent delays, 31-60 days past due indicates emerging collection issues, 61-90 days past due suggests significant payment problems, and Over 90 days past due often signals possible bad debts.

Analysis Metrics provides the quantitative framework for evaluating **receivables** within these time buckets. The percentage of total receivables in each **aging bucket** serves as a key indicator of overall collection effectiveness. For instance, a high percentage in current and 1-30 day buckets typically indicates **healthy** collection practices, while significant amounts in **older buckets** may signal **collection problems**.

The Average Collection Period measures the typical time taken for **invoices** to be paid, helping organizations understand their **cash conversion** cycle. This metric directly impacts **working capital** management and can influence decisions about **credit terms** and **collection** policies.

Ex. Past Due Categories: Take today's date and subtract the due date of each invoice. Based on the number of days past due, assign to appropriate buckets:

- 1-30 days: Invoice is up to one month late
- 31-60 days: Invoice is between one and two months late
- 61-90 days: Invoice is between two and three months late
- Over 90 days: Invoice is more than three months late

Today's date: February 15, 2025. Invoice due date: January 20, 2025.

Calculation: February 15 - January 20 = 26 days past due Result: This invoice goes in the **"1-30 days"** bucket

Analysis Metrics Calculations:

1. Percentage in Each Bucket: This calculation helps understand what portion of total receivables falls into each aging category. First, we sum all outstanding amounts in each time bucket. Then, to find the percentage, we divide each bucket's total by the total receivables and multiply by 100.
    - Sum the dollar amount in each bucket
    - Divide each bucket's sum by total receivables
    - Multiply by 100 for percentage
    - Example: 1-30 days bucket total: $50,000

      Total AR: $200,000

      Calculation: ($50,000 ÷ $200,000) × 100 = 25%. This tells us that 25% of all receivables are 1-30 days past due, providing insight into collection patterns.
2. Average Collection Period: This metric shows **how long**, on average, it takes to **collect payments**. We calculate it by: Multiplying each invoice's amount by its days outstanding; Adding all these products together; and then Dividing by the total invoice amount.
    - Sum of (Days Outstanding × Invoice Amount)
    - Divided by Total Invoice Amount
    - Example: Invoice 1: 20 days × $1,000 = 20,000 day-dollars

      Invoice 2: 45 days × $2,000 = 90,000 day-dollars

Total = 110,000 day-dollars ÷ $3,000 = 36.7 days average. This 36.7 day represents a weighted average that accounts for the two invoice amounts. The larger invoice (Invoice 2) has more influence on the final average because it represents a bigger portion of the total receivable. It takes an average of 36.7 days to **collect payment** after invoicing.

3. Bad Debt Estimation: Apply increasing risk percentages to each bucket: This calculation uses risk percentages that **increase** with age to estimate potential losses. The percentages show that **older** receivables are **less likely** to be collected:
    - Current: 0.5%
    - 1-30 days: 2%
    - 31-60 days: 5%
    - 61-90 days: 10%
    - Over 90 days: 25%

Example: Bucket amount × Risk percentage = Expected Loss

$10,000 in 61-90 days × 10% = $1,000 expected loss. This estimates we **might not** collect $1,000 of this amount for 61-90 days aging bucket.

3.3. *Python coding of Accounts Receivable Aging Analysis and Visualization*

https://anaconda.cloud/share/notebooks/e73bd848-9f05-4f59-98f1-eee23297ea5f/overview

Please use the above weblink to download the related Python coding of **Accounts Receivable Aging Analysis and Visualization**.

1. Importing Required **Libraries**

```
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
```

- pandas → Used for handling **tabular** data (dataframes).
- numpy → Used for **numerical** operations.
- datetime → Used for **date** calculations.
- matplotlib.pyplot → Used for **visualization**.

2. Defining the AR Aging Analysis Class

```
class ARAgingAnalysis:
```

This class encapsulates all functions needed for AR aging analysis, including:

- Initializing parameters (e.g., risk rates)
- Assigning aging buckets
- Analyzing receivables
- Calculating key metrics
- Visualizing results

3. Initializing the Class

```
def __init__(self, as_of_date=None):
    """Initialize AR Aging Analysis with analysis date"""
    self.as_of_date = pd.to_datetime(as_of_date) if as_of_date else pd.Timestamp.now()
```

- The constructor method (__init__) initializes the **as-of date** for aging calculations.

- If no date is provided, it defaults to the **current date**.

Risk Rates for Bad Debt Estimation:

```
self.risk_rates = {
    'Current': 0.005,  # 0.5%
    '1-30': 0.02,      # 2%
    '31-60': 0.05,     # 5%
    '61-90': 0.10,     # 10%
    'Over 90': 0.25    # 25%
}
```

Defines the **risk rates** for bad debt estimation based on the aging bucket.

4. Assigning Aging Buckets

```
def assign_aging_bucket(self, days_outstanding):
    """Assign aging bucket based on days outstanding"""
    if days_outstanding <= 0:
        return 'Current'
    elif days_outstanding <= 30:
        return '1-30'
    elif days_outstanding <= 60:
        return '31-60'
    elif days_outstanding <= 90:
        return '61-90'
    else:
        return 'Over 90'
```

This function assigns an invoice to an **aging bucket** based on days_outstanding (how many days the payment is overdue).

5. Analyzing Accounts Receivable Data

```
def analyze_receivables(self, data):
    """
    Analyze accounts receivable data
    Parameters:
    data: DataFrame with columns [invoice_date, due_date, customer, amount, payments]
    """
```

This function takes a **pandas DataFrame** containing AR data with columns:

- invoice_date: The date when the invoice was issued.
- due_date: The date when payment is due.
- customer: The customer associated with the invoice.
- amount: The total invoice amount.
- payments: The amount already paid.

Processing the Data:

```
df = data.copy()
df['invoice_date'] = pd.to_datetime(df['invoice_date'])
df['due_date'] = pd.to_datetime(df['due_date'])
```

- Creates a **copy** of the input DataFrame.
- Converts invoice_date and due_date columns to **datetime** format.

Calculating Days Outstanding:

```
df['days_outstanding'] = (self.as_of_date - df['due_date']).dt.days
```

Compute the number of days past the due date (days_outstanding).

Calculating the Remaining Balance:

```
df['balance'] = df['amount'] - df['payments']
```

Determines the **outstanding balance** for each invoice.

Assigning Aging Buckets:

```
df['aging_bucket'] = df['days_outstanding'].apply(self.assign_aging_bucket)
```

Uses the assign_aging_bucket function to classify each invoice into an **aging** category.

Return Processed Data for Further Analysis:

```
return self.calculate_metrics(df)
```

Calls calculate_metrics to compute key financial metrics.

6. Calculating Key Metrics

```
def calculate_metrics(self, df):
```

This function computes:

- **Aging bucket totals**
- **Percentage of receivables in each bucket**
- **Average collection period**
- **Estimated bad debt amount**

(1) Compute Bucket Totals

```
bucket_totals = df.groupby('aging_bucket')['balance'].sum()
total_receivables = bucket_totals.sum()
```

- Groups data by aging_bucket and sums the **balances** in each bucket.
- Computes total_receivables (sum of all outstanding invoices).

(2) Compute Bucket Percentages

```
bucket_percentages = (bucket_totals / total_receivables * 100).round(2)
```

- Calculates the percentage of each **aging bucket** relative to **total receivables**.

(3) Compute Average Collection Period

```
avg_collection_period = (
    (df['days_outstanding'] * df['balance']).sum() / df['balance'].sum()
).round(2)
```

- Uses a **weighted average formula** to compute the **average collection period**.

(4) Estimate Bad Debt

```
bad_debt_estimate = sum(
    bucket_totals[bucket] * self.risk_rates.get(bucket, 0)
    for bucket in bucket_totals.index
)
```

- Multiplies each aging bucket's total amount by its **default risk rate** to estimate **bad debt**.

(5) Return Results as Dictionary

```
return {
    'detailed_aging': df,
    'bucket_totals': bucket_totals,
    'bucket_percentages': bucket_percentages,
    'avg_collection_period': avg_collection_period,
    'bad_debt_estimate': bad_debt_estimate,
    'total_receivables': total_receivables
}
```

- Returns key metrics.

7. **Visualizing the Aging Data**

```
def visualize_aging(self, results):
```

Creates two charts:

- **Pie Chart** → Distribution of receivables across aging buckets.
- **Bar Chart** → Outstanding balances per aging bucket

Pie Chart:

```
results['bucket_percentages'].plot(
    kind='pie', autopct='%1.1f%%', ax=ax1, title='AR Aging Distribution'
)
```

- Displays the **percentage distribution** of receivables.

Bar Chart:

```
results['bucket_totals'].plot(
    kind='bar', ax=ax2, title='AR Aging Amounts', color='skyblue'
)
```

- Shows **total outstanding balances** per aging bucket.

8. Running the Analysis

```
if __name__ == "__main__":
```

Allows execution only when the script is run directly.

Sample data:

```
sample_data = pd.DataFrame({
    'invoice_date': ['2024-01-01', '2024-01-15', '2023-12-01', '2023-11-15'],
    'due_date': ['2024-02-01', '2024-02-15', '2024-01-01', '2023-12-15'],
    'customer': ['Customer A', 'Customer B', 'Customer C', 'Customer A'],
    'amount': [1000.00, 2000.00, 1500.00, 3000.00],
    'payments': [0.00, 1000.00, 500.00, 0.00]
})
```

- Creates **sample invoices** with different **due dates** and **payments**.

Perform Analysis:

```
analyzer = ARAgingAnalysis(as_of_date='2024-02-15')
results = analyzer.analyze_receivables(sample_data)
```

- Instantiates the class and analyzes the **sample data**.

Print Key Results:

```
print(f"Total Receivables: ${results['total_receivables']:,.2f}")
```

- Displays financial insights.

Generate Charts:

```
analyzer.visualize_aging(results)
plt.show()
```

- Plots the aging analysis.