Data organization: Loans data activity

This week, we will prepare and organize the loans data so that we can conduct logistic regression next week. We will follow steps to inspect, organize, and clean the data.

Step 1: Load and inspect the data

```
# Load necessary libraries
library(tidyverse)



# Inspect the structure of the dataset
str(Week0910CALoans)

# View the first few rows of the dataset
head(Week0910CALoans)
```

- After loading the data, check for obvious issues, such as missing values or incorrect data types. What do you notice?

Step 2: Convert variables to appropriate data types as necessary

Example code:

```
library(dplyr)

Week0910CALoans <- Week0910CALoans %>% mutate(RevLineCrNum= ifelse(RevLineCr
== "Y", 1, 0))
```

- Which variables did you decide to convert?

• **Date Fields:** ApprovalDate, ChgOffDate, and DisbursementDate were converted from character to Date.

• **Numeric Fields:** Variables like Term, NoEmp, DisbursementGross, and BalanceGross were converted from character to numeric.

• **Binary Fields:** RevLineCr and LowDoc were recoded into binary indicators (1 = “Y”, 0 otherwise) after standardizing the text (e.g., trimming whitespace, converting to uppercase).

Data organization: Loans data activity

- What happens when you create a binary for *RevLineCr*, which had some unexpected values? Why, and what can you do about it?

  If you directly run ifelse(RevLineCr == "Y", 1, 0), any entries not exactly matching "Y" (like "Yes", " y", or missing values) are classified as 0. This can mislabel valid "yes" responses. To fix this, you can standardize the values first (e.g., use toupper(trimws())), or include multiple valid forms of "yes" in the condition. This ensures all affirmative responses are correctly identified.

- Are there any variables that you can't convert to the right data type? Why?

  Yes, certain columns should remain as text or factors:

  • **IDs/Codes with leading zeros** (e.g., NAICS) can lose important digits if converted to numeric.

  • **Name fields** (e.g., borrower name, bank name) and other purely text-based columns don't make sense as numeric or date.

  • **Unclear categorical variables** with too many unique values may be better left as text or converted to factors, depending on analysis needs.

Step 3: Identify and remove unnecessary or redundant variables; save as a new datafile.

1. Review the dataset for redundant variables (e.g., every observation is for a California location. Can we use this variable for modeling *Default*?).
2. Remove any variables that are not predictive or are irrelevant to the analysis. Major exception: ALWAYS retain the primary key/ID variable.
3. Save as a new file in case you need to go back to the original later. I like to export as a .csv – you can save it to any location you like

```
# Option to stop viewing variables like State without overwriting the file
View(Week0910CALoans[, !names(Week0910CALoans) %in% "State"])

# Option to remove irrelevant variables (i.e., those that are totally
redundant, provide no information or that we definitely won't use in the
analysis) and then download a new csv
Week0910CALoans_clean <- dplyr::select(Week0910CALoans, -State)
write.csv(Week0910CALoans_clean, "Week0910CALoans_clean.csv", row.names =
FALSE)
```

Data organization: Loans data activity

- Why would we save a new file with a subset of the variables? What are pros and cons of doing this?

  Pros:

  1. Makes the dataset more manageable: fewer columns to navigate and analyze.
  2. Reduces file size, which can improve performance in certain workflows.
  3. Clearly defines which variables are relevant for your current analysis or model.

  Cons:

  1. You lose immediate access to columns that might be needed later for new questions or analyses.
  2. If you discover a dropped variable is actually important, you must return to the original file and re-integrate it.

- When/why might you decide to use a dataset with every variable maintained (even if some are irrelevant)?

  • You might keep all variables if you're **unsure** which will matter in the future, or if you plan on **trying different modeling approaches** (e.g., new machine learning techniques that can handle more variables).

  • Some organizations require **full data retention** for compliance or auditing.

  • Maintaining the original dataset ensures **reproducibility**—you can always revisit dropped variables without losing history.

- Why don't we just drop the variables we are not using and forget about them forever?

  • You may discover **later** that a variable you initially considered irrelevant is actually predictive.

  • **Reproducibility and traceability** demand that you preserve the original data for verification, error-checking, or future research.

  • It's a good data governance practice to maintain a master dataset and create **versioned subsets** for specific purposes, rather than permanently discarding information.

Data organization: Loans data activity

Step 4: Create useful features for next week's modeling

Tasks:

1. Convert categorical variables such as *NewExist* into binary indicators (if not already done).
2. Create any new features that might be helpful (e.g., create a flag for loans that are secured by real estate).

```
# Ensure 'New' variable is correctly created (1 for New, 0 for Existing)
Week0910CALoans$New <- ifelse(Week0910CALoans$NewExist == "New", 1, 0)

# Real estate-backed loans (already provided but recheck for consistency)
Week0910CALoans$RealEstate <- as.factor(Week0910CALoans$RealEstate)

# Recession variable (check if it makes sense to keep)
Week0910CALoans$Recession <- as.factor(Week0910CALoans$Recession)
```

What other variables should we manage now?

Several columns may need attention before modeling, depending on your analytical goals:

•**UrbanRural** (0 = Undefined, 1 = Urban, 2 = Rural)

You might want to convert this into a factor (Urban, Rural, Undefined) or binary indicators (e.g., Urban = 1/0, Rural = 1/0) and decide how to handle the "0 = Undefined" cases (treat them as missing or a separate category).

• **FranchiseCode**

If it's just "00000" or "00001" for non-franchise, and another code for franchise, consider a binary variable (1 = Franchise, 0 = No Franchise). Verify whether any other codes exist.

• **MIS_Status** or **Default**

Make sure your default indicator is correct. If Default = 1 for charged-off loans, confirm it matches MIS_Status = "CHGOFF" and that there aren't any other status codes to consider.

• **ApprovalFY**

Depending on modeling needs, you might treat it as numeric (fiscal year) or a factor. If it spans many years, numeric might be fine; if you want to group by time, a factor could help.

• **Selected**

Data organization: Loans data activity

This is often used to flag training vs. test data. Keep it if you plan to do custom splits, otherwise remove if it's no longer needed.

• **Extra/Unexplained Variables** (e.g., xx, daysterm)

Review if they have any meaningful definition. If not, they might be dropped.

- Why is it so important to create binary variables?

  • **Model Requirements:** Many algorithms (e.g., logistic regression) require numeric inputs, and a binary 0/1 format is the simplest way to represent yes/no data.

  • **Interpretability:** Binary variables make it straightforward to interpret coefficients (a 1-unit change from 0 to 1) or feature importance in tree-based models.

  • **Consistency:** Standardizing categorical variables as 0/1 avoids confusion with varied text entries like "Y", "Yes", "N", or "No," which can lead to misclassifications.

  • **Data Efficiency:** Binary encoding can reduce dimensionality compared to, say, one-hot encoding with multiple levels, which is particularly helpful for large datasets.

  In short, binary variables both simplify modeling and ensure consistent, numeric data for algorithms that don't natively handle categorical text.

Step 5: Detect and deal with outliers in the data that could impact the model.

Tasks:

1. Use boxplots to identify outliers in numeric variables like DisbursementGross, BalanceGross, and GrAppv.
2.

```
# Visualizing outliers with boxplots for numeric columns
boxplot(loans_data$DisbursementGross, main = "Disbursement Gross")
boxplot(loans_data$BalanceGross, main = "Balance Gross")
boxplot(loans_data$GrAppv, main = "Gross Amount of Loan Approved")

# Handle outliers (optional: remove or transform outliers)
# Remove extreme outliers beyond the 99th percentile for simplicity
loans_data <- loans_data %>%
  filter(DisbursementGross < quantile(DisbursementGross, 0.99)) %>%
  filter(BalanceGross < quantile(BalanceGross, 0.99)) %>%
```

Data organization: Loans data activity

```
filter(GrAppv < quantile(GrAppv, 0.99))
```

What do the boxplots suggest about outliers? What would you like to do?

1. **DisbursementGross** and **Gross Amount of Loan Approved** both have a heavy cluster of points well above the main distribution, indicating high-value outliers.

2. **BalanceGross** appears to have a very narrow range, suggesting that either most values are zero or there's very little variation.

Next Steps / Options

• **Remove or Cap Extreme Values:** If you suspect these large values might distort the model, you could remove observations above a certain percentile (e.g., 99th) or apply a winsorizing approach.

• **Log Transformation:** Converting DisbursementGross and GrAppv to a log scale often helps when data span multiple orders of magnitude.

• **Keep Them If Valid:** If these outliers represent genuine high-dollar loans, it may be important to keep them for modeling default risk, especially if larger loans behave differently.

Step 6: Objective: Select the most important predictors for the model.

- Which variables do you think will best predict *Default*, based on the research the group conducted? Will you need to make any transformations to estimate your model next week?
- State one hypothesis that you design as a small group. ***We will share these with the whole group, so be ready for one person to read one hypothesis.***

Based on prior research and the variables in our dataset, some **key predictors** might include:

• **DisbursementGross** and **GrAppv (Loan Size)**: Larger loan amounts often carry higher default risk, although a log transformation may help if they are heavily skewed.

• **SBA_Appv / GrAppv (Portion Guaranteed)**: The proportion guaranteed by the SBA can influence default behavior.

• **New (Business Age)**: Startups (new businesses) typically have higher default rates than established firms.

Data organization: Loans data activity

• **NoEmp (Firm Size)**: Smaller firms may be more vulnerable to economic shifts.

• **RealEstate**: Loans backed by real estate may be safer.

• **Recession**: Loans active during an economic downturn could be more prone to default.

• **RevLineCr** / **LowDoc**: Revolving lines of credit and low-documentation loans may correlate with different default profiles.

Possible Transformations:

• **Log Transform** on heavily skewed variables (e.g., DisbursementGross, GrAppv) to reduce the impact of extreme values.

• **Binary Indicators** for categorical fields (e.g., New, RealEstate, Recession) to make them model-ready.

One Hypothesis

Hypothesis: *"New businesses (New = 1) are more likely to default on larger loan amounts, especially if those loans are not backed by real estate, compared to established businesses."*

This hypothesis combines the idea that being a new business, having a larger loan, and lacking collateral may jointly increase default risk.