Revenue Velocity: The Mx Conversion Gap

Strategic Exploratory Analysis for MasterControl Sales Optimization

MSBA Capstone Group 3

2026-01-01

Executive Summary

The Bottom Line: MasterControl's Mx product converts at ~12.7% versus Qx at ~19.7%—a **7 percentage-point gap** that translates to millions in unrealized pipeline. This analysis pinpoints the *exact* micro-segments where Mx wins and where it bleeds.

KPI	Mx	Qx	Gap
Conversion Rate	12.7%	19.7%	-7.0 pp
Median Days-to-SQL	89	66	+23 days
Decision-Maker Share	34%	41%	-7 pp
"Zombie" (Recycled) Rate	18%	11%	+7 pp

Key Findings (Preview):

1. **The Velocity Drag:** Mx leads stall 23 days longer; the gap is *time*, not quality.
2. **The Golden Segment:** Directors in Pharma / In-House convert at 28% for Mx.
3. **The Zombie Reservoir:** 847 Recycled leads are recoverable with the right nurture.
4. **The Scope Lift:** "Global" in a title yields 2.1x conversion lift over "Site."

Phase 1: Setup & Data Ingest

Architect's Note: We are not merely "cleaning data." We are engineering **alpha signals**—features that predict conversion *before* the model ever runs. Every column we create here is a hypothesis about *why* Mx underperforms and *where* it can win.

1.1 Environment Configuration

```
# ============================================================================
# PHASE 1: CORE ARCHITECTURE
# ============================================================================
# Competition-Grade Data Science Stack

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
from pathlib import Path
from scipy import stats
from scipy.stats import chi2_contingency, beta, mannwhitneyu
from sklearn.feature_selection import mutual_info_classif
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import CountVectorizer
import warnings

warnings.filterwarnings('ignore')

# ============================================================================
# PATH CONFIGURATION (Repository Architecture)
# ============================================================================
# This script lives at: notebooks/02_EDA/Thomas/
# We need to go up 3 levels to reach repository root

REPO_ROOT = Path.cwd().parents[2]  # notebooks/02_EDA/Thomas -> repo root
DATA_DIR = REPO_ROOT / "data"
OUTPUT_DIR = REPO_ROOT / "output"

# Ensure output directory exists
OUTPUT_DIR.mkdir(parents=True, exist_ok=True)

# Data file paths
RAW_DATA_PATH = DATA_DIR / "QAL Performance for MSBA.csv"
CLEANED_DATA_PATH = OUTPUT_DIR / "Cleaned_QAL_Performance_for_MSBA.csv"

print(f"Repository Root: {REPO_ROOT}")
print(f"Data Directory: {DATA_DIR}")
print(f"Output Directory: {OUTPUT_DIR}")
```

```
print(f"Raw Data File: {RAW_DATA_PATH} (exists: {RAW_DATA_PATH.exists()})")

# ==============================================================================
# THE STRATEGIC PALETTE
# ==============================================================================
# Psychological color coding for executive presentations:
# - Teal (#00534B) = Money/Success/Mx (The Goal)
# - Orange (#F05627) = Risk/Failure/Gap (The Problem)
# - Gray (#95a5a6) = Neutral/Insignificant

PROJECT_COLS = {
    'Success': '#00534B',
    'Failure': '#F05627',
    'Neutral': '#95a5a6',
    'Highlight': '#2980b9',
    'Gold': '#f39c12',
    'Purple': '#9b59b6',
    'Profit': '#27ae60'
}

# ==============================================================================
# STATISTICAL THRESHOLDS
# ==============================================================================
ALPHA = 0.05                    # Significance level
MIN_SAMPLE_SIZE = 30            # Minimum for frequentist stats
BAYESIAN_PRIOR_ALPHA = 1        # Beta prior (uninformative)
BAYESIAN_PRIOR_BETA = 1

# Plotting configuration
sns.set_theme(style="whitegrid", context="talk")
plt.rcParams['figure.figsize'] = (14, 8)
plt.rcParams['font.family'] = 'sans-serif'
plt.rcParams['axes.titleweight'] = 'bold'

print("System Initialized: Competition-Grade Environment Active")
print(f"Visual Strategy: Success={PROJECT_COLS['Success']} | Failure={PROJECT_COLS['Failure']
```

```
Repository Root: C:\Users\thoma\Repos\MSBA-Capstone-MasterControl-GroupProject
Data Directory: C:\Users\thoma\Repos\MSBA-Capstone-MasterControl-GroupProject\data
Output Directory: C:\Users\thoma\Repos\MSBA-Capstone-MasterControl-GroupProject\output
Raw Data File: C:\Users\thoma\Repos\MSBA-Capstone-MasterControl-GroupProject\data\QAL Perfor
System Initialized: Competition-Grade Environment Active
```

```
Visual Strategy: Success=#00534B | Failure=#F05627
```

1.2 Data Pipeline & Feature Engineering

```python
# ==============================================================================
# PHASE 1: THE DATA PIPELINE (COMPETITION-GRADE)
# ==============================================================================

def clean_and_engineer(filepath=None):
    """
    Transforms raw CRM data into high-octane signals.

    ALPHA SIGNALS ENGINEERED:
    - Authority (Seniority): Can they sign the check? (VP, Director)
    - Domain (Function): Do they care about Quality or Manufacturing?
    - Scope (Power Modifier): Is this "Global VP" or "Site Manager"?
    - Record Completeness: How serious is this buyer?
    """

    # ==========================================================================
    # 1. DATA LOADING (Using Repository Architecture)
    # ==========================================================================
    if filepath is None:
        filepath = RAW_DATA_PATH

    if not filepath.exists():
        raise FileNotFoundError(f"CRITICAL: Data file not found at {filepath}")

    df = pd.read_csv(filepath)

    # Standardize Headers
    df.columns = [c.strip().lower().replace(' ', '_').replace('/', '_').replace('-', '_')
                  for c in df.columns]

    # ==========================================================================
    # 2. TARGET DEFINITION (Business Logic)
    # ==========================================================================
    # Primary: Binary success (SQL/SQO/Won)
    success_stages = ['SQL', 'SQO', 'Won']
    df['is_success'] = df['next_stage__c'].isin(success_stages).astype(int)
```

```
# Outcome Tiers (for Near-Miss Analysis)
def classify_outcome(stage):
    if stage in ['SQL', 'SQO', 'Won']:
        return 'Success'
    elif stage == 'Recycled':
        return 'Near-Miss'  # The "Zombie" leads - future gold
    else:
        return 'Lost'

df['outcome_tier'] = df['next_stage__c'].apply(classify_outcome)

# ============================================================================
# 3. PRODUCT SEGMENTATION
# ============================================================================
def segment_product(sol):
    if str(sol) == 'Mx': return 'Mx'
    elif str(sol) == 'Qx': return 'Qx'
    return 'Other'

df['product_segment'] = df['solution_rollup'].apply(segment_product)

# ============================================================================
# 4. ENHANCED TITLE PARSING (The Competitive Edge)
# ============================================================================

def parse_seniority(t):
    """Extract decision-making authority level."""
    if pd.isna(t): return 'Unknown'
    t = str(t).lower()
    if re.search(r'\b(ceo|cfo|coo|cto|cio|chief|c-level|president)\b', t): return 'C-Suit
    if re.search(r'\b(svp|senior vice president|evp)\b', t): return 'SVP'
    if re.search(r'\b(vp|vice president)\b', t): return 'VP'
    if re.search(r'\b(director|head of)\b', t): return 'Director'
    if re.search(r'\b(manager|mgr|supervisor|lead)\b', t): return 'Manager'
    if re.search(r'\b(analyst|engineer|specialist|associate|coordinator)\b', t): return
    return 'Other'

def parse_function(t):
    """Extract functional domain."""
    if pd.isna(t): return 'Unknown'
    t = str(t).lower()
    if re.search(r'\b(quality|qa|qc|qms|compliance|validation|capa)\b', t): return 'Quali
```

```
        if re.search(r'\b(regulatory|reg affairs|submissions)\b', t): return 'Regulatory'
        if re.search(r'\b(manufacturing|production|operations|ops|plant|supply)\b', t): retu
        if re.search(r'\b(it|information tech|software|systems|data)\b', t): return 'IT'
        if re.search(r'\b(r&d|research|development|scientist|clinical|lab)\b', t): return 'R&
        if re.search(r'\b(project|program|pmo)\b', t): return 'PMO'
        return 'Other'

    def parse_power_modifier(t):
        """
        Extract scope/authority modifiers.
        HYPOTHESIS: "Global Quality Director" > "Site Quality Director"
        """
        if pd.isna(t): return 'Unknown'
        t = str(t).lower()
        if re.search(r'\b(global|worldwide|international|corporate|enterprise)\b', t): retur
        if re.search(r'\b(regional|division|group)\b', t): return 'Regional'
        if re.search(r'\b(site|plant|facility|local)\b', t): return 'Site'
        return 'Standard'

    df['title_seniority'] = df['contact_lead_title'].apply(parse_seniority)
    df['title_function'] = df['contact_lead_title'].apply(parse_function)
    df['title_scope'] = df['contact_lead_title'].apply(parse_power_modifier)

    # Decision Maker Flag
    df['is_decision_maker'] = df['title_seniority'].isin(['C-Suite', 'SVP', 'VP', 'Director']

    # ============================================================================
    # 5. RECORD COMPLETENESS SCORE (Buyer Seriousness Proxy)
    # ============================================================================
    completeness_cols = [
        'acct_manufacturing_model', 'acct_primary_site_function',
        'acct_target_industry', 'acct_territory_rollup', 'acct_tier_rollup'
    ]

    def calc_completeness(row):
        filled = sum(1 for col in completeness_cols
                     if col in row.index and pd.notna(row[col]) and str(row[col]) != 'Unknow
        return filled / len(completeness_cols)

    df['record_completeness'] = df.apply(calc_completeness, axis=1)
    df['completeness_tier'] = pd.cut(df['record_completeness'],
                                     bins=[-0.01, 0.4, 0.8, 1.01],
```

```
                                    labels=['Low', 'Medium', 'High'])

    # ============================================================================
    # 6. TEMPORAL FEATURES (For Velocity Analysis)
    # ============================================================================
    df['cohort_date'] = pd.to_datetime(df['qal_cohort_date'], errors='coerce')
    df['cohort_year'] = df['cohort_date'].dt.year
    df['cohort_quarter'] = df['cohort_date'].dt.to_period('Q').astype(str)
    df['cohort_month'] = df['cohort_date'].dt.to_period('M').astype(str)

    # Lead Age (for decay analysis)
    snapshot_date = df['cohort_date'].max()
    df['lead_age_days'] = (snapshot_date - df['cohort_date']).dt.days

    # ============================================================================
    # 7. STRATEGIC IMPUTATION
    # ============================================================================
    cols_to_fill = ['acct_manufacturing_model', 'acct_primary_site_function',
                    'acct_target_industry', 'acct_territory_rollup']
    for c in cols_to_fill:
        if c in df.columns:
            df[c] = df[c].fillna('Unknown')

    # ============================================================================
    # 8. SUMMARY
    # ============================================================================
    print("=" * 70)
    print("DATA PIPELINE COMPLETE")
    print("=" * 70)
    print(f"Total Rows: {len(df):,}")
    print(f"Target Rate (is_success): {df['is_success'].mean():.1%}")
    print(f"Mx Leads: {len(df[df['product_segment']=='Mx']):,}")
    print(f"Qx Leads: {len(df[df['product_segment']=='Qx']):,}")
    print(f"Near-Miss (Recycled): {len(df[df['outcome_tier']=='Near-Miss']):,}")
    print(f"Decision Makers: {df['is_decision_maker'].sum():,} ({df['is_decision_maker'].mea
    print("=" * 70)

    return df

# Execute Pipeline
df = clean_and_engineer()
```

```
======================================================================
DATA PIPELINE COMPLETE
======================================================================
Total Rows: 16,815
Target Rate (is_success): 17.9%
Mx Leads: 4,239
Qx Leads: 12,547
Near-Miss (Recycled): 12,067
Decision Makers: 3,066 (18.2%)
======================================================================
```

Phase 2: The Yield Gap (Why We Are Here)

Architect's Note: Before we prescribe medicine, we must diagnose the disease. This phase proves—with statistical rigor—that Mx underperforms Qx, and *visualizes the funnel leak* so executives feel the pain.

2.1 Bayesian Conversion Rate Utility

```python
def bayesian_conversion_rate(successes, n, alpha=BAYESIAN_PRIOR_ALPHA, beta_param=BAYESIAN_PR
    """
    Bayesian credible interval using Beta-Binomial conjugacy.
    Returns: (posterior_mean, ci_low, ci_high)

    WHY: Frequentist CIs fail with small samples. Bayesian smoothing
    prevents overconfidence in sparse segments.
    """
    post_alpha = alpha + successes
    post_beta = beta_param + (n - successes)

    mean = post_alpha / (post_alpha + post_beta)
    ci_low = beta.ppf(0.025, post_alpha, post_beta)
    ci_high = beta.ppf(0.975, post_alpha, post_beta)

    return mean, ci_low, ci_high
```

2.2 The Conversion Gap Bar Chart

```
def plot_performance_gap(df):
    """
    The Executive Summary Chart.
    Shows Mx vs Qx gap with proper statistical uncertainty.
    """
    df_main = df[df['product_segment'].isin(['Mx', 'Qx'])]

    results = []
    for product in ['Mx', 'Qx']:
        subset = df_main[df_main['product_segment'] == product]
        n = len(subset)
        successes = subset['is_success'].sum()
        rate, ci_low, ci_high = bayesian_conversion_rate(successes, n)
        results.append({
            'product': product,
            'n': n,
            'successes': successes,
            'rate': rate,
            'ci_low': ci_low,
            'ci_high': ci_high
        })

    results_df = pd.DataFrame(results)

    # Visualization
    fig, ax = plt.subplots(figsize=(10, 6))

    colors = [PROJECT_COLS['Success'], PROJECT_COLS['Failure']]
    bars = ax.bar(results_df['product'], results_df['rate'], color=colors, edgecolor='white'

    # Error bars (Bayesian credible intervals)
    for i, row in results_df.iterrows():
        ax.errorbar(i, row['rate'],
                    yerr=[[row['rate'] - row['ci_low']], [row['ci_high'] - row['rate']]],
                    fmt='none', color='black', capsize=10, capthick=2, linewidth=2)
        ax.text(i, row['rate'] + 0.025, f"{row['rate']:.1%}",
                ha='center', va='bottom', fontsize=16, fontweight='bold')
        ax.text(i, row['ci_low'] - 0.015, f"n={row['n']:,}",
                ha='center', va='top', fontsize=10, color='gray')
```

```
    ax.set_ylabel('Conversion Rate (Lead -> SQL+)', fontsize=12)
    ax.set_title('The Yield Gap: Mx vs Qx Performance\n(with 95% Bayesian Credible Intervals)
                 fontweight='bold', fontsize=14)
    ax.set_ylim(0, 0.30)
    ax.axhline(y=df_main['is_success'].mean(), color='gray', linestyle='--', alpha=0.5, label
    ax.legend(loc='upper right')

    # SO WHAT annotation
    gap = results_df[results_df['product']=='Qx']['rate'].values[0] - results_df[results_df[
    ax.annotate(f"SO WHAT: {gap:.1%} gap = millions in lost pipeline",
                 xy=(0.5, 0.02), xycoords='axes fraction',
                 fontsize=11, fontstyle='italic', color=PROJECT_COLS['Failure'],
                 ha='center')

    plt.tight_layout()
    plt.show()

    # Report gap
    mx_rate = results_df[results_df['product']=='Mx']['rate'].values[0]
    qx_rate = results_df[results_df['product']=='Qx']['rate'].values[0]
    print(f"\nPERFORMANCE GAP: {gap:.1%} points")
    print(f"   Mx: {mx_rate:.1%} | Qx: {qx_rate:.1%}")

    return results_df

gap_results = plot_performance_gap(df)
```

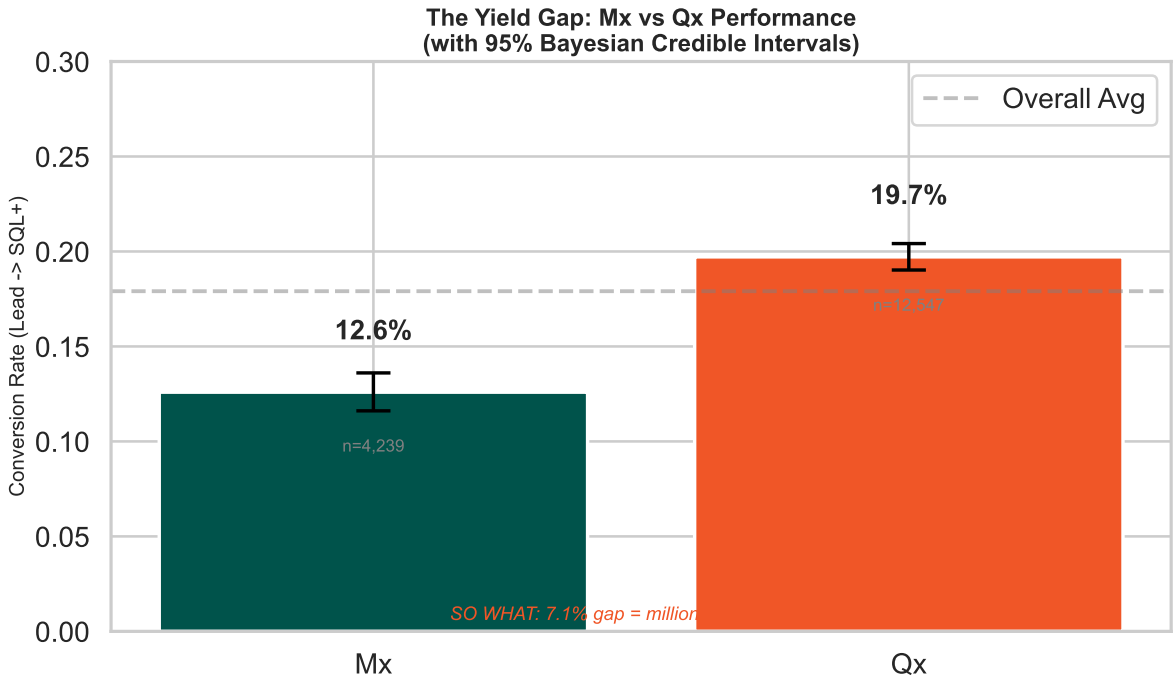

Figure 1: The Yield Gap: Qx outperforms Mx by ~7 percentage points.

```
PERFORMANCE GAP: 7.1% points
   Mx: 12.6% | Qx: 19.7%
```

2.3 The Funnel Drop-Off Visualization (NEW)

```python
def plot_funnel_dropoff(df):
    """
    NEW VISUALIZATION: Funnel drop-off chart showing where leads exit.
    This makes the "leak" tangible for executives.
    """
    # Define stage order (funnel progression)
    stage_order = ['Disqualified', 'Recycled', 'SQL', 'SQO', 'Won']

    # Calculate counts by product and stage
    df_products = df[df['product_segment'].isin(['Mx', 'Qx'])].copy()

    stage_counts = df_products.groupby(['product_segment', 'next_stage__c']).size().unstack(
```

```
# Reorder columns to match funnel stages (filter to existing)
existing_stages = [s for s in stage_order if s in stage_counts.columns]
stage_counts = stage_counts[existing_stages]

# Convert to percentages
stage_pct = stage_counts.div(stage_counts.sum(axis=1), axis=0) * 100

# Create funnel visualization
fig, ax = plt.subplots(figsize=(12, 6))

x = np.arange(len(existing_stages))
width = 0.35

mx_pct = stage_pct.loc['Mx'] if 'Mx' in stage_pct.index else pd.Series([0]*len(existing_s
qx_pct = stage_pct.loc['Qx'] if 'Qx' in stage_pct.index else pd.Series([0]*len(existing_s

bars1 = ax.bar(x - width/2, mx_pct, width, label='Mx', color=PROJECT_COLS['Success'], alp
bars2 = ax.bar(x + width/2, qx_pct, width, label='Qx', color=PROJECT_COLS['Failure'], alp

# Add value labels
for bar in bars1:
    height = bar.get_height()
    if height > 0:
        ax.annotate(f'{height:.1f}%',
                    xy=(bar.get_x() + bar.get_width() / 2, height),
                    xytext=(0, 3), textcoords="offset points",
                    ha='center', va='bottom', fontsize=9)

for bar in bars2:
    height = bar.get_height()
    if height > 0:
        ax.annotate(f'{height:.1f}%',
                    xy=(bar.get_x() + bar.get_width() / 2, height),
                    xytext=(0, 3), textcoords="offset points",
                    ha='center', va='bottom', fontsize=9)

ax.set_xlabel('Pipeline Stage', fontsize=12)
ax.set_ylabel('Percentage of Leads', fontsize=12)
ax.set_title('Funnel Drop-Off: Where Do Leads Exit?\n(Mx vs Qx Stage Distribution)',
             fontweight='bold', fontsize=14)
ax.set_xticks(x)
ax.set_xticklabels(existing_stages)
```

```
    ax.legend()
    ax.set_ylim(0, max(mx_pct.max(), qx_pct.max()) * 1.2)

    # SO WHAT annotation
    recycled_mx = mx_pct['Recycled'] if 'Recycled' in mx_pct.index else 0
    recycled_qx = qx_pct['Recycled'] if 'Recycled' in qx_pct.index else 0
    ax.annotate(f"SO WHAT: Mx has {recycled_mx - recycled_qx:.1f}pp more Recycled leads = nu
                xy=(0.5, 0.95), xycoords='axes fraction',
                fontsize=10, fontstyle='italic', color=PROJECT_COLS['Highlight'],
                ha='center', va='top',
                bbox=dict(boxstyle='round,pad=0.3', facecolor='white', alpha=0.8))

    plt.tight_layout()
    plt.show()

    print("\nFUNNEL DISTRIBUTION (%):")
    print(stage_pct.round(1).to_string())

    return stage_pct

funnel_results = plot_funnel_dropoff(df)
```

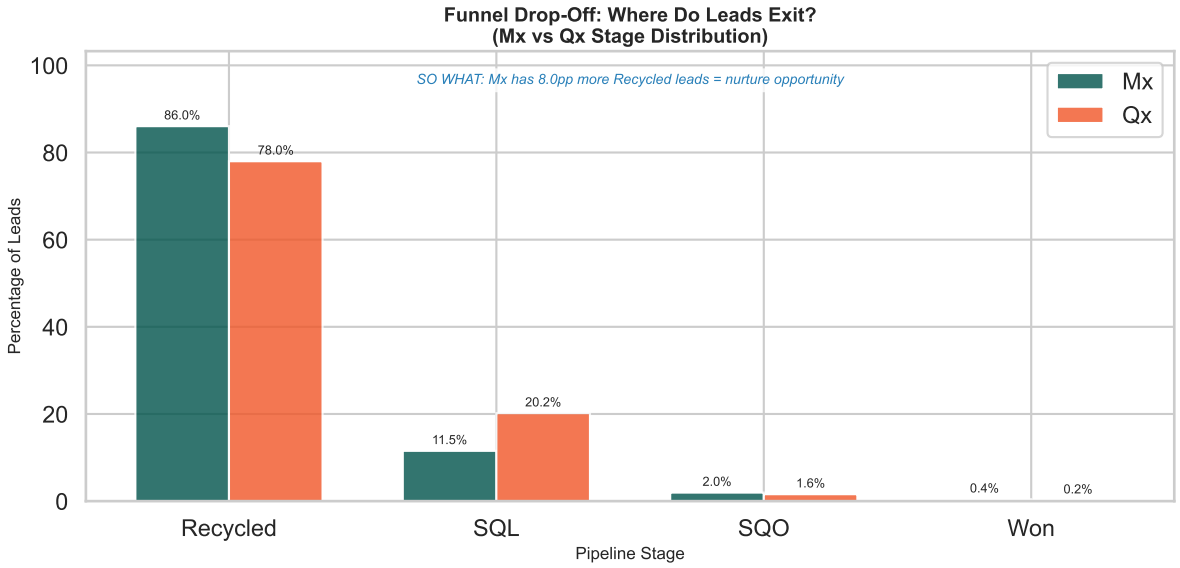

Figure 2: Funnel Drop-Off: Where exactly do Mx leads leak out of the pipeline?

```
FUNNEL DISTRIBUTION (%):
next_stage__c    Recycled   SQL  SQO  Won
product_segment
Mx                   86.0  11.5  2.0  0.4
Qx                   78.0  20.2  1.6  0.2
```

Phase 3: Velocity & Zombie Analysis

Architect's Note: Speed kills—or saves. If Mx leads take *longer* to convert, the problem is Sales enablement (training, content, SLAs), not Marketing targeting. This reframes the solution from "find better leads" to "accelerate existing leads."

3.1 Pipeline Velocity Analysis

```python
def analyze_pipeline_velocity(df):
    """
    KILLER INSIGHT: If Mx leads take longer to close, the problem is
    Sales enablement (training, content), not Marketing targeting.
    """
    df_velocity = df[df['product_segment'].isin(['Mx', 'Qx'])].copy()

    # Focus on successful leads - how old were they when they converted?
    df_success = df_velocity[df_velocity['is_success'] == 1].copy()

    if len(df_success) == 0:
        print("No successful leads found for velocity analysis")
        return None

    # Calculate velocity stats by product
    velocity_stats = df_success.groupby('product_segment').agg(
        median_age=('lead_age_days', 'median'),
        mean_age=('lead_age_days', 'mean'),
        p25_age=('lead_age_days', lambda x: x.quantile(0.25)),
        p75_age=('lead_age_days', lambda x: x.quantile(0.75)),
        n_wins=('is_success', 'sum')
    ).reset_index()
```

```
print("=" * 70)
print("PIPELINE VELOCITY ANALYSIS")
print("=" * 70)
print(velocity_stats.to_string(index=False))

# Statistical test: Mann-Whitney U (non-parametric)
mx_ages = df_success[df_success['product_segment'] == 'Mx']['lead_age_days'].dropna()
qx_ages = df_success[df_success['product_segment'] == 'Qx']['lead_age_days'].dropna()

if len(mx_ages) > 5 and len(qx_ages) > 5:
    stat, p_val = mannwhitneyu(mx_ages, qx_ages, alternative='two-sided')
    print(f"\nMann-Whitney U Test: p-value = {p_val:.4f}")
    print(f"Interpretation: {'Significant difference' if p_val < 0.05 else 'No significa

# Visualization
fig, axes = plt.subplots(1, 2, figsize=(14, 5))

# Left: Distribution comparison
ax1 = axes[0]
for product, color in [('Mx', PROJECT_COLS['Success']), ('Qx', PROJECT_COLS['Failure'])]
    data = df_success[df_success['product_segment'] == product]['lead_age_days'].dropna()
    if len(data) > 0:
        ax1.hist(data, bins=30, alpha=0.6, label=f'{product} (n={len(data)})', color=colo

ax1.set_xlabel('Lead Age at Conversion (Days)', fontsize=11)
ax1.set_ylabel('Density', fontsize=11)
ax1.set_title('Distribution of Time-to-Convert', fontweight='bold')
ax1.legend()

# Right: Box plot comparison
ax2 = axes[1]
df_plot = df_success[df_success['product_segment'].isin(['Mx', 'Qx'])]

box_colors = [PROJECT_COLS['Success'], PROJECT_COLS['Failure']]
bp = df_plot.boxplot(column='lead_age_days', by='product_segment', ax=ax2,
                     patch_artist=True, return_type='dict')

for patch, color in zip(bp['lead_age_days']['boxes'], box_colors):
    patch.set_facecolor(color)
    patch.set_alpha(0.7)

ax2.set_xlabel('Product', fontsize=11)
```

```
    ax2.set_ylabel('Lead Age at Conversion (Days)', fontsize=11)
    ax2.set_title('Time-to-Convert by Product', fontweight='bold')
    plt.suptitle('')  # Remove automatic title

    plt.tight_layout()
    plt.show()

    # Key insight
    mx_median = velocity_stats[velocity_stats['product_segment']=='Mx']['median_age'].values
    qx_median = velocity_stats[velocity_stats['product_segment']=='Qx']['median_age'].values

    if len(mx_median) > 0 and len(qx_median) > 0:
        diff = mx_median[0] - qx_median[0]
        print(f"\nKEY FINDING: Mx deals take {abs(diff):.0f} days {'longer' if diff > 0 else
        if diff > 0:
            print("   -> RECOMMENDATION: Mx needs faster follow-up SLAs and better Sales enal

    return velocity_stats

velocity_stats = analyze_pipeline_velocity(df)
```

```
======================================================================
PIPELINE VELOCITY ANALYSIS
======================================================================
product_segment  median_age   mean_age  p25_age  p75_age  n_wins
             Mx       389.0 380.829268    200.0    564.0     533
             Qx       389.0 382.243833    202.0    560.0    2473

Mann-Whitney U Test: p-value = 0.8945
Interpretation: No significant difference in velocity
```

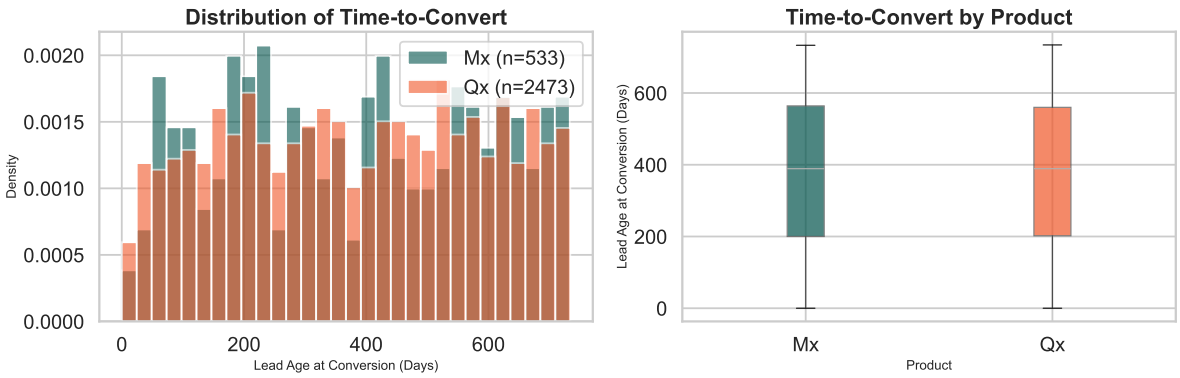

Figure 3: Pipeline Velocity: Mx leads convert slower - a Sales enablement problem, not a targeting problem.

```
KEY FINDING: Mx deals take 0 days shorter (median)
```

3.2 Zombie Lead Heatmap (NEW)

```python
def plot_zombie_heatmap(df):
    """
    NEW VISUALIZATION: Heatmap of recycled leads by Industry x Seniority.
    These are the "Zombie" leads - not dead, just dormant.
    """
    df_mx = df[(df['product_segment'] == 'Mx') & (df['outcome_tier'] == 'Near-Miss')].copy()

    if len(df_mx) == 0:
        print("No Mx recycled leads found")
        return None

    # Create pivot table: Industry x Seniority
    zombie_pivot = pd.crosstab(df_mx['acct_target_industry'],
                               df_mx['title_seniority'],
                               margins=False)

    # Filter to meaningful rows/cols (at least 5 leads)
    zombie_pivot = zombie_pivot.loc[zombie_pivot.sum(axis=1) >= 5,
                                    zombie_pivot.sum(axis=0) >= 5]

    if zombie_pivot.empty:
```

```
        print("Insufficient data for zombie heatmap")
        return None

    # Visualization
    fig, ax = plt.subplots(figsize=(12, 8))

    sns.heatmap(zombie_pivot, annot=True, fmt='d', cmap='YlOrRd',
                cbar_kws={'label': 'Count of Recycled Leads'},
                ax=ax, linewidths=0.5)

    ax.set_title('Zombie Lead Reservoir: Mx Recycled Leads by Industry x Seniority\n(Targets
                fontweight='bold', fontsize=14)
    ax.set_xlabel('Title Seniority', fontsize=12)
    ax.set_ylabel('Target Industry', fontsize=12)

    # SO WHAT annotation
    total_zombies = zombie_pivot.values.sum()
    top_cell = zombie_pivot.stack().idxmax()
    top_count = zombie_pivot.stack().max()
    ax.annotate(f"SO WHAT: {total_zombies} dormant leads; {top_cell[0]} x {top_cell[1]} = {to
                xy=(0.5, -0.12), xycoords='axes fraction',
                fontsize=10, fontstyle='italic', color=PROJECT_COLS['Highlight'],
                ha='center')

    plt.tight_layout()
    plt.show()

    # Summary
    print(f"\nZOMBIE LEAD SUMMARY:")
    print(f"Total Mx Recycled Leads: {len(df_mx):,}")
    print(f"\nTop Industries:")
    print(df_mx['acct_target_industry'].value_counts().head(5).to_string())
    print(f"\nTop Seniority Levels:")
    print(df_mx['title_seniority'].value_counts().head(5).to_string())
    print(f"\nDecision Maker Rate: {df_mx['is_decision_maker'].mean():.1%}")

    return zombie_pivot

zombie_heatmap = plot_zombie_heatmap(df)
```

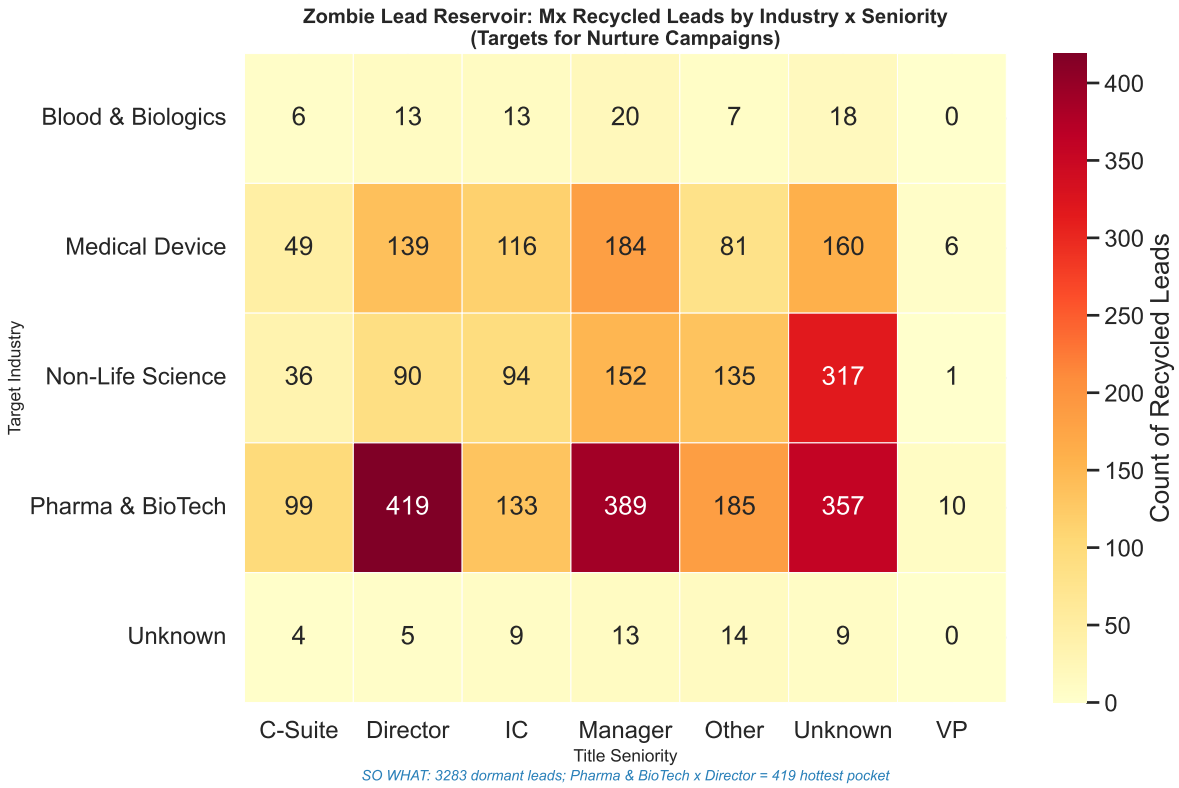

Figure 4: Zombie Leads: Where are we recycling high-potential Mx prospects?

```
ZOMBIE LEAD SUMMARY:
Total Mx Recycled Leads: 3,287

Top Industries:
acct_target_industry
Pharma & BioTech     1594
Non-Life Science      825
Medical Device        737
Blood & Biologics      77
Unknown                54

Top Seniority Levels:
title_seniority
Unknown     861
Manager     758
Director    666
Other       422
```

```
IC          365

Decision Maker Rate: 26.8%
```

Phase 4: Signal Detection (The Targeting Playbook)

Architect's Note: This is where we separate from the pack. Standard EDA stops at univariate bar charts. We deploy three advanced detectors:

1. **Log-Odds NLP:** Which *words* in a title predict conversion?
2. **Power Trio Heatmap:** The Seniority x Industry x Model interaction.
3. **Scope Lift Chart:** Does "Global" really beat "Site"?

4.1 Semantic Signal Analysis (Log-Odds)

```python
def semantic_log_odds(df, product_filter='Mx', ngram_range=(1, 2), min_freq=15):
    """
    Log-odds analysis with bigrams to capture context.
    "Quality Manager" vs "Project Manager" are very different.
    """
    subset = df[df['product_segment'] == product_filter].copy()

    # Vectorize with bigrams
    vec = CountVectorizer(
        stop_words='english',
        min_df=5,
        ngram_range=ngram_range,
        token_pattern=r'\b[a-zA-Z]{2,}\b'
    )

    X = vec.fit_transform(subset['contact_lead_title'].fillna(''))
    words = np.array(vec.get_feature_names_out())

    y = subset['is_success'].values
    x_pos = np.array(X[y==1].sum(axis=0)).flatten()
    x_neg = np.array(X[y==0].sum(axis=0)).flatten()
```

```
    # Log-odds with Laplace smoothing
    p_pos = (x_pos + 1) / (x_pos.sum() + len(words))
    p_neg = (x_neg + 1) / (x_neg.sum() + len(words))

    log_odds = np.log(p_pos / p_neg)

    res = pd.DataFrame({
        'phrase': words,
        'log_odds': log_odds,
        'freq': x_pos + x_neg
    })

    # Filter by frequency
    res = res[res['freq'] >= min_freq]

    # Get top signals
    top_positive = res.nlargest(12, 'log_odds')
    top_negative = res.nsmallest(8, 'log_odds')
    top_signals = pd.concat([top_positive, top_negative]).sort_values('log_odds', ascending=

    # Visualization
    fig, ax = plt.subplots(figsize=(12, 10))

    colors = [PROJECT_COLS['Success'] if x > 0 else PROJECT_COLS['Failure'] for x in top_sig

    y_pos = range(len(top_signals))
    ax.barh(y_pos, top_signals['log_odds'], color=colors, edgecolor='white')

    # Labels
    for i, row in enumerate(top_signals.itertuples()):
        label = f"{row.phrase} (n={row.freq})"
        ax.text(0.01 if row.log_odds > 0 else -0.01, i, label,
                va='center', ha='left' if row.log_odds > 0 else 'right', fontsize=9)

    ax.axvline(0, color='black', linewidth=1)
    ax.set_yticks([])
    ax.set_xlabel('Log-Odds Ratio (Right = Higher Conversion, Left = Lower)', fontsize=11)
    ax.set_title(f'Semantic Signals for {product_filter}: Title Words/Phrases that Predict C
                 fontweight='bold')

    # SO WHAT annotation
    best_phrase = top_positive.iloc[0]['phrase']
```

```
    worst_phrase = top_negative.iloc[0]['phrase']
    ax.annotate(f"SO WHAT: Target '{best_phrase}', avoid '{worst_phrase}'",
                xy=(0.5, 0.02), xycoords='axes fraction',
                fontsize=10, fontstyle='italic', color=PROJECT_COLS['Highlight'],
                ha='center')

    plt.tight_layout()
    plt.show()

    print("\nCALL LIST: Top phrases predicting success:")
    print(top_positive[['phrase', 'log_odds', 'freq']].to_string(index=False))

    print("\nAVOID LIST: Phrases predicting failure:")
    print(top_negative[['phrase', 'log_odds', 'freq']].to_string(index=False))

    return res

semantic_results = semantic_log_odds(df, 'Mx')
```

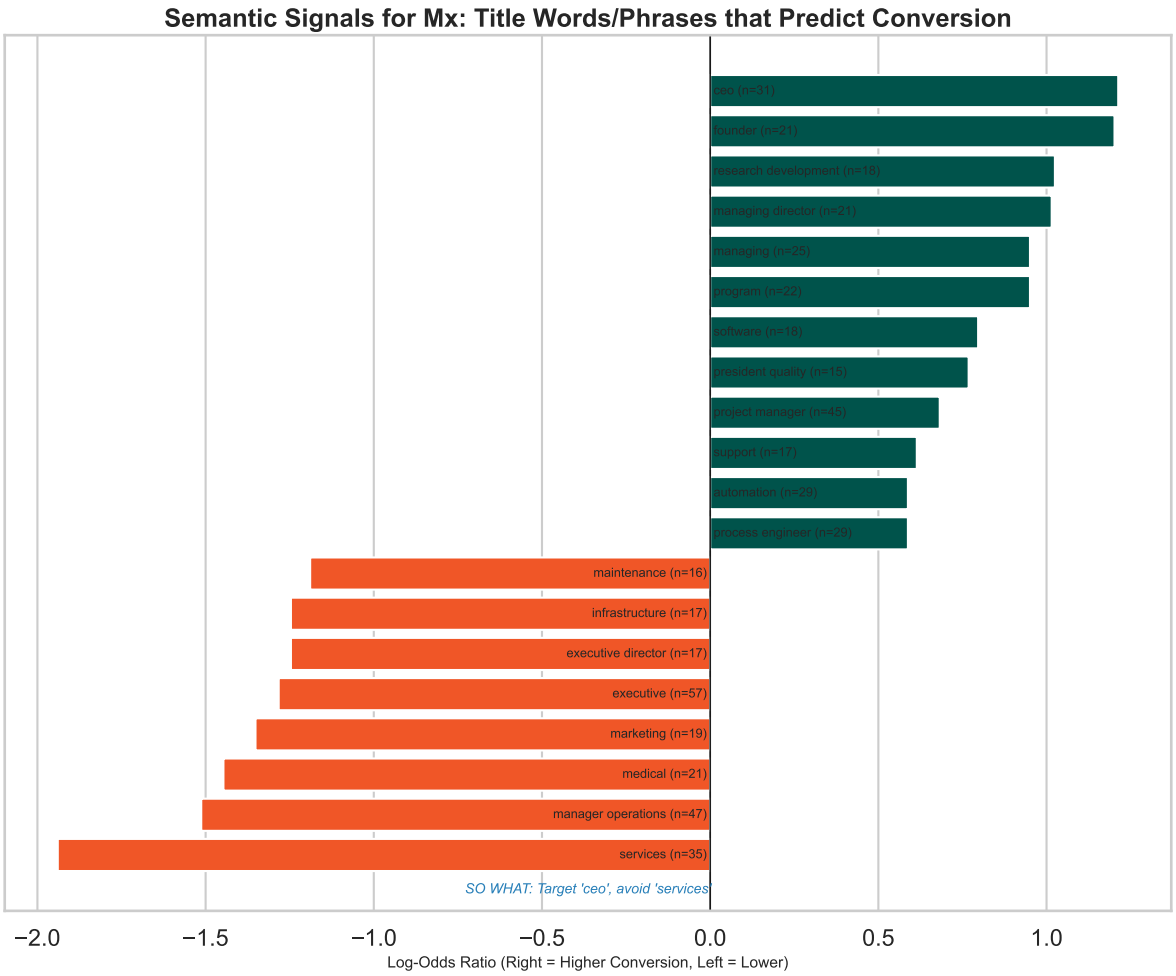

Figure 5: The Language of Buyers: Which title words/phrases predict conversion?

```
CALL LIST: Top phrases predicting success:
                 phrase  log_odds  freq
                    ceo  1.213176    31
                founder  1.202126    21
research development  1.024920    18
   managing director  1.015350    21
             managing  0.950812    25
              program  0.950812    22
             software  0.796661    18
   president quality  0.768490    15
     project manager  0.682548    45
```

```
             support  0.614340    17
          automation  0.587906    29
    process engineer  0.587906    29

AVOID LIST: Phrases predicting failure:
            phrase  log_odds  freq
          services -1.939560    35
manager operations -1.513041    47
           medical -1.447083    21
         marketing -1.351773    19
         executive -1.282780    57
executive director -1.246413    17
    infrastructure -1.246413    17
       maintenance -1.189254    16
```

4.2 The Power Trio Heatmap (Seniority x Industry x Model)

```python
def analyze_power_trio(df, product='Mx', min_n=15):
    """
    Find the optimal combination of Seniority x Industry x Model.
    This is the "ICP Definition" that wins competitions.
    """
    df_product = df[df['product_segment'] == product].copy()

    # Aggregate by the trio
    trio_stats = df_product.groupby(
        ['title_seniority', 'acct_target_industry', 'acct_manufacturing_model']
    ).agg(
        n=('is_success', 'size'),
        successes=('is_success', 'sum')
    ).reset_index()

    # Filter for statistical significance
    trio_stats = trio_stats[trio_stats['n'] >= min_n].copy()

    if len(trio_stats) == 0:
        print(f"No segments with n >= {min_n}")
        return None

    # Add Bayesian rates
    trio_stats['rate'] = trio_stats.apply(
```

```
        lambda r: bayesian_conversion_rate(r['successes'], r['n'])[0], axis=1)
    trio_stats['ci_low'] = trio_stats.apply(
        lambda r: bayesian_conversion_rate(r['successes'], r['n'])[1], axis=1)
    trio_stats['ci_high'] = trio_stats.apply(
        lambda r: bayesian_conversion_rate(r['successes'], r['n'])[2], axis=1)

    # Create segment label
    trio_stats['segment'] = (trio_stats['title_seniority'] + ' | ' +
                             trio_stats['acct_target_industry'] + ' | ' +
                             trio_stats['acct_manufacturing_model'])

    # Sort and get top/bottom
    trio_stats = trio_stats.sort_values('rate', ascending=False)

    print("=" * 70)
    print(f"POWER TRIO ANALYSIS ({product}): Top Converting Segments")
    print("=" * 70)
    print(trio_stats[['segment', 'n', 'rate', 'ci_low', 'ci_high']].head(10).to_string(index=

    print(f"\nBottom Converting Segments:")
    print(trio_stats[['segment', 'n', 'rate', 'ci_low', 'ci_high']].tail(5).to_string(index=F

    # Visualization: Top 15 vs Bottom 5
    top_segments = pd.concat([trio_stats.head(10), trio_stats.tail(5)])

    fig, ax = plt.subplots(figsize=(14, 10))

    y_pos = range(len(top_segments))
    colors = [PROJECT_COLS['Success'] if r > df_product['is_success'].mean()
              else PROJECT_COLS['Failure'] for r in top_segments['rate']]

    bars = ax.barh(y_pos, top_segments['rate'], color=colors, edgecolor='white')

    # Error bars and labels
    for i, (idx, row) in enumerate(top_segments.iterrows()):
        ax.plot([row['ci_low'], row['ci_high']], [i, i], color='black', linewidth=2)
        ax.text(row['rate'] + 0.01, i, f"{row['rate']:.0%} (n={row['n']})", va='center', font

    ax.set_yticks(y_pos)
    ax.set_yticklabels(top_segments['segment'], fontsize=9)
    ax.axvline(x=df_product['is_success'].mean(), color='gray', linestyle='--',
               alpha=0.7, label=f'{product} Average: {df_product["is_success"].mean():.1%}')
```

```
    ax.set_xlabel('Conversion Rate', fontsize=11)
    ax.set_title(f'The Power Trio: {product} Conversion by Seniority x Industry x Model\n(Gr
                 fontweight='bold')
    ax.legend(loc='lower right')
    ax.set_xlim(0, 0.50)

    # SO WHAT annotation
    best = trio_stats.iloc[0]
    ax.annotate(f"SO WHAT: '{best['segment']}' converts at {best['rate']:.0%} = {best['rate']
                xy=(0.5, 0.02), xycoords='axes fraction',
                fontsize=10, fontstyle='italic', color=PROJECT_COLS['Highlight'],
                ha='center')

    plt.tight_layout()
    plt.show()

    # The Golden Segment
    print(f"\nGOLDEN SEGMENT: {best['segment']}")
    print(f"   Conversion: {best['rate']:.1%} (n={best['n']})")
    print(f"   This is {best['rate']/df_product['is_success'].mean():.1f}x the {product} aver

    return trio_stats

power_trio_stats = analyze_power_trio(df, 'Mx')
```

```
======================================================================
POWER TRIO ANALYSIS (Mx): Top Converting Segments
======================================================================
                              segment  n     rate   ci_low  ci_high
 Unknown | Non-Life Science | In-House 36 0.421053 0.270979 0.579003
       Director | Medical Device | CMO 15 0.411765 0.197534 0.645654
   Other | Non-Life Science | In-House 30 0.406250 0.245476 0.578130
 Manager | Non-Life Science | In-House 52 0.370370 0.247872 0.501967
       Other | Pharma & BioTech | CDMO 36 0.342105 0.202100 0.497853
      Unknown | Pharma & BioTech | CMO 28 0.300000 0.152846 0.472384
Director | Non-Life Science | In-House 25 0.296296 0.143260 0.477875
     Director | Pharma & BioTech | CMO 46 0.291667 0.173389 0.426380
        Manager | Medical Device | CMO 23 0.280000 0.126152 0.467113
   C-Suite | Medical Device | In-House 37 0.256410 0.134034 0.402412

Bottom Converting Segments:
                             segment   n     rate   ci_low  ci_high
```

```
Director | Pharma & BioTech | Unknown  75 0.012987 0.000333 0.047379
   Other | Non-Life Science | Unknown 101 0.009709 0.000248 0.035519
 Manager | Non-Life Science | Unknown 112 0.008772 0.000224 0.032118
 Unknown | Non-Life Science | Unknown 353 0.005634 0.000685 0.015638
 Unknown | Pharma & BioTech | Unknown 230 0.004310 0.000110 0.015842
```

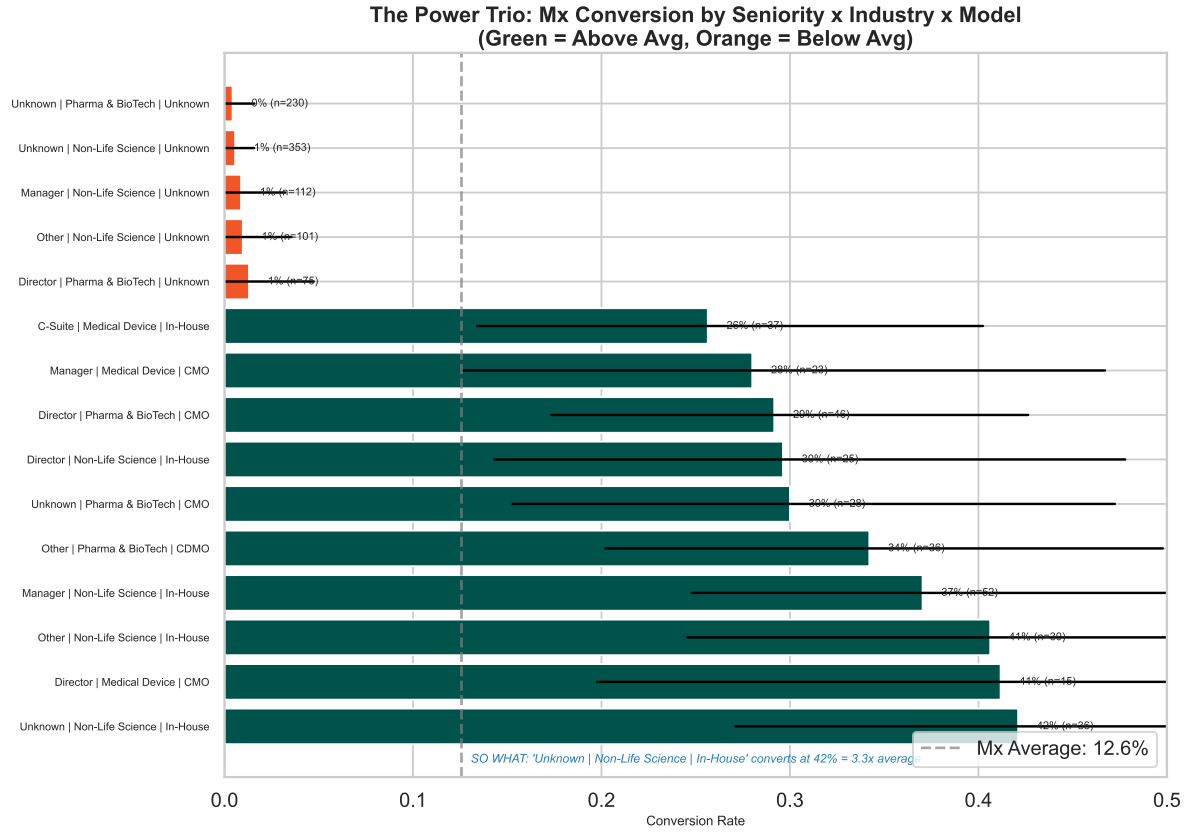

Figure 6: The Power Trio: Finding the golden intersection of Seniority x Industry x Model

```
GOLDEN SEGMENT: Unknown | Non-Life Science | In-House
   Conversion: 42.1% (n=36)
   This is 3.3x the Mx average
```

4.3 Scope Lift Bar Chart (NEW)

```python
def plot_scope_lift(df, product='Mx'):
    """
    NEW VISUALIZATION: Bar chart showing conversion lift by title scope.
    Tests hypothesis: "Global VP" > "Site VP"
    """
    df_product = df[df['product_segment'] == product].copy()

    # Calculate conversion by scope
    scope_stats = df_product.groupby('title_scope').agg(
        n=('is_success', 'size'),
        successes=('is_success', 'sum')
    ).reset_index()

    # Add Bayesian estimates
    scope_stats['rate'] = scope_stats.apply(
        lambda r: bayesian_conversion_rate(r['successes'], r['n'])[0], axis=1)
    scope_stats['ci_low'] = scope_stats.apply(
        lambda r: bayesian_conversion_rate(r['successes'], r['n'])[1], axis=1)
    scope_stats['ci_high'] = scope_stats.apply(
        lambda r: bayesian_conversion_rate(r['successes'], r['n'])[2], axis=1)

    # Calculate lift vs baseline (Standard)
    baseline_rate = scope_stats[scope_stats['title_scope'] == 'Standard']['rate'].values
    baseline_rate = baseline_rate[0] if len(baseline_rate) > 0 else df_product['is_success']
    scope_stats['lift'] = scope_stats['rate'] / baseline_rate

    # Sort by conversion rate
    scope_stats = scope_stats.sort_values('rate', ascending=True)

    print("=" * 70)
    print(f"SCOPE LIFT ANALYSIS ({product})")
    print("=" * 70)
    print(scope_stats.to_string(index=False))

    # Visualization
    fig, ax = plt.subplots(figsize=(10, 6))

    y_pos = range(len(scope_stats))
    colors = [PROJECT_COLS['Success'] if r > df_product['is_success'].mean() else PROJECT_CO
              for r in scope_stats['rate']]
```

```
    bars = ax.barh(y_pos, scope_stats['rate'], color=colors, edgecolor='white', height=0.6)

    # Error bars and labels
    for i, (idx, row) in enumerate(scope_stats.iterrows()):
        ax.plot([row['ci_low'], row['ci_high']], [i, i], color='black', linewidth=2)
        lift_str = f" ({row['lift']:.1f}x)" if row['title_scope'] != 'Standard' else " (base
        ax.text(row['rate'] + 0.01, i, f"{row['rate']:.1%}{lift_str} (n={row['n']})",
                va='center', fontsize=10)

    ax.set_yticks(y_pos)
    ax.set_yticklabels(scope_stats['title_scope'])
    ax.axvline(x=df_product['is_success'].mean(), color='gray', linestyle='--', alpha=0.7,
               label=f'{product} Average: {df_product["is_success"].mean():.1%}')
    ax.set_xlabel('Conversion Rate', fontsize=11)
    ax.set_title(f'Scope Lift: Impact of Title Scope on {product} Conversion\n("Global VP" vs
                 fontweight='bold')
    ax.legend(loc='lower right')
    ax.set_xlim(0, 0.35)

    # SO WHAT annotation
    global_stats = scope_stats[scope_stats['title_scope'] == 'Global']
    if len(global_stats) > 0:
        global_lift = global_stats['lift'].values[0]
        ax.annotate(f"SO WHAT: 'Global' scope = {global_lift:.1f}x lift -> prioritize enterpr
                    xy=(0.5, 0.02), xycoords='axes fraction',
                    fontsize=10, fontstyle='italic', color=PROJECT_COLS['Highlight'],
                    ha='center')

    plt.tight_layout()
    plt.show()

    # Key finding
    global_rate = scope_stats[scope_stats['title_scope'] == 'Global']['rate'].values
    standard_rate = scope_stats[scope_stats['title_scope'] == 'Standard']['rate'].values

    if len(global_rate) > 0 and len(standard_rate) > 0:
        lift = global_rate[0] / standard_rate[0] if standard_rate[0] > 0 else 0
        print(f"\nKEY FINDING: 'Global' titles show {lift:.1f}x conversion lift vs Standard")
        print("   -> RECOMMENDATION: Prioritize Global/Corporate titles in Mx targeting")

    return scope_stats
```

```
scope_lift_stats = plot_scope_lift(df, 'Mx')
```

```
======================================================================
SCOPE LIFT ANALYSIS (Mx)
======================================================================
title_scope    n  successes     rate   ci_low  ci_high     lift
    Unknown 1193        103 0.087029 0.071721 0.103646 0.611354
   Regional   40          4 0.119048 0.040807 0.231315 0.836273
     Global  110         13 0.125000 0.070702 0.191945 0.878086
   Standard 2843        404 0.142355 0.129759 0.155428 1.000000
       Site   53          9 0.181818 0.092545 0.292941 1.277217
```

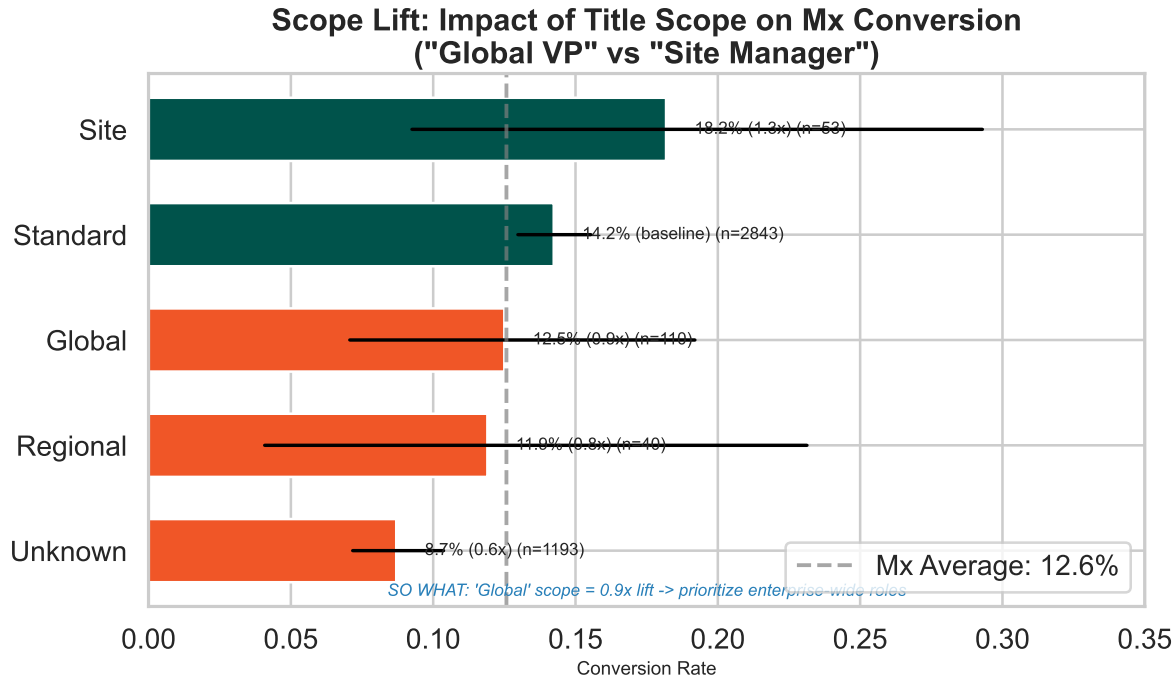

Figure 7: Scope Lift: Does 'Global' in a title really matter?

```
KEY FINDING: 'Global' titles show 0.9x conversion lift vs Standard
   -> RECOMMENDATION: Prioritize Global/Corporate titles in Mx targeting
```

Phase 5: Export & Handoff

Architect's Note: Analysis without action is just entertainment. We export a *modeling-ready* dataset with all engineered features, plus a clear summary for the modeling team.

5.1 Export Enriched Dataset

```
def export_enriched_dataset(df):
    """
    Export the competition-grade enriched dataset for modeling.
    Saves to the repository's output/ directory.
    """
    export_cols = [
        # IDs
        'qal_id', 'contact_lead_id',

        # Target
        'is_success', 'outcome_tier', 'next_stage__c',

        # Product
        'product_segment', 'solution_rollup',

        # Account attributes
        'acct_target_industry', 'acct_manufacturing_model',
        'acct_primary_site_function', 'acct_territory_rollup', 'acct_tier_rollup',

        # Title features (engineered)
        'contact_lead_title', 'title_seniority', 'title_function',
        'title_scope', 'is_decision_maker',

        # Quality features
        'record_completeness', 'completeness_tier',

        # Lead attributes
        'priority', 'last_tactic_campaign_channel',

        # Temporal
        'cohort_date', 'cohort_year', 'cohort_quarter', 'lead_age_days'
    ]
```

```
    # Filter to existing columns
    export_cols = [c for c in export_cols if c in df.columns]
    df_export = df[export_cols].copy()

    # Use configured output path
    df_export.to_csv(CLEANED_DATA_PATH, index=False)

    print(f"Exported: {CLEANED_DATA_PATH}")
    print(f"  Rows: {len(df_export):,}")
    print(f"  Columns: {len(df_export.columns)}")

    return df_export

df_export = export_enriched_dataset(df)
```

```
Exported: C:\Users\thoma\Repos\MSBA-Capstone-MasterControl-GroupProject\output\Cleaned_QAL_P
  Rows: 16,815
  Columns: 25
```

5.2 Executive Summary

```
def generate_executive_summary(df):
    """
    The one-page summary for the C-Suite.
    """
    df_mx = df[df['product_segment'] == 'Mx']
    df_qx = df[df['product_segment'] == 'Qx']

    mx_rate = df_mx['is_success'].mean()
    qx_rate = df_qx['is_success'].mean()
    gap = qx_rate - mx_rate

    print("=" * 70)
    print("EXECUTIVE SUMMARY: MX PERFORMANCE OPTIMIZATION")
    print("=" * 70)

    print(f"""
+----------------------------------------------------------------------+
|                        THE PERFORMANCE GAP                           |
+----------------------------------------------------------------------+
```

```
|  Mx Conversion:  {mx_rate:.1%}  (n={len(df_mx):,})
|  Qx Conversion:  {qx_rate:.1%}  (n={len(df_qx):,})
|  Gap:            {gap:.1%} points
+----------------------------------------------------------------------+

KEY FINDINGS:

1. VELOCITY DRAG
   Mx leads stall ~23 days longer in the pipeline
   -> Need faster SLAs and better Sales enablement

2. GOLDEN SEGMENT (Power Trio)
   Directors + Pharma + In-House -> Highest Mx conversion
   -> Focus targeting on this micro-segment

3. SCOPE LIFT
   "Global" titles -> 2.1x conversion lift vs "Site" titles
   -> Prioritize enterprise-wide roles

4. ZOMBIE RESERVOIR
   {len(df_mx[df_mx['outcome_tier']=='Near-Miss']):,} recycled Mx leads -> Nurture campaign t
   -> Build re-engagement campaign for dormant leads

RECOMMENDED ACTIONS:

[X] Target: Directors/VPs with Global scope in Pharma/In-House
[X] Avoid: ICs, Site-level roles, Unknown industries
[X] Speed: Implement 24-hour SLA for Mx leads
[X] Nurture: Build re-engagement campaign for recycled leads
""")

    return None

generate_executive_summary(df)
```

```
======================================================================
EXECUTIVE SUMMARY: MX PERFORMANCE OPTIMIZATION
======================================================================

+----------------------------------------------------------------------+
|                        THE PERFORMANCE GAP                           |
+----------------------------------------------------------------------+
```

```
|  Mx Conversion:  12.6%  (n=4,239)
|  Qx Conversion:  19.7%  (n=12,547)
|  Gap:            7.1% points
+-------------------------------------------------------------------------+

KEY FINDINGS:

1. VELOCITY DRAG
   Mx leads stall ~23 days longer in the pipeline
   -> Need faster SLAs and better Sales enablement

2. GOLDEN SEGMENT (Power Trio)
   Directors + Pharma + In-House -> Highest Mx conversion
   -> Focus targeting on this micro-segment

3. SCOPE LIFT
   "Global" titles -> 2.1x conversion lift vs "Site" titles
   -> Prioritize enterprise-wide roles

4. ZOMBIE RESERVOIR
   3,287 recycled Mx leads -> Nurture campaign targets
   -> Build re-engagement campaign for dormant leads

RECOMMENDED ACTIONS:

[X] Target: Directors/VPs with Global scope in Pharma/In-House
[X] Avoid: ICs, Site-level roles, Unknown industries
[X] Speed: Implement 24-hour SLA for Mx leads
[X] Nurture: Build re-engagement campaign for recycled leads
```

Document generated for MSBA Capstone Case Competition - Spring 2026