# StarCraft Anscombe Project — Full Python Code

## PLOT CODE FROM JUPYTER NOTEBOOK

```
from matplotlib.offsetbox import OffsetImage, AnnotationBbox
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

#Data setup
x123 = np.array([10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5])
y1 = np.array([8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82, 5.68])
y2 = np.array([9.14, 8.14, 8.74, 8.77, 9.26, 8.10, 6.13, 3.10, 9.13, 7.26, 4.74])
y3 = np.array([7.46, 6.77, 12.74, 7.11, 7.81, 8.84, 6.08, 5.39, 8.15, 6.42, 5.73])
x4 = np.array([8, 8, 8, 8, 8, 8, 8, 19, 8, 8, 8])
y4 = np.array([6.58, 5.76, 7.71, 8.84, 8.47, 7.04, 5.25, 12.50, 5.56, 7.91, 6.89])

datasets = [
    (x123, y1, 'Marine'),
    (x123, y2, 'Zergling'),
    (x123, y3, 'Zealot'),
    (x4, y4, 'Probe'),
]

#Load icons
marine_icon = Image.open('marine.png')
zergling_icon = Image.open('zergling.png')
zealot_icon = Image.open('zealot.png')
probe_icon = Image.open('probe.png')
icon_map = {
    'Marine': marine_icon,
    'Zergling': zergling_icon,
    'Zealot': zealot_icon,
    'Probe': probe_icon
}

def get_icon(image, zoom=0.1):
    return OffsetImage(image, zoom=zoom)

def fit_line(x):
    return 3 + 0.5 * x

#Scatter + fit icon plot
def plot_scatter_with_icons():
    fig, axes = plt.subplots(2, 2, figsize=(10, 10))
    axes = axes.flatten()
    for ax, (x, y, label) in zip(axes, datasets):
        x_line = np.array([min(x), max(x)])
        ax.plot(x_line, fit_line(x_line), 'r--', label='fit line')
        icon_image = icon_map[label]
        for xi, yi in zip(x, y):
            ab = AnnotationBbox(get_icon(icon_image, zoom=0.08), (xi, yi), frameon=False)
            ax.add_artist(ab)
        ax.set_title(label)
        ax.set_xlim(2, 20)
        ax.set_ylim(2, 14)
    plt.suptitle("Graph With Starcraft Units")
    plt.tight_layout(rect=[0, 0.03, 1, 0.95])
    fig.savefig("scatter_with_icons.png", dpi=250)
    plt.show()

#Residual plots using icons
def plot_residuals_with_icons():
    fig, axes = plt.subplots(2, 2, figsize=(10, 10))
    axes = axes.flatten()
    for ax, (x, y, label) in zip(axes, datasets):
        y_pred = fit_line(x)
        residuals = y - y_pred
        ax.axhline(0, color='gray', linestyle='--')
        icon_image = icon_map[label]
        for xi, ri in zip(x, residuals):
            ab = AnnotationBbox(get_icon(icon_image, zoom=0.08), (xi, ri), frameon=False)
            ax.add_artist(ab)
        ax.set_title(f"{label} — Residuals vs x (icons)")
        ax.set_xlabel("x")
        ax.set_ylabel("Residual (y − ■)")
        ax.set_xlim(2, 20)
        resid_lim = max(abs(residuals)) + 1
        ax.set_ylim(-resid_lim, resid_lim)
    plt.suptitle("Residual Plots with Icons", y=1.02)
    plt.tight_layout()
    fig.savefig("residual_with_icons.png", dpi=250)
    plt.show()

#Box + violin plots
def plot_box_violin():
    records = []
    for x, y, label in datasets:
        for xi, yi in zip(x, y):
            records.append({'dataset': label, 'x': xi, 'y': yi})
    df = pd.DataFrame(records)

    fig, axes = plt.subplots(1, 2, figsize=(12, 5))
    sns.boxplot(data=df, x='dataset', y='x', ax=axes[0])
    axes[0].set_title("Boxplot of x by Dataset")
    axes[0].set_ylabel("x values")

    sns.violinplot(data=df, x='dataset', y='y', ax=axes[1])
    axes[1].set_title("Violin plot of y by Dataset")
    axes[1].set_ylabel("y values")

    plt.tight_layout()
    fig.savefig("box_violin_with_icons1.png", dpi=250)
    plt.show()

    fig, axes = plt.subplots(1, 2, figsize=(12, 5))
    sns.violinplot(data=df, x='dataset', y='x', ax=axes[0], inner="box")
    axes[0].set_title("Violin + Box of x by dataset")
    sns.violinplot(data=df, x='dataset', y='y', ax=axes[1], inner="box")
    axes[1].set_title("Violin + Box of y by dataset")
    plt.tight_layout()
    fig.savefig("box_violin_with_icons2.png", dpi=250)
    plt.show()

#Overlaid comparison plot
def plot_overlaid_comparison():
    plt.figure(figsize=(8, 6))
    all_x = np.concatenate([x for x, y, label in datasets])
    xx_line = np.linspace(min(all_x) - 1, max(all_x) + 1, 200)
    yy_line = fit_line(xx_line)
    plt.plot(xx_line, yy_line, 'r--', label="Fit line (common)")
    for x, y, label in datasets:
        plt.scatter(x, y, label=label)
    plt.legend()
    plt.title("Overlaid Comparison of All Datasets")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.xlim(2, 20)
    plt.ylim(2, 14)
    plt.tight_layout()
    plt.savefig("overlaid.png", dpi=250)
    plt.show()

if __name__ == "__main__":
    plot_scatter_with_icons()
    plot_residuals_with_icons()
    plot_box_violin()
    plot_overlaid_comparison()
```

## CODE TO CALCULATE STATISTICS

```
import numpy as np

def summary_stats(x, y):
    n = len(x)
    mean_x = np.mean(x)
    mean_y = np.mean(y)
    var_x = np.var(x, ddof=1)
    var_y = np.var(y, ddof=1)
    std_x = np.std(x, ddof=1)
    std_y = np.std(y, ddof=1)
    cov_xy = np.sum((x - mean_x) * (y - mean_y)) / (n - 1)
    corr_xy = cov_xy / (std_x * std_y)
    slope = cov_xy / var_x
    intercept = mean_y - slope * mean_x
    y_pred = slope * x + intercept
    residuals = y - y_pred
    ss_total = np.sum((y - mean_y)**2)
    ss_res = np.sum(residuals**2)
    r_squared = 1 - (ss_res / ss_total)
    summary = {
        "Mean of x": mean_x,
        "Mean of y": mean_y,
        "Variance of x": var_x,
        "Variance of y": var_y,
        "Std dev of x": std_x,
        "Std dev of y": std_y,
        "Covariance (x,y)": cov_xy,
        "Correlation (x,y)": corr_xy,
        "Regression slope": slope,
        "Regression intercept": intercept,
        "R-squared": r_squared
    }
    return summary

x1 = np.array([10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5])
y1 = np.array([8.04, 6.95, 7.58, 8.81, 8.33, 9.96, 7.24, 4.26, 10.84, 4.82, 5.68])

x2 = np.array([10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5])
y2 = np.array([9.14, 8.14, 8.74, 8.77, 9.26, 8.10, 6.13, 3.10, 9.13, 7.26, 4.74])

x3 = np.array([10, 8, 13, 9, 11, 14, 6, 4, 12, 7, 5])
```

```
y3 = np.array([7.46, 6.77, 12.74, 7.11, 7.81, 8.84, 6.08, 5.39, 8.15, 6.42, 5.73])

x4 = np.array([8, 8, 8, 8, 8, 8, 8, 19, 8, 8, 8])
y4 = np.array([6.58, 5.76, 7.71, 8.84, 8.47, 7.04, 5.25, 12.50, 5.56, 7.91, 6.89])

datasets = [(x1, y1, "Dataset 1"), (x2, y2, "Dataset 2"),
            (x3, y3, "Dataset 3"), (x4, y4, "Dataset 4")]

for x, y, name in datasets:
    print(f"\n{name} Summary Statistics:")
    stats = summary_stats(x, y)
    for key, value in stats.items():
        print(f"{key}: {value:.4f}")
```