In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
from sklearn.metrics import r2_score

import statsmodels.api as sm
import statsmodels.formula.api as smf

import scipy.stats as stats

import warnings
warnings.filterwarnings("ignore")



drop_cols = [
    "TWD","COG","TWA", "TimeUTC", "SecondsSince1970", "ISODateTimeUTC",
    "Lat", "LatBow", "LatCenter", "LatStern", "Lon", "LonBow", "LonCenter", "LonStern",
    "Leg", "Log", "LogAlongCourse", "MagneticVariation", "Rank", "TimeLocal",
    "DistanceToLeader", "interval_id", "boat_name", "interval_duration",
    "Heel", "Heel_Lwd", "Line_R", "Line_L", "BelowLineCalc", "VMC","XTE","VMG","gain_forward","gain_lateral", "gain_vmg","Total_lines", "LoadCell_1","LoadCell_2","LoadCell_3","LoadCell_4","LoadCell_5","L
]

def show_correlation_matrix(dataframe, taille_figure=(6, 4), cmap="coolwarm"):
    plt.figure(figsize=taille_figure)
    sns.heatmap(dataframe.corr(), cmap=cmap, center=0)
    plt.title("Correlation matrix")
    plt.show()

def show_target_correlation(df, variable="SOG", taille_figure=(6, 4)):
    corr_with_vmg = df.corr()[variable].sort_values(ascending=False)
    #print(f"Correlation with {variable} :")
    # print(corr_with_vmg)

    plt.figure(figsize=taille_figure)
    sns.heatmap(corr_with_vmg.to_frame(), annot=True, cmap="coolwarm")
    plt.title(f"Correlation with {variable}")
    plt.show()

def compute_anova(df, target="SOG"):
    # Make a copy of the DataFrame to avoid modifying the original
    df_copy = df.copy()
    cols = df_copy.columns.drop(target)
    df_no_missing = df_copy.dropna(subset=cols)
    if df_no_missing.empty:
        print("⚠️ Pas assez de données non-nulles pour faire une ANOVA.")
        return pd.DataFrame()  # retourne un DataFrame vide ou None
    formula_no_missing = f"{target} ~ " + " + ".join(
        list(df_no_missing.columns.drop(target)))
    model_no_missing = smf.ols(formula=formula_no_missing, data=df_no_missing).fit()
    anova_no_missing = sm.stats.anova_lm(model_no_missing, typ=2)
    return anova_no_missing.sort_values("F", ascending=False)
"""
def linear_regression(df, target="SOG", degree=1, top_coefs=30):
    df_copy = df.copy()
    cols = df_copy.columns.drop(target)
```

```
    df_clean = df_copy.dropna(subset=cols)
    if df_clean[target].isna().any():
        df_clean = df_clean.dropna(subset=[target])

    X = df_clean[cols]
    y = df_clean[target]

    poly = PolynomialFeatures(degree=degree, include_bias=False)
    X_poly = poly.fit_transform(X)
    feature_names = poly.get_feature_names_out(X.columns)

    model = make_pipeline(
        StandardScaler(),
        LinearRegression()
    )

    model.fit(X_poly, y)
    y_pred = model.predict(X_poly)

    # Affichage de la performance
    print(f"R²: {r2_score(y, y_pred):.3f}")
    print(f"Used samples: {len(X)}")

    # Coefficients
    coefs = model.named_steps['linearregression'].coef_
    intercept = model.named_steps['linearregression'].intercept_

    coef_df = pd.DataFrame({
        'feature': feature_names,
        'coefficient': coefs
    }).sort_values('coefficient', key=abs, ascending=False)

    # Formule affichée
    print("\n Formula :")
    terms = [f"{coef:.3f} * {feat}" for feat, coef in zip(feature_names, coefs)]
    equation = " +\n  ".join(terms)
    print(f"{target} ≈ {intercept:.3f} +\n  {equation}")

    return coef_df.head(top_coefs)"""
from sklearn.linear_model import RidgeCV

def linear_regression(df, target="SOG", top_coefs=30):
    df_copy = df.copy()
    cols = df_copy.columns.drop(target)

    # Drop rows with missing values
    df_clean = df_copy.dropna(subset=cols)
    if df_clean[target].isna().any():
        df_clean = df_clean.dropna(subset=[target])

    X = df_clean[cols]
    y = df_clean[target]

    # Ridge regression with standard scaling
    model = make_pipeline(
        StandardScaler(),
        RidgeCV(alphas=[0.1, 1.0, 10.0])
    )

    model.fit(X, y)
    y_pred = model.predict(X)
```

```
    # Performance
    print(f"R²: {r2_score(y, y_pred):.3f}")
    print(f"Used samples: {len(X)}")

    # Extract coefficients
    coefs = model.named_steps['ridgecv'].coef_
    intercept = model.named_steps['ridgecv'].intercept_

    coef_df = pd.DataFrame({
        'feature': X.columns,
        'coefficient': coefs
    }).sort_values('coefficient', key=abs, ascending=False)

    # Display formula
    print("\nFormula:")
    terms = [f"{coef:.3f} * {feat}" for feat, coef in zip(X.columns, coefs)]
    equation = " +\n  ".join(terms)
    print(f"{target} ≈ {intercept:.3f} +\n  {equation}")

    return coef_df.head(top_coefs)




def t_test(df1, df2, target="SOG"):
    t_stat, p_value = stats.ttest_ind(df1[target].dropna(), df2[target].dropna())
    print(f"T-statistic: {t_stat:.3f}, p-value: {p_value:.15f}")

    # If p-value is less than 0.05, the difference is statistically significant
    if p_value < 0.05:
        print("The difference is statistically significant, keeping data split.")
    else:
        print("The difference is not statistically significant, keeping data combined.")

"""def full_analysis(df_numeric, target_variable="SOG"):
    # Display the correlation matrix
    # print("Correlation matrix:")
    #show_correlation_matrix(df_numeric)

    # Display correlation with the target variable
    print(f"\nCorrelation with {target_variable}:")
    show_target_correlation(df_numeric, variable=target_variable)

    # Compute and display ANOVA results
    print("\nANOVA:")
    anova_results = compute_anova(df_numeric)
    display(anova_results)

    # Apply and display polynomial regression results
    print("\nPolynomial fit:")
    regression_results = linear_regression(df_numeric)
    display(regression_results)"""

def full_analysis(df_numeric, target_variable="SOG"):
    # Drop 'Line_Side' column only if it's fully NaN
    if "Side_lines" in df_numeric.columns and df_numeric["Side_lines"].isna().all():
        df_numeric = df_numeric.drop(columns=["Side_lines"])
        print("Dropped column 'Side_lines' (all values were NaN).")

    # Display correlation with the target variable
    print(f"\nCorrelation with {target_variable}:")
    show_target_correlation(df_numeric, variable=target_variable)
```

```
    # Compute and display ANOVA results
    print("\nANOVA:")
    anova_results = compute_anova(df_numeric)
    display(anova_results)

    # Apply and display polynomial regression results
    print("\nPolynomial fit:")
    regression_results = linear_regression(df_numeric)
    display(regression_results)
```

In [2]:
```
df = pd.read_csv("all_data_enriched.csv")
df = df[df["boat_name"] == "SenseBoard"].copy()
load_cell_cols = ["LoadCell_1", "LoadCell_2", "LoadCell_3", "LoadCell_4", "LoadCell_5", "LoadCell_6"]
df = df.dropna(subset=load_cell_cols, how='all')

df.sample(10)
```

Out[2]:

	ISODateTimeUTC	SecondsSince1970	Heel_Abs	Heel_Lwd	Lat	LatBow	LatCenter	LatStern	Leg	Line_C	...	M_tot_X	M_tot_Y	M_front_X	M_front_Y	M_back_X	M_back_Y	P_fron
19952	2025-06-06 15:06:59.563000+00:00	1.749222e+09	50.8	50.8	43.523438	43.523437	43.523440	43.523443	1.0	116.027	...	3072.882103	6689.237125	2067.349236	-1051.347931	1005.532867	-271.710200	-97.702
39984	2025-06-07 13:27:23.260000+00:00	1.749303e+09	52.3	52.3	43.516868	43.516867	43.516872	43.516877	NaN	115.400	...	7733.804311	-1352.356905	5702.534764	1217.768700	2031.269547	-590.949603	42.031
35400	2025-06-07 13:09:13.860000+00:00	1.749302e+09	60.5	60.5	43.518167	43.518165	43.518170	43.518175	1.0	122.500	...	6498.074342	-789.342905	5014.465848	634.822666	1483.608494	61.908649	22.722
11769	2025-06-06 14:38:57.358000+00:00	1.749221e+09	52.2	52.2	43.526118	43.526117	43.526121	43.526126	1.0	126.900	...	5074.227247	4887.863065	4051.325909	474.952787	1022.901337	-504.161966	26.042
26653	2025-06-07 12:15:37.257000+00:00	1.749299e+09	46.6	46.6	43.515595	43.515594	43.515598	43.515602	1.0	109.700	...	5595.389398	594.134377	4633.371303	469.870193	962.018096	-230.083356	21.582
37343	2025-06-07 13:17:12.356000+00:00	1.749302e+09	54.0	54.0	43.519358	43.519357	43.519362	43.519367	1.0	111.300	...	7923.453902	-713.507789	5923.210259	1332.161525	2000.243643	-481.173291	43.202
31493	2025-06-07 12:47:10.752000+00:00	1.749300e+09	54.4	54.4	43.518010	43.518009	43.518013	43.518018	1.0	6.300	...	6680.752428	-696.385636	4945.444700	479.139960	1735.307728	-399.459674	21.717
29475	2025-06-07 12:36:32.755000+00:00	1.749300e+09	52.6	52.6	43.517475	43.517474	43.517478	43.517481	1.0	8.205	...	8445.570843	-3600.258743	6547.716327	1562.309828	1897.854516	-237.514808	46.614
39433	2025-06-07 13:26:55.755000+00:00	1.749303e+09	50.5	50.5	43.519212	43.519210	43.519215	43.519220	NaN	116.300	...	8377.281316	-1707.128005	6378.075401	1531.528028	1999.205915	-605.137199	45.812
36404	2025-06-07 13:11:57.756000+00:00	1.749302e+09	45.5	45.5	43.513805	43.513807	43.513801	43.513795	1.0	66.700	...	-4295.251683	1857.541306	-2441.824786	213.043494	-1853.426897	-795.559238	7.780

10 rows × 67 columns

I. All together

In [3]:
```
# Select numeric columns
df_numeric = df.select_dtypes(include=["float64", "int64"]).copy()

# Drop specified columns
df_numeric.drop(columns=[c for c in drop_cols if c in df_numeric.columns], inplace=True)

# Drop rows where 'SOG' is missing
df_numeric.dropna(subset=["SOG"], inplace=True)
```

```
# Print summary
print(f"Variables utilisées: {df_numeric.columns.tolist()}")
print(f"Number of rows after filtering: {len(df_numeric)}")
```

```
Variables utilisées: ['Heel_Abs', 'Line_C', 'ROT', 'Side_lines', 'Trim', 'TWA_Abs', 'TWS', 'SOG', 'boat_weight', 'F_front', 'F_back', 'M_tot_X', 'M_tot_Y', 'M_front_X', 'M_front_Y', 'M_back_X', 'M_back_
Y', 'P_front_X', 'P_front_Y', 'P_back_X', 'P_back_Y']
Number of rows after filtering: 9147
```

In [4]:
```
df_numeric.sample(20)
```

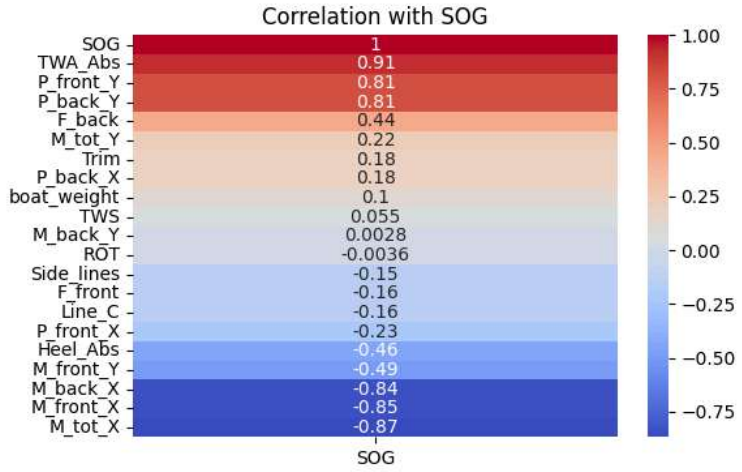
Out[4]:

	Heel_Abs	Line_C	ROT	Side_lines	Trim	TWA_Abs	TWS	SOG	boat_weight	F_front	...	M_tot_X	M_tot_Y	M_front_X	M_front_Y	M_back_X	M_back_Y	P_front_X	P_front_Y	P_b
34994	39.4	77.800	0.000	9.445	7.8	137.397	8.834	25.6	109.09	24073.076848	...	-823.769794	78.306997	-859.186851	-170.102366	35.417057	-205.637100	-7.066083	35.690778	-7.4
30453	38.7	4.000	-56.383	NaN	5.0	131.075	7.700	26.7	102.89	23675.157426	...	322.693555	-1245.933915	-546.850738	-117.453869	869.544292	98.455688	-4.961060	23.098082	4.4
12289	57.4	134.700	11.340	16.300	11.3	50.143	8.229	21.6	109.09	20909.394491	...	5854.458921	3492.431674	4418.803036	555.222166	1435.655885	-419.157991	26.553718	-211.330990	-13.2
19607	48.7	110.236	-4.211	9.800	6.5	50.978	11.626	23.3	109.09	14216.821210	...	5710.510820	5079.449460	4227.289876	23.454419	1483.220944	-628.389526	1.649765	-297.344238	-19.8
28275	57.6	8.400	-25.263	NaN	5.5	46.249	8.875	22.2	102.89	34996.022331	...	8709.365409	-2729.577896	7105.895913	1809.933362	1603.469496	-402.813454	51.718259	-203.048674	-15.8
27298	45.6	93.200	-27.523	NaN	5.7	151.405	8.768	25.5	102.89	22384.064899	...	-1877.477035	273.997164	-1812.985501	165.182395	-64.491534	107.265521	7.379464	80.994471	4.5
27823	53.7	6.400	-7.547	NaN	3.7	48.716	8.332	22.7	102.89	37091.092442	...	9631.430848	-2490.924766	7965.574023	2120.677415	1665.856825	-232.458624	57.174844	-214.757062	-8.8
31931	63.0	5.843	0.943	NaN	4.0	48.000	8.900	23.5	102.89	30028.947046	...	7520.418966	-1477.867042	5601.556873	1131.331691	1918.862093	-444.770885	37.674704	-186.538571	-16.9
18582	49.9	75.400	24.242	12.264	12.8	148.826	8.848	26.4	109.09	14112.762326	...	-5398.313963	6357.870438	-4097.005964	-863.935297	-1301.307999	-692.048159	-61.216598	290.305035	-17.9
28811	47.1	5.000	25.926	NaN	6.1	36.200	8.856	21.1	102.89	23565.780003	...	6776.152034	-1160.173670	5345.278570	718.529250	1430.873464	-404.251124	30.490366	-226.823749	-18.8
17999	56.3	100.400	3.061	10.142	9.3	42.628	8.536	21.6	109.09	15726.498572	...	6896.395658	3195.905879	4973.278748	441.763105	1923.116909	-440.186601	28.090366	-316.235602	-17.0
39911	57.9	120.600	3.922	8.700	1.9	46.497	9.337	23.7	109.09	26693.837709	...	7229.137831	-898.714893	5350.022257	1044.023657	1879.115575	-518.009245	39.111036	-200.421622	-20.8
34452	54.9	64.700	8.163	8.000	10.7	143.278	9.800	26.8	109.09	25987.166985	...	-2770.507977	2620.925022	-1777.806908	19.754900	-992.701070	-201.591954	0.760179	68.410955	-5.6
33115	54.0	108.800	24.444	13.500	3.9	31.945	8.445	22.0	109.09	22020.859331	...	6993.473475	-753.199404	5634.711097	826.440582	1358.762378	-304.912836	37.529897	-255.880618	-15.2
34948	28.0	62.615	4.124	8.485	6.8	138.171	9.029	25.3	109.09	26246.102299	...	-1083.692309	377.802520	-645.852402	-47.403904	-437.839907	-303.574158	-1.806131	24.607555	-9.8
36290	51.6	107.000	16.667	15.750	11.8	39.229	10.136	21.8	109.09	31942.181254	...	8425.587896	-3184.660611	6475.522578	1365.252235	1950.065318	-84.858578	42.741359	-202.726374	-4.0
11680	49.8	108.756	31.683	7.200	3.1	48.686	7.814	21.9	109.09	18802.618187	...	5331.427245	4086.335485	3975.814656	436.310988	1355.612590	-523.740209	23.204800	-211.450055	-16.4
20560	39.0	98.193	-13.265	15.769	1.2	151.700	9.491	28.9	109.09	11346.210547	...	-2474.186655	6041.907925	-2393.003197	-1227.173015	-81.183458	-533.034036	-108.157081	210.907702	-15.0
30821	53.8	4.900	29.114	NaN	14.1	149.300	6.747	25.5	102.89	22749.363807	...	-1798.312699	1309.206909	-1357.035363	-12.544580	-441.277336	205.841180	-0.551426	59.651574	7.6
35110	49.6	107.200	-30.208	7.700	9.8	40.971	8.900	22.3	109.09	18361.204926	...	1799.579240	-5658.800540	69.126552	-680.093335	1730.452688	-397.007263	-37.039690	-3.764816	-42.6

20 rows × 21 columns

In [5]:
```
full_analysis(df_numeric)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	3424.778775	1.0	4772.253669	0.000000e+00
TWA_Abs	517.306943	1.0	720.840708	1.839368e-150
TWS	478.615532	1.0	666.926210	6.753870e-140
M_tot_X	165.563582	1.0	230.704364	3.324978e-51
Trim	138.717969	1.0	193.296379	2.619091e-43
M_front_Y	94.342767	1.0	131.461810	3.889435e-30
M_back_Y	90.941252	1.0	126.721973	4.038607e-29
F_back	85.026643	1.0	118.480269	2.376568e-27
M_tot_Y	76.562295	1.0	106.685634	8.209234e-25
P_front_X	72.828826	1.0	101.483236	1.086863e-23
F_front	53.004331	1.0	73.858818	1.045350e-17
P_back_X	48.914515	1.0	68.159869	1.818916e-16
P_back_Y	46.027897	1.0	64.137515	1.370604e-15
Line_C	33.573402	1.0	46.782815	8.675374e-12
P_front_Y	24.490712	1.0	34.126552	5.421038e-09
M_back_X	14.204457	1.0	19.793182	8.776871e-06
M_front_X	9.003906	1.0	12.546481	3.998148e-04
ROT	1.135283	1.0	1.581959	2.085252e-01
Side_lines	0.555401	1.0	0.773922	3.790398e-01
Heel_Abs	0.265792	1.0	0.370367	5.428260e-01
Residual	4537.662432	6323.0	NaN	NaN

```
Polynomial fit:
R²: 0.904
Used samples: 6341

Formula:
SOG ≈ 24.201 +
  0.007 * Heel_Abs +
  0.123 * Line_C +
  0.013 * ROT +
  -0.010 * Side_lines +
  -0.162 * Trim +
  1.462 * TWA_Abs +
  0.294 * TWS +
  0.000 * boat_weight +
  0.266 * F_front +
  -0.011 * F_back +
  -0.619 * M_tot_X +
  -0.235 * M_tot_Y +
  -0.700 * M_front_X +
  0.275 * M_front_Y +
  -0.308 * M_back_X +
  -0.532 * M_back_Y +
  -0.218 * P_front_X +
  0.213 * P_front_Y +
  0.392 * P_back_X +
  -0.421 * P_back_Y
```

	feature	coefficient
5	TWA_Abs	1.462441
12	M_front_X	-0.700236
10	M_tot_X	-0.619303
15	M_back_Y	-0.531660
19	P_back_Y	-0.421198
18	P_back_X	0.392196
14	M_back_X	-0.308126
6	TWS	0.294440
13	M_front_Y	0.274662
8	F_front	0.265902
11	M_tot_Y	-0.234676
16	P_front_X	-0.218180
17	P_front_Y	0.213234
4	Trim	-0.162133
1	Line_C	0.122922
2	ROT	0.013023
9	F_back	-0.010809
3	Side_lines	-0.010187
0	Heel_Abs	0.006715
7	boat_weight	0.000000

II Upwind:

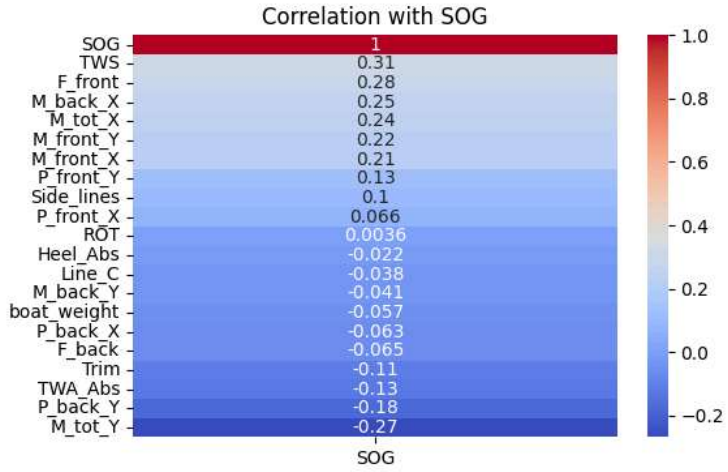
II.1. All upwind data

In [6]:
```
upwind_data = df[df['TWA'] >= 0]
df_numeric_upwind = upwind_data.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_upwind.drop(columns=[c for c in drop_cols if c in df_numeric_upwind.columns], inplace=True)
df_numeric_upwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_upwind)}")
```

```
Number of rows after filtering: 5895
```

In [7]:
```
full_analysis(df_numeric_upwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	1710.353668	1.0	3520.799492	0.000000e+00
TWS	205.591630	1.0	423.214753	1.744502e-89
M_back_X	56.199649	1.0	115.688175	1.278987e-26
M_front_X	43.804875	1.0	90.173270	3.634940e-21
Trim	34.073759	1.0	70.141559	7.538830e-17
Line_C	33.357976	1.0	68.668105	1.571179e-16
TWA_Abs	30.342300	1.0	62.460271	3.485133e-15
Side_lines	29.535584	1.0	60.799629	7.997483e-15
M_tot_X	29.384171	1.0	60.487943	9.347431e-15
F_front	25.199818	1.0	51.874363	7.031897e-13
M_front_Y	22.213919	1.0	45.727825	1.555128e-11
M_back_Y	21.833952	1.0	44.945655	2.308176e-11
F_back	16.615231	1.0	34.202808	5.361568e-09
M_tot_Y	14.276722	1.0	29.388936	6.269892e-08
P_front_X	12.173286	1.0	25.058969	5.799116e-07
P_back_X	10.454245	1.0	21.520286	3.612707e-06
P_back_Y	9.935318	1.0	20.452064	6.291009e-06
Heel_Abs	9.640824	1.0	19.845842	8.623319e-06
P_front_Y	6.094720	1.0	12.546111	4.015495e-04
ROT	0.299965	1.0	0.617485	4.320299e-01
Residual	1948.486012	4011.0	NaN	NaN

```
Polynomial fit:
R²: 0.355
Used samples: 4029

Formula:
SOG ≈ 22.264 +
  -0.054 * Heel_Abs +
  0.105 * Line_C +
  -0.009 * ROT +
  -0.112 * Side_lines +
  -0.096 * Trim +
  -0.105 * TWA_Abs +
  0.262 * TWS +
  0.000 * boat_weight +
  0.258 * F_front +
  0.092 * F_back +
  -0.035 * M_tot_X +
  -0.221 * M_tot_Y +
  -0.126 * M_front_X +
  -0.120 * M_front_Y +
  0.337 * M_back_X +
  0.341 * M_back_Y +
  0.185 * P_front_X +
  -0.138 * P_front_Y +
  -0.242 * P_back_X +
  0.191 * P_back_Y
```

	feature	coefficient
15	M_back_Y	0.341186
14	M_back_X	0.337134
6	TWS	0.262421
8	F_front	0.258034
18	P_back_X	-0.241688
11	M_tot_Y	-0.221232
19	P_back_Y	0.191014
16	P_front_X	0.184890
17	P_front_Y	-0.137992
12	M_front_X	-0.125772
13	M_front_Y	-0.119916
3	Side_lines	-0.112093
5	TWA_Abs	-0.105379
1	Line_C	0.105132
4	Trim	-0.095813
9	F_back	0.091609
0	Heel_Abs	-0.054100
10	M_tot_X	-0.035161
2	ROT	-0.009313
7	boat_weight	0.000000

II.2. Upwind: Gian vs Karl

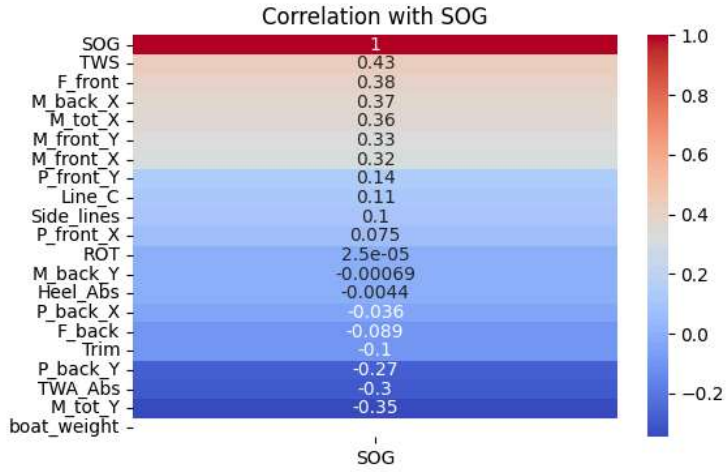
II.2.1. Upwind: Gian

In [8]:
```
gian_data_upwind = upwind_data[
    (upwind_data['boat_name'] == "Gian Stragiotti") |
    ((upwind_data['boat_name'] == "SenseBoard") & (upwind_data['opponent_name'] == "Karl Maeder"))
]
df_numeric_gian_upwind = gian_data_upwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_gian_upwind.drop(columns=[c for c in drop_cols if c in df_numeric_gian_upwind.columns], inplace=True)
df_numeric_gian_upwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_gian_upwind)}")
```

```
Number of rows after filtering: 4045
```

In [9]:
```
full_analysis(df_numeric_gian_upwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	1710.353668	1.0	3520.799492	0.000000e+00
TWS	205.591630	1.0	423.214753	1.744502e-89
M_back_X	56.199649	1.0	115.688175	1.278987e-26
M_front_X	43.804875	1.0	90.173270	3.634940e-21
Trim	34.073759	1.0	70.141559	7.538830e-17
Line_C	33.357976	1.0	68.668105	1.571179e-16
TWA_Abs	30.342300	1.0	62.460271	3.485133e-15
Side_lines	29.535584	1.0	60.799629	7.997483e-15
M_tot_X	29.384171	1.0	60.487943	9.347431e-15
F_front	25.199818	1.0	51.874363	7.031897e-13
M_front_Y	22.213919	1.0	45.727825	1.555128e-11
M_back_Y	21.833952	1.0	44.945655	2.308176e-11
F_back	16.615231	1.0	34.202808	5.361568e-09
M_tot_Y	14.276722	1.0	29.388936	6.269892e-08
P_front_X	12.173286	1.0	25.058969	5.799116e-07
P_back_X	10.454245	1.0	21.520286	3.612707e-06
P_back_Y	9.935318	1.0	20.452064	6.291009e-06
Heel_Abs	9.640824	1.0	19.845842	8.623319e-06
P_front_Y	6.094720	1.0	12.546111	4.015495e-04
ROT	0.299965	1.0	0.617485	4.320299e-01
Residual	1948.486012	4011.0	NaN	NaN

```
Polynomial fit:
R²: 0.355
Used samples: 4029

Formula:
SOG ≈ 22.264 +
  -0.054 * Heel_Abs +
  0.105 * Line_C +
  -0.009 * ROT +
  -0.112 * Side_lines +
  -0.096 * Trim +
  -0.105 * TWA_Abs +
  0.262 * TWS +
  0.000 * boat_weight +
  0.258 * F_front +
  0.092 * F_back +
  -0.035 * M_tot_X +
  -0.221 * M_tot_Y +
  -0.126 * M_front_X +
  -0.120 * M_front_Y +
  0.337 * M_back_X +
  0.341 * M_back_Y +
  0.185 * P_front_X +
  -0.138 * P_front_Y +
  -0.242 * P_back_X +
  0.191 * P_back_Y
```

	feature	coefficient
15	M_back_Y	0.341186
14	M_back_X	0.337134
6	TWS	0.262421
8	F_front	0.258034
18	P_back_X	-0.241688
11	M_tot_Y	-0.221232
19	P_back_Y	0.191014
16	P_front_X	0.184890
17	P_front_Y	-0.137992
12	M_front_X	-0.125772
13	M_front_Y	-0.119916
3	Side_lines	-0.112093
5	TWA_Abs	-0.105379
1	Line_C	0.105132
4	Trim	-0.095813
9	F_back	0.091609
0	Heel_Abs	-0.054100
10	M_tot_X	-0.035161
2	ROT	-0.009313
7	boat_weight	0.000000

II.2.2. Upwind: Karl

In [10]:
```
karl_data_upwind = upwind_data[((upwind_data['boat_name'] == "SenseBoard") & (upwind_data['opponent_name'] == "Gian Stragiotti"))
]
df_numeric_karl_upwind = karl_data_upwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_karl_upwind.drop(columns=[c for c in drop_cols if c in df_numeric_karl_upwind.columns], inplace=True)
df_numeric_karl_upwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_karl_upwind)}")
print(df_numeric_karl_upwind.shape)
print(df_numeric_karl_upwind.columns)
```

```
Number of rows after filtering: 1850
(1850, 21)
Index(['Heel_Abs', 'Line_C', 'ROT', 'Side_lines', 'Trim', 'TWA_Abs', 'TWS',
       'SOG', 'boat_weight', 'F_front', 'F_back', 'M_tot_X', 'M_tot_Y',
       'M_front_X', 'M_front_Y', 'M_back_X', 'M_back_Y', 'P_front_X',
       'P_front_Y', 'P_back_X', 'P_back_Y'],
      dtype='object')
```

In [11]:
```
df_numeric_karl_upwind.head(10)
```

Out[11]:

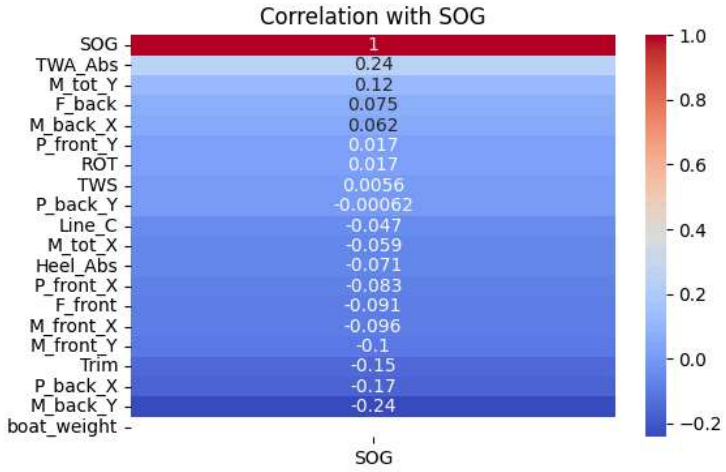
	Heel_Abs	Line_C	ROT	Side_lines	Trim	TWA_Abs	TWS	SOG	boat_weight	F_front	...	M_tot_X	M_tot_Y	M_front_X	M_front_Y	M_back_X	M_back_Y	P_front_X	P_front_Y	P_back_X
25553	59.5	93.3	11.000	NaN	7.5	41.174	9.352	21.4	102.89	27567.598622	...	6530.224065	-713.048129	5638.902083	1219.704616	891.321982	-322.846790	44.244137	-204.548178	-13.091291
25556	57.4	88.1	-14.141	NaN	12.3	42.564	9.372	21.4	102.89	30282.257552	...	7305.571614	-1763.759238	6091.881706	1425.550444	1213.689908	-429.662418	47.075435	-201.169998	-17.514047
25557	59.8	98.4	1.020	NaN	8.7	42.454	9.391	21.3	102.89	30629.872010	...	6811.715689	-1398.451796	5764.919843	1233.056963	1046.795846	-415.817770	40.256680	-188.212339	-15.668410
25560	57.8	108.1	15.000	NaN	1.5	40.944	9.411	21.4	102.89	29534.445709	...	6924.413623	-1221.335306	5731.189744	1214.808948	1193.223878	-368.446330	41.131937	-194.051035	-14.352249
25562	53.0	111.4	-7.619	NaN	4.6	41.734	9.432	21.4	102.89	31969.060202	...	7732.919196	-1202.778863	6309.024879	1556.310267	1423.894317	-377.563733	48.681765	-197.347837	-13.942098
25563	54.8	111.3	-10.309	NaN	7.0	42.724	9.452	21.4	102.89	32674.719191	...	8570.026241	-1558.310970	6720.991498	1772.416496	1849.034744	-457.739613	54.244276	-205.693933	-17.299894
25565	54.6	110.7	11.224	NaN	7.0	41.614	9.471	21.3	102.89	34743.378192	...	8671.218286	-1997.018077	7073.531115	1910.525524	1597.687170	-431.450793	54.989630	-203.593648	-16.053770
25571	51.7	104.6	1.000	NaN	8.5	42.784	9.516	21.4	102.89	30655.266332	...	7395.790468	-1079.527292	6394.101992	1592.460392	1001.688476	-461.493984	51.947368	-208.580866	-17.578087
25573	60.9	110.8	21.782	NaN	10.3	40.574	9.526	21.3	102.89	35339.840155	...	8713.352894	-2557.710299	7160.962384	1994.616379	1552.390511	-700.983149	56.441013	-202.631431	-25.963291
25575	65.1	123.4	-12.000	NaN	7.7	41.764	9.536	21.2	102.89	35994.499228	...	8971.350435	-2722.784403	7263.796996	2023.220608	1707.553439	-701.947032	56.209161	-201.802974	-25.841774

10 rows × 21 columns

In [12]:
```
full_analysis(df_numeric_karl_upwind)
```

```
Dropped column 'Side_lines' (all values were NaN).

Correlation with SOG:
```




```
ANOVA:
```

	sum_sq	df	F	PR(>F)
boat_weight	351.499504	1.0	840.239706	2.134337e-152
TWA_Abs	33.632840	1.0	80.397404	7.367066e-19
Trim	23.360502	1.0	55.841959	1.209428e-13
M_back_X	11.266167	1.0	26.931136	2.342117e-07
M_front_X	10.700326	1.0	25.578526	4.671802e-07
Line_C	9.054082	1.0	21.643271	3.518732e-06
Heel_Abs	7.519618	1.0	17.975223	2.349368e-05
TWS	4.767615	1.0	11.396714	7.511020e-04
M_tot_X	3.511536	1.0	8.394129	3.809026e-03
P_back_Y	2.835445	1.0	6.777972	9.303451e-03
P_front_X	2.392165	1.0	5.718335	1.688895e-02
F_back	2.077321	1.0	4.965719	2.597545e-02
M_tot_Y	1.594115	1.0	3.810643	5.107997e-02
P_front_Y	1.296915	1.0	3.100202	7.844934e-02
F_front	0.974339	1.0	2.329101	1.271477e-01
ROT	0.431169	1.0	1.030686	3.101315e-01
M_back_Y	0.277384	1.0	0.663071	4.155835e-01
P_back_X	0.158945	1.0	0.379950	5.377065e-01
M_front_Y	0.006165	1.0	0.014738	9.033882e-01
Residual	766.803312	1833.0	NaN	NaN

```
Polynomial fit:
R²: 0.221
Used samples: 1850

Formula:
SOG ≈ 22.364 +
  -0.069 * Heel_Abs +
  -0.105 * Line_C +
  0.015 * ROT +
  -0.117 * Trim +
  0.146 * TWA_Abs +
  0.063 * TWS +
  0.000 * boat_weight +
  0.036 * F_front +
  0.154 * F_back +
  -0.058 * M_tot_X +
  -0.099 * M_tot_Y +
  -0.188 * M_front_X +
  -0.254 * M_front_Y +
  0.326 * M_back_X +
  -0.156 * M_back_Y +
  0.108 * P_front_X +
  -0.099 * P_front_Y +
  -0.023 * P_back_X +
  0.173 * P_back_Y
```

	feature	coefficient
13	M_back_X	0.325644
12	M_front_Y	-0.253670
11	M_front_X	-0.188006
18	P_back_Y	0.173064
14	M_back_Y	-0.155508
8	F_back	0.154243
4	TWA_Abs	0.146322
3	Trim	-0.117161
15	P_front_X	0.107892
1	Line_C	-0.104822
10	M_tot_Y	-0.099476
16	P_front_Y	-0.099081
0	Heel_Abs	-0.068812
5	TWS	0.063450
9	M_tot_X	-0.057618
7	F_front	0.036083
17	P_back_X	-0.023395
2	ROT	0.015199
6	boat_weight	0.000000

II.2.3. Upwind: Karl vs Gian t_test

In [13]:
```
t_test(df_numeric_gian_upwind,df_numeric_karl_upwind)
```

```
T-statistic: -4.351, p-value: 0.000013771261761
The difference is statistically significant, keeping data split.
```

II.3. Upwind: Master vs Slave

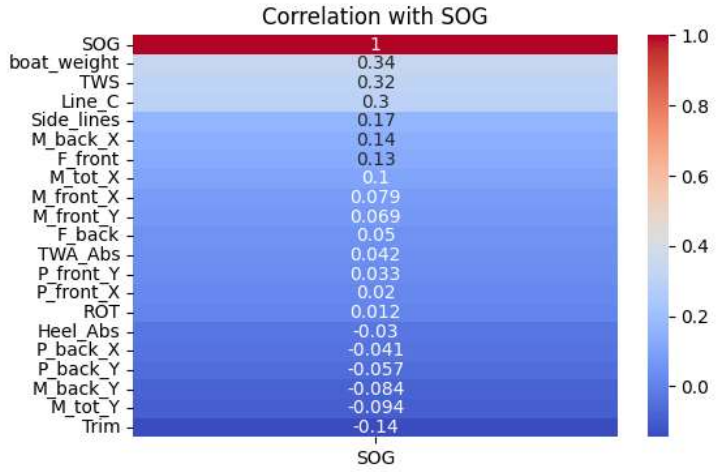
II.3.1. Master

In [14]:
```
master_data_upwind = upwind_data[upwind_data['boat_role'] == "master"]
df_numeric_master_upwind = master_data_upwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_master_upwind.drop(columns=[c for c in drop_cols if c in df_numeric_master_upwind.columns], inplace=True)
df_numeric_master_upwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_master_upwind)}")
```

```
Number of rows after filtering: 2867
```

In [15]:
```
full_analysis(df_numeric_master_upwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	827.717837	1.0	2469.483328	0.000000e+00
TWS	82.212130	1.0	245.278615	5.179438e-52
TWA_Abs	55.846578	1.0	166.617400	1.454441e-36
P_back_Y	12.605613	1.0	37.608651	1.054259e-09
P_back_X	8.733098	1.0	26.055062	3.658625e-07
M_front_Y	8.174837	1.0	24.389499	8.576741e-07
M_tot_X	7.994822	1.0	23.852428	1.129492e-06
Line_C	7.277872	1.0	21.713417	3.391434e-06
Trim	6.085619	1.0	18.156351	2.137081e-05
M_back_X	5.467605	1.0	16.312515	5.590028e-05
M_back_Y	5.274517	1.0	15.736439	7.558122e-05
P_front_X	5.068143	1.0	15.120727	1.044082e-04
F_front	2.389140	1.0	7.127961	7.655430e-03
ROT	2.256939	1.0	6.733542	9.536346e-03
Heel_Abs	1.845449	1.0	5.505870	1.905765e-02
F_back	1.745831	1.0	5.208659	2.258761e-02
M_front_X	0.823555	1.0	2.457064	1.171681e-01
Side_lines	0.493363	1.0	1.471940	2.251945e-01
M_tot_Y	0.427413	1.0	1.275180	2.589434e-01
P_front_Y	0.106127	1.0	0.316628	5.737092e-01
Residual	620.750671	1852.0	NaN	NaN

```
Polynomial fit:
R²: 0.410
Used samples: 1870

Formula:
SOG ≈ 22.733 +
  -0.035 * Heel_Abs +
  0.074 * Line_C +
  -0.036 * ROT +
  -0.018 * Side_lines +
  -0.060 * Trim +
  -0.244 * TWA_Abs +
  0.269 * TWS +
  0.000 * boat_weight +
  0.224 * F_front +
  -0.023 * F_back +
  0.100 * M_tot_X +
  -0.340 * M_tot_Y +
  0.087 * M_front_X +
  -0.356 * M_front_Y +
  0.119 * M_back_X +
  0.100 * M_back_Y +
  0.188 * P_front_X +
  -0.037 * P_front_Y +
  -0.310 * P_back_X +
  0.290 * P_back_Y
```

	feature	coefficient
13	M_front_Y	-0.356279
11	M_tot_Y	-0.339683
18	P_back_X	-0.309715
19	P_back_Y	0.290120
6	TWS	0.268510
5	TWA_Abs	-0.244044
8	F_front	0.223872
16	P_front_X	0.188385
14	M_back_X	0.119166
10	M_tot_X	0.100265
15	M_back_Y	0.100082
12	M_front_X	0.086786
1	Line_C	0.073681
4	Trim	-0.060080
17	P_front_Y	-0.037024
2	ROT	-0.036375
0	Heel_Abs	-0.034615
9	F_back	-0.022618
3	Side_lines	-0.017860
7	boat_weight	0.000000

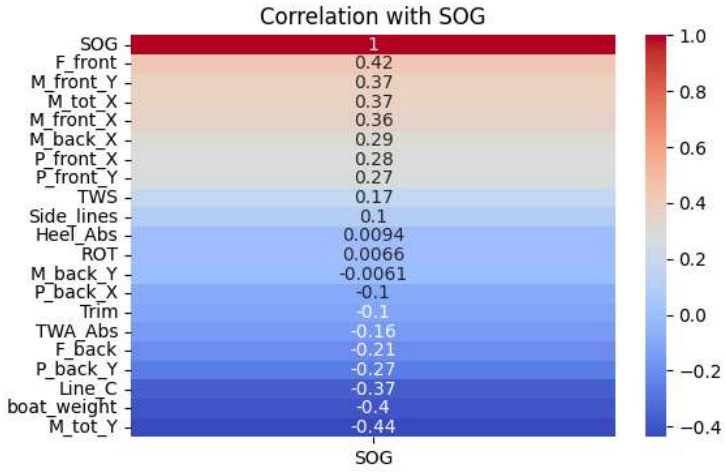
II.3.2 Slave

In [16]:
```
slave_data_upwind = upwind_data[upwind_data['boat_role'] == "slave"]
df_numeric_slave_upwind = slave_data_upwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_slave_upwind.drop(columns=[c for c in drop_cols if c in df_numeric_slave_upwind.columns], inplace=True)
df_numeric_slave_upwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_slave_upwind)}")
```

```
Number of rows after filtering: 3028
```

In [17]:
```
full_analysis(df_numeric_slave_upwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	769.811067	1.0	2069.984447	8.185373e-317
TWA_Abs	66.297860	1.0	178.271715	4.050752e-39
M_front_Y	28.090869	1.0	75.534977	7.002827e-18
M_back_Y	24.569579	1.0	66.066401	7.293222e-16
F_front	17.565194	1.0	47.231950	8.240824e-12
Side_lines	16.943416	1.0	45.560019	1.899676e-11
P_back_X	15.964026	1.0	42.926488	7.097502e-11
Line_C	12.523694	1.0	33.675603	7.480383e-09
M_tot_Y	12.389230	1.0	33.314036	8.982827e-09
F_back	11.915082	1.0	32.039076	1.713951e-08
Trim	10.364000	1.0	27.868292	1.429856e-07
M_tot_X	8.430399	1.0	22.668933	2.053312e-06
M_back_X	5.461816	1.0	14.686556	1.306076e-04
P_back_Y	4.975173	1.0	13.377997	2.607396e-04
Heel_Abs	4.646168	1.0	12.493319	4.170749e-04
M_front_X	2.137343	1.0	5.747211	1.660007e-02
ROT	1.706606	1.0	4.588982	3.229063e-02
P_front_Y	0.237626	1.0	0.638964	4.241748e-01
P_front_X	0.203965	1.0	0.548453	4.590316e-01
TWS	0.000173	1.0	0.000464	9.828139e-01
Residual	796.221198	2141.0	NaN	NaN

```
Polynomial fit:
R²: 0.334
Used samples: 2159

Formula:
SOG ≈ 21.857 +
  -0.051 * Heel_Abs +
  0.090 * Line_C +
  -0.029 * ROT +
  -0.119 * Side_lines +
  -0.072 * Trim +
  -0.230 * TWA_Abs +
  0.001 * TWS +
  0.000 * boat_weight +
  0.255 * F_front +
  0.166 * F_back +
  0.028 * M_tot_X +
  -0.157 * M_tot_Y +
  -0.016 * M_front_X +
  -0.164 * M_front_Y +
  0.198 * M_back_X +
  0.654 * M_back_Y +
  0.023 * P_front_X +
  -0.031 * P_front_Y +
  -0.538 * P_back_X +
  0.207 * P_back_Y
```

	feature	coefficient
15	M_back_Y	0.654210
18	P_back_X	-0.538467
8	F_front	0.255148
5	TWA_Abs	-0.229890
19	P_back_Y	0.207205
14	M_back_X	0.197538
9	F_back	0.166148
13	M_front_Y	-0.163692
11	M_tot_Y	-0.156954
3	Side_lines	-0.118993
1	Line_C	0.090029
4	Trim	-0.072473
0	Heel_Abs	-0.051217
17	P_front_Y	-0.030705
2	ROT	-0.029123
10	M_tot_X	0.027749
16	P_front_X	0.023339
12	M_front_X	-0.015555
6	TWS	0.001076
7	boat_weight	0.000000

III.3.3. Upwind: Master vs Slave t_test

In [18]:
```
t_test(df_numeric_master_upwind,df_numeric_slave_upwind)
```

```
T-statistic: 23.713, p-value: 0.000000000000000
The difference is statistically significant, keeping data split.
```

III Downwind

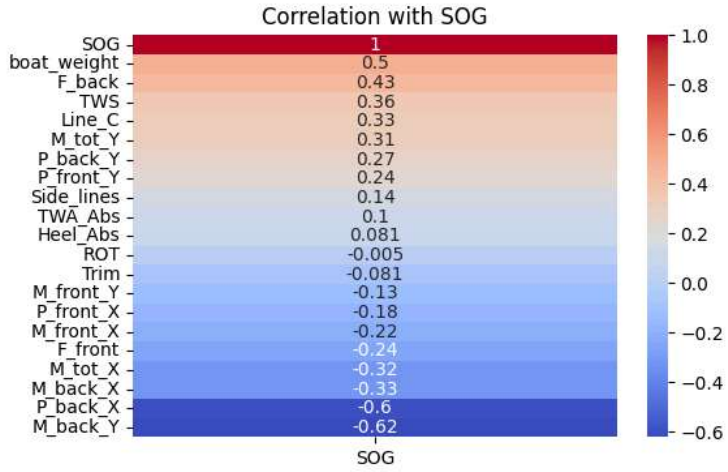
III.1. All downwind data

In [19]:
```
downwind_data = df[df['TWA'] < 0]
df_numeric_downwind = downwind_data.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_downwind.drop(columns=[c for c in drop_cols if c in df_numeric_downwind.columns], inplace=True)
df_numeric_downwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_downwind)}")
```

```
Number of rows after filtering: 3252
```

In [20]:
```
full_analysis(df_numeric_downwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	455.523144	1.0	637.346342	2.672430e-124
TWS	68.166485	1.0	95.375308	4.217503e-22
Trim	61.329638	1.0	85.809516	4.414967e-20
Heel_Abs	40.981245	1.0	57.339011	5.282257e-14
P_back_X	33.721702	1.0	47.181804	8.302549e-12
M_tot_X	21.374323	1.0	29.905938	5.027348e-08
P_front_Y	17.550853	1.0	24.556319	7.744030e-07
P_back_Y	16.553172	1.0	23.160412	1.587267e-06
F_front	12.792202	1.0	17.898241	2.421934e-05
M_tot_Y	10.928665	1.0	15.290869	9.483659e-05
Line_C	10.591773	1.0	14.819506	1.215410e-04
F_back	10.369975	1.0	14.509176	1.431423e-04
M_back_Y	7.221591	1.0	10.104107	1.499098e-03
M_back_X	6.204488	1.0	8.681025	3.247803e-03
P_front_X	5.123127	1.0	7.168036	7.474273e-03
M_front_Y	5.038947	1.0	7.050255	7.980212e-03
TWA_Abs	4.636008	1.0	6.486482	1.093459e-02
ROT	2.863077	1.0	4.005881	4.545938e-02
Side_lines	2.517192	1.0	3.521935	6.068855e-02
M_front_X	0.198025	1.0	0.277067	5.986811e-01
Residual	1639.563959	2294.0	NaN	NaN

```
Polynomial fit:
R²: 0.406
Used samples: 2312

Formula:
SOG ≈ 27.577 +
  0.179 * Heel_Abs +
  0.101 * Line_C +
  -0.038 * ROT +
  0.044 * Side_lines +
  -0.228 * Trim +
  0.052 * TWA_Abs +
  0.196 * TWS +
  0.000 * boat_weight +
  0.205 * F_front +
  0.201 * F_back +
  -0.339 * M_tot_X +
  0.077 * M_tot_Y +
  -0.145 * M_front_X +
  0.190 * M_front_Y +
  -0.377 * M_back_X +
  0.575 * M_back_Y +
  -0.226 * P_front_X +
  -0.413 * P_front_Y +
  -0.996 * P_back_X +
  -0.644 * P_back_Y
```

	feature	coefficient
18	P_back_X	-0.996403
19	P_back_Y	-0.643512
15	M_back_Y	0.574579
17	P_front_Y	-0.413402
14	M_back_X	-0.376708
10	M_tot_X	-0.339402
4	Trim	-0.228453
16	P_front_X	-0.225973
8	F_front	0.205271
9	F_back	0.201386
6	TWS	0.195777
13	M_front_Y	0.190487
0	Heel_Abs	0.179107
12	M_front_X	-0.145299
1	Line_C	0.101033
11	M_tot_Y	0.076912
5	TWA_Abs	0.052157
3	Side_lines	0.043839
2	ROT	-0.037560
7	boat_weight	0.000000

III.2. Downwind: Gian vs Karl

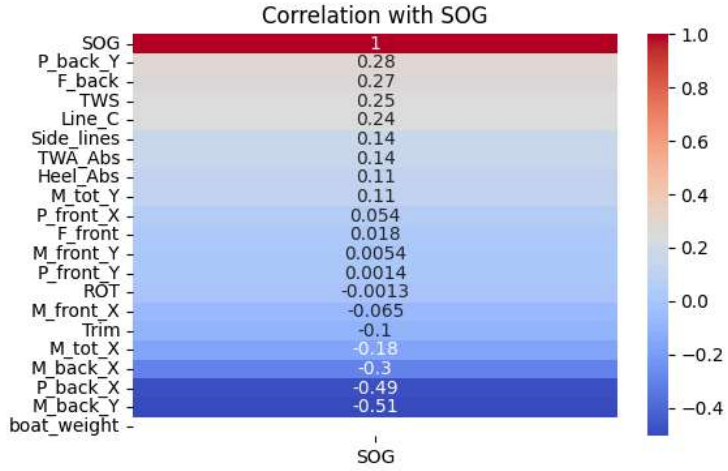
III.2.1. Downwind: Gian

In [21]:
```
gian_data_downwind = downwind_data[
    (downwind_data['boat_name'] == "Gian Stragiotti") |
    ((downwind_data['boat_name'] == "SenseBoard") & (downwind_data['opponent_name'] == "Karl Maeder"))
]
df_numeric_gian_downwind = gian_data_downwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_gian_downwind.drop(columns=[c for c in drop_cols if c in df_numeric_gian_downwind.columns], inplace=True)
df_numeric_gian_downwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_gian_downwind)}")
```

```
Number of rows after filtering: 2327
```

In [22]:
```
full_analysis(df_numeric_gian_downwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	455.523144	1.0	637.346342	2.672430e-124
TWS	68.166485	1.0	95.375308	4.217503e-22
Trim	61.329638	1.0	85.809516	4.414967e-20
Heel_Abs	40.981245	1.0	57.339011	5.282257e-14
P_back_X	33.721702	1.0	47.181804	8.302549e-12
M_tot_X	21.374323	1.0	29.905938	5.027348e-08
P_front_Y	17.550853	1.0	24.556319	7.744030e-07
P_back_Y	16.553172	1.0	23.160412	1.587267e-06
F_front	12.792202	1.0	17.898241	2.421934e-05
M_tot_Y	10.928665	1.0	15.290869	9.483659e-05
Line_C	10.591773	1.0	14.819506	1.215410e-04
F_back	10.369975	1.0	14.509176	1.431423e-04
M_back_Y	7.221591	1.0	10.104107	1.499098e-03
M_back_X	6.204488	1.0	8.681025	3.247803e-03
P_front_X	5.123127	1.0	7.168036	7.474273e-03
M_front_Y	5.038947	1.0	7.050255	7.980212e-03
TWA_Abs	4.636008	1.0	6.486482	1.093459e-02
ROT	2.863077	1.0	4.005881	4.545938e-02
Side_lines	2.517192	1.0	3.521935	6.068855e-02
M_front_X	0.198025	1.0	0.277067	5.986811e-01
Residual	1639.563959	2294.0	NaN	NaN

```
Polynomial fit:
R²: 0.406
Used samples: 2312

Formula:
SOG ≈ 27.577 +
  0.179 * Heel_Abs +
  0.101 * Line_C +
  -0.038 * ROT +
  0.044 * Side_lines +
  -0.228 * Trim +
  0.052 * TWA_Abs +
  0.196 * TWS +
  0.000 * boat_weight +
  0.205 * F_front +
  0.201 * F_back +
  -0.339 * M_tot_X +
  0.077 * M_tot_Y +
  -0.145 * M_front_X +
  0.190 * M_front_Y +
  -0.377 * M_back_X +
  0.575 * M_back_Y +
  -0.226 * P_front_X +
  -0.413 * P_front_Y +
  -0.996 * P_back_X +
  -0.644 * P_back_Y
```

	feature	coefficient
18	P_back_X	-0.996403
19	P_back_Y	-0.643512
15	M_back_Y	0.574579
17	P_front_Y	-0.413402
14	M_back_X	-0.376708
10	M_tot_X	-0.339402
4	Trim	-0.228453
16	P_front_X	-0.225973
8	F_front	0.205271
9	F_back	0.201386
6	TWS	0.195777
13	M_front_Y	0.190487
0	Heel_Abs	0.179107
12	M_front_X	-0.145299
1	Line_C	0.101033
11	M_tot_Y	0.076912
5	TWA_Abs	0.052157
3	Side_lines	0.043839
2	ROT	-0.037560
7	boat_weight	0.000000

III.2.2. Downwind: Karl

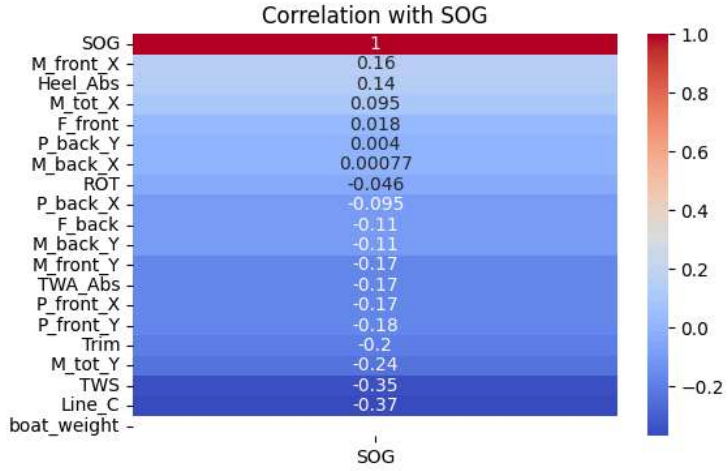
In [23]:
```
karl_data_downwind = downwind_data[
    (downwind_data['boat_name'] == "Karl Maeder") |
    ((downwind_data['boat_name'] == "SenseBoard") & (downwind_data['opponent_name'] == "Gian Stragiotti"))
]
df_numeric_karl_downwind = karl_data_downwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_karl_downwind.drop(columns=[c for c in drop_cols if c in df_numeric_karl_downwind.columns], inplace=True)
df_numeric_karl_downwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_karl_downwind)}")
```

```
Number of rows after filtering: 925
```

In [24]:
```
full_analysis(df_numeric_karl_downwind)
```

```
Dropped column 'Side_lines' (all values were NaN).

Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	82.007094	1.0	148.035617	1.174499e-31
Trim	68.781953	1.0	124.162171	4.077421e-27
Line_C	65.293152	1.0	117.864339	6.710535e-26
Heel_Abs	65.150486	1.0	117.606805	7.527790e-26
TWA_Abs	61.013096	1.0	110.138170	2.137886e-24
ROT	8.819297	1.0	15.920208	7.139871e-05
M_tot_X	3.771901	1.0	6.808871	9.220016e-03
F_front	3.124369	1.0	5.639974	1.776259e-02
F_back	2.634372	1.0	4.755453	2.946193e-02
M_front_X	2.544853	1.0	4.593858	3.235215e-02
P_front_Y	2.541802	1.0	4.588349	3.245574e-02
P_back_X	1.631871	1.0	2.945782	8.644346e-02
M_tot_Y	1.558315	1.0	2.813003	9.384777e-02
M_back_X	0.765337	1.0	1.381552	2.401453e-01
P_front_X	0.608648	1.0	1.098704	2.948299e-01
TWS	0.382422	1.0	0.690332	4.062703e-01
M_back_Y	0.091620	1.0	0.165388	6.843394e-01
M_front_Y	0.069841	1.0	0.126073	7.226215e-01
P_back_Y	0.061145	1.0	0.110376	7.397930e-01
Residual	502.449585	907.0	NaN	NaN

```
Polynomial fit:
R²: 0.357
Used samples: 924

Formula:
SOG ≈ 26.220 +
  0.326 * Heel_Abs +
  -0.519 * Line_C +
  -0.101 * ROT +
  -0.324 * Trim +
  0.405 * TWA_Abs +
  -0.049 * TWS +
  0.000 * boat_weight +
  0.132 * F_front +
  -0.036 * F_back +
  0.116 * M_tot_X +
  -0.179 * M_tot_Y +
  0.186 * M_front_X +
  -0.025 * M_front_Y +
  0.008 * M_back_X +
  0.148 * M_back_Y +
  0.031 * P_front_X +
  0.109 * P_front_Y +
  -0.087 * P_back_X +
  0.053 * P_back_Y
```

	feature	coefficient
1	Line_C	-0.519430
4	TWA_Abs	0.404880
0	Heel_Abs	0.326287
3	Trim	-0.323779
11	M_front_X	0.186291
10	M_tot_Y	-0.179363
14	M_back_Y	0.147521
7	F_front	0.132195
9	M_tot_X	0.116328
16	P_front_Y	0.108985
2	ROT	-0.100793
17	P_back_X	-0.087028
18	P_back_Y	0.052927
5	TWS	-0.049370
8	F_back	-0.036320
15	P_front_X	0.031304
12	M_front_Y	-0.024672
13	M_back_X	0.007935
6	boat_weight	0.000000

III.2.3. Downwind: Karl vs Gian t_test

In [25]:
```
t_test(df_numeric_karl_downwind,df_numeric_gian_downwind)
```

```
T-statistic: -33.288, p-value: 0.000000000000000
The difference is statistically significant, keeping data split.
```

III.3. Downwind: Master vs Slave

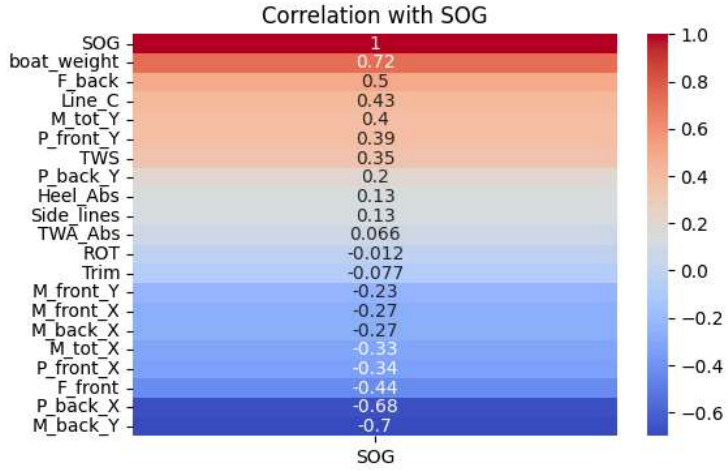
III.3.1 Downwind Master

In [26]:
```
master_data_downwind = downwind_data[downwind_data['boat_role'] == "master"]
df_numeric_master_downwind = master_data_downwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_master_downwind.drop(columns=[c for c in drop_cols if c in df_numeric_master_downwind.columns], inplace=True)
df_numeric_master_downwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_master_downwind)}")
```

```
Number of rows after filtering: 1673
```

In [27]:
```
full_analysis(df_numeric_master_downwind)
```

```
Correlation with SOG:
```




```
ANOVA:
```

	sum_sq	df	F	PR(>F)
boat_weight	116.657192	1.0	192.759904	2.814579e-40
Line_C	19.012061	1.0	31.414805	2.679794e-08
Trim	15.515553	1.0	25.637309	4.885090e-07
P_back_X	14.758800	1.0	24.386880	9.199841e-07
F_front	12.947486	1.0	21.393933	4.218857e-06
TWS	12.896209	1.0	21.309204	4.405493e-06
M_tot_Y	11.106153	1.0	18.351384	2.010116e-05
Heel_Abs	10.874884	1.0	17.969245	2.448163e-05
M_back_Y	10.725206	1.0	17.721922	2.781720e-05
F_back	10.491618	1.0	17.335950	3.396103e-05
M_front_Y	10.066495	1.0	16.633494	4.886798e-05
TWA_Abs	4.725917	1.0	7.808925	5.296130e-03
M_tot_X	4.251992	1.0	7.025830	8.158408e-03
P_back_Y	3.856650	1.0	6.372581	1.174018e-02
Side_lines	1.292217	1.0	2.135211	1.442580e-01
ROT	1.166214	1.0	1.927008	1.653886e-01
P_front_Y	1.161855	1.0	1.919804	1.661805e-01
M_back_X	0.776009	1.0	1.282247	2.577472e-01
P_front_X	0.118401	1.0	0.195641	6.583560e-01
M_front_X	0.028930	1.0	0.047803	8.269758e-01
Residual	618.508556	1022.0	NaN	NaN

```
Polynomial fit:
R²: 0.243
Used samples: 1040

Formula:
SOG ≈ 27.765 +
  0.142 * Heel_Abs +
  0.218 * Line_C +
  -0.036 * ROT +
  -0.048 * Side_lines +
  -0.171 * Trim +
  0.086 * TWA_Abs +
  0.133 * TWS +
  0.000 * boat_weight +
  0.434 * F_front +
  0.546 * F_back +
  -0.232 * M_tot_X +
  0.126 * M_tot_Y +
  -0.122 * M_front_X +
  -0.033 * M_front_Y +
  -0.213 * M_back_X +
  0.999 * M_back_Y +
  -0.056 * P_front_X +
  -0.152 * P_front_Y +
  -0.865 * P_back_X +
  -0.503 * P_back_Y
```

	feature	coefficient
15	M_back_Y	0.999338
18	P_back_X	-0.865282
9	F_back	0.545633
19	P_back_Y	-0.502918
8	F_front	0.433570
10	M_tot_X	-0.231638
1	Line_C	0.217625
14	M_back_X	-0.213070
4	Trim	-0.170999
17	P_front_Y	-0.152307
0	Heel_Abs	0.142138
6	TWS	0.132659
11	M_tot_Y	0.126045
12	M_front_X	-0.122275
5	TWA_Abs	0.086331
16	P_front_X	-0.056068
3	Side_lines	-0.047655
2	ROT	-0.036298
13	M_front_Y	-0.033356
7	boat_weight	0.000000

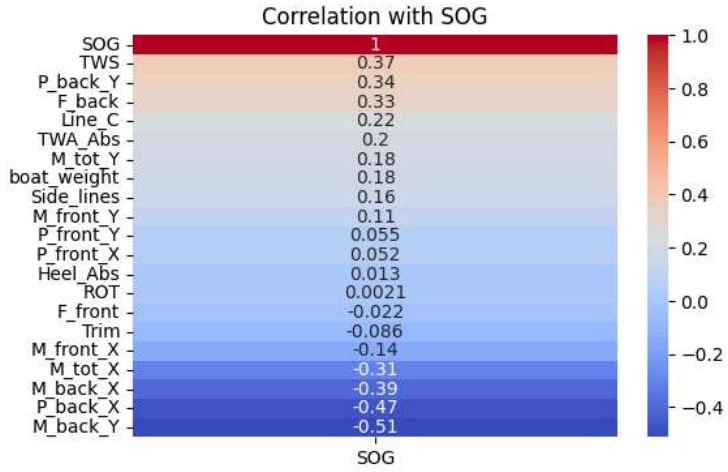
III.3.2 Downwind Slave

In [28]:
```
slave_data_downwind = downwind_data[downwind_data['boat_role'] == "slave"]
df_numeric_slave_downwind = slave_data_downwind.select_dtypes(include=["float64", "int64"]).copy()
df_numeric_slave_downwind.drop(columns=[c for c in drop_cols if c in df_numeric_slave_downwind.columns], inplace=True)
df_numeric_slave_downwind.dropna(subset=["SOG"], inplace=True)
print(f"Number of rows after filtering: {len(df_numeric_slave_downwind)}")
```

```
Number of rows after filtering: 1579
```

In [29]:
```
full_analysis(df_numeric_slave_downwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
boat_weight	183.591788	1.0	290.137687	1.098204e-58
TWS	59.447749	1.0	93.947734	1.795355e-21
Trim	33.475988	1.0	52.903487	6.153368e-13
P_front_Y	31.035907	1.0	49.047325	4.064087e-12
TWA_Abs	23.422396	1.0	37.015380	1.554257e-09
Heel_Abs	17.890282	1.0	28.272752	1.246442e-07
P_back_X	17.178718	1.0	27.148238	2.201187e-07
P_front_X	9.155276	1.0	14.468462	1.493883e-04
Side_lines	7.126972	1.0	11.263049	8.143217e-04
M_tot_X	5.869559	1.0	9.275907	2.370325e-03
F_back	5.551519	1.0	8.773295	3.114302e-03
F_front	5.460452	1.0	8.629378	3.368290e-03
M_tot_Y	5.386116	1.0	8.511901	3.591189e-03
M_front_X	2.952052	1.0	4.665250	3.096797e-02
ROT	2.846112	1.0	4.497828	3.413376e-02
P_back_Y	2.391100	1.0	3.778754	5.213072e-02
Line_C	1.222609	1.0	1.932139	1.647713e-01
M_back_Y	1.201419	1.0	1.898653	1.684750e-01
M_front_Y	0.124170	1.0	0.196231	6.578566e-01
M_back_X	0.058984	1.0	0.093215	7.601798e-01
Residual	793.499476	1254.0	NaN	NaN

```
Polynomial fit:
R²: 0.577
Used samples: 1272

Formula:
SOG ≈ 27.424 +
  0.160 * Heel_Abs +
  -0.045 * Line_C +
  -0.051 * ROT +
  0.099 * Side_lines +
  -0.236 * Trim +
  0.188 * TWA_Abs +
  0.242 * TWS +
  0.000 * boat_weight +
  0.080 * F_front +
  0.023 * F_back +
  -0.379 * M_tot_X +
  0.135 * M_tot_Y +
  -0.385 * M_front_X +
  0.512 * M_front_Y +
  -0.005 * M_back_X +
  0.397 * M_back_Y +
  -0.396 * P_front_X +
  -0.846 * P_front_Y +
  -1.056 * P_back_X +
  -0.306 * P_back_Y
```

	feature	coefficient
18	P_back_X	-1.055956
17	P_front_Y	-0.846085
13	M_front_Y	0.512025
15	M_back_Y	0.397000
16	P_front_X	-0.396178
12	M_front_X	-0.385088
10	M_tot_X	-0.378664
19	P_back_Y	-0.306287
6	TWS	0.242216
4	Trim	-0.236456
5	TWA_Abs	0.188347
0	Heel_Abs	0.160027
11	M_tot_Y	0.134828
3	Side_lines	0.099253
8	F_front	0.080312
2	ROT	-0.050914
1	Line_C	-0.045453
9	F_back	0.022894
14	M_back_X	-0.004623
7	boat_weight	0.000000

III.3.3. Downwind: Master vs Slave t_test

In [30]:
```
t_test(df_numeric_master_downwind,df_numeric_slave_downwind)
```

```
T-statistic: -6.001, p-value: 0.000000002171240
The difference is statistically significant, keeping data split.
```