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
from sklearn.preprocessing import RobustScaler
import warnings
warnings.filterwarnings("ignore")



drop_cols = [
    "TWD","COG","TWA", "TimeUTC", "SecondsSince1970", "ISODateTimeUTC","Latitude","Longitude","Euler_X (deg)","Euler_Y (deg)",
    "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"
]

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)

# With simple standard scaler
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
# Regularisaton with lasso
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)

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import RobustScaler
from sklearn.metrics import r2_score

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

    # Drop missing values
    df_clean = df_copy.dropna(subset=[target] + [col for col in df_copy.columns if col != target])
    X = df_clean.drop(columns=[target])
    y = df_clean[target]

    # Manual robust scaling
    scaler = RobustScaler()
    X_scaled = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)

    # OLS Regression
    model = LinearRegression()
    model.fit(X_scaled, y)
    y_pred = model.predict(X_scaled)

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

```
    # Coefficient analysis
    coefs = model.coef_
    intercept = model.intercept_

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

    # Print readable 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"):
    # 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
27182	2025-06-07 12:18:12.359000+00:00	1.749299e+09	38.3	38.3	43.514855	43.514857	43.514851	43.514846	1.0	67.000	...	-1283.160374	-284.229902	-815.910320	-92.576328	-467.250054	3(
55595	2025-06-09 13:28:07.360000+00:00	1.749476e+09	47.6	47.6	43.506216	43.506218	43.506212	43.506206	NaN	5.300	...	-2371.706268	1418.689814	-1040.980773	-197.525690	-1330.725495	-8(
63645	2025-06-10 12:44:24.857000+00:00	1.749559e+09	53.3	53.3	43.534359	43.534357	43.534363	43.534369	NaN	115.715	...	10011.516880	1616.431532	7203.701558	1764.848024	2807.815321	-3(
68295	2025-06-10 13:02:46.457000+00:00	1.749561e+09	56.3	56.3	43.533271	43.533269	43.533275	43.533281	NaN	101.400	...	10863.739839	7005.677461	6543.846389	1262.898776	4319.893450	-13(
34502	2025-06-07 13:03:47.063000+00:00	1.749301e+09	50.4	50.4	43.514103	43.514105	43.514100	43.514096	1.0	77.076	...	-777.176717	-765.271102	-864.138144	-122.082255	86.961427	-(
34468	2025-06-07 13:03:45.357000+00:00	1.749301e+09	33.7	33.7	43.513952	43.513953	43.513948	43.513944	1.0	78.200	...	-1272.297894	-255.606557	-1437.347670	-123.292255	165.049777	-1(
78715	2025-06-10 13:43:12.553000+00:00	1.749563e+09	45.7	45.7	43.535173	43.535175	43.535169	43.535163	NaN	93.900	...	-1079.209155	4447.514202	-508.607748	296.444850	-570.601407	-9(
26537	2025-06-07 12:15:31.444000+00:00	1.749299e+09	58.6	58.6	43.516040	43.516038	43.516044	43.516049	1.0	117.100	...	5743.769612	-393.365826	4733.782084	743.396538	1009.987527	-3(
66590	2025-06-10 12:53:38.471000+00:00	1.749560e+09	65.6	65.6	43.531291	43.531289	43.531295	43.531300	NaN	109.200	...	7106.104112	7502.418021	5057.937720	334.410612	2048.166392	-6(
72213	2025-06-10 13:22:39.556000+00:00	1.749562e+09	56.9	56.9	43.534803	43.534801	43.534807	43.534812	NaN	144.000	...	11601.677276	6143.395780	7424.225717	1738.473400	4177.451558	-13(

10 rows × 71 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']
Number of rows after filtering: 30546
```

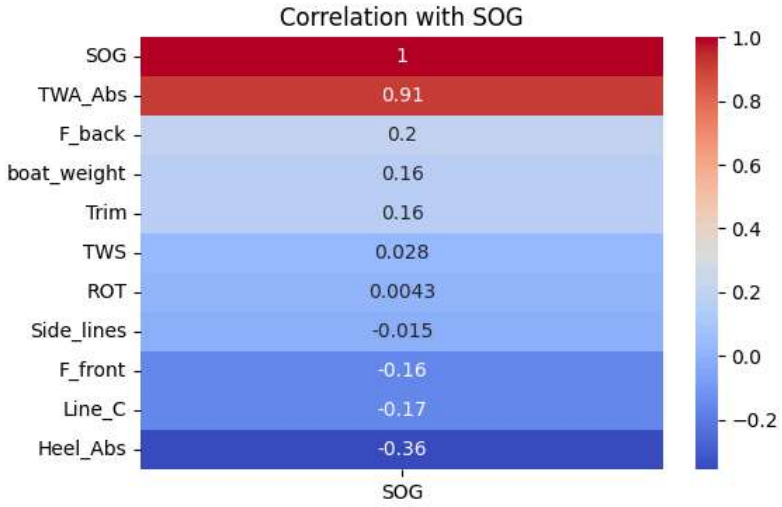
In [4]:
```
df_numeric.sample(20)
```

Out[4]:

	Heel_Abs	Line_C	ROT	Side_lines	Trim	TWA_Abs	TWS	SOG	boat_weight	F_front	F_back
62817	53.2	90.100	4.587	NaN	7.8	130.200	6.866	24.6	102.89	34784.008390	53934.845417
62370	59.1	83.500	4.545	NaN	14.1	126.665	7.165	25.4	102.89	37443.014849	56549.027043
19652	55.4	115.092	-36.893	4.500	6.3	52.200	11.856	23.6	109.09	11881.282987	27775.947539
32198	49.1	7.900	-9.783	NaN	9.6	144.107	7.758	26.0	102.89	28630.250976	34016.103360
34549	48.1	87.358	-3.960	11.600	8.0	143.617	8.800	26.6	109.09	23115.218456	27020.527284
19480	46.1	100.200	-38.000	6.827	7.6	52.084	11.200	22.8	109.09	13793.963897	32081.398393
20552	44.3	89.200	6.000	15.000	9.6	157.000	9.411	28.6	109.09	19012.873518	41587.145802
50972	49.3	2.600	-7.292	92.863	6.6	141.296	9.465	27.8	115.09	24486.993259	42587.045290
68552	59.9	119.192	2.174	NaN	6.4	59.000	5.600	22.7	102.89	43521.757989	58379.976922
56028	53.1	4.500	24.490	130.200	11.0	47.048	7.774	22.3	115.09	34156.267070	43871.435195
56202	58.1	6.600	-7.071	116.700	6.8	51.691	7.500	21.8	115.09	36110.393162	40964.210846
28885	58.7	5.300	19.608	NaN	8.2	28.726	8.600	20.5	102.89	27353.351567	20124.249880
63237	64.7	109.600	3.738	NaN	10.4	51.600	6.000	21.5	102.89	37185.222885	47818.874979
49321	54.7	9.500	20.952	130.341	9.3	52.816	6.400	23.8	115.09	39280.990480	35715.303430
54763	37.3	5.300	17.431	92.330	4.7	58.736	6.864	23.6	115.09	34150.443219	34482.984248
12027	57.1	128.822	7.865	15.029	8.7	42.025	9.000	21.7	109.09	20042.851331	33621.791142
37906	62.9	117.200	-9.184	13.622	9.3	35.556	10.955	23.2	109.09	23649.054986	28524.625068
62934	54.7	127.700	-23.913	NaN	4.6	129.300	6.400	24.3	102.89	36206.378717	57227.840030
16330	48.7	68.927	8.511	7.816	14.1	142.317	9.791	26.3	109.09	11463.527923	34696.482826
53159	49.3	6.800	-3.158	115.032	7.8	143.765	6.735	27.5	115.09	25801.322442	46554.912277

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

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
TWA_Abs	50237.647144	1.0	69919.907652	0.000000e+00
TWS	1296.353938	1.0	1804.243486	0.000000e+00
Side_lines	711.900262	1.0	990.810745	1.504496e-212
F_front	705.964962	1.0	982.550094	7.807193e-211
Trim	431.924574	1.0	601.145317	6.488255e-131
boat_weight	275.959843	1.0	384.076243	9.205647e-85
Line_C	153.576337	1.0	213.744949	3.609029e-48
Heel_Abs	31.854273	1.0	44.334238	2.836645e-11
ROT	27.570528	1.0	38.372195	5.954434e-10
F_back	8.513609	1.0	11.849097	5.780156e-04
Residual	15106.520684	21025.0	NaN	NaN

```
Polynomial fit:
R²: 0.876
Used samples: 21036

 Formula :
SOG ≈ 24.463 +
  0.049 * Heel_Abs +
  0.336 * Line_C +
  0.036 * ROT +
  0.738 * Side_lines +
  -0.153 * Trim +
  2.482 * TWA_Abs +
  0.356 * TWS +
  -0.163 * boat_weight +
  0.277 * F_front +
  0.034 * F_back
```

	feature	coefficient
5	TWA_Abs	2.482302
3	Side_lines	0.738299
6	TWS	0.355919
1	Line_C	0.335935
8	F_front	0.277414
7	boat_weight	-0.162902
4	Trim	-0.152871
0	Heel_Abs	0.048917
2	ROT	0.036293
9	F_back	0.034038

II Upwind:

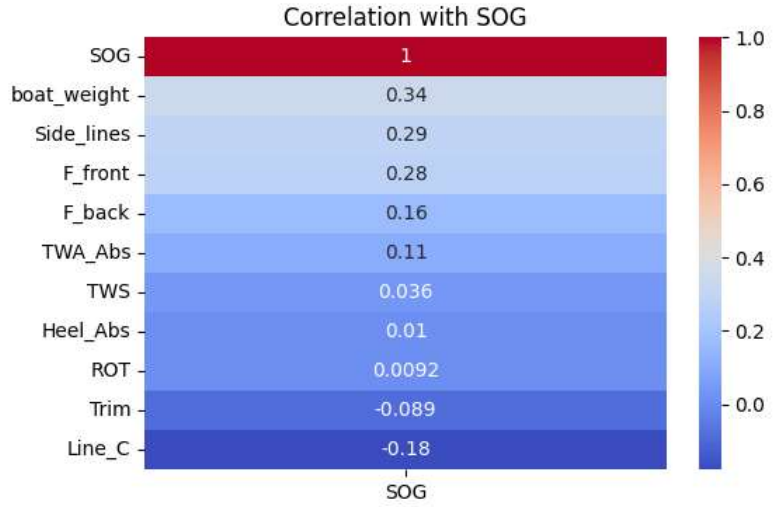
II.1. All upwind data

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: 18876
```

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

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
F_front	745.352709	1.0	1412.437003	8.775914e-294
TWS	275.919193	1.0	522.864509	1.582006e-113
Trim	161.972730	1.0	306.936938	5.933708e-68
Side_lines	101.922591	1.0	193.142438	1.327238e-43
TWA_Abs	64.713004	1.0	122.630589	2.238728e-28
F_back	36.586519	1.0	69.331140	9.137337e-17
Heel_Abs	23.480768	1.0	44.495855	2.650378e-11
ROT	11.204361	1.0	21.232167	4.106812e-06
Line_C	1.860791	1.0	3.526183	6.042871e-02
boat_weight	0.355136	1.0	0.672980	4.120292e-01
Residual	6977.340227	13222.0	NaN	NaN

```
Polynomial fit:
R²: 0.321
Used samples: 13233

 Formula :
SOG ≈ 22.761 +
  -0.045 * Heel_Abs +
  0.050 * Line_C +
  0.030 * ROT +
  0.385 * Side_lines +
  -0.112 * Trim +
  0.082 * TWA_Abs +
  0.218 * TWS +
  0.007 * boat_weight +
  0.372 * F_front +
  0.093 * F_back
```

	feature	coefficient
3	Side_lines	0.384715
8	F_front	0.371517
6	TWS	0.217549
4	Trim	-0.111691
9	F_back	0.093087
5	TWA_Abs	0.082455
1	Line_C	0.050125
0	Heel_Abs	-0.045306
2	ROT	0.029856
7	boat_weight	0.007225

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)}")
```

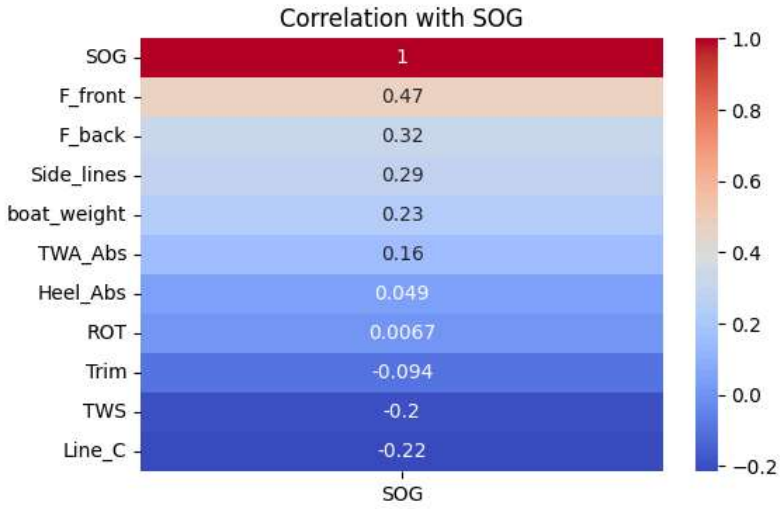
```
Number of rows after filtering: 13275
```

In [9]:
```
print(gian_data_upwind["boat_weight"].mean)
```

```
<bound method Series.mean of 11633    109.09
11636    109.09
11637    109.09
11640    109.09
11641    109.09
          ...
80121    109.09
80124    109.09
80126    109.09
80127    109.09
80129    109.09
Name: boat_weight, Length: 13275, dtype: float64>
```

In [10]:
```
full_analysis(df_numeric_gian_upwind)
```

Correlation with SOG:



ANOVA:

	sum_sq	df	F	PR(>F)
F_front	745.352709	1.0	1412.437003	8.775914e-294
TWS	275.919193	1.0	522.864509	1.582006e-113
Trim	161.972730	1.0	306.936938	5.933708e-68
Side_lines	101.922591	1.0	193.142438	1.327238e-43
TWA_Abs	64.713004	1.0	122.630589	2.238728e-28
F_back	36.586519	1.0	69.331140	9.137337e-17
Heel_Abs	23.480768	1.0	44.495855	2.650378e-11
ROT	11.204361	1.0	21.232167	4.106812e-06
Line_C	1.860791	1.0	3.526183	6.042871e-02
boat_weight	0.355136	1.0	0.672980	4.120292e-01
Residual	6977.340227	13222.0	NaN	NaN

```
Polynomial fit:
R²: 0.321
Used samples: 13233

 Formula :
SOG ≈ 22.761 +
  -0.045 * Heel_Abs +
  0.050 * Line_C +
  0.030 * ROT +
  0.385 * Side_lines +
  -0.112 * Trim +
  0.082 * TWA_Abs +
  0.218 * TWS +
  0.007 * boat_weight +
  0.372 * F_front +
  0.093 * F_back
```

	feature	coefficient
3	Side_lines	0.384715
8	F_front	0.371517
6	TWS	0.217549
4	Trim	-0.111691
9	F_back	0.093087
5	TWA_Abs	0.082455
1	Line_C	0.050125
0	Heel_Abs	-0.045306
2	ROT	0.029856
7	boat_weight	0.007225

II.2.2. Upwind: Karl

In [11]:
```
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: 5601
(5601, 11)
Index(['Heel_Abs', 'Line_C', 'ROT', 'Side_lines', 'Trim', 'TWA_Abs', 'TWS',
       'SOG', 'boat_weight', 'F_front', 'F_back'],
      dtype='object')
```

In [12]:
```
print(karl_data_upwind["boat_weight"].mean)
```

```
<bound method Series.mean of 25553    102.89
25555    102.89
25557    102.89
25560    102.89
25562    102.89
          ...
68904    102.89
68907    102.89
68908    102.89
68911    102.89
68912    102.89
Name: boat_weight, Length: 5601, dtype: float64>
```

In [13]:
```
df_numeric_karl_upwind.head(10)
```

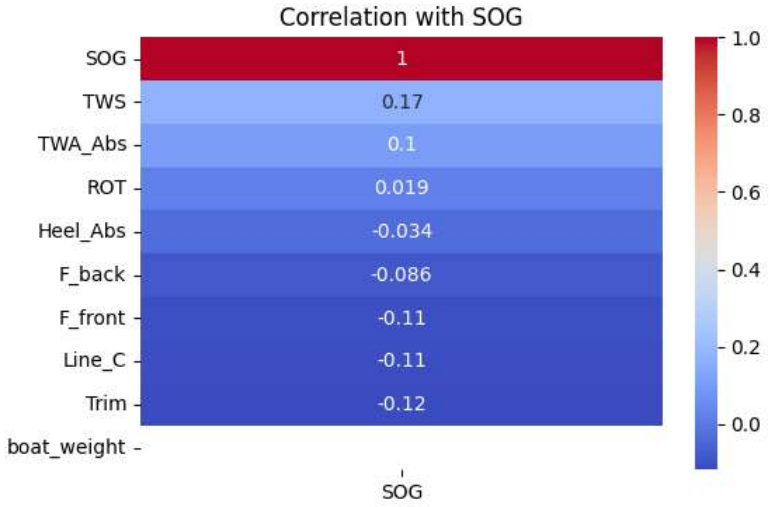
Out[13]:

	Heel_Abs	Line_C	ROT	Side_lines	Trim	TWA_Abs	TWS	SOG	boat_weight	F_front	F_back
25553	59.5	93.3	11.000	NaN	7.5	41.174	9.352	21.4	102.89	27567.598622	24661.188272
25555	57.4	88.1	-14.141	NaN	12.3	42.564	9.372	21.4	102.89	30282.257552	24532.446152
25557	59.8	98.4	1.020	NaN	8.7	42.454	9.391	21.3	102.89	30629.872010	26538.607253
25560	57.8	108.1	15.000	NaN	1.5	40.944	9.411	21.4	102.89	29534.445709	25671.679650
25562	53.0	111.4	-7.619	NaN	4.6	41.734	9.432	21.4	102.89	31969.060202	27080.840797
25563	54.8	111.3	-10.309	NaN	7.0	42.724	9.452	21.4	102.89	32674.719191	26459.098753
25565	54.6	110.7	11.224	NaN	7.0	41.614	9.471	21.3	102.89	34743.378192	26875.356725
25567	47.0	109.9	-19.802	NaN	6.2	43.604	9.492	21.4	102.89	34743.378192	26875.356725
25569	41.2	107.5	7.000	NaN	4.4	42.894	9.506	21.3	102.89	30655.266332	26253.937186
25571	51.7	104.6	1.000	NaN	8.5	42.784	9.516	21.4	102.89	30655.266332	26253.937186

In [14]:
```
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	4598.244751	1.0	9685.361760	0.000000e+00
TWA_Abs	246.517194	1.0	519.243393	5.706783e-110
TWS	195.231855	1.0	411.220205	2.842816e-88
Trim	93.653566	1.0	197.264112	4.622077e-44
F_back	15.631601	1.0	32.925110	1.008229e-08
ROT	12.608568	1.0	26.557643	2.646028e-07
Heel_Abs	9.943821	1.0	20.944841	4.829555e-06
Line_C	5.844075	1.0	12.309476	4.542211e-04
F_front	2.640326	1.0	5.561364	1.839543e-02
Residual	2646.799895	5575.0	NaN	NaN

```
Polynomial fit:
R²: 0.135
Used samples: 5584

 Formula :
SOG ≈ 22.227 +
  0.046 * Heel_Abs +
  -0.045 * Line_C +
  0.049 * ROT +
  -0.136 * Trim +
  0.274 * TWA_Abs +
  0.336 * TWS +
  0.000 * boat_weight +
  -0.029 * F_front +
  0.086 * F_back
```

	feature	coefficient
5	TWS	3.358031e-01
4	TWA_Abs	2.736814e-01
3	Trim	-1.357438e-01
8	F_back	8.641826e-02
2	ROT	4.868985e-02
0	Heel_Abs	4.570711e-02
1	Line_C	-4.452198e-02
7	F_front	-2.866813e-02
6	boat_weight	5.551115e-17

II.2.3. Upwind: Karl vs Gian t_test

In [15]:
```
t_test(df_numeric_gian_upwind,df_numeric_karl_upwind)
```

```
T-statistic: 39.897, p-value: 0.000000000000000
The difference is statistically significant, keeping data split.
```

II.3. Upwind: Master vs Slave

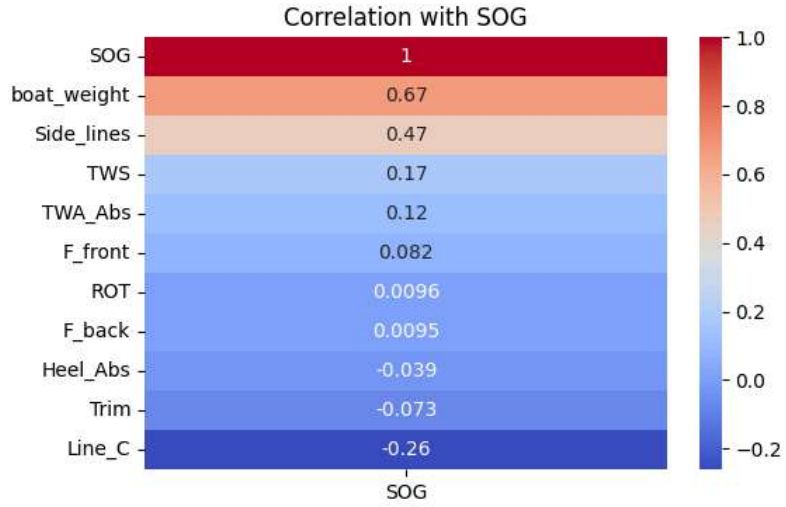
II.3.1. Master

In [16]:
```
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: 9638
```

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

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
F_front	168.253949	1.0	480.684713	1.004899e-102
Side_lines	70.414496	1.0	201.167177	5.723545e-45
Trim	50.526690	1.0	144.349702	6.826664e-33
TWS	18.385077	1.0	52.524329	4.760053e-13
Heel_Abs	8.065963	1.0	23.043649	1.620350e-06
boat_weight	7.376923	1.0	21.075133	4.502315e-06
Line_C	4.703126	1.0	13.436360	2.488331e-04
F_back	1.449286	1.0	4.140464	4.191165e-02
ROT	0.959391	1.0	2.740884	9.786063e-02
TWA_Abs	0.929970	1.0	2.656830	1.031562e-01
Residual	2196.086636	6274.0	NaN	NaN

```
Polynomial fit:
R²: 0.358
Used samples: 6285

 Formula :
SOG ≈ 23.170 +
  -0.039 * Heel_Abs +
  0.110 * Line_C +
  0.013 * ROT +
  0.445 * Side_lines +
  -0.091 * Trim +
  -0.017 * TWA_Abs +
  0.091 * TWS +
  0.059 * boat_weight +
  0.270 * F_front +
  0.029 * F_back
```

	feature	coefficient
3	Side_lines	0.445396
8	F_front	0.270209
1	Line_C	0.110405
6	TWS	0.091217
4	Trim	-0.090709
7	boat_weight	0.058505
0	Heel_Abs	-0.038598
9	F_back	0.028592
5	TWA_Abs	-0.017015
2	ROT	0.012810

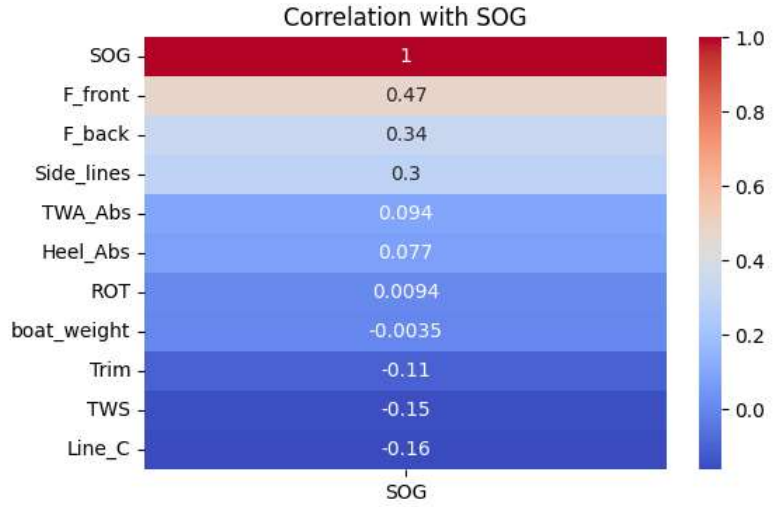
II.3.2 Slave

In [18]:
```
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: 9238
```

In [19]:
```
full_analysis(df_numeric_slave_upwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
F_front	258.589307	1.0	616.290151	2.058431e-130
F_back	85.092555	1.0	202.799196	2.221636e-45
Trim	63.317539	1.0	150.903284	2.497524e-34
Side_lines	48.725358	1.0	116.126064	7.272527e-27
TWS	42.270916	1.0	100.743336	1.515060e-23
boat_weight	33.370761	1.0	79.531794	5.983729e-19
Heel_Abs	22.101350	1.0	52.673657	4.366700e-13
TWA_Abs	10.436726	1.0	24.873618	6.270176e-07
Line_C	0.323681	1.0	0.771422	3.798080e-01
ROT	0.067776	1.0	0.161529	6.877648e-01
Residual	2910.697207	6937.0	NaN	NaN

```
Polynomial fit:
R²: 0.401
Used samples: 6948

 Formula :
SOG ≈ 22.390 +
  -0.061 * Heel_Abs +
  0.030 * Line_C +
  0.003 * ROT +
  0.381 * Side_lines +
  -0.097 * Trim +
  -0.048 * TWA_Abs +
  0.134 * TWS +
  -0.091 * boat_weight +
  0.315 * F_front +
  0.200 * F_back
```

	feature	coefficient
3	Side_lines	0.380791
8	F_front	0.315048
9	F_back	0.200336
6	TWS	0.133552
4	Trim	-0.096585
7	boat_weight	-0.090734
0	Heel_Abs	-0.060942
5	TWA_Abs	-0.047994
1	Line_C	0.030051
2	ROT	0.003210

III.3.3. Upwind: Master vs Slave t_test

In [20]:
```
t_test(df_numeric_master_upwind,df_numeric_slave_upwind)
```

```
T-statistic: 25.880, p-value: 0.000000000000000
The difference is statistically significant, keeping data split.
```

III Downwind

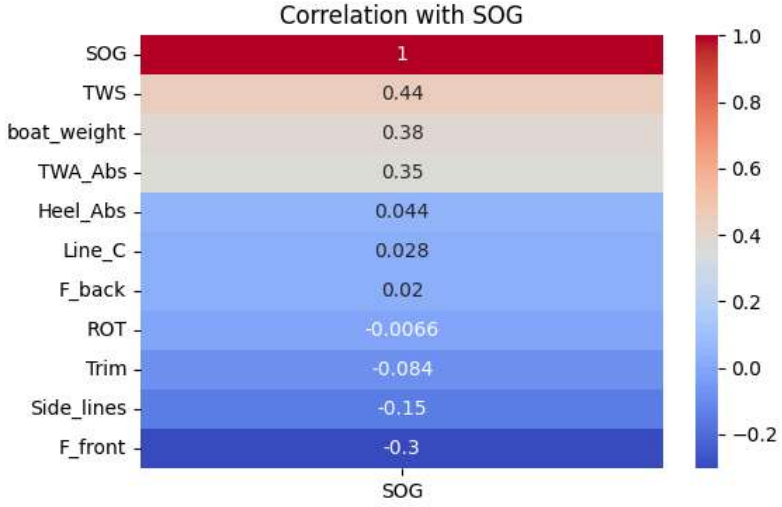
III.1. All downwind data

In [21]:
```
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: 11670
```

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

Correlation with SOG:



ANOVA:

	sum_sq	df	F	PR(>F)
TWS	962.827764	1.0	1191.018987	5.314048e-243
Side_lines	379.737135	1.0	469.735247	3.407176e-101
boat_weight	209.666280	1.0	259.357416	1.989188e-57
Line_C	144.803508	1.0	179.122096	2.100678e-40
Trim	133.760353	1.0	165.461702	1.742471e-37
Heel_Abs	129.867362	1.0	160.646067	1.869130e-36
TWA_Abs	74.036677	1.0	91.583450	1.405424e-21
F_back	36.755196	1.0	45.466217	1.663794e-11
F_front	33.688069	1.0	41.672178	1.143975e-10
ROT	1.926524	1.0	2.383113	1.226936e-01
Residual	6299.105237	7792.0	NaN	NaN

```
Polynomial fit:
R²: 0.262
Used samples: 7803

 Formula :
SOG ≈ 27.350 +
  0.174 * Heel_Abs +
  0.503 * Line_C +
  -0.016 * ROT +
  0.828 * Side_lines +
  -0.158 * Trim +
  0.124 * TWA_Abs +
  0.498 * TWS +
  -0.264 * boat_weight +
  -0.100 * F_front +
  0.112 * F_back
```

	feature	coefficient
3	Side_lines	0.827594
1	Line_C	0.503088
6	TWS	0.497873
7	boat_weight	-0.264347
0	Heel_Abs	0.173604
4	Trim	-0.158352
5	TWA_Abs	0.124349
9	F_back	0.111924
8	F_front	-0.100203
2	ROT	-0.016486

III.2. Downwind: Gian vs Karl

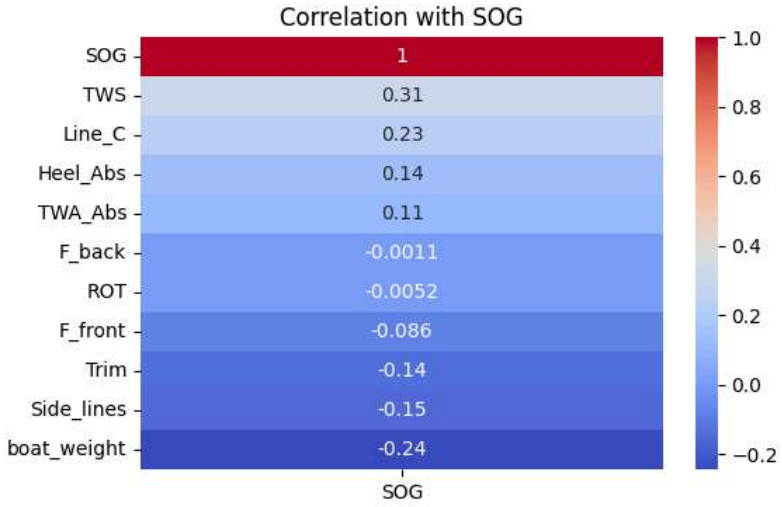
III.2.1. Downwind: Gian

In [23]:
```
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: 7826
```

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

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
TWS	962.827764	1.0	1191.018987	5.314048e-243
Side_lines	379.737135	1.0	469.735247	3.407176e-101
boat_weight	209.666280	1.0	259.357416	1.989188e-57
Line_C	144.803508	1.0	179.122096	2.100678e-40
Trim	133.760353	1.0	165.461702	1.742471e-37
Heel_Abs	129.867362	1.0	160.646067	1.869130e-36
TWA_Abs	74.036677	1.0	91.583450	1.405424e-21
F_back	36.755196	1.0	45.466217	1.663794e-11
F_front	33.688069	1.0	41.672178	1.143975e-10
ROT	1.926524	1.0	2.383113	1.226936e-01
Residual	6299.105237	7792.0	NaN	NaN

```
Polynomial fit:
R²: 0.262
Used samples: 7803

 Formula :
SOG ≈ 27.350 +
  0.174 * Heel_Abs +
  0.503 * Line_C +
  -0.016 * ROT +
  0.828 * Side_lines +
  -0.158 * Trim +
  0.124 * TWA_Abs +
  0.498 * TWS +
  -0.264 * boat_weight +
  -0.100 * F_front +
  0.112 * F_back
```

	feature	coefficient
3	Side_lines	0.827594
1	Line_C	0.503088
6	TWS	0.497873
7	boat_weight	-0.264347
0	Heel_Abs	0.173604
4	Trim	-0.158352
5	TWA_Abs	0.124349
9	F_back	0.111924
8	F_front	-0.100203
2	ROT	-0.016486

III.2.2. Downwind: Karl

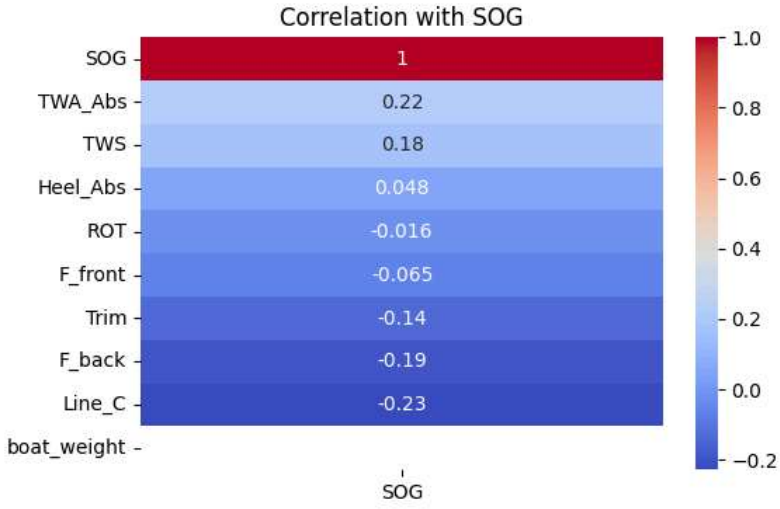
In [25]:
```
karl_data_downwind = downwind_data[
    (downwind_data['boat_name'] == "Karl Maeder") |
    ((downwind_data['boat_name'] == "SenseBoard") & (downwind_data['opponent_name'] == "Gian Stragiotti"))
]
df_numeric_karl_downwind = karl_data_downwind.select_dtypes(include=["float64", "int64"]).copy()
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: 3844
```

In [26]:
```
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	1978.470978	1.0	3024.729178	0.000000e+00
Trim	292.014819	1.0	446.438565	8.025132e-94
Heel_Abs	177.459225	1.0	271.303498	6.013933e-59
Line_C	161.363777	1.0	246.696429	6.355927e-54
F_front	133.389888	1.0	203.929342	4.098325e-45
TWA_Abs	55.567557	1.0	84.952882	4.914748e-20
F_back	16.445748	1.0	25.142614	5.565583e-07
TWS	11.771670	1.0	17.996784	2.265131e-05
ROT	6.755128	1.0	10.327385	1.321582e-03
Residual	2504.543358	3829.0	NaN	NaN

```
Polynomial fit:
R²: 0.230
Used samples: 3838

 Formula :
SOG ≈ 25.975 +
  0.258 * Heel_Abs +
  -0.332 * Line_C +
  -0.045 * ROT +
  -0.325 * Trim +
  0.204 * TWA_Abs +
  0.090 * TWS +
  0.000 * boat_weight +
  0.462 * F_front +
  -0.231 * F_back
```

	feature	coefficient
7	F_front	4.622215e-01
1	Line_C	-3.324574e-01
3	Trim	-3.247854e-01
0	Heel_Abs	2.577719e-01
8	F_back	-2.305324e-01
4	TWA_Abs	2.043650e-01
5	TWS	9.002057e-02
2	ROT	-4.452262e-02
6	boat_weight	3.330669e-16

III.2.3. Downwind: Karl vs Gian t_test

In [27]:
```
t_test(df_numeric_karl_downwind,df_numeric_gian_downwind)
```

```
T-statistic: -69.413, p-value: 0.000000000000000
The difference is statistically significant, keeping data split.
```

III.3. Downwind: Master vs Slave

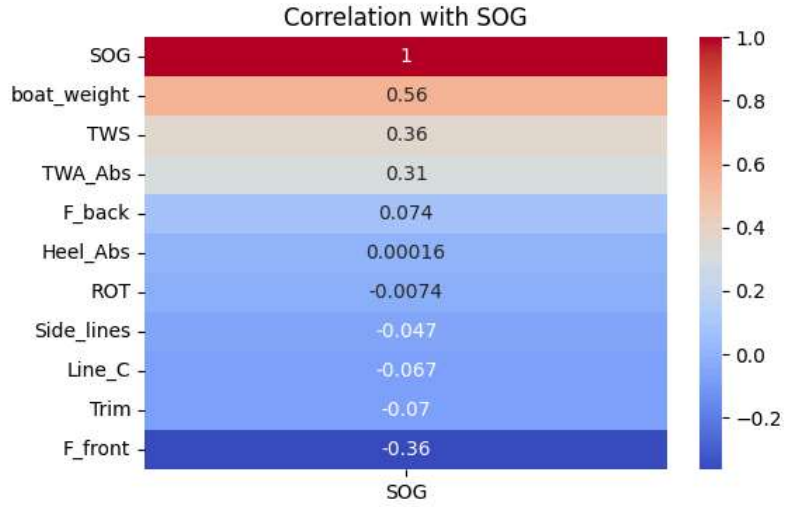
III.3.1 Downwind Master

In [28]:
```
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: 6136
```

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

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
TWS	182.630180	1.0	308.683119	1.885832e-66
TWA_Abs	74.777435	1.0	126.389471	7.328615e-29
Trim	53.496760	1.0	90.420690	3.343194e-21
Side_lines	50.591508	1.0	85.510207	3.779641e-20
Heel_Abs	44.092802	1.0	74.526037	8.729937e-18
boat_weight	19.757555	1.0	33.394392	8.140845e-09
F_front	18.181873	1.0	30.731160	3.169275e-08
Line_C	5.539048	1.0	9.362147	2.230889e-03
F_back	4.718138	1.0	7.974639	4.768922e-03
ROT	2.745746	1.0	4.640884	3.128283e-02
Residual	2204.461489	3726.0	NaN	NaN

```
Polynomial fit:
R²: 0.183
Used samples: 3737

 Formula :
SOG ≈ 27.537 +
  0.143 * Heel_Abs +
  0.146 * Line_C +
  -0.028 * ROT +
  0.478 * Side_lines +
  -0.143 * Trim +
  0.179 * TWA_Abs +
  0.400 * TWS +
  -0.138 * boat_weight +
  -0.105 * F_front +
  0.061 * F_back
```

	feature	coefficient
3	Side_lines	0.478106
6	TWS	0.400064
5	TWA_Abs	0.179439
1	Line_C	0.145757
4	Trim	-0.142889
0	Heel_Abs	0.142679
7	boat_weight	-0.138464
8	F_front	-0.104683
9	F_back	0.060866
2	ROT	-0.028425

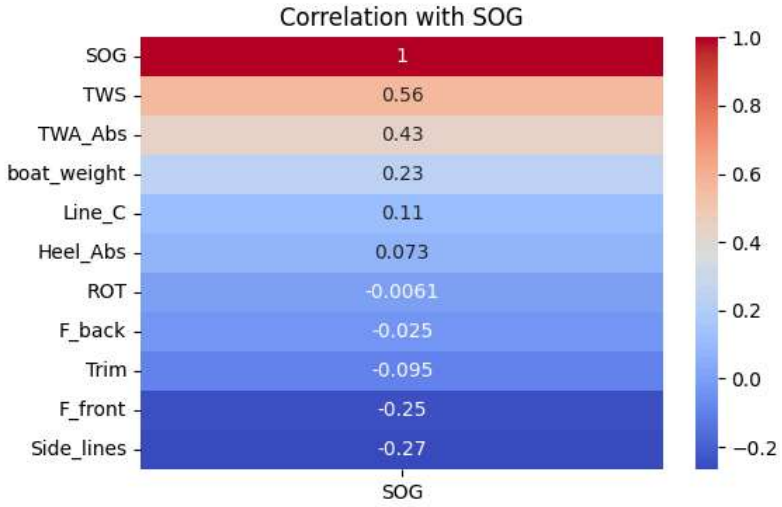
III.3.2 Downwind Slave

In [30]:
```
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: 5534
```

In [31]:
```
full_analysis(df_numeric_slave_downwind)
```

```
Correlation with SOG:
```



ANOVA:

	sum_sq	df	F	PR(>F)
TWS	348.757145	1.0	385.778206	4.032141e-82
Side_lines	238.573968	1.0	263.898929	1.534198e-57
Line_C	163.060321	1.0	180.369403	2.885847e-40
boat_weight	67.939673	1.0	75.151565	6.205239e-18
Trim	55.352436	1.0	61.228174	6.437538e-15
Heel_Abs	46.431791	1.0	51.360591	9.088353e-13
F_back	30.010050	1.0	33.195658	8.952174e-09
TWA_Abs	16.079200	1.0	17.786029	2.525872e-05
F_front	7.832009	1.0	8.663388	3.265090e-03
ROT	0.131462	1.0	0.145417	7.029743e-01
Residual	3665.863451	4055.0	NaN	NaN

```
Polynomial fit:
R²: 0.344
Used samples: 4066

 Formula :
SOG ≈ 27.178 +
  0.147 * Heel_Abs +
  0.739 * Line_C +
  -0.006 * ROT +
  0.871 * Side_lines +
  -0.145 * Trim +
  0.083 * TWA_Abs +
  0.464 * TWS +
  -0.246 * boat_weight +
  -0.072 * F_front +
  0.139 * F_back
```

	feature	coefficient
3	Side_lines	0.870608
1	Line_C	0.739499
6	TWS	0.463966
7	boat_weight	-0.246087
0	Heel_Abs	0.147356
4	Trim	-0.145263
9	F_back	0.139004
5	TWA_Abs	0.083093
8	F_front	-0.071689
2	ROT	-0.005988

III.3.3. Downwind: Master vs Slave t_test

In [32]:
```
t_test(df_numeric_master_downwind,df_numeric_slave_downwind)
```

```
T-statistic: 9.267, p-value: 0.000000000000000
The difference is statistically significant, keeping data split.
```