In [15]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
import seaborn as sns; sns.set_theme(color_codes=True)
```

1. Plots and regression planes

Given a dataset of vehicles we are trying to visualize and establish relationships between horsepower, prices and fuel efficiencies and assess if linear regression is a suitable analysis method.

In [16]:
```
df = pd.read_csv("imports-85.csv").dropna()
log_price = np.log(df["price"] - df["price"].min() + 1)
sq_price = df["price"] ** 2
df
```

Out[16]:

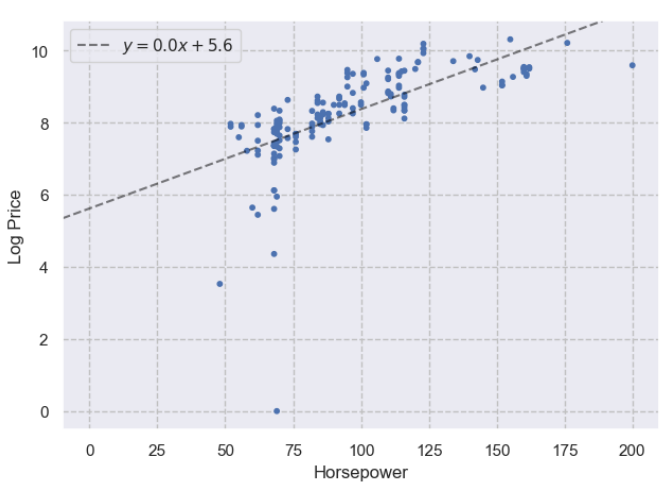
	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	f sys
3	2	164.0	audi	gas	std	four	sedan	fwd	front	99.8	...	109	
4	2	164.0	audi	gas	std	four	sedan	4wd	front	99.4	...	136	
6	1	158.0	audi	gas	std	four	sedan	fwd	front	105.8	...	136	
8	1	158.0	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	
10	2	192.0	bmw	gas	std	two	sedan	rwd	front	101.2	...	108	
...	...	...	...	...	...	...	...	...	...	...	...	...	
200	-1	95.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	
201	-1	95.0	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	
202	-1	95.0	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	
203	-1	95.0	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	
204	-1	95.0	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	

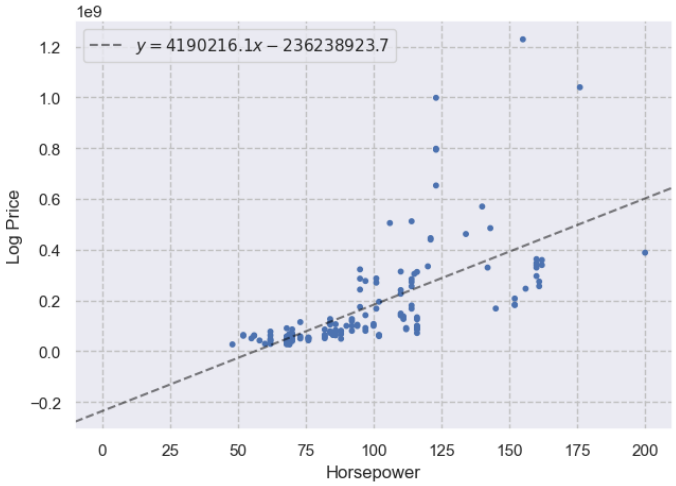
159 rows × 26 columns

In [23]:
```
plt.figure(figsize=(16,5))
plt.title("Plotting horsepower against price")
plt.subplot(1,2,1)
plt.grid(linestyle="--", color="silver")
plt.xlabel("Horsepower")
plt.ylabel("Log Price")
reg1 = sm.OLS(log_price, sm.add_constant(df["horsepower"])).fit(cov="HC0")
b, m = reg1.params
plt.axline(xy1=(0, b), slope=m, label=f'$y = {m:.1f}x {b:+.1f}$', alpha=0.5, color="blac
plt.scatter(df["horsepower"], log_price, s=[10 for i in range(len(df))])
plt.legend()

plt.subplot(1,2,2)
plt.grid(linestyle="--", color="silver")
plt.xlabel("Horsepower")
plt.ylabel("Log Price")
reg2 = sm.OLS(sq_price, sm.add_constant(df["horsepower"])).fit(cov="HC0")
```

```
b, m = reg2.params
plt.axline(xy1=(0, b), slope=m, label=f'$y = {m:.1f}x {b:+.1f}$', alpha=0.5, color="blac
plt.scatter(df["horsepower"], sq_price, s=[10 for i in range(len(df))])
plt.legend()
```

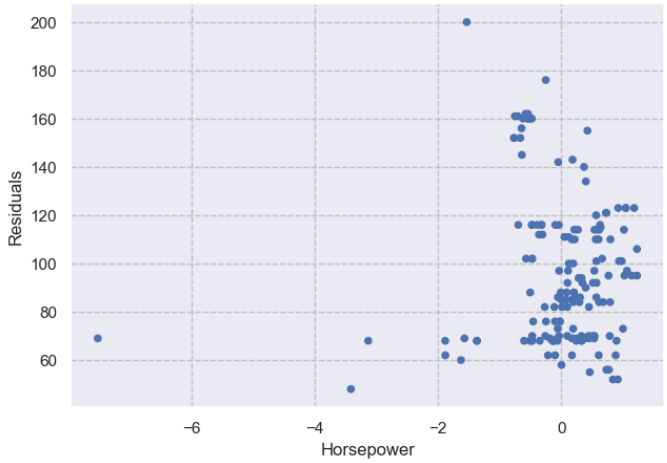
Out[23]:
```
<matplotlib.legend.Legend at 0x2014102d340>
```

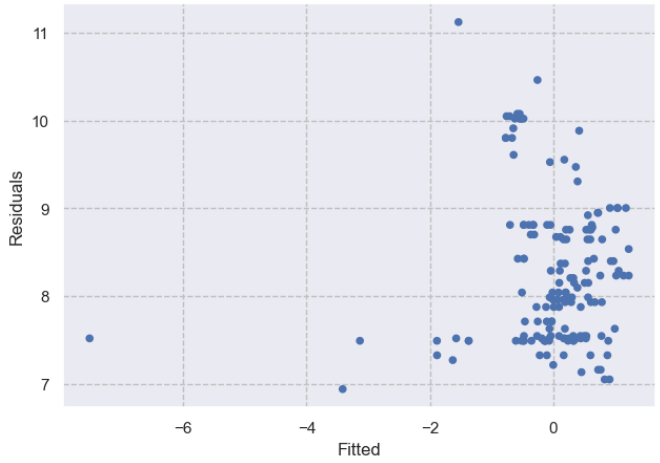




In [24]:
```
plt.figure(figsize=(16,5))
plt.title("Regression diagnostics")
plt.subplot(1,2,1)
plt.grid(linestyle="--", color="silver")
plt.xlabel("Horsepower")
plt.ylabel("Residuals")
plt.scatter(reg1.resid, df["horsepower"], s=[20 for i in range(len(df))])

plt.subplot(1,2,2)
plt.grid(linestyle="--", color="silver")
plt.xlabel("Fitted")
plt.ylabel("Residuals")
plt.scatter(reg1.resid, reg1.fittedvalues, s=[20 for i in range(len(df))])
```

Out[24]:
```
<matplotlib.collections.PathCollection at 0x20141124e80>
```



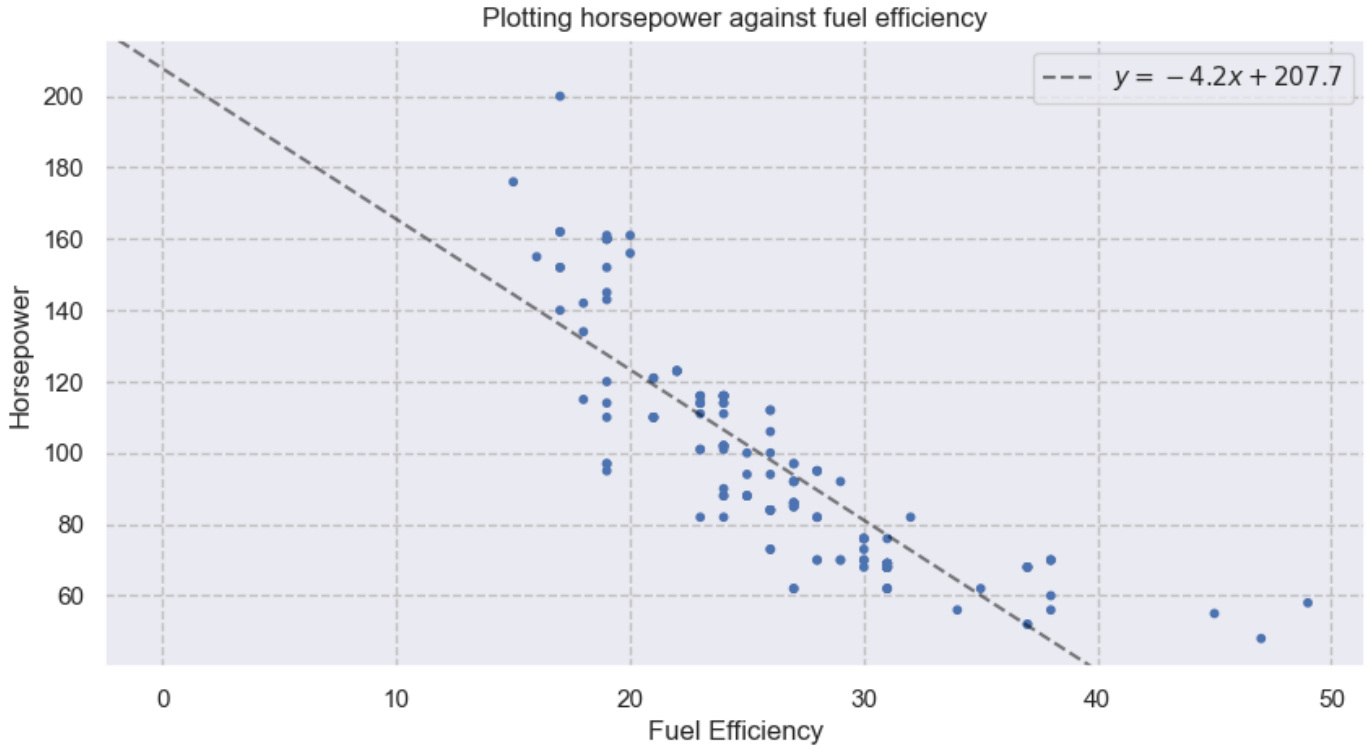


There is very strong correlation between residuals and horsepower despite using robust standard errors. This is an indication of heteroskedasticty and indicates that horsepower and prices dont follow a linear relationship.

In [25]:
```
plt.figure(figsize=(10,5))
plt.title("Plotting horsepower against fuel efficiency")
plt.xlabel("Fuel Efficiency")
plt.ylabel("Horsepower")
plt.grid(linestyle="--", color="silver")
```

```
reg3 = sm.OLS(df["horsepower"], sm.add_constant(df["city-mpg"])).fit(cov="HC0")
b, m = reg3.params
plt.axline(xy1=(0, b), slope=m, label=f'$y = {m:.1f}x {b:+.1f}$', alpha=0.5, color="blac
plt.scatter(df["city-mpg"], df["horsepower"], s=[10 for i in range(len(df))])
plt.legend()
```

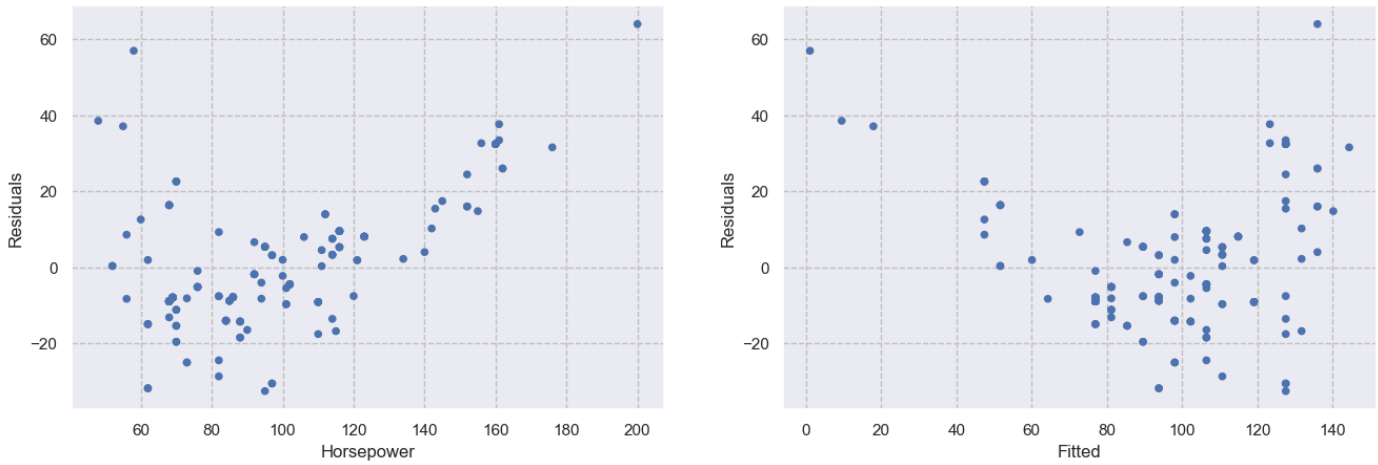
Out[25]:
```
<matplotlib.legend.Legend at 0x20140e6b880>
```



A linear trend is noticable in between horsepower and fuel efficiency.

In [7]:
```
plt.figure(figsize=(16,5))
plt.title("Regression diagnostics")
plt.subplot(1,2,1)
plt.grid(linestyle="--", color="silver")
plt.xlabel("Horsepower")
plt.ylabel("Residuals")
plt.scatter(df["horsepower"], reg3.resid, s=[20 for i in range(len(df))])

plt.subplot(1,2,2)
plt.grid(linestyle="--", color="silver")
plt.xlabel("Fitted")
plt.ylabel("Residuals")
plt.scatter(reg3.fittedvalues, reg3.resid, s=[20 for i in range(len(df))])
```

Out[7]:
```
<matplotlib.collections.PathCollection at 0x2013f604dc0>
```



There is no apparent correlation between residuals and horsepower. However there is some indication of heteroskedasticity. Horsepower and fuel efficiency could be modelled using regression.

2. Deciles

Given a stock return data we will divide the dataset into deciles based on log(Issue). We will perform value weighted aggregation over excess returns for each year and run some heuristics around the calculations.

In [8]:
```
df = pd.read_csv("StockRetAcct_DT.csv", index_col=[0]).dropna()
df["ex_ret"] = df["lnAnnRet"] - df["lnRf"]
grp_yr = df.groupby("year")
grp_yr.first()
```

Out[8]:

year	FirmID	lnAnnRet	lnRf	MEwt	lnIssue	lnMom	lnME	lnProf	lnEP	lnInv	lnI
1980	6	0.363631	0.078944	0.000281	0.031344	0.075355	12.581472	0.201767	0.146411	0.093626	0.69
1981	6	-0.290409	0.130199	0.000321	0.044213	0.512652	12.907996	0.215661	0.102555	0.087242	0.70
1982	6	0.186630	0.130703	0.000266	-0.068195	-0.220505	12.557775	0.184087	0.119548	0.111663	0.73
1983	6	0.489819	0.089830	0.000170	-0.071780	0.046218	12.561954	0.165531	0.115924	-0.033117	0.71
1984	50	-0.003886	0.112608	0.000072	0.340671	-0.120749	11.572387	-0.171332	-0.355613	-0.195077	0.70
1985	50	-0.321691	0.073987	0.000054	0.364280	-0.387309	11.543515	-0.093638	-0.154869	0.158610	0.65
1986	120	-0.030707	0.062209	0.000891	0.048186	0.190390	14.617939	0.202912	0.117003	0.082769	0.83
1987	120	0.011290	0.066003	0.000709	0.034266	0.006250	14.528989	0.245821	0.095697	0.034062	0.77
1988	15	0.141739	0.072406	0.000193	0.160109	0.108250	13.123762	0.239861	0.119085	0.120877	0.54
1989	15	-0.018070	0.078056	0.000207	-0.012286	0.032554	13.305965	0.237256	0.100771	0.432045	0.84
1990	15	-0.372159	0.076470	0.000164	-0.010896	-0.025881	13.148136	0.282474	0.134671	0.005042	1.08
1991	10	-0.508005	0.061216	0.000033	0.115204	1.341053	11.565831	0.239788	0.023147	0.300051	0.41
1992	15	-0.027066	0.039831	0.000099	-0.014229	0.401821	12.826307	0.165353	0.103128	0.040362	0.94
1993	15	-0.520980	0.034398	0.000073	0.007545	-0.169992	12.668312	0.148366	0.085650	0.003218	0.91
1994	15	-0.160343	0.052850	0.000042	0.005065	-0.370789	12.148072	-0.433071	-0.652975	-0.108707	1.49
1995	53	-0.278713	0.054031	0.000033	0.532959	0.101678	12.132327	0.828802	0.079983	0.052285	2.72
1996	15	-0.095310	0.054900	0.000037	-0.002631	0.561892	12.518220	-0.005844	-0.088492	-0.365402	2.00
1997	15	0.061694	0.054563	0.000027	0.010633	-0.272415	12.419437	0.308843	0.102915	0.014336	1.61
1998	15	0.274779	0.051995	0.000022	0.285650	0.184075	12.517771	0.083038	-0.010618	-0.082505	1.56
1999	15	-0.113329	0.050487	0.000023	0.215107	0.226922	12.765973	0.319904	-0.028912	-0.265644	1.56
2000	12	-1.356847	0.061977	0.000012	0.165238	0.251746	12.275754	-0.382327	-0.023783	-0.174606	0.82
2001	22	0.004517	0.038043	0.000019	0.077566	1.786021	12.524199	0.340062	-0.011976	-0.087827	2.18
2002	23	-0.358965	0.019225	0.000034	-0.016249	0.219182	12.895978	0.266932	0.054609	0.019986	0.36
2003	28	0.161213	0.010340	0.000043	0.039805	-0.590056	13.094610	0.093007	-0.011073	-0.031322	0.30
2004	28	0.052663	0.019921	0.000043	0.034062	0.286738	13.273294	0.032173	-0.097354	-0.054351	0.39

2005	28	0.877166	0.034307	0.000043	0.078836	-0.097535	13.332599	0.110067	-0.057831	-0.013372	0.44
2006	28	-0.397459	0.050748	0.000102	0.073469	1.188965	14.273145	0.136010	-0.012474	0.095625	0.56
2007	28	0.185651	0.048189	0.000060	-0.022319	-0.624053	13.879166	0.188793	0.086669	0.288728	0.50
2008	23	0.280786	0.023388	0.000034	0.000653	-0.237225	13.145796	0.210173	0.053417	0.109610	0.18
2009	23	0.169272	0.005498	0.000061	-0.008651	0.308894	13.402636	0.187824	0.041576	0.071338	0.18
2010	23	0.178812	0.003067	0.000063	-0.004938	0.230481	13.564600	0.226684	0.054740	0.074115	0.17
2011	23	0.179764	0.001880	0.000059	0.025942	0.188689	13.743263	0.230488	0.052707	0.095691	0.16
2012	23	0.283810	0.002083	0.000071	0.021170	0.003521	13.926254	0.203738	0.053899	0.129328	0.15
2013	23	0.202738	0.001553	0.000079	0.006276	0.303075	14.193411	0.201279	0.044114	0.090589	0.14
2014	23	0.175317	0.001175	0.000077	-0.009054	0.213144	14.377582	0.208677	0.038164	0.068284	0.13

In [9]:
```
decile_returns = [0 for i in range(10)]
for name, group in grp_yr:

    # decile creation based on lnIssue
    group["decile"] = pd.qcut(group["lnIssue"], q=10, labels=[(i+1)*10 for i in range(10

    # assignments for fama-mcbeth regression
    group["transIssue"] = 0
    group.loc[group["decile"]==10, "transIssue"] = -1
    group.loc[group["decile"]==90, "transIssue"] = 1
    df.loc[df["year"]==name, "transIssue"] = group["transIssue"]

    # in every decile portfolio each year we calculate total portfolio value weighted re
    grp_dec = group.groupby("decile")
    i = 0
    for dec_name, dec_group in grp_dec:
        dec_group["wtd_ret"] = dec_group["MEwt"] * dec_group["ex_ret"] / dec_group["MEwt
        decile_returns[i] += dec_group["wtd_ret"].sum()
        i = i+1
decile_returns = [np.exp(x/len(grp_yr)) for x in decile_returns]
print("Value weighted average Decile Returns across years 1980 to 2014 ranging from 10%
```

```
Value weighted average Decile Returns across years 1980 to 2014 ranging from 10% percent
ile to 100% percentile in order:
 [1.070461331012063, 1.037649561385707, 1.0338476837004964, 1.0433341984048485, 1.044605
009520356, 1.018953389450726, 1.014051630271287, 0.9855606837565917, 0.9949337768900871,
0.9580212172770405]
```

In [10]:
```
ax = sns.regplot(x=[(i+1)*10 for i in range(10)], y=decile_returns)
ax.set(xlabel='Portfolio Percentile', ylabel='Value weighted Avg returns', title='Decile
```

Out[10]:
```
[Text(0.5, 0, 'Portfolio Percentile'),
 Text(0, 0.5, 'Value weighted Avg returns'),
 Text(0.5, 1.0, 'Decile Returns Plot')]
```

The pattern seems fairly linear with no apparent autocorrelation or heteroskedasticity.

Fama Mcbeth

Assuming that the most scope for returns is in the uppermost and lowermost decile, we will run fama mcbeth regression to find which stocks to long and short.

In [11]:
```
grp_stock = df.groupby("FirmID")
fama_mcbeth = {}
flag = True
for name, group in grp_stock:
   #fama mcbeth regression

    reg = sm.OLS(group["ex_ret"], sm.add_constant(group["transIssue"])).fit(cov="HC0")
    if (len(reg.params) == 2):
        c,beta = reg.params
        fama_mcbeth[name] = beta
    else:
        fama_mcbeth[name] = reg.params[0]
```

In [12]:
```
fm_df = pd.DataFrame()
fm_df["Stock ID"] = fama_mcbeth.keys()
fm_df["Fama McBeth Beta"] = fama_mcbeth.values()
fm_df.loc[fm_df["Fama McBeth Beta"]>0,"Position"] = "Long"
fm_df.loc[fm_df["Fama McBeth Beta"]<0,"Position"] = "Short"
fm_df.loc[fm_df["Fama McBeth Beta"]==0,"Position"] = "No Position"
fm_df
```

Out[12]:

	Stock ID	Fama McBeth Beta	Position
0	6	-0.295918	Short
1	10	0.000000	No Position

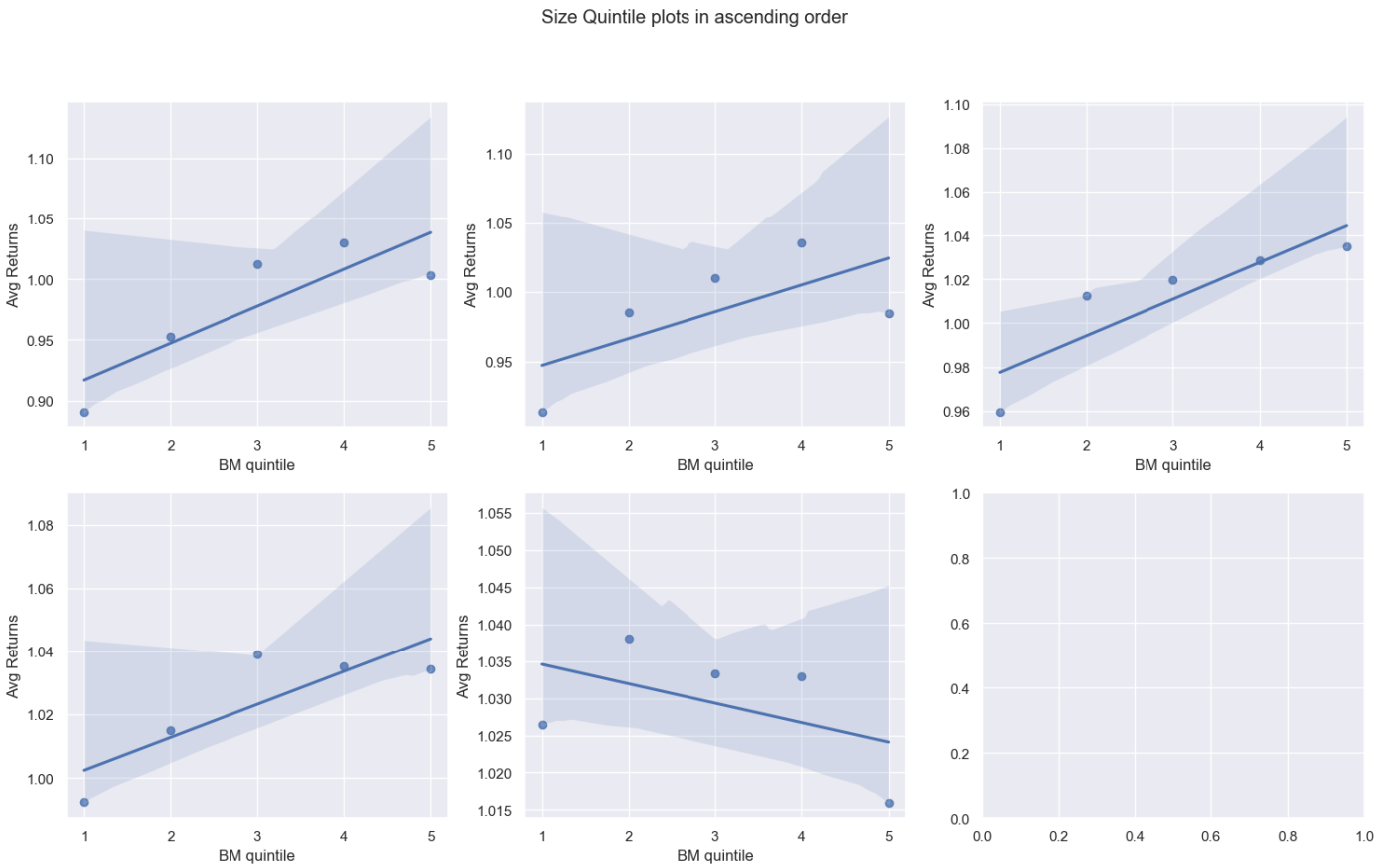
2	12	0.000000	No Position
3	15	0.000379	Long
4	19	0.085891	Long
...	...	...	...
6651	20304	0.000000	No Position
6652	20305	-0.215558	Short
6653	20307	-0.161059	Short
6654	20308	0.000000	No Position
6655	20314	-0.907084	Short

6656 rows × 3 columns

3. Quintiles

The dataset will be grouped based into tow types of quintiles based on book to market and company size. We will use seaborn based visualization tools to assess the relationship of returns with the quintiles.

In [13]:
```
qt_df = pd.DataFrame().from_dict({"Size quintile": [], "BM quintile": []})

# initializing quintiles
for i in range(5):
    for j in range(5):
        qt_df.loc[len(qt_df)] = [i+1, j+1]

qt_df["Avg Returns"] = 0
for name, group in grp_yr:
    # quintile creation based on book to market
    group["quintile_BM"] = pd.qcut(group["lnBM"], q=5, labels=[(i+1) for i in range(5)])
    # quintile creation based on size
    group["quintile_size"] = pd.qcut(group["lnME"], q=5, labels=[(i+1) for i in range(5)

    grp_qtsize = group.groupby("quintile_size")

    for name2, group2 in grp_qtsize:
        grp_qtbm = group2.groupby("quintile_BM")
        for name3, group3 in grp_qtbm:
            group3["wtd_ret"] = group3["MEwt"] * group3["ex_ret"] / group3["MEwt"].sum()
            qt_df.loc[(qt_df["Size quintile"]==name2) & (qt_df["BM quintile"]==name3), "


qt_df["Avg Returns"] = np.exp(qt_df["Avg Returns"]/len(grp_yr))
grp_obj = qt_df.groupby("Size quintile")
row, col = [0,0]
fig, axes = plt.subplots(2, 3, figsize=(18, 10))
fig.suptitle('Size Quintile plots in ascending order')
for name, group in grp_obj:
    sns.regplot(ax=axes[row, col], data=group, x='BM quintile', y='Avg Returns')
    col += 1
    if col == 3:
        row += 1
        col = 0
```



Assumption of conditional linearity seems to be violated due to very lucid presence of heteroskedasticity hence a different model would be preferable.