Applying the Fama-French 5 Factor Model to ARRK Inovation Fund

The Fama French 5 Factor Model is an emperical asset-pricing model, and is an extension of the CAPM model. In additon to the beta of an investment and the market risk premium, the FF 5 factor model adds SMB, HML, RMW, and CMA to the equation to better determine expected returns of a given investment. The new factors are defined below. Tests done by Fama and French in 2014 indicate that the Fama-French 5 factor model explains between 71% and 94% of the cross-section variance of expected returns for the size,value,profitability, and investment patterns in average stock returns. In the analysis below, I apply the model to Cathie Wood's ARRK Inovation fund over a period beginning November, 2015 and ending November, 2021.

Rit — RFt = αi + βi(RMt — RFt) + siSMBt + hiHMLt + riRMWt + ciCMAt + eit

- Rft is the return in month t of one of the portfolios
- RFt is the riskfree rate
- Rm - Rf is the return spread between the capitalization-weighted stock market and cash
- SMB is the return spread of small minus large stocks (i.e. the size effect)
- HML is the return spread of cheap minus expensive stocks (i.e. the value effect)
- RMW is the return spread of the most profitable firms minus the least profitable
- CMA is the return spread of firms that invest conservatively minus aggressively (AQR, 2014)

Analysis by Calvin Crosby

In [554]:
```
# Importing required libraries
import pandas as pd
import pandas_datareader.data as reader
import datetime as dt
import statsmodels.api as sm
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sorcery import dict_of
```

In [659]:
```
# Establishing the beginning and end of our historical period
end = dt.date(2021,11,30)
start = dt.date(2014,12,30)

aark_returns = pd.read_csv(r"C:/Users/crosb/OneDrive/Desktop/finance_data/ARKK.csv",index_col=False)
aark_returns['Date'] = pd.to_datetime(aark_returns['Date'],format = '%Y-%m-%d')
aark_returns = aark_returns[~(aark_returns['Date'] > pd.Timestamp(end))]
aark_returns['ARRK'] = aark_returns['Adj Close'].pct_change()

fama_french_factors = pd.read_csv(r"C:/Users/crosb/OneDrive/Desktop/finance_data/factors.csv")
fama_french_factors['Date'] = pd.to_datetime(fama_french_factors['Date'],format = '%Y%m')
fama_french_factors = fama_french_factors[~(fama_french_factors['Date'] <= pd.Timestamp(start))]
```

After reading the csv file into Pandas, let's store it in a varible and examine the data frame

In [508]:
```
aark_returns.head()
```

Out[508]:

	Date	Open	High	Low	Close	Adj Close	Volume	ARRK
0	2015-01-26	20.750000	20.750000	20.750000	20.750000	18.881573	200	NaN
1	2015-01-27	20.440001	20.440001	20.440001	20.440001	18.599487	1200	-0.014940
2	2015-01-28	20.440001	20.440001	20.350000	20.350000	18.517593	300	-0.004403
3	2015-01-29	20.334000	20.389999	20.080000	20.389999	18.553991	900	0.001966
4	2015-01-30	20.420000	20.420000	20.240000	20.240000	18.417500	1900	-0.007356

Before running ARKK's returns through the model, we resample the data on a monthly basis rather than daily. The sum function is used to aggregate the data together. The index of the DataFrame is set to the Date column, so merging can be done with the Fama French table

In [660]:
```
cols = ['Date','ARRK'] # These are the desired cols from the DataFrame
aark_change = aark_returns[cols]
aark_change.Date = pd.to_datetime(aark_change.Date)
date_col = aark_change['Date']

aark_change.set_index('Date', inplace=True)
aark_change = aark_change.resample("MS").sum()
```

```
C:\Users\crosb\AppData\Local\Continuum\anaconda3\lib\site-packages\pandas\core\generic.py:5096: Setti
ngWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#index
ing-view-versus-copy
  self[name] = value
```

In [661]:
```
aark_change.head()
```

Out[661]:

Date	ARRK
2015-01-01	-0.024734
2015-02-01	0.061510
2015-03-01	-0.030981
2015-04-01	0.007315
2015-05-01	0.030609

In [408]:
```
fama_french_factors.set_index('Date', inplace=True)
fama_french_factors.head()
```

Out[408]:

Date	Mkt-RF	SMB	HML	RMW	CMA	RF
2015-01-01	-3.11	-0.90	-3.61	1.64	-1.68	0.0
2015-02-01	6.13	0.30	-1.86	-1.12	-1.78	0.0
2015-03-01	-1.12	3.10	-0.41	0.15	-0.54	0.0
2015-04-01	0.59	-3.07	1.83	0.01	-0.64	0.0
2015-05-01	1.36	0.82	-1.12	-1.77	-0.72	0.0

Our tables are ready for merging. Let's confirm they're all the same length. We should have 83 rows in our tables, one for each month in the historical period

In [390]:
```
print(aark_change.shape)
print(fama_french_factors.shape)
```

```
(83, 1)
(83, 1)
(83, 6)
```

In [409]:
```
aark_change.index = fama_french_factors.index
```

In [512]:
```
merge = pd.merge(aark_change,fama_french_factors,on='Date')
```

Below is the merged DataFrame containing the monthly returns for ARRK alongside the 5 factors of the model and the risk free rate. One final step before running the multivariate linear regression is to ensure each column in the data frame is on the same scale. If you look below, you can see that the 5 factors are stored as percentages, so we must divide each of those factor columns by 100.

In [513]:
```
merge.head()
```

Out[513]:

	Date	ARRK	Mkt-RF	SMB	HML	RMW	CMA	RF
0	2015-01-01	-0.024734	-3.11	-0.90	-3.61	1.64	-1.68	0.0
1	2015-02-01	0.061510	6.13	0.30	-1.86	-1.12	-1.78	0.0
2	2015-03-01	-0.030981	-1.12	3.10	-0.41	0.15	-0.54	0.0
3	2015-04-01	0.007315	0.59	-3.07	1.83	0.01	-0.64	0.0
4	2015-05-01	0.030609	1.36	0.82	-1.12	-1.77	-0.72	0.0

In [514]:
```
merge[['Mkt-RF', 'SMB', 'HML','RMW','CMA','RF']] = merge[['Mkt-RF', 'SMB', 'HML','RMW','CMA','RF']]/100
```

In [515]:
```
merge['AARK-RF'] = merge['ARRK'] - merge['RF']
```

Our merged table is ready! All values are on the same scale and an additional column has been added which is the difference between AARK's returns and the risk free rate. This is what the dependant variable will be in the model, which you can see again in the below equation.

In [516]:
```
merge.head()
```

Out[516]:

	Date	ARRK	Mkt-RF	SMB	HML	RMW	CMA	RF	AARK-RF
0	2015-01-01	-0.024734	-0.0311	-0.0090	-0.0361	0.0164	-0.0168	0.0	-0.024734
1	2015-02-01	0.061510	0.0613	0.0030	-0.0186	-0.0112	-0.0178	0.0	0.061510
2	2015-03-01	-0.030981	-0.0112	0.0310	-0.0041	0.0015	-0.0054	0.0	-0.030981
3	2015-04-01	0.007315	0.0059	-0.0307	0.0183	0.0001	-0.0064	0.0	0.007315
4	2015-05-01	0.030609	0.0136	0.0082	-0.0112	-0.0177	-0.0072	0.0	0.030609

In [517]:
```
y = merge['AARK-RF'] #The dependent variable, returns of AARK less the risk free rate
X = merge[['Mkt-RF', 'SMB', 'HML','RMW','CMA']] #The independent variables, the 5 Fama French factors

X_sm = sm.add_constant(X)
```

```
C:\Users\crosb\AppData\Local\Continuum\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:2495: Fu
tureWarning: Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp instea
d.
  return ptp(axis=axis, out=out, **kwargs)
```

Rit — RFt = ai + bi(RMt — RFt) + siSMBt + hiHMLt + riRMWt + ciCMAt + eit

- Rft is the return in month t of one of the portfolios
- RFt is the riskfree rate
- Rm - Rf is the return spread between the capitalization-weighted stock market and cash
- SMB is the return spread of small minus large stocks (i.e. the size effect)
- HML is the return spread of cheap minus expensive stocks (i.e. the value effect)
- RMW is the return spread of the most profitable firms minus the least profitable
- CMA is the return spread of firms that invest conservatively minus aggressively (AQR, 2014)

In [518]:
```
arrk_model = sm.OLS(y,X_sm) #ordinary least squares regression with y as independent variable and 5 fac
tors as constant
results = arrk_model.fit()
results.summary()
```

Out[518]:

OLS Regression Results

Dep. Variable:	AARK-RF	R-squared:	0.799
Model:	OLS	Adj. R-squared:	0.786
Method:	Least Squares	F-statistic:	61.36
Date:	Wed, 26 Jan 2022	Prob (F-statistic):	1.88e-25
Time:	16:03:06	Log-Likelihood:	153.55
No. Observations:	83	AIC:	-295.1
Df Residuals:	77	BIC:	-280.6
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.0050	0.005	1.093	0.278	-0.004	0.014
Mkt-RF	1.4856	0.116	12.810	0.000	1.255	1.717
SMB	0.5951	0.202	2.946	0.004	0.193	0.997
HML	-0.9129	0.168	-5.426	0.000	-1.248	-0.578
RMW	-0.8079	0.261	-3.092	0.003	-1.328	-0.288
CMA	-0.1957	0.305	-0.642	0.522	-0.802	0.411

Omnibus:	8.607	Durbin-Watson:	2.356
Prob(Omnibus):	0.014	Jarque-Bera (JB):	8.184
Skew:	0.708	Prob(JB):	0.0167
Kurtosis:	3.601	Cond. No.	74.0

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Discussion:

The results of the regression indicate that the model does well in predicting the expected returns of ARRK Inovation. With an R-squared value of almost 0.8, the FF 5 factor model can explain almost 80% ARRK's returns over the period. The coefficients computed for each factor are all statistically significant, with the exception of CMA and alpha. While the ARRK fund does have a positive alpha value of 50 basis points, its P-value isn't statistically significant. As expected with an agressive growth fund like ARRK, it has a high beta of 1.48, nearly 50% higher than the market. The next heaviest weighting comes from HML, or "the value effect". HML represents the spread in returns between high book-to-market stocks (or "value") and low book-to-market (or "growth"). ARRK has a strong negative correlation with HML, meaning it suffers when value outperforms. In other words, for each percentage point increase in value's outperformance, ARRK's return will decrease by 91 basis points all else equal. The two share a near perfect negative correlation. ARRK shares another strong negative correlation with RMW, or the profitability factor. Like HML, RMW computes a spread between two classes of stocks. RMW computes the return spread of the most profitable firms minus the least profitable. ARRK has a RMW coefficient of -0.81, indicating that returns for the fund move against those of profittable firms. The last statistically significant coefficient to mention is SMB, or "the size effect". Small minus big measures the excess return that smaller companies return versus larger companies. The positive SMB coefficient of 0.6 suggests ARRK has a medium correlation with small cap firms. Taken together, ARRK will perform best in an environment that is friendly to fast growing, unprofitable companies with a low book to market ratio. With the high inflation of today's environment, these firms may see a rise in their cost of capital as interest rates rise and their cost of debt grows. This would more adversely affect the companies ARRK invests in, which have projected cash flows deeper into the future than value tilted firms.

In [519]:
```
fit_vals = results.fittedvalues.to_frame()
```

In [654]:
```
# Define function to output plot of the model coefficients
# credit to jessica forest baldini

def coefplot(results):
    '''
    Takes in results of OLS model and returns a plot of
    the coefficients with 95% confidence intervals.

    Removes intercept, so if uncentered will return error.
    '''
    # Create dataframe of results summary
    coef_df = pd.DataFrame(results.summary().tables[1].data)

    # Add column names
    coef_df.columns = coef_df.iloc[0]

    # Drop the extra row with column labels
    coef_df=coef_df.drop(0)

    # Set index to variable names
    coef_df = coef_df.set_index(coef_df.columns[0])

    # Change datatype from object to float
    coef_df = coef_df.astype(float)

    # Get errors; (coef - lower bound of conf interval)
    errors = coef_df['coef'] - coef_df['[0.025']

    # Append errors column to dataframe
    coef_df['errors'] = errors

    # Drop the constant for plotting
    coef_df = coef_df.drop(['const'])

    # Sort values by coef ascending
    coef_df = coef_df.sort_values(by=['coef'])

    ### Plot Coefficients ###

    # x-labels
    variables = list(coef_df.index.values)

    # Add variables column to dataframe
    coef_df['variables'] = variables

    # Set sns plot style back to 'poster'
    # This will make bars wide on plot
    sns.set_context("poster")

    # Define figure, axes, and plot
    fig, ax = plt.subplots(figsize=(15, 10))

    # Error bars for 95% confidence interval
    # Can increase capsize to add whiskers
    coef_df.plot(x='variables', y='coef', kind='bar',
                 ax=ax, color='none', fontsize=22,
                 ecolor='steelblue',capsize=15,
                 yerr='errors', legend=False)

    # Set title & labels
    plt.title('Coefficients of Fama French Factors w/ 95% Confidence Intervals',fontsize=25)
    ax.set_ylabel('Coefficients',fontsize=22)
    ax.set_xlabel('',fontsize=22)

    # Coefficients
    ax.scatter(x=pd.np.arange(coef_df.shape[0]),
               marker='o', s=80,
               y=coef_df['coef'], color='steelblue')
    #Added labels for coefficients
    num=0
    for index, row in coef_df.iterrows():
        ax.annotate(str(row['coef']), xy=(num+0.05,row['coef']),size=15)
        num+=1


    # Line to define zero on the y-axis
    ax.axhline(y=0, linestyle='--', color='red', linewidth=1)

    return plt.show()
```

Lets add cumulative return columns for ARRK and the regressive model so we can plot them against each other

In [521]:
```
cum_return = merge['ARRK'].sum()
cum_return
merge['Cum_Return'] = merge['ARRK'].cumsum()
merge['OLS'] = fit_vals
merge['Cum_OLS'] = merge['OLS'].cumsum()
merge.reset_index(inplace=True)
merge.head()
```

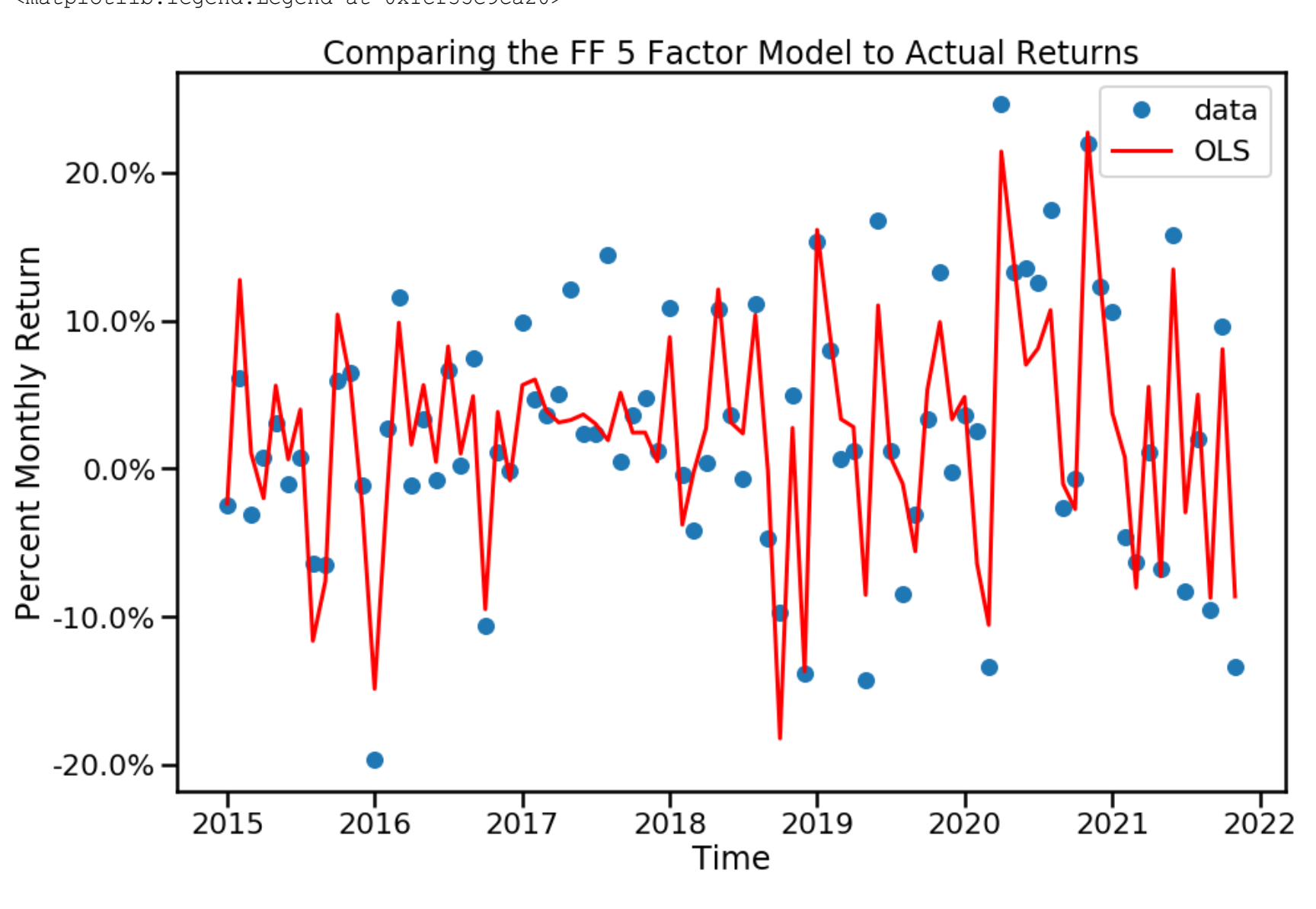
Out[521]:

	index	Date	ARRK	Mkt-RF	SMB	HML	RMW	CMA	RF	AARK-RF	Cum_Return	OLS	Cum_OLS
0	0	2015-01-01	-0.024734	-0.0311	-0.0090	-0.0361	0.0164	-0.0168	0.0	-0.024734	-0.024734	-0.023534	-0.023534
1	1	2015-02-01	0.061510	0.0613	0.0030	-0.0186	-0.0112	-0.0178	0.0	0.061510	0.036776	0.127396	0.103861
2	2	2015-03-01	-0.030981	-0.0112	0.0310	-0.0041	0.0015	-0.0054	0.0	-0.030981	0.005795	0.010425	0.114286
3	3	2015-04-01	0.007315	0.0059	-0.0307	0.0183	0.0001	-0.0064	0.0	0.007315	0.013111	-0.020008	0.094279
4	4	2015-05-01	0.030609	0.0136	0.0082	-0.0112	-0.0177	-0.0072	0.0	0.030609	0.043719	0.056047	0.150326

The graph below is one way to visualize the Model's ability to predict the returns of ARRK over the period. The blue dots represent the actual monthly returns exhibited in the market and the red line represents the Fama-French OLS regressive equation. If the model were perfectly accurate, the line would connect all the blue dots without deviating above or below each one.

In [656]:
```
pred_ols = results.get_prediction()
iv_l = pred_ols.summary_frame()["obs_ci_lower"]
iv_u = pred_ols.summary_frame()["obs_ci_upper"]
x = merge['Date']
y= merge['ARRK']

fig_a, ax_a = plt.subplots(figsize=(15,10))

ax_a.plot(x, y*100, "o", label='data')
ax_a.plot(x, results.fittedvalues*100, "r-", label="OLS")

plt.title("Comparing the FF 5 Factor Model to Actual Returns")
plt.xlabel("Time")
plt.ylabel("Percent Monthly Return")
plt.gca().yaxis.set_major_formatter(plt.FuncFormatter('{}%'.format))

ax.plot(x, iv_u, "y-")
ax.plot(x, iv_l, "y-")
ax_a.legend(loc="best")
```

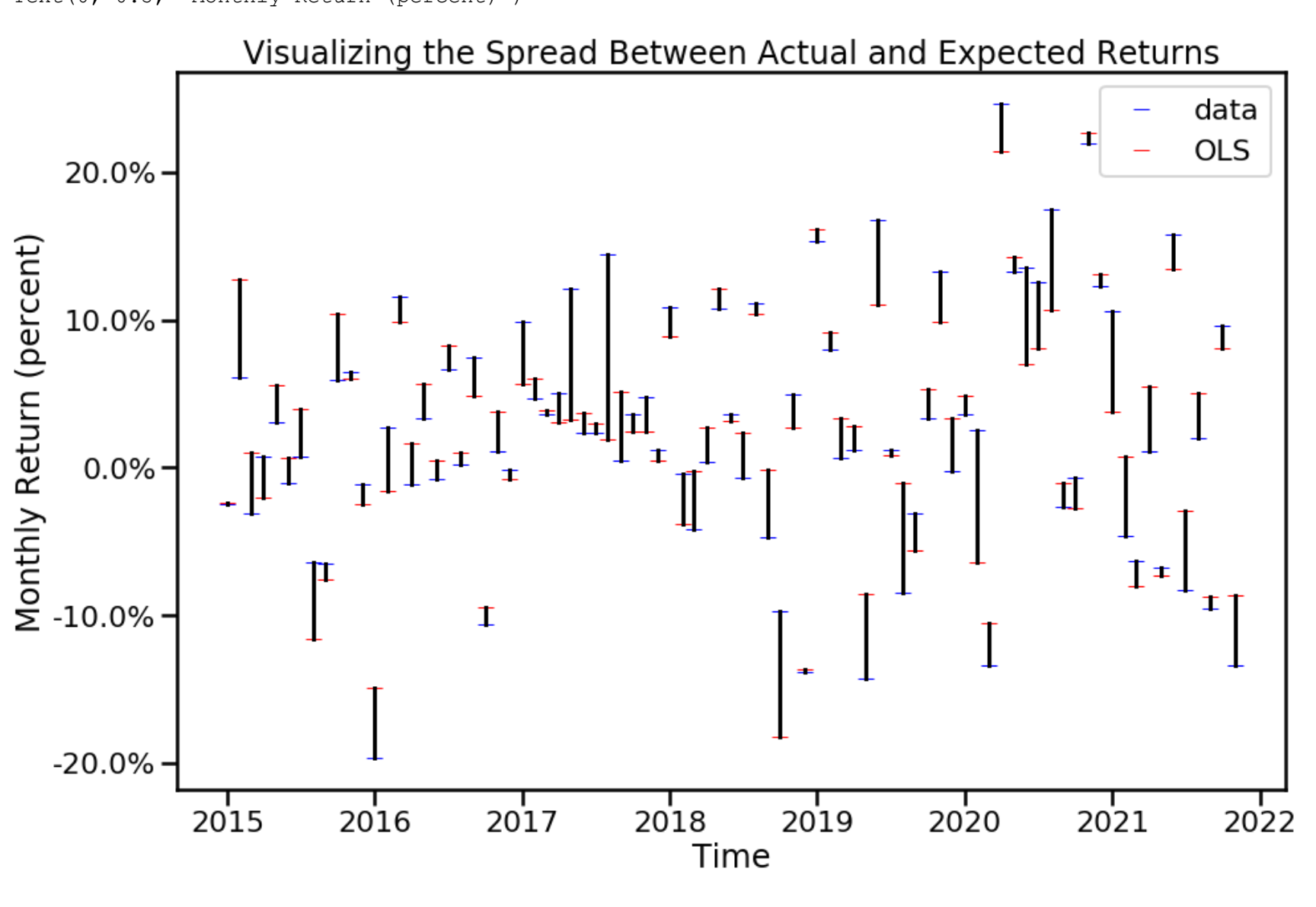
Out[656]: `<matplotlib.legend.Legend at 0x1cf35e9ca20>`



One additional way of visualizing the accuracy of the model is the graph below. Here, the spreads between the actual return for a given month is plotted against the expected return as computed by the model. The shorter the line, the more accurate the model was for that month.

In [501]:
```
pred_ols = results.get_prediction()
iv_l = pred_ols.summary_frame()["obs_ci_lower"]
iv_u = pred_ols.summary_frame()["obs_ci_upper"]
x = merge['Date']
y= merge['ARRK']

fig, ax = plt.subplots(figsize=(15,10))

ax.plot(x, y*100, "b_", label='data')
ax.plot(x, results.fittedvalues*100, "r_", label="OLS")
ax.plot((x,x),([i for i in y*100], [j for j in results.fittedvalues*100]),c='black')
ax.legend(loc="best")

plt.gca().yaxis.set_major_formatter(plt.FuncFormatter('{}%'.format))
plt.title("Visualizing the Spread Between Actual and Expected Returns")
plt.xlabel("Time")
plt.ylabel("Monthly Return (percent)")
```

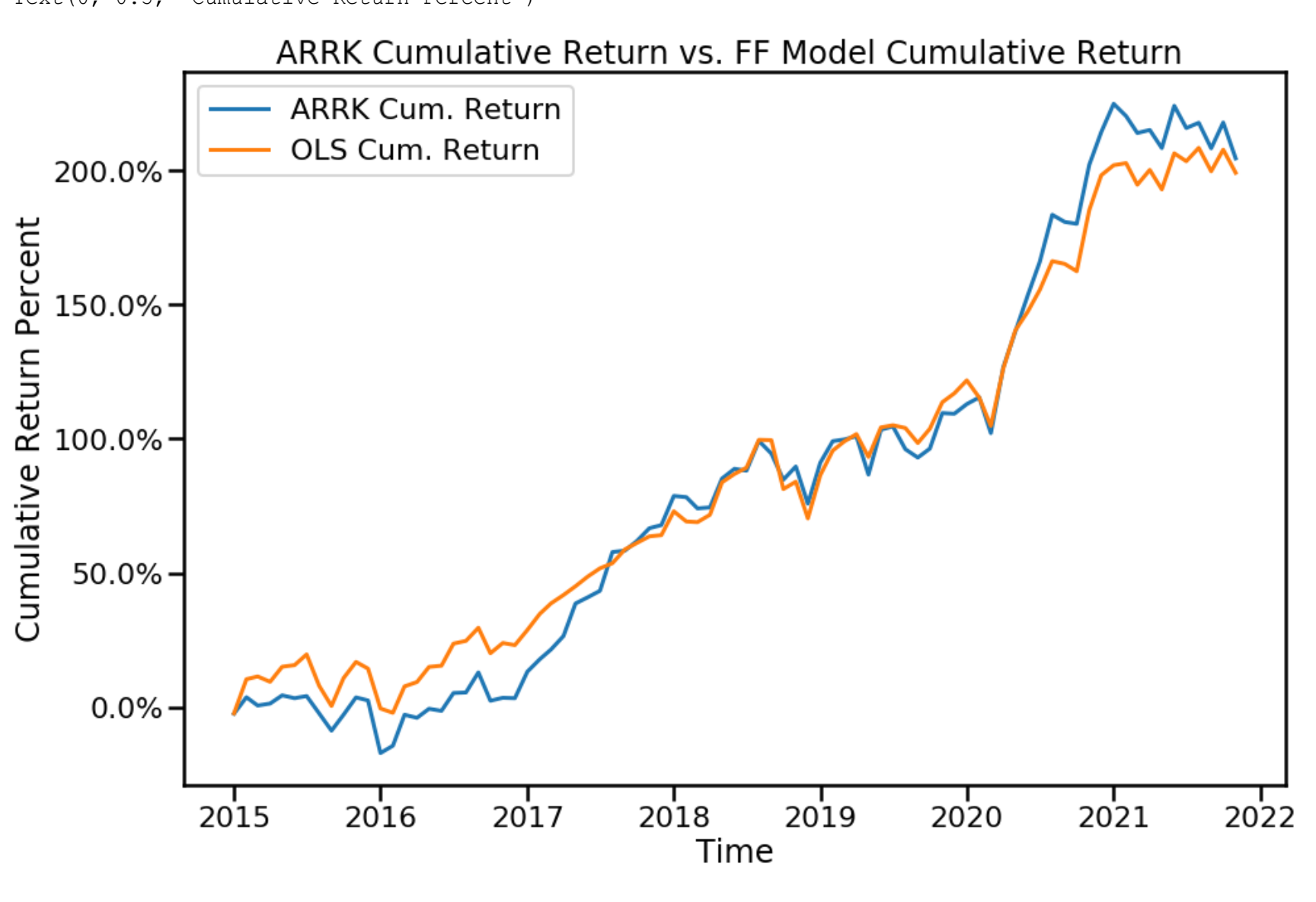
Out[501]: `Text(0, 0.5, 'Monthly Return (percent)')`



ARRK's Cumulative return over the 6 year period (blue) is plotted against the model's cumulative returns. As you can see, the model does a fairly good job at predicting returns of ARRK.

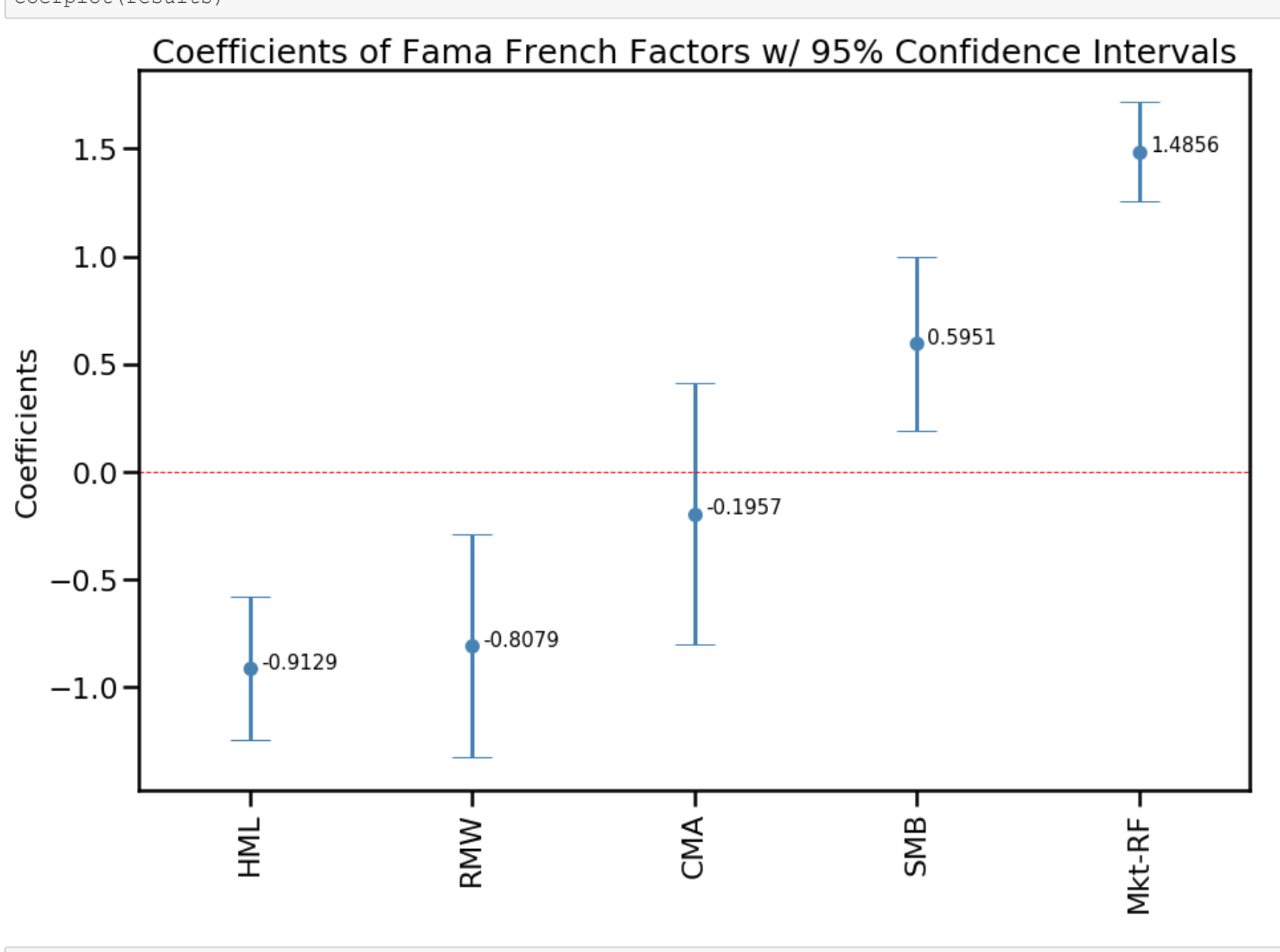
In [657]:
```
fig2, ax2 = plt.subplots(figsize=(15,10))

ax2.plot(x,merge['Cum_Return']*100)
ax2.plot(x,merge['Cum_OLS']*100)
ax2.legend(['ARRK Cum. Return','OLS Cum. Return'])

plt.gca().yaxis.set_major_formatter(plt.FuncFormatter('{}%'.format))
plt.title("ARRK Cumulative Return vs. FF Model Cumulative Return ")
plt.xlabel("Time")
plt.ylabel("Cumulative Return Percent")
```

Out[657]: `Text(0, 0.5, 'Cumulative Return Percent')`



This plot provides a visualization of the factor coefficients generated by the regression. The further a coefficient value deviates from 0, the greater effect it has on ARRK's expected returns. Here we can see the market factor contributes the most, followed by HML in the other direction. Both of these factors have a very strong, statistically significant correlation with ARRK Innovation. The whiskers extending from the points indicate the 95% confidence interval for the given factor.

In [655]:
```
coefplot(results)
```



In [658]:
```
# Extracting factor coefficients
mkt_rf = results.params[1]
smb = results.params[2]
hml = results.params[3]
rmw = results.params[4]
cma = results.params[5]
```

Now we are going to plot sensitivities to individual factors by holding everything else constant. To do this we will create a function to compute expected return using the Fama-French 5 factor equation and the regression coefficients.

Rit — RFt = ai + bi(RMt — RFt) + siSMBt + hiHMLt + riRMWt + ciCMAt + eit

- Mkt-RF = 1.485624
- SMB = 0.595054
- HML = -0.912912
- RMW = -0.807892
- CMA = -0.195699

In [648]:
```
coef = dict_of(mkt_rf,smb,hml,rmw,cma)

def expected_return(factors, coefs = coef):
    '''Given a list of factor values this function will
        compute the expected return using the Fama-French 5 factor model.

        Keyword arguments:
        coef -- a dictionary of fama french factors
        factors -- a list of factors
            factor[0] is Mkt-RF
            factor[1] is SMB
            factor[2] is HML
            factor[3] is RMW
            factor[4] is CMA
            factor[5] is RF'''

    return coef['mkt_rf']*factors[0] + coef['smb']*factors[1] + coef['hml']*factors[2] +coef['rmw']*fac
tors[3] + coef['cma']*factors[4]+ factors[5]
```

In [649]:
```
f = [[0.0486,0.0703,0.0821,-0.0017,0.0370,0.0001],[0.0486,0.0703,0.0821,-0.0017,0.0370,0.0001]]

df = pd.DataFrame(f,columns = ['MKT-RF','SMB','HML','RMW','CMA','RF'])
df['OLS'] = df.apply(lambda row: expected_return([row['MKT-RF'],row['SMB'],row['HML'],row['RMW'],row['C
MA'],row['RF']],coef),axis=1)
df
```

Out[649]:

	MKT-RF	SMB	HML	RMW	CMA	RF	OLS
0	0.0486	0.0703	0.0821	-0.0017	0.037	0.0001	0.033316
1	0.0486	0.0703	0.0821	-0.0017	0.037	0.0001	0.033316

In [650]:
```
df = pd.concat([df]*10,ignore_index=True)
df['HML_Seq'] = df['HML']
df['RMW_Seq'] = df['RMW']
df['MKT_Seq'] = df['MKT-RF']
df['SMB_Seq'] = df['SMB']

#Generating sequences of change by applying equal number
#negative for a positive correlation,positive for a negative correlation
for index,row in df.iterrows():
    row['HML_Seq'] = row['HML'] + 0.005*index
    row['RMW_Seq'] = row['RMW'] + 0.005*index
    row['MKT_Seq'] = row['MKT-RF']-0.005*index
    row['SMB_Seq'] = row['SMB'] - 0.005*index
df['OLS_HML'] = df.apply(lambda row: expected_return([row['MKT-RF'],row['SMB'],row['HML_Seq'],row['RMW'
],row['CMA'],row['RF']],coef),axis=1)
df['OLS_RMW']= df.apply(lambda row: expected_return([row['MKT-RF'],row['SMB'],0.0821,row['RMW_Seq'],row
['CMA'],row['RF']],coef),axis=1)
df['OLS_MKT']= df.apply(lambda row: expected_return([row['MKT_Seq'],row['SMB'],0.0821,row['RMW'],row['C
MA'],row['RF']],coef),axis=1)
df['OLS_SMB']= df.apply(lambda row: expected_return([row['MKT-RF'],row['SMB_Seq'],0.0821,row['RMW'],row
['CMA'],row['RF']],coef),axis=1)
```

In [651]:
```
df.head()
```

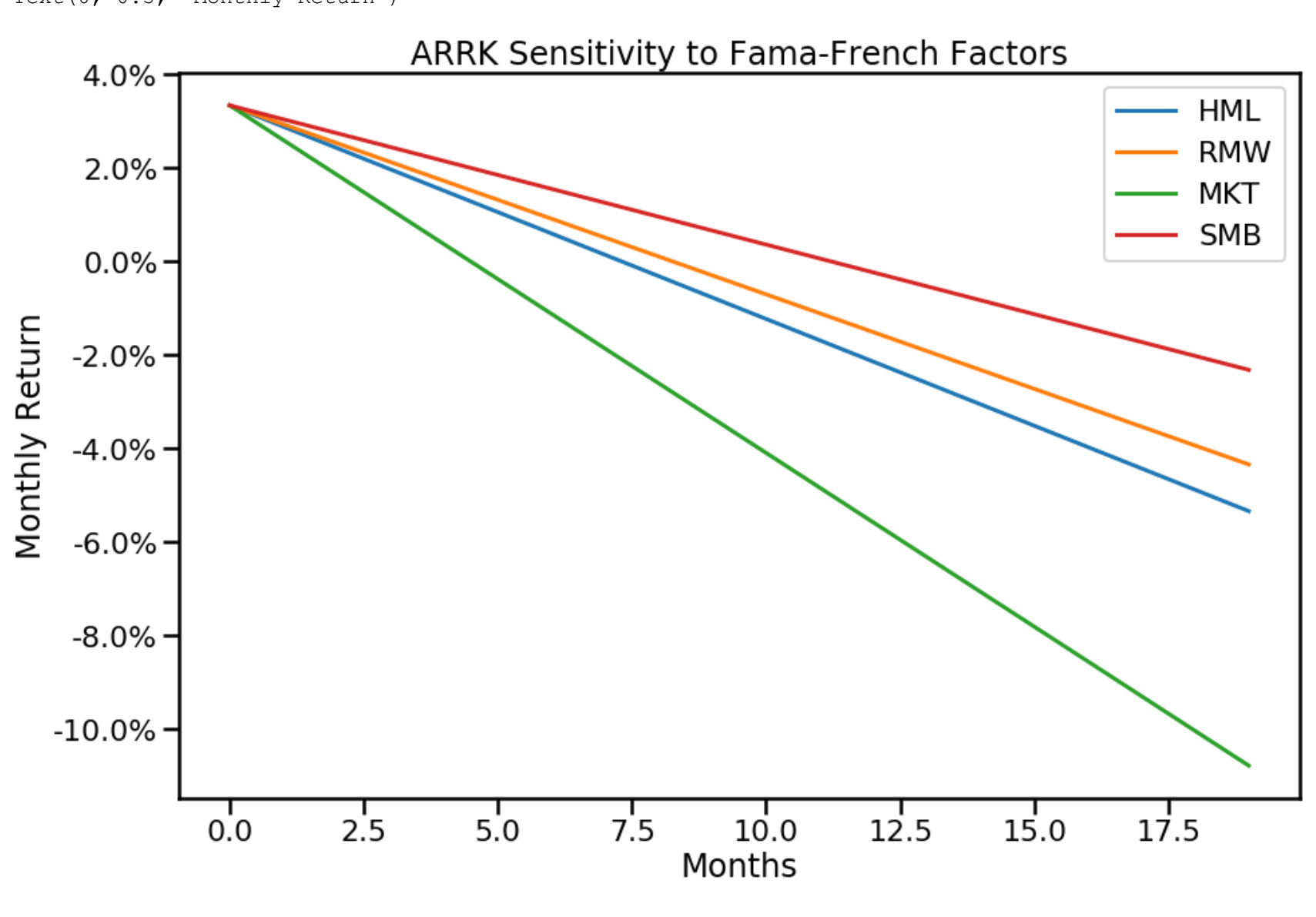
Out[651]:

	MKT-RF	SMB	HML	RMW	CMA	RF	OLS	HML_Seq	RMW_Seq	MKT_Seq	SMB_Seq	OLS_HML	OLS_RMW	OLS_MKT
0	0.0486	0.0703	0.0821	-0.0017	0.037	0.0001	0.033316	0.0821	-0.0017	0.0486	0.0703	0.033316	0.033316	0.033316
1	0.0486	0.0703	0.0821	-0.0017	0.037	0.0001	0.033316	0.0871	0.0033	0.0436	0.0653	0.028752	0.029277	0.025888
2	0.0486	0.0703	0.0821	-0.0017	0.037	0.0001	0.033316	0.0921	0.0083	0.0386	0.0603	0.024187	0.025237	0.018460
3	0.0486	0.0703	0.0821	-0.0017	0.037	0.0001	0.033316	0.0971	0.0133	0.0336	0.0553	0.019622	0.021198	0.011032
4	0.0486	0.0703	0.0821	-0.0017	0.037	0.0001	0.033316	0.1021	0.0183	0.0286	0.0503	0.015058	0.017158	0.003604

The plot below shows the sensitivity of ARRK to the Fama-French factors found to be statistically significant. Given an unfavorable change in the value of each factor, holding all else equal, it is clear that the market factor holds the most sway over ARRK's returns, followed by HML, RMW, and SMB. The change each month was calculated from a disadvantageous 50 basis point move beyond the previous months'.

In [652]:
```
fig5, ax5 = plt.subplots(figsize=(15,10))

ax5.plot(df.index,df['OLS_HML']*100)
ax5.plot(df.index,df['OLS_RMW']*100)
ax5.plot(df.index,df['OLS_MKT']*100)
ax5.plot(df.index,df['OLS_SMB']*100)
ax5.legend(['HML','RMW','MKT','SMB'])

plt.gca().yaxis.set_major_formatter(plt.FuncFormatter('{}%'.format))
plt.title("ARRK Sensitivity to Fama-French Factors")
plt.xlabel("Months")
plt.ylabel("Monthly Return")
```

Out[652]: `Text(0, 0.5, 'Monthly Return')`



Calvin Crosby

In [ ]: