Imports and utility functions

```
from google.colab import files
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import pylab
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import RocCurveDisplay, auc
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import MinMaxScaler
```

```
def import_dataset():
    uploaded = files.upload()

    for fn in uploaded.keys():
        print('User uploaded file "{name}" with length {length} bytes'.format(
        name=fn, length=len(uploaded[fn])))

    with open(fn, 'r') as opened_file:
        txt_lines = opened_file.readlines()

    sequences = []
    label_array = np.zeros(len(txt_lines))
    for i in range (len(txt_lines)):
        line_elements = txt_lines[i].split('\t')
        label_array[i] = int(line_elements[0])
        seq = list(line_elements[1][:-1])
        sequences.append(seq)

    raw_dataset = pd.DataFrame(data=sequences, columns=range(1,201))
    raw_dataset['label'] = label_array
    return raw_dataset

def calc_mean_letter_case(vec):
    return ((vec < 'a').sum() - (vec >= 'a').sum()) / len(vec)

def max_element(arr):
    abs_arr = np.abs(arr)
    max_index = np.argmax(abs_arr)
    return arr[max_index]
```

SRSF1

ENCSR432XUP

Import Dataset and fit model

```
SRSF1_ENCSR432XUP_ds = import_dataset()
[SRSF1_ENCSR432XUP_train, SRSF1_ENCSR432XUP_test] = train_test_split(SRSF1_ENCSR432XUP_ds,train_size=0.8, random_state=103)

# One-Hot Encoding
SRSF1_ENCSR432XUP_train_features = pd.get_dummies(SRSF1_ENCSR432XUP_train.iloc[:, 0:200]).to_numpy()
SRSF1_ENCSR432XUP_train_labels = SRSF1_ENCSR432XUP_train['label'].to_numpy()

SRSF1_ENCSR432XUP_test_features = pd.get_dummies(SRSF1_ENCSR432XUP_test.iloc[:, 0:200]).to_numpy()
SRSF1_ENCSR432XUP_test_labels = SRSF1_ENCSR432XUP_test['label'].to_numpy()

# C=1e-3 worked best
SRSF1_ENCSR432XUP_model = svm.SVC(C=1e-3, kernel="linear")
SRSF1_ENCSR432XUP_model = SRSF1_ENCSR432XUP_model.fit(SRSF1_ENCSR432XUP_train_features, SRSF1_ENCSR432XUP_train_labels)
```

```
יש לבחור קבצים  לא נבחר קובץ   Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving SRSF1_ENCSR432XUP_dataset.txt to SRSF1_ENCSR432XUP_dataset.txt
User uploaded file "SRSF1_ENCSR432XUP_dataset.txt" with length 816000 bytes
```

Import RBPmap predictions

```
uploaded = files.upload()
```

```
for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))

with open(fn, 'r') as opened_file:
    txt_lines = opened_file.readlines()
txt_lines = txt_lines[1:]

SRSF1_ENCSR432XUP_rbp_no_conservation = np.array([int(line.split('\t')[0]) for line in txt_lines])
SRSF1_ENCSR432XUP_true_labels = np.array([int(line.split('\t')[1]) for line in txt_lines])
```

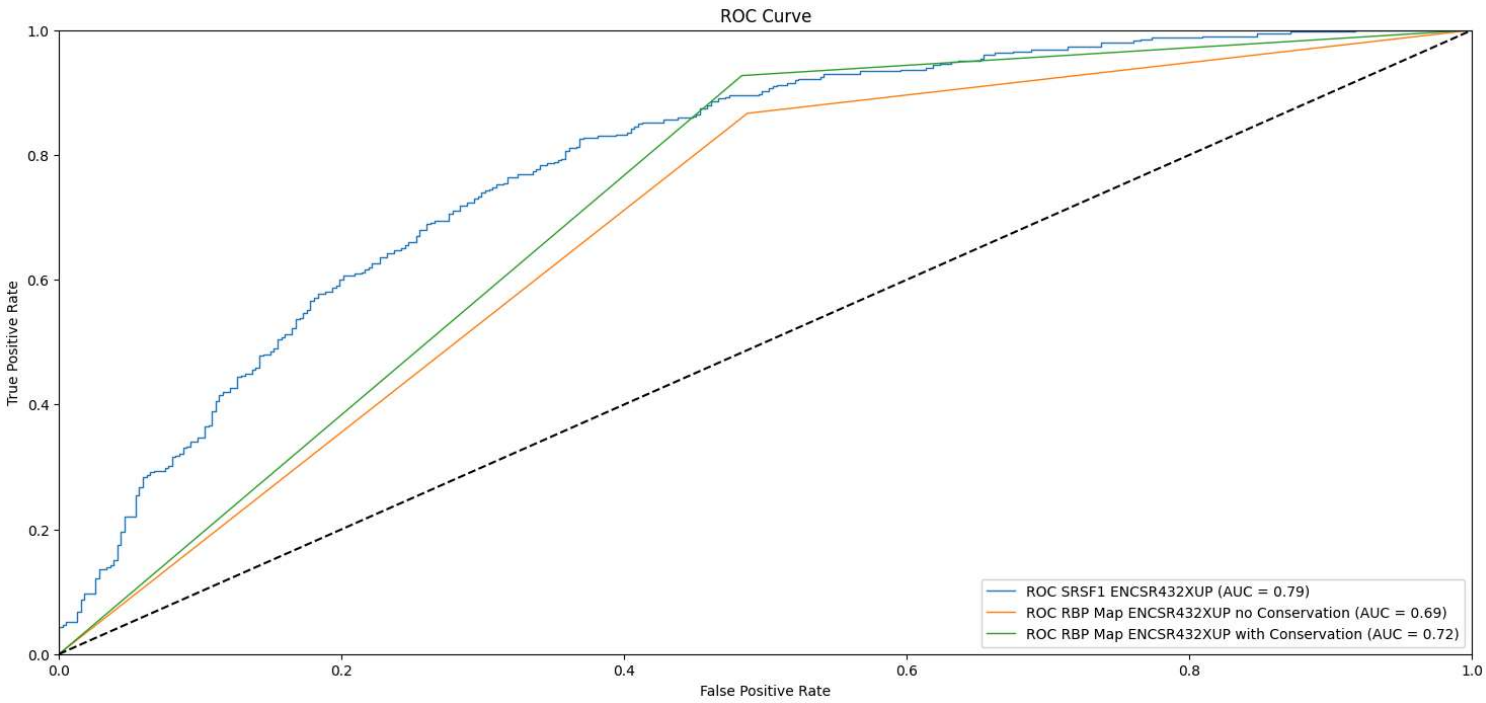
```
יש לבחור קבצים  לא נבחר קובץ
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving SRSF1_ENCSR432XUP_rbp_predictions_no_conservation.txt to SRSF1_ENCSR432XUP_rbp_predictions_no_conservation (1).txt
User uploaded file "SRSF1_ENCSR432XUP_rbp_predictions_no_conservation (1).txt" with length 20013 bytes
```

```
uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))

with open(fn, 'r') as opened_file:
    txt_lines = opened_file.readlines()
txt_lines = txt_lines[1:]

SRSF1_ENCSR432XUP_rbp_with_conservation = np.array([int(line.split('\t')[0]) for line in txt_lines])
```

```
יש לבחור קבצים  לא נבחר קובץ
Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving SRSF1_ENCSR432XUP_rbp_predictions_with_conservation.txt to SRSF1_ENCSR432XUP_rbp_predictions_with_conservation.txt
User uploaded file "SRSF1_ENCSR432XUP_rbp_predictions_with_conservation.txt" with length 20013 bytes
```

Display results

```
fig, ax = plt.subplots(figsize=(18, 8))

viz = RocCurveDisplay.from_estimator(
    SRSF1_ENCSR432XUP_model,
    SRSF1_ENCSR432XUP_test_features,
    SRSF1_ENCSR432XUP_test_labels,
    name=f"ROC SRSF1 ENCSR432XUP",
    lw=1,
    ax=ax
)

viz = RocCurveDisplay.from_predictions(
    SRSF1_ENCSR432XUP_rbp_no_conservation,
    SRSF1_ENCSR432XUP_true_labels,
    name=f"ROC RBP Map ENCSR432XUP no Conservation",
    lw=1,
    ax=ax
)

viz = RocCurveDisplay.from_predictions(
    SRSF1_ENCSR432XUP_rbp_with_conservation,
    SRSF1_ENCSR432XUP_true_labels,
    name=f"ROC RBP Map ENCSR432XUP with Conservation",
    lw=1,
    ax=ax
)

ax.set(
    xlabel="False Positive Rate",
    ylabel="True Positive Rate",
    title=f"ROC Curve",
)

ax.legend(loc="lower right")
ax.plot(np.linspace(0,1,10), np.linspace(0,1,10), '--',color='black')
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.show()
```

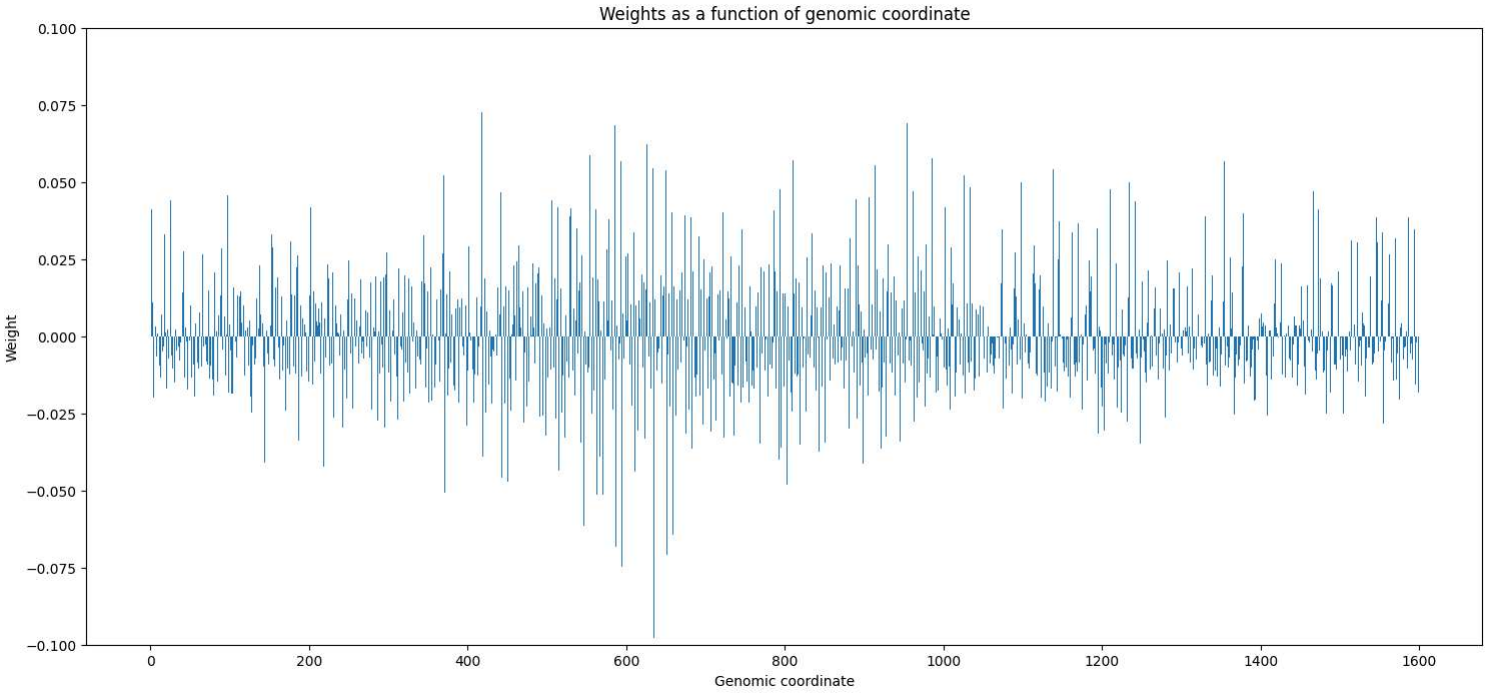


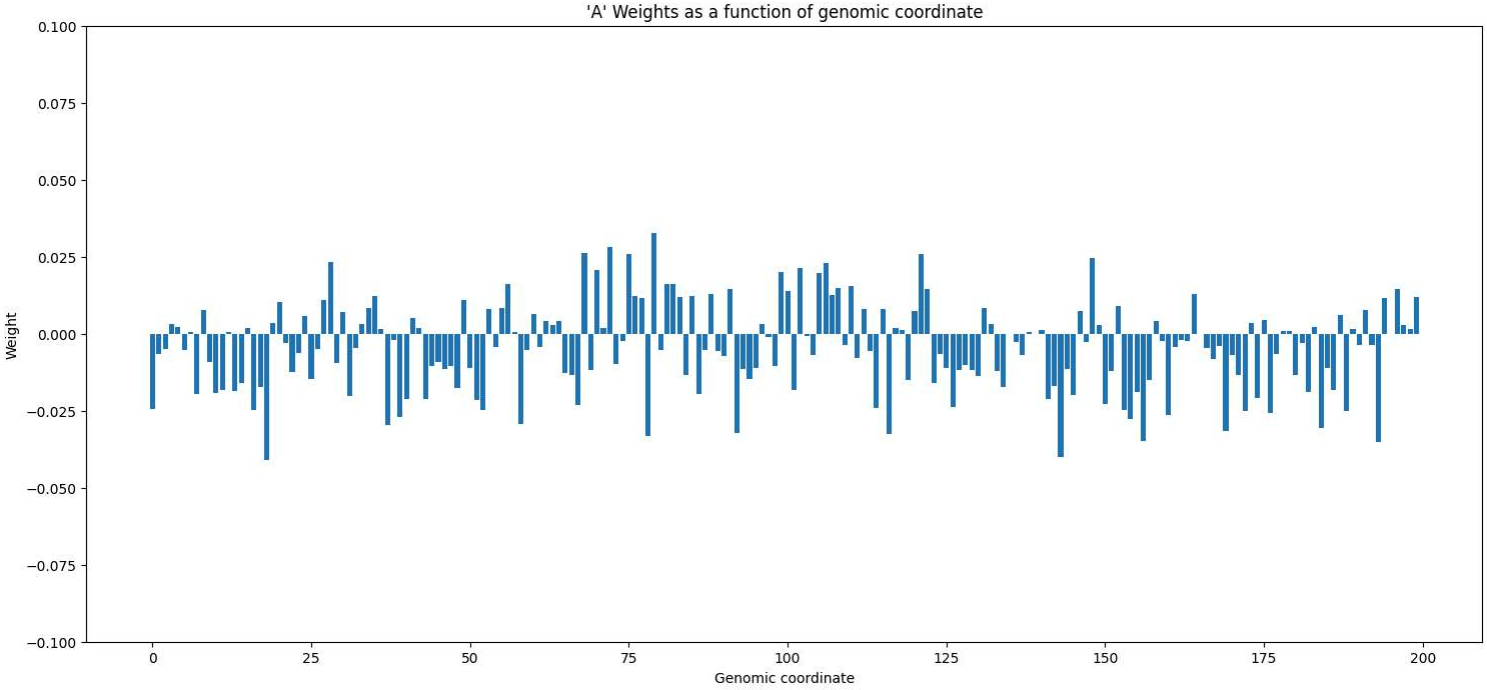
Display trained weights

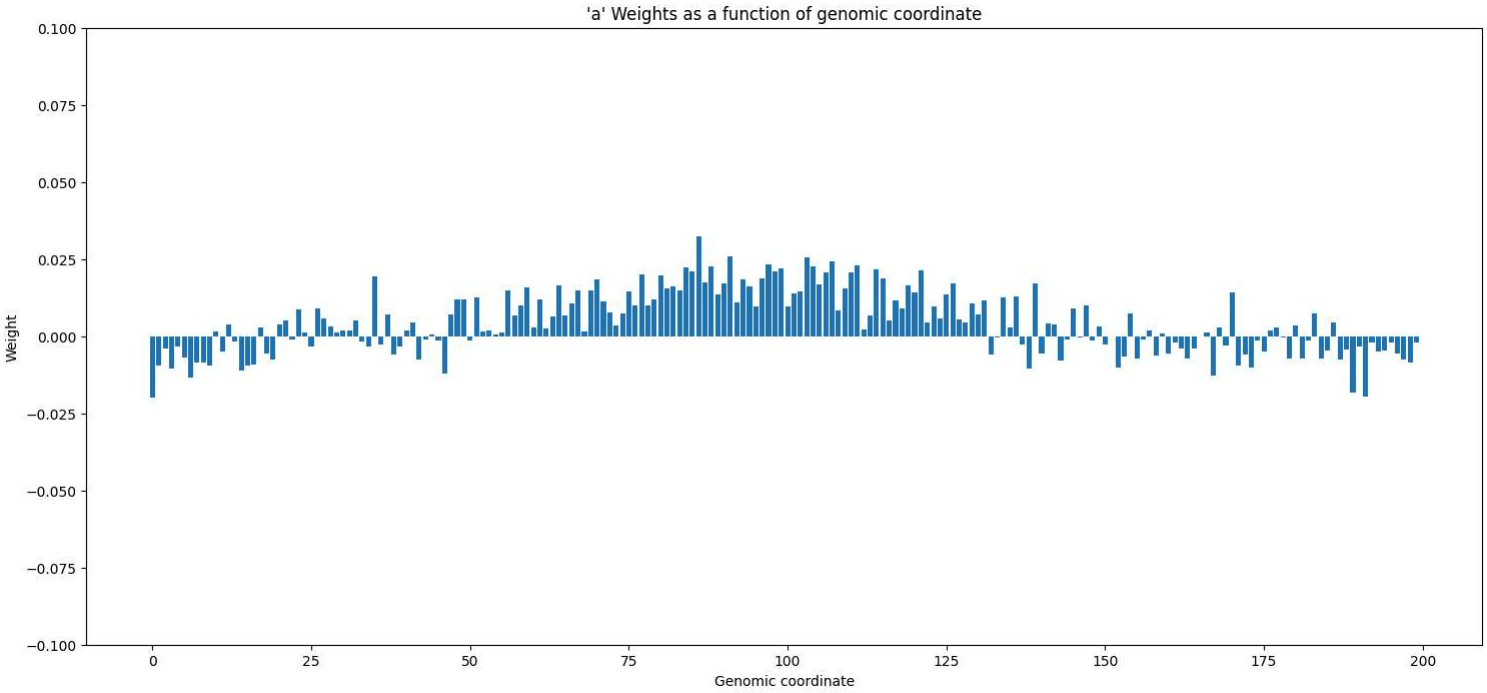
```
SRSF1_ENCSR432XUP_weights = SRSF1_ENCSR432XUP_model.coef_[0]
SRSF1_ENCSR432XUP_A_weights = SRSF1_ENCSR432XUP_weights[::8]
SRSF1_ENCSR432XUP_a_weights = SRSF1_ENCSR432XUP_weights[4::8]
SRSF1_ENCSR432XUP_C_weights = SRSF1_ENCSR432XUP_weights[1::8]
SRSF1_ENCSR432XUP_c_weights = SRSF1_ENCSR432XUP_weights[5::8]
SRSF1_ENCSR432XUP_G_weights = SRSF1_ENCSR432XUP_weights[2::8]
SRSF1_ENCSR432XUP_g_weights = SRSF1_ENCSR432XUP_weights[6::8]
SRSF1_ENCSR432XUP_T_weights = SRSF1_ENCSR432XUP_weights[3::8]
SRSF1_ENCSR432XUP_t_weights = SRSF1_ENCSR432XUP_weights[7::8]
SRSF1_ENCSR432XUP_a_weights = np.zeros(1, len(SRSF1_ENCSR432XUP_a_weights))
SRSF1_ENCSR432XUP_c_weights = np.zeros(1, len(SRSF1_ENCSR432XUP_c_weights))
SRSF1_ENCSR432XUP_g_weights = np.zeros(1, len(SRSF1_ENCSR432XUP_g_weights))
SRSF1_ENCSR432XUP_t_weights = np.zeros(1, len(SRSF1_ENCSR432XUP_t_weights))

SRSF1_ENCSR432XUP_avg_weights = SRSF1_ENCSR432XUP_weights.reshape(-1, 8).mean(axis=1)
SRSF1_ENCSR432XUP_avg_uppercase_weights = SRSF1_ENCSR432XUP_weights.reshape(-1, 4).mean(axis=1)[::2]
SRSF1_ENCSR432XUP_avg_lowercase_weights = SRSF1_ENCSR432XUP_weights.reshape(-1, 4).mean(axis=1)[1::2]

SRSF1_ENCSR432XUP_max_weights = np.apply_along_axis(max_element, axis=1, arr=SRSF1_ENCSR432XUP_weights.reshape(-1, 8))
SRSF1_ENCSR432XUP_max_uppercase_weights = np.apply_along_axis(max_element, axis=1, arr=SRSF1_ENCSR432XUP_weights.reshape(-1, 4)[::2])
SRSF1_ENCSR432XUP_max_lowercase_weights = np.apply_along_axis(max_element, axis=1, arr=SRSF1_ENCSR432XUP_weights.reshape(-1, 4)[1::2])

print('Average weights mean is ', SRSF1_ENCSR432XUP_avg_weights.mean())
print('Average weights variance is ', SRSF1_ENCSR432XUP_avg_weights.var())
```

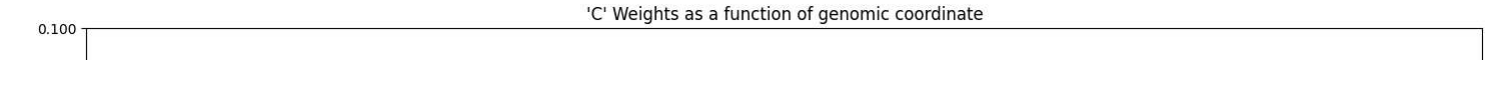
```
Average weights mean is  2.3018696324039745e-17
Average weights variance is  6.310540212325845e-35
```

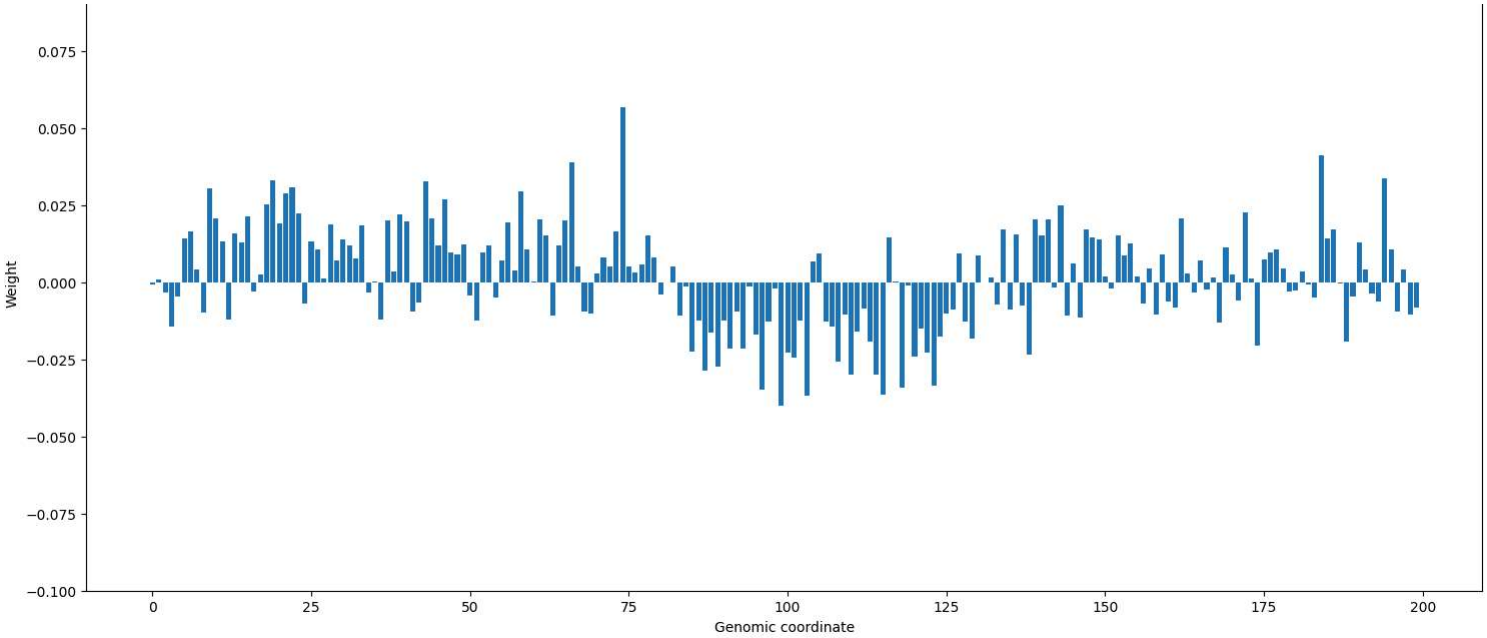
```
plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_weights)), SRSF1_ENCSR432XUP_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_A_weights)), SRSF1_ENCSR432XUP_A_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'A' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_a_weights)), SRSF1_ENCSR432XUP_a_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'a' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_C_weights)), SRSF1_ENCSR432XUP_C_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'C' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_c_weights)), SRSF1_ENCSR432XUP_c_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'c' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_G_weights)), SRSF1_ENCSR432XUP_G_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'G' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_g_weights)), SRSF1_ENCSR432XUP_g_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'g' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_T_weights)), SRSF1_ENCSR432XUP_T_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'T' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_t_weights)), SRSF1_ENCSR432XUP_t_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'t' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()
```


Weights as a function of genomic coordinate
Weight
Genomic coordinate

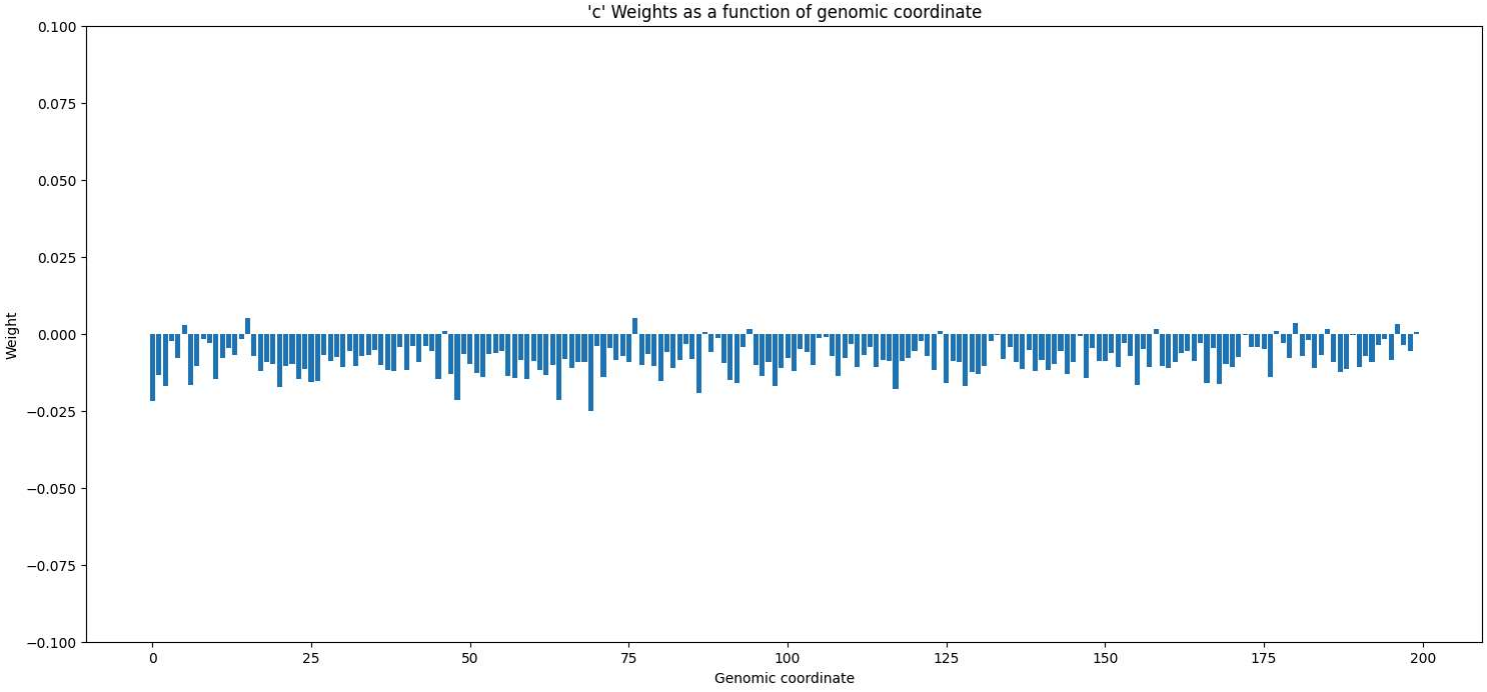

'A' Weights as a function of genomic coordinate
Weight
Genomic coordinate


'a' Weights as a function of genomic coordinate
Weight
Genomic coordinate

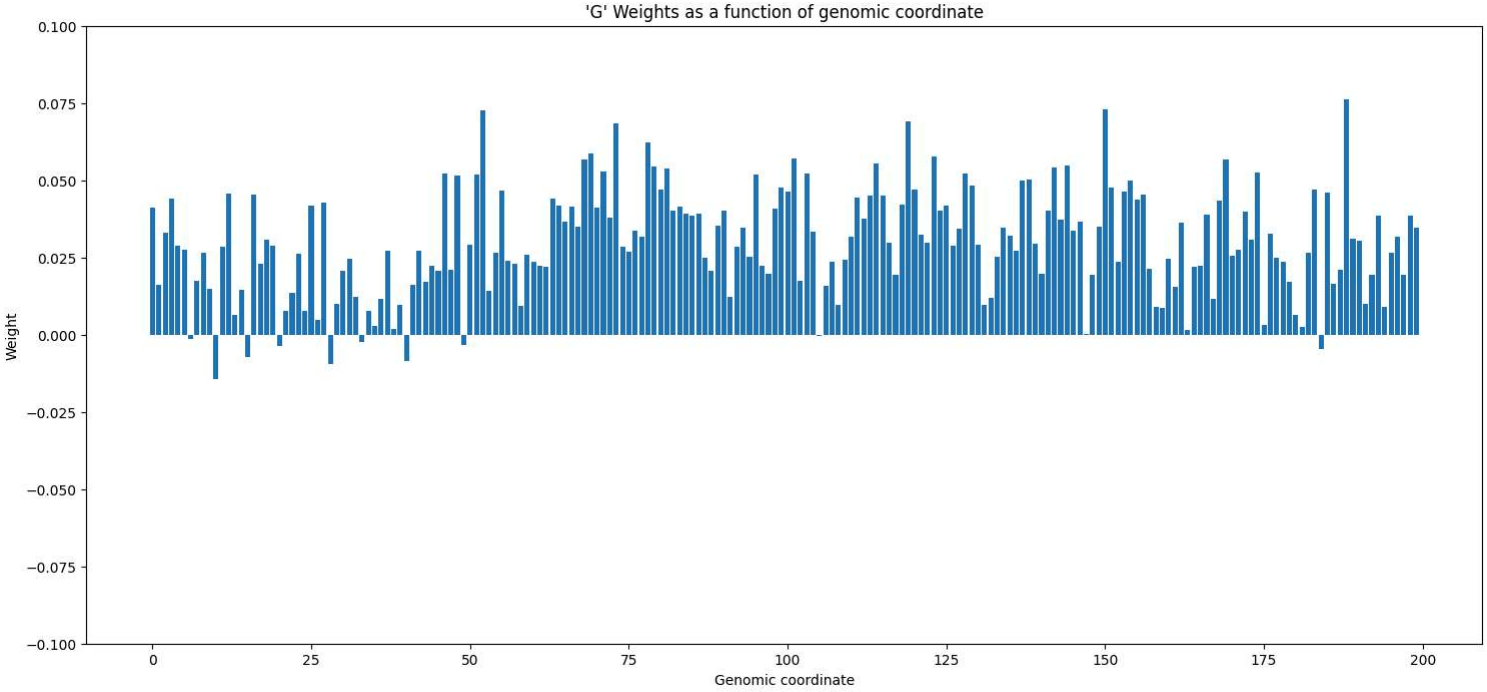

'C' Weights as a function of genomic coordinate



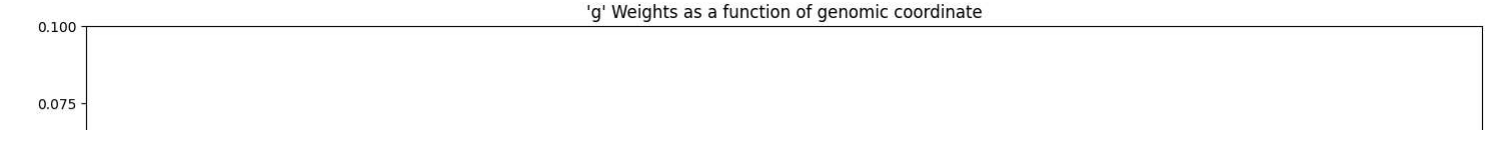
'c' Weights as a function of genomic coordinate

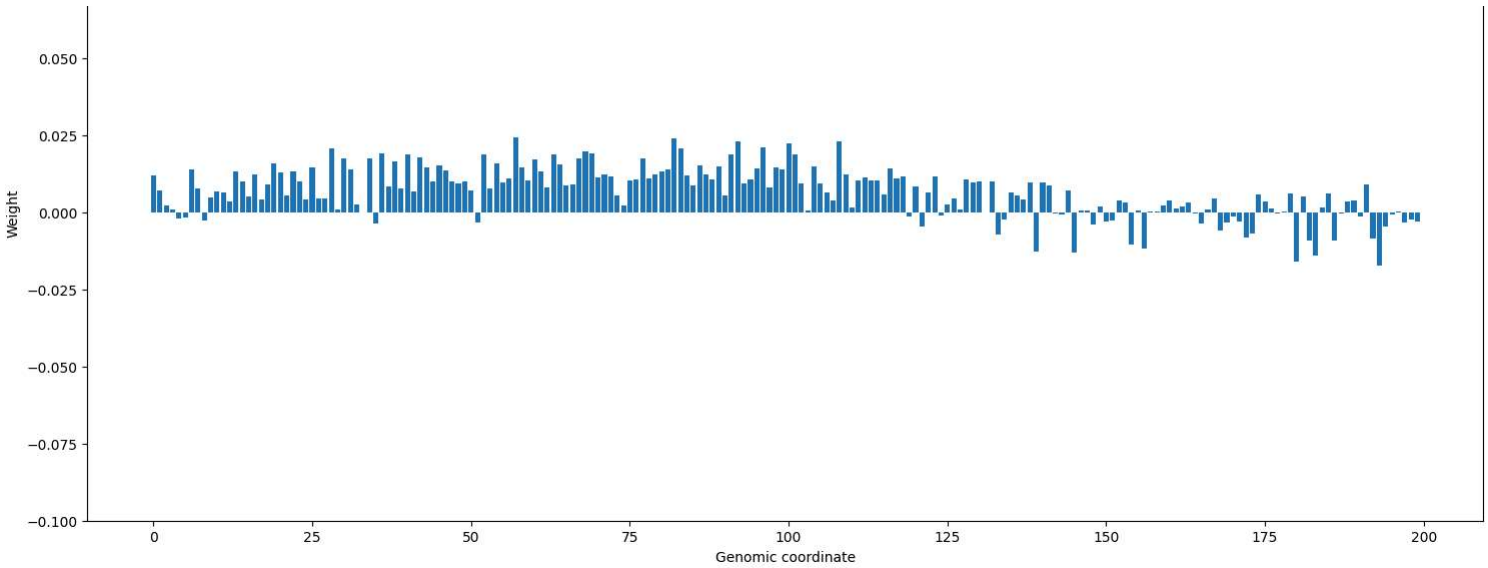


'G' Weights as a function of genomic coordinate

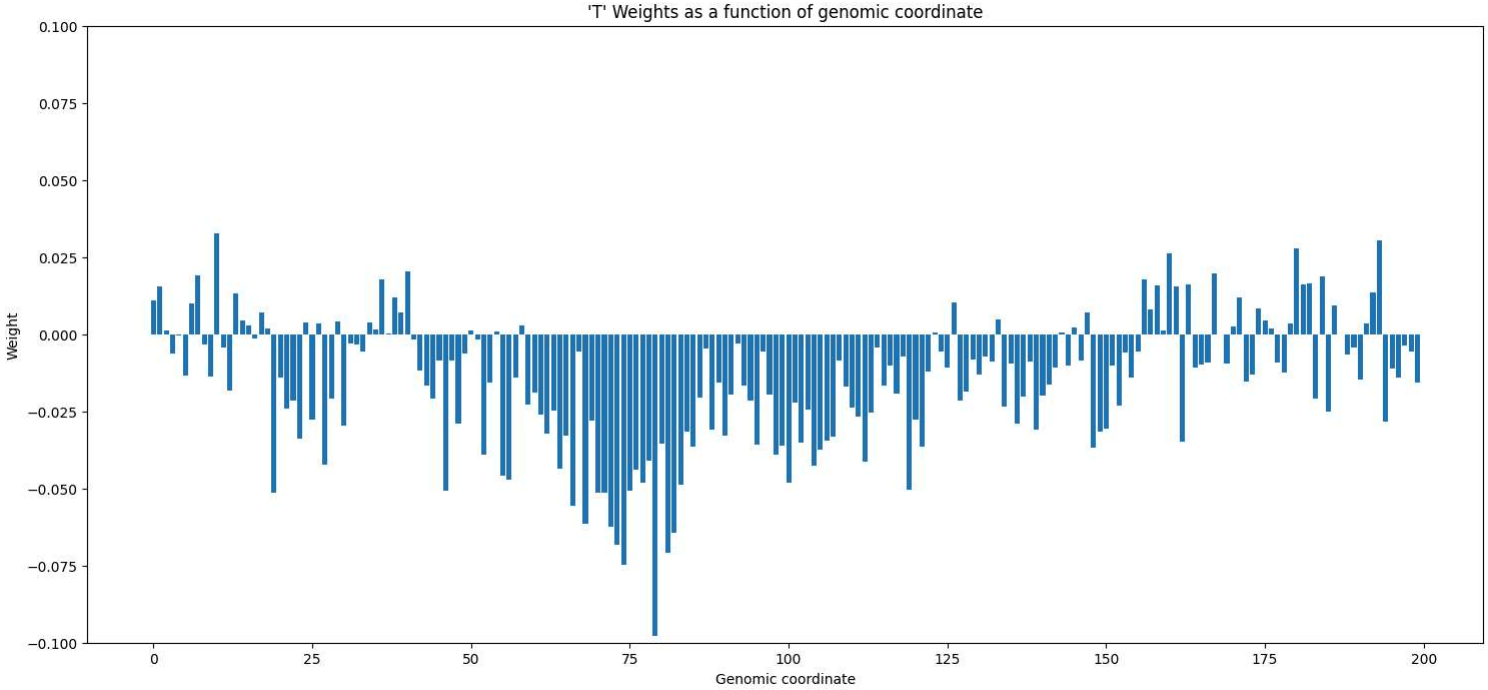


'g' Weights as a function of genomic coordinate

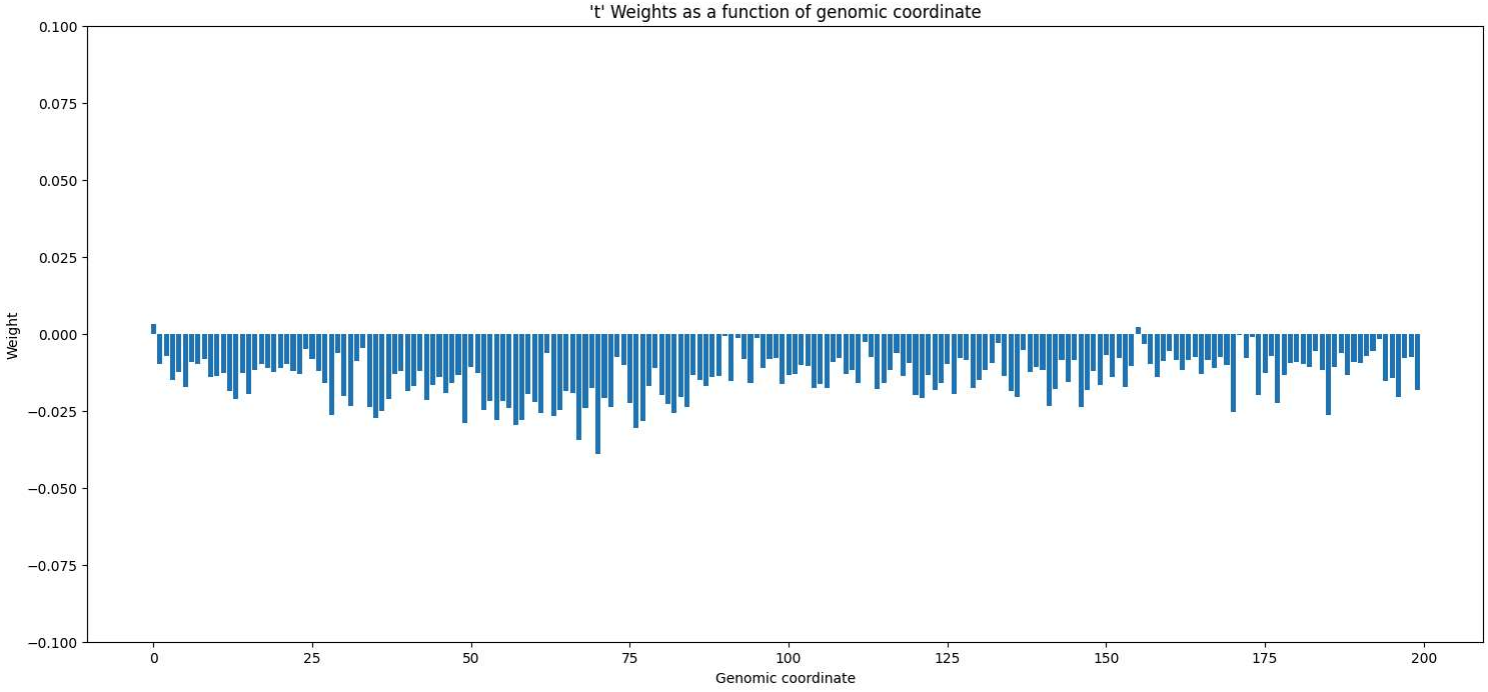




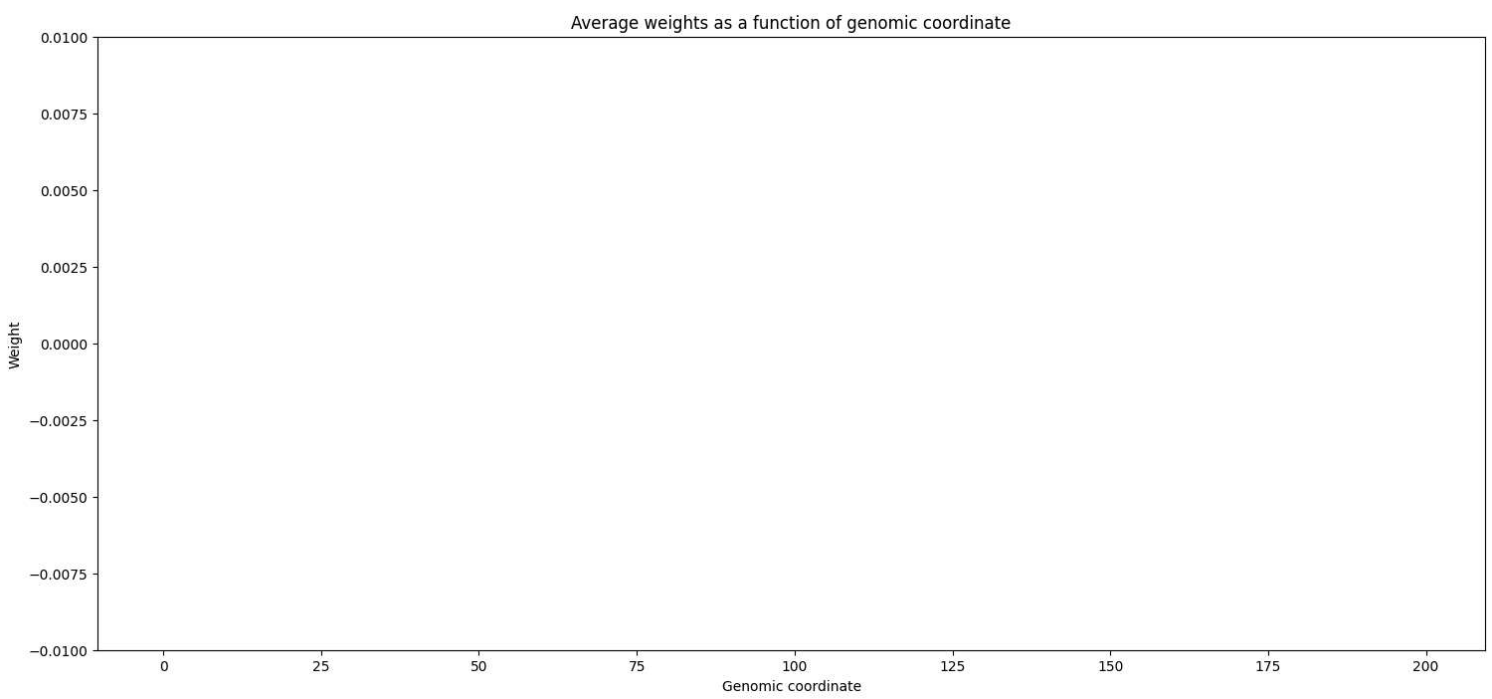
'T' Weights as a function of genomic coordinate

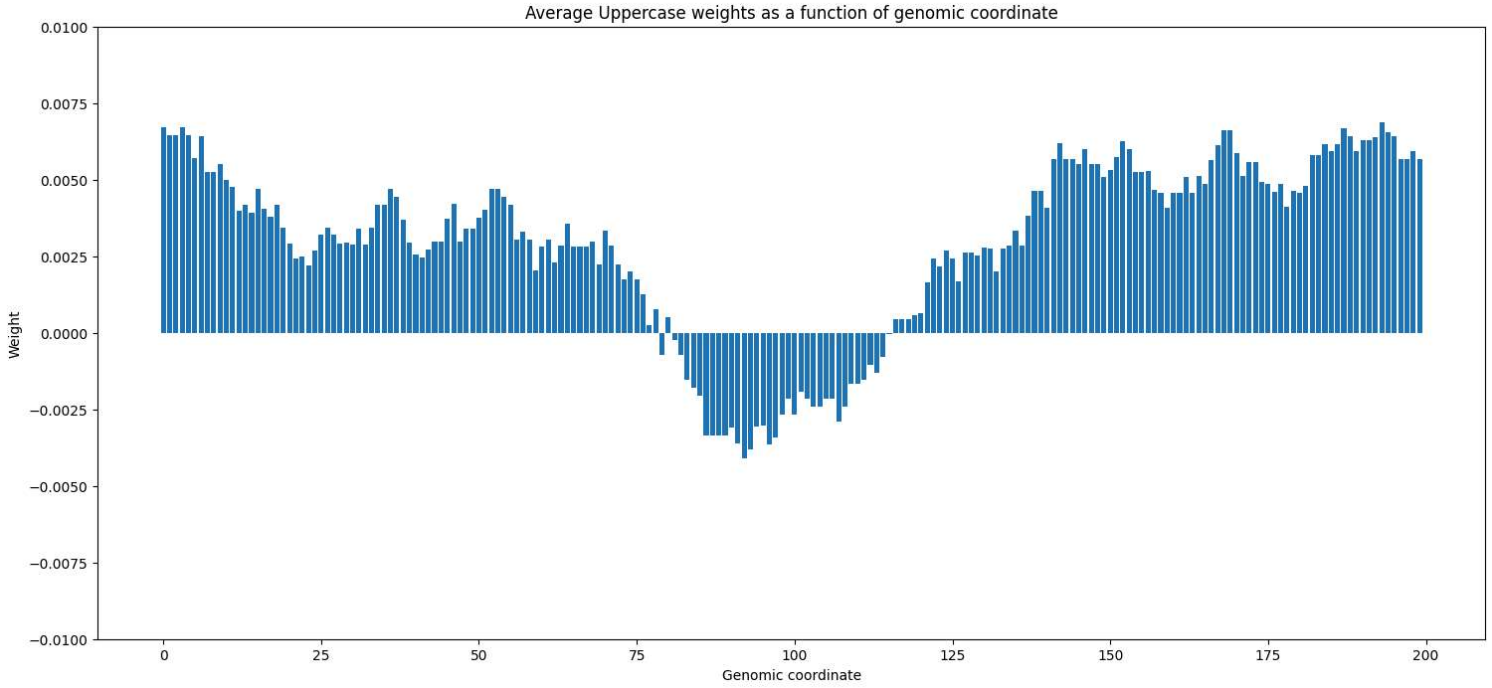


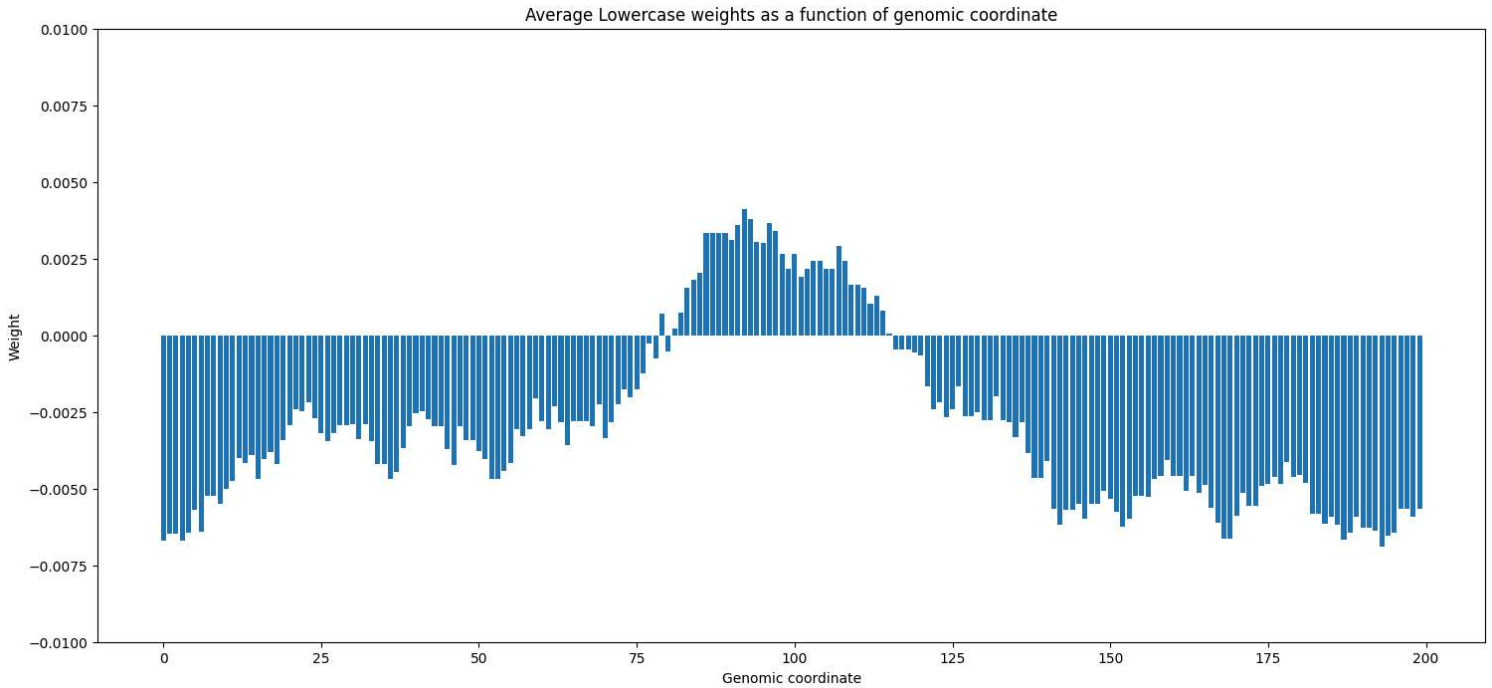
't' Weights as a function of genomic coordinate



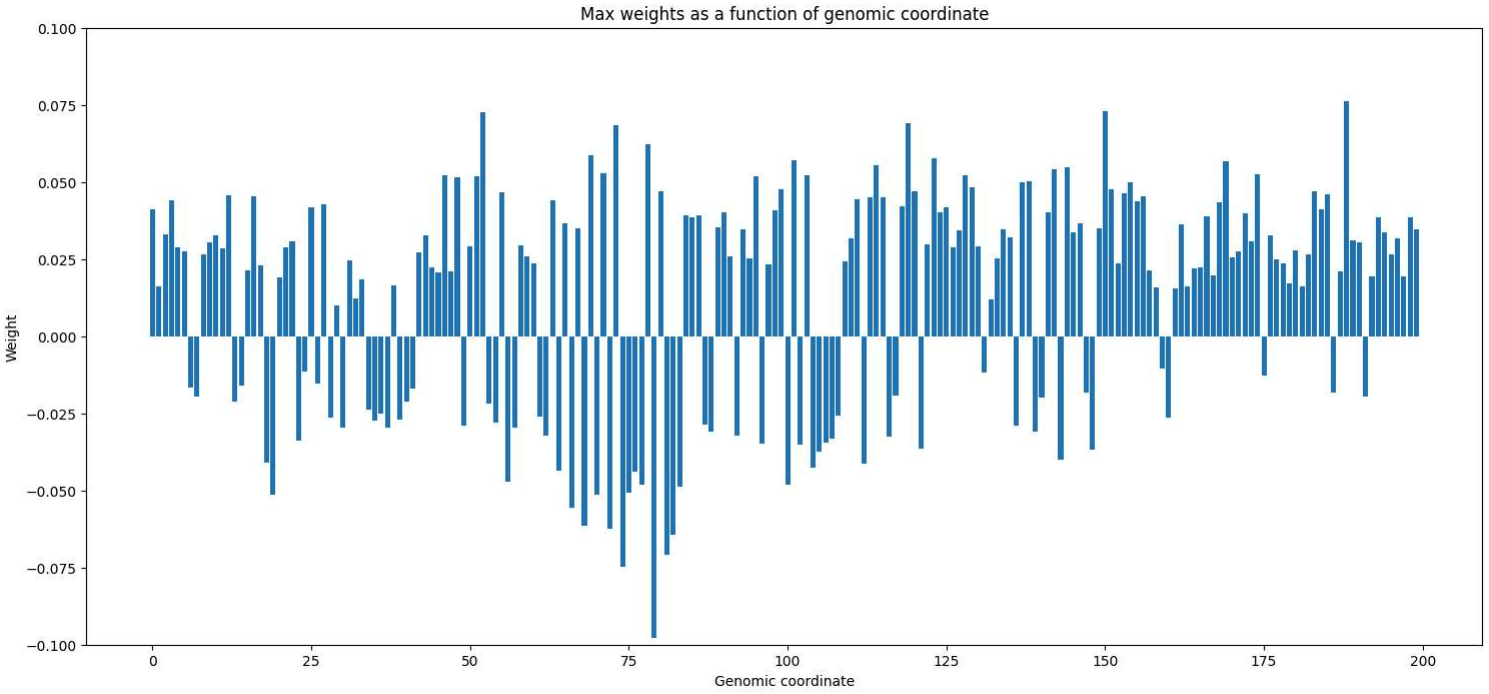
```
plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_avg_weights)), SRSF1_ENCSR432XUP_avg_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Average weights as a function of genomic coordinate")
plt.ylim([-0.01, 0.01])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_avg_uppercase_weights)), SRSF1_ENCSR432XUP_avg_uppercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Average Uppercase weights as a function of genomic coordinate")
plt.ylim([-0.01, 0.01])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_avg_lowercase_weights)), SRSF1_ENCSR432XUP_avg_lowercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Average Lowercase weights as a function of genomic coordinate")
plt.ylim([-0.01, 0.01])
plt.show()
```

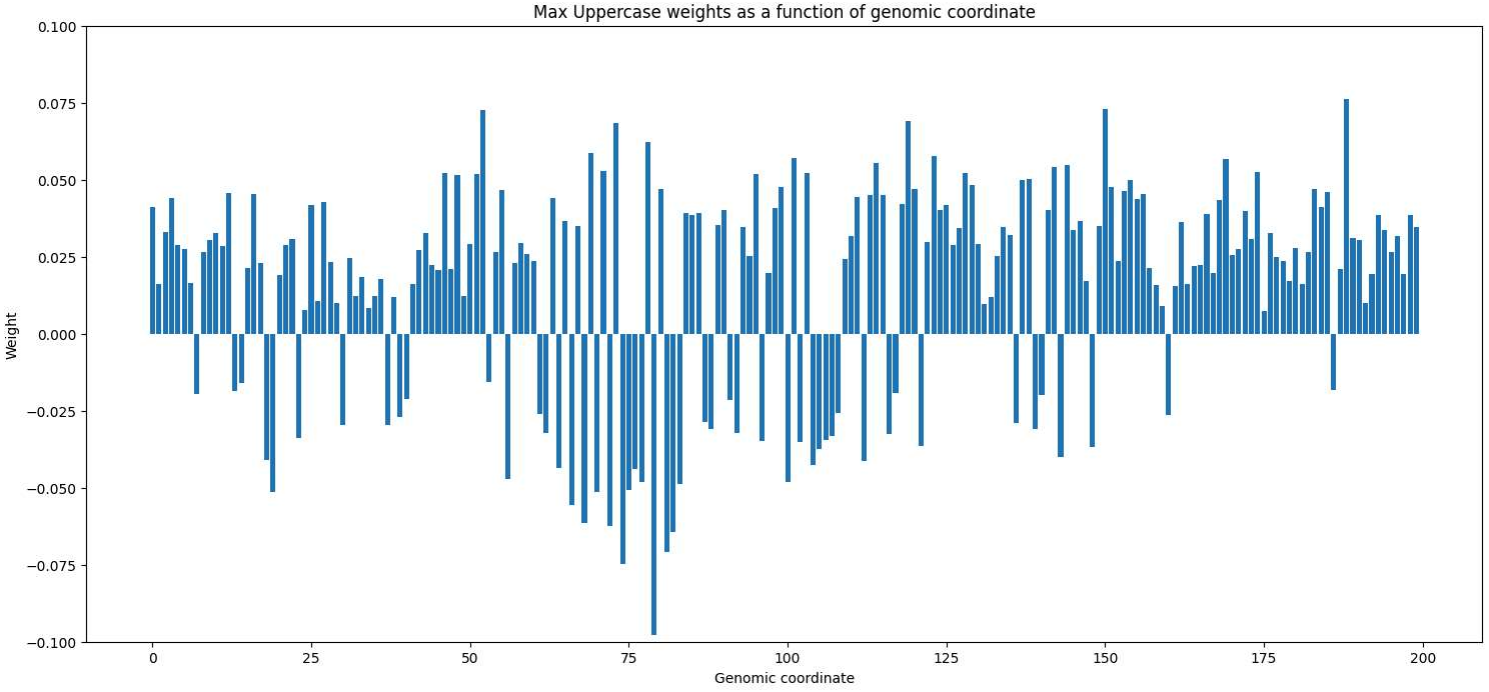

Average weights as a function of genomic coordinate
Weight
Genomic coordinate
0.0100
0.0075
0.0050
0.0025
0.0000
−0.0025
−0.0050
−0.0075
−0.0100
0
25
50
75
100
125
150
175
200

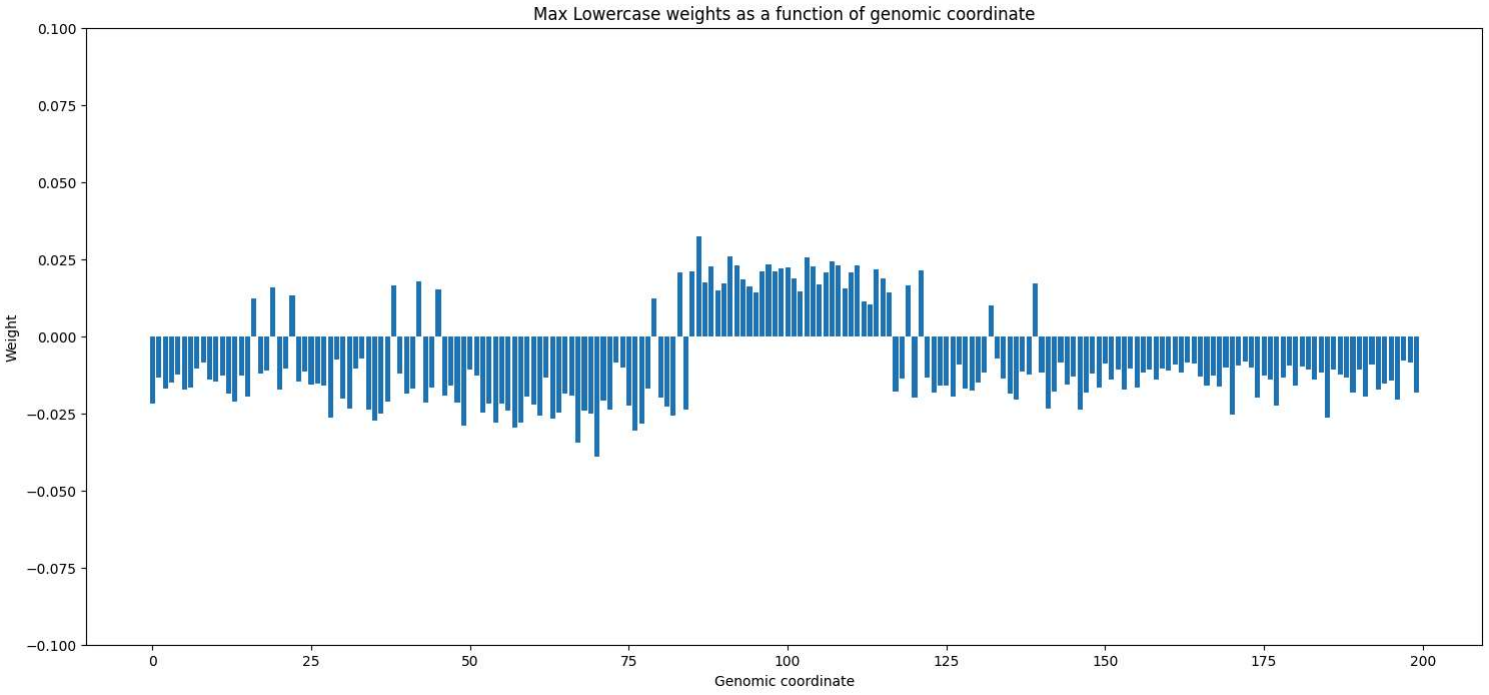

Average Uppercase weights as a function of genomic coordinate
Weight
Genomic coordinate
0.0100
0.0075
0.0050
0.0025
0.0000
−0.0025
−0.0050
−0.0075
−0.0100
0
25
50
75
100
125
150
175
200


Average Lowercase weights as a function of genomic coordinate
Weight
Genomic coordinate
0.0100
0.0075
0.0050
0.0025
0.0000
−0.0025
−0.0050
−0.0075
−0.0100
0
25
50
75
100
125
150
175
200

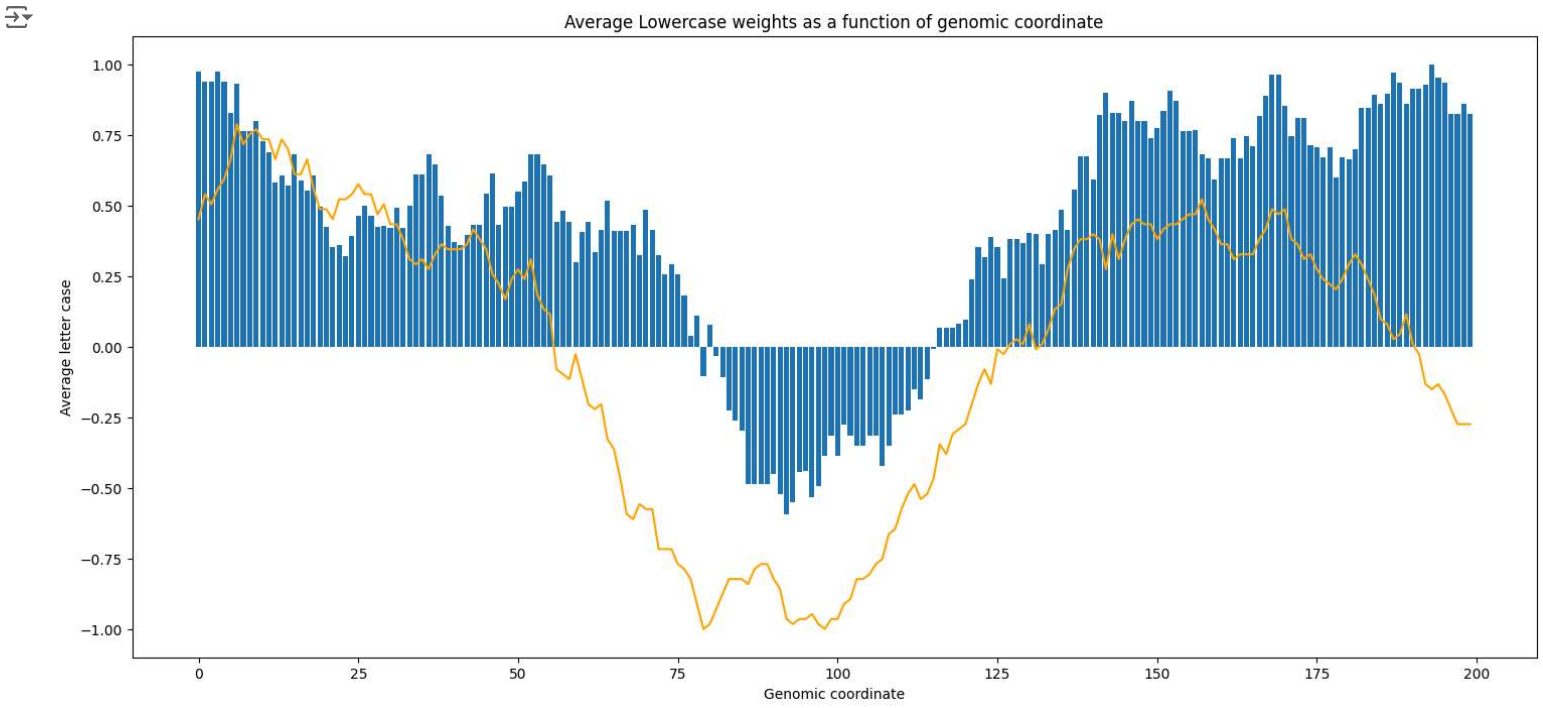
```
plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_max_weights)), SRSF1_ENCSR432XUP_max_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Max weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_max_uppercase_weights)), SRSF1_ENCSR432XUP_max_uppercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Max Uppercase weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_max_lowercase_weights)), SRSF1_ENCSR432XUP_max_lowercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Max Lowercase weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()
```


Max weights as a function of genomic coordinate
Weight
Genomic coordinate
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
0
25
50
75
100
125
150
175
200


Max Uppercase weights as a function of genomic coordinate
Weight
Genomic coordinate
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
0
25
50
75
100
125
150
175
200


Max Lowercase weights as a function of genomic coordinate
Weight
Genomic coordinate
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
0
25
50
75
100
125
150
175
200

```
SRSF1_ENCSR432XUP_mean_letter_case = np.array([calc_mean_letter_case(SRSF1_ENCSR432XUP_ds.iloc[:, i]) for i in range(200)])
SRSF1_ENCSR432XUP_mean_letter_case = (SRSF1_ENCSR432XUP_mean_letter_case - SRSF1_ENCSR432XUP_mean_letter_case.mean())
SRSF1_ENCSR432XUP_mean_letter_case = SRSF1_ENCSR432XUP_mean_letter_case / max(abs(SRSF1_ENCSR432XUP_mean_letter_case))
normalized_SRSF1_ENCSR432XUP_avg_uppercase_weights = SRSF1_ENCSR432XUP_avg_uppercase_weights / max(abs(SRSF1_ENCSR432XUP_avg_uppercase_weights))

plt.figure(figsize=(18, 8))
plt.plot(range(len(SRSF1_ENCSR432XUP_mean_letter_case)), SRSF1_ENCSR432XUP_mean_letter_case, color='orange')
plt.bar(range(len(normalized_SRSF1_ENCSR432XUP_avg_uppercase_weights)), normalized_SRSF1_ENCSR432XUP_avg_uppercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Average letter case')
plt.title("Average Lowercase weights as a function of genomic coordinate")
plt.show()
```



ENCSR321PWZ

Import Dataset and fit model

```
SRSF1_ENCSR321PWZ_ds = import_dataset()
[SRSF1_ENCSR321PWZ_train, SRSF1_ENCSR321PWZ_test] = train_test_split(SRSF1_ENCSR321PWZ_ds,train_size=0.8, random_state=103)

# One-Hot Encoding
SRSF1_ENCSR321PWZ_train_features = pd.get_dummies(SRSF1_ENCSR321PWZ_train.iloc[:, 0:200]).to_numpy()
SRSF1_ENCSR321PWZ_train_labels = SRSF1_ENCSR321PWZ_train['label'].to_numpy()

SRSF1_ENCSR321PWZ_test_features = pd.get_dummies(SRSF1_ENCSR321PWZ_test.iloc[:, 0:200]).to_numpy()
SRSF1_ENCSR321PWZ_test_labels = SRSF1_ENCSR321PWZ_test['label'].to_numpy()

# C=1e-3 worked best
SRSF1_ENCSR321PWZ_model = svm.SVC(C=1e-3, kernel="linear")
SRSF1_ENCSR321PWZ_model = SRSF1_ENCSR321PWZ_model.fit(SRSF1_ENCSR321PWZ_train_features, SRSF1_ENCSR321PWZ_train_labels)
```

```
בחר קובץ | לא נבחר קובץ  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving SRSF1_ENCSR321PWZ_dataset.txt to SRSF1_ENCSR321PWZ_dataset.txt
User uploaded file "SRSF1_ENCSR321PWZ_dataset.txt" with length 816000 bytes
```

Import RBPmap predictions

```
uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))

with open(fn, 'r') as opened_file:
    txt_lines = opened_file.readlines()
txt_lines = txt_lines[1:]

SRSF1_ENCSR321PWZ_rbp_no_conservation = np.array([int(line.split('\t')[0]) for line in txt_lines])
SRSF1_ENCSR321PWZ_true_labels = np.array([int(line.split('\t')[1]) for line in txt_lines])
```

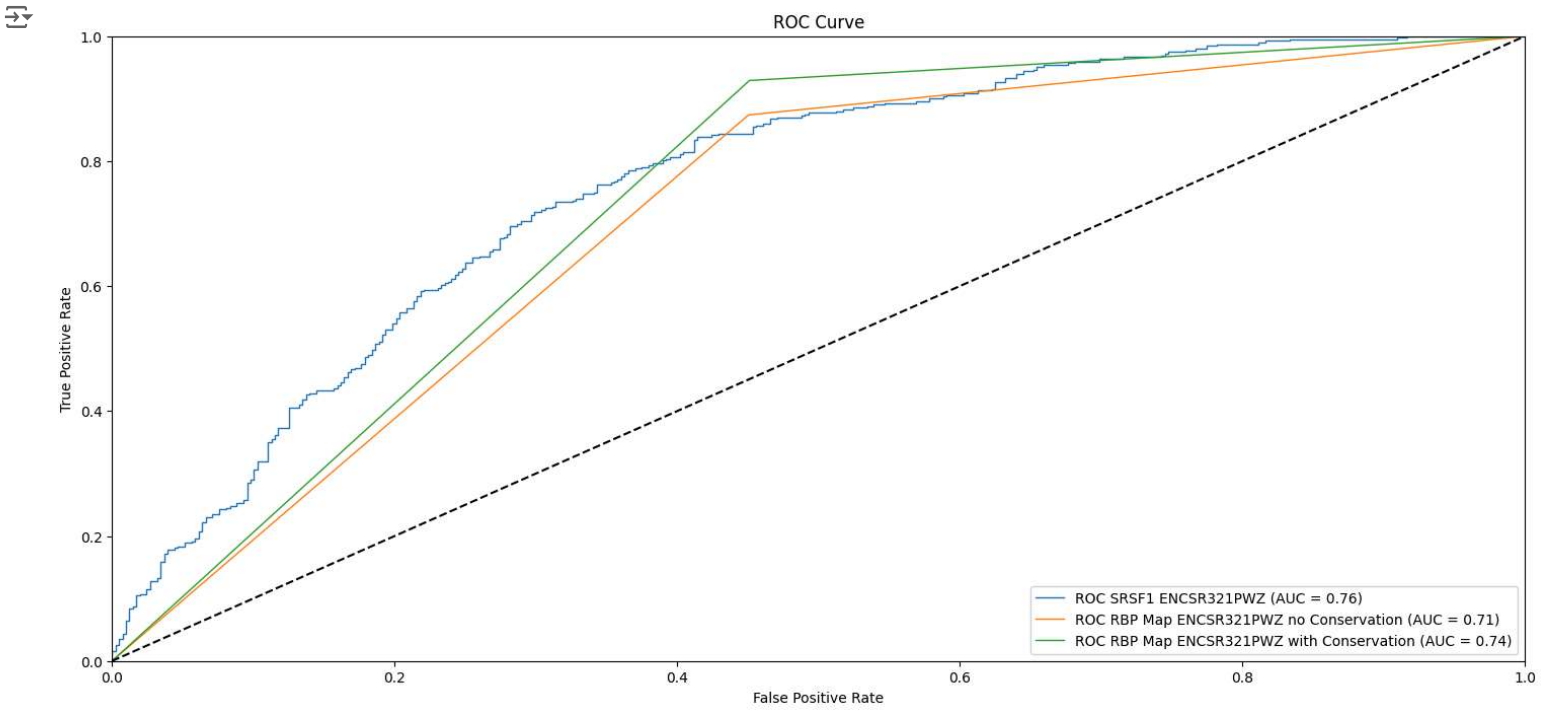
```
בחר קובץ | לא נבחר קובץ  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving SRSF1_ENCSR321PWZ_rbp_predictions_no_conservation.txt to SRSF1_ENCSR321PWZ_rbp_predictions_no_conservation.txt
User uploaded file "SRSF1_ENCSR321PWZ_rbp_predictions_no_conservation.txt" with length 20013 bytes
```

```
uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))

with open(fn, 'r') as opened_file:
    txt_lines = opened_file.readlines()
txt_lines = txt_lines[1:]

SRSF1_ENCSR321PWZ_rbp_with_conservation = np.array([int(line.split('\t')[0]) for line in txt_lines])
```

```
בחר קובץ | לא נבחר קובץ  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving SRSF1_ENCSR321PWZ_rbp_predictions_with_conservation.txt to SRSF1_ENCSR321PWZ_rbp_predictions_with_conservation.txt
User uploaded file "SRSF1_ENCSR321PWZ_rbp_predictions_with_conservation.txt" with length 20013 bytes
```

Display results

```
fig, ax = plt.subplots(figsize=(18, 8))

viz = RocCurveDisplay.from_estimator(
    SRSF1_ENCSR321PWZ_model,
    SRSF1_ENCSR321PWZ_test_features,
    SRSF1_ENCSR321PWZ_test_labels,
    name=f"ROC SRSF1 ENCSR321PWZ",
    lw=1,
    ax=ax
)

viz = RocCurveDisplay.from_predictions(
    SRSF1_ENCSR321PWZ_rbp_no_conservation,
    SRSF1_ENCSR321PWZ_true_labels,
    name=f"ROC RBP Map ENCSR321PWZ no Conservation",
    lw=1,
    ax=ax
)

viz = RocCurveDisplay.from_predictions(
    SRSF1_ENCSR321PWZ_rbp_with_conservation,
    SRSF1_ENCSR321PWZ_true_labels,
    name=f"ROC RBP Map ENCSR321PWZ with Conservation",
    lw=1,
    ax=ax
)

ax.set(
    xlabel="False Positive Rate",
    ylabel="True Positive Rate",
    title=f"ROC Curve",
)
ax.legend(loc="lower right")
ax.plot(np.linspace(0,1,10), np.linspace(0,1,10), '--',color='black')
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.show()
```

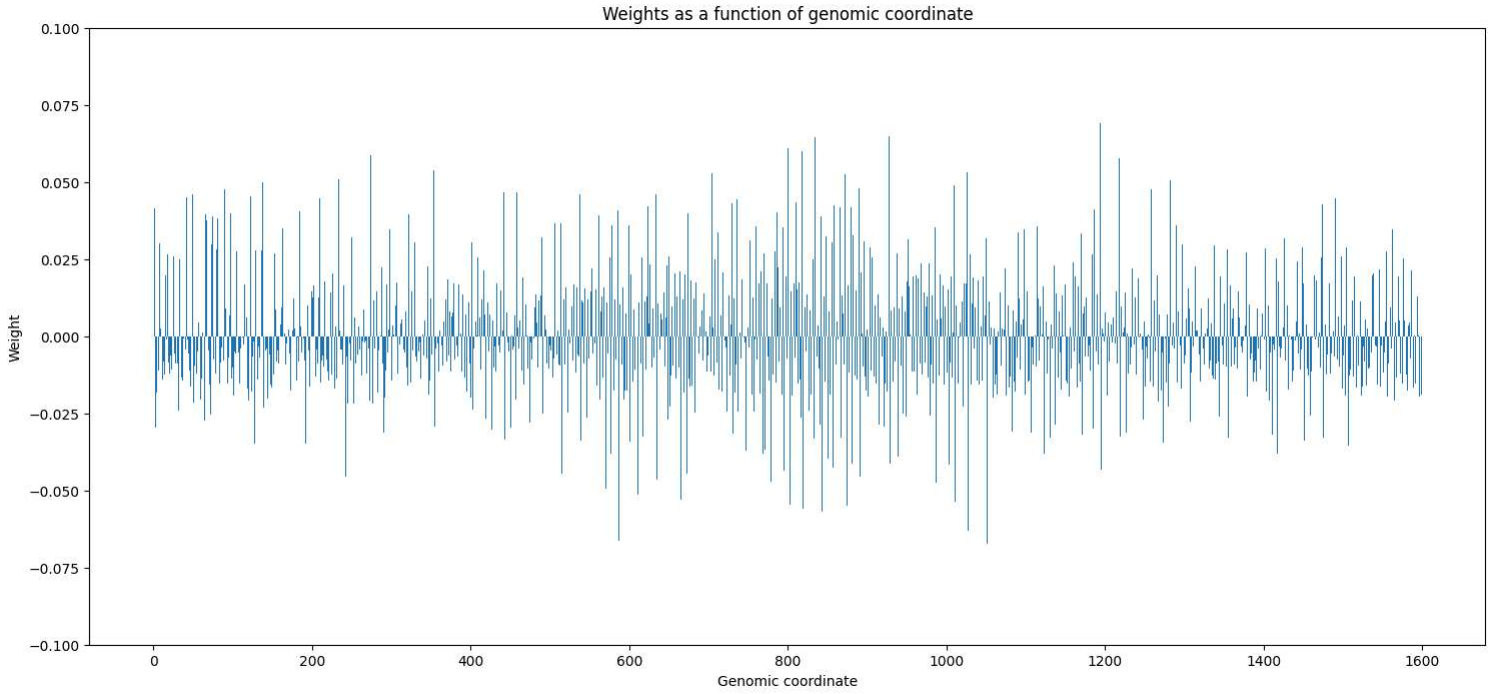


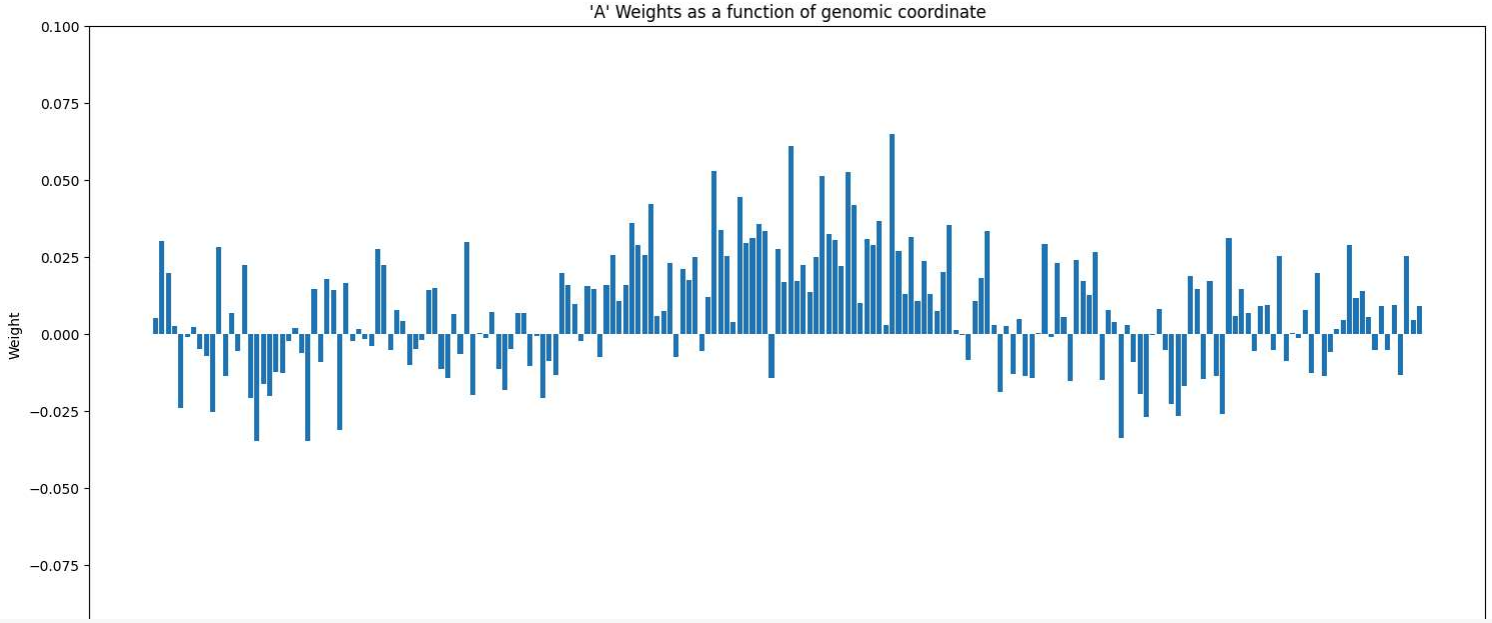
Display trained weights

```
SRSF1_ENCSR321PWZ_weights = SRSF1_ENCSR321PWZ_model.coef_[0]
SRSF1_ENCSR321PWZ_A_weights = SRSF1_ENCSR321PWZ_weights[::8]
SRSF1_ENCSR321PWZ_a_weights = SRSF1_ENCSR321PWZ_weights[4::8]
SRSF1_ENCSR321PWZ_C_weights = SRSF1_ENCSR321PWZ_weights[1::8]
SRSF1_ENCSR321PWZ_c_weights = SRSF1_ENCSR321PWZ_weights[5::8]
SRSF1_ENCSR321PWZ_G_weights = SRSF1_ENCSR321PWZ_weights[2::8]
SRSF1_ENCSR321PWZ_g_weights = SRSF1_ENCSR321PWZ_weights[6::8]
SRSF1_ENCSR321PWZ_T_weights = SRSF1_ENCSR321PWZ_weights[3::8]
SRSF1_ENCSR321PWZ_t_weights = SRSF1_ENCSR321PWZ_weights[7::8]

SRSF1_ENCSR321PWZ_avg_weights = SRSF1_ENCSR321PWZ_weights.reshape(-1, 8).mean(axis=1)
SRSF1_ENCSR321PWZ_avg_uppercase_weights = SRSF1_ENCSR321PWZ_weights.reshape(-1, 4).mean(axis=1)[::2]
SRSF1_ENCSR321PWZ_avg_lowercase_weights = SRSF1_ENCSR321PWZ_weights.reshape(-1, 4).mean(axis=1)[1::2]

SRSF1_ENCSR321PWZ_max_uppercase_weights = np.apply_along_axis(max_element, axis=1, arr=SRSF1_ENCSR321PWZ_weights.reshape(-1, 4)[::2])
SRSF1_ENCSR321PWZ_max_lowercase_weights = np.apply_along_axis(max_element, axis=1, arr=SRSF1_ENCSR321PWZ_weights.reshape(-1, 4)[1::2])

print('Average weights mean is ', SRSF1_ENCSR321PWZ_avg_weights.mean())
print('Average weights variance is ', SRSF1_ENCSR321PWZ_avg_weights.var())
```

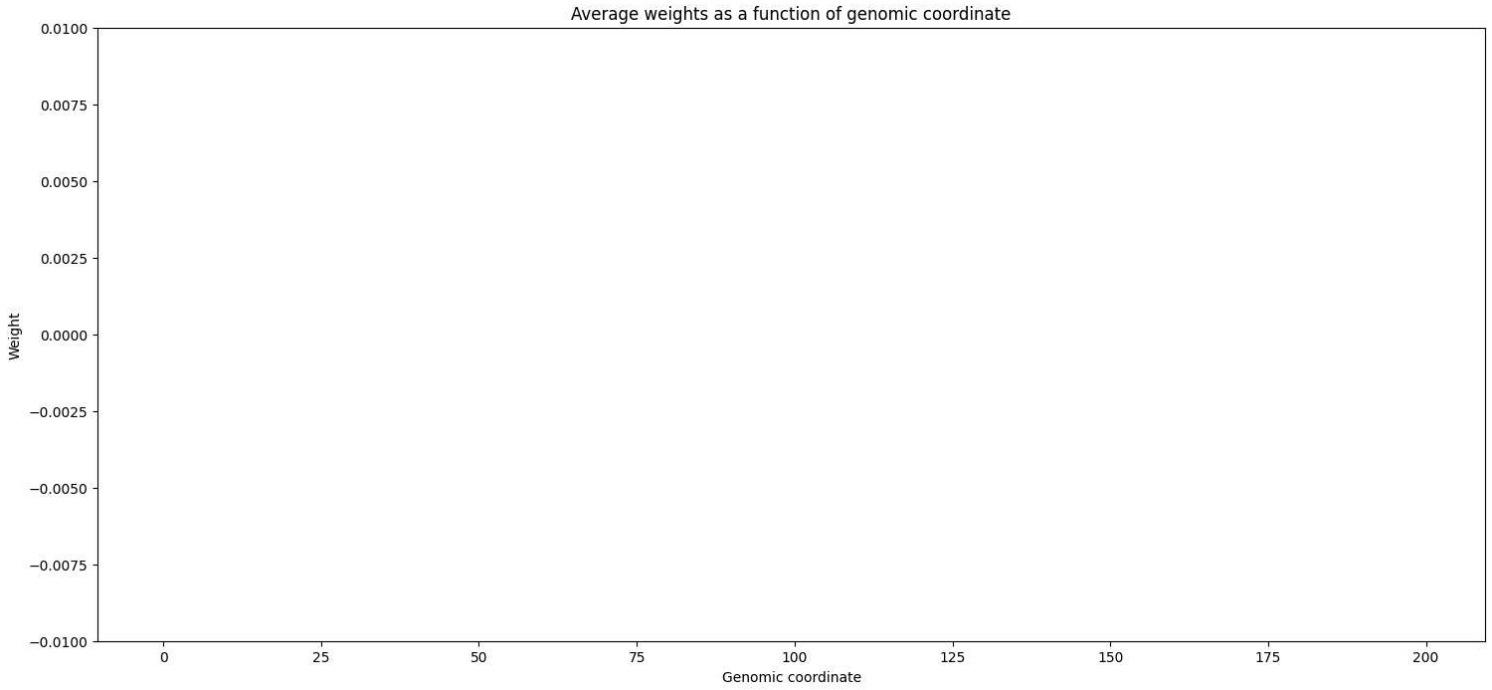
```
Average weights mean is  1.9766089806583232e-17
Average weights variance is  7.727104944614765e-35
```

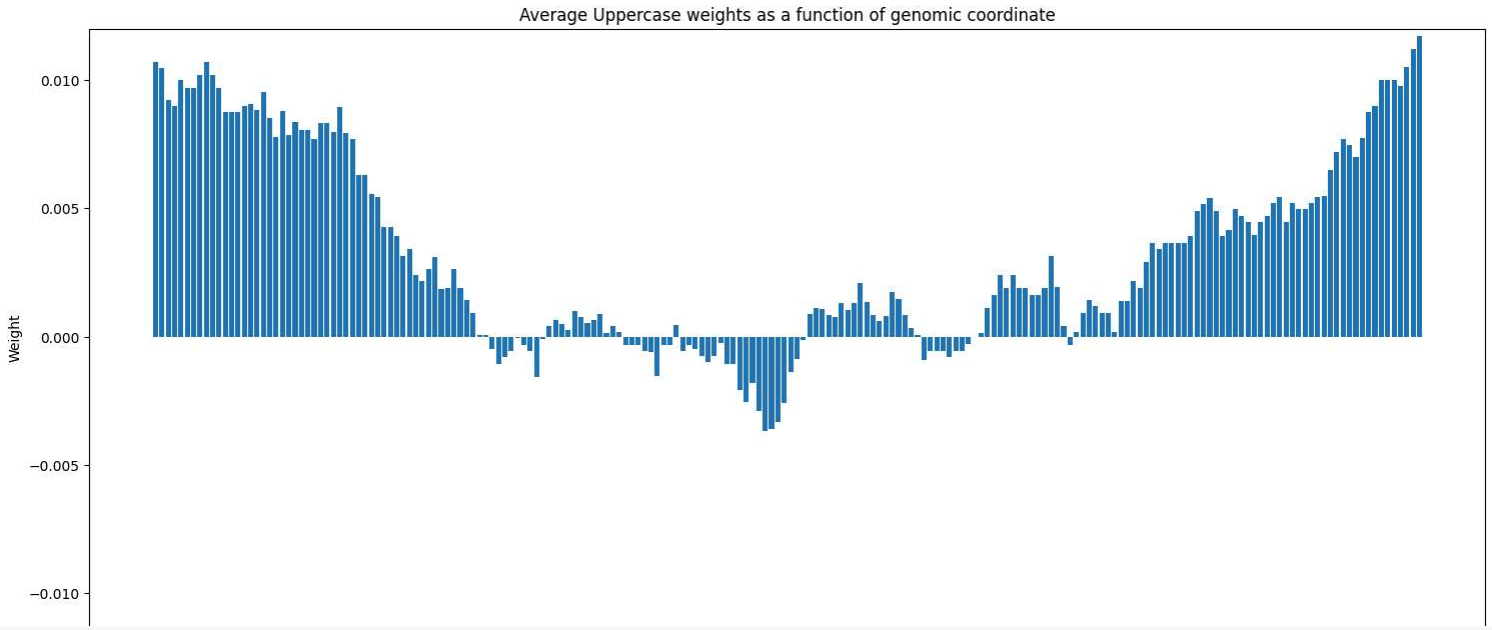
```
plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR321PWZ_weights)), SRSF1_ENCSR321PWZ_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR321PWZ_A_weights)), SRSF1_ENCSR321PWZ_A_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'A' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR321PWZ_a_weights)), SRSF1_ENCSR321PWZ_a_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'a' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR321PWZ_C_weights)), SRSF1_ENCSR321PWZ_C_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'C' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR321PWZ_c_weights)), SRSF1_ENCSR321PWZ_c_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'c' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR321PWZ_G_weights)), SRSF1_ENCSR321PWZ_G_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'G' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR321PWZ_g_weights)), SRSF1_ENCSR321PWZ_g_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'g' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR321PWZ_T_weights)), SRSF1_ENCSR321PWZ_T_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'T' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR321PWZ_t_weights)), SRSF1_ENCSR321PWZ_t_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'t' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()
```



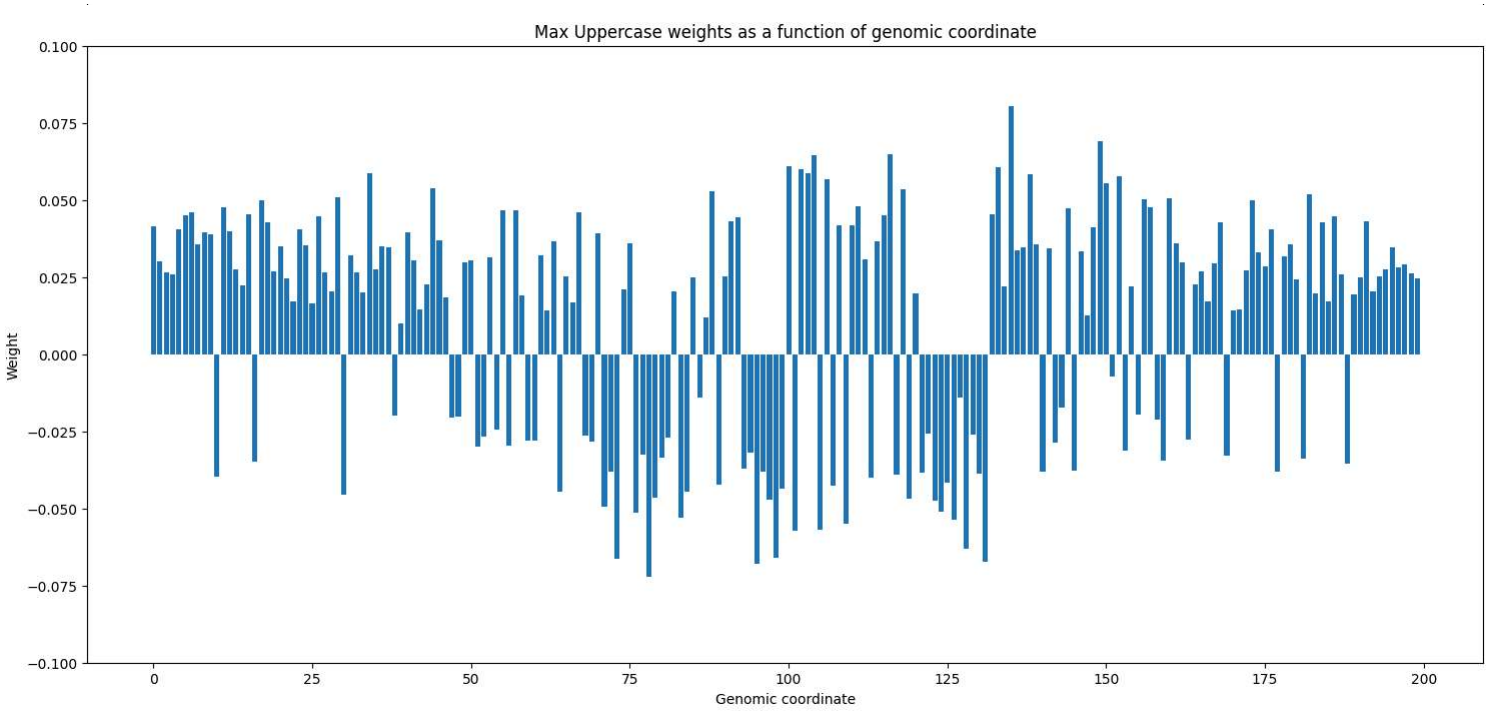


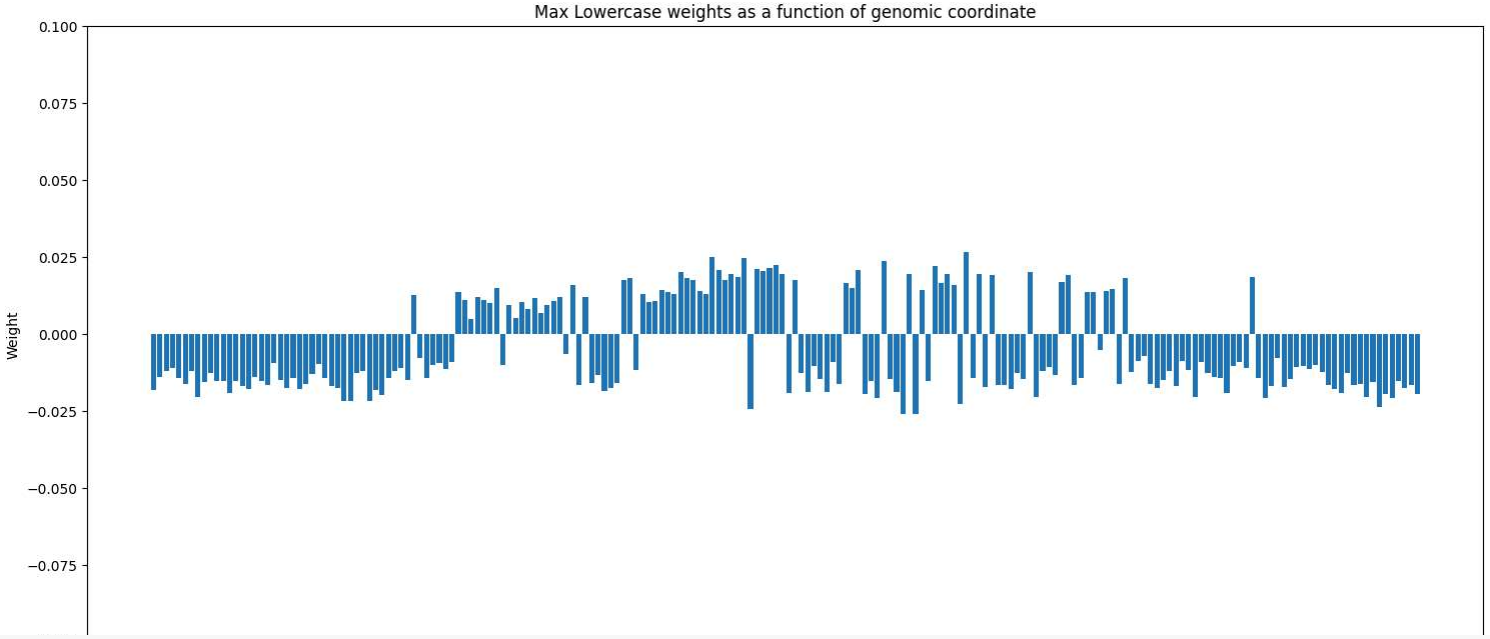

```
plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR321PWZ_avg_weights)), SRSF1_ENCSR321PWZ_avg_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Average weights as a function of genomic coordinate")
plt.ylim([-0.01, 0.01])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR321PWZ_avg_uppercase_weights)), SRSF1_ENCSR321PWZ_avg_uppercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Average Uppercase weights as a function of genomic coordinate")
plt.ylim([-0.012, 0.012])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR321PWZ_avg_lowercase_weights)), SRSF1_ENCSR321PWZ_avg_lowercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Average Lowercase weights as a function of genomic coordinate")
plt.ylim([-0.012, 0.012])
plt.show()
```



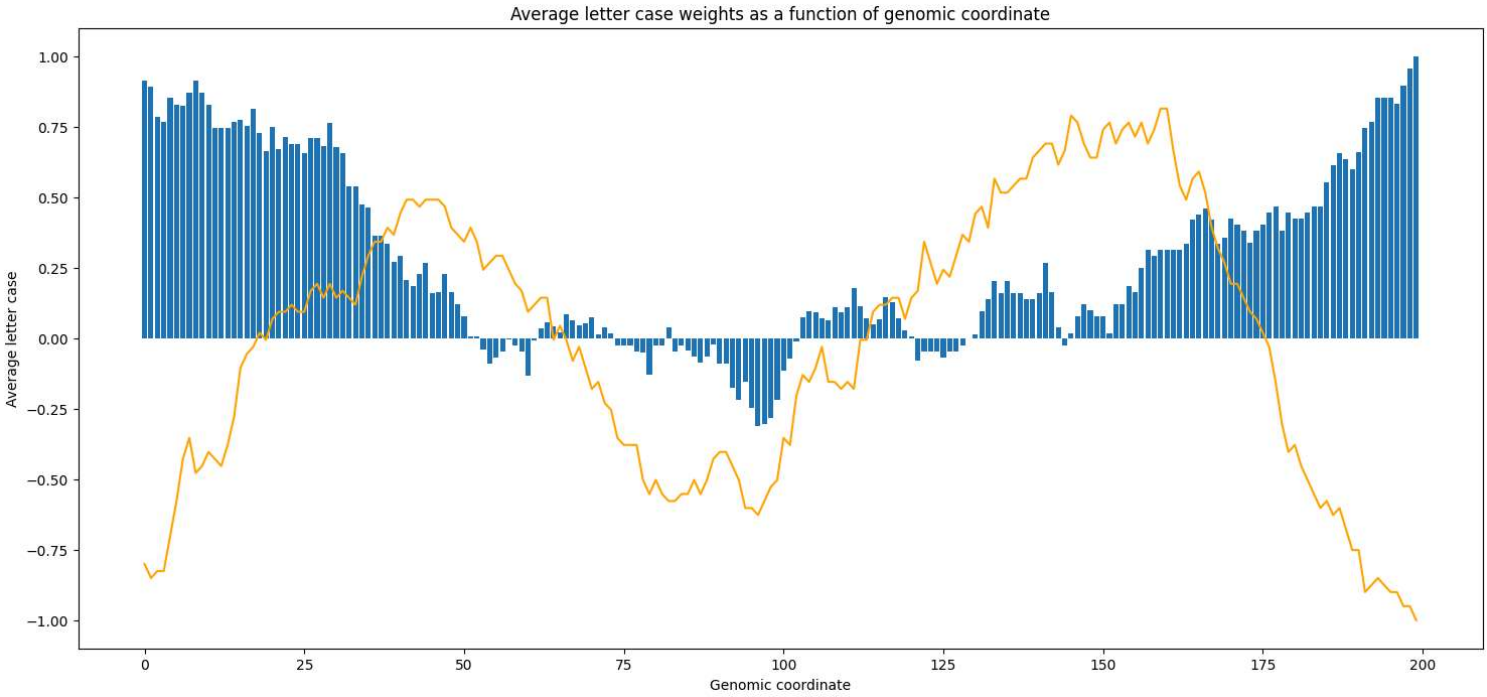



```
plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR321PWZ_max_uppercase_weights)), SRSF1_ENCSR321PWZ_max_uppercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Max Uppercase weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR321PWZ_max_lowercase_weights)), SRSF1_ENCSR321PWZ_max_lowercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Max Lowercase weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()
```



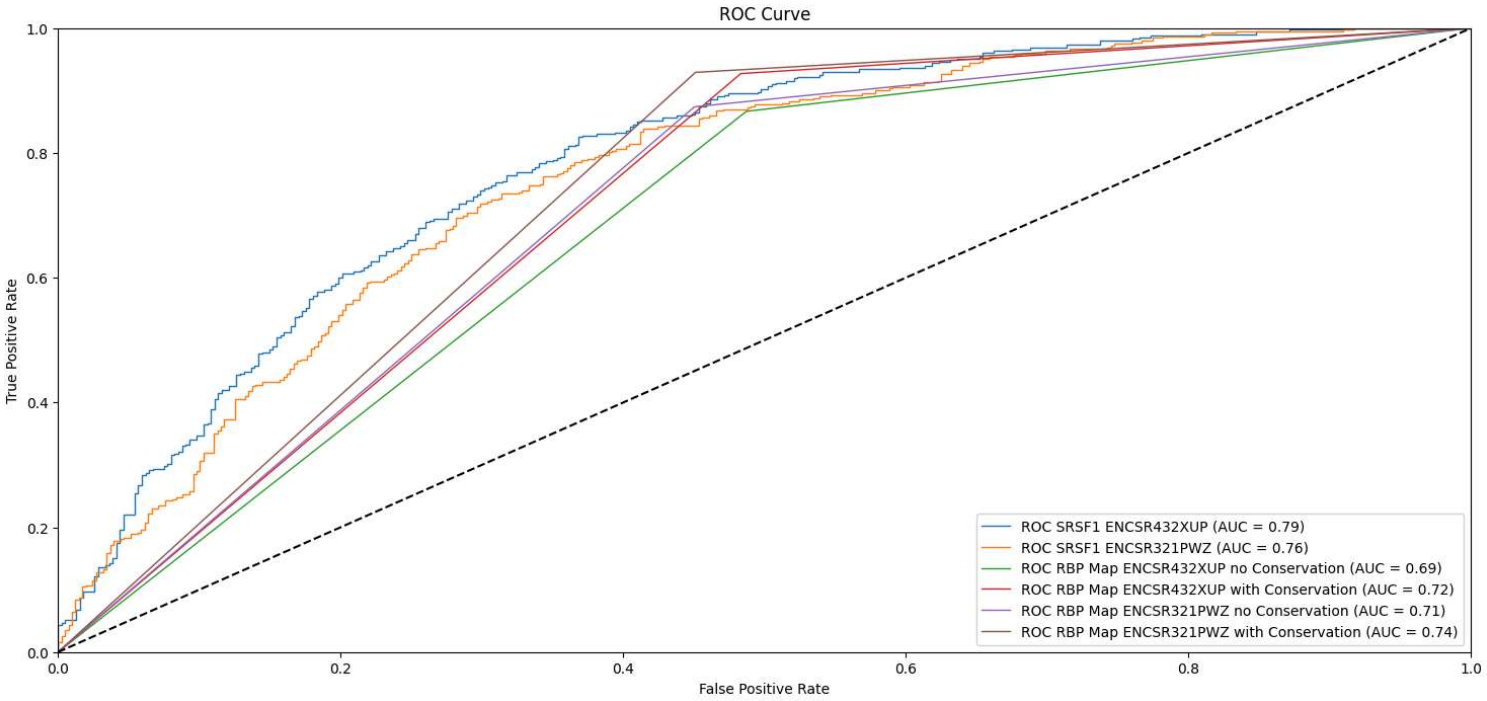



```
SRSF1_ENCSR321PWZ_mean_letter_case = np.array([calc_mean_letter_case(SRSF1_ENCSR321PWZ_ds.iloc[:, i]) for i in range(200)])
SRSF1_ENCSR321PWZ_mean_letter_case = (SRSF1_ENCSR321PWZ_mean_letter_case - SRSF1_ENCSR321PWZ_mean_letter_case.mean())
SRSF1_ENCSR321PWZ_mean_letter_case = SRSF1_ENCSR321PWZ_mean_letter_case / max(abs(SRSF1_ENCSR321PWZ_mean_letter_case))
normalized_SRSF1_ENCSR321PWZ_avg_uppercase_weights = SRSF1_ENCSR321PWZ_avg_uppercase_weights / max(abs(SRSF1_ENCSR321PWZ_avg_uppercase_weights))

plt.figure(figsize=(18, 8))
plt.plot(range(len(SRSF1_ENCSR321PWZ_mean_letter_case)), SRSF1_ENCSR321PWZ_mean_letter_case, color='orange')
plt.bar(range(len(normalized_SRSF1_ENCSR321PWZ_avg_uppercase_weights)), normalized_SRSF1_ENCSR321PWZ_avg_uppercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Average letter case')
plt.title("Average letter case weights as a function of genomic coordinate")
plt.show()
```



Compare experiments

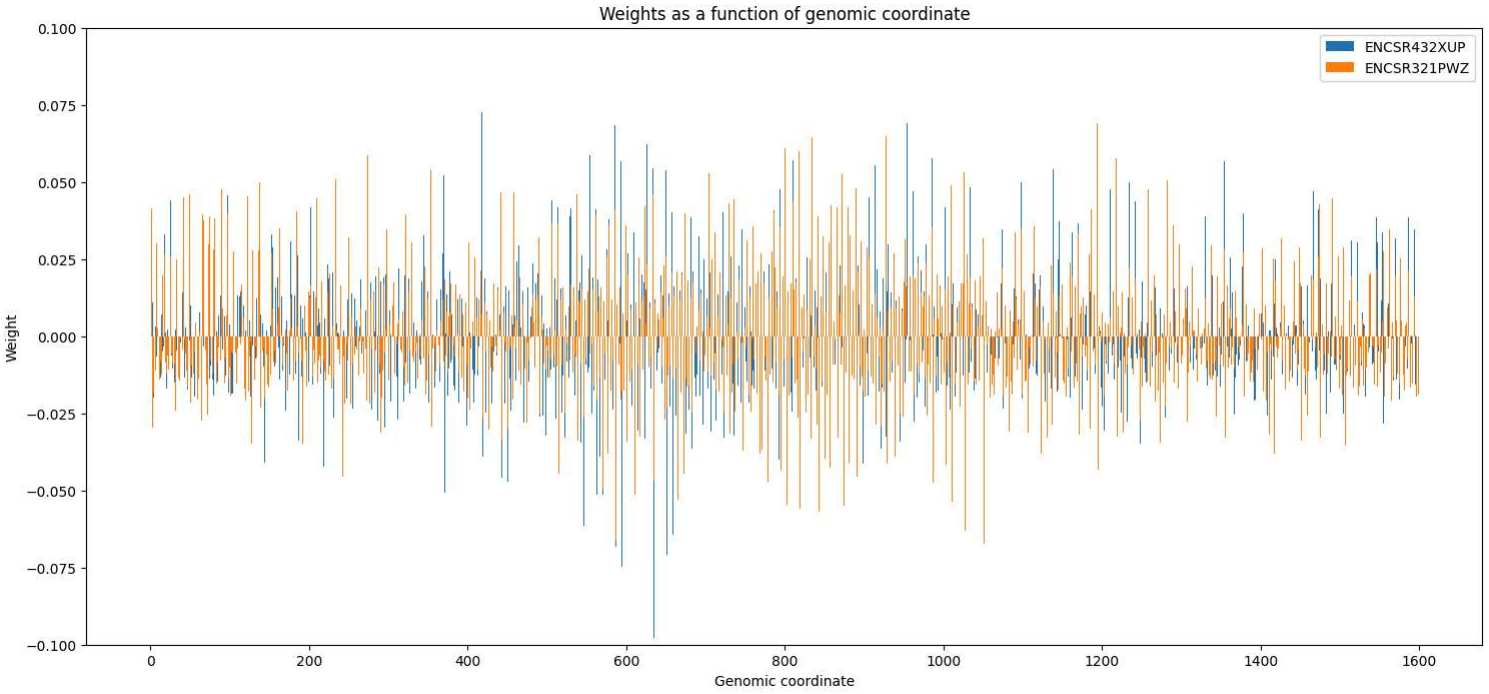
```
fig, ax = plt.subplots(figsize=(18, 8))

viz = RocCurveDisplay.from_estimator(
    SRSF1_ENCSR432XUP_model,
    SRSF1_ENCSR432XUP_test_features,
    SRSF1_ENCSR432XUP_test_labels,
    name=f"ROC SRSF1 ENCSR432XUP",
    lw=1,
    ax=ax
)

viz = RocCurveDisplay.from_estimator(
    SRSF1_ENCSR321PWZ_model,
    SRSF1_ENCSR321PWZ_test_features,
    SRSF1_ENCSR321PWZ_test_labels,
    name=f"ROC SRSF1 ENCSR321PWZ",
    lw=1,
    ax=ax
)

viz = RocCurveDisplay.from_predictions(
    SRSF1_ENCSR432XUP_rbp_no_conservation,
    SRSF1_ENCSR432XUP_true_labels,
    name=f"ROC RBP Map ENCSR432XUP no Conservation",
    lw=1,
    ax=ax
)

viz = RocCurveDisplay.from_predictions(
    SRSF1_ENCSR432XUP_rbp_with_conservation,
    SRSF1_ENCSR432XUP_true_labels,
    name=f"ROC RBP Map ENCSR432XUP with Conservation",
    lw=1,
    ax=ax
)

viz = RocCurveDisplay.from_predictions(
    SRSF1_ENCSR321PWZ_rbp_no_conservation,
    SRSF1_ENCSR321PWZ_true_labels,
    name=f"ROC RBP Map ENCSR321PWZ no Conservation",
    lw=1,
    ax=ax
)

viz = RocCurveDisplay.from_predictions(
    SRSF1_ENCSR321PWZ_rbp_with_conservation,
    SRSF1_ENCSR321PWZ_true_labels,
    name=f"ROC RBP Map ENCSR321PWZ with Conservation",
    lw=1,
    ax=ax
)

ax.set(
    xlabel="False Positive Rate",
    ylabel="True Positive Rate",
    title=f"ROC Curve",
)
ax.legend(loc="lower right")
ax.plot(np.linspace(0,1,10), np.linspace(0,1,10), '--',color='black')
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.show()
```

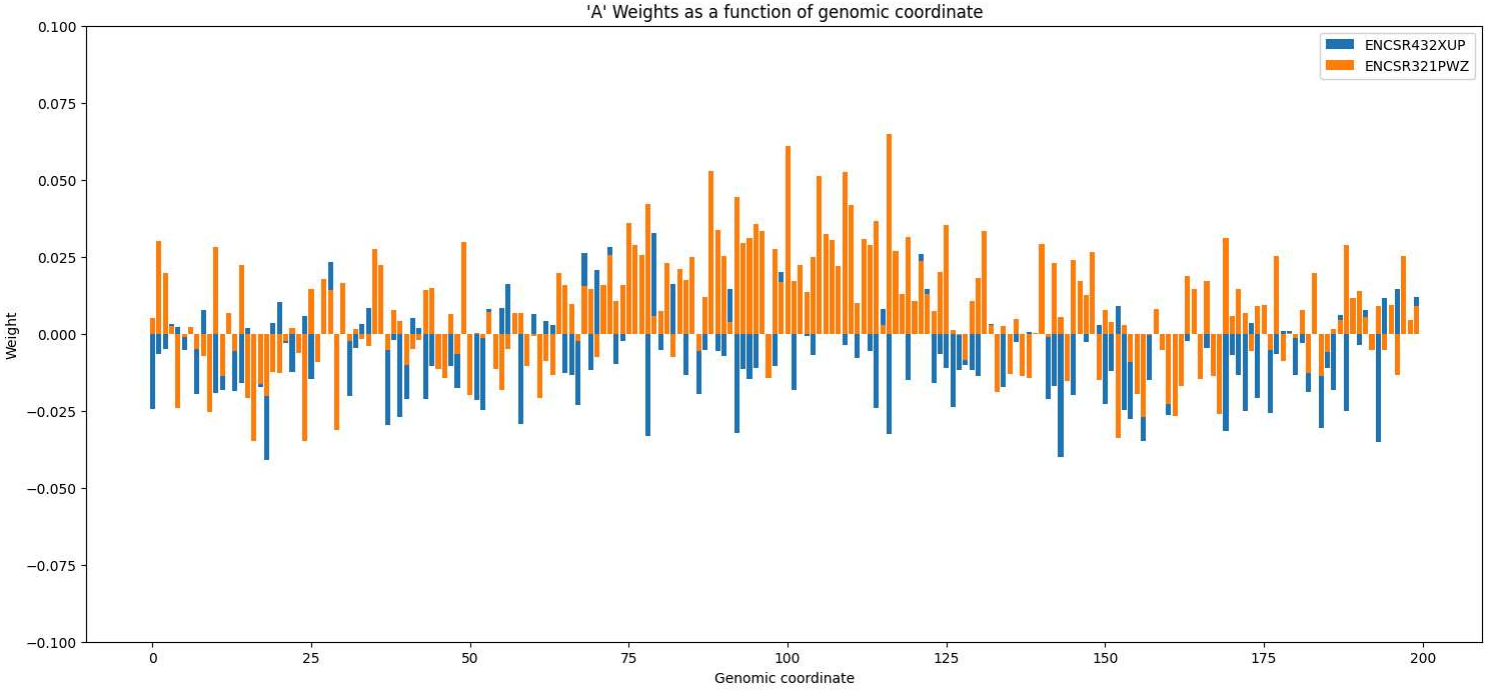


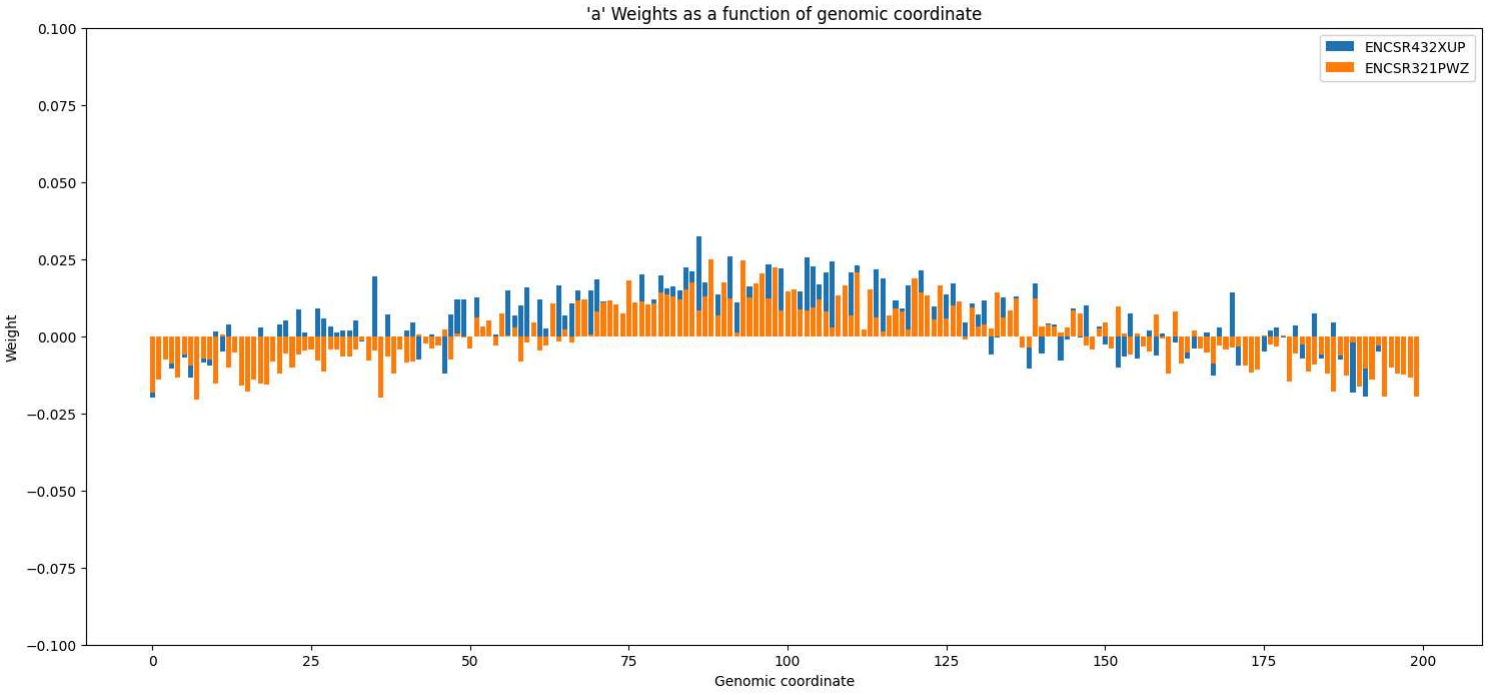
Compare model weights

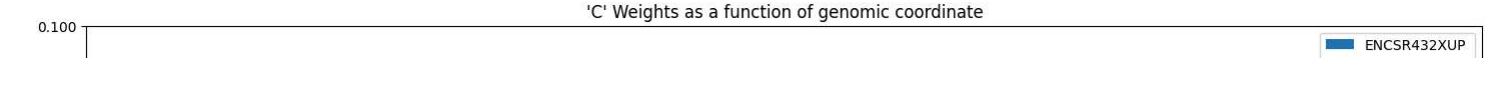
```
plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_weights)), SRSF1_ENCSR432XUP_weights)
plt.bar(range(len(SRSF1_ENCSR321PWZ_weights)), SRSF1_ENCSR321PWZ_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Weights as a function of genomic coordinate")
plt.legend(['ENCSR432XUP', 'ENCSR321PWZ'])
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_A_weights)), SRSF1_ENCSR432XUP_A_weights)
plt.bar(range(len(SRSF1_ENCSR321PWZ_A_weights)), SRSF1_ENCSR321PWZ_A_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'A' Weights as a function of genomic coordinate")
plt.legend(['ENCSR432XUP', 'ENCSR321PWZ'])
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_a_weights)), SRSF1_ENCSR432XUP_a_weights)
plt.bar(range(len(SRSF1_ENCSR321PWZ_a_weights)), SRSF1_ENCSR321PWZ_a_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'a' Weights as a function of genomic coordinate")
plt.legend(['ENCSR432XUP', 'ENCSR321PWZ'])
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_C_weights)), SRSF1_ENCSR432XUP_C_weights)
plt.bar(range(len(SRSF1_ENCSR321PWZ_C_weights)), SRSF1_ENCSR321PWZ_C_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'C' Weights as a function of genomic coordinate")
plt.legend(['ENCSR432XUP', 'ENCSR321PWZ'])
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_c_weights)), SRSF1_ENCSR432XUP_c_weights)
plt.bar(range(len(SRSF1_ENCSR321PWZ_c_weights)), SRSF1_ENCSR321PWZ_c_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'c' Weights as a function of genomic coordinate")
plt.legend(['ENCSR432XUP', 'ENCSR321PWZ'])
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_G_weights)), SRSF1_ENCSR432XUP_G_weights)
plt.bar(range(len(SRSF1_ENCSR321PWZ_G_weights)), SRSF1_ENCSR321PWZ_G_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'G' Weights as a function of genomic coordinate")
plt.legend(['ENCSR432XUP', 'ENCSR321PWZ'])
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_g_weights)), SRSF1_ENCSR432XUP_g_weights)
plt.bar(range(len(SRSF1_ENCSR321PWZ_g_weights)), SRSF1_ENCSR321PWZ_g_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'g' Weights as a function of genomic coordinate")
plt.legend(['ENCSR432XUP', 'ENCSR321PWZ'])
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_T_weights)), SRSF1_ENCSR432XUP_T_weights)
plt.bar(range(len(SRSF1_ENCSR321PWZ_T_weights)), SRSF1_ENCSR321PWZ_T_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'T' Weights as a function of genomic coordinate")
plt.legend(['ENCSR432XUP', 'ENCSR321PWZ'])
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_t_weights)), SRSF1_ENCSR432XUP_t_weights)
plt.bar(range(len(SRSF1_ENCSR321PWZ_t_weights)), SRSF1_ENCSR321PWZ_t_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
```

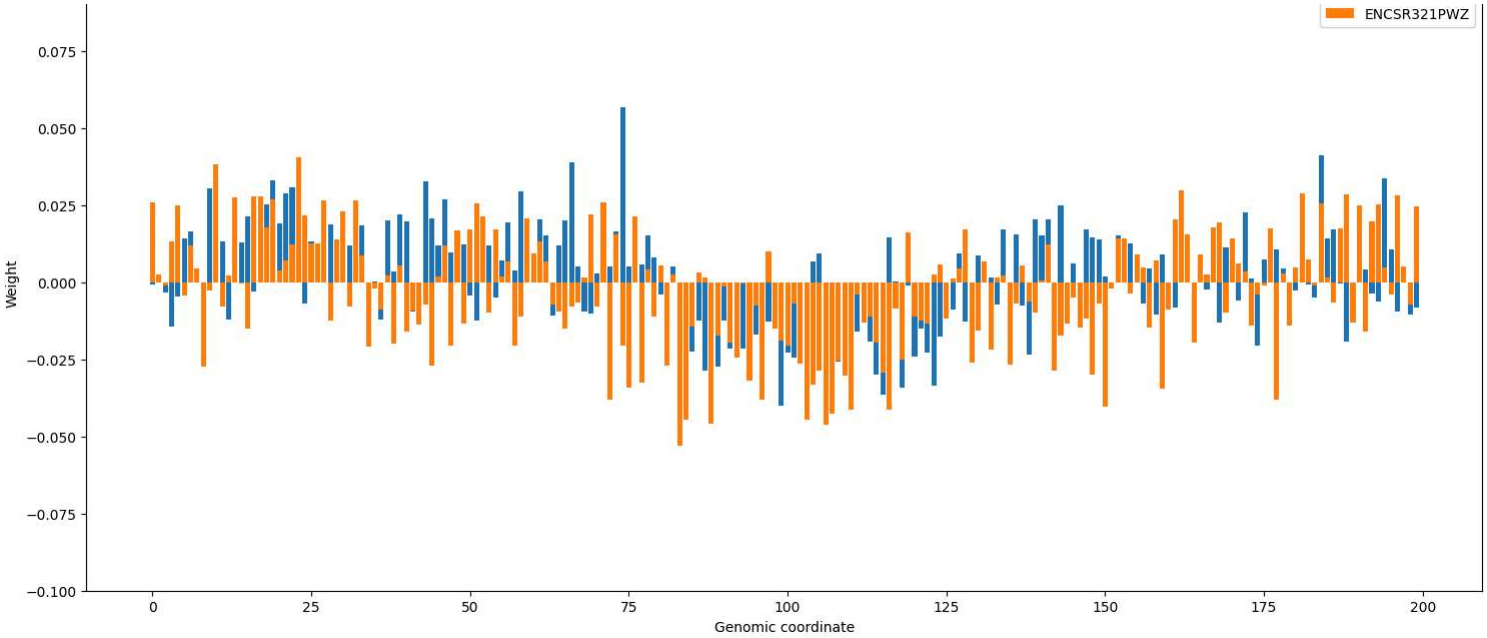
```
plt.title("'t' Weights as a function of genomic coordinate")
plt.legend(['ENCSR432XUP', 'ENCSR321PWZ'])
plt.ylim([-0.1, 0.1])
plt.show()
```


Weights as a function of genomic coordinate
ENCSR432XUP
ENCSR321PWZ
Weight
Genomic coordinate

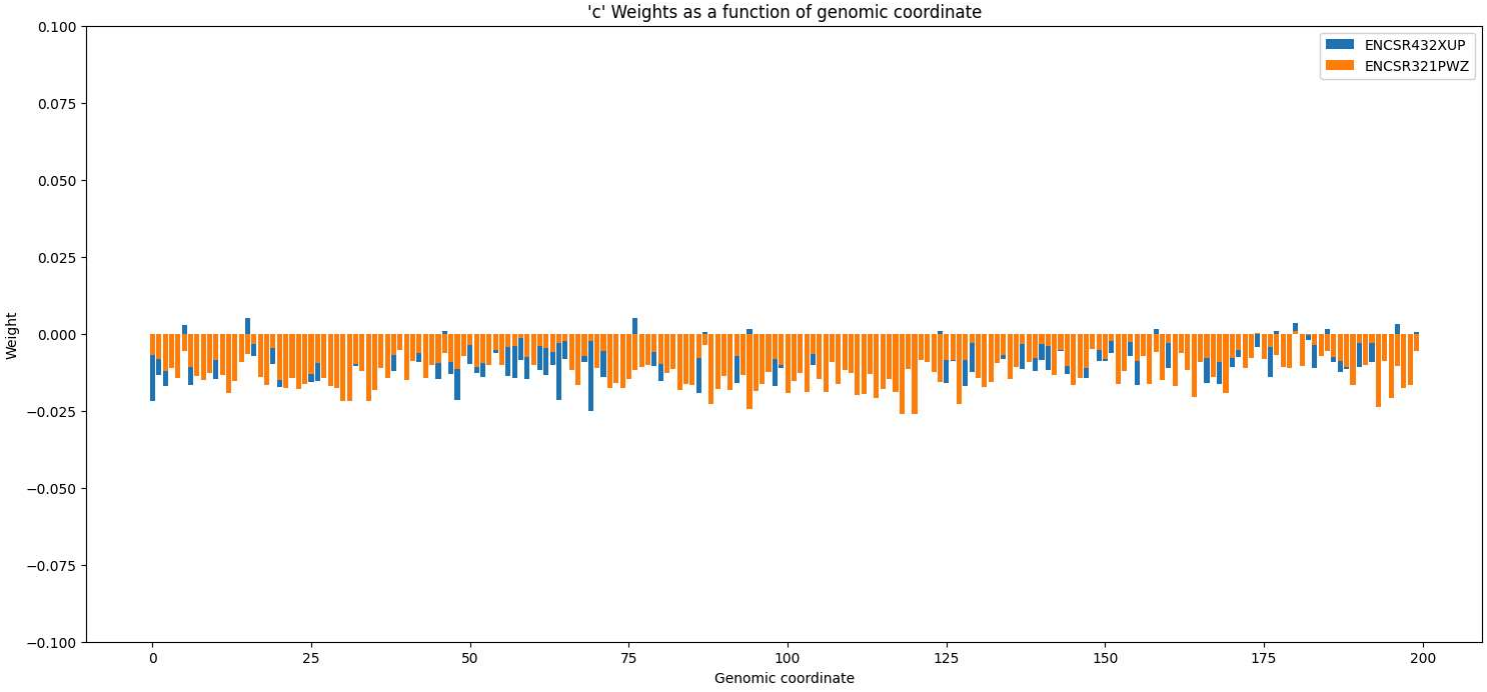

'A' Weights as a function of genomic coordinate
ENCSR432XUP
ENCSR321PWZ
Weight
Genomic coordinate


'a' Weights as a function of genomic coordinate
ENCSR432XUP
ENCSR321PWZ
Weight
Genomic coordinate

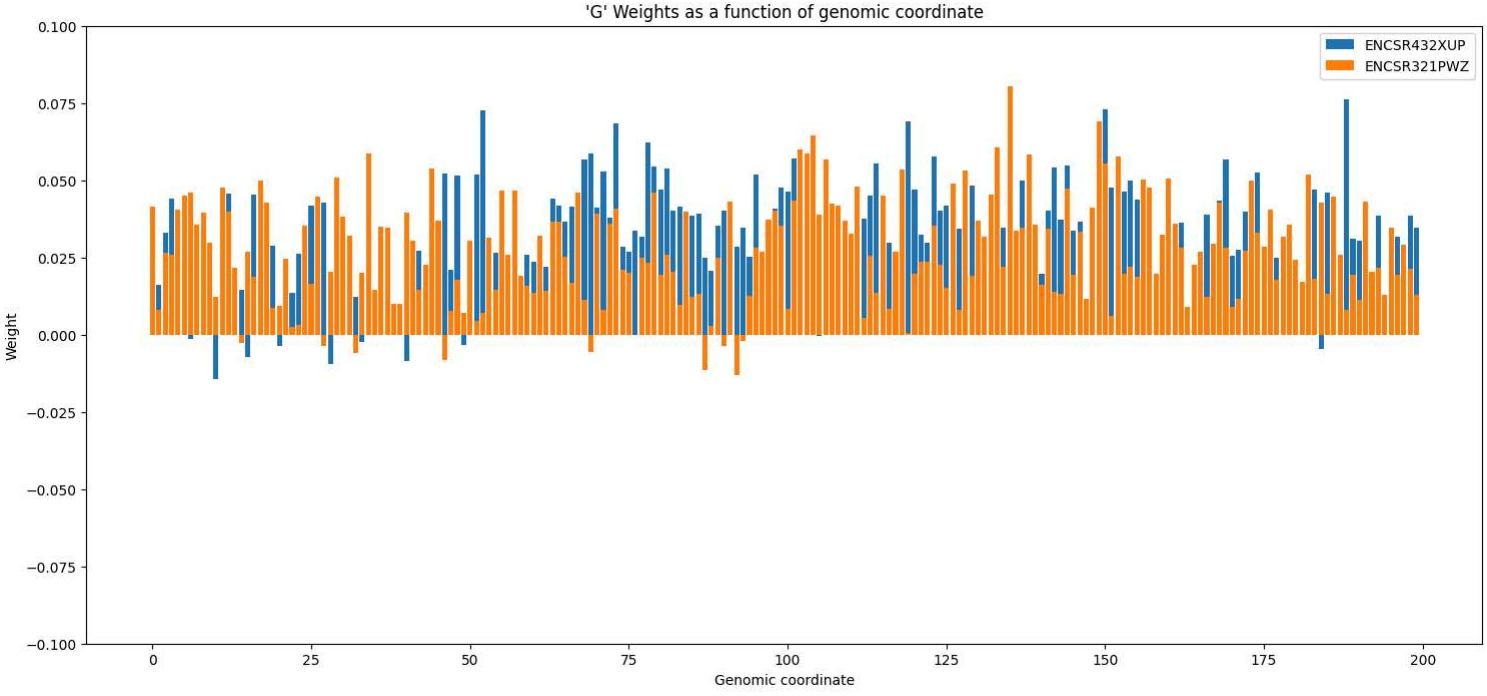

'C' Weights as a function of genomic coordinate
ENCSR432XUP



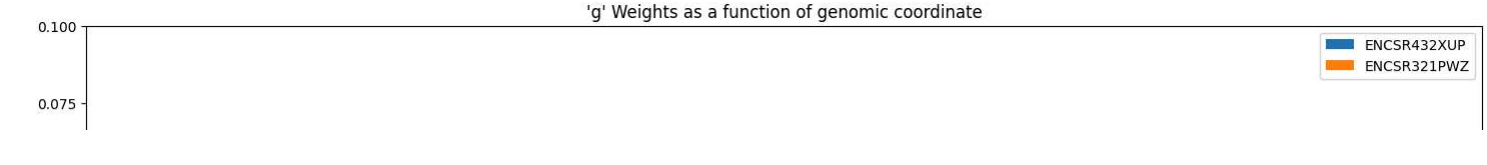
'c' Weights as a function of genomic coordinate

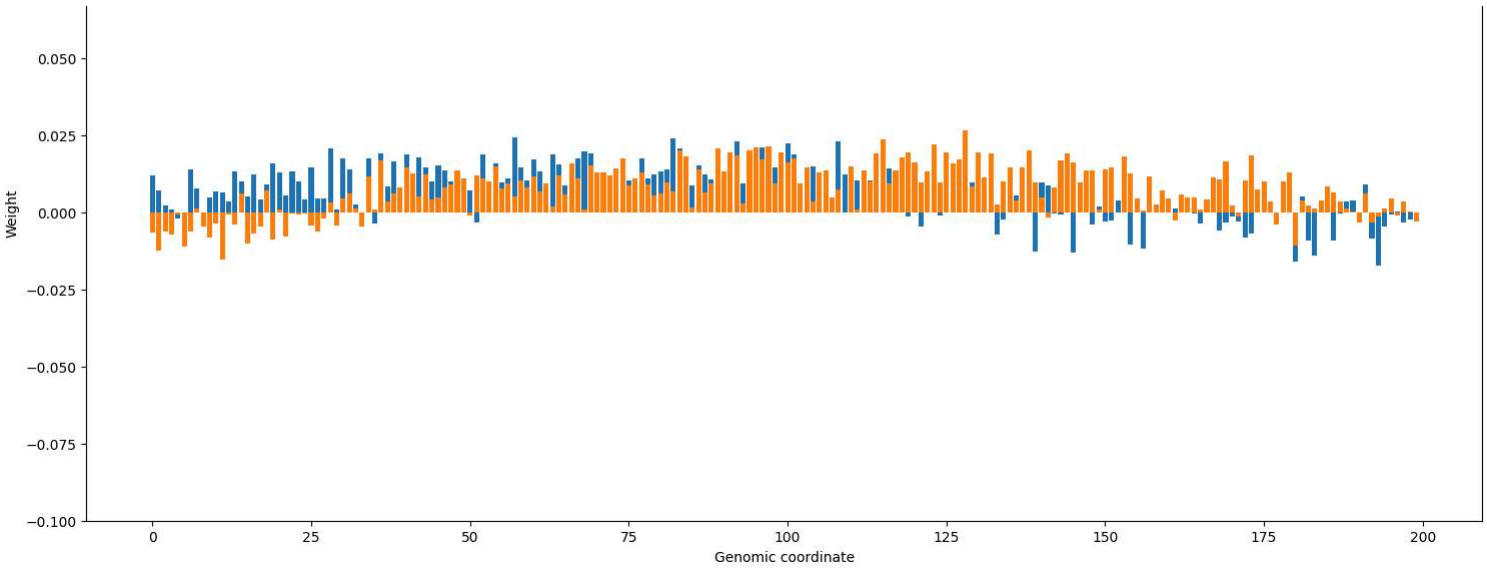


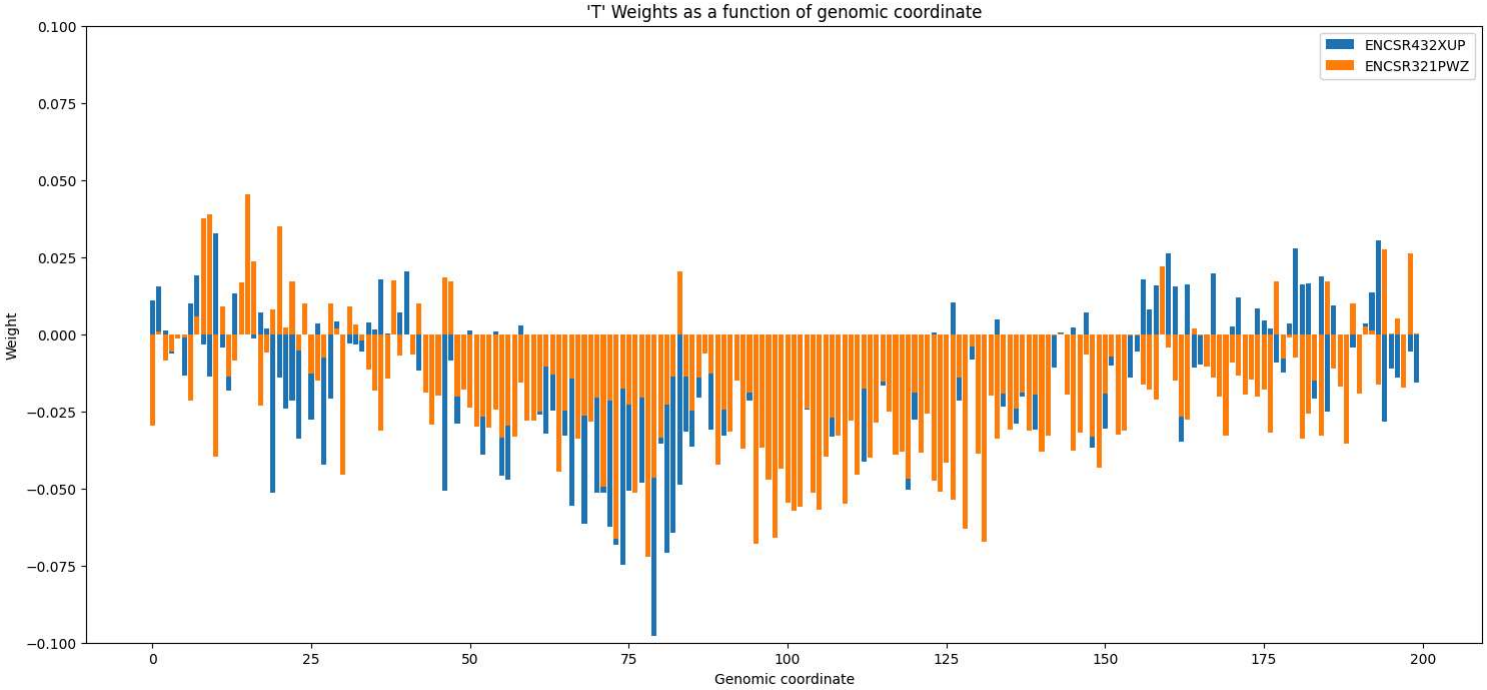
'G' Weights as a function of genomic coordinate

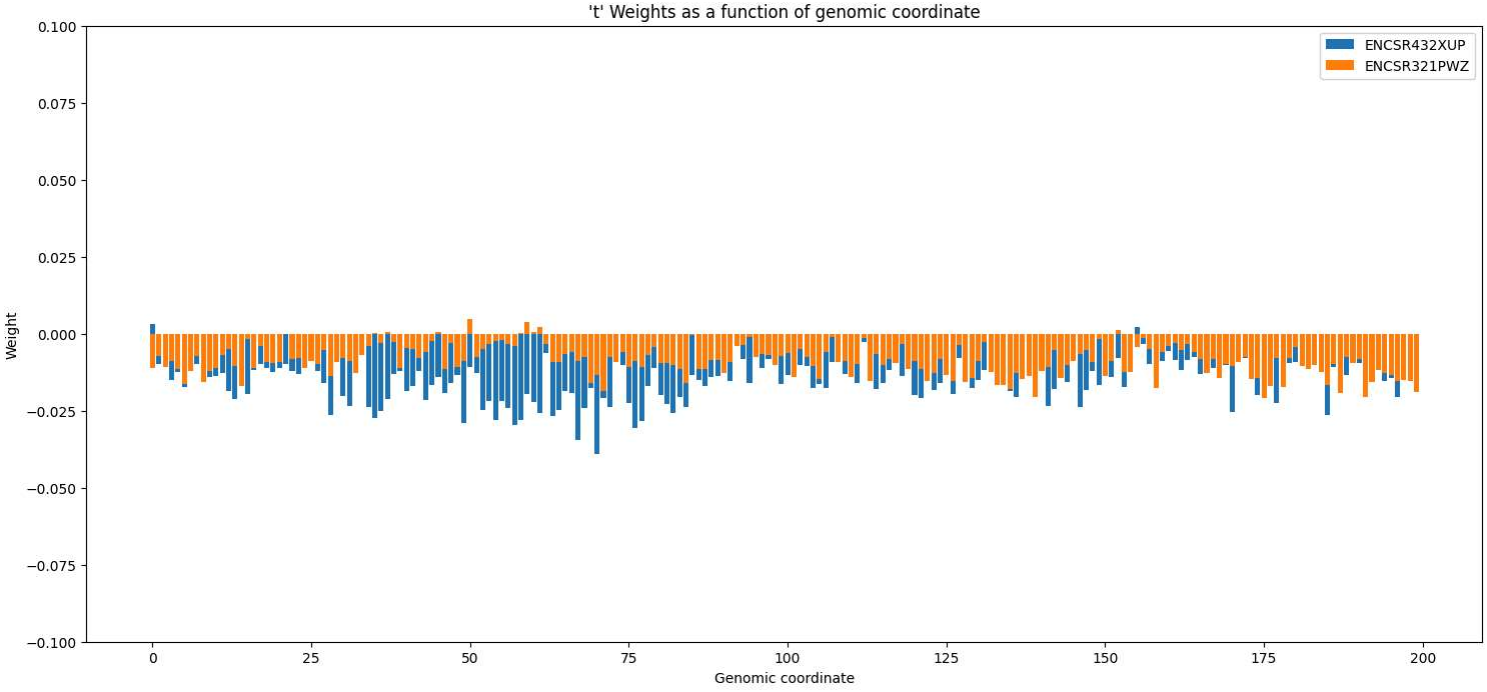


'g' Weights as a function of genomic coordinate

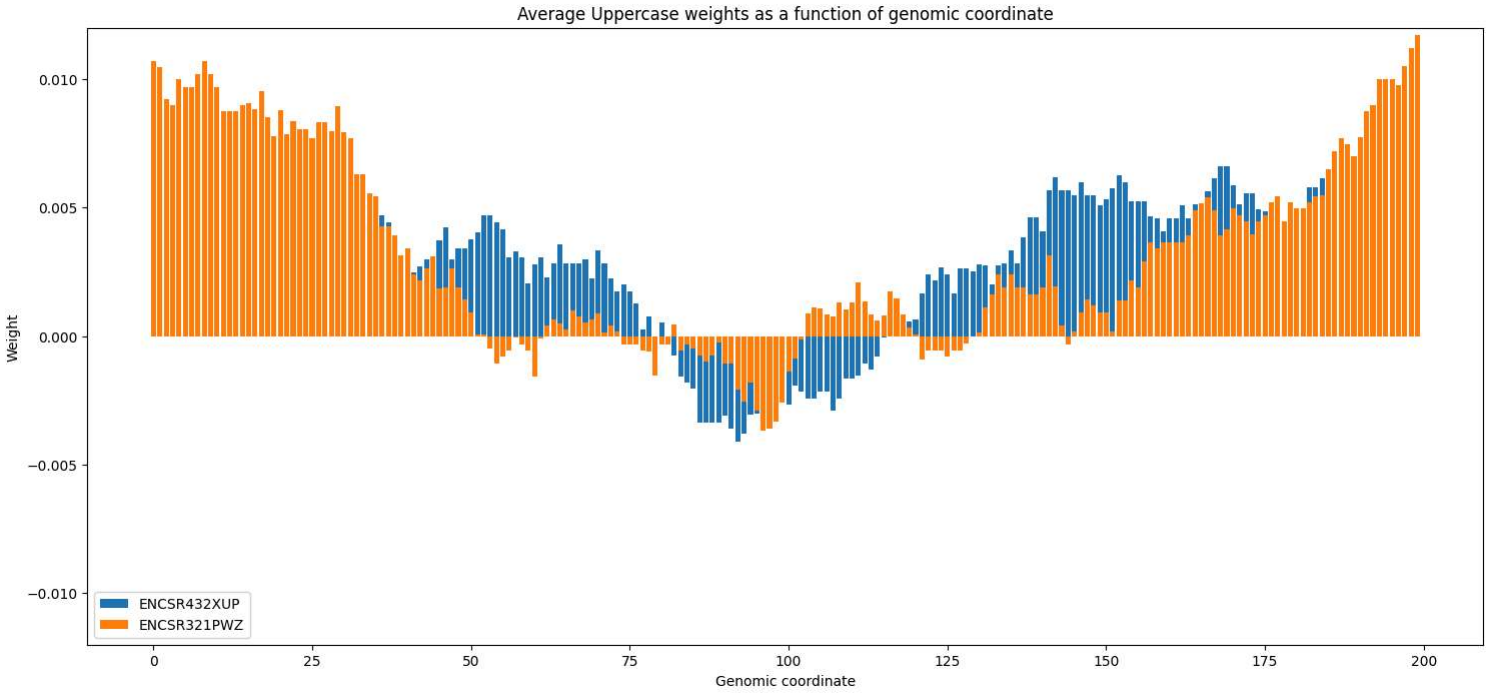


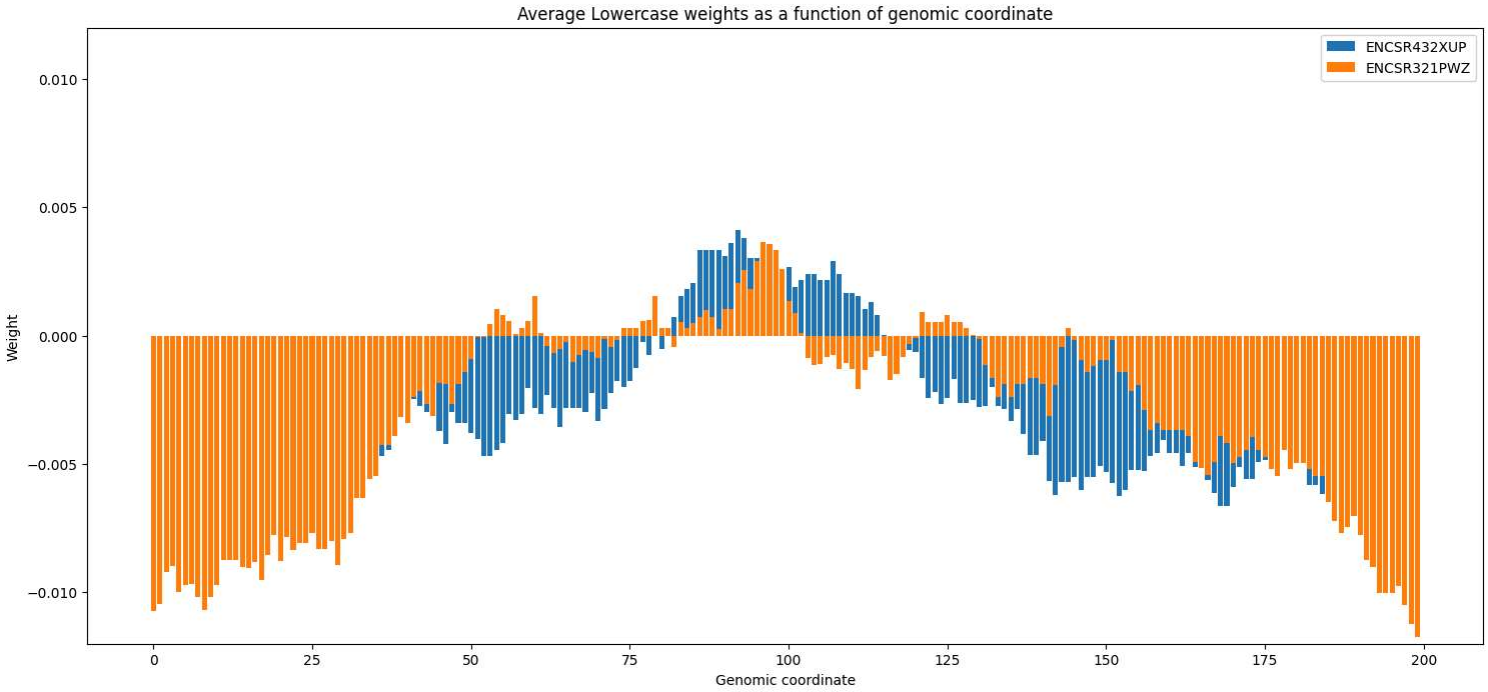

0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
Weight
0
25
50
75
100
125
150
175
200
Genomic coordinate


'T' Weights as a function of genomic coordinate
ENCSR432XUP
ENCSR321PWZ
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
Weight
0
25
50
75
100
125
150
175
200
Genomic coordinate


't' Weights as a function of genomic coordinate
ENCSR432XUP
ENCSR321PWZ
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
Weight
0
25
50
75
100
125
150
175
200
Genomic coordinate

```
plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_avg_uppercase_weights)), SRSF1_ENCSR432XUP_avg_uppercase_weights)
plt.bar(range(len(SRSF1_ENCSR321PWZ_avg_uppercase_weights)), SRSF1_ENCSR321PWZ_avg_uppercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Average Uppercase weights as a function of genomic coordinate")
plt.legend(['ENCSR432XUP', 'ENCSR321PWZ'])
plt.ylim([-0.012, 0.012])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(SRSF1_ENCSR432XUP_avg_lowercase_weights)), SRSF1_ENCSR432XUP_avg_lowercase_weights)
plt.bar(range(len(SRSF1_ENCSR321PWZ_avg_lowercase_weights)), SRSF1_ENCSR321PWZ_avg_lowercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Average Lowercase weights as a function of genomic coordinate")
plt.legend(['ENCSR432XUP', 'ENCSR321PWZ'])
plt.ylim([-0.012, 0.012])
plt.show()
```





PUM2

ENCSR661ICQ

Import dataset and fit model

```
PUM2_ENCSR661ICQ_ds = import_dataset()
[PUM2_ENCSR661ICQ_train, PUM2_ENCSR661ICQ_test] = train_test_split(PUM2_ENCSR661ICQ_ds,train_size=0.8, random_state=103)

# One-Hot Encoding
PUM2_ENCSR661ICQ_train_features = pd.get_dummies(PUM2_ENCSR661ICQ_train.iloc[:, 0:200]).to_numpy()
PUM2_ENCSR661ICQ_train_labels = PUM2_ENCSR661ICQ_train['label'].to_numpy()

PUM2_ENCSR661ICQ_test_features = pd.get_dummies(PUM2_ENCSR661ICQ_test.iloc[:, 0:200]).to_numpy()
PUM2_ENCSR661ICQ_test_labels = PUM2_ENCSR661ICQ_test['label'].to_numpy()

# C=1e-3 worked best
PUM2_ENCSR661ICQ_model = svm.SVC(C=1e-3, kernel="linear")
PUM2_ENCSR661ICQ_model = PUM2_ENCSR661ICQ_model.fit(PUM2_ENCSR661ICQ_train_features, PUM2_ENCSR661ICQ_train_labels)
```

```
יש לבחור קבצים  לא נבחר קובץ   Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving PUM2_ENCSR661ICQ_dataset.txt to PUM2_ENCSR661ICQ_dataset.txt
User uploaded file "PUM2_ENCSR661ICQ_dataset.txt" with length 816000 bytes
```

Import RBPmap predictions

```
uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))

with open(fn, 'r') as opened_file:
    txt_lines = opened_file.readlines()
txt_lines = txt_lines[1:]

PUM2_ENCSR661ICQ_rbp_no_conservation = np.array([int(line.split('\t')[0]) for line in txt_lines])
PUM2_ENCSR661ICQ_true_labels = np.array([int(line.split('\t')[1]) for line in txt_lines])
```

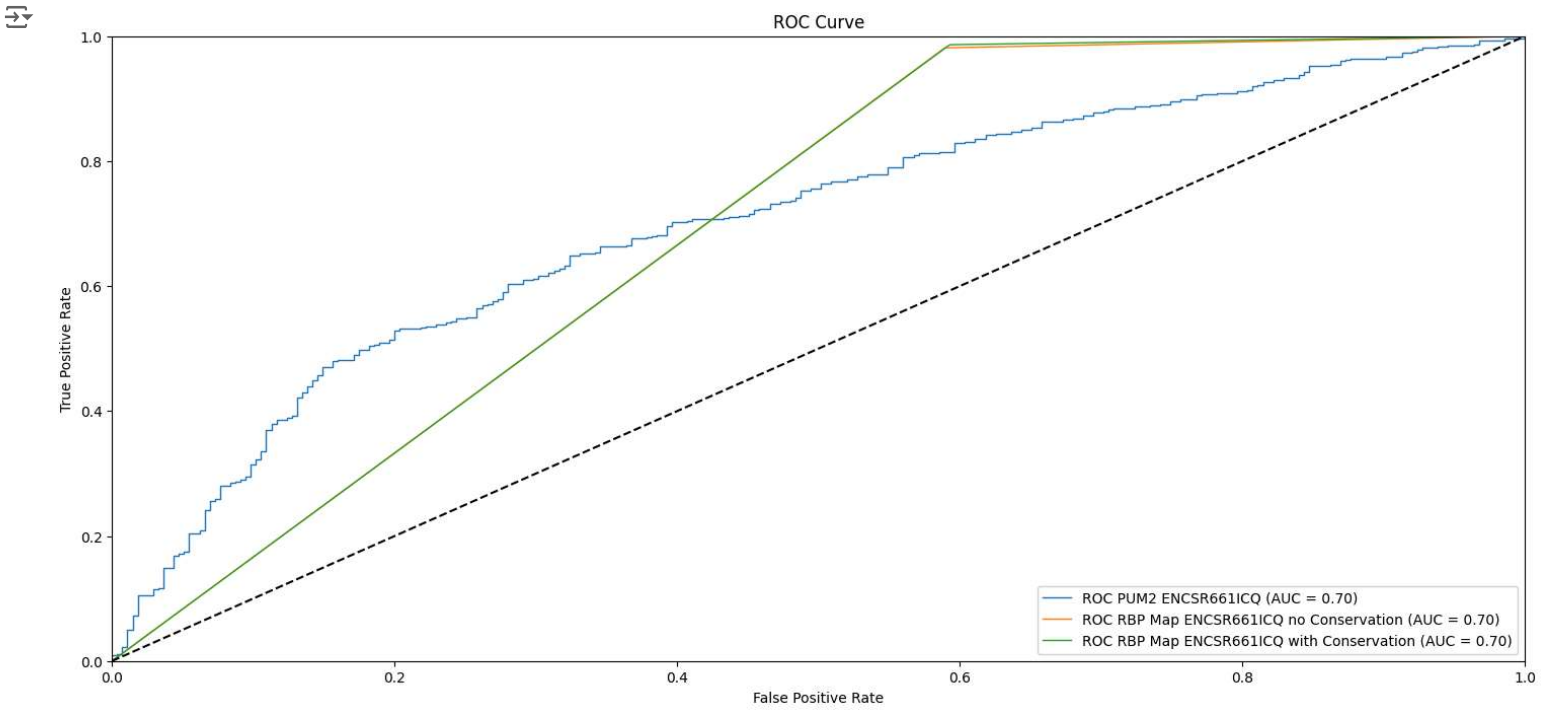
```
יש לבחור קבצים  לא נבחר קובץ   Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving PUM2_ENCSR661ICQ_rbp_predictions_no_conservation.txt to PUM2_ENCSR661ICQ_rbp_predictions_no_conservation.txt
User uploaded file "PUM2_ENCSR661ICQ_rbp_predictions_no_conservation.txt" with length 20013 bytes
```

```
uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))

with open(fn, 'r') as opened_file:
    txt_lines = opened_file.readlines()
txt_lines = txt_lines[1:]

PUM2_ENCSR661ICQ_rbp_with_conservation = np.array([int(line.split('\t')[0]) for line in txt_lines])
```

```
יש לבחור קבצים  לא נבחר קובץ   Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving PUM2_ENCSR661ICQ_rbp_predictions_with_conservation.txt to PUM2_ENCSR661ICQ_rbp_predictions_with_conservation.txt
User uploaded file "PUM2_ENCSR661ICQ_rbp_predictions_with_conservation.txt" with length 20013 bytes
```

Display results

```
fig, ax = plt.subplots(figsize=(18, 8))

viz = RocCurveDisplay.from_estimator(
    PUM2_ENCSR661ICQ_model,
    PUM2_ENCSR661ICQ_test_features,
    PUM2_ENCSR661ICQ_test_labels,
    name=f"ROC PUM2 ENCSR661ICQ",
    lw=1,
    ax=ax
)

viz = RocCurveDisplay.from_predictions(
    PUM2_ENCSR661ICQ_rbp_no_conservation,
    PUM2_ENCSR661ICQ_true_labels,
    name=f"ROC RBP Map ENCSR661ICQ no Conservation",
    lw=1,
    ax=ax
)

viz = RocCurveDisplay.from_predictions(
    PUM2_ENCSR661ICQ_rbp_with_conservation,
    PUM2_ENCSR661ICQ_true_labels,
    name=f"ROC RBP Map ENCSR661ICQ with Conservation",
    lw=1,
    ax=ax
)

ax.set(
    xlabel="False Positive Rate",
    ylabel="True Positive Rate",
    title=f"ROC Curve",
)
ax.legend(loc="lower right")
ax.plot(np.linspace(0,1,10), np.linspace(0,1,10), '--',color='black')
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.show()
```

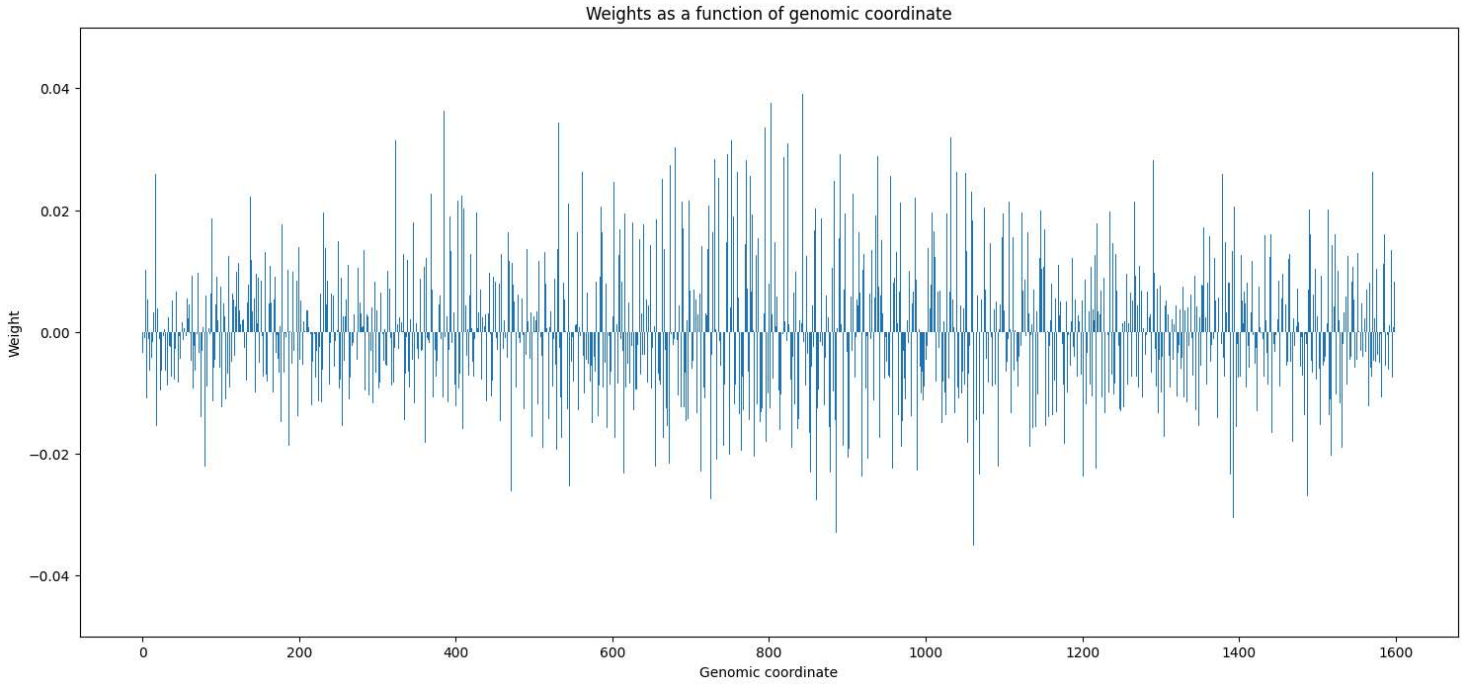


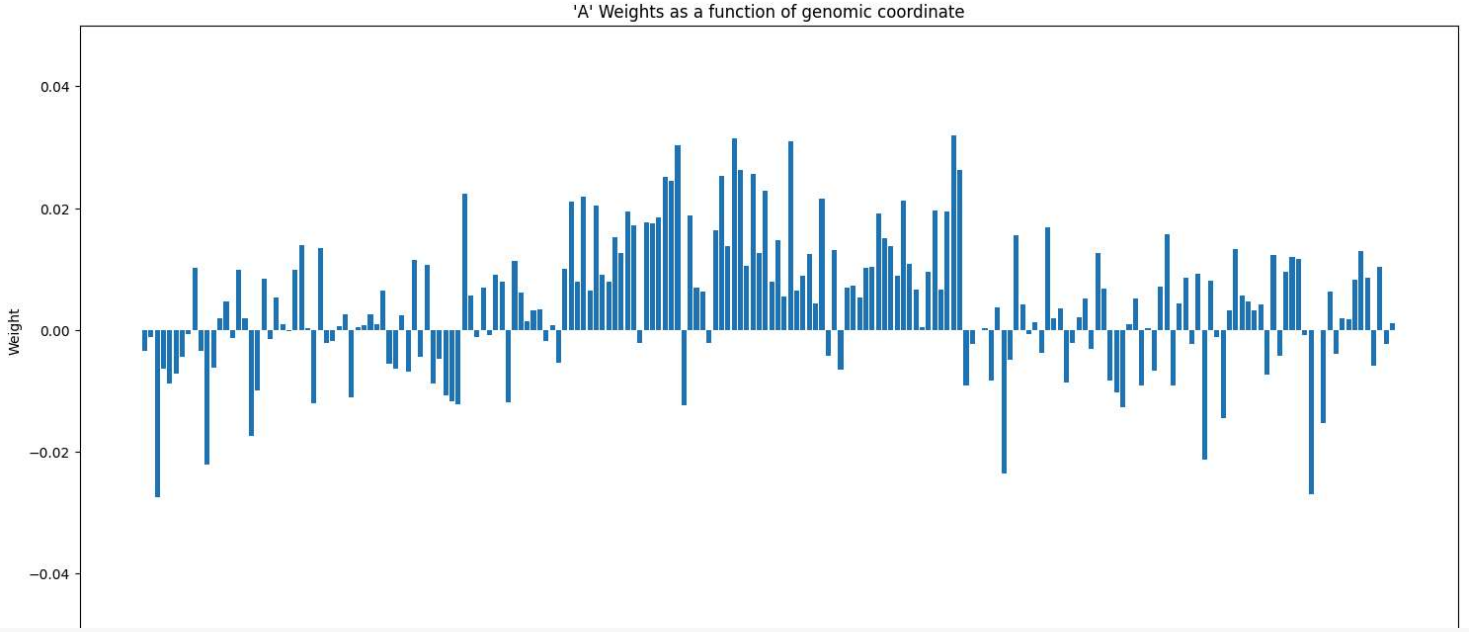
Display trained weights

```
PUM2_ENCSR661ICQ_weights = PUM2_ENCSR661ICQ_model.coef_[0]
PUM2_ENCSR661ICQ_A_weights = PUM2_ENCSR661ICQ_weights[::8]
PUM2_ENCSR661ICQ_a_weights = PUM2_ENCSR661ICQ_weights[4::8]
PUM2_ENCSR661ICQ_C_weights = PUM2_ENCSR661ICQ_weights[1::8]
PUM2_ENCSR661ICQ_c_weights = PUM2_ENCSR661ICQ_weights[5::8]
PUM2_ENCSR661ICQ_G_weights = PUM2_ENCSR661ICQ_weights[2::8]
PUM2_ENCSR661ICQ_g_weights = PUM2_ENCSR661ICQ_weights[6::8]
PUM2_ENCSR661ICQ_T_weights = PUM2_ENCSR661ICQ_weights[3::8]
PUM2_ENCSR661ICQ_t_weights = PUM2_ENCSR661ICQ_weights[7::8]

PUM2_ENCSR661ICQ_avg_weights = PUM2_ENCSR661ICQ_weights.reshape(-1, 8).mean(axis=1)
PUM2_ENCSR661ICQ_avg_uppercase_weights = PUM2_ENCSR661ICQ_weights.reshape(-1, 4).mean(axis=1)[::2]
PUM2_ENCSR661ICQ_avg_lowercase_weights = PUM2_ENCSR661ICQ_weights.reshape(-1, 4).mean(axis=1)[1::2]

print('Average weights mean is ', PUM2_ENCSR661ICQ_avg_weights.mean())
print('Average weights variance is ', PUM2_ENCSR661ICQ_avg_weights.var())
```

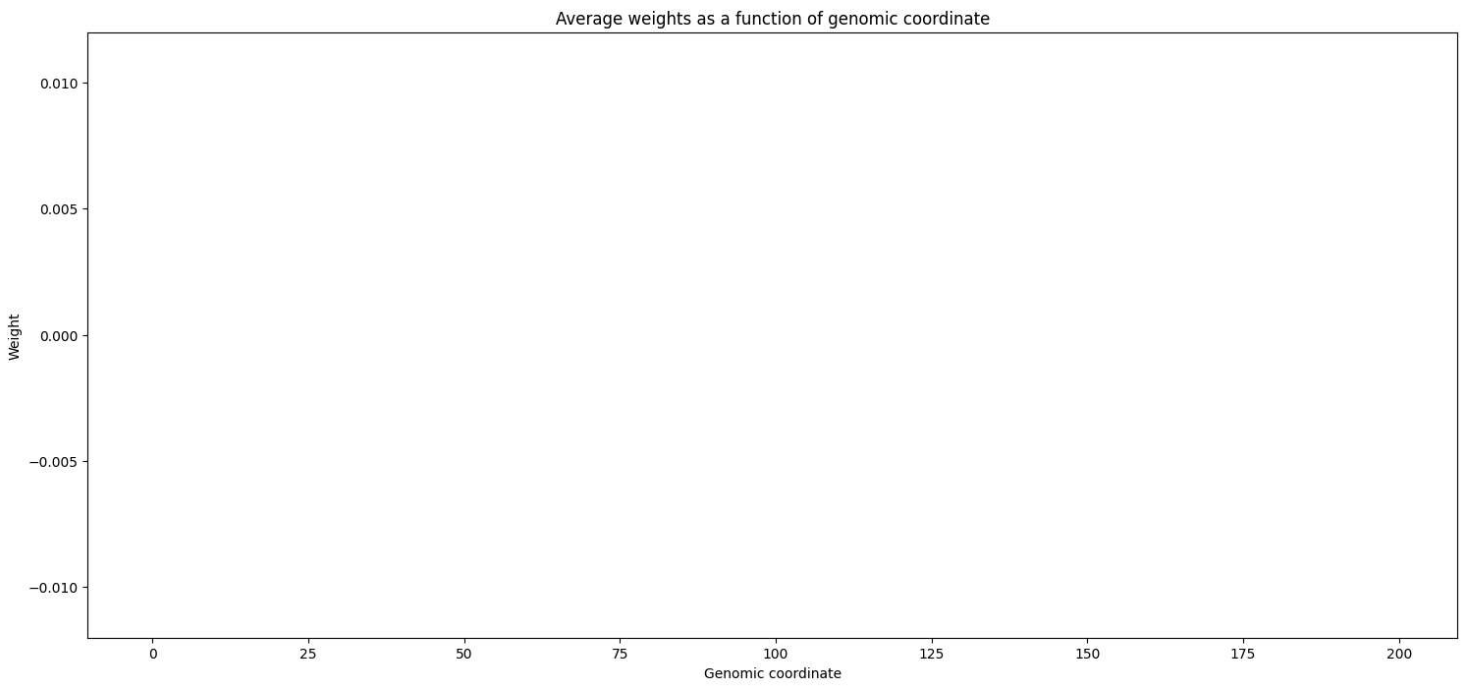
```
Average weights mean is  -1.522003009735151e-17
Average weights variance is  6.519348210938682e-35
```

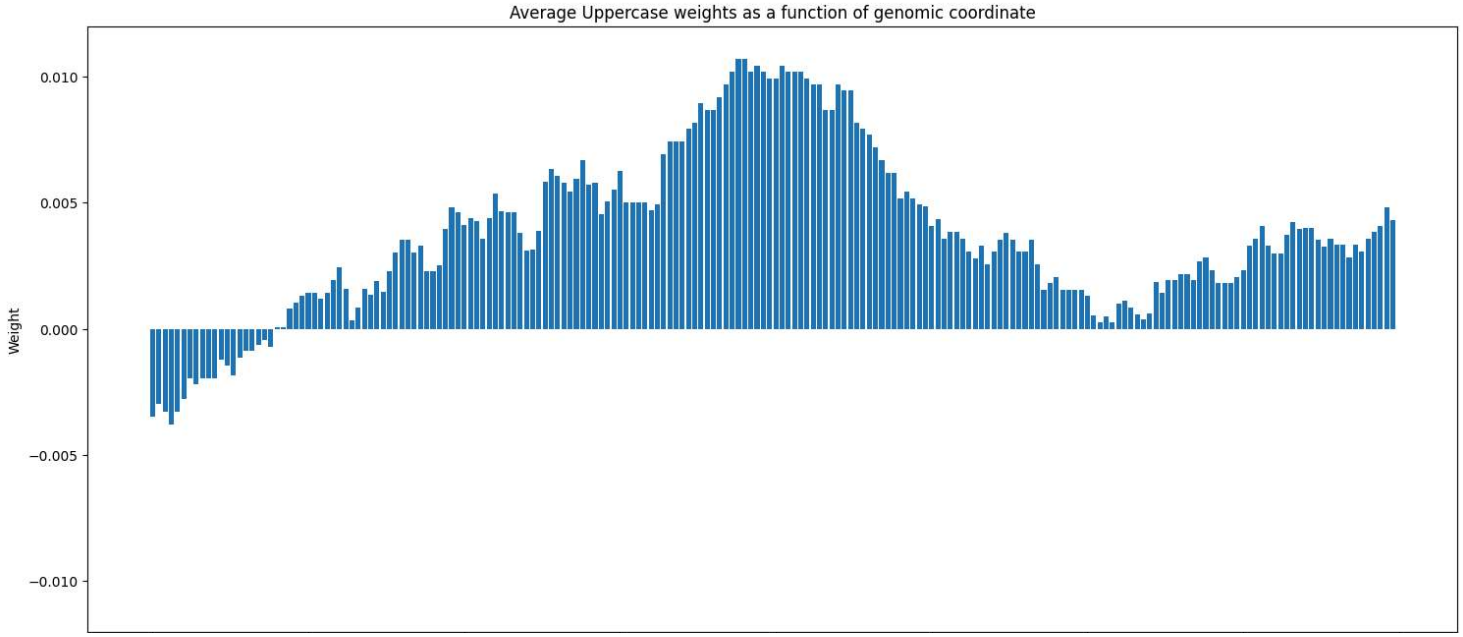
```
plt.figure(figsize=(18, 8))
plt.bar(range(len(PUM2_ENCSR661ICQ_weights)), PUM2_ENCSR661ICQ_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Weights as a function of genomic coordinate")
plt.ylim([-0.05, 0.05])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(PUM2_ENCSR661ICQ_A_weights)), PUM2_ENCSR661ICQ_A_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'A' Weights as a function of genomic coordinate")
plt.ylim([-0.05, 0.05])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(PUM2_ENCSR661ICQ_a_weights)), PUM2_ENCSR661ICQ_a_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'a' Weights as a function of genomic coordinate")
plt.ylim([-0.05, 0.05])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(PUM2_ENCSR661ICQ_C_weights)), PUM2_ENCSR661ICQ_C_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'C' Weights as a function of genomic coordinate")
plt.ylim([-0.05, 0.05])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(PUM2_ENCSR661ICQ_c_weights)), PUM2_ENCSR661ICQ_c_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'c' Weights as a function of genomic coordinate")
plt.ylim([-0.05, 0.05])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(PUM2_ENCSR661ICQ_G_weights)), PUM2_ENCSR661ICQ_G_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'G' Weights as a function of genomic coordinate")
plt.ylim([-0.05, 0.05])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(PUM2_ENCSR661ICQ_g_weights)), PUM2_ENCSR661ICQ_g_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'g' Weights as a function of genomic coordinate")
plt.ylim([-0.05, 0.05])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(PUM2_ENCSR661ICQ_T_weights)), PUM2_ENCSR661ICQ_T_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'T' Weights as a function of genomic coordinate")
plt.ylim([-0.05, 0.05])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(PUM2_ENCSR661ICQ_t_weights)), PUM2_ENCSR661ICQ_t_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'t' Weights as a function of genomic coordinate")
plt.ylim([-0.05, 0.05])
plt.show()
```






```
plt.figure(figsize=(18, 8))
plt.bar(range(len(PUM2_ENCSR661ICQ_avg_weights)), PUM2_ENCSR661ICQ_avg_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Average weights as a function of genomic coordinate")
plt.ylim([-0.012, 0.012])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(PUM2_ENCSR661ICQ_avg_uppercase_weights)), PUM2_ENCSR661ICQ_avg_uppercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Average Uppercase weights as a function of genomic coordinate")
plt.ylim([-0.012, 0.012])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(PUM2_ENCSR661ICQ_avg_lowercase_weights)), PUM2_ENCSR661ICQ_avg_lowercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Average Lowercase weights as a function of genomic coordinate")
plt.ylim([-0.012, 0.012])
plt.show()
```





QKI

ENCSR366YOG

Import dataset and fit model

```
QKI_ENCSR366YOG_ds = import_dataset()
[QKI_ENCSR366YOG_train, QKI_ENCSR366YOG_test] = train_test_split(QKI_ENCSR366YOG_ds,train_size=0.8, random_state=103)

# One-Hot Encoding
QKI_ENCSR366YOG_train_features = pd.get_dummies(QKI_ENCSR366YOG_train.iloc[:, 0:200]).to_numpy()
QKI_ENCSR366YOG_train_labels = QKI_ENCSR366YOG_train['label'].to_numpy()

QKI_ENCSR366YOG_test_features = pd.get_dummies(QKI_ENCSR366YOG_test.iloc[:, 0:200]).to_numpy()
QKI_ENCSR366YOG_test_labels = QKI_ENCSR366YOG_test['label'].to_numpy()

# C=1e-3 worked best
QKI_ENCSR366YOG_model = svm.SVC(C=1e-3, kernel="linear")
QKI_ENCSR366YOG_model = QKI_ENCSR366YOG_model.fit(QKI_ENCSR366YOG_train_features, QKI_ENCSR366YOG_train_labels)
```

לא נבחר קובץ יש לבחור קבצים Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving QKI_ENCSR366YOG_dataset.txt to QKI_ENCSR366YOG_dataset.txt
User uploaded file "QKI_ENCSR366YOG_dataset.txt" with length 816000 bytes
```

Import RBPmap predictions

```
uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))

with open(fn, 'r') as opened_file:
    txt_lines = opened_file.readlines()
txt_lines = txt_lines[1:]

QKI_ENCSR366YOG_rbp_no_conservation = np.array([int(line.split('\t')[0]) for line in txt_lines])
QKI_ENCSR366YOG_true_labels = np.array([int(line.split('\t')[1]) for line in txt_lines])
```

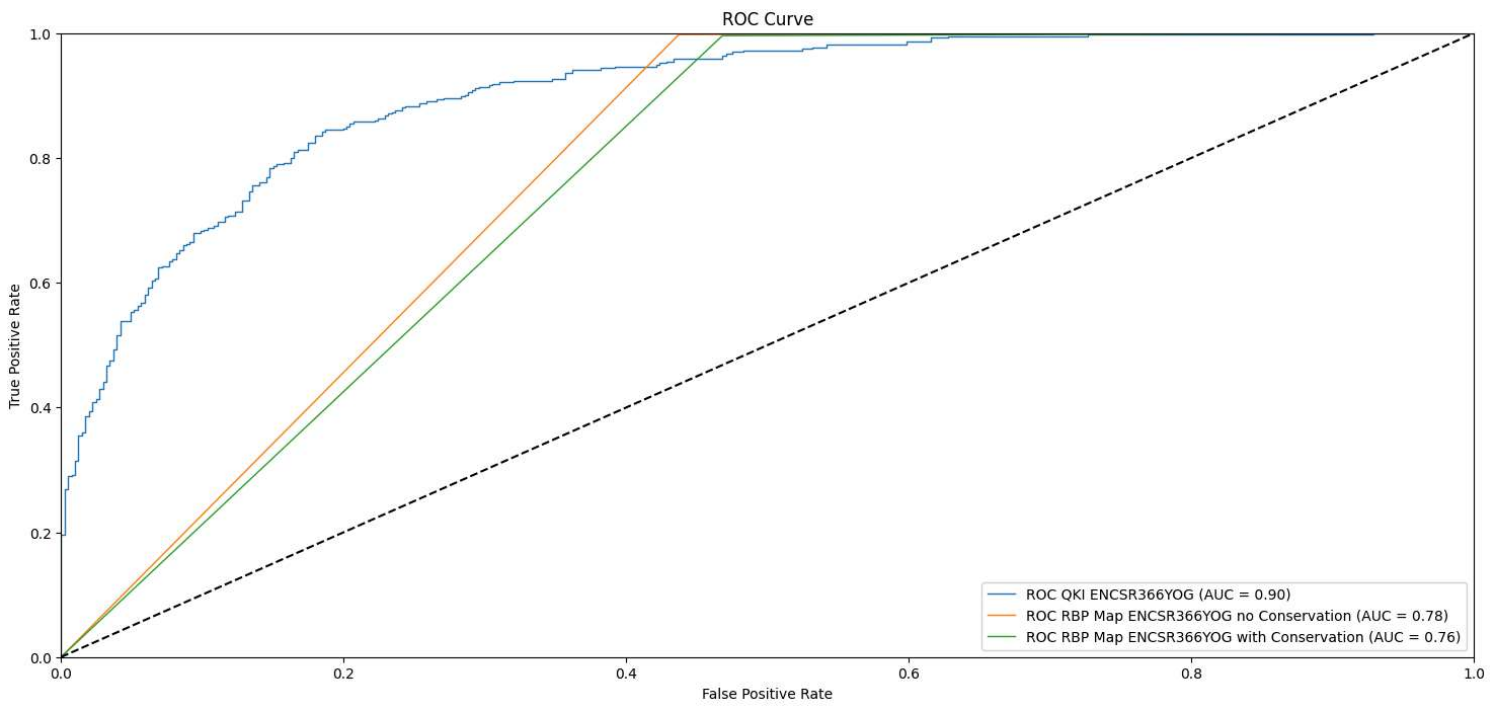
```
יש לבחור קבצים  לא נבחר קובץ  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving QKI_ENCSR366YOG_rbp_predictions_no_conservation.txt to QKI_ENCSR366YOG_rbp_predictions_no_conservation.txt
User uploaded file "QKI_ENCSR366YOG_rbp_predictions_no_conservation.txt" with length 20013 bytes
```

```
uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))

with open(fn, 'r') as opened_file:
    txt_lines = opened_file.readlines()
txt_lines = txt_lines[1:]

QKI_ENCSR366YOG_rbp_with_conservation = np.array([int(line.split('\t')[0]) for line in txt_lines])
```

```
יש לבחור קבצים  לא נבחר קובץ  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving QKI_ENCSR366YOG_rbp_predictions_with_conservation.txt to QKI_ENCSR366YOG_rbp_predictions_with_conservation.txt
User uploaded file "QKI_ENCSR366YOG_rbp_predictions_with_conservation.txt" with length 20013 bytes
```

Display results

```
fig, ax = plt.subplots(figsize=(18, 8))

viz = RocCurveDisplay.from_estimator(
    QKI_ENCSR366YOG_model,
    QKI_ENCSR366YOG_test_features,
    QKI_ENCSR366YOG_test_labels,
    name=f"ROC QKI ENCSR366YOG",
    lw=1,
    ax=ax
)

viz = RocCurveDisplay.from_predictions(
    QKI_ENCSR366YOG_rbp_no_conservation,
    QKI_ENCSR366YOG_true_labels,
    name=f"ROC RBP Map ENCSR366YOG no Conservation",
    lw=1,
    ax=ax
)

viz = RocCurveDisplay.from_predictions(
    QKI_ENCSR366YOG_rbp_with_conservation,
    QKI_ENCSR366YOG_true_labels,
    name=f"ROC RBP Map ENCSR366YOG with Conservation",
    lw=1,
    ax=ax
)

ax.set(
    xlabel="False Positive Rate",
    ylabel="True Positive Rate",
    title=f"ROC Curve",
)
ax.legend(loc="lower right")
ax.plot(np.linspace(0,1,10), np.linspace(0,1,10), '--',color='black')
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.show()
```

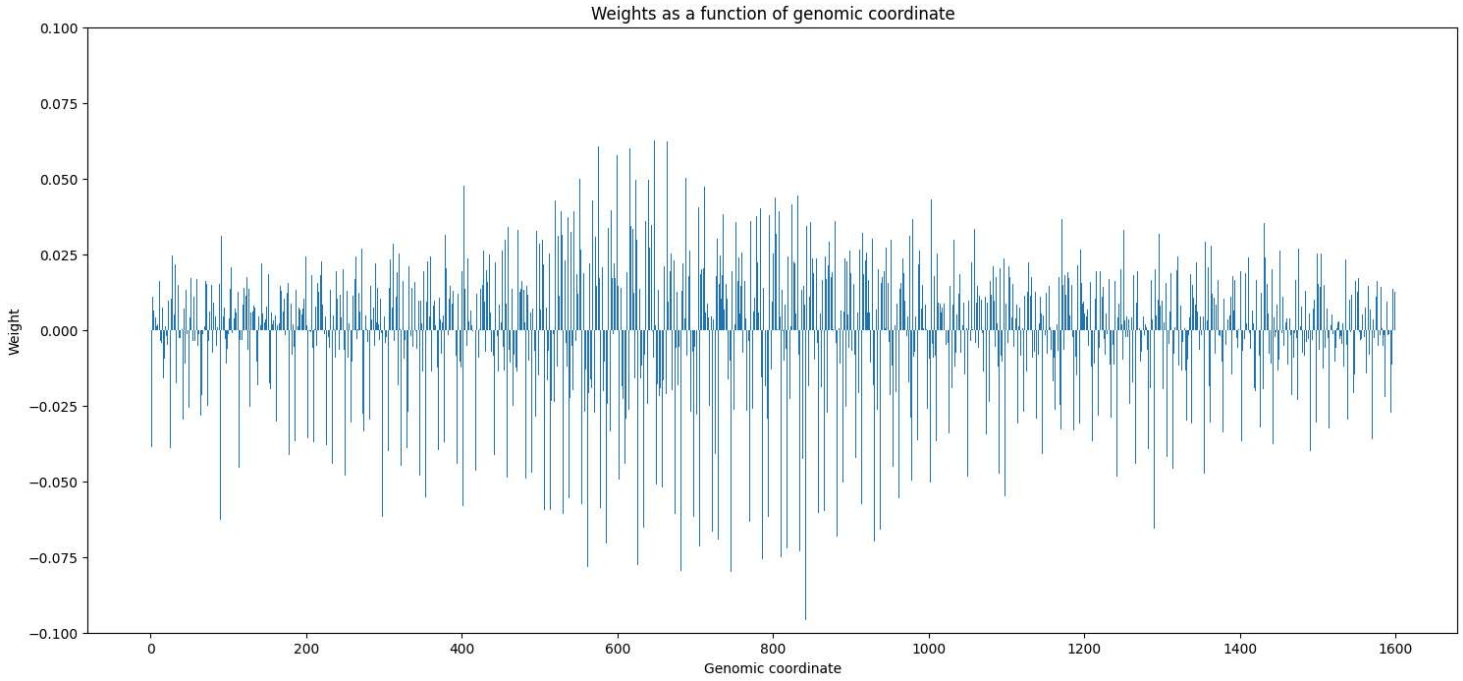


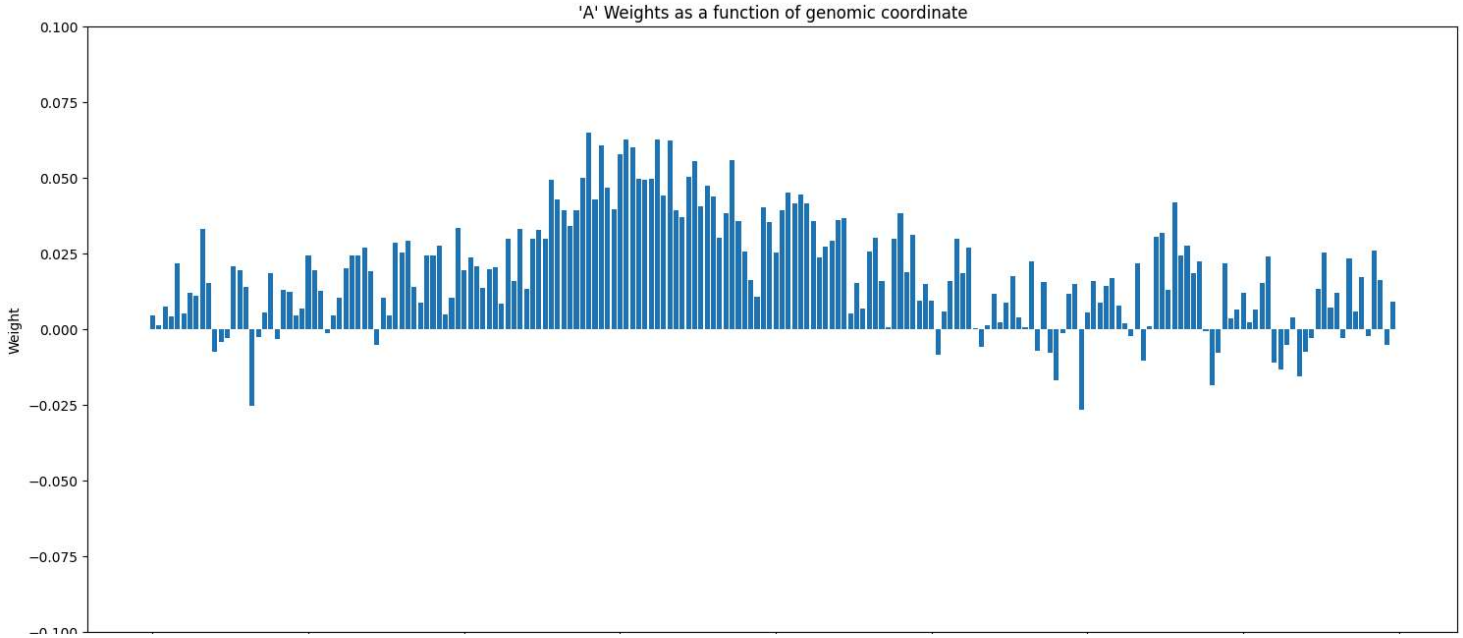
Display trained weights

```
QKI_ENCSR366YOG_weights = QKI_ENCSR366YOG_model.coef_[0]
QKI_ENCSR366YOG_A_weights = QKI_ENCSR366YOG_weights[::8]
QKI_ENCSR366YOG_a_weights = QKI_ENCSR366YOG_weights[4::8]
QKI_ENCSR366YOG_C_weights = QKI_ENCSR366YOG_weights[1::8]
QKI_ENCSR366YOG_c_weights = QKI_ENCSR366YOG_weights[5::8]
QKI_ENCSR366YOG_G_weights = QKI_ENCSR366YOG_weights[2::8]
QKI_ENCSR366YOG_g_weights = QKI_ENCSR366YOG_weights[6::8]
QKI_ENCSR366YOG_T_weights = QKI_ENCSR366YOG_weights[3::8]
QKI_ENCSR366YOG_t_weights = QKI_ENCSR366YOG_weights[7::8]

QKI_ENCSR366YOG_avg_weights = QKI_ENCSR366YOG_weights.reshape(-1, 8).mean(axis=1)
QKI_ENCSR366YOG_avg_uppercase_weights = QKI_ENCSR366YOG_weights.reshape(-1, 4).mean(axis=1)[::2]
QKI_ENCSR366YOG_avg_lowercase_weights = QKI_ENCSR366YOG_weights.reshape(-1, 4).mean(axis=1)[1::2]

print('Average weights mean is ', QKI_ENCSR366YOG_avg_weights.mean())
print('Average weights variance is ', QKI_ENCSR366YOG_avg_weights.var())
```

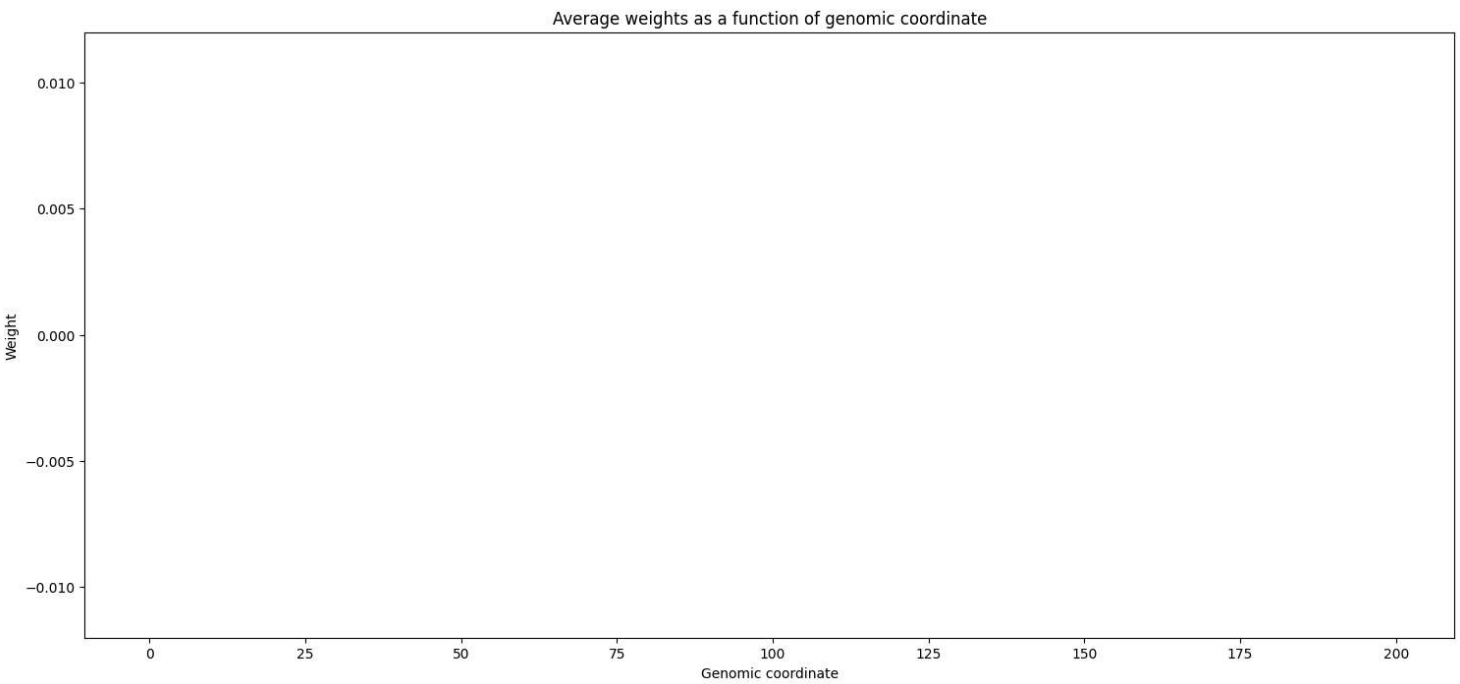
```
Average weights mean is  1.6225085511245574e-18
Average weights variance is  6.008814175419986e-35
```

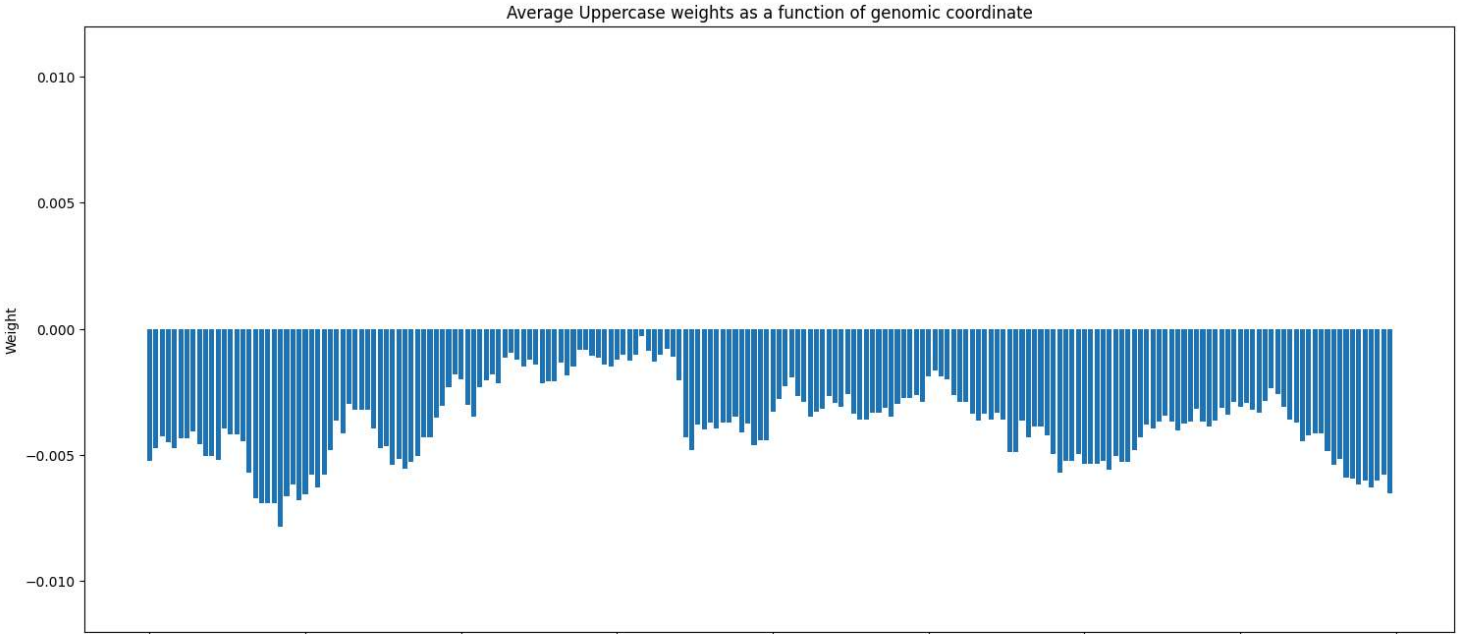
```
plt.figure(figsize=(18, 8))
plt.bar(range(len(QKI_ENCSR366YOG_weights)), QKI_ENCSR366YOG_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(QKI_ENCSR366YOG_A_weights)), QKI_ENCSR366YOG_A_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'A' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(QKI_ENCSR366YOG_a_weights)), QKI_ENCSR366YOG_a_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'a' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(QKI_ENCSR366YOG_C_weights)), QKI_ENCSR366YOG_C_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'C' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(QKI_ENCSR366YOG_c_weights)), QKI_ENCSR366YOG_c_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'c' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(QKI_ENCSR366YOG_G_weights)), QKI_ENCSR366YOG_G_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'G' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(QKI_ENCSR366YOG_g_weights)), QKI_ENCSR366YOG_g_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'g' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(QKI_ENCSR366YOG_T_weights)), QKI_ENCSR366YOG_T_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'T' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(QKI_ENCSR366YOG_t_weights)), QKI_ENCSR366YOG_t_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("'t' Weights as a function of genomic coordinate")
plt.ylim([-0.1, 0.1])
plt.show()
```


Weights as a function of genomic coordinate
Weight
Genomic coordinate
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
0
200
400
600
800
1000
1200
1400
1600


'A' Weights as a function of genomic coordinate
Weight
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075

```
plt.figure(figsize=(18, 8))
plt.bar(range(len(QKI_ENCSR366YOG_avg_weights)), QKI_ENCSR366YOG_avg_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Average weights as a function of genomic coordinate")
plt.ylim([-0.012, 0.012])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(QKI_ENCSR366YOG_avg_uppercase_weights)), QKI_ENCSR366YOG_avg_uppercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Average Uppercase weights as a function of genomic coordinate")
plt.ylim([-0.012, 0.012])
plt.show()

plt.figure(figsize=(18, 8))
plt.bar(range(len(QKI_ENCSR366YOG_avg_lowercase_weights)), QKI_ENCSR366YOG_avg_lowercase_weights)
plt.xlabel('Genomic coordinate')
plt.ylabel('Weight')
plt.title("Average Lowercase weights as a function of genomic coordinate")
plt.ylim([-0.012, 0.012])
plt.show()
```


Average weights as a function of genomic coordinate
0.010
0.005
0.000
−0.005
−0.010
Weight
0
25
50
75
100
125
150
175
200
Genomic coordinate


Average Uppercase weights as a function of genomic coordinate
0.010
0.005
0.000
−0.005
−0.010
Weight