```
'''This cell adapted from:
https://colab.research.google.com/drive/1DofKEdQYaXmDWBzuResXWWvxhLgDeVyl#scrollTo=0hy
'''

'''
To use the Kaggle API, sign up for a Kaggle account at
https://www.kaggle.com. Then go to the 'Account' tab of
your user profile (https://www.kaggle.com/<username>/account)
and select 'Create API Token'. This will trigger the download
of kaggle.json, a file containing your API credentials.
'''

# Run this cell and select the kaggle.json file downloaded
# from the Kaggle account settings page.
from google.colab import files
files.upload()

# Next, install the Kaggle API client.
!pip install -q kaggle

# The Kaggle API client expects this file to be in ~/.kaggle,
# so move it there.
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/

# This permissions change avoids a warning on Kaggle tool startup.
!chmod 600 ~/.kaggle/kaggle.json

# List available datasets.
!kaggle datasets list
```

```
Choose Files kaggle.json
```

- **kaggle.json**(application/json) - 70 bytes, last modified: 6/11/2019 - 100% done

```
Saving kaggle.json to kaggle.json
ref                                                         title
----------------------------------------------------------  ----------------------
chicago/chicago-copa-cases                                  Chicago COPA Cases
himanshupoddar/zomato-bangalore-restaurants                 Zomato Bangalore Restau
sfinspiredu/synchrotron-data-set                            Synchrotron Data Set
crisparada/brazilian-cities                                 Brazilian Cities
taniaj/australian-election-2019-tweets                      Australian Election 201
romainpessia/artificial-lunar-rocky-landscape-dataset       Artificial Lunar Landsc
gqfiddler/scotus-opinions                                   SCOTUS Opinions
se18m502/bee-hive-metrics                                   Beehive Metrics
brittabettendorf/berlin-airbnb-data                         Berlin Airbnb Data
PromptCloudHQ/world-happiness-report-2019                   World Happiness Report
jlesuffleur/granddebat                                      Le Grand Débat National
thegurus/spanish-high-speed-rail-system-ticket-pricing      Spanish High Speed Rail
leomauro/smmnet                                             SMMnet
snocco/missing-migrants-project                             Missing Migrants Projec
austinreese/craigslist-carstrucks-data                      Craigslist Cars+Trucks
robseidl/tennis-atp-tour-australian-open-final-2019         Tennis ATP Tour Austral
cityofLA/los-angeles-traffic-collision-data                 Los Angeles Traffic Col
inIT-OWL/versatileproductionsystem                          Versatile Production Sy
alvarob96/spanish-stocks-historical-data                    Spanish Stocks Historic
mfekadu/darpa-timit-acousticphonetic-continuous-speech      DARPA TIMIT Acoustic-Ph
```

```
'''The following cells (through the import pandas line)
serve to download the Kaggle dataset to my google drive.
```

```
Do not run them.'''
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call d
```

```
cd drive/My\ Drive/Datasets
```

```
/content/drive/My Drive/Datasets
```

```
mkdir Kaggle_HIV
```

```
cd Kaggle_HIV/
```

```
/content/drive/My Drive/Datasets/Kaggle_HIV
```

```
!kaggle competitions download -c hivprogression
```

```
Downloading test_data.csv to /content/drive/My Drive/Datasets/Kaggle_HIV
  0% 0.00/876k [00:00<?, ?B/s]
100% 876k/876k [00:00<00:00, 29.0MB/s]
Downloading training_data.csv to /content/drive/My Drive/Datasets/Kaggle_HIV
  0% 0.00/1.19M [00:00<?, ?B/s]
100% 1.19M/1.19M [00:00<00:00, 38.2MB/s]
```

```
ls
```

```
test_data.csv  training_data.csv
```

```
import pandas as pd
```

```
#Look at the data we've got:

data = pd.read_csv('training_data.csv')
data.head()
```

	PatientID	Resp	PR Seq	
0	1	0	CCTCAAATCACTCTTTGGCAACGACCCCTCGTCCCAATAAGGATAG...	CCC
1	2	0	CCTCAAATCACTCTTTGGCAACGACCCCTCGTCGCAATAAAGATAG...	CC(
2	3	0	CCTCAAATCACTCTTTGGCAACGACCCCTCGTCGCAATAAAGGTAG...	CC(
3	4	0	CCTCAAATCACTCTTTGGCAACGACCCCTCGTCGCAATAAGGATAG...	CC(
4	5	0	CCTCAAATCACTCTTTGGCAACGACCCCTCGTCGCAGTAAAGATAG...	CC(

```
#What are the columns?
data.columns
```

```
Index(['PatientID', 'Resp', 'PR Seq', 'RT Seq', 'VL-t0', 'CD4-t0'], dtype='obje
```

```
#How is the data distributed?
data['Resp'].value_counts()
```

```
0    794
1    206
Name: Resp, dtype: int64
```

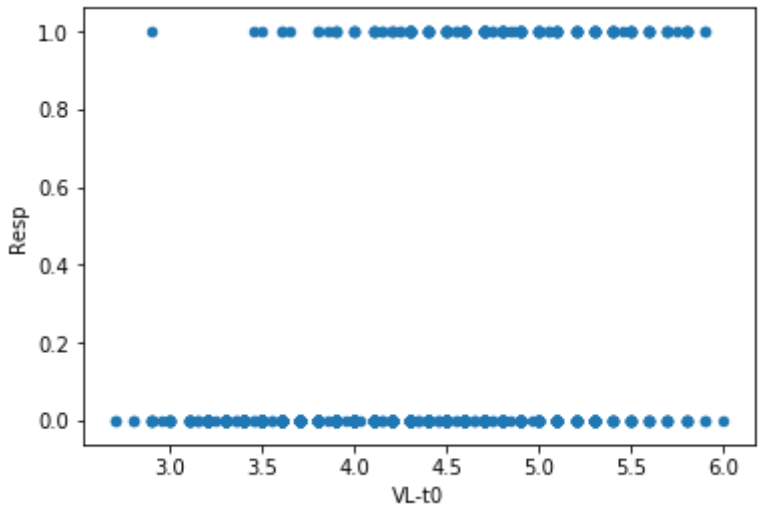
```
#How do the sequences vary? This looks at length:

data['PR Seq'].dropna().apply(lambda x: len(str(x))).value_counts()
```

```
297    889
294     14
267      6
270      3
285      2
252      2
276      1
261      1
255      1
216      1
Name: PR Seq, dtype: int64
```

```
#How is Viral Load and Response linked?
data.plot.scatter('VL-t0', 'Resp')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb6478960b8>
```




```
#The above plot looks like the relationship follows
#a logistic regression - this is common for binary
#data (given Resp is either 0 or 1, we call the data binary)
```

```
#Run a simple Logistic regression to predict
```

```
#responsiveness from Viral Load and CD4 count:
#(You can ignore the warnings this generates, if any.)

from sklearn.linear_model import LogisticRegression

model = LogisticRegression(solver='lbfgs')
X = data[['VL-t0', 'CD4-t0']]
y = data['Resp']
model.fit(X,y)
y_pred = model.predict(X)
acc = sum(data['Resp']==y_pred)/len(data['Resp'])
print('Accuracy of Logistic Regression Model: {}%'.format(acc*100))
```

```
Accuracy of Logistic Regression Model: 78.7%
```