Dunyeng Huh Flatiron Interview

In [3]:
```
import seaborn as sns
import pandas as pd
import datetime as dt
import numpy as np
import matplotlib.pyplot as plt
import statistics as st
from tabulate import tabulate
from scipy.stats import skew, kurtosis


# Change this line to get data
treatment = pd.read_csv("data/TreatmentSample01.csv")
diagnosis = pd.read_csv("data/DiagnosisSample01.csv")

# treatment = pd.read_csv("/Users/dunyenghuh/Desktop/Grad School/Quarter1/TreatmentSample01.
csv")
```

Exploratory Data Analysis (Understanding both tables)

Treatment Sample CSV

In [4]:
```
print(treatment.shape)
#double check over missing data
print(treatment.isnull().sum())
treatment.head()
```

```
(714, 3)
PatientID        0
TreatmentDate    0
DrugCode         0
dtype: int64
```

Out[4]:

	PatientID	TreatmentDate	DrugCode
0	2038	2010-01-24	A
1	2038	2010-01-27	A
2	2038	2010-01-30	A
3	2038	2010-02-02	A
4	2038	2010-02-06	A

In [5]:
```
# Patient IDs
print("Unique Patient IDs in Treatment CSV: ", treatment.PatientID.nunique())
```

```
Unique Patient IDs in Treatment CSV:  27
```

In [6]:
```
# Treatment Dates
print("Unique Dates in Treatment CSV: ",treatment.TreatmentDate.nunique())
treatment.agg({'TreatmentDate':[min,max]})
```

```
Unique Dates in Treatment CSV:  469
```

Out[6]:

	TreatmentDate
min	2010-01-24
max	2014-10-25

In [7]:
```
# Drug Codes
print("Unique DrugCodes in Treatment CSV: ",treatment.DrugCode.nunique())
print(treatment.DrugCode.value_counts())
```

```
Unique DrugCodes in Treatment CSV:  3
C    281
A    231
B    202
Name: DrugCode, dtype: int64
```

Diagnosis Sample CSV

In [8]:
```
print(diagnosis.shape)
#double check over missing data
print(diagnosis.isnull().sum())
diagnosis.head()
```

```
(44, 5)
PatientID            0
DiagnosisDate        0
DiagnosisCode        0
Diagnosis            0
IsCancerDiagnosis    0
dtype: int64
```

Out[8]:

	PatientID	DiagnosisDate	DiagnosisCode	Diagnosis	IsCancerDiagnosis
0	2634	2011-02-19	285.8	Anemia	False
1	5657	2012-06-07	285.8	Anemia	False
2	7937	2013-01-06	285.8	Anemia	False
3	8615	2013-07-18	284.9	Anemia	False
4	4354	2012-02-04	284.9	Anemia	False

In [9]:
```
#Patient ID
print("Unique Patient IDs in Diagnosis CSV: ",diagnosis.PatientID.nunique())
print("Note that the number of Patient IDs in two files differs")

#some patients have multiple diagnosis
diagnosis.groupby(['PatientID']).Diagnosis.count().reset_index().sort_values(by='Diagnosis',
```

```
Unique Patient IDs in Diagnosis CSV:  32
Note that the number of Patient IDs in two files differs
```

Out[9]:

	PatientID	Diagnosis
23	6922	3
14	4374	3
9	3449	2
10	3757	2
21	6877	2
22	6889	2
13	4354	2
12	4256	2
8	3095	2
24	7230	2
0	2038	1
19	6321	1
25	7242	1
26	7796	1
27	7937	1
28	7976	1
29	8615	1
30	8827	1
20	6837	1
16	5259	1
18	6281	1
17	5657	1
1	2120	1
15	4692	1
11	3948	1
7	2770	1
6	2763	1
5	2634	1
4	2462	1
3	2425	1
2	2407	1
31	9331	1

In [10]:
```
# Diagnosis Date

print("Unique Dates in Diagnosis CSV: ",diagnosis.DiagnosisDate.nunique())
diagnosis.agg({'DiagnosisDate':[min,max]})
```

```
Unique Dates in Diagnosis CSV:  38
```

Out[10]:

	DiagnosisDate
min	2010-01-09
max	2013-08-23

In [11]:
```
#Diagnosis Code
print("Unique DiagnosisCode in Diagnosis CSV: ", diagnosis.DiagnosisCode.nunique())
diagnosis.DiagnosisCode.value_counts()
```

```
Unique DiagnosisCode in Diagnosis CSV:  20
```

Out[11]:
```
174.9    6
285.8    4
174.1    4
174.8    3
153.5    3
401.9    3
153.3    2
153.9    2
284.9    2
174.5    2
174.7    2
153.4    2
174.3    2
153.6    1
285.9    1
153.8    1
174.6    1
174.4    1
401.1    1
174.2    1
Name: DiagnosisCode, dtype: int64
```

In [12]:
```
#Diagnosis
print("Unique Diagnosis in Diagnosis CSV: ", diagnosis.Diagnosis.nunique())
diagnosis.Diagnosis.value_counts()
```

```
Unique Diagnosis in Diagnosis CSV:  5
```

Out[12]:
```
Breast Cancer    22
Colon Cancer     11
Anemia            7
Hypertension      3
Hypertension      1
Name: Diagnosis, dtype: int64
```

In [13]:
```
#Is Cancer Diagnosis
diagnosis.IsCancerDiagnosis.value_counts()
```

Out[13]:
```
True     33
False    11
Name: IsCancerDiagnosis, dtype: int64
```

Merging Two CSV Files to make a comprehensive table

In [14]:
```
diag0 = diagnosis.copy()
tret1 = treatment.copy()
print("Diagnosis csv shape: ",diag0.shape)
print("Treatment csv shape: ",tret1.shape)

# Patients who are diagnosed of cancer are going to be the ones who are treated by medicine.
# Separate Cancer Patients and Non-Cancer Patients
diag_true = diag0[diag0.IsCancerDiagnosis == True]
diag_false = diag0[diag0.IsCancerDiagnosis == False]

# make column 'alldate' for the ease of merging
diag_true['alldate'] = diag_true['DiagnosisDate']
tret1['alldate'] = tret1['TreatmentDate']

# sort values before merging
diag_true = diag_true.sort_values(by=['PatientID', 'alldate'])
tret1 = tret1.sort_values(by=['PatientID', 'alldate'])

# outer join on 'PatientID' and 'alldate'
merged = pd.merge(diag_true, tret1, how='outer', on=['PatientID', 'alldate'])
merged = merged.sort_values(by=['PatientID', 'alldate']).reset_index(drop=True)

# The two tables are joined on the 'DiagnosedDate', which is earlier than 'TreatmentDate'
# Patients can get multiple treatments for each diagnosis. Thus, forward fill the below 4 co
lumns for same diagnosis
merged['DiagnosisCode'] = merged.groupby('PatientID')['DiagnosisCode'].ffill()
merged['Diagnosis'] = merged.groupby('PatientID')['Diagnosis'].ffill()
merged['IsCancerDiagnosis'] = merged.groupby('PatientID')['IsCancerDiagnosis'].ffill()
merged['DiagnosisDate'] = merged.groupby('PatientID')['DiagnosisDate'].ffill()

# The DiagnosedDate, in which two tables are joined on, has NaN values because treatment did
not occur yet for this date
# Rather than having a NaN value for this rows, backward fill the below two columns.
merged['TreatmentDate'] = merged.groupby('PatientID')['TreatmentDate'].bfill()
merged['DrugCode'] = merged.groupby('PatientID')['DrugCode'].bfill()

# select the necessary columns and drop duplicate rows made from forwardfill and backwardfil
l
df1 = merged[['PatientID', 'DiagnosisDate', 'DiagnosisCode', 'Diagnosis', 'IsCancerDiagnosi
s', 'TreatmentDate','DrugCode']]
df1 = df1.drop_duplicates()

# to concat the diag_false dataset, make 2 new columns.
# Since these patients are not diagnosed of cancers, they do not receive treatments and the
 columns below remain as NaN
diag_false['TreatmentDate'] = np.NaN
diag_false['DrugCode'] = np.NaN

# Append the non-cancer dataset and sort the data again to finalize the comprehensive datase
t
df1 = df1.append(diag_false)
```

```
Diagnosis csv shape:  (44, 5)
Treatment csv shape:  (714, 3)
final data shape:  (730, 7)
```

```
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:12: SettingWithCopyWarning:
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: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
  if sys.path[0] == '':
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:41: SettingWithCopyWarning:
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: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:42: SettingWithCopyWarning:
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: https://pandas.pydata.org/pandas-docs/stable/user_guid
e/indexing.html#returning-a-view-versus-a-copy
```

Out[14]:

	PatientID	DiagnosisDate	DiagnosisCode	Diagnosis	IsCancerDiagnosis	TreatmentDate	DrugCode
0	2038	2010-01-21	174.9	Breast Cancer	True	2010-01-24	A
1	2038	2010-01-21	174.9	Breast Cancer	True	2010-01-27	A
2	2038	2010-01-21	174.9	Breast Cancer	True	2010-01-30	A
3	2038	2010-01-21	174.9	Breast Cancer	True	2010-02-02	A
4	2038	2010-01-21	174.9	Breast Cancer	True	2010-02-06	A

Questions

Questions 1

1a) Which types of cancer does the clinic see patients for?

1b) How many patients does the clinic see for each cancer type?

In [15]:
```
# Select cancer patients from the diagnosis data and group them by Diagnosis type
# Then count unique PatientIDs, as a Patient can have multiple diagnosis.
cancer_count = diagnosis[diagnosis['IsCancerDiagnosis'] == True].groupby(['Diagnosis'],as_in
dex=False).PatientID.nunique()
cancer_count = cancer_count.rename(columns={"Diagnosis": "Daignosis Type","PatientID": "Uniq
ue Patient Count"})

print("There are 2 types of cancer: Breast and Colon.")
print("There are 20 unique patients for Breast Cancer and 9 for Colon Cancer")

cancer_count

#alternatively, the above result can be found thru the code below - used for sanity check
```

```
There are 2 types of cancer: Breast and Colon.
There are 20 unique patients for Breast Cancer and 9 for Colon Cancer
```

Out[15]:

	Daignosis Type	Unique Patient Count
0	Breast Cancer	20
1	Colon Cancer	9

Question 2: The clinic wants to know how long it takes for patients to start therapy after being diagnosed, which they consider to be helpful in understanding the quality of care for the patients.

2a) How long after being diagnosed do cancer patients start treatment for each cancer type?

In [16]:
```
# select patients with breast cancer and colon cancer from the comprehensive dataset
# group by PatientID and DiagnosisDate and follow the earliest Treatment Date
breast_cancer = df1[df1.Diagnosis == 'Breast Cancer'].groupby(['PatientID','DiagnosisDate'],
as_index=False).agg({'TreatmentDate':min})
colon_cancer = df1[df1.Diagnosis == 'Colon Cancer'].groupby(['PatientID','DiagnosisDate'],as
_index=False).agg({'TreatmentDate':min})
# change both dates in the 2 tables to datetime format to calculate the difference in days
breast_cancer['DiagnosisDate'] = pd.to_datetime(breast_cancer['DiagnosisDate'])
breast_cancer['TreatmentDate'] = pd.to_datetime(breast_cancer['TreatmentDate'])
colon_cancer['DiagnosisDate'] = pd.to_datetime(colon_cancer['DiagnosisDate'])
colon_cancer['TreatmentDate'] = pd.to_datetime(colon_cancer['TreatmentDate'])
# calculate the difference in days
breast_cancer['nday'] = (breast_cancer['TreatmentDate'] - breast_cancer['DiagnosisDate'] ).d
t.days
colon_cancer['nday'] = (colon_cancer['TreatmentDate'] - colon_cancer['DiagnosisDate'] ).dt.d
ays
breast_cancer_nday = list(breast_cancer.nday)
colon_cancer_nday = list(colon_cancer.nday)

print("Mean time difference from diagnosis to treatment for breast cancer is :",
      st.mean(breast_cancer_nday),"days, for colon cancer :",st.mean(colon_cancer_nday),"day
s")
print("Median time difference from diagnosis to treatment for breast cancer is :",
      st.median(breast_cancer_nday),"days, for colon cancer :",st.median(colon_cancer_nday),
```

```
Mean time difference from diagnosis to treatment for breast cancer is : 5.1 days, for colon c
ancer : 13.1 days
Median time difference from diagnosis to treatment for breast cancer is : 4.0 days, for colon
cancer : 14.5 days
```

2b) Are there any patients which are diagnosed but not treated at the practice?

In [18]:
```
# select cancer patients whose drugcode is null
print(df1[(df1.IsCancerDiagnosis == True)&(df1.DrugCode.isnull())])

# alternatively, check if all patients with cancer and are treated
patientid_with_cancer = diagnosis[diagnosis.IsCancerDiagnosis == True].PatientID.unique().to
list()
patientid_treated = treatment.PatientID.unique().tolist()
print([i for i in patientid_with_cancer if i not in patientid_treated])
```

```
Empty DataFrame
Columns: [PatientID, DiagnosisDate, DiagnosisCode, Diagnosis, IsCancerDiagnosis, TreatmentDat
e, DrugCode]
Index: []
[]
```

Question 3: After being treated with a first line of treatment (a drug or combination of drugs), what proportion of all cancer patients go on to be treated with a second line of treatment? (For more information on the concept of "first-line therapy", please reference https://www.cancer.net/navigating-cancer-care/how-cancer-treated/when-first-treatment-doesnt-work)

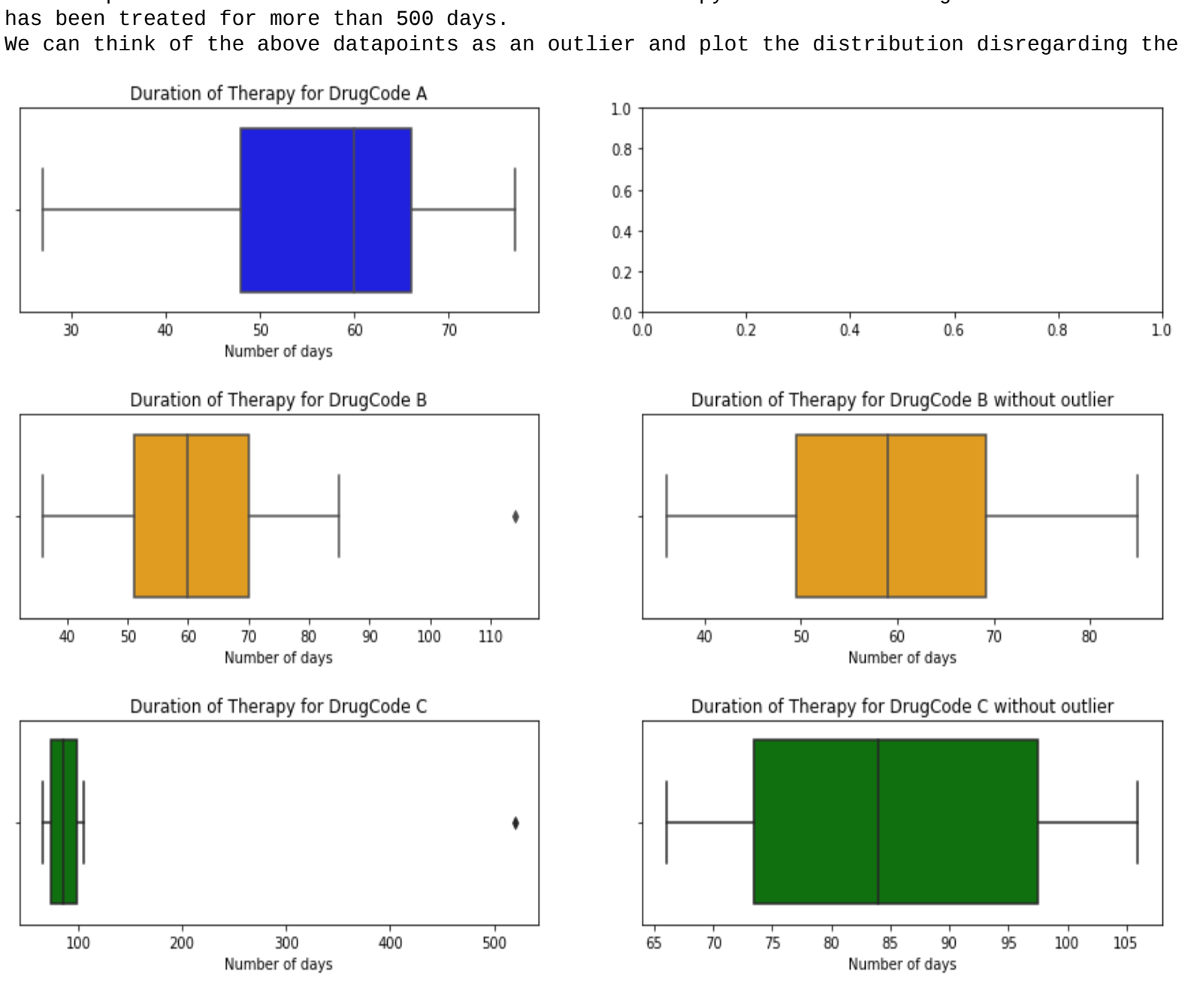
In [19]:
```
# Calculate the number of all cancer patients :27
df1[df1.IsCancerDiagnosis == True].PatientID.nunique()

# Some Patients are treated through monotherapy, while others are treated through combinatio
n of drugs
# Because treatments with combination of drugs occur on the same treatment date, group the d
ata by PatientId and Treatment Date
# Then, join the DrugCodes so that patients with DrugCode (A,B) will appear as 'AB'
change_in_treat = df1[df1.IsCancerDiagnosis == True].groupby(['PatientID','TreatmentDate'])[
'DrugCode'].apply(''.join).reset_index()

# Sort the data to use the shift function
change_in_treat.sort_values(by=['PatientID','TreatmentDate'], inplace=True)

# There are cases where patients are diagnosed with more than 1 cancer and treated with 1 dr
ugs.
# Because I am going to be using the shift function, I want to make sure that above cases ar
e captured normally
# If any values show same DrugCode appearing more than once, just keep the last value
change_in_treat["DrugCode"] = change_in_treat["DrugCode"].str.replace(r"(.)(?=.*\1)", "").st
r.strip()

# to see the change in treatment we want to compare the treatment from current date and the
 treatment from the previous hospital visit
# create a column 'lag' which shows the DrugCode from the previous treatment (thus need to g
roupby patient ID)
change_in_treat['Lag'] = change_in_treat.groupby('PatientID')['DrugCode'].shift()
# fill the NaN values occuring at the first treatment with backfill method
change_in_treat['Lag'] = change_in_treat.groupby('PatientID')['Lag'].bfill()

# if the DrugCode column and Lag column differs, put 1, otherwise put 0
change_in_treat['difference'] = np.where(change_in_treat['DrugCode'] != change_in_treat['La
g'], 1, 0)
change_in_treat[(change_in_treat.difference != 0)].difference.sum()
```

```
There are in total 27 cancer patients
7 Cancer patients experienced second line of treatment

The proportion of cancer patients treated with second line of treatment is:  0.26
```

Question 4: How does each drug used at the clinic compare in terms of its duration of therapy?

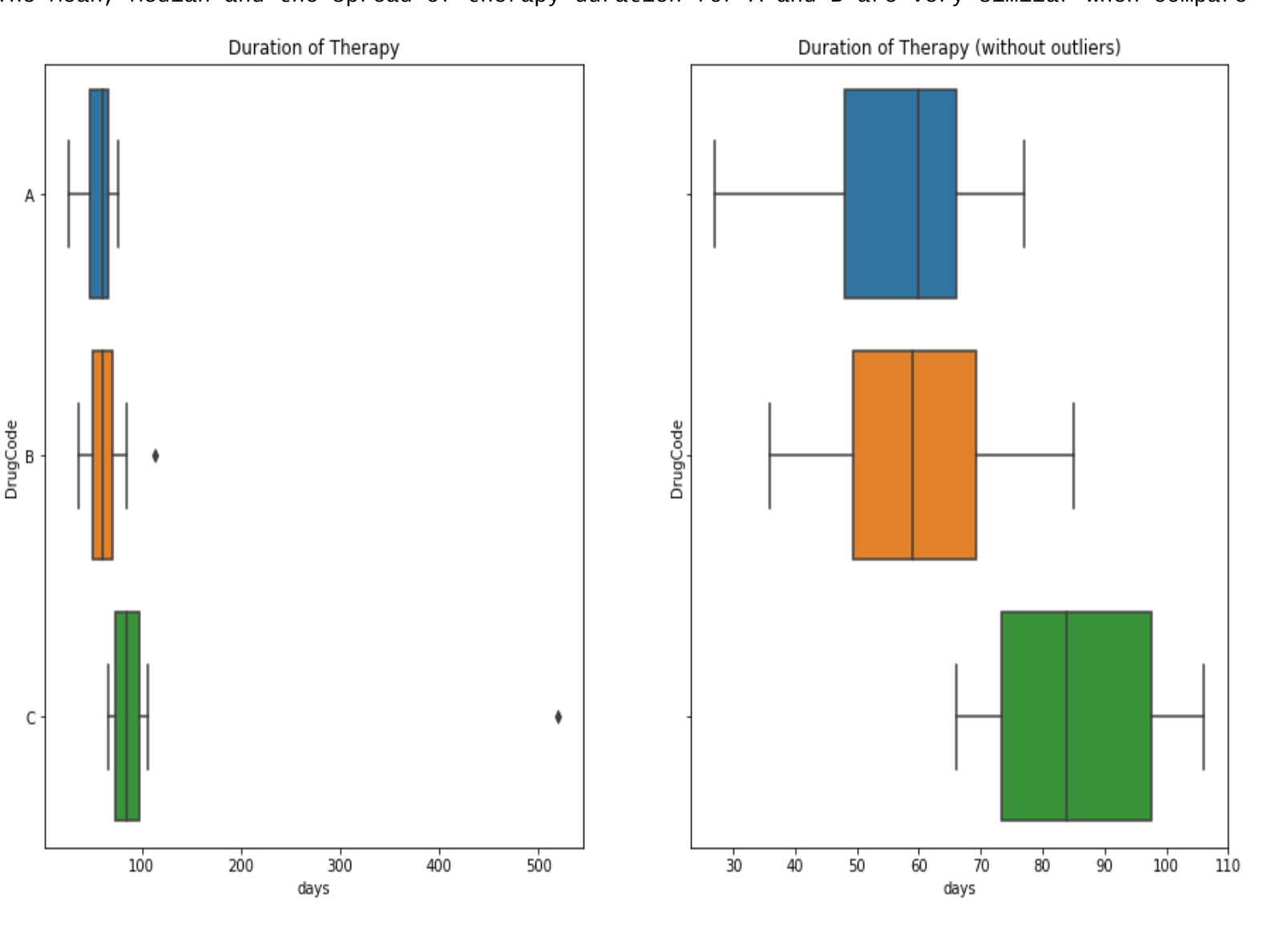
In [20]:
```
# change the treatment date to datetime format
df1['TreatmentDate'] = pd.to_datetime(df1['TreatmentDate'])

# group the data by patientID and drugcode to calcualte the earliest and latest treatment da
te for each (patient, drugcode)
date_per_drug = df1[df1.IsCancerDiagnosis == True].groupby(['PatientID','DrugCode'],as_index
=False).agg({'TreatmentDate':[min,max]})

# assume that duration of therapy can be represented by (latest treatment date - earliest tr
eatment date)
date_per_drug['nday'] = date_per_drug['TreatmentDate']['max'] - date_per_drug['TreatmentDat
e']['min']
date_per_drug
# make individual list for each drugcode
A = list(date_per_drug[date_per_drug.DrugCode == 'A'].nday.dt.days)
B = list(date_per_drug[date_per_drug.DrugCode == 'B'].nday.dt.days)
C = list(date_per_drug[date_per_drug.DrugCode == 'C'].nday.dt.days)

#plot the data
fig,ax = plt.subplots(3,2,figsize =(15,10))
plt.subplots_adjust(hspace = 0.5)

sns.boxplot(A, color= "blue", ax=ax[0,0])
ax[0,0].set_title("Duration of Therapy for DrugCode A")
ax[0,0].set_xlabel("Number of days")

sns.boxplot(B, color= "orange", ax=ax[1,0])
ax[1,0].set_title("Duration of Therapy for DrugCode B")
ax[1,0].set_xlabel("Number of days")

B_new = B.copy()
B_new.remove(114)
sns.boxplot(B_new, color= "orange", ax=ax[1,1])
ax[1,1].set_title("Duration of Therapy for DrugCode B without outlier")
ax[1,1].set_xlabel("Number of days")

sns.boxplot(C, color= "green", ax=ax[2,0])
ax[2,0].set_title("Duration of Therapy for DrugCode C")
ax[2,0].set_xlabel("Number of days")

C_new = C.copy()
C_new.remove(520)
sns.boxplot(C_new, color= "green", ax=ax[2,1])
ax[2,1].set_title("Duration of Therapy for DrugCode C without outlier")
ax[2,1].set_xlabel("Number of days")

print("The distribution of therapy duration for drug A is a bit skewed to the left. It's ske
wness is",  skew(A))

print("\nOne datapoint stands out from the distribution of therapy duration for drug B: Pati
entID 7242 has been treated for more than 114 days.")
print("We can think of the above datapoints as an outlier and plot the distribution disregar
ding the datapoint.")
print("The new distribution of therapy duration for drug B is approximately symmetric and ju
st slightly skewed to the right. It's skewness is",  skew(B_new))

print("\nOne datapoint stands out from the distribution of therapy duration for drug C: Pati
```

```
The distribution of therapy duration for drug A is a bit skewed to the left. It's skewness is
-0.5669376185649837

One datapoint stands out from the distribution of therapy duration for drug B: PatientID 7242
has been treated for more than 114 days.
We can think of the above datapoints as an outlier and plot the distribution disregarding the
datapoint.
The new distribution of therapy duration for drug B is approximately symmetric and just sligh
tly skewed to the right. It's skewness is 0.18742239669160704

One datapoint stands out from the distribution of therapy duration for drug C: PatientID 7242
has been treated for more than 500 days.
We can think of the above datapoints as an outlier and plot the distribution disregarding the
```



In [21]:
```
# make 2 new dataframes to compare data with outliers and without outliers
a= pd.DataFrame(A, columns = ['days'])
a['DrugCode'] = 'A'
b= pd.DataFrame(B, columns = ['days'])
b['DrugCode'] = 'B'
c = pd.DataFrame(C, columns = ['days'])
c['DrugCode'] = 'C'
distribution= a.append(b)
distribution= distribution.append(c)
# data without outliers
b_new= pd.DataFrame(B_new, columns = ['days'])
b_new['DrugCode'] = 'B'
c_new = pd.DataFrame(C_new, columns = ['days'])
c_new['DrugCode'] = 'C'
distribution_new= a.append(b_new)
distribution_new= distribution_new.append(c_new)

#plotting
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(15, 9), sharey=True)
sns.boxplot(x="days", y="DrugCode", data=distribution, ax= ax1)
sns.boxplot(x="days", y="DrugCode", data=distribution_new, ax = ax2)
ax1.set_title('Duration of Therapy')
ax2.set_title('Duration of Therapy (without outliers)')

# print the summary statistics
print("Now let's take a look at the summary statistics while ignoring the outliers (outlier
 point comes from the same patient)")

print("\nMean therapy duration for drug A:", st.mean(A),", B:",st.mean(B_new),", C:",st.mean
(C_new) )
print("Median therapy duration for drug A:", st.median(A),", B:",st.median(B_new),", C:",st.
median(C_new) )
print("The Spread of therapy duration for drug A:", max(A)-min(A),", B:",max(B_new)-min(B_ne
w),", C:",max(C_new)-min(C_new))
print("Maximum therapy duration for drug A:", max(A),", B:",max(B_new),", C:",max(C_new) )
print("Minimum therapy duration for drug A:", min(A),", B:",min(B_new),", C:",min(C_new) )

print("\nDrugCode A has the lowest mean therapy duration, followed by B and C")
print("DrugCode B has the lowest median therapy duration, followed by A and C")
```

```
Now let's take a look at the summary statistics while ignoring the outliers (outlier point co
mes from the same patient)

Mean therapy duration for drug A: 56.13333333333333 , B: 59.25 , C: 84.86666666666666
Median therapy duration for drug A: 60 , B: 59.0 , C: 84
The Spread of therapy duration for drug A: 50 , B: 49 , C: 40
Maximum therapy duration for drug A: 77 , B: 85 , C: 106
Minimum therapy duration for drug A: 27 , B: 36 , C: 66

DrugCode A has the lowest mean therapy duration, followed by B and C
DrugCode B has the lowest median therapy duration, followed by A and C

The Mean, Median and the spread of therapy duration for A and B are very similar when compare
```



Done