2006 Pike's Peek 10k Data Exploration

Mission Analytics Data Exercise

Freddie Abel Perez

Questions for Exploration:

- 1.) What are the *Mean*, *Median*, *Mode*, and *Range* of the race results for all racers by *Gender*?
- 2.) Analyze the *difference* between *gun* and *net time* race results.
- 3.) How much *time* separates *Chris Doe* from the top *10 Percentile* of Racers of the same *Division*?
- 4.) Compare the Race Results of each *Division*.

Data Cleansing:

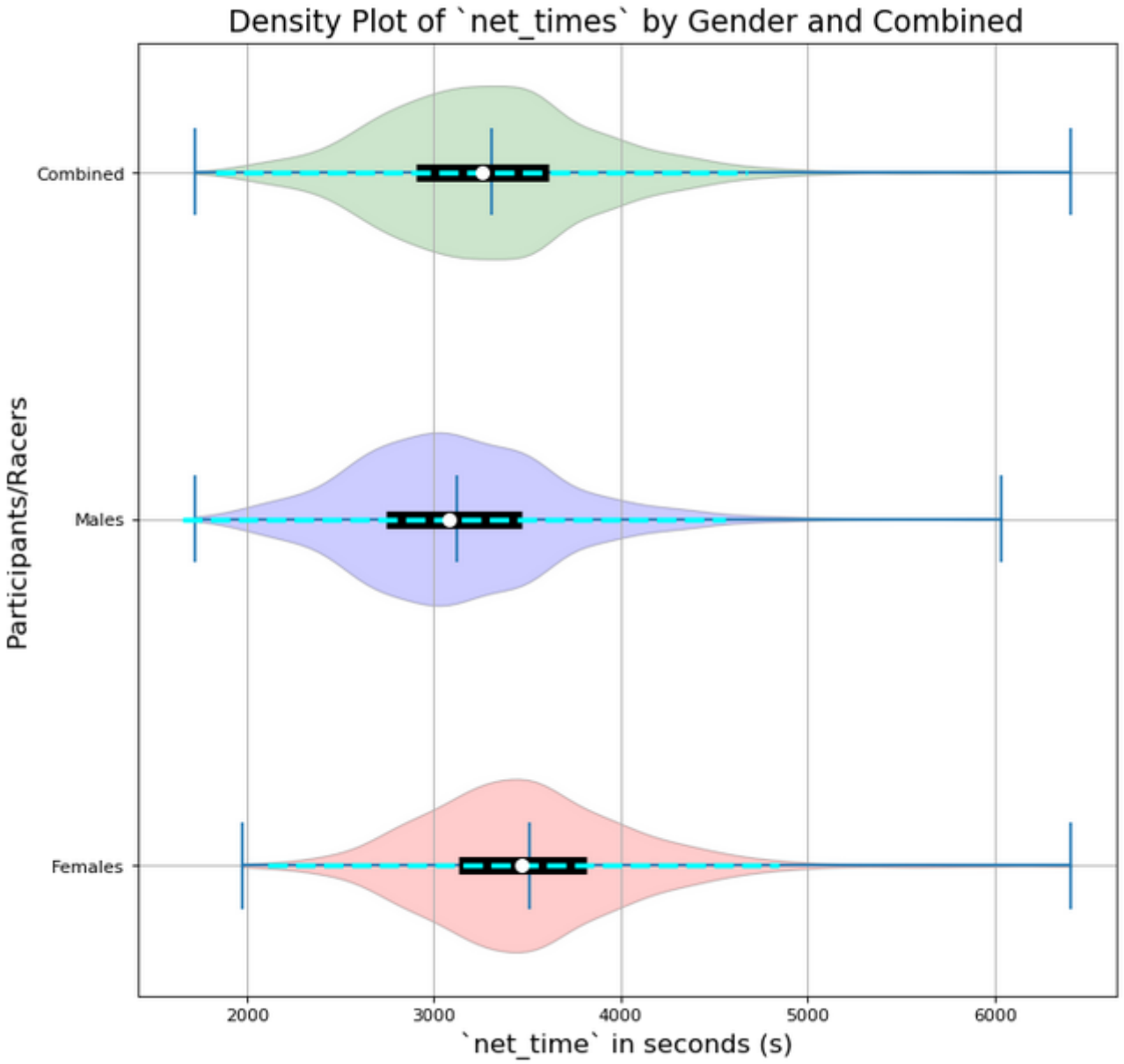
- When first looking at the two datasets to get a feel of how to clean/handle them prior to visualizations, I observed the following:
- **Missing Data**:
  - ***Hometown***: Some entries had missing City, State, or international values entered in the `hometown` column.
    - Missing state values inferred based on area research
    - 35 missing values left in Males.txt file after filling in.
    - 26 missing values left in Females.txt file after filling in.
  - ***Age***: Some of the contestants either had missing ages, an age set to 0 or less than 0 randomly scattered throughout.
    - 4 age values were missing within the Males.txt file
    - 4 age values were missing within the Females.txt file.

Data Cleansing:

- There were many instances unstructured discrepancies or injected special characters mixed in within features in the data sets.
  - **Net Time, Gun Time:** Features have varying datetime structures that needed alignment.
  - **Div/tot:** Very unstructured, mismatched information, division number did not correlate with age column.
    - Featured Engineered new Division Feature based on Age
  - **Text Files:** There was an issue with the encoding, ingested using `utf-8` and `latin-1`
  - **Text String Normalization:** Applied Regex Pattern matching, removed excess whitespaces.

Data Features Dictionary:

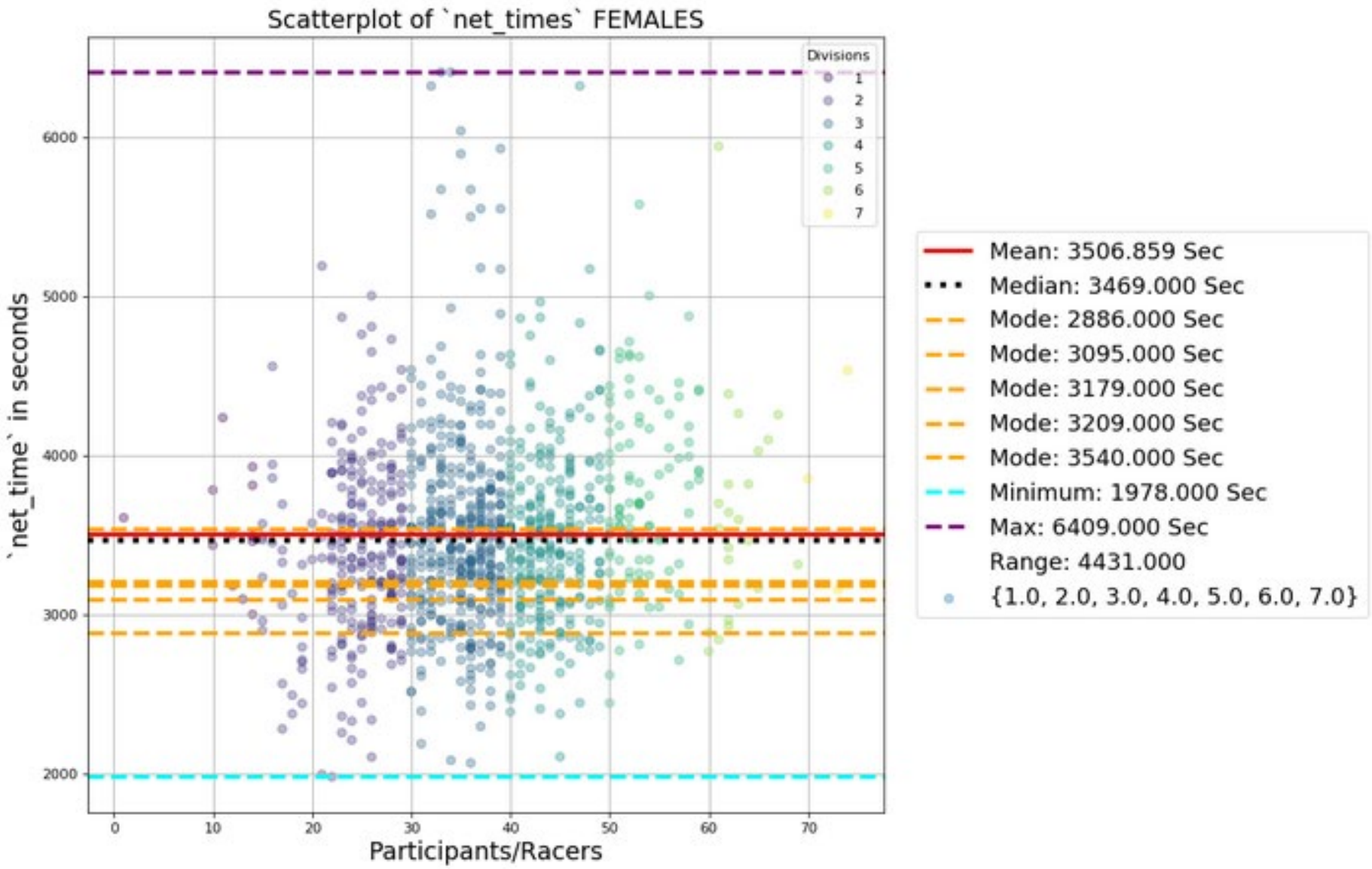
- ***place***: The order in which each racer finished relative to racers of the same gender.
- ***age***: Age of the racer.
- ***division_new***: New Column that parses out the division of the racers based on their age (originally from ***div/tot*** & ***age***).
- ***num***: Racer's bib number.
- ***name***: Name of racer.
- ***city***: Hometown of the racer (originally from ***hometown*** column)
- ***state***: State of the racer (originally from ***hometown***)
- ***gun_time***: Elapsed time from the formal start of the race and when the racer crossed the finish line.
- ***net_time***: Elapsed time from when the racer crossed the starting line and when they crossed the finish line.
- ***diff_time***: Difference of time between ***gun_time*** & ***net_time*** (for question 2)
- ***pace***: Racer's average time per mile during the race.



- **Initial Observations:**
  - Looking at the violin plot with respect to Net Time for each participants, it seems like the Male participants performed better overall in the 10k Race compared to the Female participants.
  - There is a wide spread of Net Time values among the participants, which may be used to show the spread in level of fitness among them as well.
  - Overall, a large number of each subpopulation (Male, Female, and the Combined) are densely located about the Mean and Median Values with slight variations in the distributions.

Female Contestants Statistics:

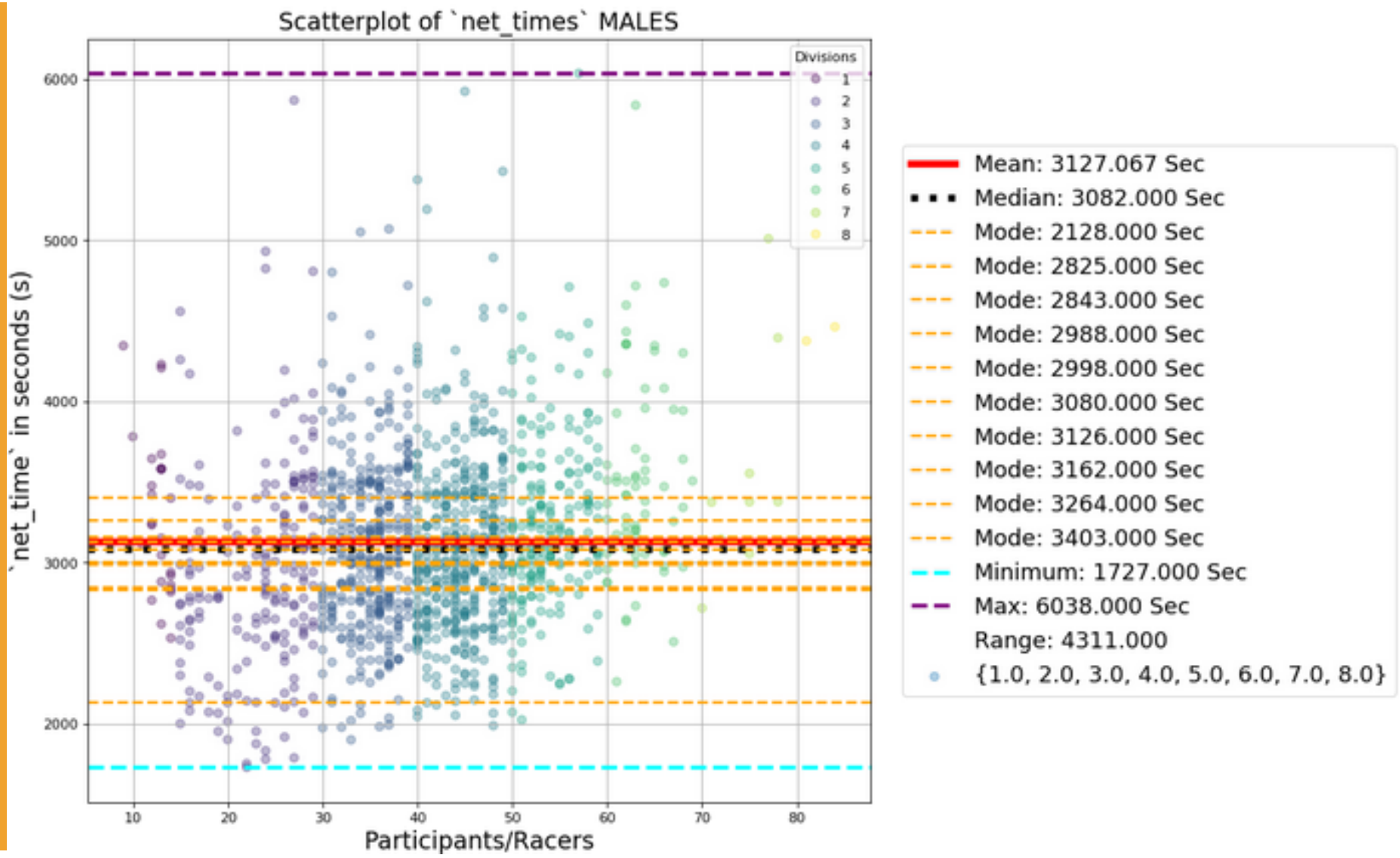
Mean = 58 mins 26.9 sec
Median = 57 mins 49.2 sec
Range = 73 mins 51 sec
Modes =
48 mins 6 sec
52 mins 36.6 sec
52 mins 59.4 sec
53 mins 14.4 sec
59 mins 0 sec

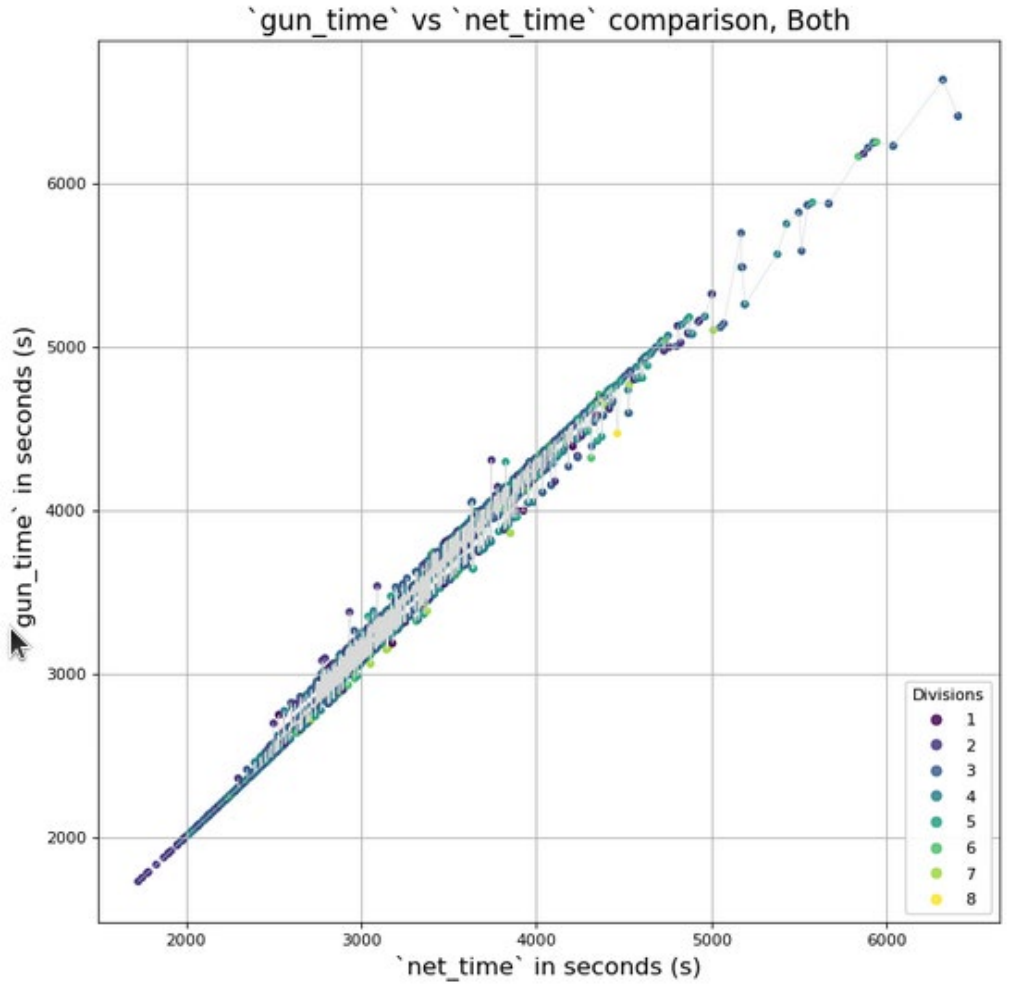


Male Contestants Statistics:

Mean = 52 mins 7.06 sec
Median = 51 mins 22 sec
Range = 71 mins 51 sec
Mode =

35 mins 28 sec
47 mins 5 sec
47 mins 23 sec
49 mins 48 sec
49 mins 58 sec
51 mins 20 sec
52 mins 6 sec
52 mins 42 sec
54 mins 24 sec
56 mins 43 sec

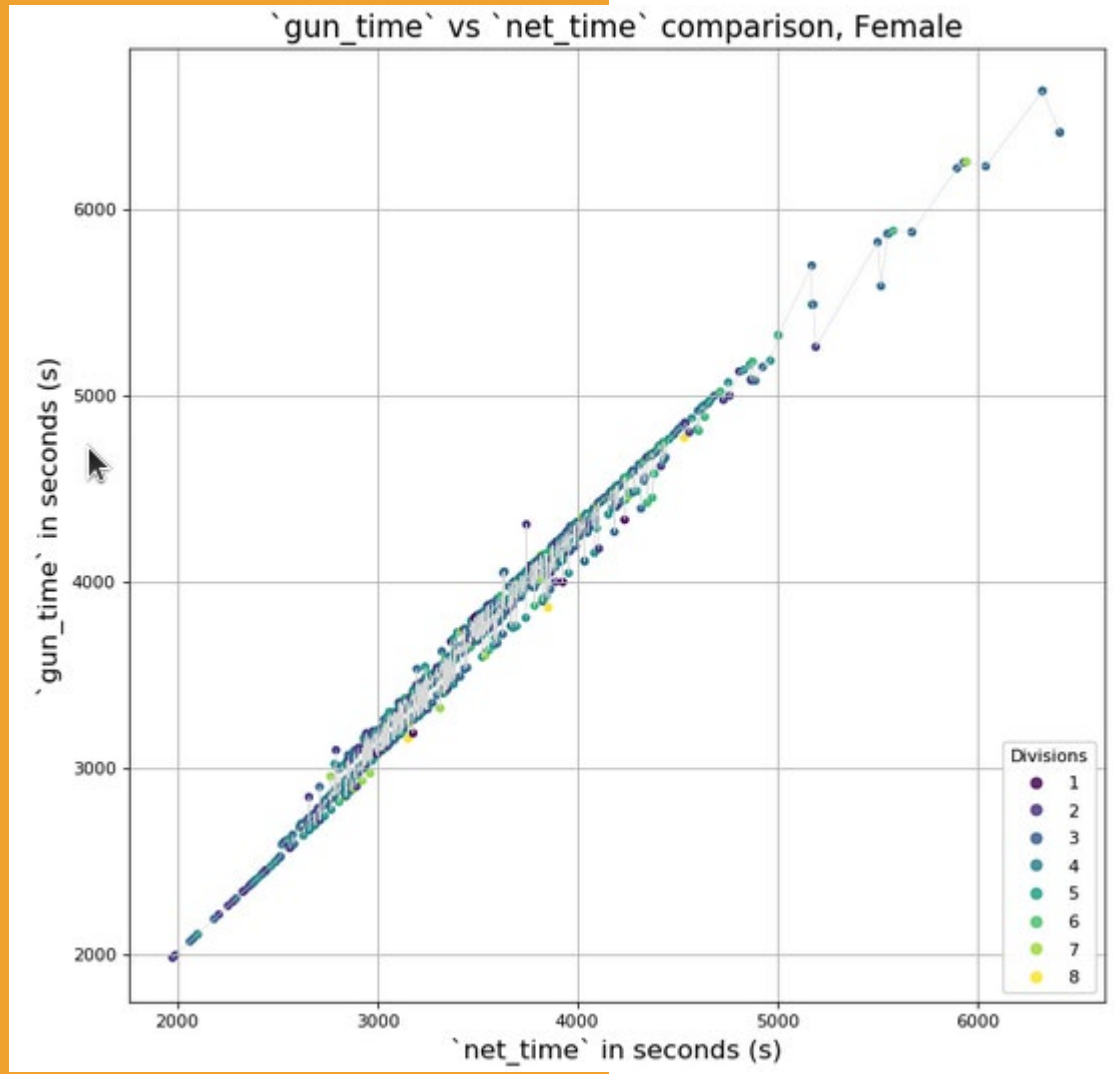




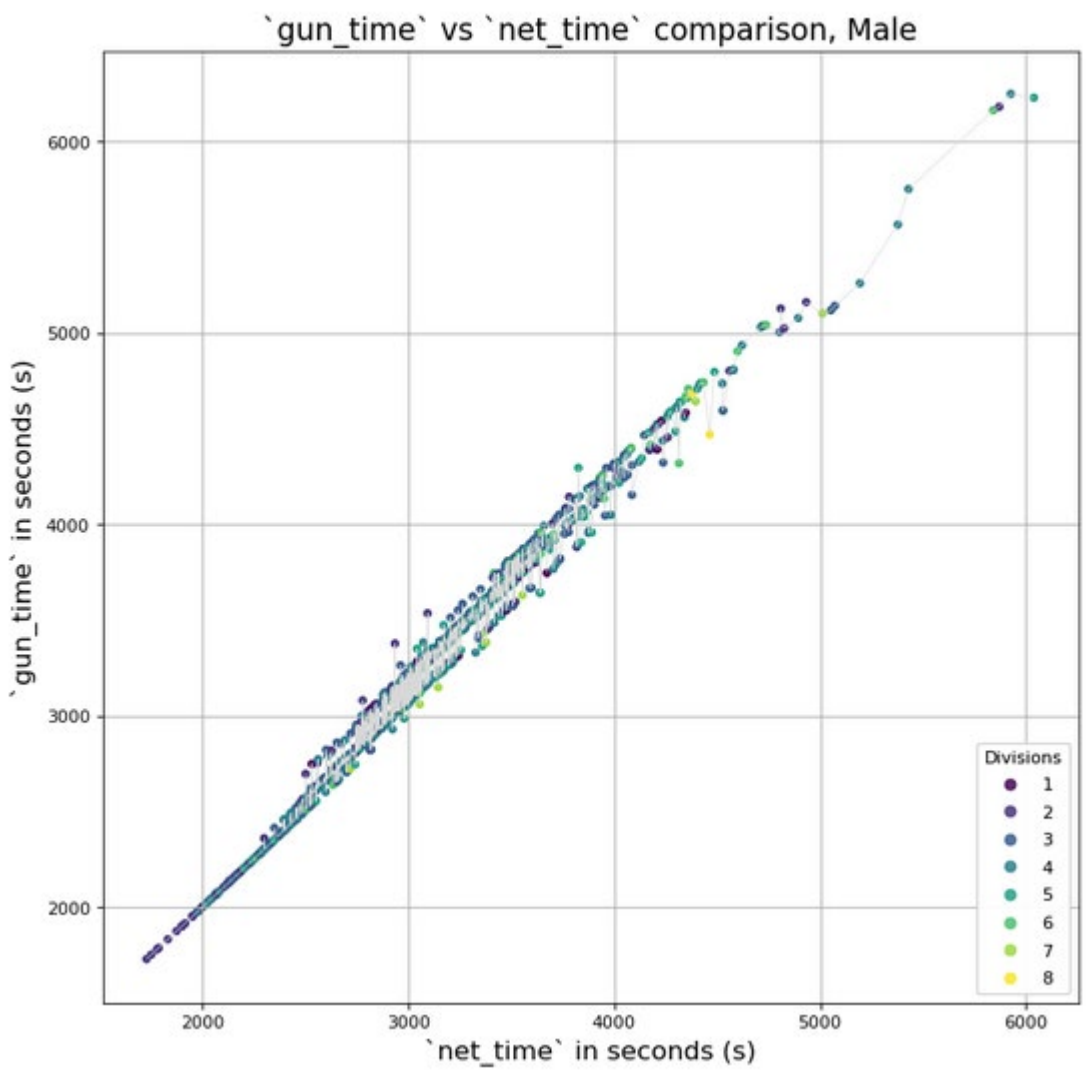
Pearson Correlation Value: 0.9942

- Given the two different time features available `gun_time` and `net_time`, some observations include:
  - A noticeable difference toward the center/Interquartile range of the whole data set.
  - The graph shows it widening as the `net_time` gets larger. This could capture more general or casual runners
  - The points closer to the origin seem to have no difference between their `net_time` and their `gun_time` values.
  - These racers could be seen more dedicated to the race, or were at a more favorable starting position to allow these two values to align more.
  - Toward the upper most right corner we can see that population as even more casual runners.
  - We can see this trend continue when looking at specific genders.

Difference in Gun and Net Time By Gender

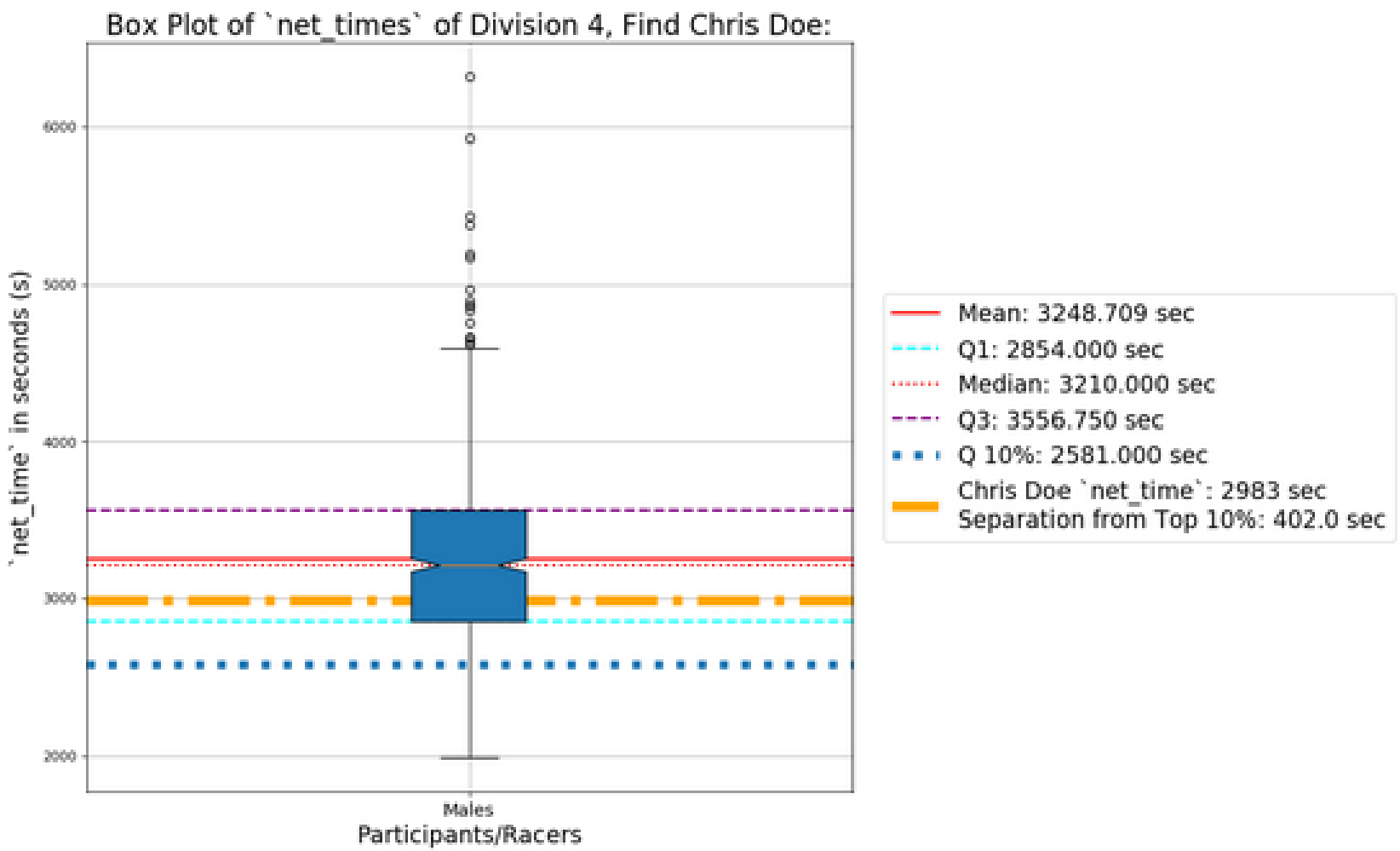


Pearson Correlation Value:
0.99486

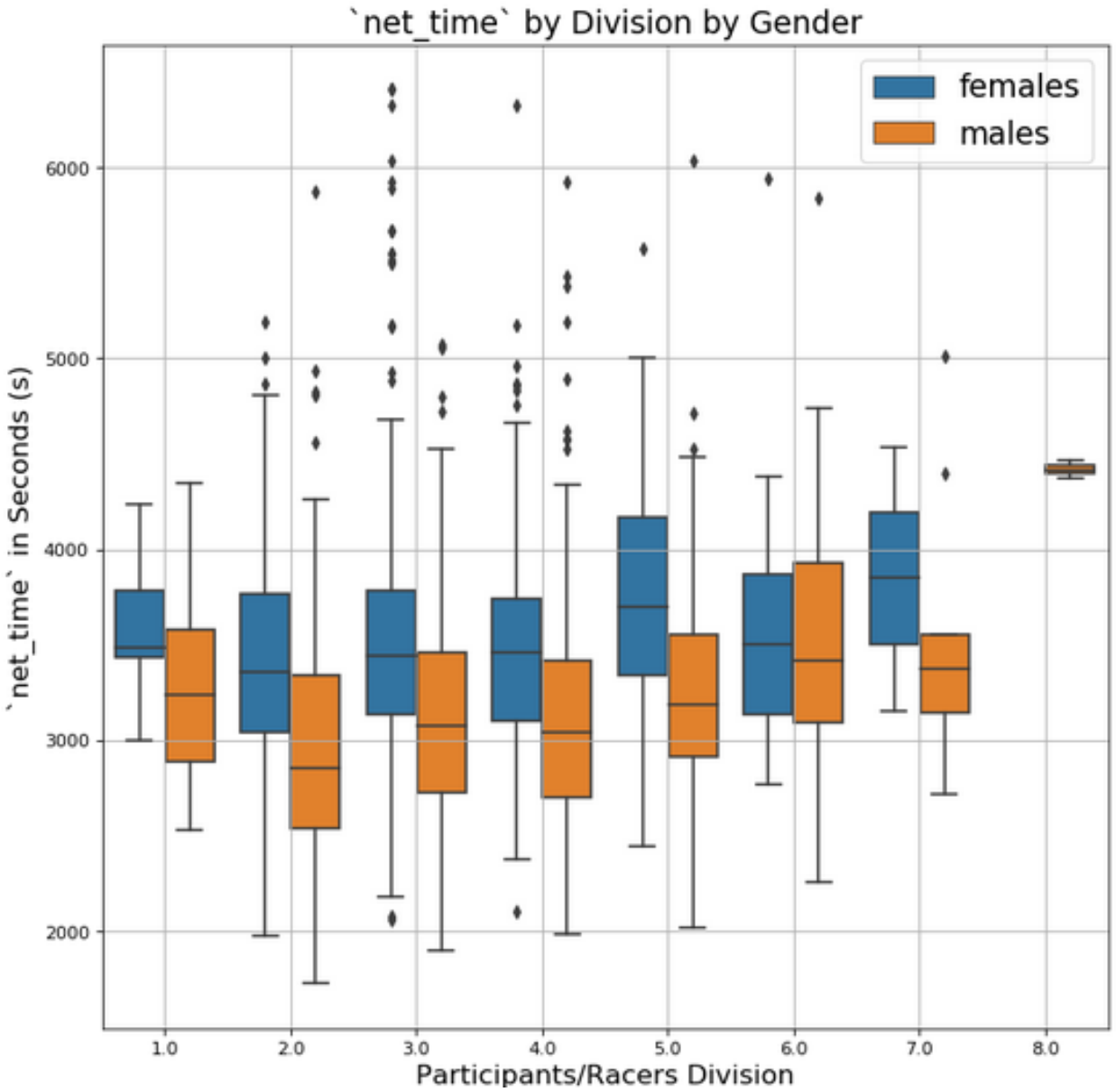


Pearson Correlation Value:
0.994457

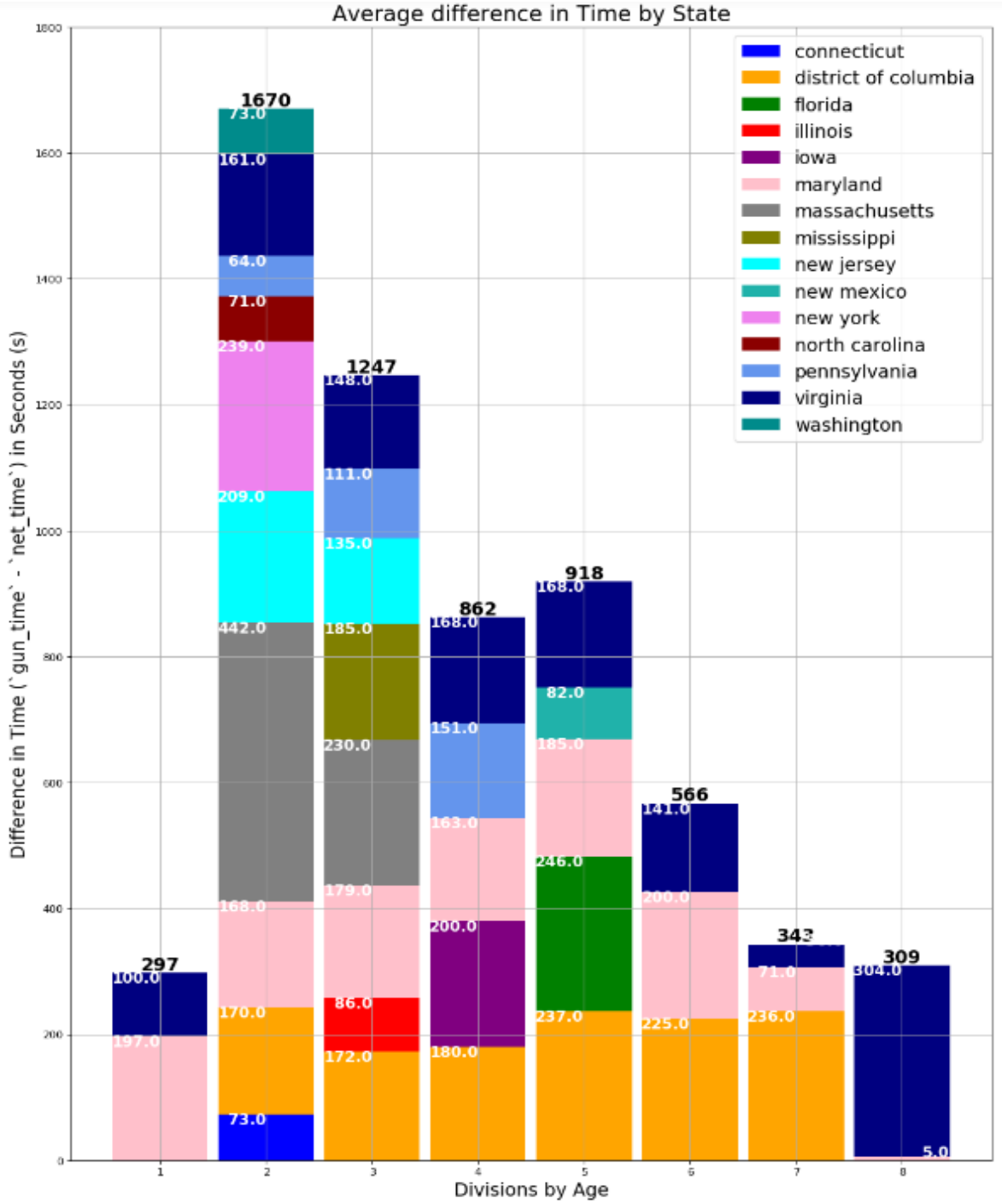
How well does Chris Doe Perform?



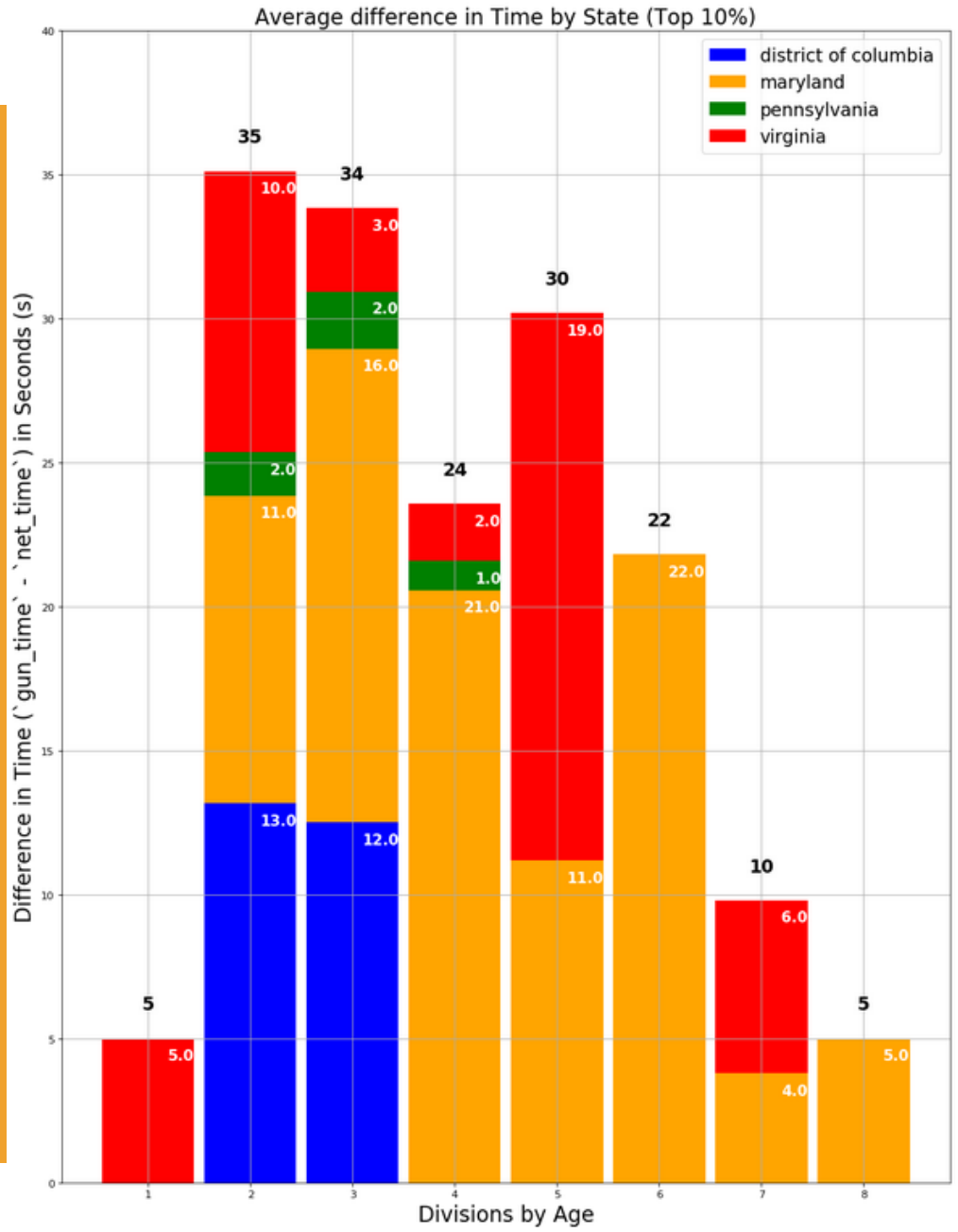
- Chris Doe's Net Time performance falls between the Q1 (25^{th}) and the Median (50^{th}) percentiles of those participating within his division (4^{th}).
- The Top 10 Percentile completed the race within 43mins 1sec (2581 secs).
- Chris Doe comes in at 49 mins, 43 secs (2983 secs), 6mins 21secs (402 secs) longer when compared to the threshold above for the top 10 Percentile.



- After comparing the racers by division, some insights include:
- Divisions 2, 3, & 4were the fastest (looking at the Minimum and the overall Average values) among both Male and Female Participants in the 2006s Pike's Peak 10k Race.
- Most of the runners seem to fall within Divisions 2-5 (ages 15-49) while Divisions 1 (ages 0-14) and Divisions 6-8 (ages 51-79) contained the least



- When viewing the divisions by State, we can see that the majority of the racers are from the DMV area.
- The Maryland, followed by Virginian and Washington DC Contestants.
- Looking at the most represented populations of Marylanders, Virginian, and DC Participants, it seemed like Virginian Racers had the most difference between Gun and Net time, followed by the Marylanders



- Looking at the Top 10 percentile in the average difference of Gun and Net time, we can see that a few participants from Pennsylvania (4 Total) had the best overall time.
- This is followed by Virginian (32 Total) and Maryland (181 Total) participants.
- Within the Top 10 Percentile of participants, DC Participants (12 total) had the least amount of time difference.

Follow on Steps:

A few points that I'd like to follow up would involve:

- The large population from Washington State
- Further explore the various outliers that can be associated with Division 1 (ages 1-14) and Division 8 (70-79). See if these were outliers or active participants at these ages.
- Potentially generate a geographical overlay that utilizes both City and State information to further explore the concentration of all the racers hometowns across the United Continental States and International countries.

Q & A?

Resources:

- *MA_Exer_PikesPeak_Female.txt*
- *MA_Exer_PikesPeak_Male.txt*
- *FAQ MA Pikes Peak Data Exercise.docx*
- *MA Exercise_20170112-2.pdf*

Code Book:

```
#!/home/rperez/miniconda3/envs/deloitte/bin/python
"""
This module contains the scripts and calls used to examine the
Pike's Peek 10K datasets for the Deloitte Data Exercise.
"""

import string
import re
import pandas as pd
import numpy as np
from datetime import datetime

def try_parsing_date(text):
    """
    Parsing out date and time values from different variations of time entries
    """
    for fmt in ('%H:%M:%S', '%M:%S', ':%S'):
        try:
            return datetime.strptime(text, fmt)
        except ValueError:
            pass
    raise ValueError('no valid date format found')

def rreplace(s, old=' ', new=', ', occurrence=1):
    li = s.rsplit(old, occurrence)
    return new.join(li)

def division_parser(x):
    """
    Parsing out actual divisions number based on """
    if (pd.isnull(x) or x<0):
        return np.nan
    elif (x>0) and (x<=14):
        return 1
    elif (x>=15) and (x<=19):
        return 2
    else:
        return int(x/10)

class Deloitte:
    """
    A class that contains all of the scripts used for the exercise
    """
    def __init__(self):
        self.file_path = None
        self.f_df = None
        self.m_df = None
        self.all_df = None

    def clean_data(self):
        raw_data = pd.read_csv('data/raw/{}'.format(self.file_path),encoding='latin-1', sep='\t')

        # renaming column names
        raw_data.columns = map(str.lower, raw_data.columns)
        raw_data.rename(columns={'div/tot':'div_total', 'ag':'age','gun tim':'gun_time', 'net tim':'net_time'},\
                inplace=True)
        clean_cols = raw_data.columns.tolist()
        clean_cols.pop(3) # removes `name` column
        clean_cols.pop(4) # removes `hometown` column

        # cleaning special symbols from columns to normalize data
        for col in clean_cols:
            raw_data[col].replace(to_replace='[#*^a-zA-Z ]',value='',regex=True, inplace=True)
        raw_data['hometown'].replace(to_replace='[,.]', value='', regex=True, inplace=True)

        # cleaning special symbols from columns to normalize data
        for col in clean_cols:
            raw_data[col].replace(to_replace='[#*^a-zA-Z ]',value='',regex=True, inplace=True)
        raw_data['hometown'].replace(to_replace='[,.]', value='', regex=True, inplace=True)

        # Separating Hometown from the State
        raw_data['hometown'] = raw_data['hometown'].map(rreplace)
        raw_data[['city', 'state']] = raw_data.hometown.str.split(',', expand=True)
        raw_data['state'].replace(to_replace=' ', value='', regex=True, inplace=True)

        # Changing abbreviated state names to full names
        raw_data['state'] = raw_data['state'].map(abbrev_to_us_state)

        # Adding in missing values
        missing_states={'Ellicott City':'Maryland','Fredericksburg':'Virginia','North Potomac':'Maryland',\
                'Silver Spring':'Maryland','Washington':'District of Columbia'}
        subset = raw_data.loc[raw_data['city'].isin(missing_states.keys()),'city']
        raw_data.loc[subset.index,'state']=raw_data.loc[subset.index,'city'].map(missing_states)

        # Normalizing/fixing timed features
        time_cols = ['gun_time', 'net_time', 'pace']
        for col in time_cols:
            # Applies function to all rows
            raw_data[col] = raw_data[col].map(try_parsing_date)
            # Removes the default date
            raw_data[col] = raw_data[col] -datetime(1900, 1, 1)
            # Finding total time in seconds
            raw_data[col] = raw_data[col].dt.total_seconds()

        raw_data['diff_time'] = raw_data['gun_time']-raw_data['net_time']
        raw_data['division_new'] = raw_data['age'].map(division_parser)

        # Adding Gender Column
        gender = self.file_path.split('.')[0].split('_')[-1].lower()
        raw_data['gender'] = gender

        # Normalizing string values (lower)
        cols_lower = ['name', 'hometown','city','state']
        for col in cols_lower:
            raw_data[col] = raw_data[col].str.lower()
        if gender=='females':
            self.f_df = raw_data
        elif gender=='males':
            self.m_df = raw_data
        return raw_data


def combine(self):
    if all([self.f_df.empty==False, self.m_df.empty==False]):
        self.all_df = pd.concat(\
                [self.f_df, self.m_df],\
                axis=0).reset_index(drop=True)
    else:
        print("Please load both Female and Male Data Sets")
```

```
def vis_q1(self):
    # Visualizing Violin Boxplot (All three: Females, Males, Combined)
    figure(figsize=(10,10), dpi=80)
    colors = ['red','blue','green']
    net_array = np.array([self.f_df['net_time'], self.m_df['net_time'], self.all_df['net_time']])
    vp = plt.violinplot(net_array, vert=False, showextrema=True, showmeans=True)
    med1,med2,med3 = np.quantile(net_array[0],.50), np.quantile(net_array[1],.50), np.quantile(net_array[2],.50)
    plt.scatter([med1,med2,med3], [1,2,3], marker='o', color='white', s=50, zorder=3)
    q11,q21,q31 = np.quantile(net_array[0],.25), np.quantile(net_array[1],.25), np.quantile(net_array[2],.25)
    q13,q23,q33 = np.quantile(net_array[0],.75), np.quantile(net_array[1],.75), np.quantile(net_array[2],.75)

    whiskers_min = np.array([q11,q21,q31]) - (np.array([q13,q23,q33]) - np.array([q11,q21,q31])) * 1.5
    whiskers_max = np.array([q13,q23,q33]) + (np.array([q13,q23,q33]) - np.array([q11,q21,q31])) * 1.5

    plt.hlines([1,2,3],[q11,q21,q31], [q13,q23,q33], color='k', linestyle='-', lw=10)
    plt.hlines([1,2,3], whiskers_min, whiskers_max, color='aqua', linestyle='dashed', lw=3)
    plt.yticks([1,2,3],['Females','Males','Combined'])

    for i in range(len(vp['bodies'])):
        vp['bodies'][i].set(facecolor=colors[i])
        vp['bodies'][i].set(edgecolor='black')
        vp['bodies'][i].set(alpha=0.2)

    plt.title('Density Plot of `net_times` by Gender and Combined', size=17)
    plt.grid()
    plt.xlabel('`net_time` in seconds (s)', size=15)
    plt.ylabel('Participants/Racers', size=15)
    plt.show()

    # Visualizing Scatterplot & Stats for Females
    figure(figsize=(10,10), dpi=80)
    subset = self.f_df[self.f_df.division_new.notnull()]
    subset = subset[subset.division_new>0]
    colors = subset['division_new']
    scatter = plt.scatter(subset['age'], subset['net_time'], c=colors, alpha=0.35, label=set(colors))
    mu,mi,med,mx = subset['net_time'].describe().loc[['mean','min','50%','max']]

    mod = subset['net_time'].mode()
    meanli = plt.axhline(y=mu,color='red', linewidth=3, linestyle='-', label='Mean: {0:.3f} Sec'.format(mu))
    medli = plt.axhline(y=med,color='k', linewidth=4, linestyle='dotted', label='Median: {0:.3f} Sec'.format(med))

    modli=[]
    for i in range(len(mod)):
        modli.append(plt.axhline(y=mod[i],color='orange', linewidth=3,
                     linestyle='--', label='Mode: {0:.3f} Sec'.format(mod[i])))

    minli = plt.axhline(y=mi,color='aqua', linewidth=3,
                        linestyle='--', label='Minimum: {0:.3f} Sec'.format(mi))
    maxli = plt.axhline(y=mx,color='purple', linewidth=3,
                        linestyle='--', label='Max: {0:.3f} Sec'.format(mx))

    extra = plt.axhline(y=mi,linewidth=0, label='Range: {0:.3f}'.format(mx-mi))


    first_legend = plt.legend(*scatter.legend_elements(), title='Divisions',\
                              loc='upper right')
    plt.gca().add_artist(first_legend)

    plt.legend(handles=[meanli,medli,minli,maxli,extra].append(modli), bbox_to_anchor=(1.04,0.5),
               loc='center left',borderaxespad=0, prop={'size':16})
    plt.grid()
    plt.title('Scatterplot of `net_times` FEMALES', size=17)
    plt.ylabel('`net_time` in seconds', size=17)
    plt.xlabel('Participants/Racers', size=17)
    plt.show()
```

```
    # Visualizing Scatterplot & Stats for Males
    figure(figsize=(10,10), dpi=80)
    subset = self.m_df[self.m_df.division_new.notnull()]
    subset = subset[subset.division_new>0]
    colors = subset['division_new']
    scatter = plt.scatter(subset['age'], subset['net_time'],
                c=colors, alpha=0.35, label=set(colors))
    mu,mi,med,mx =\
        subset['net_time'].describe().loc[['mean','min','50%','max']]
    mod = subset['net_time'].mode()[0]

    mod = subset['net_time'].mode()
    meanli = plt.axhline(y=mu,color='red', linewidth=5,
                         linestyle='-', label='Mean: {0:.3f} Sec'.format(mu))
    medli = plt.axhline(y=med,color='k', linewidth=5,
                         linestyle='dotted', label='Median: {0:.3f} Sec'.format(med))

    #modeli = plt.axhline(y=mod,color='orange', linewidth=4,
    #                     linestyle='--', label='Mode: {0:.3f} Sec'.format(mod))
    modli=[]
    for i in range(len(mod)):
        modli.append(plt.axhline(y=mod[i],color='orange', linewidth=2,
                         linestyle='--', label='Mode: {0:.3f} Sec'.format(mod[i])))

    minli = plt.axhline(y=mi,color='aqua', linewidth=3,
                         linestyle='--', label='Minimum: {0:.3f} Sec'.format(mi))
    maxli = plt.axhline(y=mx,color='purple', linewidth=3,
                         linestyle='--', label='Max: {0:.3f} Sec'.format(mx))
    extra = plt.axhline(y=mi,linewidth=0, label='Range: {0:.3f}'.format(mx-mi))

    first_legend = plt.legend(*scatter.legend_elements(), title='Divisions',\
                              loc='upper right')
    plt.gca().add_artist(first_legend)

    plt.legend(handles=[meanli,medli,minli,maxli,extra].append(modli), bbox_to_anchor=(1.04,0.5),
               loc='center left',borderaxespad=0, prop={'size':16})
    plt.grid()
    plt.title('Scatterplot of `net_times` MALES', size=17)
    plt.ylabel('`net_time` in seconds (s)', size=17)
    plt.xlabel('Participants/Racers', size=17)
    plt.show()
```

```
# Correlation Plots between Gun and Net time
def vis_q2(self):

    # Combined Gender Correlation Data Plot
    figure(figsize=(10,10), dpi=80)
    subset = self.all_df[self.all_df.division_new.notnull()]
    subset = subset[subset.division_new>0].sort_values('net_time')
    colors = subset['division_new']

    scatter = plt.scatter(subset['net_time'], subset['gun_time'],\
              c=colors, alpha=1, s=15, label=set(colors))
    plt.plot(subset['net_time'], subset['gun_time'],\
            linewidth=.5, color='0.85')
    first_legend = plt.legend(*scatter.legend_elements(), title='Divisions',\
                              loc='lower right')
    plt.gca().add_artist(first_legend)

    plt.grid()
    plt.title('`gun_time` vs `net_time` comparison, Both',size=17)
    plt.xlabel('`net_time` in seconds (s)', size=15)
    plt.ylabel('`gun_time` in seconds (s)',size=15)
    plt.show()

    # Female Correlation Data plot
    figure(figsize=(10,10), dpi=80)
    subset = self.f_df[self.f_df.division_new.notnull()]
    subset = subset[subset.division_new>0]
    colors = subset['division_new']

    plt.scatter(subset['net_time'], subset['gun_time'],\
              c=colors, alpha=1, s=15, label=set(colors))
    plt.plot(subset['net_time'], subset['gun_time'],\
            linewidth=.5, color='0.85')
    first_legend = plt.legend(*scatter.legend_elements(), title='Divisions',\
                              loc='lower right')
    plt.gca().add_artist(first_legend)

    plt.grid()
    plt.title('`gun_time` vs `net_time` comparison, Female', size=17)
    plt.xlabel('`net_time` in seconds (s)', size=15)
    plt.ylabel('`gun_time` in seconds (s)', size=15)
    plt.show()

    # Male Correlation Data Plot
    figure(figsize=(10,10), dpi=80)
    subset = self.m_df[self.m_df.division_new.notnull()]
    subset = subset[subset.division_new>0]
    colors = subset['division_new']

    plt.scatter(subset['net_time'], subset['gun_time'],\
              c=colors, alpha=1, s=15, label=set(colors))
    plt.plot(subset['net_time'], subset['gun_time'],\
            linewidth=.5, color='0.85')
    first_legend = plt.legend(*scatter.legend_elements(), title='Divisions',\
                              loc='lower right')
    plt.gca().add_artist(first_legend)

    plt.grid()
    plt.title('`gun_time` vs `net_time` comparison, Male',size=17)
    plt.xlabel('`net_time` in seconds (s)',size=15)
    plt.ylabel('`gun_time` in seconds (s)',size=15)
    plt.show()
```

```
def vis_q3(self):
    figure(figsize=(10,10), dpi=80)
    chrisdoe = self.all_df.loc[self.all_df.name=='chris doe',:]
    cd_nt = int(chrisdoe.net_time)
    net_array =\
        np.array(self.all_df.loc[self.all_df.division_new==float(chrisdoe.division_new),'net_time'])
    colors = ['blue']

    # Box plot generation
    bp = plt.boxplot(net_array, patch_artist=True, notch=True)
    q1, q50,q3,q10= np.quantile(net_array, [.25,.50,.75,.1])
    mu = net_array.mean()
    meanli = plt.axhline(y=mu,color='red', linewidth=2,
                         linestyle='-', label='Mean: {0:.3f} sec'.format(mu))
    q1li = plt.axhline(y=q1,color='aqua', linewidth=2,
                         linestyle='--', label='Q1: {0:.3f} sec'.format(q1))
    medli = plt.axhline(y=q50,color='red', linewidth=2,
                         linestyle='dotted', label='Median: {0:.3f} sec'.format(q50))
    q3li = plt.axhline(y=q3,color='purple', linewidth=2,
                         linestyle='--', label='Q3: {0:.3f} sec'.format(q3))
    q10li = plt.axhline(y=q10,linewidth=6,\
                         linestyle=':',label='Q 10%: {0:.3f} sec'.format(q10))

    cdli = plt.axhline(y=cd_nt, color='orange', linewidth=7,linestyle='-.',\
                       label='Chris Doe `net_time`: {} sec\nSeparation from Top 10%: {} sec'\
                      .format(int(chrisdoe.net_time), int(chrisdoe.net_time)-q10))
    plt.legend(handles=[meanli,q1li,medli,q3li, q10li, cdli], bbox_to_anchor=(1.04,0.5),\
               loc='center left', borderaxespad=0, prop={'size':17})

    plt.xticks([1],['Males'], size=13)
    plt.title('Box Plot of `net_times` of Division {}, Find Chris Doe:'\
              .format(int(chrisdoe.division_new)), size=20)
    plt.ylabel('`net_time` in seconds (s)', size=17)
    plt.xlabel('Participants/Racers', size=17)
    plt.grid()
    plt.show()
```

```python
def vis_q4(self):
    # Building Stacked histogram for average time difference per state
    import seaborn as sns
    subset = self.all_df[self.all_df.division_new.notnull()]
    subset = self.all_df[self.all_df.state.notnull()]
    subset = subset[subset.division_new>0].sort_values('net_time')
    color = ['blue','orange','green','red','purple',\
             'pink','gray', 'olive','cyan','lightseagreen','violet',\
            'darkred', 'cornflowerblue','navy','darkcyan'] #list(np.random.choice(range(256), size=15))

    color = color[:len(set(subset.state))]

    agg_stacked =\
        subset.groupby(['division_new','state'])['diff_time'].mean().unstack().fillna(0)

    fig, ax = plt.subplots(figsize=(15,20), dpi=80)
    bottom = np.zeros(len(agg_stacked))

    for i, col in enumerate(agg_stacked.columns):
        ax.bar(agg_stacked.index, agg_stacked[col],bottom=bottom,label=col,\
               width=.9, color=color[i])
        bottom+=np.array(agg_stacked[col])

    avg_totals = agg_stacked.sum(axis=1)
    y_offset=5

    for i, total in enumerate(avg_totals):
        ax.text(avg_totals.index[i], total+y_offset, round(total),\
                ha='center', weight='bold', size=17)

    y_offset=-15
    for i, bar in enumerate(ax.patches):
        if bar.get_height()>0:
            if bar.get_height()<50:
            # Putting the text in the middle of each bar
                ax.text(\
                    bar.get_x()+ bar.get_width(),\
                    bar.get_height() + bar.get_y(),\
                    round(bar.get_height()),
                    ha='right', color='w', weight='bold', size=14)
            else:
                ax.text(\
                    bar.get_x() +bar.get_width()/2,\
                    bar.get_height() + bar.get_y() +y_offset,\
                    round(bar.get_height()),
                    ha='right', color='w', weight='bold', size=14)

    plt.grid()
    ax.set_ylim([0,1800])
    ax.set_title('Average difference in Time by State', size=20)
    ax.set_xlabel('Divisions by Age' , size=17)
    ax.set_ylabel('Difference in Time (`gun_time` - `net_time`) in Seconds (s)', size=17)
    ax.legend(prop={'size':17})

    # Stacked bar graph with counts of each state
    subset = self.all_df[self.all_df.division_new.notnull()]
    subset = self.all_df[self.all_df.state.notnull()]
    subset = subset[subset.division_new>0].sort_values('net_time')

    color = ['blue','orange','green','red','purple',\
             'pink','gray', 'olive','cyan','lightseagreen','violet',\
            'darkred', 'cornflowerblue','navy','darkcyan'] #list(np.random.choice(range(256), size=15))

    color = color[:len(set(subset.state))]

    agg_stacked =\
        subset.groupby(['division_new','state'])['diff_time'].count().unstack().fillna(0)

    fig, ax = plt.subplots(figsize=(15,20), dpi=80)
    bottom = np.zeros(len(agg_stacked))

    for i, col in enumerate(agg_stacked.columns):
        ax.bar(agg_stacked.index, agg_stacked[col],bottom=bottom,label=col,\
               width=.9, color=color[i])
        bottom+=np.array(agg_stacked[col])

    avg_totals = agg_stacked.sum(axis=1)
    y_offset=1

    for i, total in enumerate(avg_totals):
        ax.text(avg_totals.index[i], total+y_offset, round(total),\
                ha='center', weight='bold', size=17)

    y_offset=-15
    for i, bar in enumerate(ax.patches):
        if bar.get_height()>0:
            if bar.get_height()<50:
            # Putting the text in the middle of each bar
                ax.text(\
                    bar.get_x()+ bar.get_width(),\
                    bar.get_height() + bar.get_y() -.75,\
                    round(bar.get_height()),
                    ha='right', color='w', weight='bold', size=14)
            else:
                ax.text(\
                    bar.get_x() +bar.get_width()/2,\
                    bar.get_height() + bar.get_y() -5,\
                    round(bar.get_height()),
                    ha='right', color='w', weight='bold', size=14)

    plt.grid()
    ax.set_ylim([0,1000])
    ax.set_title('Counts of Participants by State', size=20)
    ax.set_xlabel('Divisions by Age' , size=20)
    ax.set_ylabel('Difference in Time (`gun_time` - `net_time`) in Seconds (s)', size=20)
    ax.legend(prop={'size':17})
```

```
# Average difference by state (10 % percentile)
subset = self.all_df[self.all_df.division_new.notnull()]
subset = self.all_df[self.all_df.state.notnull()]
subset = subset[subset.division_new>0].sort_values('net_time')
q10= np.quantile(subset.diff_time, .1)
subset = subset[subset.diff_time<q10]
color = ['blue','orange','green','red','purple',\
         'pink','gray', 'olive','cyan','lightseagreen','violet',\
        'darkred', 'cornflowerblue','navy','darkcyan'] #list(np.random.choice(range(256), size=15))

color = color[:len(set(subset.state))]

agg_stacked =\
    subset.groupby(['division_new','state'])['diff_time'].mean().unstack().fillna(0)

fig, ax = plt.subplots(figsize=(15,20), dpi=80)
bottom = np.zeros(len(agg_stacked))

for i, col in enumerate(agg_stacked.columns):
    ax.bar(agg_stacked.index, agg_stacked[col],bottom=bottom,label=col,\
           width=.9, color=color[i])
    bottom+=np.array(agg_stacked[col])

avg_totals = agg_stacked.sum(axis=1)
y_offset=1

for i, total in enumerate(avg_totals):
    ax.text(avg_totals.index[i], total+y_offset, round(total),\
            ha='center', weight='bold', size=17)

y_offset=-15
for i, bar in enumerate(ax.patches):
    if bar.get_height()>0:
        if bar.get_height()<50:
        # Putting the text in the middle of each bar
            ax.text(\
                bar.get_x()+ bar.get_width(),\
                bar.get_height() + bar.get_y() -.75,\
                round(bar.get_height()),
                ha='right', color='w', weight='bold', size=14)
        else:
            ax.text(\
                bar.get_x() +bar.get_width()/2,\
                bar.get_height() + bar.get_y() -5,\
                round(bar.get_height()),
                ha='right', color='w', weight='bold', size=14)

plt.grid()
ax.set_ylim([0,40])
ax.set_title('Average difference in Time by State (Top 10%)', size=20)
ax.set_xlabel('Divisions by Age' , size=20)
ax.set_ylabel('Difference in Time (`gun_time` - `net_time`) in Seconds (s)', size=20)
ax.legend(prop={'size':17})

# Counts of participants by state (Top 10%)
subset = self.all_df[self.all_df.division_new.notnull()]
subset = self.all_df[self.all_df.state.notnull()]
subset = subset[subset.division_new>0].sort_values('net_time')
q10= np.quantile(subset.diff_time, .1)
subset = subset[subset.diff_time<q10]
```

```
color = ['blue','orange','green','red','purple',\
         'pink','gray', 'olive','cyan','lightseagreen','violet',\
        'darkred', 'cornflowerblue','navy','darkcyan']
color = color[:len(set(subset.state))]

agg_stacked =\
    subset.groupby(['division_new','state'])['diff_time'].count().unstack().fillna(0)

fig, ax = plt.subplots(figsize=(15,20), dpi=80)
bottom = np.zeros(len(agg_stacked))

for i, col in enumerate(agg_stacked.columns):
    ax.bar(agg_stacked.index, agg_stacked[col],bottom=bottom,label=col,\
           width=.9, color=color[i])
    bottom+=np.array(agg_stacked[col])

avg_totals = agg_stacked.sum(axis=1)
y_offset=1

for i, total in enumerate(avg_totals):
    ax.text(avg_totals.index[i], total+y_offset, round(total),\
            ha='center', weight='bold', size=17)

y_offset=-15
for i, bar in enumerate(ax.patches):
    if bar.get_height()>0:
        if bar.get_height()<50:
        # Putting the text in the middle of each bar
            ax.text(\
                bar.get_x()+ bar.get_width(),\
                bar.get_height() + bar.get_y() -.75,\
                round(bar.get_height()),
                ha='right', color='w', weight='bold', size=14)
        else:
            ax.text(\
                bar.get_x() +bar.get_width()/2,\
                bar.get_height() + bar.get_y() -5,\
                round(bar.get_height()),
                ha='right', color='w', weight='bold', size=14)

plt.grid()
ax.set_ylim([0,70])
ax.set_title('Counts of Participants by State (Top 10%)', size=20)
ax.set_xlabel('Divisions by Age' , size=20)
ax.set_ylabel('Difference in Time (`gun_time` - `net_time`) in Seconds (s)', size=20)
ax.legend(prop={'size':17})

# Side by Side Box Plot comparison of gender race results by division
figure(figsize=(10,10), dpi=80)
subset.head()
ax=sns.boxplot(x='division_new', y='net_time', hue='gender',data=subset)
plt.grid()
plt.title("`net_time` by Division by Gender",size=17)
plt.xlabel("Participants/Racers Division", size=15)
plt.ylabel("`net_time` in Seconds (s)", size=15)
plt.legend(prop={'size':17})
plt.show()
```