Mahidol University
Faculty of Medicine
Siriraj Hospital

Term Project Presentation

SIRE 504 Programming in Bioinformatics

Bowornpol Nuim

Master's degree student 6437932 SIMB/M

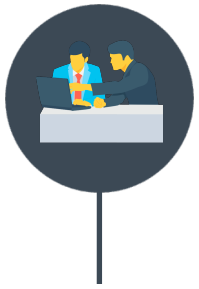
Naphat Sanguansakpakdee

Master's degree student 6437931 SIMB/M

Mattika Thaweesuvannasak

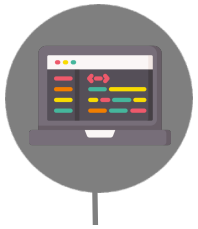
Master's degree student 6436307 SIBB/M

Project Timeline



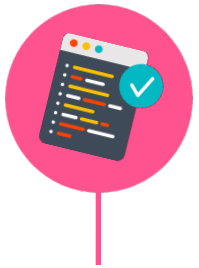
- Understand the input information (FASTQ)
- FASTQ → CSV

Oct 28, 2021



- Feature development
- Create the package [argparser()]

Nov 2 – 25, 2021



- Merge all features to develop via GitHub
- Recheck error in our program

Report: HTML

Nov 26, 2021

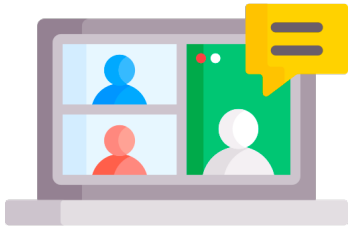
Term project presentation

Dec 2, 2021

Team collaboration

Approach

Slack



Online meeting

Via google meet, zoom

GitHub

On-site meeting

Advantage

- Clear assignment of work (feature development) to each members.
- Good time management.
- Availability for a meeting anytime.
- Easily collaborate with others, by letting other members contribute to your projects or you contributing to member's projects.

Team collaboration

Challenge

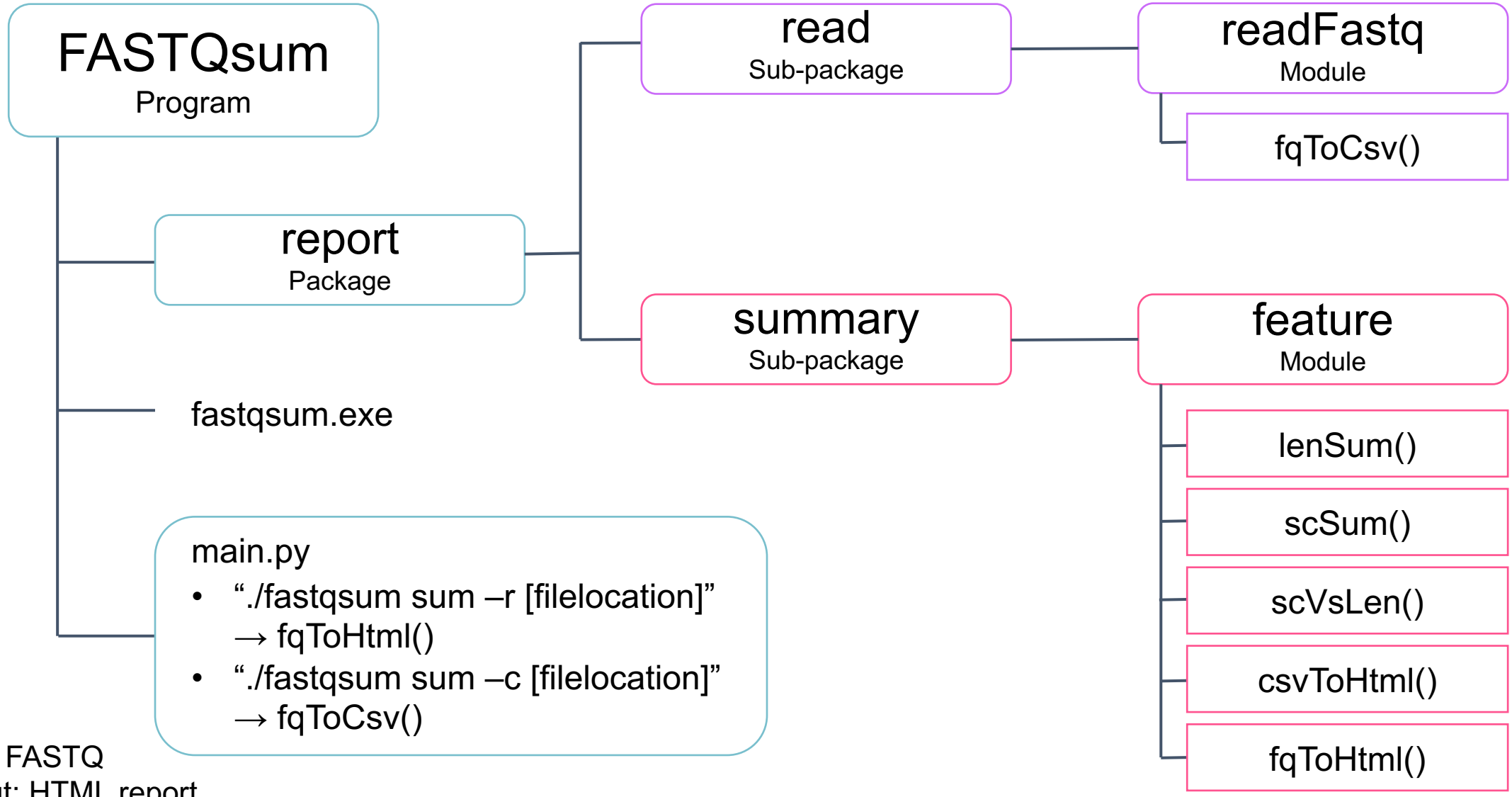
- GitHub: It’s quite complex system and spend the time a lot for understanding.
- HTML: It's quite our first time to create the html report.
- Plotly python package (visualization): It has a lot of the form to produce the results.

Solution

- Meeting and consults with our members.
- Consults with professor directly via chat box in slack, Q&A session, and on-site.

Our Program

Our Program



Input: FASTQ
Output: HTML report

Our Program

Input

- FASTQ file (.gz)

```
def getFq(filePath) :

    seqList=[]
    with gzip.open(filePath,'rt') as f:
    #with gzip.open('../fastq/ont.exp2.fastq.gz','rt') as f: # Decompress ont.exp2.fastq.gz
        for idx, seq_record in enumerate(SeqIO.parse(f, "fastq")): # Read and parse
            seqList.append(seq_record)
            if idx == 1000: # set how many reads we want
                break
    return seqList
```

Our Program

Parameters

- -r, --report (option to html report)
- -c, --save2csv (option to only .csv summary)

```
def argparserLocal():
    from argparse import ArgumentParser
    parser=ArgumentParser(prog='fastqsum',description='Summarize fastq to html report')

    subparsers = parser.add_subparsers(
        title='commands',description='Please choose command below:',
        dest='command'
    )
    subparsers.required = True
    sum_command = subparsers.add_parser('sum',help='Generate html report')
    sum_command.add_argument("-r","--report",type=str,default=None,
                             help="generate complete html report")
    sum_command.add_argument("-c","--save2csv",type=str,default=None,
                             help="generate only .csv summary")

    #print(parser.print_help())
    return parser
```

Our Program

Output

- HTML report

FASTQ SUM

Basecall summary

Basecall summary

Number of reads	Total bases	Mean read length	Median read length	Read length (N50)	Longest pass read
365178	1564374428	4283.868217691099	2679	7886	193225

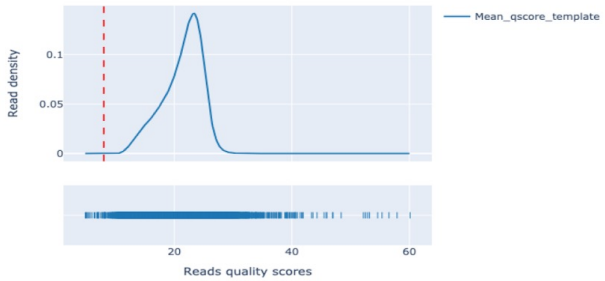
Note: In computational biology, N50 is statistics of a set of contig or scaffold lengths. The N50 is similar to a mean or median of lengths, but has greater weight given to the longer contigs. It is used widely in genome assembly, especially in reference to contig lengths within a draft assembly.

Basecalled reads length

Basecalled reads length for all barcodes

Basecalled reads length for all barcodes

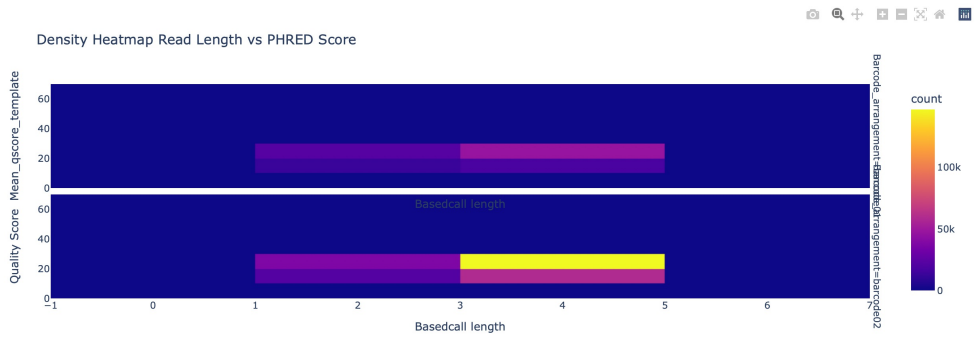
Basecalled reads PHRED quality



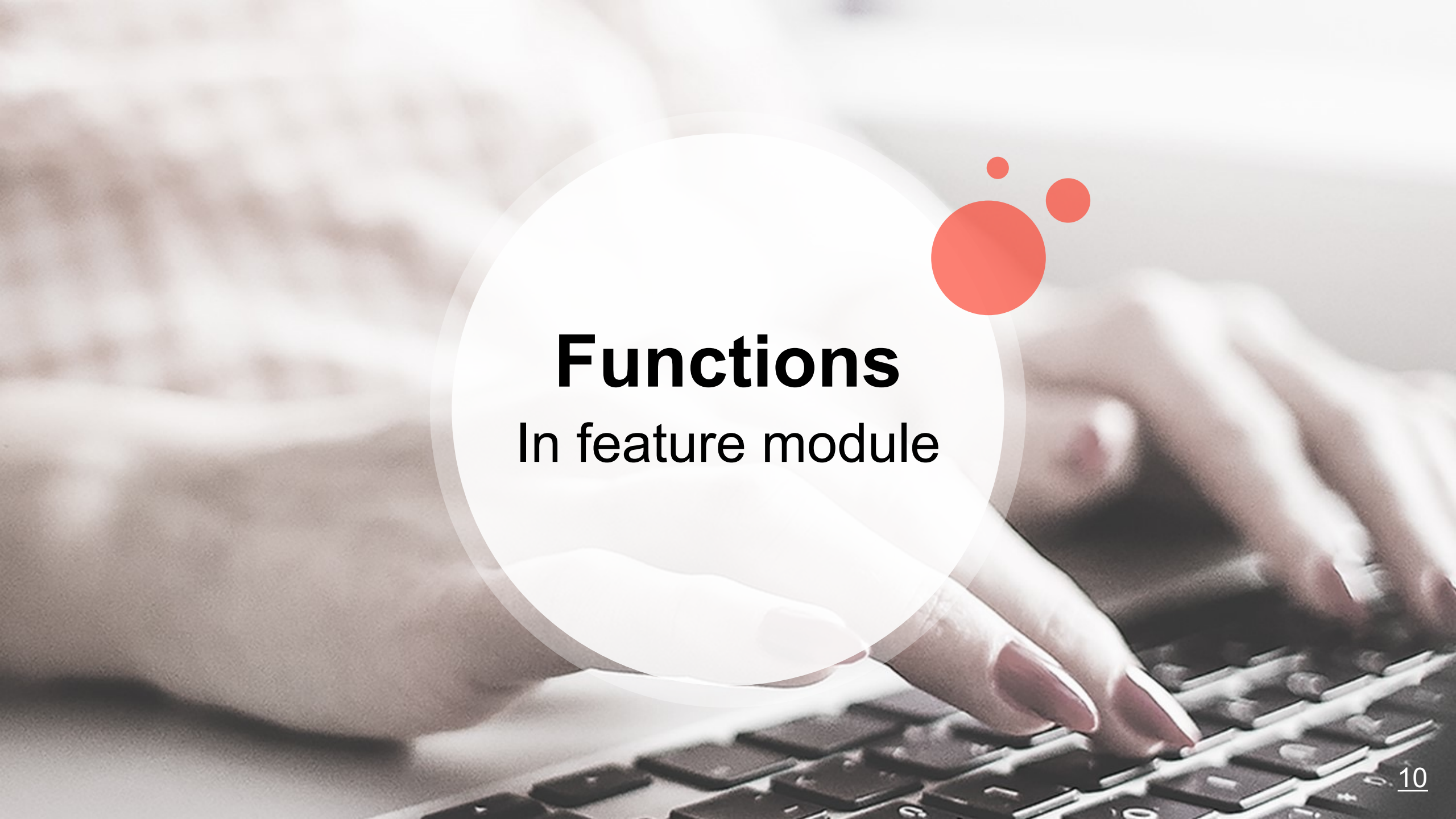
Red line: Cut-off line suggestion (Mean quality score at 8.0)

Explanation: Basecalled reads PHRED quality distplot represents the distribution of mean quality score for all reads.

Heat Map



Explanation: The heatmaps share the same x-y axis with scatter plot. These colorful interactive plot illustrate how many reads are in each range of read length and quality score. Heatmap draws a grid that count reads in each partition. To inteprete, yellow color represent high frequency (many reads). In contrast, purple color means the lower count. The read count of each shade is as displayed in the color bar.



Functions

In feature module

01 Read Length

By Mattika

Part 1: Calculate all values, including number of reads, total base, Mean read length, Median read length, Read length (N50), and Longest pass read

Python code:

```
86  csvOut = pd.read_csv('test-1.csv')
87  templatelength = []
88  templatelength = csvOut['Sequence_length_template']
89  templatelength = list(templatelength)
90  numberreads = len(csvOut)
91  totalbase = sum(templatelength)
92  meanlength = statistics.mean(templatelength)
93  medianlength = statistics.median(templatelength)
94
95  halflength = totalbase/2
96  lengthsort = templatelength
97  lengthsort.sort(reverse=True)
98
99  total = 0
100
101 for i in lengthsort:
102     total += i
103     if total >= halflength:
104         break
105
106 N50size = i
107
108 lengthsort = templatelength
109 lengthsort.sort()
110 maxread = max(lengthsort)
```

Output:

```
Number of reads    :  120001
Total base         :  434722429
Mean read length   :  3622.6567195273374
Median read length :  2046
Read length (N50)  :  7103
Longest pass read  :  193225
```

N50 size

Def: 50% of the genome is in contigs as large as the N50 value

Example: 1 Mbp genome

N50 size = 30 kbp
(300k+100k+45k+45k+30k = 520k >= 500kbp)

01 Read Length

By Mattika

Part 2: Make table with Plotly (plotly.graph_objects)

Python code:

```
120  figTable = go.Figure(data=[go.Table(header=dict(values=['Number of reads','Total bases','Mean read length',
121                'Median read length','Read length (N50)','Longest pass read'], fill_color= 'lavender'),
122                cells=dict(values=[numberreads, totalbase, meanlength, medianlength, N50size, maxread],
123                fill_color= 'lightcyan'))], layout=go.Layout(title=go.layout.Title(text="Basecall summary")
124                ))
125
126  figTable.show()
127  # figTable.write_image("figTable.png")
128  # figTable.write_html('figTable.html')
```

Output:

Basecall summary

Number of reads	Total bases	Mean read length	Median read length	Read length (N50)	Longest pass read
365178	1564374428	4283.868217691099	2679	7886	193225

01 Read Length

Part 3: Make graph for all barcodes with Plotly (plotly.figure_factory)

By Mattika

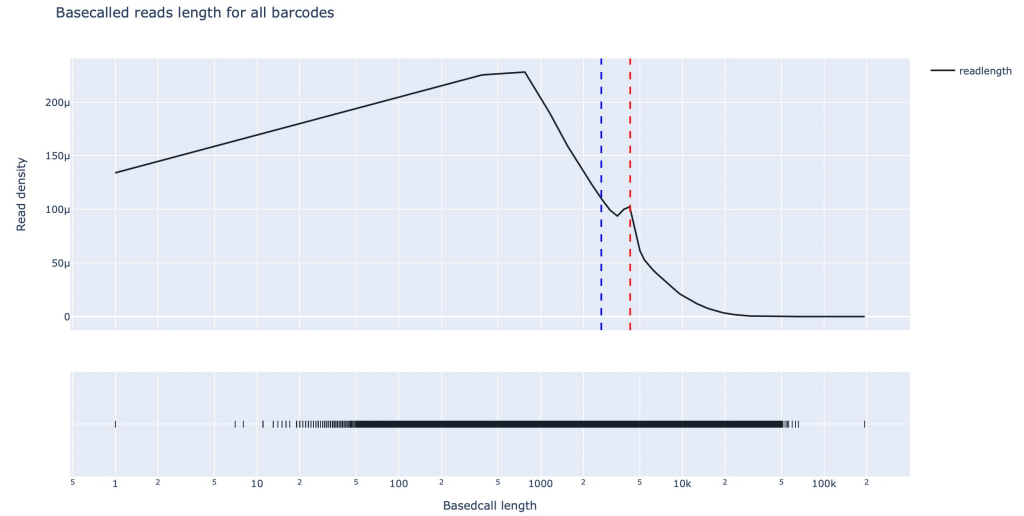
Python code:

```
group_label = ['readlength']

lenfig = ff.create_distplot([csvOut['Sequence_length_template']], group_label, colors = ['#17202A'],
                            show_hist = False)
lenfig.update_layout(
    title="Basecalled reads length for all barcodes",
    xaxis_title="Basedcall length",
    yaxis_title="Read density",
)
lenfig.add_vline(x=meanlength, line_width=2, line_dash="dash", line_color="Red",
                 annotation_text="Mean", annotation_position="top right")
lenfig.add_vline(x=medianlength, line_width=2, line_dash="dash", line_color="Blue",
                 annotation_text="Median", annotation_position="top left")
lenfig.update_xaxes(type='log')

lenfig.show()
lenfig.write_image("lenfig.png")
```

Output:



01 Read Length

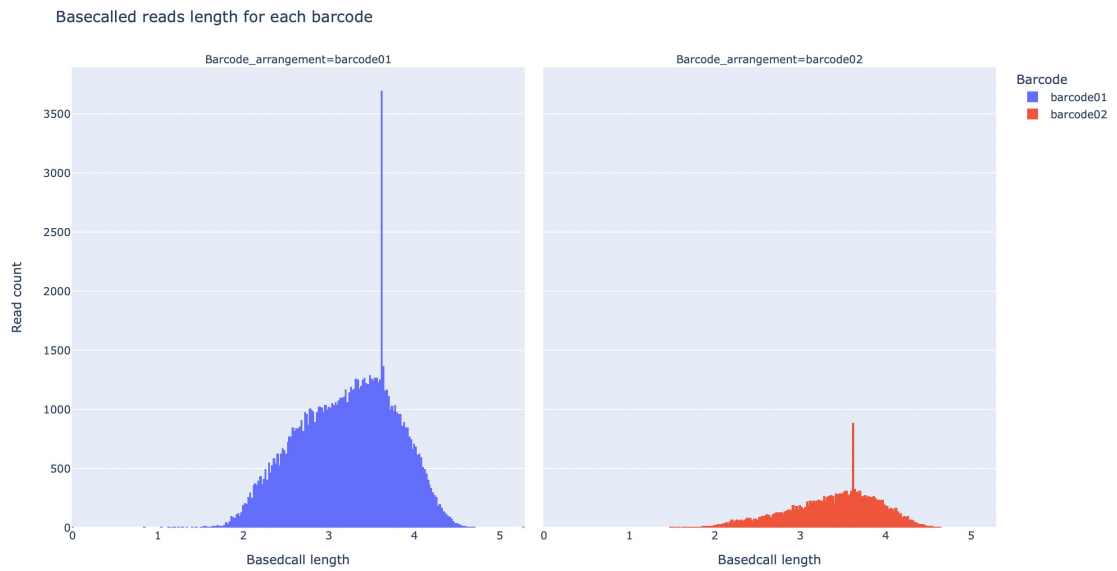
Part 4: Make graph for each barcode with Plotly (plotly.express)

By Mattika

Python code:

```
152  csvOut['log_Sequence_length_template']=np.log10(csvOut['Sequence_length_template'])
153  lenfigSplit = px.histogram(data_frame=csvOut, x=csvOut['log_Sequence_length_template'],
154                             facet_col='Barcode_arrangement', color='Barcode_arrangement',
155                             title="Basecalled reads length for each barcode", log_x=False)
156  lenfigSplit.update_layout(
157      xaxis_title="Basedcall length",
158      xaxis2=dict(title="Basedcall length"),
159      yaxis_title="Read count",
160      legend_title="Barcode"
161  )
162  lenfigSplit.show()
163  lenfigSplit.write_image("lenfigSplit.png")
```

Output:



01 Read Length

Part 5: Write HTML

By Mattika

Python code:

```
96    ## Write HTML
97    html_template = '''<!doctype html>
98    <html lang="en">
99    <head>
100   </head>
101   <body>
102   {results}
103   </body>
104   </html>
105   '''
106
107   figTable_html = '<div><h2>Basecall summary</h2>'+figTable.to_html(full_html=False, include_plotlyjs='cd
108   lenfig_html = '<div><h2>Basecalled reads length</h2><h3 style="color:Navy;"><ins>Basecalled reads lengt
109   lenfigSplit_html = '<div><h3 style="color:Navy;"><ins>Basecalled reads length for each barcode</ins></h
110
111   results = figTable_html+lenfig_html+lenfigSplit_html
112   print("Read Length Summary Complete")
113   return results
```

Output:

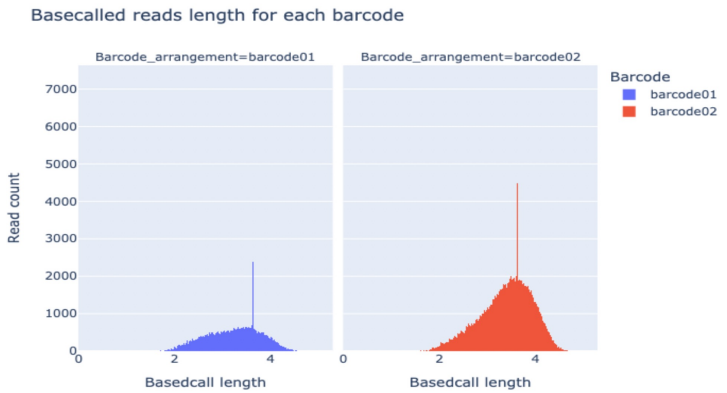
Basecall summary

Basecall summary

Number of reads	Total bases	Mean read length	Median read length	Read length (N50)	Longest pass read
365178	1564374428	4283.868217691099	2679	7886	193225

Note: In computational biology, N50 is statistics of a set of contig or scaffold lengths. The N50 is similar to a mean or median of lengths, but has greater weight given to the longer contigs. It is used widely in genome assembly, especially in reference to contig lengths within a draft assembly.

Basecalled reads length for each barcode



Explanation: Basecalled reads length represents histogram plot (histplot) to show the relationship between count as number of reads on y-axis and basecall length as a logarithmic scale on x-axis for each barcode in FASTQ file.

02 Quality Score

Part 1: Make the summary table

By Bowornpol

Python code:

```
def scSum(csv):
    sum = pd.read_csv(csv)

    #Summary table
    grouptable = sum.groupby('Barcode_arrangement')
    qsummarytable = grouptable['Mean_qscore_template'].describe()
    qfigTable = ff.create_table(qsummarytable, index='Barcode_arrangement')
```

Output:

	count	mean	std	min	25%	50%	75%	max
barcode01	99747.0	21.344272649343154	3.3974275230148194	4.846153846153846	19.239613080251083	21.96660341555977	23.824779712025887	55.33276232371588
barcode02	265431.0	21.333555482932464	3.3833523144277233	5.0	19.25505330015229	21.995972382048333	23.814051053147125	60.12236286919831

02 Quality Score

By Bowornpol

Part 2: Make the distribution plot for all reads

Python code:

```
#Basecalled reads PHRED quality (Distribution plot - All reads)
group_label = ['Mean_qscore_template']
qDfig = ff.create_distplot([sum['Mean_qscore_template']], group_label, show_hist = False)
qDfig.update_layout(xaxis_title="Reads quality scores", yaxis_title="Read density")
qDfig.add_vline(x=8.0, line_width=1.5, line_dash="dash", line_color="red")
qDfig.write_image("qDfig.png")
```

Output:

02 Quality Score

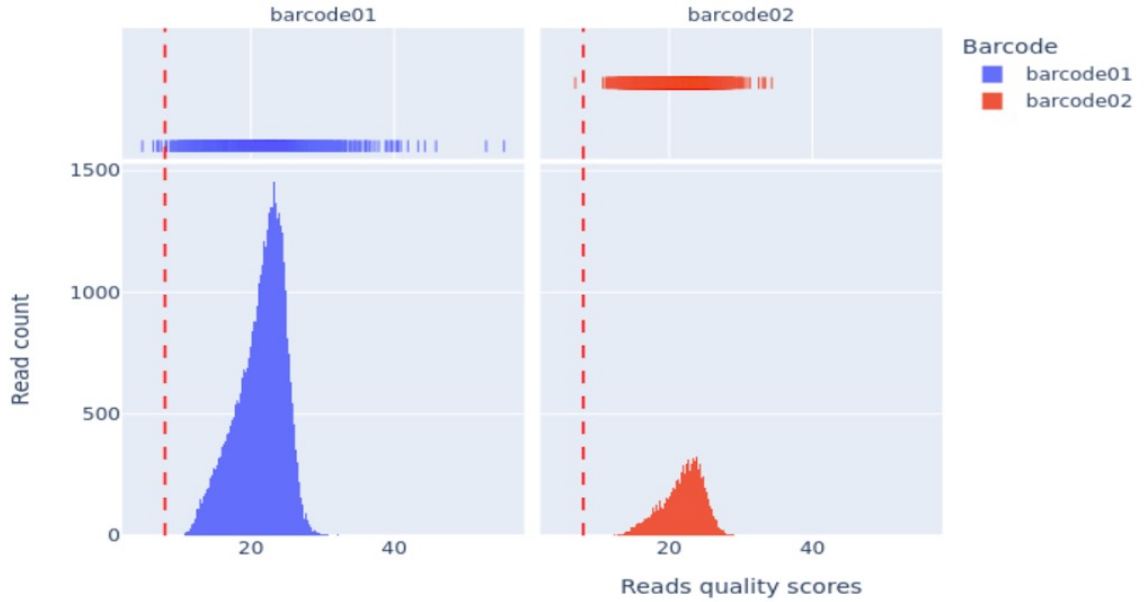
By Bowornpol

Part 3: Make the histogram plot to visualize data in each barcode arrangement

Python code:

```
#Basecalled reads PHRED quality (Histrogram plot - Each of barcode)
qHfig = px.histogram(sum, x=sum['Mean_qscore_template'], color=sum['Barcode_arrangement'], facet_col=sum['Barcode_arrangement'], marginal="rug")
qHfig.update_xaxes(title='')
qHfig.update_layout(xaxis2=dict(title="Reads quality scores"), yaxis_title="Read count", legend_title="Barcode")
qHfig.for_each_annotation(lambda a: a.update(text=a.text.replace("Barcode_arrangement=", "")))
qHfig.add_vline(x=8.0, line_width=1.5, line_dash="dash", line_color="red")
qHfig.write_image("qHfig.png")
```

Output:



02 Quality Score

Part 4: Make a pie chart of the number of reads per quality score

By Bowornpol

Python code:

```
#Number of reads per quality score
PassReads =  sum.query('Mean_qscore_template >= 8')
FailReads =  sum.query('Mean_qscore_template < 8')

qlabels = ['Passed Reads','Failed Reads']
qvalues = [len(PassReads), len(FailReads)]

qPiefig = go.Figure(data=[go.Pie(labels=qlabels, values=qvalues, textinfo='label+percent', insidetextorientation='radial', pull=[0, 0.2])])
qPiefig.update_layout(legend_title="Type of reads")
```

Output:

02 Quality Score

Part 5: Write the content division element for HTML

By Bowornpol

Python code:

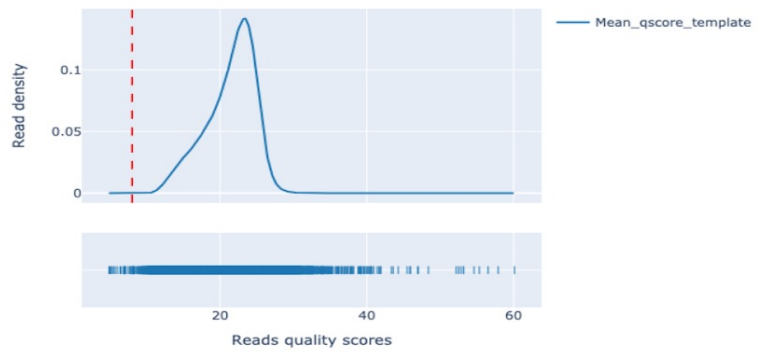
```
#HTML
qfigTable_html = '<div><h2>Quality score summary</h2>'+qfigTable.to_html(full_html=False, include_plotlyjs='cdn')+'<p><strong>Explanation: </strong>The quality score
qfig_html = '<div><h2>Basecalled reads PHRED quality</h2><p style="text-align:center;"><img src="qDfig.png" width="1000" height="600"></p><p style="color:Tomato;">Red
qPiefig_html = '<div><h2>Number of reads per quality score</h2>'+qPiefig.to_html(full_html=False, include_plotlyjs='cdn')+'<p><strong>Explanation: </strong>Number of

results = qfigTable_html+qfig_html+qPiefig_html

return results
```

Output:

Basecalled reads PHRED quality



Red line: Cut-off line suggestion (Mean quality score at 8.0)

Explanation: Basecalled reads PHRED quality distplot represents the distribution of mean quality score for all reads.

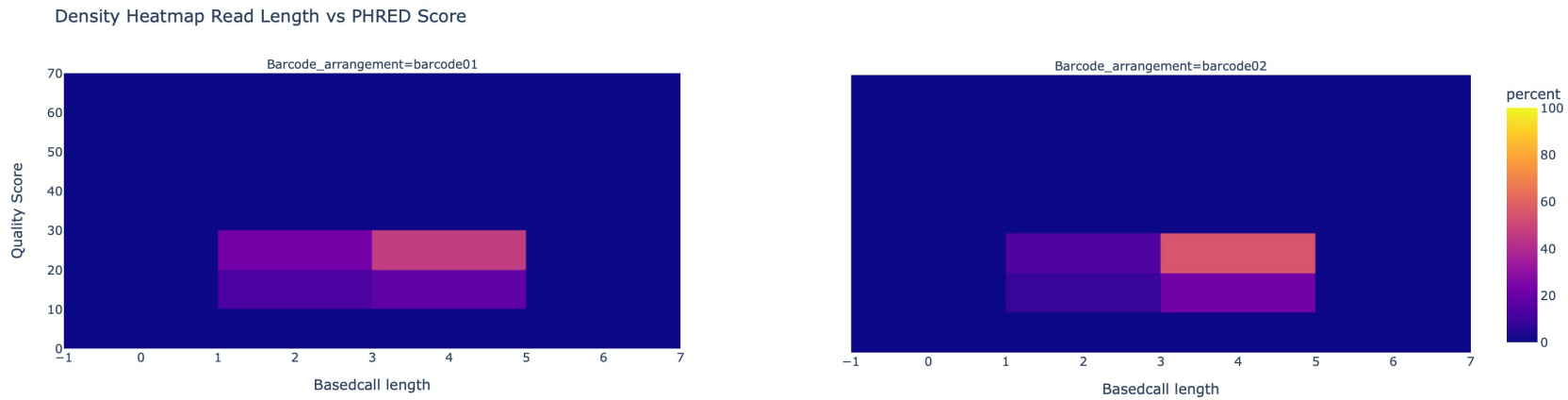
03 Quality vs. Read length

By Naphat

Heatmap

```
heatlog=px.density_heatmap(data_frame=c,x=c.loc[:,'log_Sequence_length_templa
te'],y=c.loc[:,'Mean_qscore_template'],facet_row='Barcode_arrangement',nbinsx
=5,nbinsy=10,title='Density Heatmap Read Length vs PHRED Score')
```

- Heatmap are generated by Plotly Express heatmap
- 5 bin in x-axis (length)
- 10 bin in y-axis (score)
- Yellow = high density
- Purple = low density



03 Quality vs. Read length

By Naphat

Scatter plot

```
scatterlog=px.scatter(data_frame=c,x=c.loc[:,'log_Sequence_length_template'
],y=c.loc[:,'Mean_qscore_template'],facet_row='Barcode_arrangement',opacity
=0.2,title='Scatter plot Read Length vs Quality Score')
```

- Scatter plot generated by plotly express scatter
- point opacity = 0.2 to show when point overlap