In [1]:
```
import pandas as pd
import numpy as np
import os
import sys
```

In [2]:
```
from bokeh.io import show, output_notebook, push_notebook
from bokeh.plotting import figure

from bokeh.models import CategoricalColorMapper, HoverTool, ColumnDataSource, Pa
from bokeh.models.widgets import CheckboxGroup, Slider, RangeSlider, Tabs

from bokeh.layouts import column, row, WidgetBox
from bokeh.palettes import Category20_16

from bokeh.application.handlers import FunctionHandler
from bokeh.application import Application

output_notebook()
```

(https://bokeh.pydata.org)

In [3]:
```
df = pd.read_csv('FIFA_1112.csv', index_col=0)
df.head()
```

Out[3]:

	Name	Club	Country	Continent	League	Overall Rating	Position	Position Group	Skill	Weak Foot
0	Lionel Messi	FC Barcelona	Argentina	SA	LaLiga Santander	94	RW	Attacker	4	4
1	Cristiano Ronaldo	Piemonte Calcio	Portugal	EU	Serie A TIM	93	ST	Attacker	5	4
2	Neymar Jr	Paris Saint-Germain	Brazil	SA	Ligue 1 Conforama	92	LW	Attacker	5	5
3	Kevin De Bruyne	Manchester City	Belgium	EU	Premier League	91	CAM	Midfieder	4	5
4	Eden Hazard	Real Madrid	Belgium	EU	LaLiga Santander	91	LW	Attacker	4	4

In [4]:
```
df['Overall Rating'].describe()
```

Out[4]:
```
count    16359.000000
mean        66.606639
std          6.865687
min         48.000000
25%         62.000000
50%         66.000000
75%         71.000000
max         94.000000
Name: Overall Rating, dtype: float64
```

Histrogram- Data for plotting

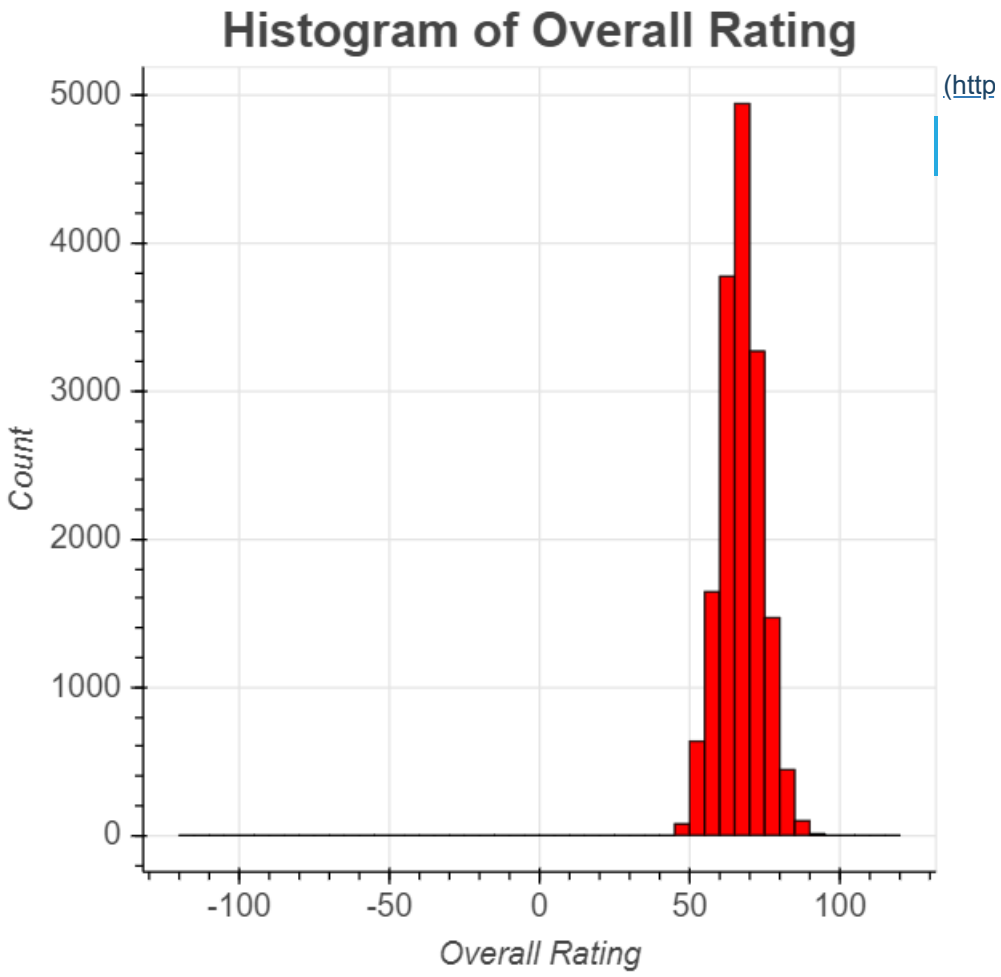
In [5]:
```
# Bins will be five unit in width
arr_hist, edges = np.histogram(df['Overall Rating'], bins = int(240/5), range =
```

In [6]:
```
# Set up the figure
p = figure(plot_width = 500, plot_height = 500, title = 'Histogram of Overall Ra
          x_axis_label = 'Overall Rating', y_axis_label = 'Count')

# Add a quad glyph
p.quad(bottom=0, top=arr_hist, left=edges[:-1], right=edges[1:], fill_color='red

# To show in notebook
output_notebook()

# Show the plot
show(p)
```

(https://bokeh.pydata.org)BokehJS 1.3.4 successfully loaded.

In [7]:
```
def style(p):
    p.title.align = 'center'
    p.title.text_font_size = '18pt'
    p.xaxis.axis_label_text_font_size = '12pt'
    p.xaxis.major_label_text_font_size = '12pt'
    p.yaxis.axis_label_text_font_size = '12pt'
    p.yaxis.major_label_text_font_size = '12pt'

    return p
```

In [8]:
```
styled_p = style(p)
show(styled_p)
```



In [9]:
```
arr_df = pd.DataFrame({'count': arr_hist, 'left': edges[:-1], 'right': edges[1:]}
arr_df['f_count'] = ['%d ' % count for count in arr_df['count']]
arr_df['f_interval'] = ['%d to %d ' % (left, right) for left, right in zip(arr_d
arr_df.head()
```

Out[9]:

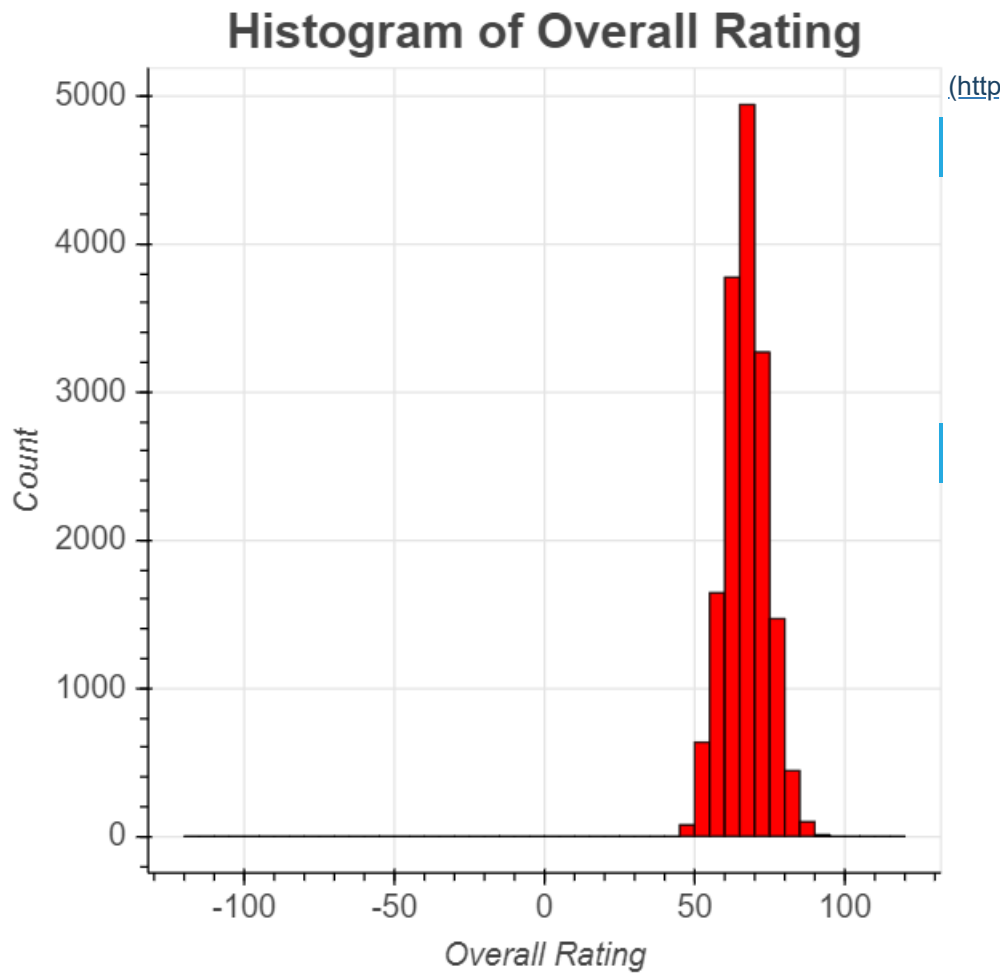
	count	left	right	f_count	f_interval
0	0	-120.0	-115.0	0	-120 to -115
1	0	-115.0	-110.0	0	-115 to -110
2	0	-110.0	-105.0	0	-110 to -105
3	0	-105.0	-100.0	0	-105 to -100
4	0	-100.0	-95.0	0	-100 to -95

In [10]:
```
arr_src = ColumnDataSource(arr_df)
```

In [11]:
```
arr_src.data.keys()
```

Out[11]:
```
dict_keys(['index', 'count', 'left', 'right', 'f_count', 'f_interval'])
```

In [12]:
```
# Set up the figure same as before
p = figure(plot_width = 500, plot_height = 500, title = 'Histogram of Overall Ra
          x_axis_label = 'Overall Rating', y_axis_label = 'Count')

# Add a quad glyph with source this time
p.quad(bottom=0, top='count', left='left', right='right', source=arr_src,
       fill_color='red', line_color='black')

# Add style to the plot
styled_p = style(p)

# Add a hover tool referring to the formatted columns
hover = HoverTool(tooltips = [('Interval', '@f_interval'),
                              ('Count', '@f_count')])

# Add the hover tool to the graph
styled_p.add_tools(hover)

# Show the plot
show(styled_p)
```



(http

In [13]:
```
arr_df.head()
```

Out[13]:

	count	left	right	f_count	f_interval
0	0	-120.0	-115.0	0	-120 to -115
1	0	-115.0	-110.0	0	-115 to -110
2	0	-110.0	-105.0	0	-110 to -105
3	0	-105.0	-100.0	0	-105 to -100
4	0	-100.0	-95.0	0	-100 to -95

In [14]:
```
arr_df.head()
```

Out[14]:

	count	left	right	f_count	f_interval
0	0	-120.0	-115.0	0	-120 to -115
1	0	-115.0	-110.0	0	-115 to -110
2	0	-110.0	-105.0	0	-110 to -105
3	0	-105.0	-100.0	0	-105 to -100
4	0	-100.0	-95.0	0	-100 to -95

In [15]:
```
# Available carrier list
available_carriers = list(df['Country'].unique())

# Sort the list in-place (alphabetical order)
available_carriers.sort()
```

In [16]:
```
# Available carrier list
available_carriers = list(df['Country'].unique())

# Sort the list in-place (alphabetical order)
available_carriers.sort()
```

In [17]:
```
df['Country']
```

Out[17]:
```
0        Argentina
1         Portugal
2           Brazil
3          Belgium
4          Belgium
            ...
16744     China PR
16745      England
16746      England
16747      England
16748       Cyprus
Name: Country, Length: 16359, dtype: object
```

In [18]:
```
from bokeh.io import output_file, show
from bokeh.layouts import widgetbox
from bokeh.models.widgets import Dropdown
from bokeh.plotting import curdoc

#output_file("dropdown.html")

def modify_doc(doc):

    def make_dataset(carrier_list, range_start = 50, range_end = 110, bin_width =

        by_carrier = pd.DataFrame(columns=['proportion', 'left', 'right',
                                           'f_proportion', 'f_interval',
                                           'Country', 'color'])
        range_extent = range_end - range_start

        # Iterate through all the carriers
        for i, carrier_name in enumerate(carrier_list):

            # Subset to the carrier
            subset = df[df['Country'] == carrier_name]

            # Create a histogram with 5 minute bins
            arr_hist, edges = np.histogram(subset['Overall Rating'],
                                           bins = int(range_extent / bin_width),
                                           range = [range_start, range_end])

            # Divide the counts by the total to get a proportion
            arr_df = pd.DataFrame({'proportion': arr_hist, 'left': edges[:-1], '

            # Format the proportion
            arr_df['f_proportion'] = ['%0.5f' % proportion for proportion in arr_

            # Format the interval
            arr_df['f_interval'] = ['%d to %d' % (left, right) for left, right in

            # Assign the carrier for labels
            arr_df['Country'] = carrier_name

            # Color each carrier differently
            arr_df['color'] = Category20_16[i]

            # Add to the overall dataframe
            by_carrier = by_carrier.append(arr_df)

        # Overall dataframe
        by_carrier = by_carrier.sort_values(['Country', 'left'])

        return ColumnDataSource(by_carrier)

    def style(p):
        # Title
        p.title.align = 'center'
        p.title.text_font_size = '20pt'
        p.title.text_font = 'serif'
```

```
    # Axis titles
    p.xaxis.axis_label_text_font_size = '14pt'
    p.xaxis.axis_label_text_font_style = 'bold'
    p.yaxis.axis_label_text_font_size = '14pt'
    p.yaxis.axis_label_text_font_style = 'bold'

    # Tick labels
    p.xaxis.major_label_text_font_size = '12pt'
    p.yaxis.major_label_text_font_size = '12pt'

    return p

def make_plot(src):
    # Blank plot with correct labels
    p = figure(plot_width = 700, plot_height = 700,
              title = 'Histogram of Overall Rating',
              x_axis_label = 'Overall Rating', y_axis_label = 'Counts')

    # Quad glyphs to create a histogram
    p.quad(source = src, bottom = 0, top = 'proportion', left = 'left', right
           color = 'color', fill_alpha = 0.7, hover_fill_color = 'color', leg
           hover_fill_alpha = 1.0, line_color = 'black')

    # Hover tool with vline mode
    hover = HoverTool(tooltips=[('Country', '@Country'),
                                ('Interval', '@f_interval'),
                                ('Count', '@f_proportion')],
                      mode='vline')

    p.add_tools(hover)

    # Styling
    p = style(p)

    return p

def update(attr, old, new):
    carriers_to_plot = [carrier_selection.labels[i] for i in carrier_selectio

    new_src = make_dataset(carriers_to_plot,
                           range_start = range_select.value[0],
                           range_end = range_select.value[1],
                           bin_width = binwidth_select.value)

    src.data.update(new_src.data)

def function_to_call(attr, old, new):
    dropdown.on_change('value', function_to_call)
    dropdown.on_click(function_to_call)
    show(widgetbox(dropdown))


menu = list(df['Country'].unique())
dropdown = Dropdown(label="Dropdown button", button_type="warning", menu=menu
```

```
    menu = list(df['Country'].unique())
    dropdown = Dropdown(label="Dropdown button", button_type="warning", menu=menu

    carrier_selection = CheckboxGroup(labels=available_carriers, active = [0, 1]
    carrier_selection.on_change('active', update)

    binwidth_select = Slider(start = 1, end = 30,
                             step = 1, value = 5,
                             title = 'Width ')
    binwidth_select.on_change('value', update)

    range_select = RangeSlider(start = -60, end = 180, value = (40, 100),
                               step = 5, title = 'Overall rating range for Posit
    range_select.on_change('value', update)



    initial_carriers = [carrier_selection.labels[i] for i in carrier_selection.a

    src = make_dataset(initial_carriers,
                       range_start = range_select.value[0],
                       range_end = range_select.value[1],
                       bin_width = binwidth_select.value)

    p = make_plot(src)

#     #Put dropdown menu

#     menu = list(df['Country'].unique())
#     dropdown = Dropdown(label="Dropdown button", button_type="warning", menu=me

#     #show(dropdown)


    # Put controls in a single element
    controls = WidgetBox(dropdown, carrier_selection, binwidth_select, range_sel

    # Create a row layout
    layout = row(controls, p)

    # Make a tab with the layout
    tab = Panel(child=layout, title = 'OverallRating Histogram for Postion = GK'
    tabs = Tabs(tabs=[tab])

    doc.add_root(tabs)


# Set up an application
handler = FunctionHandler(modify_doc)
app = Application(handler)
```

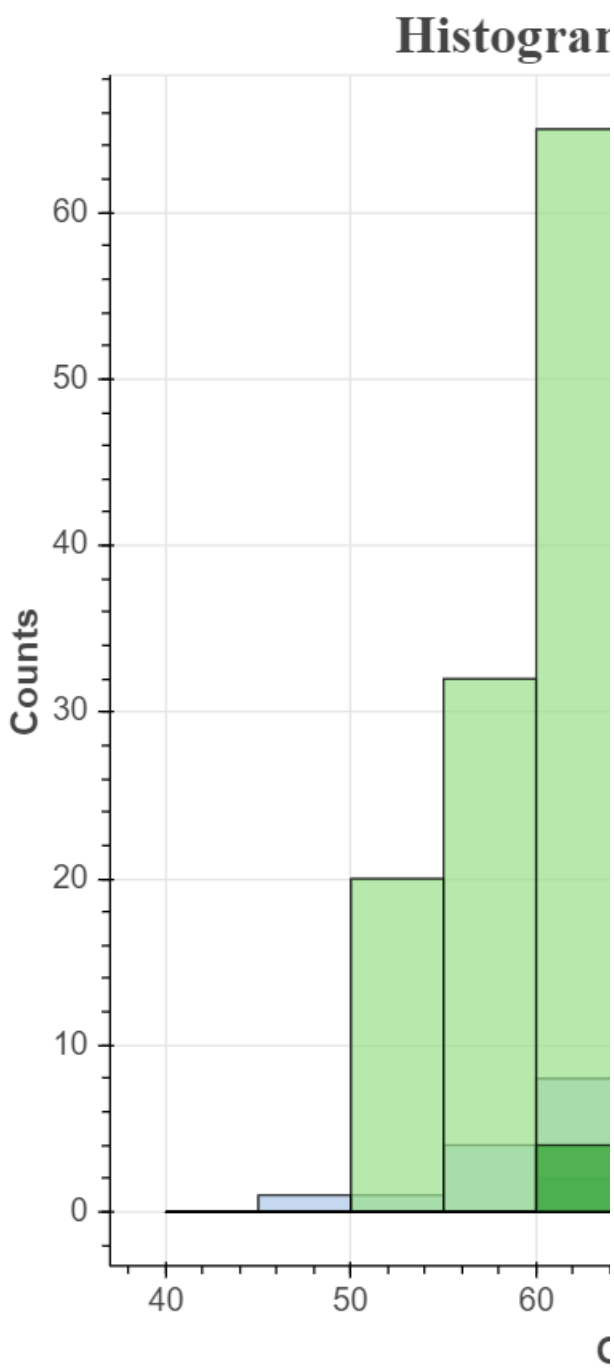
In [19]:
```
doc = app.create_document()
```

In [22]:
```
show(app, notebook_url="localhost:8872")
```

OverallRating Histogram for Postion = GK

Dropdown button

- [x] Afghanistan
- [x] Albania
- [ ] Algeria
- [ ] Angola
- [ ] Antigua and Barbuda
- [ ] Argentina
- [ ] Armenia
- [x] Australia
- [ ] Austria
- [ ] Azerbaijan
- [x] Bahrain
- [ ] Belgium
- [x] Belize
- [x] Benin
- [ ] Bermuda
- [ ] Bolivia
- [ ] Bosnia and Herzegovina
- [ ] Brazil
- [ ] Bulgaria
- [ ] Burkina Faso
- [ ] Burundi
- [ ] Cameroon
- [ ] Canada
- [ ] Cape Verde Islands
- [ ] Central African Republic
- [ ] Chad
- [ ] Chile
- [ ] China PR
- [ ] Chinese Taipei
- [ ] Colombia
- [ ] Comoros
- [ ] Congo
- [ ] Congo DR
- [ ] Costa Rica
- [ ] Croatia
- [ ] Cuba
- [ ] Curaçao
- [ ] Cyprus
- [ ] Czech Republic
- [ ] Côte d'Ivoire
- [ ] Denmark
- [ ] Dominican Republic
- [ ] Ecuador
- [ ] Egypt
- [ ] El Salvador



- [ ] England
- [ ] Equatorial Guinea
- [ ] Eritrea
- [ ] Estonia
- [ ] Ethiopia
- [ ] FYR Macedonia
- [ ] Faroe Islands
- [ ] Finland
- [ ] France
- [ ] Gabon
- [ ] Gambia
- [ ] Georgia
- [ ] Germany
- [ ] Ghana
- [ ] Gibraltar
- [ ] Greece
- [ ] Grenada
- [ ] Guam
- [ ] Guatemala
- [ ] Guinea
- [ ] Guinea-Bissau
- [ ] Guyana
- [ ] Haiti
- [ ] Holland
- [ ] Honduras
- [ ] Hong Kong
- [ ] Hungary
- [ ] Iceland
- [ ] Indonesia
- [ ] Iran
- [ ] Iraq
- [ ] Israel
- [ ] Italy
- [ ] Jamaica
- [ ] Japan
- [ ] Jordan
- [ ] Kazakhstan
- [ ] Kenya
- [ ] Korea DPR
- [ ] Korea Republic
- [ ] Kosovo
- [ ] Latvia
- [ ] Lebanon
- [ ] Liberia
- [ ] Libya
- [ ] Liechtenstein
- [ ] Lithuania
- [ ] Luxembourg
- [ ] Madagascar
- [ ] Malawi
- [ ] Mali
- [ ] Malta

- [ ] Mauritania
- [ ] Mauritius
- [ ] Mexico
- [ ] Moldova
- [ ] Montenegro
- [ ] Montserrat
- [ ] Morocco
- [ ] Mozambique
- [ ] Namibia
- [ ] New Caledonia
- [ ] New Zealand
- [ ] Niger
- [ ] Nigeria
- [ ] Northern Ireland
- [ ] Norway
- [ ] Palestine
- [ ] Panama
- [ ] Paraguay
- [ ] Peru
- [ ] Philippines
- [ ] Poland
- [ ] Portugal
- [ ] Puerto Rico
- [ ] Republic of Ireland
- [ ] Romania
- [ ] Russia
- [ ] Rwanda
- [ ] Saudi Arabia
- [ ] Scotland
- [ ] Senegal
- [ ] Serbia
- [ ] Sierra Leone
- [ ] Slovakia
- [ ] Slovenia
- [ ] South Africa
- [ ] South Sudan
- [ ] Spain
- [ ] St. Kitts and Nevis
- [ ] St. Lucia
- [ ] Sudan
- [ ] Suriname
- [ ] Sweden
- [ ] Switzerland
- [ ] Syria
- [ ] São Tomé e Príncipe
- [ ] Tanzania
- [ ] Thailand
- [ ] Togo
- [ ] Trinidad and Tobago
- [ ] Tunisia
- [ ] Turkey
- [ ] Uganda

Uganda
Ukraine
United Arab Emirates
United States
Uruguay
Uzbekistan
Venezuela
Vietnam
Wales
Zambia
Zimbabwe

Width : **5**

Overall rating range for Position = GK: **40 .. 100**

In [21]:
```
show(app, notebook_url="localhost:8888")
```

```
ERROR:bokeh.server.views.ws:Refusing websocket connection from Origin 'http://l
ocalhost:8889';                       use --allow-websocket-origin=localhost:88
89 or set BOKEH_ALLOW_WS_ORIGIN=localhost:8889 to permit this; currently we all
ow origins {'localhost:8888'}
WARNING:tornado.access:403 GET /ws?bokeh-protocol-version=1.0&bokeh-session-id=
bdiuk6utfNrDJ1nbKyiyJ0mCWTCjEeynLF1Nu52DE6cT (::1) 1.99ms
```

In [47]:
```
df.head()
dfnew=df[df['Position'] == 'GK']
```

In [172]:

```
def modify_doc(doc):

    def make_dataset(carrier_list, range_start = 50, range_end = 110, bin_width =

        by_carrier = pd.DataFrame(columns=['proportion', 'left', 'right',
                                           'f_proportion', 'f_interval',
                                           'Country', 'color'])
        range_extent = range_end - range_start

        # Iterate through all the carriers
        for i, carrier_name in enumerate(carrier_list):

            # Subset to the carrier
            subset = dfnew[dfnew['Country'] == carrier_name]

            # Create a histogram with 5 minute bins
            arr_hist, edges = np.histogram(subset['Overall Rating'],
                                           bins = int(range_extent / bin_width),
                                           range = [range_start, range_end])

            # Divide the counts by the total to get a proportion
            arr_df = pd.DataFrame({'proportion': arr_hist, 'left': edges[:-1], '

            # Format the proportion
            arr_df['f_proportion'] = ['%0.5f' % proportion for proportion in arr_

            # Format the interval
            arr_df['f_interval'] = ['%d to %d' % (left, right) for left, right in

            # Assign the carrier for labels
            arr_df['Country'] = carrier_name

            # Color each carrier differently
            arr_df['color'] = Category20_16[i]

            # Add to the overall dataframe
            by_carrier = by_carrier.append(arr_df)

        # Overall dataframe
        by_carrier = by_carrier.sort_values(['Country', 'left'])

        return ColumnDataSource(by_carrier)

    def style(p):
        # Title
        p.title.align = 'center'
        p.title.text_font_size = '20pt'
        p.title.text_font = 'serif'

        # Axis titles
        p.xaxis.axis_label_text_font_size = '14pt'
        p.xaxis.axis_label_text_font_style = 'bold'
        p.yaxis.axis_label_text_font_size = '14pt'
        p.yaxis.axis_label_text_font_style = 'bold'
```

```
    # Tick labels
    p.xaxis.major_label_text_font_size = '12pt'
    p.yaxis.major_label_text_font_size = '12pt'

    return p

def make_plot(src):
    # Blank plot with correct labels
    p = figure(plot_width = 700, plot_height = 700,
               title = 'Histogram of Overall Rating for position = GK',
               x_axis_label = 'Overall Rating', y_axis_label = 'Counts')

    # Quad glyphs to create a histogram
    p.quad(source = src, bottom = 0, top = 'proportion', left = 'left', right
           color = 'color', fill_alpha = 0.7, hover_fill_color = 'color', leg
           hover_fill_alpha = 1.0, line_color = 'black')

    # Hover tool with vline mode
    hover = HoverTool(tooltips=[('Country', '@Country'),
                                ('Interval', '@f_interval'),
                                ('Count', '@f_proportion')],
                      mode='vline')

    p.add_tools(hover)

    # Styling
    p = style(p)

    return p

def update(attr, old, new):
    carriers_to_plot = [carrier_selection.labels[i] for i in carrier_selectio

    new_src = make_dataset(carriers_to_plot,
                           range_start = range_select.value[0],
                           range_end = range_select.value[1],
                           bin_width = binwidth_select.value)

    src.data.update(new_src.data)


carrier_selection = CheckboxGroup(labels=available_carriers, active = [0, 1])
carrier_selection.on_change('active', update)

binwidth_select = Slider(start = 1, end = 30,
                         step = 1, value = 5,
                         title = 'Width ')
binwidth_select.on_change('value', update)

range_select = RangeSlider(start = -60, end = 180, value = (40, 100),
                           step = 5, title = 'Overall rating range for Positi
range_select.on_change('value', update)



initial_carriers = [carrier_selection.labels[i] for i in carrier_selection.ac
```

```
    src = make_dataset(initial_carriers,
                       range_start = range_select.value[0],
                       range_end = range_select.value[1],
                       bin_width = binwidth_select.value)

    p = make_plot(src)

    # Put controls in a single element
    controls = WidgetBox(carrier_selection, binwidth_select, range_select)

    # Create a row layout
    layout = row(controls, p)

    # Make a tab with the layout
    tab = Panel(child=layout, title = 'OverallRating Histogram for Postion = GK')
    tabs = Tabs(tabs=[tab])

    doc.add_root(tabs)


# Set up an application
handler = FunctionHandler(modify_doc)
app = Application(handler)
```

In [173]:
```
show(app, notebook_url="localhost:8889")
```

In [ ]:
```

```