SI649 W23 Altair Homework #1

Overview

For this assignment we're going to recreate a visualization from a FiveThirtyEight article (https://fivethirtyeight.com/features/competitive-hot-dog-eaters-have-made-america-great-again/), as well as some new and different ones. We'll be teaching you different pieces of Altair over the next few weeks so we'll focus on just a few basic chart times this time:

1. Replicate 1 visualizations from the original article (slightly modified)
2. Implementing 4 new visualizations according to our specifications

Lab Instructions (read the full version on the handout of the previous lab)

- Save, rename, and submit the ipynb file (use your username in the name).
- Complete all the checkpoints, to create the required visualization at each cell
- Run every cell (do Runtime -> Restart and run all to make sure you have a clean working version), print to pdf, submit the pdf file.
- For each visualization, we will ask you to write down a "Grammar of Graphics" plan first (basically a description of what you'll code).
- If you end up stuck, show us your work by including links (URLs) that you have searched for. You'll get partial credit for showing your work in progress.

You may also want to, on your own, go through some additional Altair tutorials:

- UW Course
- Altair tutorial

Resources

- Altair Documentation
- Colab Overview
- Markdown Cheatsheet
- Pandas DataFrame Introduction
- Vega-Lite documentation
- Vega/Vega-Lite editor

In [1]:
```
# imports we will use
import altair as alt
import pandas as pd
import datetime as dt
from altair_saver import save
#from collections import defaultdict
alt.renderers.enable('html') #run this line if you are running jupyter notebook
```

Out[1]:
```
RendererRegistry.enable('html')
```

In [2]:
```
# Load the data we'll need (available on Canvs)
df = pd.read_csv('hotdogs_clean.csv', header=0, index_col=0)
print(df.shape)
df.sample(5)
```

```
(2315, 7)
```

Out[2]:

	Place	Consumed	Name	Contest	Location	Date	Minutes
671	3rd	19	Pedram "In Jaws We Trust" Esmaeelzadeh	Nathan's Famous Hot Dog Eating Contest Qualifi...	Nashville, TN	2015-10-21	10.0
1219	1st	18.5	Larell Marie "The Real Deal" Mele	Nathan's Famous Hot Dog Eating Contest Qualifi...	Atlantic City, NJ	2012-05-19	10.0
1809	2nd	26.5	Brad "The Lunatic" Sciullo	Nathan's Famous Hot Dog Eating Contest qualifier	West Chester, PA	2007-06-28	12.0
1091	3rd	12.5	Andrew Kossuth	Nathan's Famous Hot Dog Eating Contest Qualifi...	Brooklyn, NY	2013-03-23	10.0
2167	?	?	"Hungry" Charles Hardy	Nathan's Famous Hot Dog Eating Contest	Brooklyn, NY	2003-07-04	12.0

In [3]:
```
# Drop rows with question marks (?) for Place or Consumed
df = df[(df['Place'] != '?') & (df['Consumed'] != '?')]
df.shape
```

Out[3]: (2037, 7)

In [4]:
```
# Check the data types
df.dtypes
```

Out[4]:
```
Place        object
Consumed     object
Name         object
Contest      object
Location     object
Date         object
Minutes     float64
dtype: object
```

In [5]:
```
# Now that we've dropped the question marks, convert the Consumed and Date columns to the right data types
df['Consumed'] = df['Consumed'].astype('float')
df['Date'] = pd.to_datetime(df['Date'])
df.dtypes
```

Out[5]:
```
Place                object
Consumed            float64
Name                 object
Contest              object
Location             object
Date         datetime64[ns]
Minutes             float64
dtype: object
```

In [6]:
```
df.sample(5)
```

Out[6]:

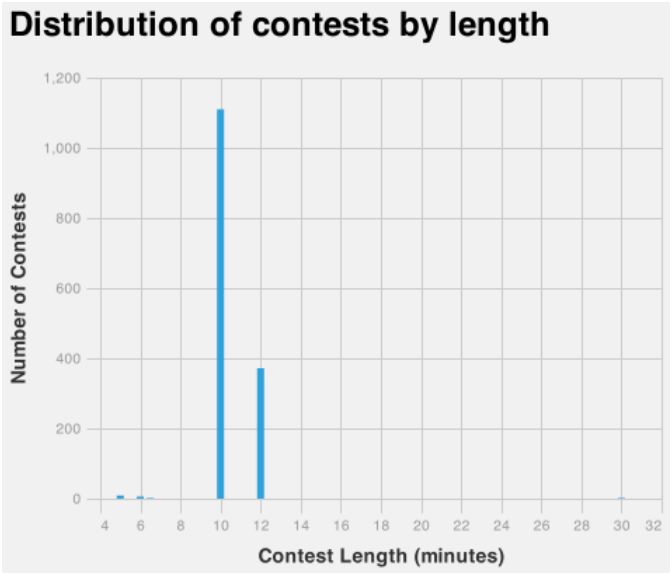
	Place	Consumed	Name	Contest	Location	Date	Minutes
1486	2nd	13.0	Ron Koch	Nathan's Famous Hot Dog Eating Contest qualifier	Las Vegas, NV	2010-05-06	10.0
1851	2nd	15.5	Joe "Big Boss" Tursi	Nathan's Famous Hot Dog Eating Contest qualifier	Norfolk, VA	2007-06-16	12.0
1681	1st	23.0	Allen "Shredder" Goldstein	Nathan's Famous Hot Dog Eating Contest qualifier	Myrtle Beach, SC	2008-06-21	10.0
395	1st	41.0	Miki Sudo	Nathan's Famous Hot Dog Eating Contest - women	Brooklyn, NY	2017-07-04	10.0
1129	3rd	14.0	William "Wild Bill" Myers	Nathan's Famous Hot Dog Eating Contest Qualifi...	Boston, MA	2012-06-23	10.0

In [7]:
```
# Finally, drop the data from 2017 onwards, and the very short and long contests
# This more closely matches the data used by FiveThirtyEight, although there are still some minor differences
df = df[(df['Date'] < dt.datetime(2017, 1, 1)) & (df['Minutes'] >= 5) & (df['Minutes'] < 60)]
df.shape
```

Out[7]: (1501, 7)

Task #1

First, let's examine the distribution of contests by length (in minutes). Recreate the visualization below



In [8]:
```
# Enable the FiveThirtyEight theme
alt.theme.enable('fivethirtyeight')
```

Out[8]:
```
ThemeRegistry.enable('fivethirtyeight')
```

Step 1: Write down your plan for the visualization (edit this cell)

- mark type: bar
- Encoding Specification:
- > x : position : Contest Length (minutes) : quantitative
- > y : position : Number of Contests: quantitative

Example encoding, if we had the nominal variable 'Location' and we wanted to use color, it would be:

color : Location : nominal

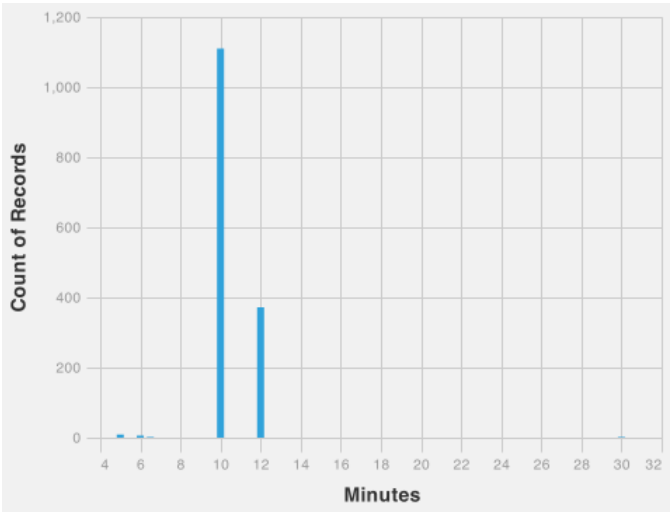
Step 2: Create your chart, step by step

For each task, look at all the checkpoints. You can follow the checkpoint to work through the problem step-by-step. For each checkpoint, you should add code to the cell below it so as to create the required visualizaion. You can search for the keyword "TODO" to locate cells that need your edits

checkpoint 1: basic histogram chart. You will get full point if you:

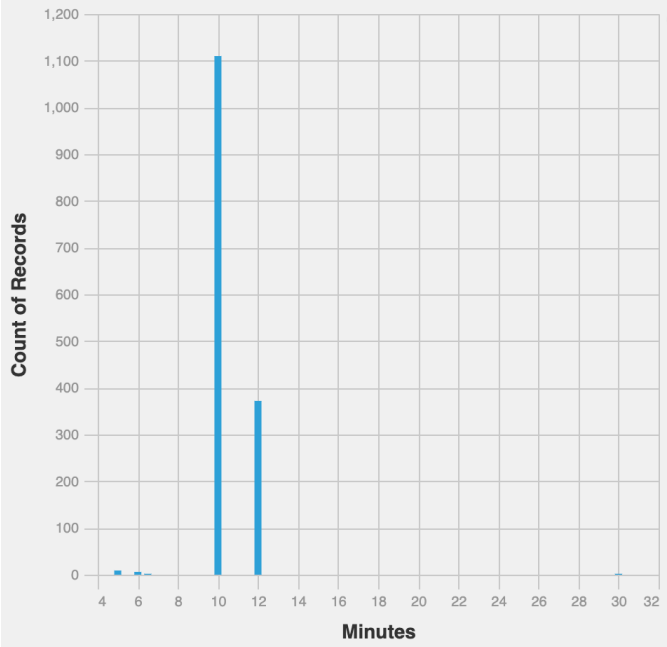
- Plot the right data
- Specify the correct mark
- Use the correct x and y encoding

You chart should look like: (it's okay if the grid lines don't exactly match)



In [9]:
```
#TODO: Replicate task 1, checkpoint 1
# Plot the histogram of the minutes and count of records with altair

# Create the histogram
hist = alt.Chart(df).mark_bar().encode(
    alt.X('Minutes:Q'),
    alt.Y('count():Q')
).properties(
    width = 400,
    height = 400
)

# Show the histogram
hist
```

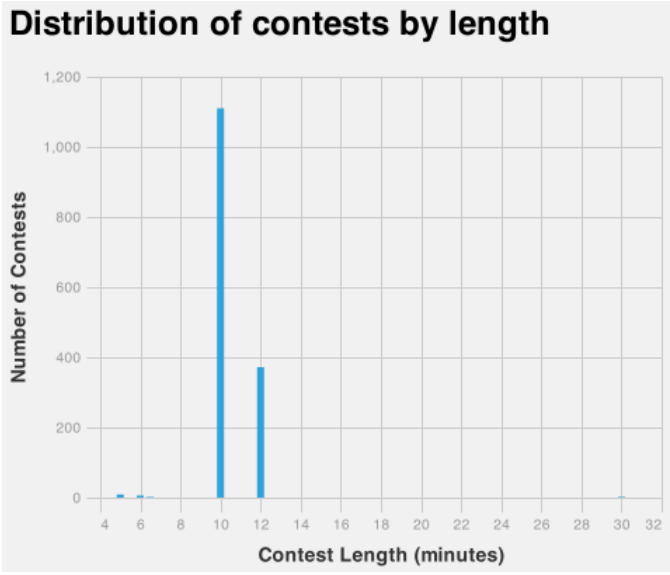
Out[9]:



checkpoint 2: basic bar chart with title and axis labels. You will get full point if you:

- Completed checkpoint 1
- Add the proper labels on x-axis and y-axis
- Add a chart title

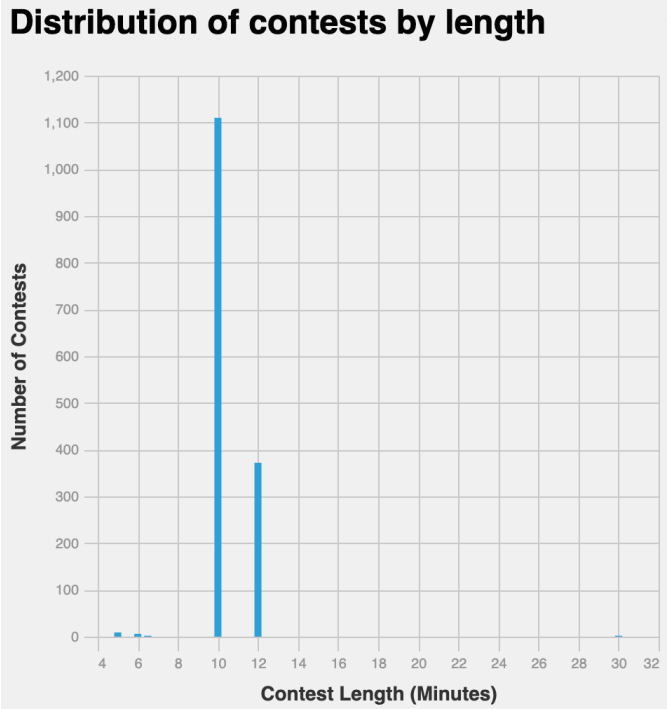
You chart should look like: (it's okay if the grid lines don't exactly match)



In [10]:
```
#TODO: Replicate task 1, checkpoint 2
hist = alt.Chart(df).mark_bar().encode(
    alt.X('Minutes:Q', title="Contest Length (Minutes)"),
    alt.Y('count():Q', title="Number of Contests")
).properties(
    width = 400,
    height = 400,
```

```
    title = "Distribution of contests by length"
)

hist
```

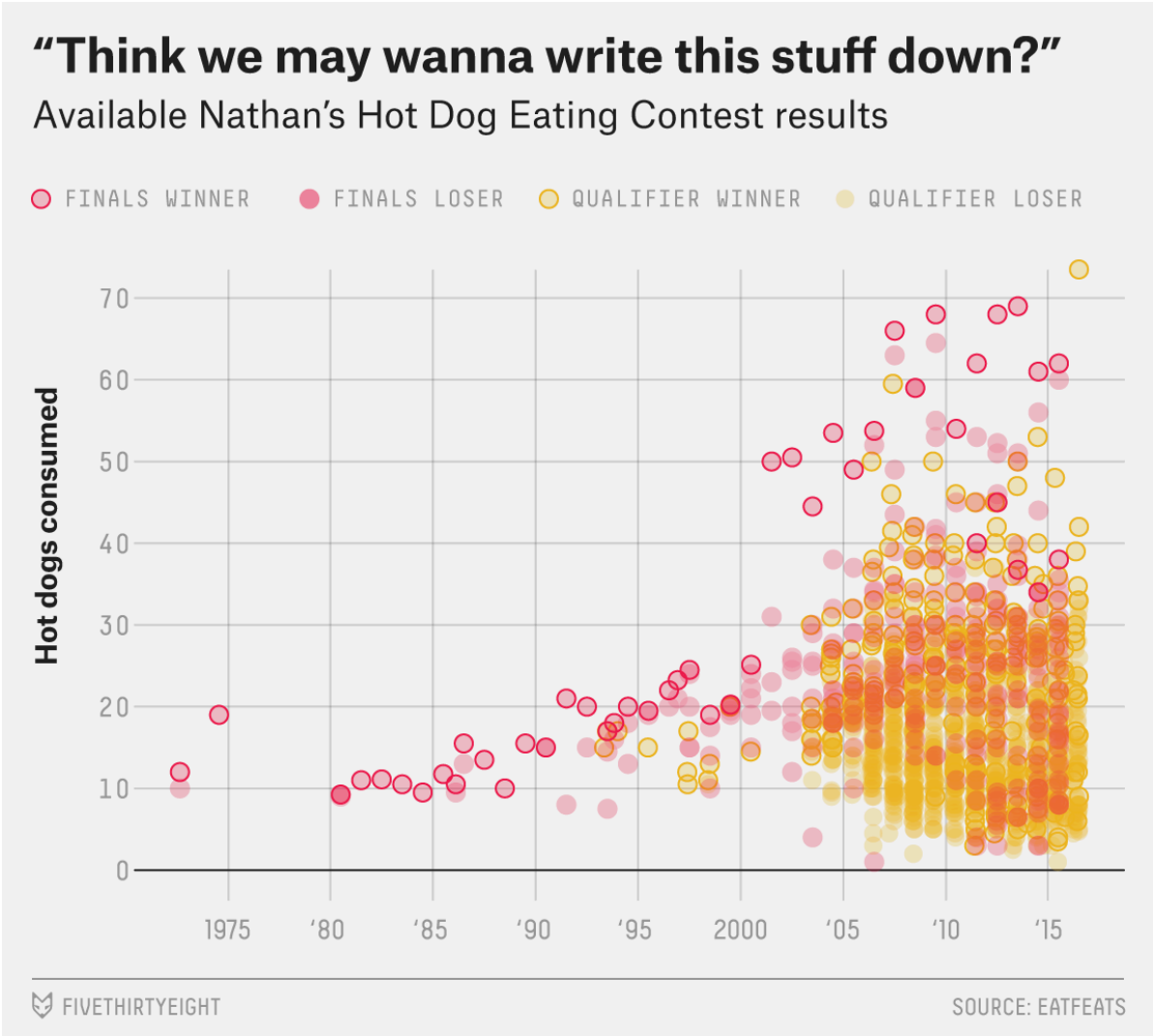
Out[10]:



Task #2

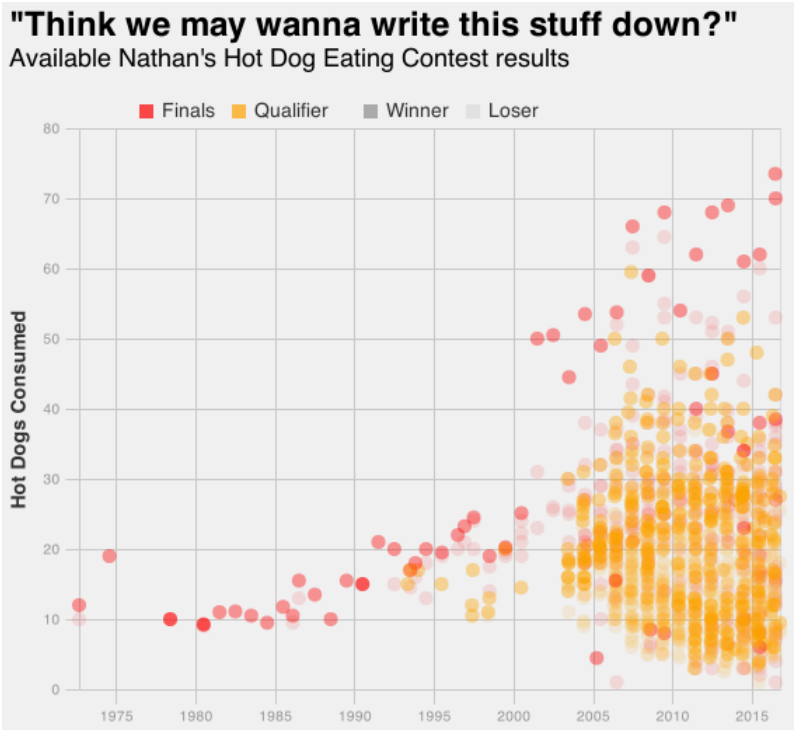
Now, let's recreate a visualization from the FiveThirtyEight article

Here is the original:



We'll learn how to get closer to the original next week (using layering), but for now, we'll make a slightly modified version.

Here is what you should aim to create:



Step 1: Write down your plan for the visualization (edit this cell)

- mark type: circle
- Encoding Specification:
- > x : position: year: Temporal (date transformed to year)
- > y : position: hot dogs consumed: quantitative
- > color : contest type (final, qualifier): nominal (inherently it is ordinal as first qualifier then final, but we treat is as nominal to use different colors here)
- > opacity : player position (winner, loser): ordinal

Step 2: Transform the relevant data using pandas

We need to idenfity which records were for qualifiers, and which were winners

In [11]:
```
# First, let's identify the winners
# Note: there are many ways to do this; this is not compact, but it is fairly easy to understand

winners = []
for val in df['Place']:
    if val == '1st':
        winners.append('Winner')
    else:
        winners.append('Loser')
df['Winners'] = winners
```

In [12]:
```
df.sample(5)
```

Out[12]:

	Place	Consumed	Name	Contest	Location	Date	Minutes	Winners
813	2nd	20.0	Gideon Oji	Nathan's Famous Hot Dog Eating Qualifier - men	Atlanta, GA	2015-04-12	10.0	Loser
1780	2nd	21.0	Ron Koch	Nathan's Famous Hot Dog Eating Contest qualifier	Las Vegas, NV	2008-05-08	10.0	Loser
1749	3rd	16.0	Russ "The Black Hole" Keeler	Nathan's Famous Hot Dog Eating Contest qualifier	Philadelphia, PA	2008-05-24	10.0	Loser
2139	2nd	17.5	Allen "Shredder" Goldstein	Nathan's Famous Hot Dog Eating Contest qualifier	Elmont, NY	2004-05-30	12.0	Loser
824	6th	8.0	Yariel Vignau	Nathan's Famous Hot Dog Eating Qualifier - men	Plant City, FL	2015-03-01	10.0	Loser

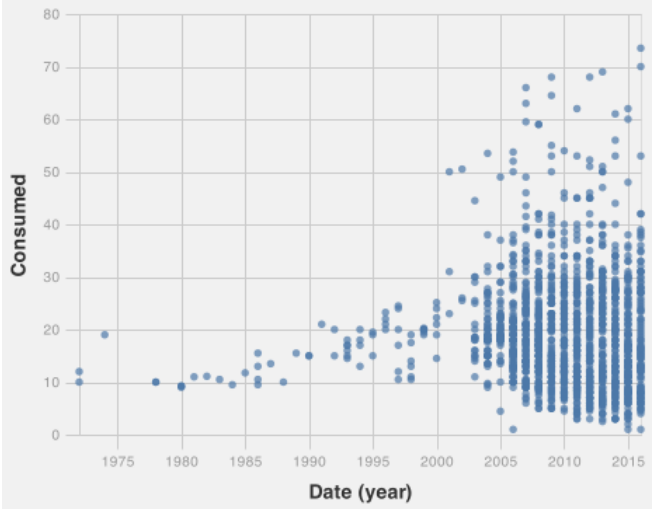
In [13]:
```
# Repeat the above process to identify the records that correspond to qualifiers, and add a column to the dataf

qualifiers = []
for val in df['Contest']:
    # if val includes qualifier/Qualifier
    if 'qualifier' in val.lower():
        qualifiers.append('Qualifier')
    else:
        qualifiers.append('Final')

df['Qualifiers'] = qualifiers

### TODO: fill in the code necessary to identify the qualifiers, based on the above
```

Step 3: Create your chart, step by step

checkpoint 1: basic scatter plot of Year vs Number of Hot Dogs Consumed. You will get full point if you:

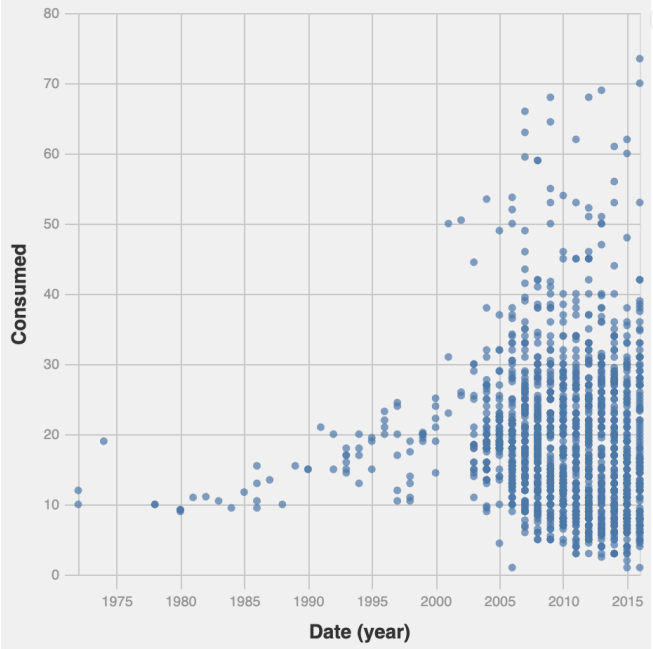
- Plot the right data
- Specify the correct mark
- Use the correct x and y encoding (including converting dates to years)

You chart should look like:



In [14]:
```
#TODO: Replicate task 2, checkpoint 1

# create the scatterplot of year and hotdogs consumed
# Also convert the dates into years
scatter = alt.Chart(df).mark_circle().encode(
    alt.X('year(Date):T'),
    alt.Y('Consumed:Q')
).properties(
    width = 400,
    height = 400
)

scatter
```

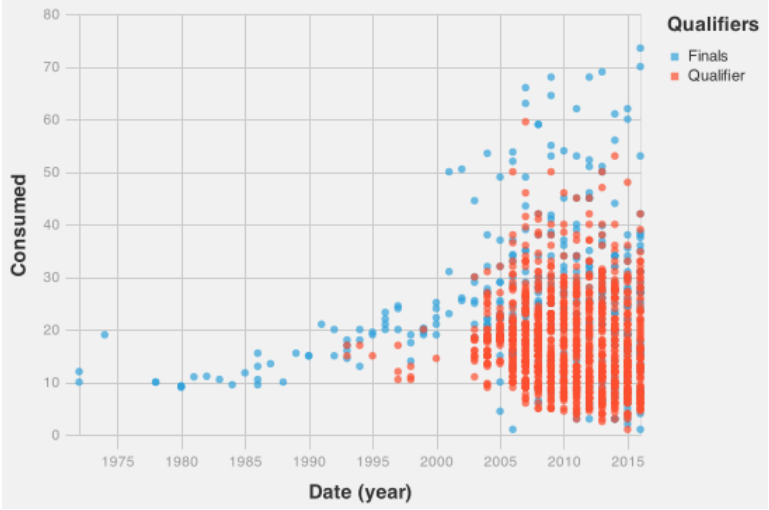
Out[14]:



checkpoint 2: add color to the above scatterplot, corresponding to which records are qualifiers. You will get full point if you:

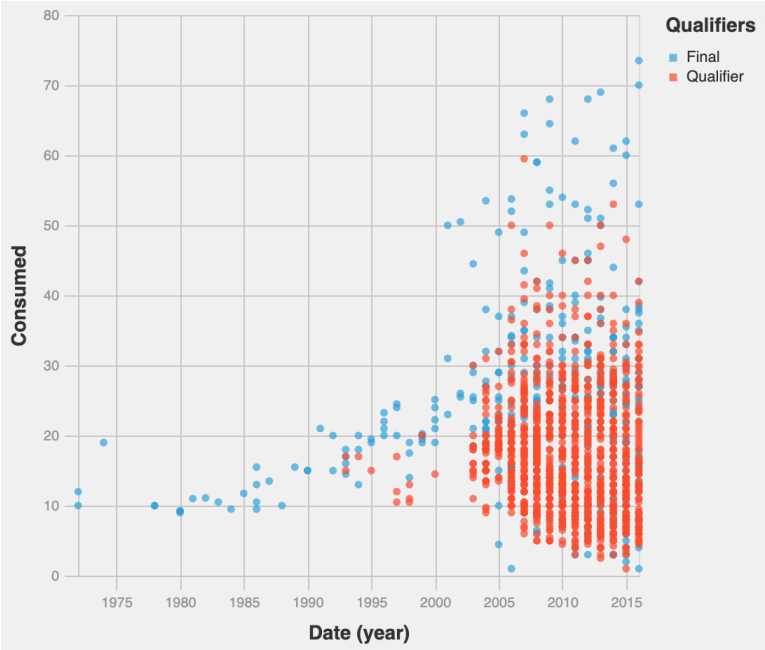
- Completed checkpoint 1
- Add a color channel to distinguish qualifiers from finals

You chart should look like:



In [15]:
```
#TODO: Replicate task 2, checkpoint 2
# Add the color encoding for the qualifiers
scatter = alt.Chart(df).mark_circle().encode(
    alt.X('year(Date):T'),
    alt.Y('Consumed:Q'),
    color='Qualifiers:N'
).properties(
    width = 400,
    height = 400
)

scatter
```

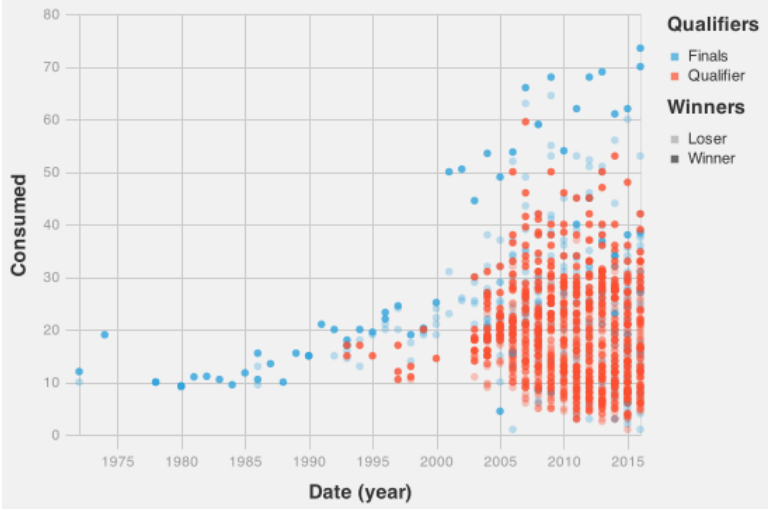
Out[15]:



checkpoint 3: add opacity values corresponding to 1st place vs other. You will get full point if you:

- Completed checkpoint 2
- Add an opacity channel to distinguish 1st place winners vs all other competitors

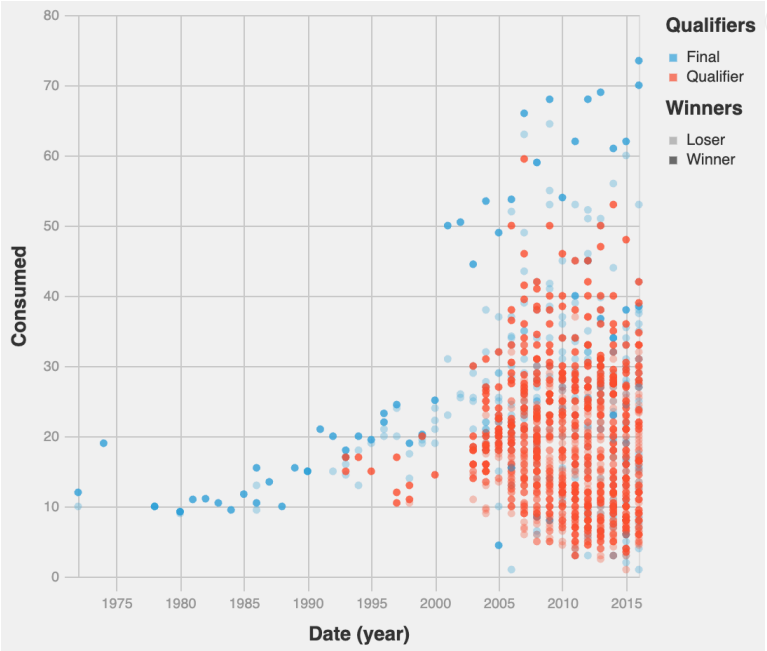
You chart should look like:



In [16]:

```
#TODO: Replicate task 2, checkpoint 3
# Add opacity encoding for the winners
scatter = alt.Chart(df).mark_circle().encode(
    alt.X('year(Date):T'),
    alt.Y('Consumed:Q'),
    color='Qualifiers:N',
    opacity='Winners:O'
).properties(
    width = 400,
    height = 400
)

scatter
```

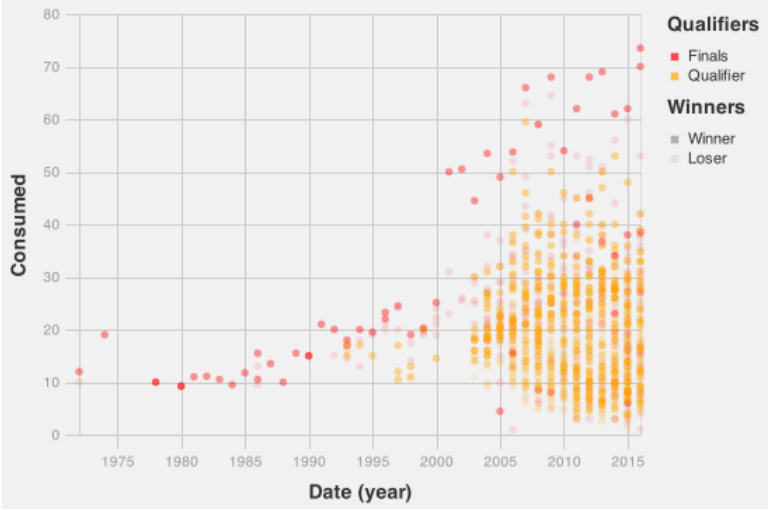
Out[16]:



checkpoint 4: adjust the colors and opacity levels to match the plot specification. You will get full point if you:

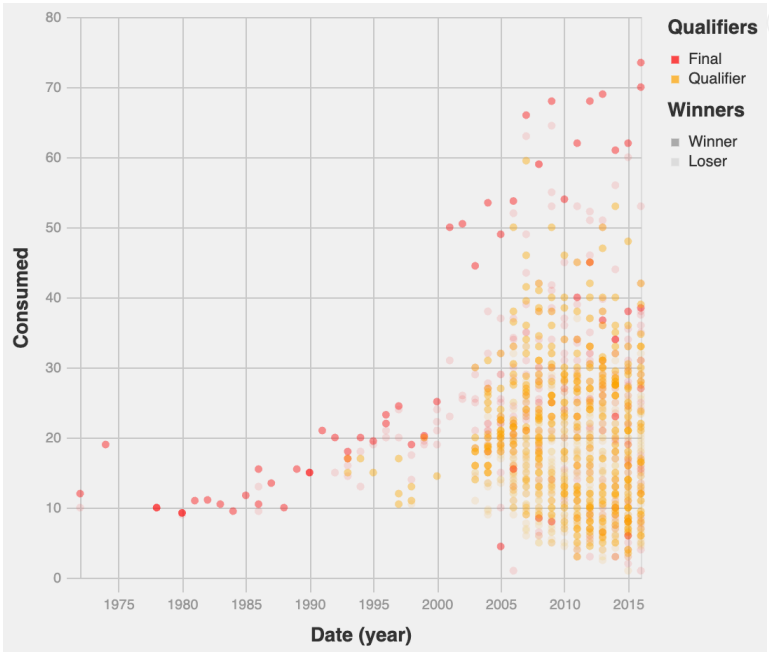
- Completed checkpoint 3
- Change the colors to be red and orange
- Change the opacity levels to be specific values

You chart should look like:



In [17]:
```
#TODO: Replicate task 2, checkpoint 4

# Hint: you can set the target colors and opacity levels using scale=alt.Scale() as an argument of alt.Color()
# good colors for the plot are "red" and "orange"
# good opacity levels are 0.4 and 0.1

scatter = alt.Chart(df).mark_circle().encode(
    alt.X('year(Date):T'),
    alt.Y('Consumed:Q'),
    color=alt.Color('Qualifiers:N', scale=alt.Scale(domain=['Final', 'Qualifier'], range=['red', 'orange'])),
    opacity=alt.Opacity('Winners:O', scale=alt.Scale(domain=['Winner', 'Loser'], range=[0.4, 0.1]))
).properties(
    width = 400,
    height = 400
)

scatter
```

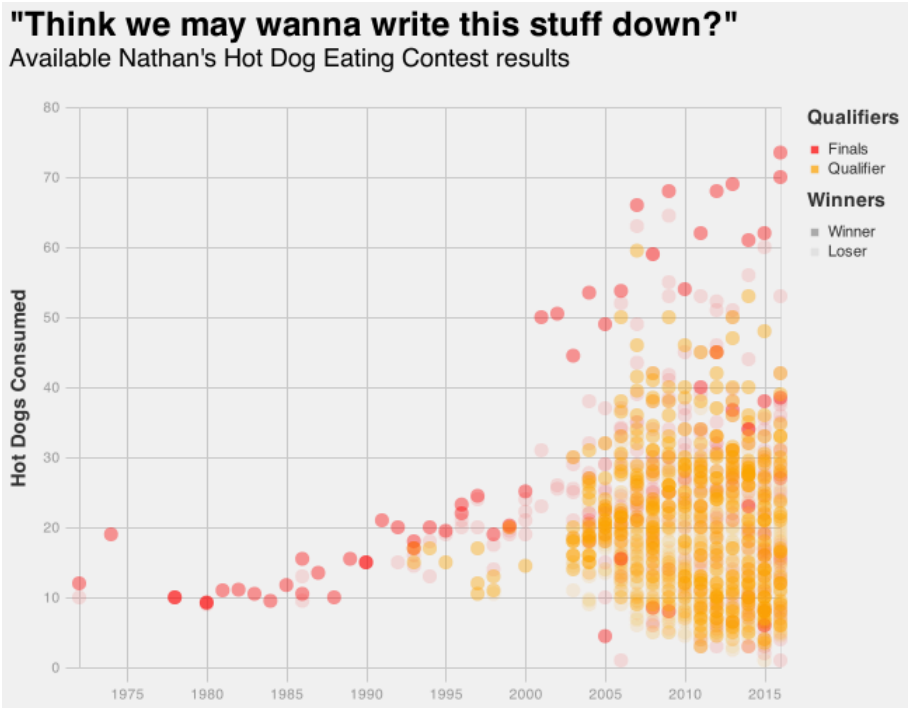
Out[17]:



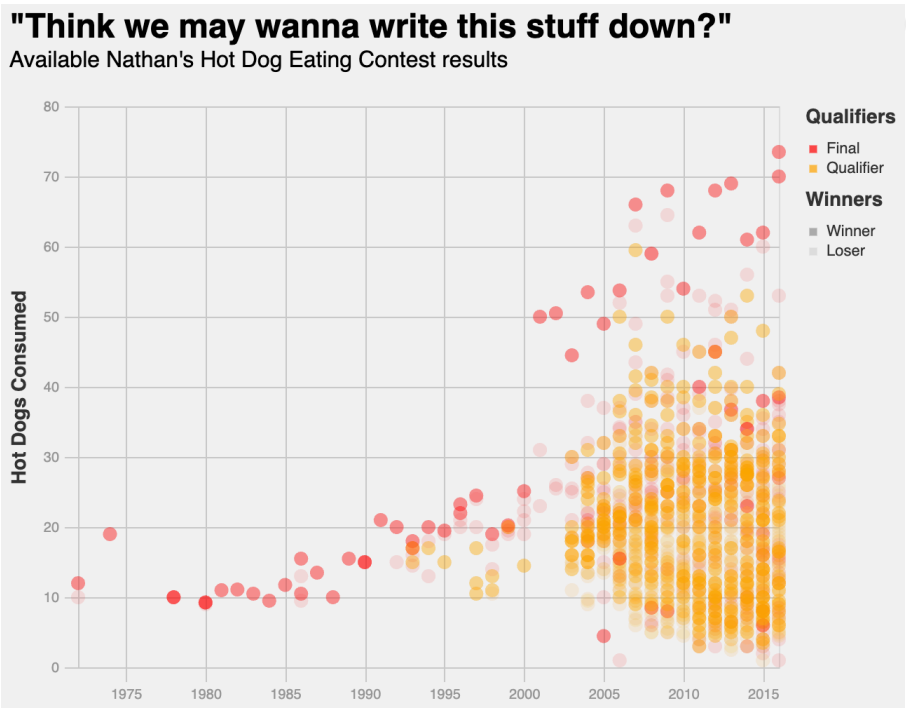
checkpoint 5: add labels and title; adjust plot size; increase point size. You will get full point if you:

- Completed checkpoint 4
- Increase the mark size to 100
- Remove the x-axis label
- Change the y-axis label
- Add a chart title (and subtitle if you can)
- Change the plot dimensions to 500 x 400

You chart should look like:



In [18]:
```
#TODO: Replicate task 2, checkpoint 5

scatter = alt.Chart(df).mark_circle(size=100).encode(
    alt.X('year(Date):T', axis=alt.Axis(title=None)),
    alt.Y('Consumed:Q', title="Hot Dogs Consumed"),
    color=alt.Color('Qualifiers:N', scale=alt.Scale(domain=['Final', 'Qualifier'], range=['red', 'orange'])),
    opacity=alt.Opacity('Winners:O', scale=alt.Scale(domain=['Winner', 'Loser'], range=[0.4, 0.1]))
).properties(
    width = 500,
    height = 400,
    title = {
        "text": "\"Think we may wanna write this stuff down?\"",
        "subtitle": "Available Nathan's Hot Dog Eating Contest results",
        "subtitleFontSize": 16
    }
)

scatter
```

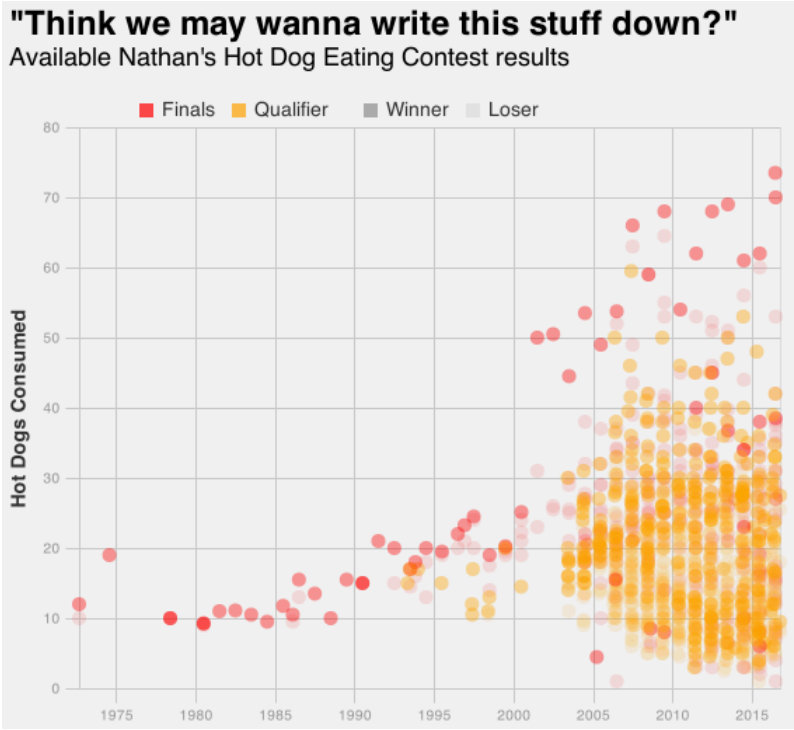
Out[18]:


checkpoint 6 (BONUS): Move the legends to the top of the plot and make it horizontal with a larger font. You will get full point if you:

- Completed checkpoint 5

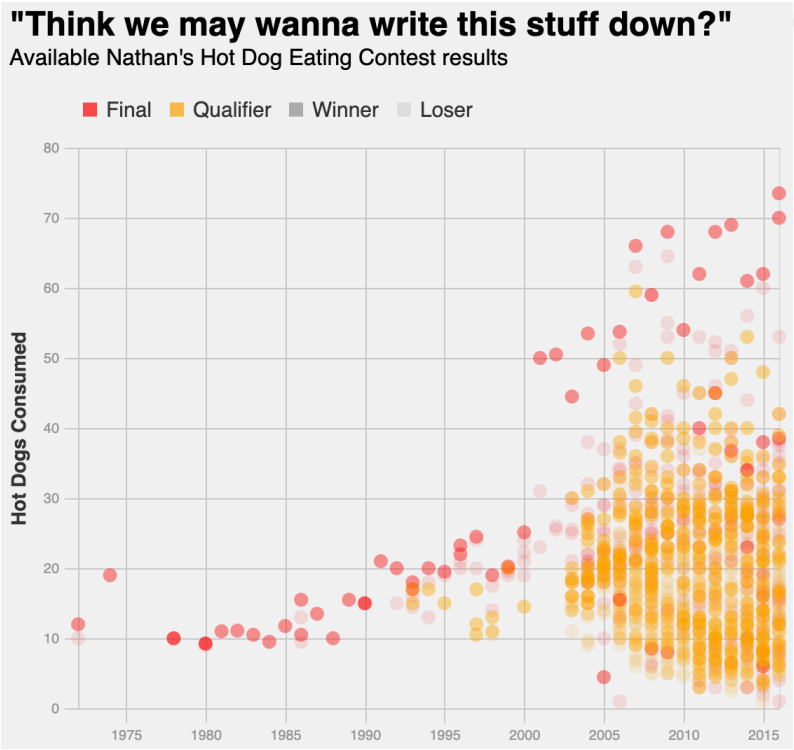
- Move the legends to the top of the plot
- Lay them out horizontally
- Increase the plot size in the legend

You chart should look like:



In [19]:
```
#TODO: Replicate task 2, checkpoint 6
# Place the legend above the scatter plot, layout the legend horizontally, increase the font size of the legend

## Notes:
## use symbolSize the increase the size of the legend symbols
## use orient and direction to layout the legend horizontally
## use labelFontSize to increase the font size of the legend

scatter = alt.Chart(df).mark_circle(size=100).encode(
    alt.X('year(Date):T', axis=alt.Axis(title=None)),
    alt.Y('Consumed:Q', title="Hot Dogs Consumed"),
    alt.Color('Qualifiers:N', scale=alt.Scale(domain=['Final', 'Qualifier'], range=['red', 'orange']),
                              legend=alt.Legend(title=None, labelFontSize=15, symbolSize=100, orient='top', direc
    alt.Opacity('Winners:O', scale=alt.Scale(domain=['Winner', 'Loser'], range=[0.4, 0.1]),
                              legend=alt.Legend(title=None, labelFontSize=15, symbolSize=100, orient='top', direc
).properties(
    width = 500,
    height = 400,
    title = {
        "text": "\"Think we may wanna write this stuff down?\"",
        "subtitle": "Available Nathan's Hot Dog Eating Contest results",
        "subtitleFontSize": 16
    }
)

scatter
```

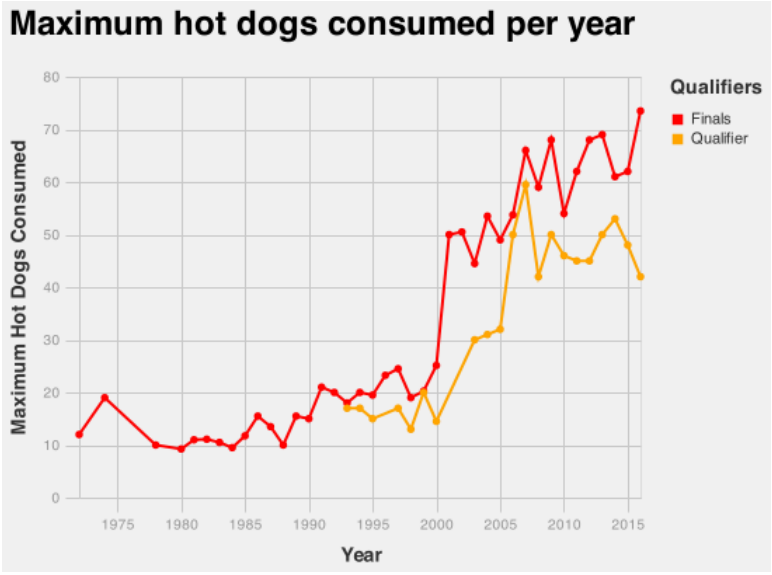
Out[19]:



Task #3

Create another new plot, showing the maximum hot dogs consumed per year, in qualifiers and finals

Here is what you should aim to create:



Step 1: Write down your plan for the visualization (edit this cell)

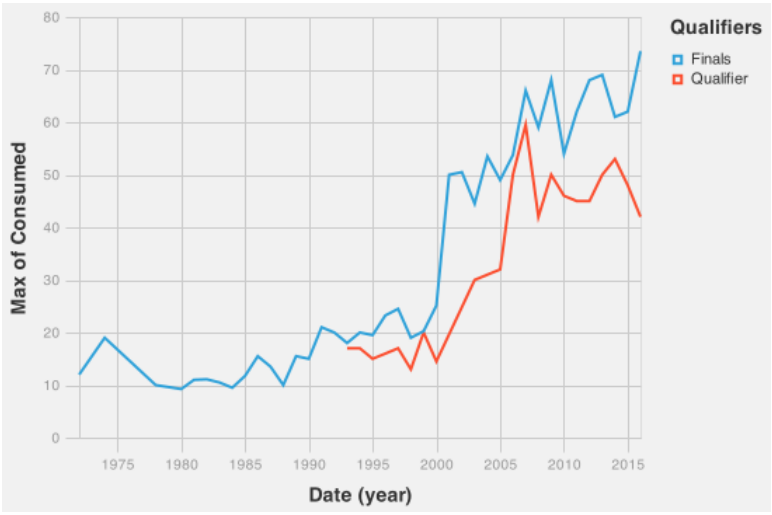
- mark type: line+point
- Encoding Specification:
- > x : position: year: Temporal (date transformed to year)
- > y : position: max hot dogs consumed: quantitative
- > color : contest type (final, qualifier): nominal

Step 2: Create your chart, step by step

checkpoint 1: plot the maximum per year, with a different color for qualifiers vs finals. You will get full point if you:

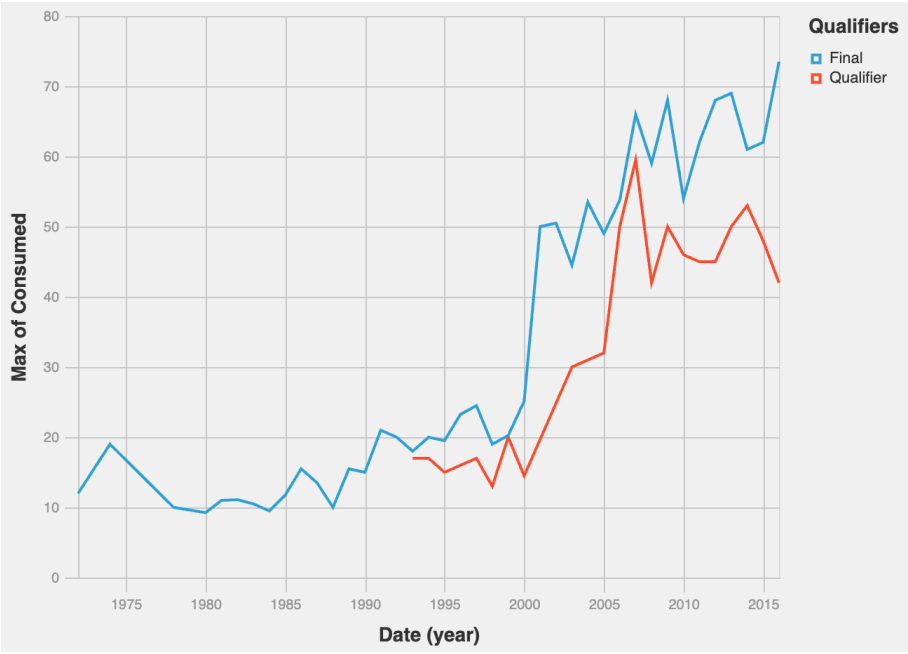
- Plot the right data
- Specify the correct mark
- Use the correct x, y, and color encodings

You chart should look like:



In [20]:
```
#TODO: Replicate task 3, checkpoint 1

max_consumed_line = alt.Chart(df).mark_line().encode(
    alt.X('year(Date):T'),
    alt.Y('max(Consumed):Q'),
    alt.Color('Qualifiers:N')
).properties(
    width = 500,
    height = 400
)

max_consumed_line
```

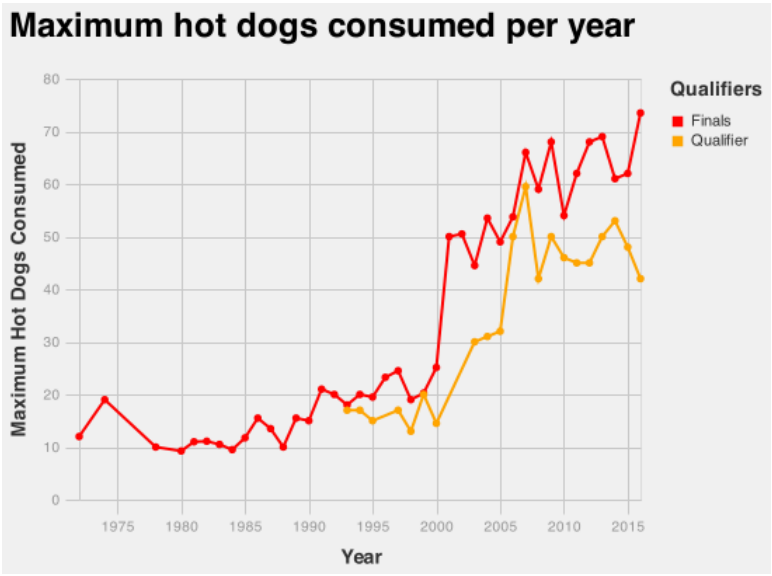
Out[20]:



checkpoint 2: change the colors to match the target plot, add points, and clean up labels and title. You will get full point if you:

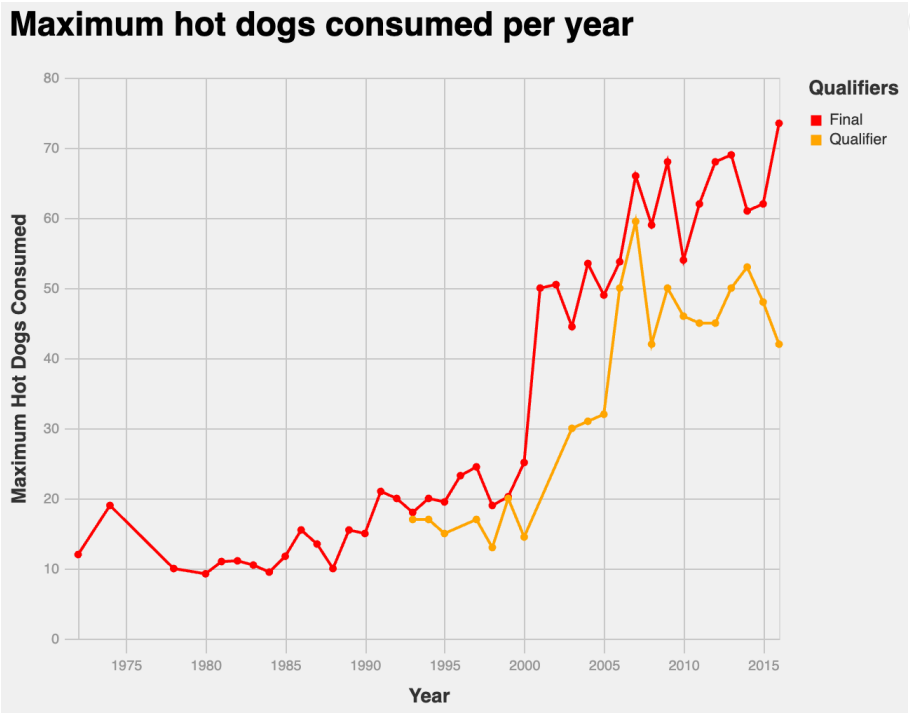
- Completed checkpoint 1
- Change the colors to match Task 2
- add points to the line pot
- Change the x-axis and y-axis labels to match the specificatio
- Add a plot title

You chart should look like:



In [21]:
```
#TODO: Replicate task 3, checkpoint 2

# hint, you can use mark_line(point=True) to add points to a line plot
max_consumed_line = alt.Chart(df).mark_line(point=True).encode(
    alt.X('year(Date):T', title = "Year"),
    alt.Y('max(Consumed):Q', title = "Maximum Hot Dogs Consumed"),
    alt.Color('Qualifiers:N', scale=alt.Scale(domain=['Final', 'Qualifier'], range=['red', 'orange']))
).properties(
    width = 500,
    height = 400,
    title = "Maximum hot dogs consumed per year"
)

max_consumed_line
```

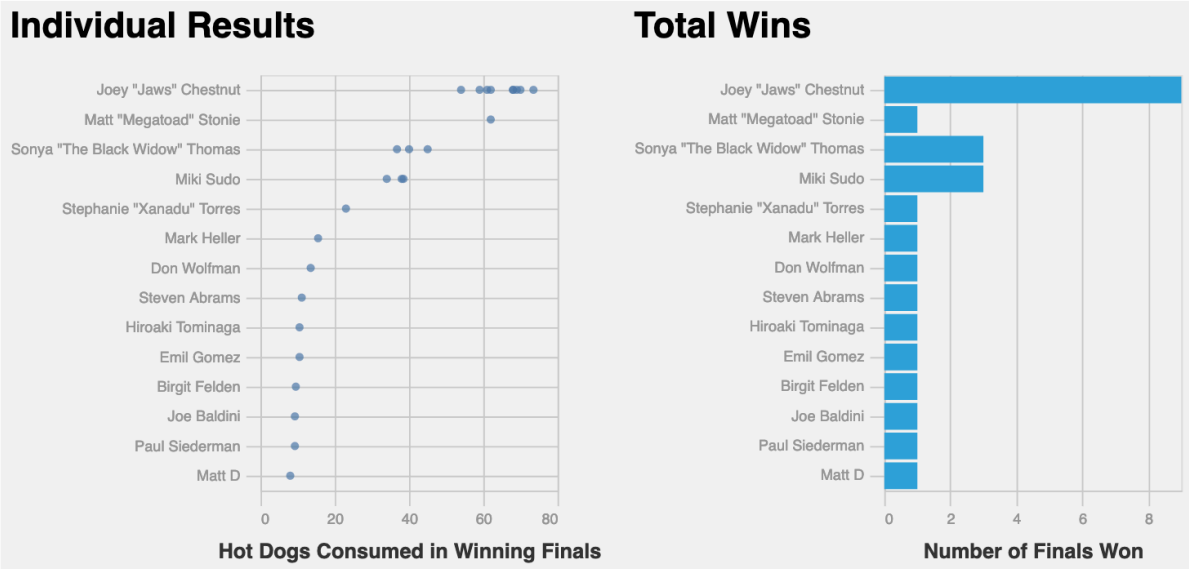
Out[21]:



Task #4

Create a pair of plots, showing the Winners of Finals of the 10 minute competitions

Here is what you should aim to create:



Step 1: Write down your plan for the visualization (edit this cell)

Left chart:

- mark type: point
- Encoding Specification:
- > x: position: hot dogs consumed in finals: quantitative
- > y: position: individual player: nominal

Right chart:

- mark type: bar
- Encoding Specification:
- > x: position/length: Number of Finals won: quantitative
- > y: position: individual player: nominal

Compound Method (how to join these charts together?): horizontal concatenation

Step 2: select the relevant data using pandas

Select the set of rows where:

- Place = 1st
- The competition is a NOT a qualifier
- The Duration is 10 minutes

In [22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1501 entries, 540 to 2312
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Place       1501 non-null   object
 1   Consumed    1501 non-null   float64
 2   Name        1501 non-null   object
 3   Contest     1501 non-null   object
 4   Location    1501 non-null   object
 5   Date        1501 non-null   datetime64[ns]
 6   Minutes     1501 non-null   float64
 7   Winners     1501 non-null   object
 8   Qualifiers  1501 non-null   object
dtypes: datetime64[ns](1), float64(2), object(6)
memory usage: 117.3+ KB
```

In [23]:
```
# TODO: Extract a subset of the rows to match the criteria given above

# Hint, refer back to where we excluded the "?" values above, for a hint on how to do this
# Hint, this should give you 26 rows, when selected from the filtered data frame above
# Note: there is some ambiguity in what counts as a qualifier. To get 26 rows, exclude every row where Contest
```

```
# Alternatively, if you get a slightly different number (e.g., 27) that will also be accepted, but will produce

final_winner_df = df[(df['Qualifiers'] == "Final") & (df['Winners'] == "Winner") & (df['Minutes']==10)]

final_winner_df.shape
```

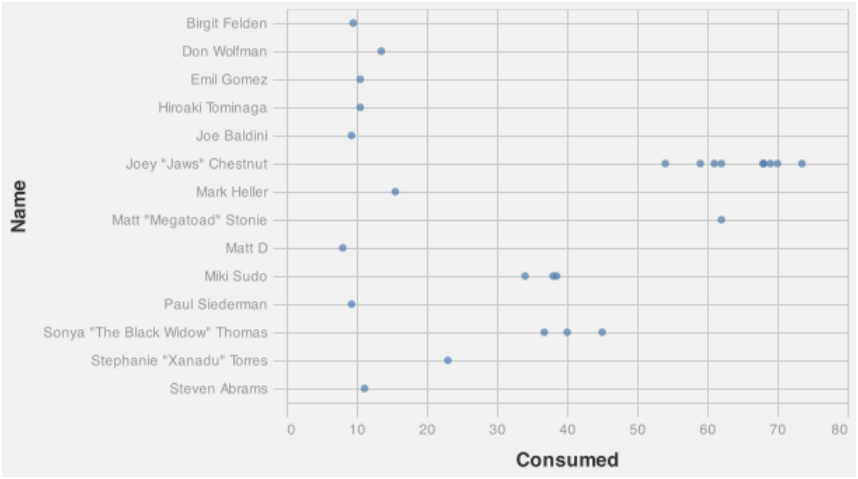
Out[23]: (26, 9)

Step 3: Create your chart

checkpoint 1: plot the number of hot dogs consumed by each competitor in the 10 minute finals they have won. You will get full point if you:

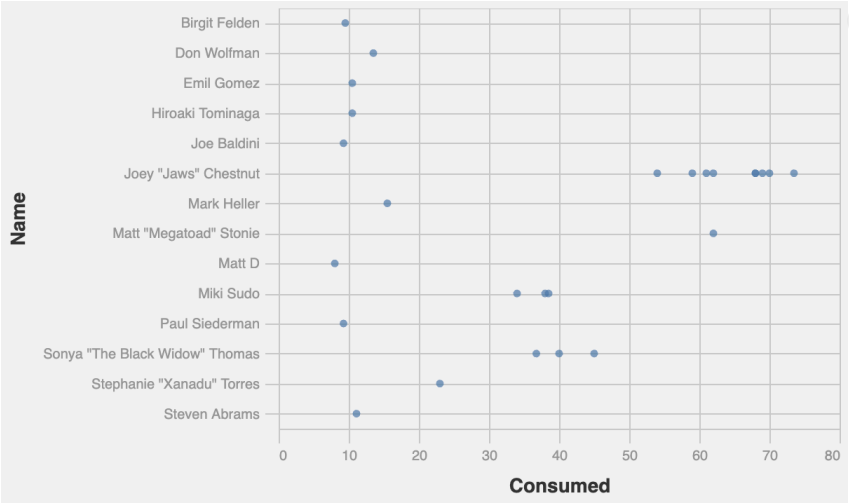
- Plot the right data
- Specify the correct mark
- Use the correct x and y encodings

You chart should look like:



In [24]:
```
#TODO: Replicate task 4, checkpoint 1

final_winner_point = alt.Chart(final_winner_df).mark_point().encode(
    alt.X('Consumed:Q'),
    alt.Y('Name:N'),
).properties(
    width = 400,
    height = 300,
)

final_winner_point
```

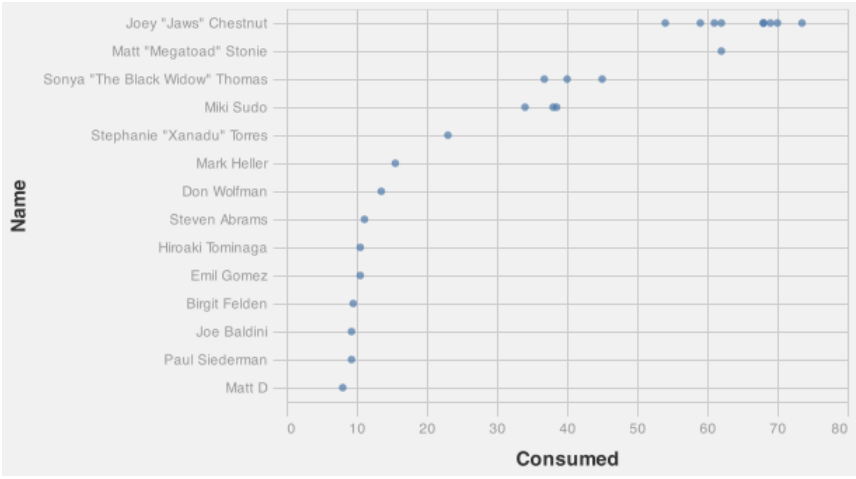
Out[24]:



checkpoint 2: sort the names by average number of hot dogs consumed in those competitions. You will get full point if you:

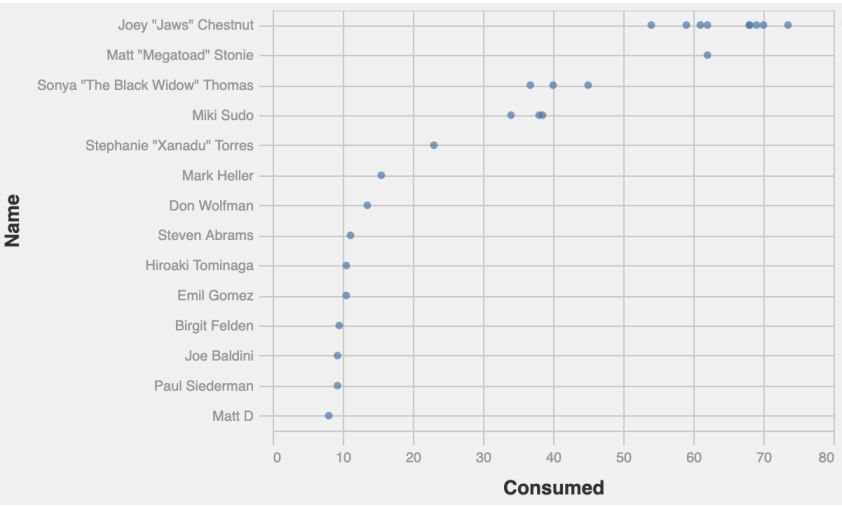
- Completed checkpoint 1
- Sort the names in the proper order

You chart should look like:



In [25]:
```
#TODO: Replicate task 4, checkpoint 2
# Sort the Y axis based on the average number of hot dogs consumed by each contestant

final_winner_point = alt.Chart(final_winner_df).mark_point().encode(
    alt.X('Consumed:Q'),
    alt.Y('Name:N', sort=alt.EncodingSortField(
        field="Consumed",
        op="mean",
        order="descending"
    ))
).properties(
    width = 400,
    height = 300,
)

final_winner_point
```

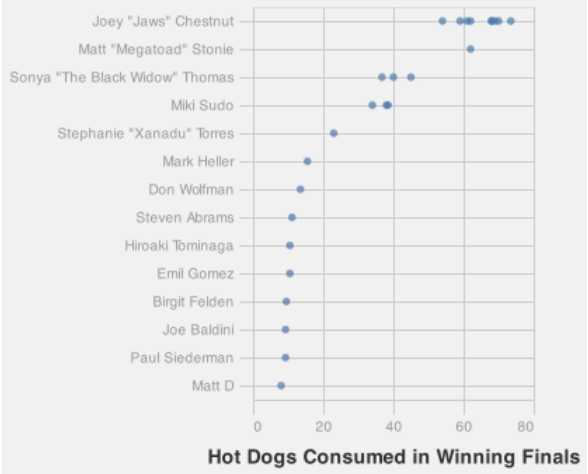
Out[25]:



checkpoint 3: fix the axis labels, and set the chart width to 200. You will get full point if you:

- Completed checkpoint 2
- Fix the x-axis labels
- Remove the y-axis label
- Set the plot width to be 200

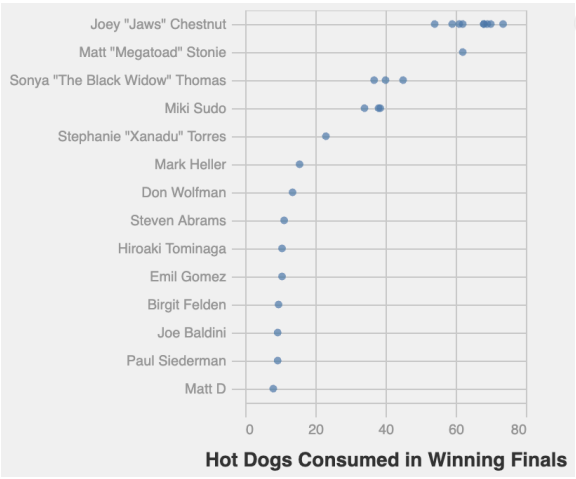
You chart should look like:



In [26]:

```
#TODO: Replicate task 4, checkpoint 3

final_winner_point = alt.Chart(final_winner_df).mark_point().encode(
    alt.X('Consumed:Q', title="Hot Dogs Consumed in Winning Finals"),
    alt.Y('Name:N', sort=alt.EncodingSortField(
            field="Consumed",
            op="mean",
            order="descending"
            ),
        title = None)
).properties(
    width = 200
)

final_winner_point
```

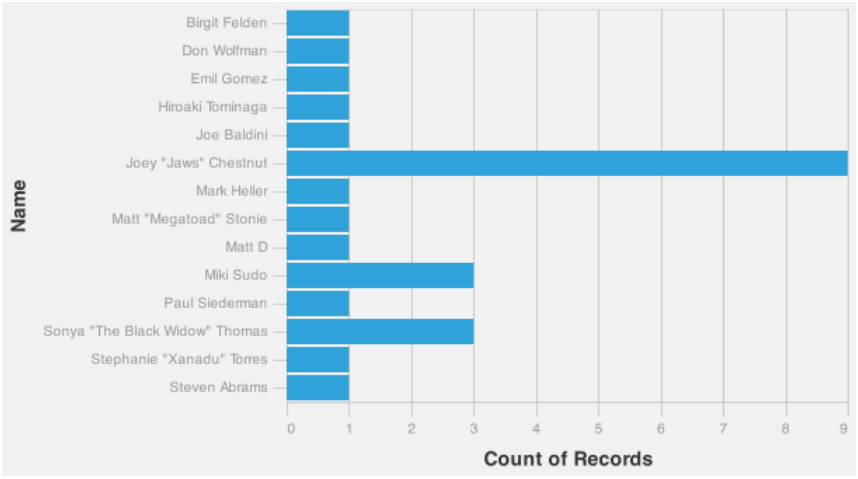
Out[26]:



checkpoint 4: plot the number of 10 minute finals competitions won by each competitor as a new chart. You will get full point if you:

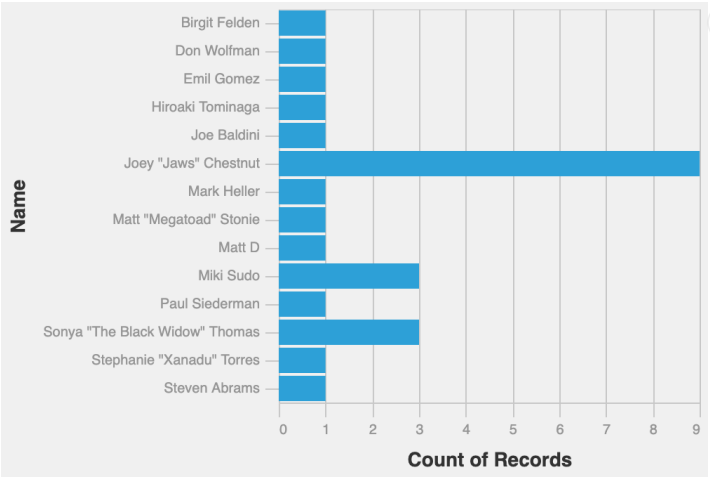
- Plot the right data
- Specify the correct mark
- Use the correct x and y encodings

You chart should look like:



In [27]:
```
#TODO: Replicate task 4, checkpoint 4

final_winner_count_bar = alt.Chart(final_winner_df).mark_bar().encode(
    alt.X('count():Q', title="Count of Records"),
    alt.Y('Name:N'),
)

final_winner_count_bar
```

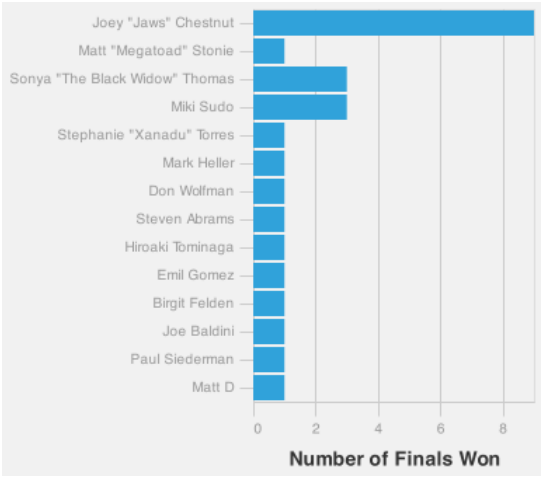
Out[27]:



checkpoint 5: apply the same sort order as above, and fix the axis labels and width. You will get full point if you:

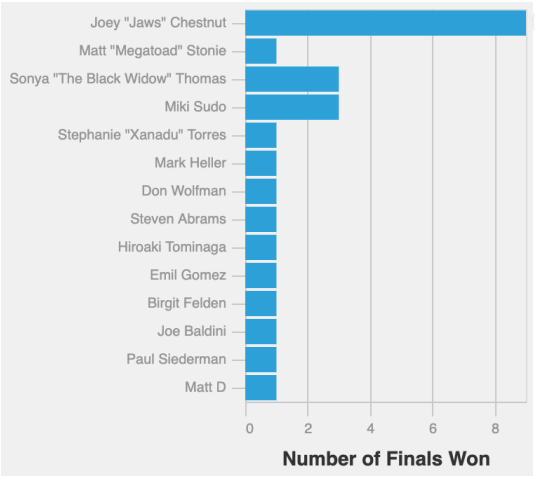
- Completed checkpoint 4
- Applied the correct sort order
- Fix the x-axis labels
- Remove the y-axis label
- Set the plot width to be 200

You chart should look like:



In [28]:
```
#TODO: Replicate task 4, checkpoint 5

final_winner_count_bar = alt.Chart(final_winner_df).mark_bar().encode(
    alt.X('count():Q', title="Number of Finals Won"),
    alt.Y('Name:N', title=None, sort=alt.EncodingSortField(
            field="Consumed",
            op="mean",
            order="descending"
            ))
).properties(
    width = 200
)

final_winner_count_bar
```

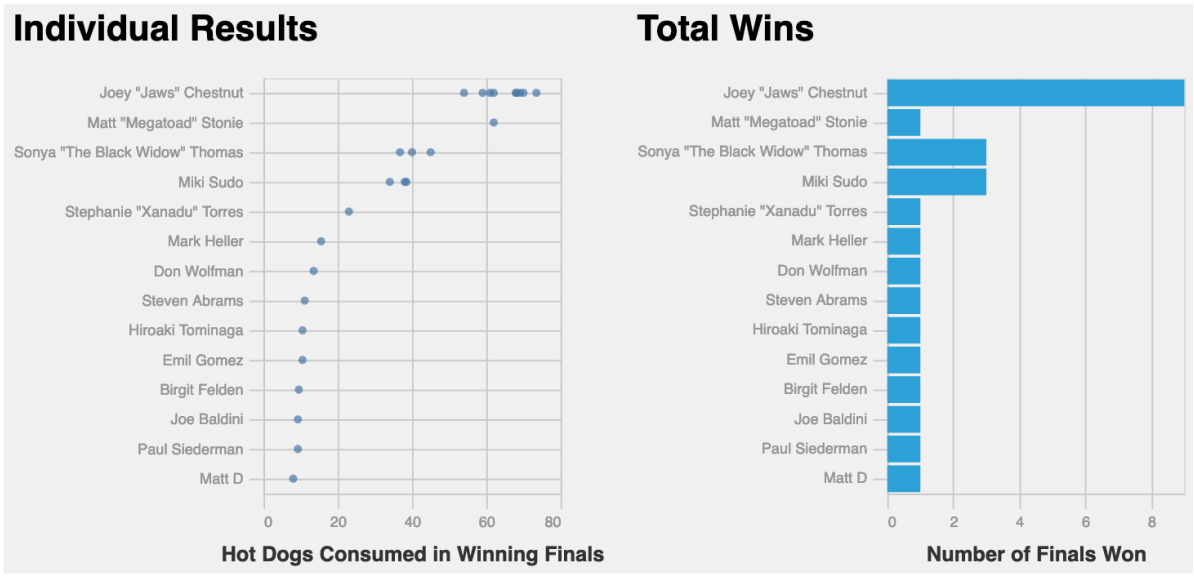
Out[28]:



checkpoint 6: place the two charts side by side. You will get full point if you:

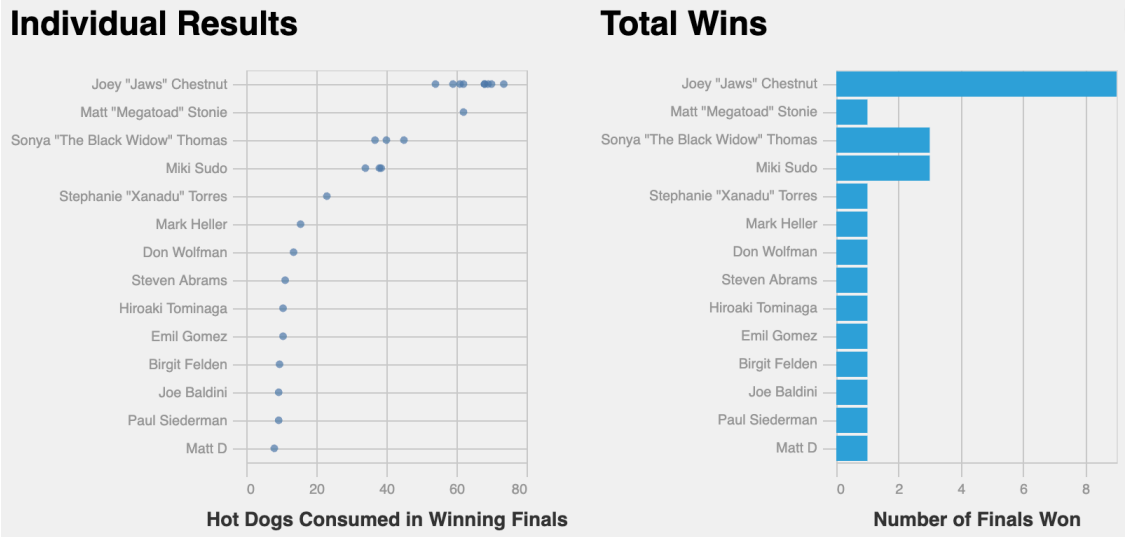
- Completed checkpoints 3 and 5
- Placed the two charts side by side
- Add a title to each chart

You chart should look like:



In [29]:
```
#TODO: Replicate task 4, checkpoint 6

# add title to each chart
final_winner_point = final_winner_point.properties(
    title = "Individual Results")

final_winner_count_bar = final_winner_count_bar.properties(
    title = "Total Wins")

# concatenate the two charts

final_concat = alt.hconcat(final_winner_point, final_winner_count_bar)

final_concat
```

Out[29]:



End of Assignment

Please run all cells (Runtime->Run all), and

1. save to PDF
   - We suggest using your browser's print feature: File->Print->Save PDF, you can try the notebook File->Download As->PDF, but we've noticed this doesn't work as well. If you're a Windows user and need help, take a look here
2. save to ipynb (File -> Download .ipynb)

Rename both files with your uniqname: e.g. uniqname.pdf/ uniqname.ipynb Upload both files to canvas.

In [ ]: