SIADS 591-592 Milestone I-II Dog Breed Popularity Analysis

This notebook will contain the analysis of dog breed popularity data

Python Library Loading

In [1]:
```
# Install packages that are needed that are currently not part of the environment
! pip install altair
! pip install vega_datasets
```

```
WARNING: The directory '/home/jovyan/.cache/pip/http' or its parent directory is not owned by the current user and the cache has been disabled. Please ch
eck the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
WARNING: The directory '/home/jovyan/.cache/pip' or its parent directory is not owned by the current user and caching wheels has been disabled. check the
permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
Requirement already satisfied: altair in /opt/conda/lib/python3.7/site-packages (4.1.0)
Requirement already satisfied: pandas>=0.18 in /opt/conda/lib/python3.7/site-packages (from altair) (0.25.0)
Requirement already satisfied: entrypoints in /opt/conda/lib/python3.7/site-packages (from altair) (0.3)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from altair) (1.17.0)
Requirement already satisfied: jsonschema in /opt/conda/lib/python3.7/site-packages (from altair) (3.0.2)
Requirement already satisfied: jinja2 in /opt/conda/lib/python3.7/site-packages (from altair) (2.10.1)
Requirement already satisfied: toolz in /opt/conda/lib/python3.7/site-packages (from altair) (0.10.0)
Requirement already satisfied: python-dateutil>=2.6.1 in /opt/conda/lib/python3.7/site-packages (from pandas>=0.18->altair) (2.8.0)
Requirement already satisfied: pytz>=2017.2 in /opt/conda/lib/python3.7/site-packages (from pandas>=0.18->altair) (2019.2)
Requirement already satisfied: six>=1.11.0 in /opt/conda/lib/python3.7/site-packages (from jsonschema->altair) (1.12.0)
Requirement already satisfied: pyrsistent>=0.14.0 in /opt/conda/lib/python3.7/site-packages (from jsonschema->altair) (0.15.4)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.7/site-packages (from jsonschema->altair) (41.0.1)
Requirement already satisfied: attrs>=17.4.0 in /opt/conda/lib/python3.7/site-packages (from jsonschema->altair) (19.1.0)
Requirement already satisfied: MarkupSafe>=0.23 in /opt/conda/lib/python3.7/site-packages (from jinja2->altair) (1.1.1)
WARNING: The directory '/home/jovyan/.cache/pip/http' or its parent directory is not owned by the current user and the cache has been disabled. Please ch
eck the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
WARNING: The directory '/home/jovyan/.cache/pip' or its parent directory is not owned by the current user and caching wheels has been disabled. check the
permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
Requirement already satisfied: vega_datasets in /opt/conda/lib/python3.7/site-packages (0.9.0)
Requirement already satisfied: pandas in /opt/conda/lib/python3.7/site-packages (from vega_datasets) (0.25.0)
Requirement already satisfied: pytz>=2017.2 in /opt/conda/lib/python3.7/site-packages (from pandas->vega_datasets) (2019.2)
Requirement already satisfied: python-dateutil>=2.6.1 in /opt/conda/lib/python3.7/site-packages (from pandas->vega_datasets) (2.8.0)
Requirement already satisfied: numpy>=1.13.3 in /opt/conda/lib/python3.7/site-packages (from pandas->vega_datasets) (1.17.0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from python-dateutil>=2.6.1->pandas->vega_datasets) (1.12.0)
```

In [2]:
```
# Import required python libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import lxml
from bs4 import BeautifulSoup
import altair as alt
from io import StringIO
from vega_datasets import data
from sklearn.linear_model import LinearRegression
alt.themes.enable('fivethirtyeight')
```

Out[2]:
```
ThemeRegistry.enable('fivethirtyeight')
```

Data Loading

Web Scraping

In [3]:
```
# When trying to connect to the web page for web scraping, there is a proxy error so the page was downloaded locally for loading
with open('project_data/MostPopularDogBreeds_ DogBreedPopularity2018.html','r') as akc_html:
    content = akc_html.read()
    soup = BeautifulSoup(content,'lxml')
    table = soup.find('table', class_='content-body__responsive-table')#.replace(' ','')
    title = ''
    for row in table.find_all('th'):
        title = title + row.text + ','

    body =''
    for row in table.find_all('tr'):
        for cell in row.find_all('td'):
            body = body + cell.text +','
        body = body + '\n'

    akc = title + body
```

In [4]:
```
popularity_df = pd.read_csv(StringIO(akc), sep=",")
popularity_df.drop(columns = 'Unnamed: 6', inplace=True)
popularity_df.head()
```

Out[4]:

	Breed	2018 Rank	2017 Rank	2016 Rank	2015 Rank	2014 Rank
0	Labrador Retrievers	1	1.0	1.0	1.0	1.0
1	German Shepherd Dogs	2	2.0	2.0	2.0	2.0
2	Golden Retrievers	3	3.0	3.0	3.0	3.0
3	French Bulldogs	4	4.0	6.0	6.0	9.0
4	Bulldogs	5	5.0	4.0	4.0	4.0

Loading CSV Files Using Pandas

In [5]:
```
# Read in the data sets
best_show_df = pd.read_csv("project_data/best_in_show.csv", skiprows=[1])
best_show_df.drop(list(best_show_df.filter(regex = 'Unnamed')), axis = 1, inplace = True)
best_show_df.rename(columns={"Dog breed": "Breed"}, inplace=True)
best_show_df.head()
```

Out[5]:

	Breed	category	datadog score	POPULARITY IN US	POPULARITY IN US.1	LIFETIME COST, $	5 LIFETIME COST	1 INTELLIGENCE (TRAINABILITY) ranking	INTELLIGENCE (TRAINABILITY) ranking	2 LONGEVITY	...	food per lifetime, $	Other regular costs, total per lifetime, $	total per year, $	total, per year, £	toys, presents, treats, per year, £	pet sitters, per year, £	groom per
0	Border Collie	herding	3.64	45.0	39.0	$20,143	48%	1	100%	12.52	...	3,486	13,095	1,046	784.0	121.0	126.0	2
1	Border Terrier	terrier	3.61	80.0	61.0	$22,638	14%	30	70%	14.00	...	3,898	14,643	1,046	784.0	121.0	126.0	2
2	Brittany	sporting	3.54	30.0	30.0	$22,589	16%	19	80%	12.92	...	5,171	13,514	1,046	784.0	121.0	126.0	2
3	Cairn Terrier	terrier	3.53	59.0	48.0	$21,992	22%	35	61%	13.84	...	3,854	14,476	1,046	784.0	121.0	126.0	2
4	Welsh Springer Spaniel	sporting	3.34	130.0	81.0	$20,224	47%	31	69%	12.49	...	3,478	13,064	1,046	784.0	121.0	126.0	2

5 rows × 61 columns

In [6]:
```
breed_info_df = pd.read_csv("project_data/AKC_Breed_Info.csv", encoding= 'unicode_escape')
breed_info_df.head()
```

Out[6]:

	Breed	height_low_inches	height_high_inches	weight_low_lbs	weight_high_lbs
0	Akita	26	28	80	120
1	Anatolian Sheepdog	27	29	100	150
2	Bernese Mountain Dog	23	27	85	110
3	Bloodhound	24	26	80	120
4	Borzoi	26	28	70	100

In [7]:
```
#add two more datasets
top_dog_by_state = pd.read_csv("project_data/top_dog_breeds_by_state.csv")
top_dog_by_state.head()
```

Out[7]:

	State	Top 1	Top 2	Top 3
0	Alabama	Labrador retriever	German shepherd	Beagle
1	Alaska	Labrador retriever	German shepherd	Golden retriever
2	Arizona	Labrador retriever	German shepherd	Golden retriever
3	Arkansas	Labrador retriever	German shepherd	Beagle
4	California	Labrador retriever	French bulldog	German shepherd

In [8]:
```
dog_iq = pd.read_csv("project_data/dog_intelligence.csv")
dog_iq.head()
```

Out[8]:

	Breed	Classification	obey	reps_lower	reps_upper
0	Border Collie	Brightest Dogs	95%	1	4
1	Poodle	Brightest Dogs	95%	1	4
2	German Shepherd	Brightest Dogs	95%	1	4
3	Golden Retriever	Brightest Dogs	95%	1	4
4	Doberman Pinscher	Brightest Dogs	95%	1	4

In [9]:
```
top_dog_by_state = pd.read_csv("project_data/top_dog_breeds_by_state.csv")
states = pd.read_csv("project_data/states.csv")
state_lat_long = pd.read_csv('project_data/statelatlong.csv')
```

Generate Some Mean Statistics

In [10]:
```
popularity_df["Mean Rank"] = popularity_df.apply(lambda row: np.mean(row[1:]), axis=1)
popularity_df["2014-2015 Change"] = -(popularity_df["2015 Rank"] - popularity_df["2014 Rank"])
popularity_df["2015-2016 Change"] = -(popularity_df["2016 Rank"] - popularity_df["2015 Rank"])
popularity_df["2016-2017 Change"] = -(popularity_df["2017 Rank"] - popularity_df["2016 Rank"])
popularity_df["2017-2018 Change"] = -(popularity_df["2018 Rank"] - popularity_df["2017 Rank"])
popularity_df.replace(-0.0, 0.0, inplace=True)
popularity_df.head()
```

Out[10]:

	Breed	2018 Rank	2017 Rank	2016 Rank	2015 Rank	2014 Rank	Mean Rank	2014-2015 Change	2015-2016 Change	2016-2017 Change	2017-2018 Change
0	Labrador Retrievers	1	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0
1	German Shepherd Dogs	2	2.0	2.0	2.0	2.0	2.0	0.0	0.0	0.0	0.0
2	Golden Retrievers	3	3.0	3.0	3.0	3.0	3.0	0.0	0.0	0.0	0.0
3	French Bulldogs	4	4.0	6.0	6.0	9.0	5.8	3.0	0.0	2.0	0.0
4	Bulldogs	5	5.0	4.0	4.0	4.0	4.4	0.0	0.0	-1.0	0.0

In [11]:

```
cols=[i for i in breed_info_df.columns if i not in ["Breed"]]
for col in cols:
    breed_info_df[col]=pd.to_numeric(breed_info_df[col], errors='coerce')
breed_info_df.dropna(inplace=True)
breed_info_df["Mean Height"] = breed_info_df.apply(lambda row: np.mean(row[1:3]), axis=1)
breed_info_df["Mean Weight"] = breed_info_df.apply(lambda row: np.mean(row[3:5]), axis=1)
breed_info_df.head()
```

Out[11]:

	Breed	height_low_inches	height_high_inches	weight_low_lbs	weight_high_lbs	Mean Height	Mean Weight
0	Akita	26.0	28.0	80.0	120.0	27.0	100.0
1	Anatolian Sheepdog	27.0	29.0	100.0	150.0	28.0	125.0
2	Bernese Mountain Dog	23.0	27.0	85.0	110.0	25.0	97.5
3	Bloodhound	24.0	26.0	80.0	120.0	25.0	100.0
4	Borzoi	26.0	28.0	70.0	100.0	27.0	85.0

Data Cleaning and Merging

In [12]:

```
# The popularity data frame has the dog breed names in plural for all breeds. The trailing 's' needs to be removed
popularity_df['Breed'] = popularity_df['Breed'].str.strip('s')
popularity_df['Breed'] = popularity_df['Breed'].str.strip(" ")
print(sorted(popularity_df['Breed']))
```

```
['Affenpinscher', 'Afghan Hound', 'Airedale Terrier', 'Akita', 'Alaskan Malamute', 'American English Coonhounds', 'American Eskimo Dog', 'American Foxhou
nds', 'American Hairless Terrier', 'American Staffordshire Terrier', 'American Water Spaniels', 'Anatolian Shepherd Dog', 'Australian Cattle Dog', 'Austr
alian Shepherd', 'Australian Terrier', 'Basenji', 'Basset Hound', 'Beagle', 'Bearded Collie', 'Beauceron', 'Bedlington Terrier', 'Belgian Malinoi', 'Belg
ian Sheepdog', 'Belgian Tervuren', 'Bergamasco Sheepdogs', 'Berger Picard', 'Bernese Mountain Dog', 'Bichon Frise', 'Black Russian Terrier', 'Black and T
an Coonhounds', 'Bloodhound', 'Bluetick Coonhound', 'Boerboel', 'Border Collie', 'Border Terrier', 'Borzoi', 'Boston Terrier', 'Bouviers des Flandre', 'B
oxer', 'Boykin Spaniel', 'Briard', 'Brittany', 'Brussels Griffon', 'Bull Terrier', 'Bulldog', 'Bullmastiff', 'Cairn Terrier', 'Canaan Dogs', 'Cani Cors
i', 'Cardigan Welsh Corgi', 'Cavalier King Charles Spaniel', 'Cesky Terriers', 'Chesapeake Bay Retriever', 'Chihuahua', 'Chinese Crested', 'Chinese Shar-
Pei', 'Chinook', 'Chow Chow', 'Cirnechi dell’Etna', 'Clumber Spaniel', 'Cocker Spaniel', 'Collie', 'Coton de Tulear', 'Curly-Coated Retrievers', 'Dachshu
nd', 'Dalmatian', 'Dandie Dinmont Terriers', 'Doberman Pinscher', 'Dogues de Bordeaux', 'English Cocker Spaniel', 'English Foxhounds', 'English Setter',
'English Springer Spaniel', 'English Toy Spaniel', 'Entlebucher Mountain Dog', 'Field Spaniel', 'Finnish Lapphund', 'Finnish Spitz', 'Flat-Coated Retriev
er', 'French Bulldog', 'German Pinscher', 'German Shepherd Dog', 'German Shorthaired Pointer', 'German Wirehaired Pointer', 'Giant Schnauzer', 'Glen of I
maal Terriers', 'Golden Retriever', 'Gordon Setter', 'Grand Basset Griffon Vendeens', 'Great Dane', 'Great Pyrenee', 'Greater Swiss Mountain Dog', 'Greyh
ound', 'Harrier', 'Havanese', 'Ibizan Hound', 'Icelandic Sheepdogs', 'Irish Red and White Setter', 'Irish Setter', 'Irish Terrier', 'Irish Water Spaniel
s', 'Irish Wolfhound', 'Italian Greyhound', 'Japanese Chin', 'Keeshonden', 'Kerry Blue Terrier', 'Komondorok', 'Kuvaszok', 'Labrador Retriever', 'Lagotti
Romagnoli', 'Lakeland Terrier', 'Leonberger', 'Lhasa Apso', 'Lowchen', 'Maltese', 'Manchester Terrier', 'Mastiff', 'Miniature American Shepherd', 'Miniat
ure Bull Terrier', 'Miniature Pinscher', 'Miniature Schnauzers', 'Neapolitan Mastiff', 'Nederlandse Kooikerhondje', 'Newfoundland', 'Norfolk Terrier', 'N
orwegian Buhunds', 'Norwegian Elkhound', 'Norwegian Lundehund', 'Norwich Terrier', 'Nova Scotia Duck Tolling Retriever', 'Old English Sheepdog', 'Otterho
und', 'Papillon', 'Parson Russell Terrier', 'Pekingese', 'Pembroke Welsh Corgi', 'Petit Basset Griffon Vendeen', 'Pharoah Hounds', 'Plott Hounds', 'Point
er', 'Polish Lowland Sheepdogs', 'Pomeranian', 'Poodle', 'Portuguese Podengo Pequeno', 'Portuguese Water Dog', 'Pug', 'Pulik', 'Pumik', 'Pyrenean Shepher
ds', 'Rat Terrier', 'Redbone Coonhounds', 'Rhodesian Ridgeback', 'Rottweiler', 'Russell Terrier', 'Saluki', 'Samoyed', 'Schipperke', 'Scottish Deerhound
s', 'Scottish Terrier', 'Sealyham Terriers', 'Shetland Sheepdog', 'Shiba Inu', 'Shih Tzu', 'Siberian Huskie', 'Silky Terrier', 'Skye Terriers', 'Slough
i', 'Smooth Fox Terrier', 'Soft Coated Wheaten Terrier', 'Spanish Water Dog', 'Spinoni Italiani', 'St. Bernard', 'Staffordshire Bull Terrier', 'Standard
Schnauzer', 'Sussex Spaniels', 'Swedish Vallhunds', 'Tibetan Mastiff', 'Tibetan Spaniel', 'Tibetan Terrier', 'Toy Fox Terrier', 'Treeing Walker Coonhoun
d', 'Vizsla', 'Weimaraner', 'Welsh Springer Spaniel', 'Welsh Terrier', 'West Highland White Terrier', 'Whippet', 'Wire Fox Terrier', 'Wirehaired Pointing
Griffon', 'Wirehaired Vizslas', 'Xoloitzcuintli', 'Yorkshire Terrier']
```

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

Out[13]:

	Breed	2018 Rank	2017 Rank	2016 Rank	2015 Rank	2014 Rank	Mean Rank	2014-2015 Change	2015-2016 Change	2016-2017 Change	2017-2018 Change
0	Labrador Retriever	1	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0
1	German Shepherd Dog	2	2.0	2.0	2.0	2.0	2.0	0.0	0.0	0.0	0.0
2	Golden Retriever	3	3.0	3.0	3.0	3.0	3.0	0.0	0.0	0.0	0.0
3	French Bulldog	4	4.0	6.0	6.0	9.0	5.8	3.0	0.0	2.0	0.0
4	Bulldog	5	5.0	4.0	4.0	4.0	4.4	0.0	0.0	-1.0	0.0

In [14]:
```
# Lets see what dogs are unique in the popularity and breed info frames
popularity_unique = [x for x in set(popularity_df['Breed']) if x not in set(breed_info_df['Breed'])]
breed_info_unique = [x for x in set(breed_info_df['Breed']) if x not in set(popularity_df['Breed'])]
print("Popularity unique: {}".format(sorted(popularity_unique)))
print("Breed Info unique: {}".format(sorted(breed_info_unique)))
```

```
Popularity unique: ['Airedale Terrier', 'Alaskan Malamute', 'American English Coonhounds', 'American Eskimo Dog', 'American Foxhounds', 'American Hairles
s Terrier', 'American Water Spaniels', 'Anatolian Shepherd Dog', 'Belgian Malinoi', 'Bergamasco Sheepdogs', 'Berger Picard', 'Black and Tan Coonhounds',
'Bluetick Coonhound', 'Boerboel', 'Bouviers des Flandre', 'Boykin Spaniel', 'Bulldog', 'Canaan Dogs', 'Cani Corsi', 'Cesky Terriers', 'Chinese Shar-Pei',
'Chinook', 'Cirnechi dell’Etna', 'Cocker Spaniel', 'Collie', 'Coton de Tulear', 'Curly-Coated Retrievers', 'Dandie Dinmont Terriers', 'Dogues de Bordeau
x', 'English Cocker Spaniel', 'English Foxhounds', 'Entlebucher Mountain Dog', 'Finnish Lapphund', 'Flat-Coated Retriever', 'Glen of Imaal Terriers', 'Gr
and Basset Griffon Vendeens', 'Great Pyrenee', 'Greater Swiss Mountain Dog', 'Havanese', 'Icelandic Sheepdogs', 'Irish Red and White Setter', 'Irish Wate
r Spaniels', 'Keeshonden', 'Komondorok', 'Kuvaszok', 'Lagotti Romagnoli', 'Leonberger', 'Lhasa Apso', 'Lowchen', 'Manchester Terrier', 'Miniature America
n Shepherd', 'Miniature Bull Terrier', 'Miniature Pinscher', 'Miniature Schnauzers', 'Neapolitan Mastiff', 'Nederlandse Kooikerhondje', 'Norfolk Terrie
r', 'Norwegian Buhunds', 'Norwegian Lundehund', 'Norwich Terrier', 'Old English Sheepdog', 'Otterhound', 'Parson Russell Terrier', 'Pekingese', 'Pembroke
Welsh Corgi', 'Pharoah Hounds', 'Plott Hounds', 'Polish Lowland Sheepdogs', 'Poodle', 'Portuguese Podengo Pequeno', 'Pulik', 'Pumik', 'Pyrenean Shepherd
s', 'Rat Terrier', 'Redbone Coonhounds', 'Russell Terrier', 'Scottish Deerhounds', 'Sealyham Terriers', 'Shetland Sheepdog', 'Siberian Huskie', 'Skye Ter
riers', 'Sloughi', 'Smooth Fox Terrier', 'Soft Coated Wheaten Terrier', 'Spanish Water Dog', 'Spinoni Italiani', 'St. Bernard', 'Sussex Spaniels', 'Swedi
sh Vallhunds', 'Treeing Walker Coonhound', 'Wire Fox Terrier', 'Wirehaired Vizslas', 'Xoloitzcuintli']
Breed Info unique: ['Airdale Terrier', 'American Eskimo', 'American Foxhound', 'American Water Spaniel', 'Anatolian Sheepdog', 'Belgian Malinois', 'Black
And Tan Coonhound', 'Bouvier Des Flandres', 'Bull Dog', 'Canaan Dog', 'Chinese Shar Pei', 'Cocker Spaniel-American', 'Cocker Spaniel-English', 'Collie (R
ough) & (Smooth)', 'Curly Coated Retriever', 'Dandie Dinmont Terrier', 'English Foxhound', 'Flat Coated Retriever', 'Fox Terrier Â\x89Ð»Ð² Smooth', 'Fox
Terrier Â\x89Ð»Ð² Wirehair', 'Glen Imaal Terrier', 'Great Pyrenees', 'Great Swiss Mountain Dog', 'Irish Water Spaniel', 'Keeshond', 'Komondor', 'Kuvasz',
'Manchester Terrier (Standard)', 'Manchester Terrier (Toy)', 'Neopolitan Mastiff', 'Old English Sheepdog (Bobtail)', 'Otter Hound', 'Pharaoh Hound', 'Plo
tt Hound', 'Polish Lowland Sheepdog', 'Poodle Miniature', 'Poodle Standard', 'Poodle Toy', 'Puli', 'Redbone Coonhound', 'Saint Bernard', 'Scottish Deerho
und', 'Sealyham Terrier', 'Shetland Sheepdog (Sheltie)', 'Siberian Husky', 'Skye Terrier', 'Soft-Coated Wheaten Terrier', 'Spinone Italiano', 'Sussex Spa
niel']
```

In [15]:
```
# We can see that there are some breed that are named similarly but just need to be renamed in each so they match
# For example, there are multiple spellings of Airedale Terrier
# Let's try and rescue some of this data by creating a dictionary where the key is the current entry and the value is what it
# should be corrected to
correction_dict = {
    "Airdale Terrier": "Airedale Terrier",
    "American English Coonhounds": "American English Coonhound",
    "American Eskimo": "American Eskimo Dog",
    "American Foxhounds": "American Foxhound",
    "American Water Spaniels": "American Water Spaniel",
    "Anatolian Sheepdog": "Anatolian Shepherd Dog",
    "Belgian Malinois": "Belgian Malinoi",
    "Black and Tan Coonhounds": "Black and Tan Coonhound",
    "Black and Tan Coonhound": "Black And Tan Coonhound",
    "Bouvier Des Flandres": "Bouviers des Flandre",
    "Bull Dog": "Bulldog",
    "Canaan Dogs": "Canaan Dog",
    "Cane Corso": "Cani Corsi",
    "Cesky Terriers": "Cesky Terrier",
    "Chinese Shar-Pei": "Chinese Shar Pei",
    "Cocker Spaniel-American": "Cocker Spaniel",
    "Collie (Rough) & (Smooth)": "Collie",
    "Curly-Coated Retrievers": "Curly Coated Retriever",
    "Dandie Dinmont Terriers": "Dandie Dinmont Terrier",
    "English Foxhounds": "English Foxhound",
    "Flat Coated Retriever": "Flat-Coated Retriever",
    "German Shepherd": "German Shepherd Dog",
    "Glen of Imaal Terriers": "Glen of Imaal Terrier",
    "Great Pyrenees": "Great Pyrenee",
    "Great Swiss Mountain Dog": "Greater Swiss Mountain Dog",
    "Icelandic Sheepdogs": "Icelandic Sheepdog",
    "Irish Water Spaniels": "Irish Water Spaniel",
    "Keeshonden": "Keeshond",
    "Komondorok": "Komondor",
    "Kuvaszok": "Kuvasz",
    "Löwchen": "Lowchen",
    "Manchester Terrier (Standard)": "Manchester Terrier",
    "Miniature Schnauzers": "Miniature Schnauzer",
    "Neopolitan Mastiff": "Neapolitan Mastiff",
    "Norwegian Buhunds": "Norwegian Buhund",
    "Old English Sheepdog (Bobtail)": "Old English Sheepdog",
    "Otter Hound": "Otterhound",
    "Pharaoh Hounds": "Pharaoh Hound",
    "Plott Hounds": "Plott Hound",
    "Plott": "Plott Hound",
    "Polish Lowland Sheepdogs": "Polish Lowland Sheepdog",
    "Poodle Standard": "Poodle",
    "Pulik": "Puli",
    "Pyrenean Shepherds": "Pyrenean Shepherd",
    "Redbone Coonhounds": "Redbone Coonhound",
    "St. Bernard": "Saint Bernard",
    "Scottish Deerhounds": "Scottish Deerhound",
    "Sealyham Terriers": "Sealyham Terrier",
    "Shetland Sheepdog (Sheltie)": "Shetland Sheepdog",
    "Siberian Huskie": "Siberian Husky",
    "Skye Terriers": "Skye Terrier",
    "Soft Coated Wheaten Terrier": "Soft-Coated Wheaten Terrier",
```

```
    "Spinone Italiano": "Spinoni Italiani",
    "Sussex Spaniels": "Sussex Spaniel",
    "Swedish Vallhunds": "Swedish Vallhund",
    "Wirehaired Vizslas": "Wirehaired Vizsla"
}
```

In [16]:
```
# Now that we have a change dictionary, let's update the breeds in each of the dictionaries so that they match
popularity_df['Breed'] = popularity_df['Breed'].map(correction_dict).fillna(popularity_df['Breed'])
breed_info_df['Breed'] = breed_info_df['Breed'].map(correction_dict).fillna(breed_info_df['Breed'])
best_show_df['Breed'] = best_show_df['Breed'].map(correction_dict).fillna(best_show_df['Breed'])
dog_iq['Breed'] = dog_iq['Breed'].map(correction_dict).fillna(dog_iq['Breed'])
```

In [17]:
```
# Now let's try to figure out how many rows are missing data for height and weight
print("Found {} rows without weight data".format(len(best_show_df[best_show_df['weight (lbs)'] == 'no data'])))
print("Found {} rows without height data".format(len(best_show_df[best_show_df['shoulder height (in)'] == 'no data'])))
```

```
Found 85 rows without weight data
Found 13 rows without height data
```

In [18]:
```
# We are going to try and do some replacements to fill these values by using the data in the breed_info_df
# First thing we have to do is set the index of the dataframes to the breed so we can use that when updating
best_show_df.head()
```

Out[18]:

	Breed	category	datadog score	POPULARITY IN US	POPULARITY IN US.1	LIFETIME COST, $	5 LIFETIME COST	1 INTELLIGENCE (TRAINABILITY) ranking	INTELLIGENCE (TRAINABILITY) ranking	2 LONGEVITY	...	food per lifetime, $	Other regular costs, total per lifetime, $	total per year, $	total, per year, £	toys, presents, treats, per year, £	pet sitters, per year, £	groon per y
0	Border Collie	herding	3.64	45.0	39.0	$20,143	48%	1	100%	12.52	...	3,486	13,095	1,046	784.0	121.0	126.0	2
1	Border Terrier	terrier	3.61	80.0	61.0	$22,638	14%	30	70%	14.00	...	3,898	14,643	1,046	784.0	121.0	126.0	2
2	Brittany	sporting	3.54	30.0	30.0	$22,589	16%	19	80%	12.92	...	5,171	13,514	1,046	784.0	121.0	126.0	2
3	Cairn Terrier	terrier	3.53	59.0	48.0	$21,992	22%	35	61%	13.84	...	3,854	14,476	1,046	784.0	121.0	126.0	2
4	Welsh Springer Spaniel	sporting	3.34	130.0	81.0	$20,224	47%	31	69%	12.49	...	3,478	13,064	1,046	784.0	121.0	126.0	2

5 rows × 61 columns

In [19]:
```
# Set the missing height and weight to 0 as we will use that value during the replacement
best_show_df[['weight (lbs)']] = best_show_df[['weight (lbs)']].replace(['no data','NA (3 classes)'], 0)
best_show_df[['shoulder height (in)']] = best_show_df[['shoulder height (in)']].replace(['no data','NA (3 classes)'], 0)
```

In [20]:
```
# Now that we have all the data frames with the same index, let's try a replacement for height and weight
breed_info_name_list = list(breed_info_df['Breed'])
for i in range(len(best_show_df)):
    if (best_show_df['weight (lbs)'][i] == 0) and (best_show_df['Breed'][i] in breed_info_name_list) :
        best_show_df['weight (lbs)'][i] = float(breed_info_df[breed_info_df['Breed']==(best_show_df['Breed'][i])]['Mean Weight'])
    if (best_show_df['shoulder height (in)'][i] == 0) and (best_show_df['Breed'][i] in breed_info_name_list) :
        best_show_df['shoulder height (in)'][i] = float(breed_info_df[breed_info_df['Breed']==(best_show_df['Breed'][i])]['Mean Height'])
```

In [21]:
```
# Now let's check to see how we are doing after the replacement
print("Found {} rows without weight data".format(len(best_show_df[best_show_df['weight (lbs)'] == 0])))
print("Found {} rows without height data".format(len(best_show_df[best_show_df['shoulder height (in)'] == 0])))
```

```
Found 17 rows without weight data
Found 1 rows without height data
```

In [22]:
```
# The last thing to do would be to merge the popularity_df and best_show_df to create our merged dataframe
merged_df = pd.merge(pd.merge(popularity_df,best_show_df,on='Breed'),dog_iq,on='Breed')
merged_df.head()
```

Out[22]:

	Breed	2018 Rank	2017 Rank	2016 Rank	2015 Rank	2014 Rank	Mean Rank	2014-2015 Change	2015-2016 Change	2016-2017 Change	...	toys, presents, treats, per year, £	pet sitters, per year, £	grooming, per year, £	vet fees per year, £	kennels per year, £	one offs, $	Classification	obey	reps_lower	reps_upper
0	Labrador Retriever	1	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	...	121.0	126.0	244.0	177.0	116.0	200.0	Brightest Dogs	95%	1	4
1	German Shepherd Dog	2	2.0	2.0	2.0	2.0	2.0	0.0	0.0	0.0	...	121.0	126.0	244.0	177.0	116.0	200.0	Brightest Dogs	95%	1	4
2	Golden Retriever	3	3.0	3.0	3.0	3.0	3.0	0.0	0.0	0.0	...	121.0	126.0	244.0	177.0	116.0	200.0	Brightest Dogs	95%	1	4
3	French Bulldog	4	4.0	6.0	6.0	9.0	5.8	3.0	0.0	2.0	...	121.0	126.0	244.0	177.0	116.0	200.0	Fair Working/Obedience Intelligence	30%	41	80
4	Bulldog	5	5.0	4.0	4.0	4.0	4.4	0.0	0.0	-1.0	...	121.0	126.0	244.0	177.0	116.0	200.0	Lowest Degree of Working/Obedience Intelligence	NaN	81	100

5 rows × 75 columns

In [23]:

```
#only contain breed rank,height,weight,iq
sort_df = merged_df[['Breed','Mean Rank','category','weight (lbs)','shoulder height (in)','Classification', 'obey']].sort_values('Mean Rank')
sort_df.replace(np.nan, '>30%', regex=True, inplace=True)
sort_df['Rank'] = range(1, len(sort_df)+1)
sort_df['Rank'] = sort_df['Rank'].astype('str')
sort_df['id'] = 1
sort_df['weight (lbs)'] = sort_df['weight (lbs)'].astype('float')
sort_df['shoulder height (in)'] = sort_df['shoulder height (in)'].astype('float')
sort_df.head()
```

Out[23]:

	Breed	Mean Rank	category	weight (lbs)	shoulder height (in)	Classification	obey	Rank	id
0	Labrador Retriever	1.0	sporting	67.5	23.00	Brightest Dogs	95%	1	1
1	German Shepherd Dog	2.0	herding	82.5	24.00	Brightest Dogs	95%	2	1
2	Golden Retriever	3.0	sporting	60.0	22.75	Brightest Dogs	95%	3	1
4	Bulldog	4.4	non-sporting	45.0	14.00	Lowest Degree of Working/Obedience Intelligence	>30%	4	1
5	Beagle	5.4	hound	24.0	14.00	Lowest Degree of Working/Obedience Intelligence	>30%	5	1

In [24]:

```
lst=[]
for i in range(100):
    if i < 20:
        lst.append('top 1-20')
    elif 20<i<40:
        lst.append('top 21-40')
    elif 40<i<60:
        lst.append('top 41-60')
    elif 60<i<80:
        lst.append('top 61-80')
    else:
        lst.append('top 81-100')
df = sort_df[:100]
df['Group']=lst
df.head()
```

Out[24]:

	Breed	Mean Rank	category	weight (lbs)	shoulder height (in)	Classification	obey	Rank	id	Group
0	Labrador Retriever	1.0	sporting	67.5	23.00	Brightest Dogs	95%	1	1	top 1-20
1	German Shepherd Dog	2.0	herding	82.5	24.00	Brightest Dogs	95%	2	1	top 1-20
2	Golden Retriever	3.0	sporting	60.0	22.75	Brightest Dogs	95%	3	1	top 1-20
4	Bulldog	4.4	non-sporting	45.0	14.00	Lowest Degree of Working/Obedience Intelligence	>30%	4	1	top 1-20
5	Beagle	5.4	hound	24.0	14.00	Lowest Degree of Working/Obedience Intelligence	>30%	5	1	top 1-20

In [25]:

```
# this dataframe is for the top dog breeds in different states
state_lat_long = state_lat_long.rename(columns={'State':'name','City':'State'})
new_state = pd.DataFrame({'State':['District of Columbia'], 'Top 1':'NaN', 'Top 2':'NaN', 'Top 3':'NaN'}, index=[50])
concat = pd.concat([top_dog_by_state,new_state])
concat = concat.sort_values('State').reset_index(drop=True)
new_state2 = pd.DataFrame({'State':['Puerto Rico'], 'Top 1':'NaN', 'Top 2':'NaN', 'Top 3':'NaN'}, index=[51])
concat = pd.concat([concat,new_state2])
lst=list(states['id'])
concat['id'] = lst
concat = pd.merge(concat,state_lat_long,on='State')
concat = concat.drop([8])
concat.head()
```

Out[25]:

	State	Top 1	Top 2	Top 3	id	name	Latitude	Longitude
0	Alabama	Labrador retriever	German shepherd	Beagle	1	AL	32.601011	-86.680736
1	Alaska	Labrador retriever	German shepherd	Golden retriever	2	AK	61.302501	-158.775020
2	Arizona	Labrador retriever	German shepherd	Golden retriever	4	AZ	34.168219	-111.930907
3	Arkansas	Labrador retriever	German shepherd	Beagle	5	AR	34.751928	-92.131378
4	California	Labrador retriever	French bulldog	German shepherd	6	CA	37.271875	-119.270415

Data Visualizaion

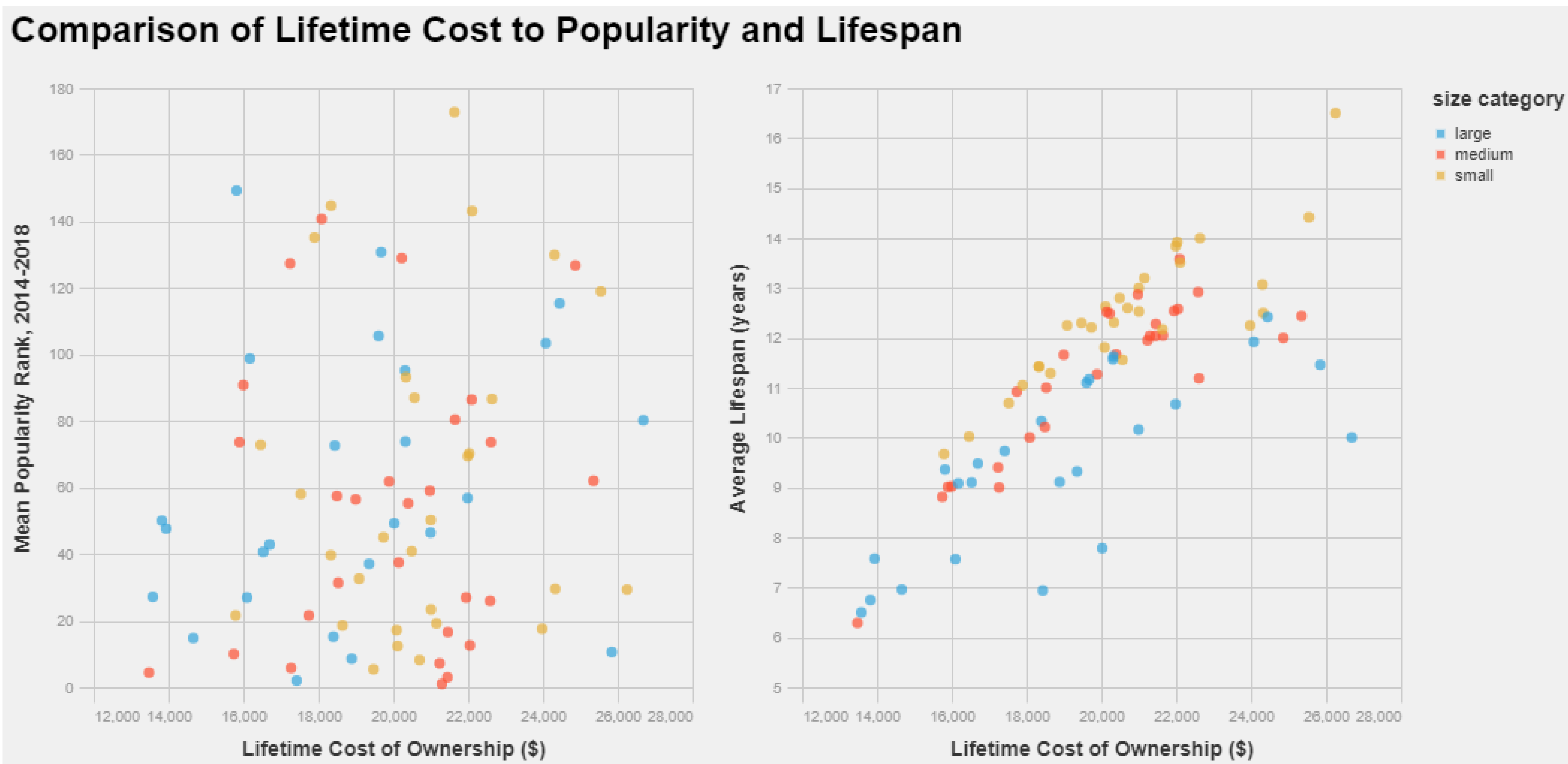
Lifetime cost vs. popularity

The first thing we want to look at is to see the relationship between lifetime cost and popularity. Are the dogs that are the most popular the most expensive? Is there a correlation between popularity and lifetime cost of ownership?

There does not look to be any trend in the data. The most popular breed, Laborador Retreiver, is in the middle of the pack for lifetime cost of ownership at 21,299. Interestingly, a chihuahua has a very high cost of ownership of 26,250. Maybe there is some relationship between lifespan and cost of ownership. Let's take a look at that

In [26]:
```
comparison_df = merged_df[['Breed', '2 LONGEVITY', 'LIFETIME COST, $', 'Mean Rank', 'size category']]
comparison_df = comparison_df[comparison_df['LIFETIME COST, $'] != 'no data']
comparison_df['LIFETIME COST, $'] = comparison_df['LIFETIME COST, $'].str.replace('$', '')
comparison_df['LIFETIME COST, $'] = comparison_df['LIFETIME COST, $'].str.replace(',', '').astype(int)
comparison_df = comparison_df[~comparison_df['2 LONGEVITY'].isin(['no data', '1.83 - really?'])]
comparison_df['2 LONGEVITY'] = comparison_df['2 LONGEVITY'].astype(float)

brush = alt.selection(type='interval', resolve='global')
color=alt.condition(brush, 'size category', alt.value('lightgray'))

base= alt.Chart(comparison_df).mark_circle(size=50).encode(
    x=alt.X('LIFETIME COST, $', title="Lifetime Cost of Ownership ($)", scale=alt.Scale(domain=[12000, 28000])),
    color=color
).add_selection(brush).properties(
    height=400,
    width=400)

combined_plots = (base.encode(y=alt.Y('Mean Rank', title='Mean Popularity Rank, 2014-2018'),
            tooltip=['Breed', 'LIFETIME COST, $', 'Mean Rank']) |
base.encode(y=alt.Y('2 LONGEVITY', title="Average Lifespan (years)", scale=alt.Scale(domain=[5, 17])),
            tooltip=['Breed', 'LIFETIME COST, $', '2 LONGEVITY'])).properties(title="Comparison of Lifetime Cost to Popularity and Lifespan")
combined_plots
```

Out[26]:



There does not look to be any pattern when comparing cost of ownership to popularity. The most popular breed, Laborador Retreiver, is in the middle of the pack for lifetime cost of ownership at 21,299. Interestingly, a chihuahua has a very high cost of ownership of 26,250.

We do see a trend when comparing cost of owndership to average lifespan. We can see that the doges with longer lifespans have a higher cost of ownership. We also see some interesting groupings when we color the data by size category as breeds of the same size category tend to be close together, almost in bands. I think it's worth running a linear regression analysis on this data so let's go ahead and do that now.

In [27]:
```
x = np.array(comparison_df['LIFETIME COST, $']).reshape(-1, 1)
y = comparison_df['2 LONGEVITY']
model = LinearRegression().fit(x, y)
r_sq = model.score(x, y)
print('coefficient of determination:', r_sq)
print('intercept:', model.intercept_)
print('slope:', model.coef_)
```

```
coefficient of determination: 0.5945146463559632
intercept: 1.161364673414143
slope: [0.00050039]
```

With a CoD of around 0.6, we can see there is a positive correlation between the average lifespan in years of a dog and the lifetime cost of ownership.

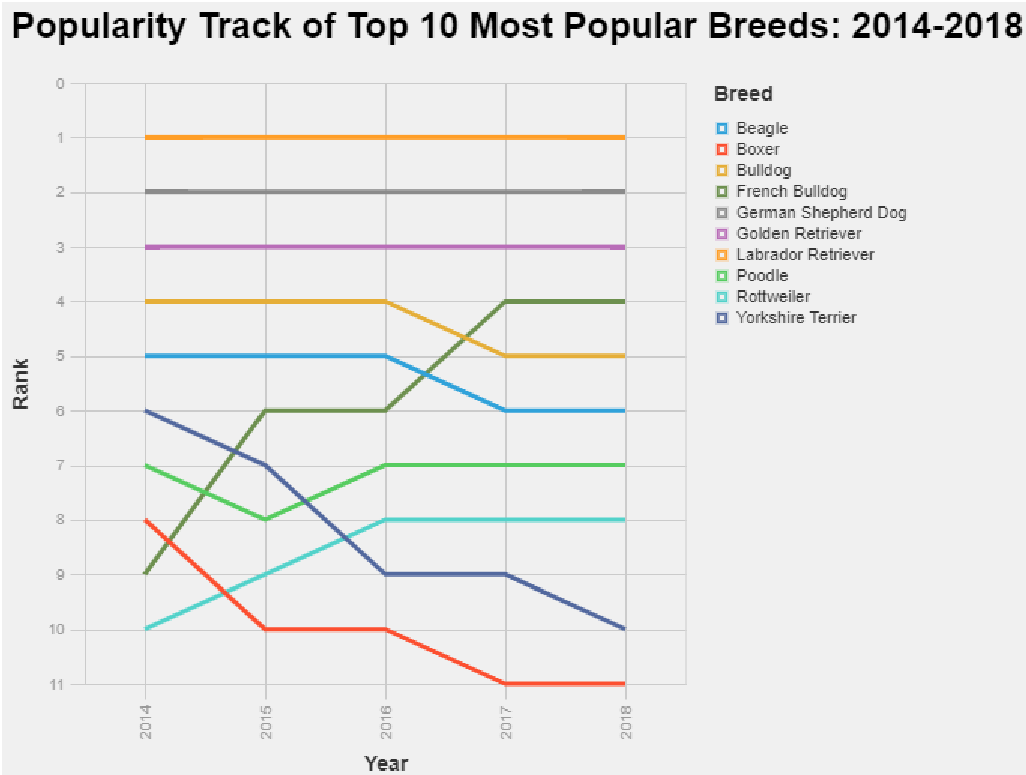
Trends in popularity data

Now let's take a look at the most popular dog breeds. We are interested in knowing which dogs had the largest change in popularity from year to year.

In [28]:
```
pop_trend_df = merged_df[['Breed', '2018 Rank', '2017 Rank', '2016 Rank', '2015 Rank', '2014 Rank']]
pop_trend_df.rename(columns={'2018 Rank': '2018',
                             '2017 Rank': '2017',
                             '2016 Rank': '2016',
                             '2015 Rank': '2015',
                             '2014 Rank': '2014'}, inplace=True)

pop_trend_df = pop_trend_df.melt(id_vars=['Breed'], var_name='Year', value_name='Rank')
mean_df = popularity_df[['Breed', 'Mean Rank']]
pop_trend_df = pop_trend_df.merge(mean_df, on='Breed')
```

In [29]:
```
alt.Chart(pop_trend_df[pop_trend_df['Mean Rank'] <= 10]).mark_line(size=3).encode(
    x=alt.X('Year'),
    y=alt.Y('Rank', sort='descending'),
    color='Breed'
).properties(height=400, width=400, title='Popularity Track of Top 10 Most Popular Breeds: 2014-2018')
```

Out[29]:



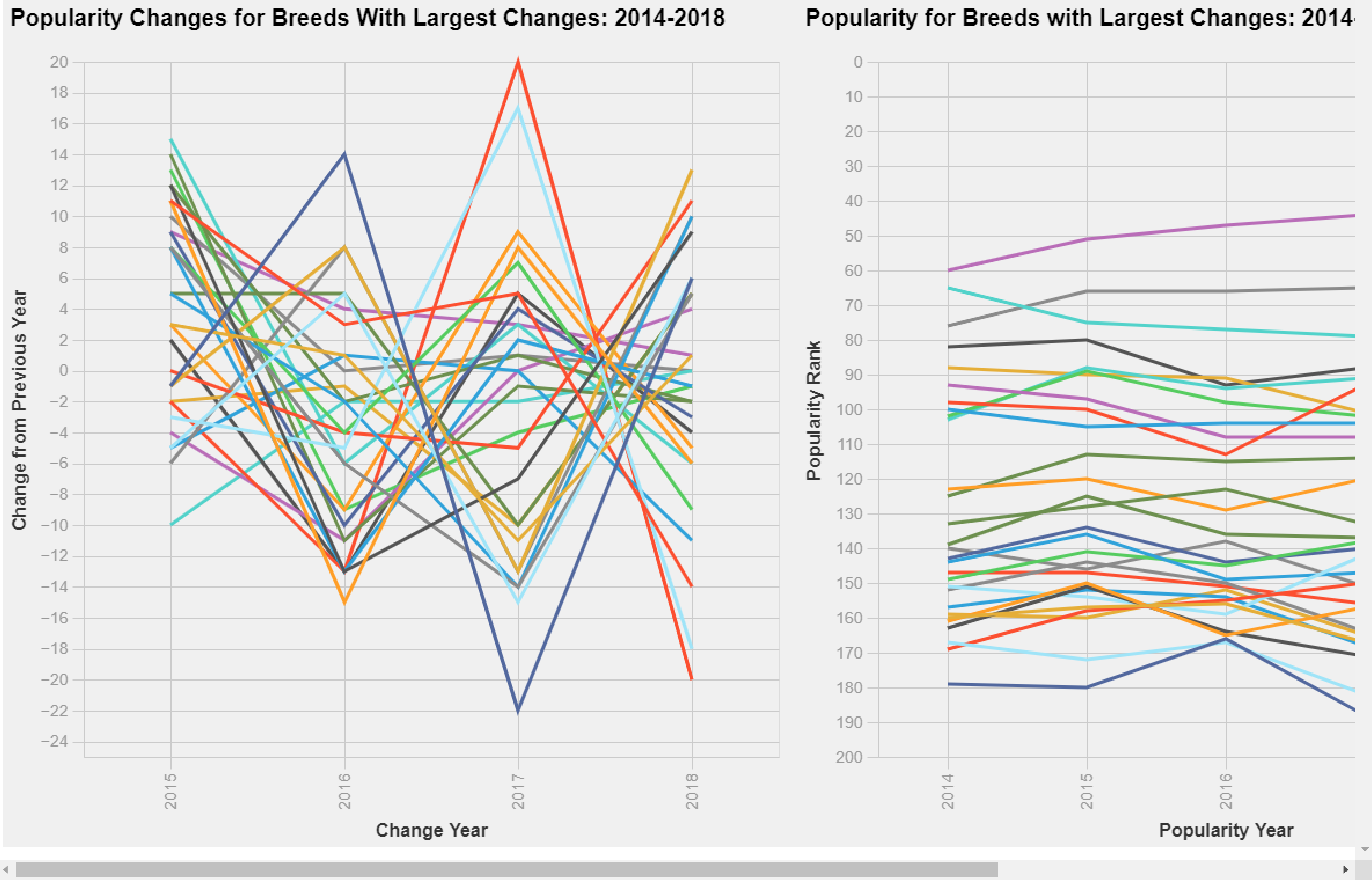
Here we can see that the top 3 most popular dogs have stayed consistent within the past 3 years: Laborador Retriever, German Shepherd Dog, and Golden Retriever. There have been some changes however as we can see the French Bulldog is increasing in popularity while both the Yorkshire Terrier and Boxer are decreasing in popularity. Let's take a look at the breeds with the biggest year to year changes.

In [30]:
```
pop_change_df = merged_df[['Breed', '2014-2015 Change', '2015-2016 Change',  '2016-2017 Change', '2017-2018 Change']]
pop_change_df.rename(columns={'2017-2018 Change': '2018',
                              '2016-2017 Change': '2017',
                              '2015-2016 Change': '2016',
                              '2014-2015 Change': '2015'}, inplace=True)

pop_change_df["Mean Change"] = pop_change_df.apply(lambda row: np.mean(row[1:]), axis=1)
pop_change_df["Max Change"] = pop_change_df.apply(lambda row: np.max(np.abs(row[1:])), axis=1)

change_df = pop_change_df[['Breed', 'Mean Change', 'Max Change']]

pop_change_df.drop(columns=['Mean Change', 'Max Change'], inplace=True)
pop_change_df = pop_change_df.melt(id_vars=['Breed'], var_name='Year', value_name='Rank')
pop_change_df = pop_change_df.merge(change_df, on='Breed')

top_change_limit = sorted(set(change_df['Max Change']))[-11:][0]
```

In [31]:

```
filtered_change_df = pop_change_df[(pop_change_df['Max Change'] >= top_change_limit) & (pop_change_df['Max Change'] < 100)]
filtered_breed_df = pop_trend_df[pop_trend_df['Breed'].isin(filtered_change_df['Breed'])]

filtered_breed_df.rename(columns={"Year": "Popularity Year", "Rank": "Popularity Rank", "Mean Rank": "Mean Popularity Rank"}, inplace=True)
filtered_change_df.rename(columns={"Year": "Change Year", "Rank": "Popularity Change"}, inplace=True)

complete_pop_change_frame = filtered_breed_df.merge(filtered_change_df, on='Breed')

multi = alt.selection_multi(fields=['Breed'])

base = alt.Chart(complete_pop_change_frame).properties(
    width=600,
    height=600
).add_selection(multi)

change_chart = base.mark_line(size=3).encode(
    x=alt.X('Change Year'),
    y=alt.Y('Popularity Change', title='Change from Previous Year'),
    color=alt.condition(multi,
                        alt.Color('Breed:N'),
                        alt.value('lightgray')),
    tooltip=['Breed', 'Popularity Change']
).properties(title='Popularity Changes for Breeds With Largest Changes: 2014-2018')

change_pop_chart = base.mark_line(size=3).encode(
    x=alt.X('Popularity Year'),
    y=alt.Y('Popularity Rank', sort='descending'),
    color=alt.condition(multi,
                        alt.Color('Breed:N'),
                        alt.value('lightgray')),
    tooltip=['Breed', 'Popularity Rank']
).properties(title='Popularity for Breeds with Largest Changes: 2014-2018')

alt.hconcat(change_chart, change_pop_chart).configure_title(
    fontSize=20).configure_axis(labelFontSize=14, titleFontSize=16)
```

Out[31]:

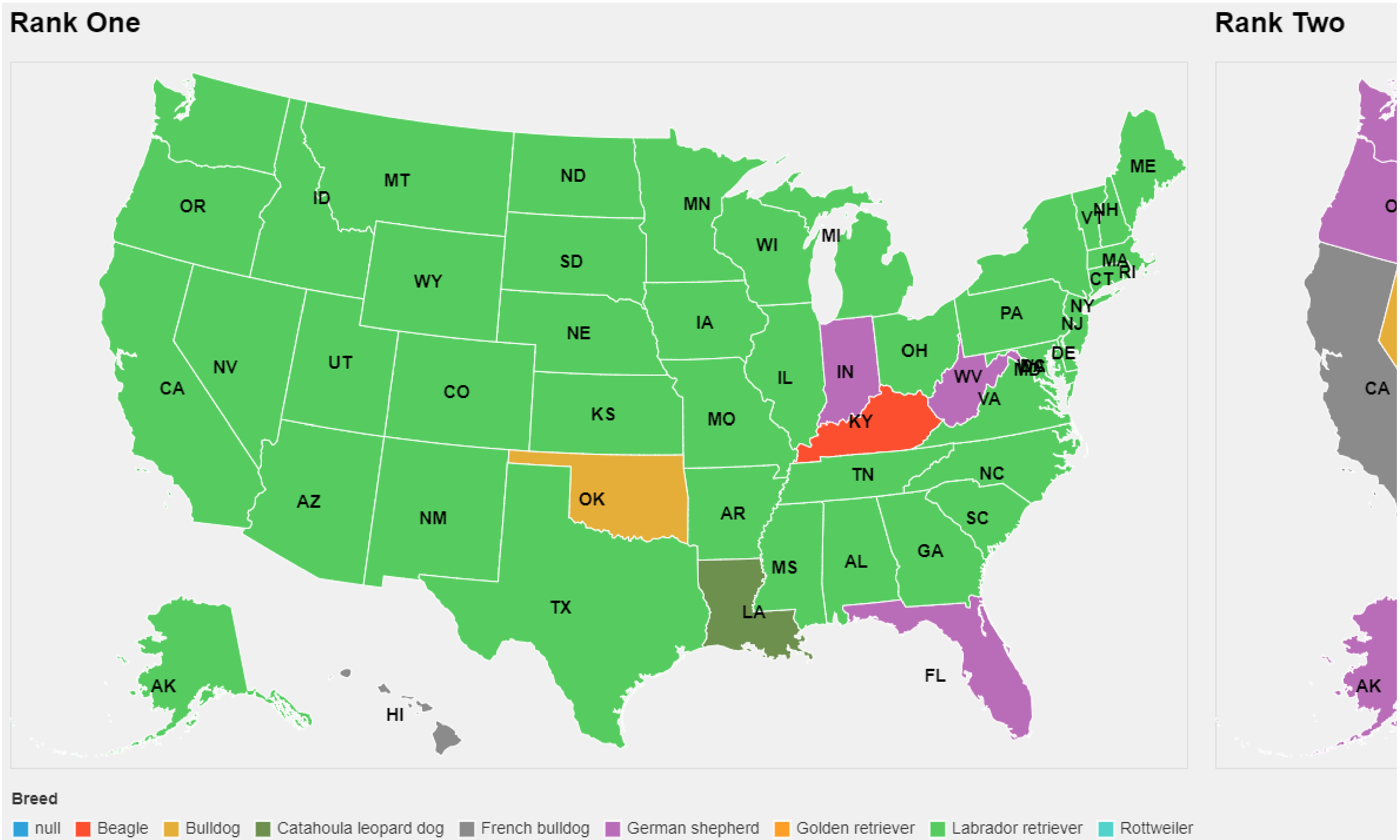


Most Popularity Dog Breeds in Every State

In [32]:
```
state_lst = list(top_dog_by_state['State'])
states = alt.topo_feature(data.us_10m.url, feature='states')

selection = alt.selection_single()

colorCondition1 = alt.condition(selection,'Top 1:N',alt.value('lightgrey'), title='Breed')
colorCondition2 = alt.condition(selection,'Top 2:N',alt.value('lightgrey'), title='Breed')
colorCondition3 = alt.condition(selection,'Top 3:N',alt.value('lightgrey'), title='Breed')

map1= alt.Chart(states).mark_geoshape(stroke='white').add_selection(selection).encode(
        color= colorCondition1,
        tooltip=['State:N','Top 1:N']
    ).transform_lookup(
        lookup='id',
        from_=alt.LookupData(concat, 'id', list(concat.columns))
    ).properties(
        width=1000,
        height=600,
        title='Rank One'
    ).project(
        type='albersUsa'
    )

map2= alt.Chart(states).mark_geoshape(stroke='white').add_selection(selection).encode(
        color= colorCondition2, #alt.Color('Top 1:N', scale=alt.Scale(scheme='set3')),
        tooltip=['State:N','Top 2:N']
    ).properties(
        title='Rank Two'
    ).transform_lookup(
        lookup='id',
        from_=alt.LookupData(concat, 'id', list(concat.columns))
    )

map3= alt.Chart(states).mark_geoshape(stroke='white').add_selection(selection).encode(
        color= colorCondition3, #alt.Color('Top 1:N', scale=alt.Scale(scheme='set3')),
        tooltip=['State:N','Top 3:N']
    ).properties(
        title='Rank Three'
    ).transform_lookup(
        lookup='id',
        from_=alt.LookupData(concat, 'id', list(concat.columns))
    )

annotation = alt.Chart(state_lat_long).mark_text(
        align='right',
        baseline='middle',
        fontSize = 15,
        fontStyle = 'bold',
        lineBreak='\n',
        dx = 7
    ).encode(
        longitude = 'Longitude:Q',
        latitude = 'Latitude:Q',
        text='name'
    )

map1 = map1+annotation
```

```
map2 = map2+annotation
map3 = map3+annotation

us_map = alt.hconcat(map1,map2,map3).configure_legend(
  orient='bottom',
  titleFontSize=14,
  labelFontSize=14,
  symbolSize=200
)
us_map
```

Out[32]:

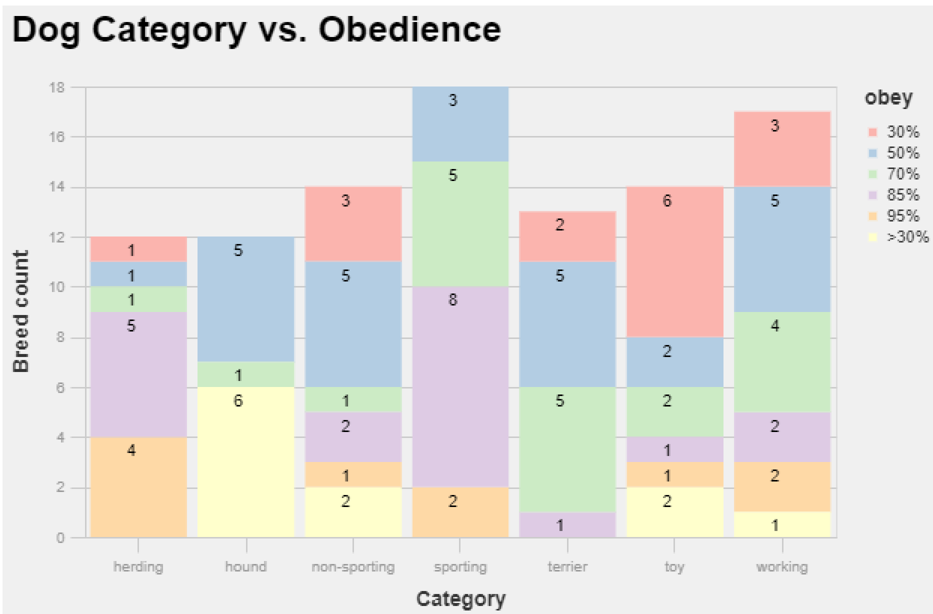


Stacked Bar Charts

In [33]:

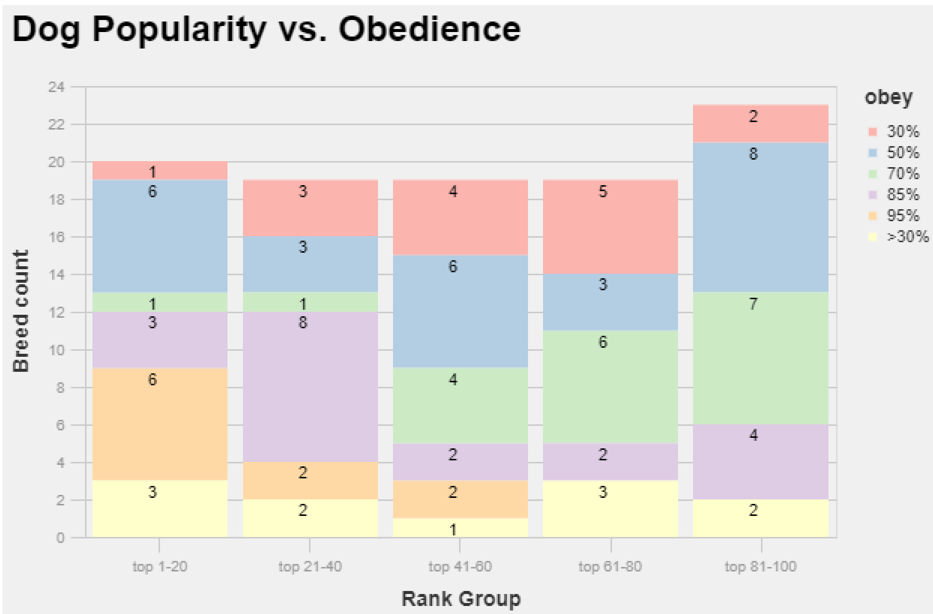
```
stacked_bar_category = alt.Chart(df).mark_bar().encode(
    x=alt.X('category:N', title='Category', axis=alt.Axis(labelAngle=0)),
    y=alt.Y('sum(id):Q',title='Breed count'),
    color=alt.Color('obey:N', scale=alt.Scale(scheme='pastel1'))
    ).properties(
    width=500,
    height=300,
    title='Dog Category vs. Obedience '
    )

text = alt.Chart(df).mark_text(dx=-5, dy=10, color='black').encode(
    x=alt.X('category:N'),
    y=alt.Y('sum(id):Q',stack='zero'),
    detail='obey:N',
    text=alt.Text('sum(id):Q')
)
stacked_bar_category + text
```

Out[33]:



In [34]:
```
stacked_bar_popular = alt.Chart(df).mark_bar().encode(
    x=alt.X('Group:N', title='Rank Group', axis=alt.Axis(labelAngle=0)),
    y=alt.Y('sum(id):Q',title='Breed count'),
    color=alt.Color('obey:N', scale=alt.Scale(scheme='pastel1'))
    ).properties(
    width=500,
    height=300,
    title='Dog Popularity vs. Obedience'
    )

text = alt.Chart(df).mark_text(dx=-5, dy=8, color='black').encode(
    x=alt.X('Group:N'),
    y=alt.Y('sum(id):Q',stack='zero'),
    detail='obey:N',
    text=alt.Text('sum(id):Q')
)
stacked_bar_popular + text
```

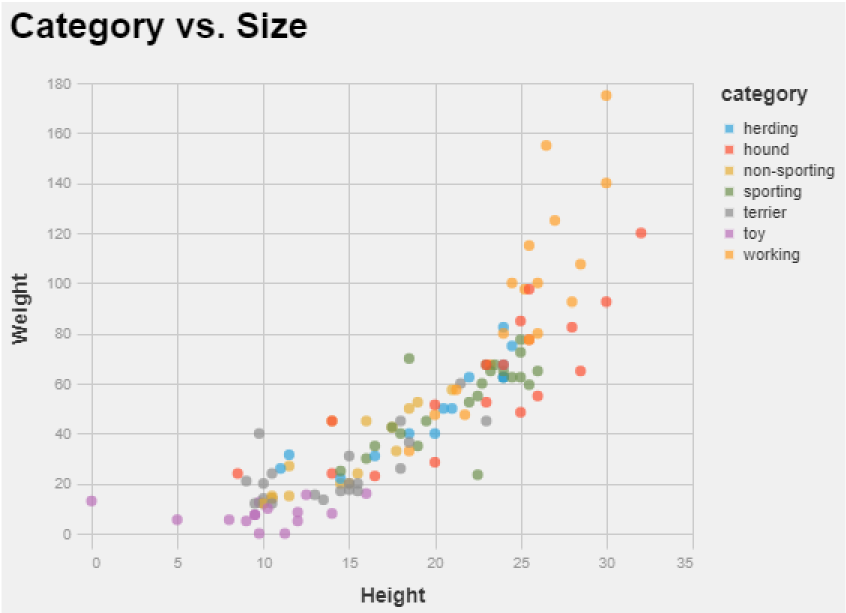
Out[34]:



Scatter plot

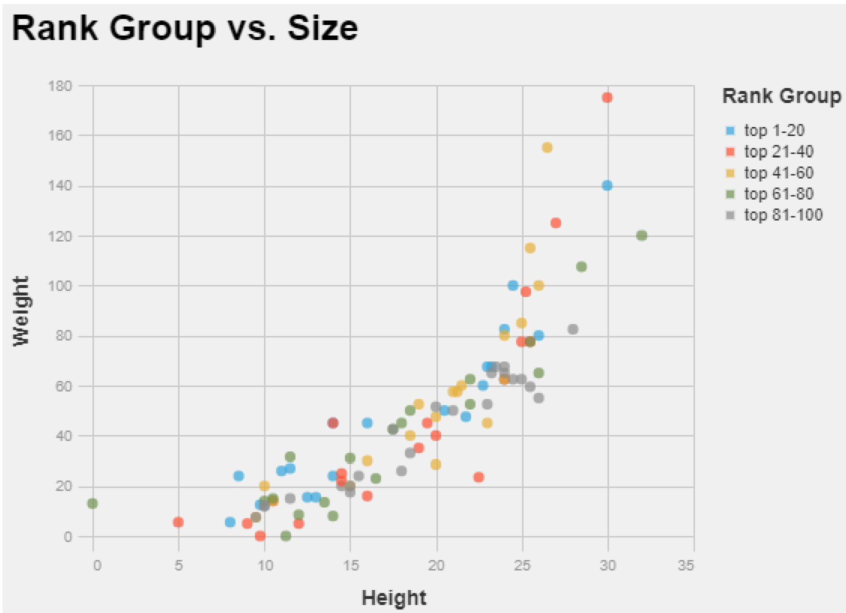
In [35]:
```
selection = alt.selection_single(encodings=['color'])
colorCondition = alt.condition(selection,'category:N',alt.value('lightgrey'), title='category')
scatter_size = alt.Chart(sort_df).mark_circle(size=50).add_selection(selection).encode(
     x=alt.X('shoulder height (in):Q', title='Height', axis=alt.Axis(tickMinStep=2)),
     y=alt.Y('weight (lbs):Q', title='Weight', axis=alt.Axis(tickMinStep=10)),
     color=colorCondition,
     tooltip=['Breed:N','weight (lbs):Q','shoulder height (in):Q']
     ).properties(
     title='Category vs. Size'
     )
scatter_size
```

Out[35]:



In [36]:
```
selection = alt.selection(type='interval', encodings=['x','y'])
colorCondition = alt.condition(selection,'Group:N',alt.value('lightgrey'), title='Rank Group')
scatter_size = alt.Chart(df).mark_circle(size=50).add_selection(selection).encode(
     x=alt.X('shoulder height (in):Q', title='Height', axis=alt.Axis(tickMinStep=2)),
     y=alt.Y('weight (lbs):Q', title='Weight', axis=alt.Axis(tickMinStep=10)),
     color=colorCondition,
     tooltip=['Breed:N','weight (lbs):Q','shoulder height (in):Q']
     ).properties(
     title='Rank Group vs. Size'
     )
scatter_size
```

Out[36]:



Linear regression to determine predictors of popularity

Using all the different characteristics of the dog breed that we have, we are interested in understanding which factors have the biggest impact of dog popularity.

I don't believe that linear regression is covered under the required courses for Mileston I/II. Because linear regression and other supervised learning techniques were taught during SIADS 524: Supervised Learning which is not a prerequisite for this course, this task is currently out of scope but could be tackled during future courses/analysis.

In [ ]:

```
```