Data 550: Data Visualization I

Lecture 4: Visualizing Distributions

Dr. Irene Vrbik

University of British Columbia Okanagan

https://github.com/ubco-mds-2022/Data-550

Introduction

- We have now learned how to create visualization using various graphical markings, channels and facets.
- Today we'll look at how to visualize distributions of data.
- While we have discussed some general guidelines on creating *effective* visualizations, we will dive deeper into when it is appropriate to use certain charts instead of others.
- This interactive flow chart serves as a useful reference finding the most appropriate graph for your data.

https://github.com/ubco-mds-2022/Data-550

Lecture learning goals

By the end of the lecture you will be able to:

- Select an appropriate distribution plot for the data.
- Create density plots to compare distributions.
- Create boxplots to compare many distributions.

https://github.com/ubco-mds-2022/Data-550

Setup

```
import altair as alt
import pandas as pd

movies = pd.read_csv("https://irene.vrbik.ok.ubc.ca/data/movies.csv")
```

```
movies.head()
```

	id	title	runtime	budget	revenue	genres	countries	vote_average	vote_loved_it	vot
0	12	Finding Nemo	100	94000000.0	9.403355e+08	Animation	United States of America	3.86	0.322	338
1	22	Pirates of the Caribbean: The Curse of the Bla...	143	140000000.0	6.550112e+08	Fantasy	United States of America	3.81	0.320	363
2	35	The Simpsons Movie	87	75000000.0	5.270689e+08	Animation	United States of America	3.44	0.155	862
3	58	Pirates of the Caribbean: Dead	151	200000000.0	1.065660e+09	Fantasy	United States of America	3.47	0.198	150

https://github.com/ubco-mds-2022/Data-550

	id	title	runtime	budget	revenue	genres	countries	vote_average	vote_loved_it	vot
		Man's Chest								
4	75	Mars Attacks!	106	70000000.0	1.013710e+08	Fantasy	United States of America	2.96	0.086	195

https://github.com/ubco-mds-2022/Data-550

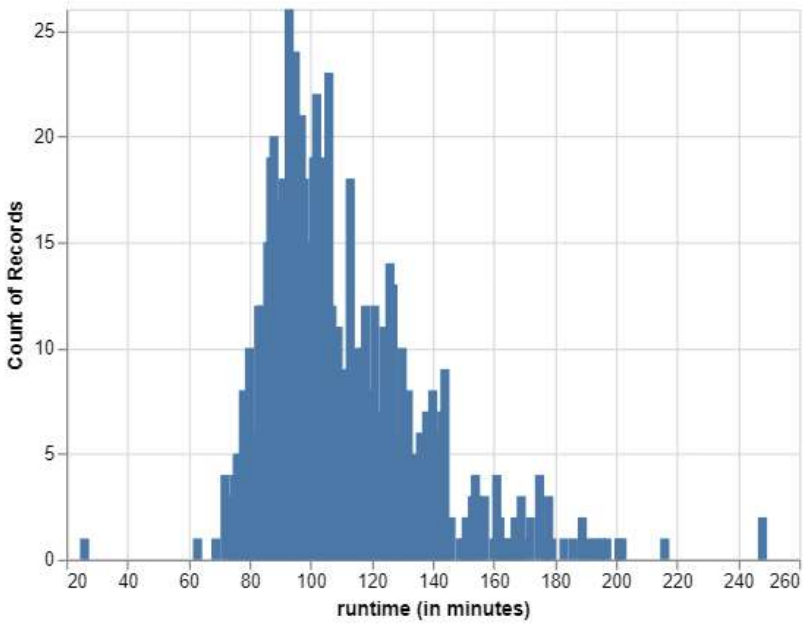
4

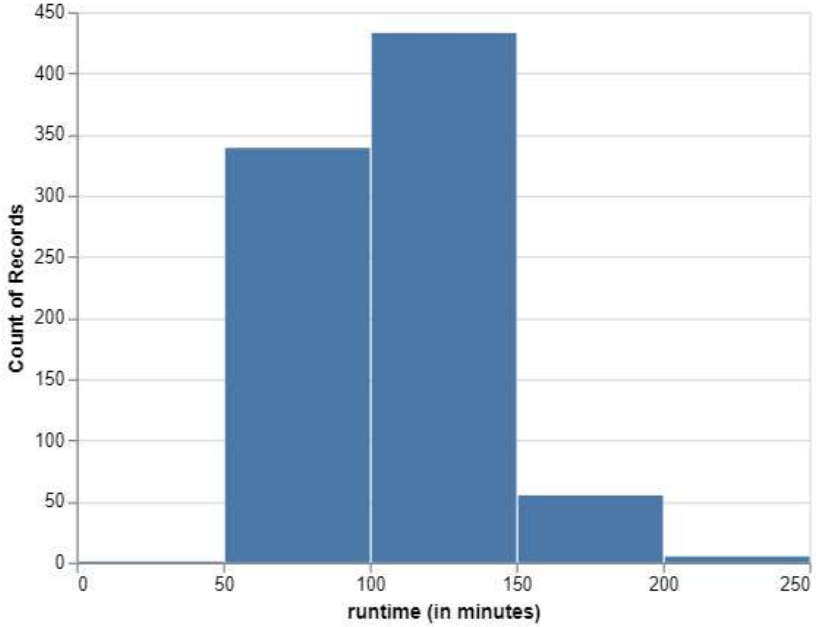
Visualizing data from one column

https://github.com/ubco-mds-2022/Data-550

Histogram

▶ Click to show the code



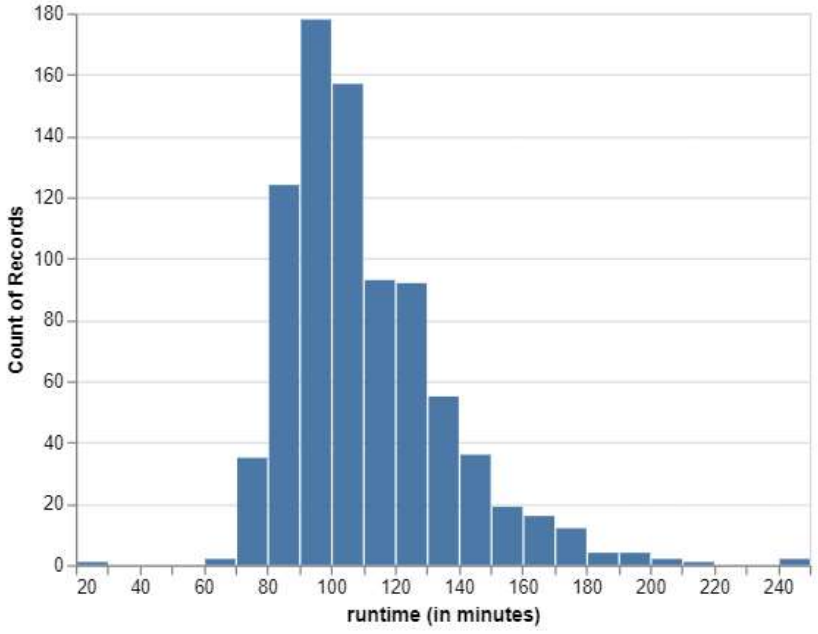


https://github.com/ubco-mds-2022/Data-550

Histogram (cont'd)

Let's make a histogram using the runtime of the movies.

```
alt.Chart(movies).mark_bar().encode(
    alt.X('runtime', bin=alt.Bin(maxbins=30), title='runtime (in minutes)'),
    alt.Y('count()'))
```



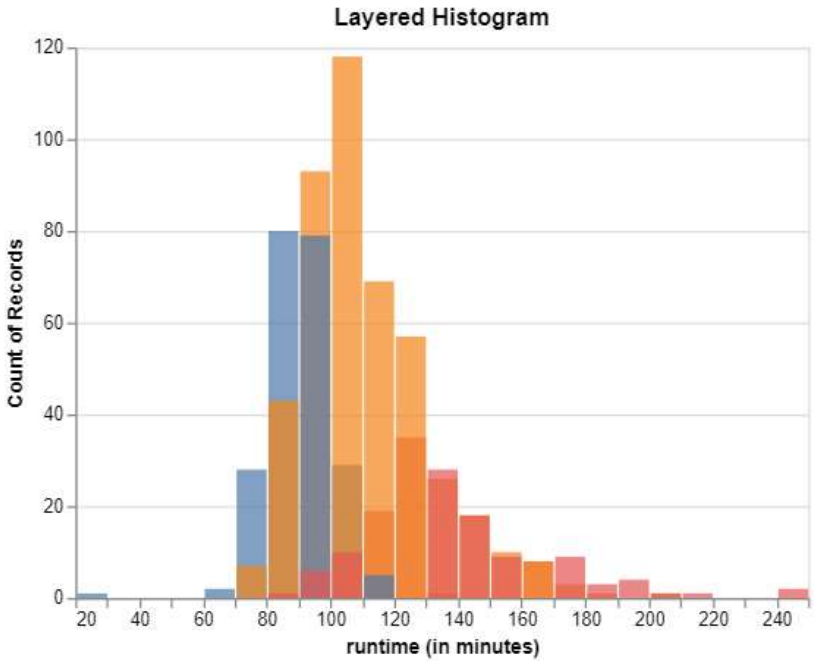
- *Do the runtimes differ between genres?*
- *Do the runtimes differ between countries?*

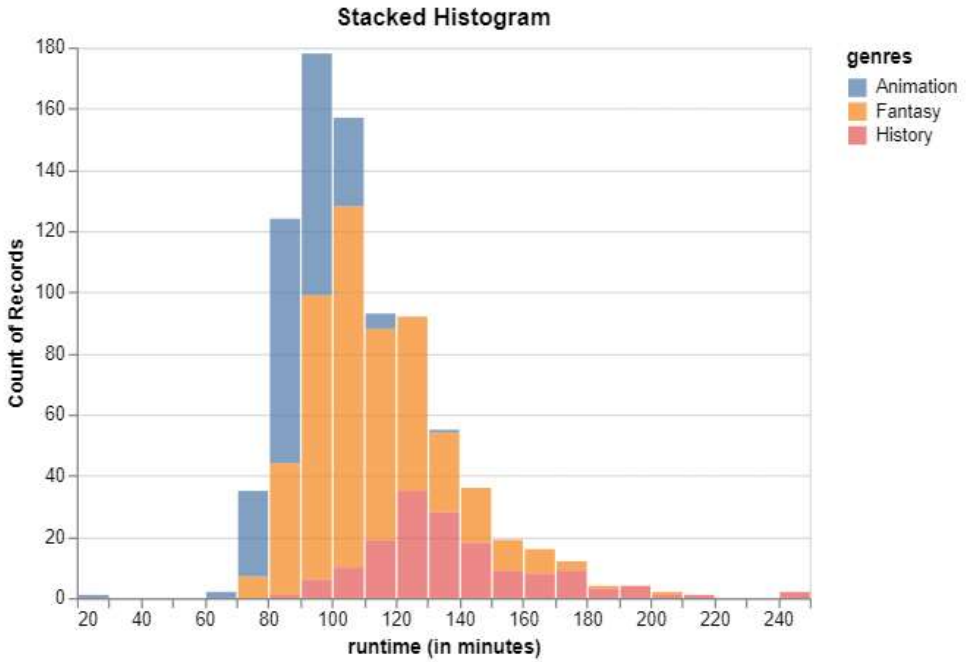
https://github.com/ubco-mds-2022/Data-550

Layered/Stacked Histogram

Let’s make a histogram using the runtime of the movies.

▶ Click to show the code





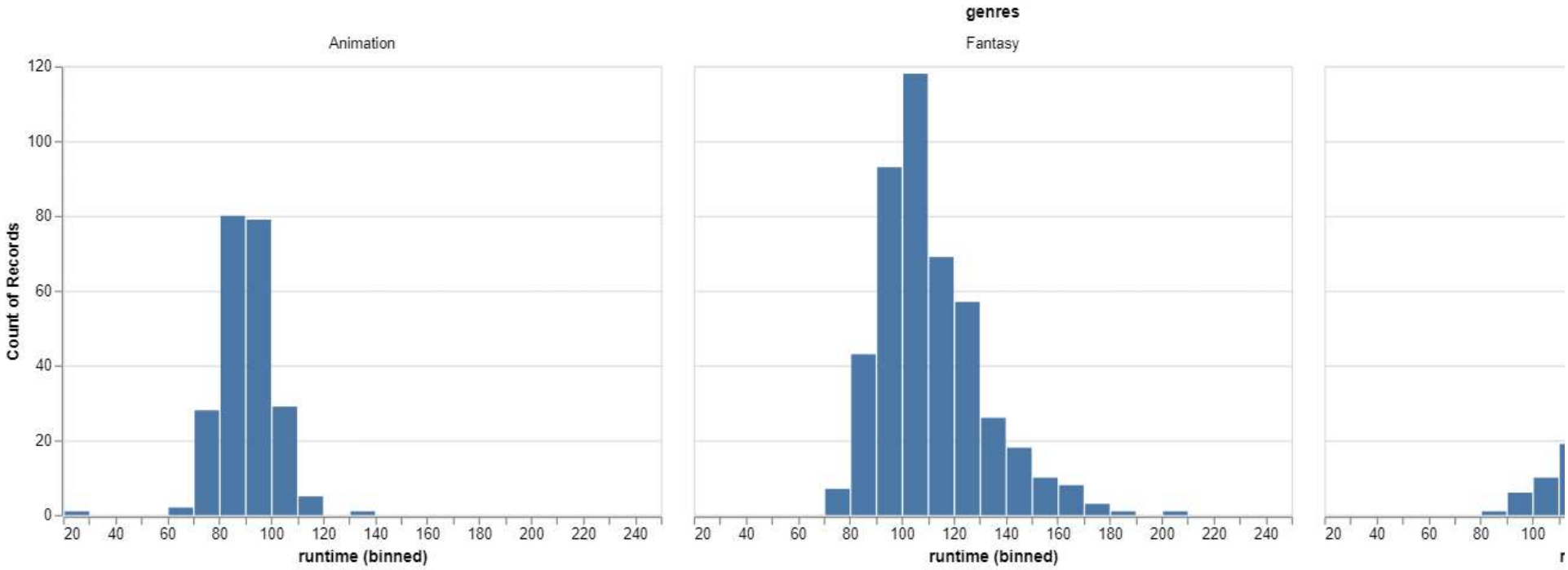
https://github.com/ubco-mds-2022/Data-550

jump head to a better way to visualize this data

Runtime according to genre

Are there differences in movie runtimes between genres?

```
alt.Chart(movies).mark_bar().encode(
    alt.X('runtime', bin=alt.Bin(maxbins=30)),
    y='count()').facet('genres')
```



https://github.com/ubco-mds-2022/Data-550

Density Plots

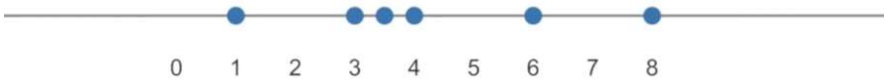
https://github.com/ubco-mds-2022/Data-550

Kernel Density Estimation

- Kernel Density Estimation (KDE) creates a smooth curve to help visualize the “shape” given a set of data.
- Both histograms and KDEs provide an estimate of the underlying (unknown) probability density function.
- You can think of it as a continuous replacement for the discrete histogram.
- Like a probability density function, the area under our KDE curve should equal 1.

https://github.com/ubco-mds-2022/Data-550

Construction of KDE



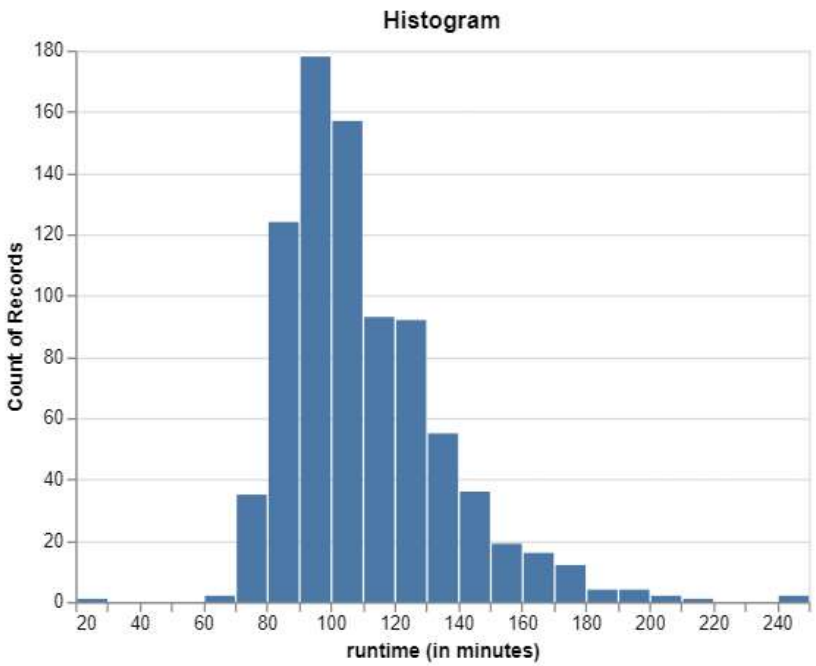
From Kimberly Fesse's KDE video

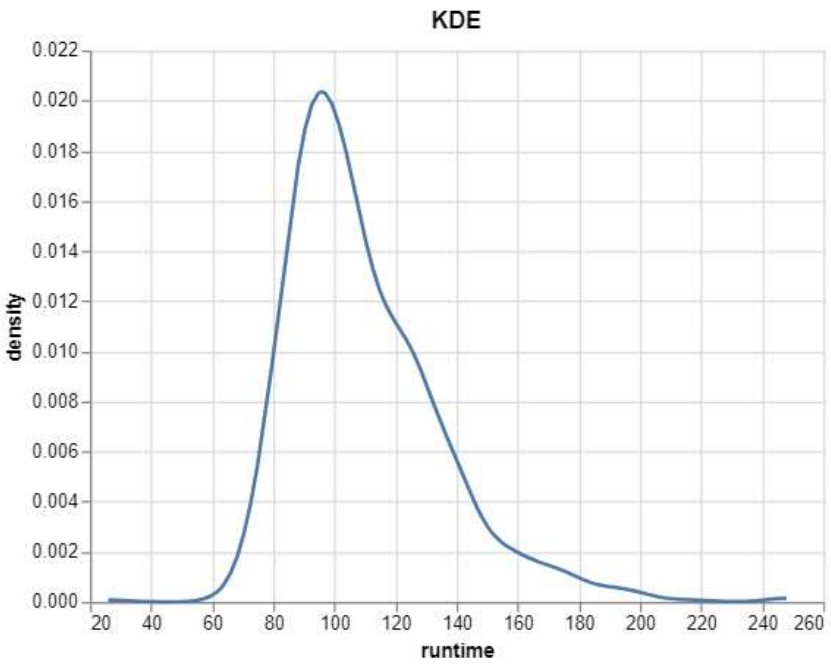
- center a Gaussian kernel[1] on each data point
- add them together
- the resulting curve is our KDE which serves as a good guess for the underlying probability density function

jump to groupby example

https://github.com/ubco-mds-2022/Data-550

et voilà!





https://github.com/ubco-mds-2022/Data-550

KDEs in Altair

To create a density plot in Altair, we need to:

1. Place the bell-shaped kernels an add them together <– *this is done using* `transform_density()`
2. Plot a line or area mark for the newly calculated sum of kernels.

Note: The creation of density plots is a lot less involved using `ggplot`.

https://github.com/ubco-mds-2022/Data-550

Data Transformations

Partial table from Altair documentation: Data Transformation

Transform	Method	Description
Aggregate Transforms	`transform_aggregate()`	Create a new data column by aggregating an existing column.
Bin transforms	`transform_bin()`	Create a new data column by binning an existing column.
Calculate Transform	`transform_calculate()`	Create a new data column using an arithmetic calculation on an existing column.
Density Transform	`transform_density()`	Create a new data column with the kernel density estimate of the input.
Filter Transform	`transform_filter()`	Select a subset of data based on a condition.

...

https://github.com/ubco-mds-2022/Data-550

Transform Options

See full list of transform options here

Property	Type	Description
as	array(any)	The output fields for the sample value and corresponding density estimate. **Default value:** `["value", "density"]`
bandwidth	`number`	The bandwidth (standard deviation) of the Gaussian kernel. If unspecified or set to zero, the bandwidth value is automatically estimated from the input data using Scott's rule.
cumulative	`boolean`	A boolean flag indicating whether to produce density estimates (false) or cumulative density estimates (true). **Default value:** `false`
counts	`boolean`	A boolean flag indicating if the output values should be probability estimates (false) or smoothed counts (true). **Default value:** `false`
groupby	array(**`FieldName`**)	The data fields to group by. If not specified, a single group containing all data objects will be used.
...		

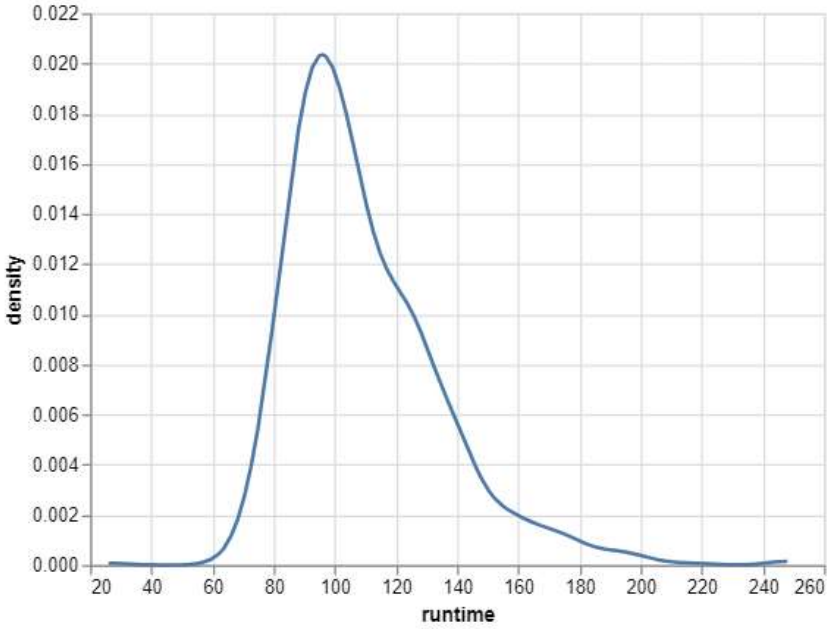
https://github.com/ubco-mds-2022/Data-550

Runtime KDE

Quiz material up to this

The code for creating the KDE plot on this slide:

```
alt.Chart(movies).transform_density(
    'runtime', # which data column we want to use for the calculation
    as_=['runtime', 'density'] # creates a new data column called "density"
).mark_line().encode(
    x='runtime',
    y='density:Q') # recall we have to specify the data type (quantitative)
```

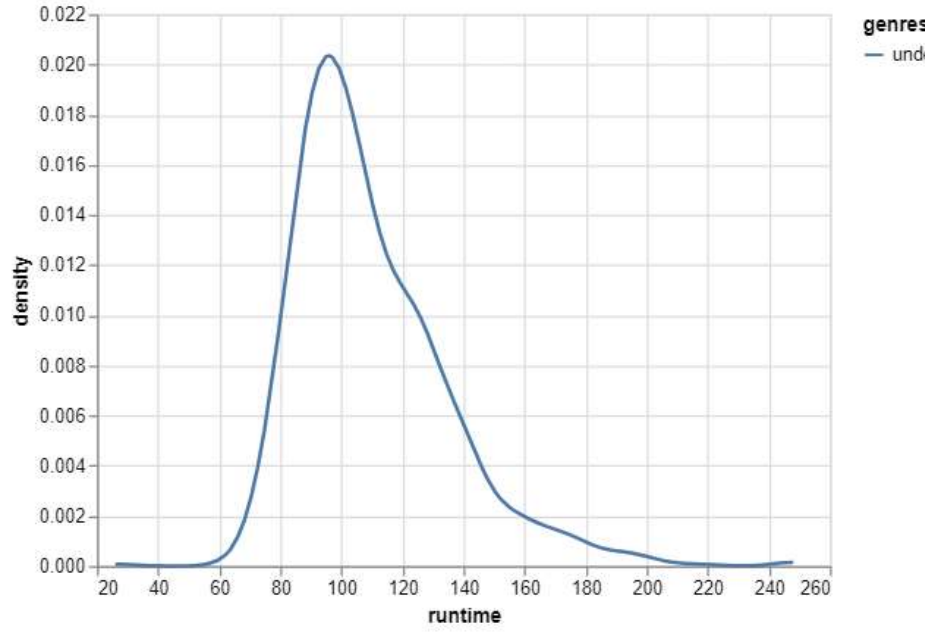


https://github.com/ubco-mds-2022/Data-550

Genre Runtime KDE (wrong)

If we want to create a KDE for each genre, the following will *not* work since `density` is calculate using all of the `runtimes`.

```
alt.Chart(movies
).transform_density(
    'runtime',
    as_=['runtime', 'density']
).mark_line().encode(
    x='runtime',
    y='density:Q',
    color='genres')
```



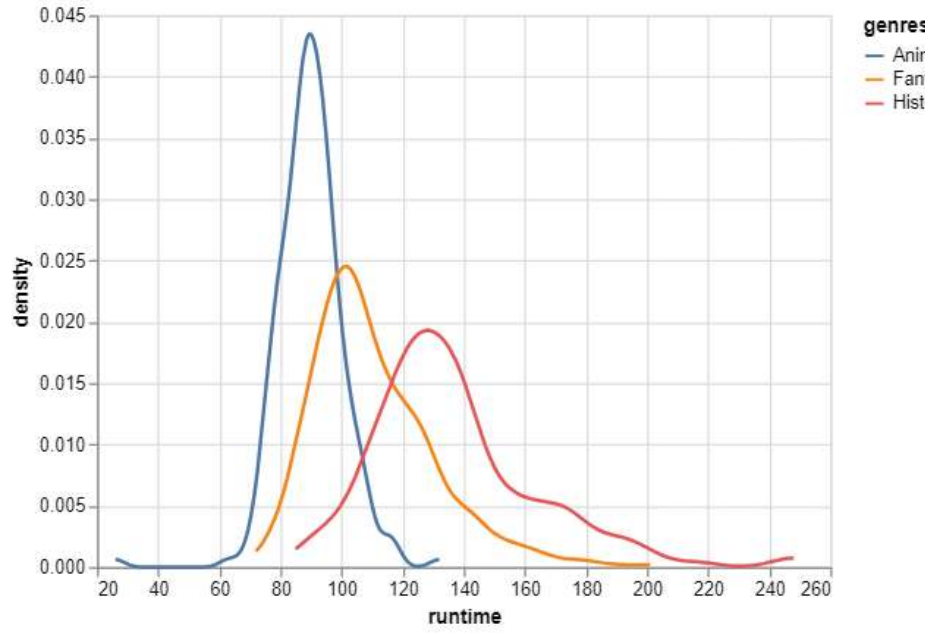
Recall how KDEs are constructed

https://github.com/ubco-mds-2022/Data-550

Genre Runtime KDE (lines)

To remedy this we need to *groupby* `genres` in the density transformation.

```
alt.Chart(movies).transform_densit
    'runtime',
     groupby=['genres'],
    as_=['runtime', 'density']
).mark_line().encode(
    x='runtime',
    y='density:Q',
    color='genres')
```

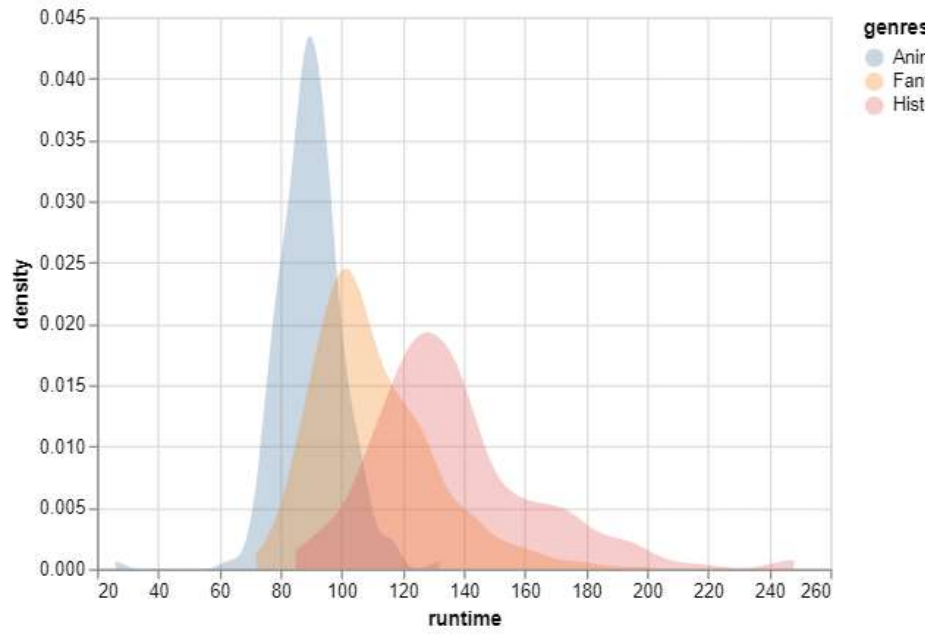


https://github.com/ubco-mds-2022/Data-550

Genre Runtime KDE (area)

Unlike layered/stacked histograms, layered density plot effectively convey the differences between runtimes of movies from different genres.

```
alt.Chart(movies).transform_densit
    'runtime',
     groupby=['genres'],
    as_=['runtime', 'density']
).mark_area(opacity=0.3).encode(
    x='runtime',
    y='density:Q',
    color='genres')
```



https://github.com/ubco-mds-2022/Data-550

Comparing Runtimes between Countries

Do the runtimes differ between countries?

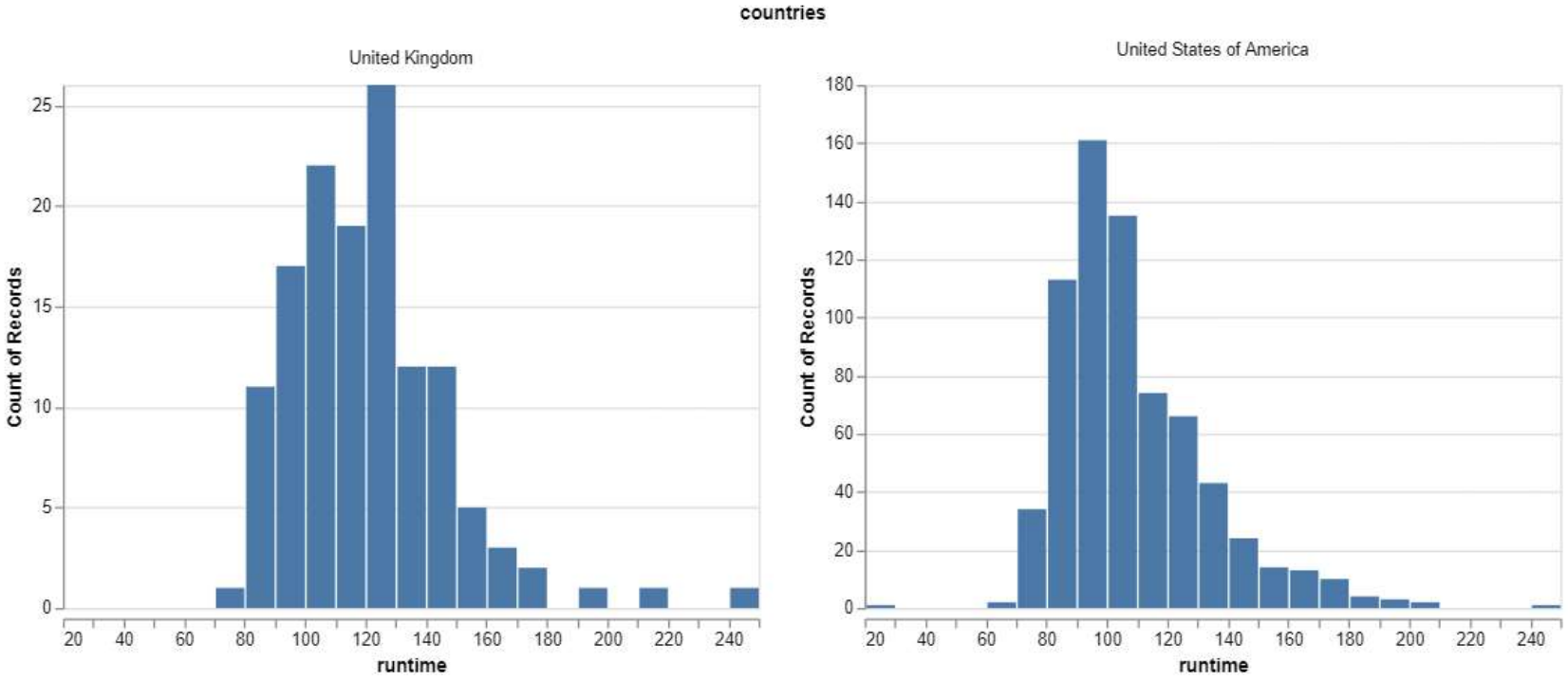
```
alt.Chart(movies).mark_bar().encode(
    alt.X('runtime', bin=alt.Bin(maxbins=30), title = 'runtime'),
    y='count()').facet('countries')
```

https://github.com/ubco-mds-2022/Data-550

Resolved Facet

Do the runtimes differ between countries?

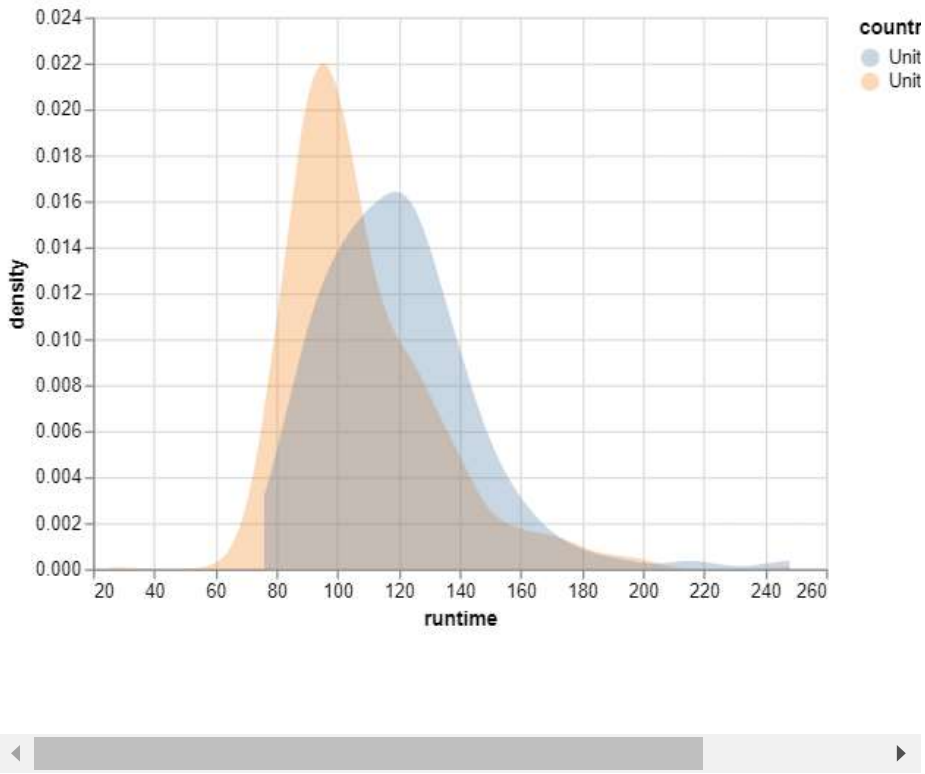
```
alt.Chart(movies).mark_bar().encode(
    alt.X('runtime', bin=alt.Bin(maxbins=30), title = 'runtime'),
    y='count()').facet('countries'
    ).resolve_scale(y='independent')
```



https://github.com/ubco-mds-2022/Data-550

Density Plots

```
(alt.Chart(movies)
.transform_density(
     'runtime',
     groupby=['countries'],
     as_=['runtime', 'density'])
 .mark_area(opacity=0.3).encode(
     x='runtime',
     y='density:Q',
     color='countries'))
```



If you wrap the chart in parentheses you can move methods to new lines without a syntax

https://github.com/ubco-mds-2022/Data-550

Transform Options (cont'd)

See complete table here

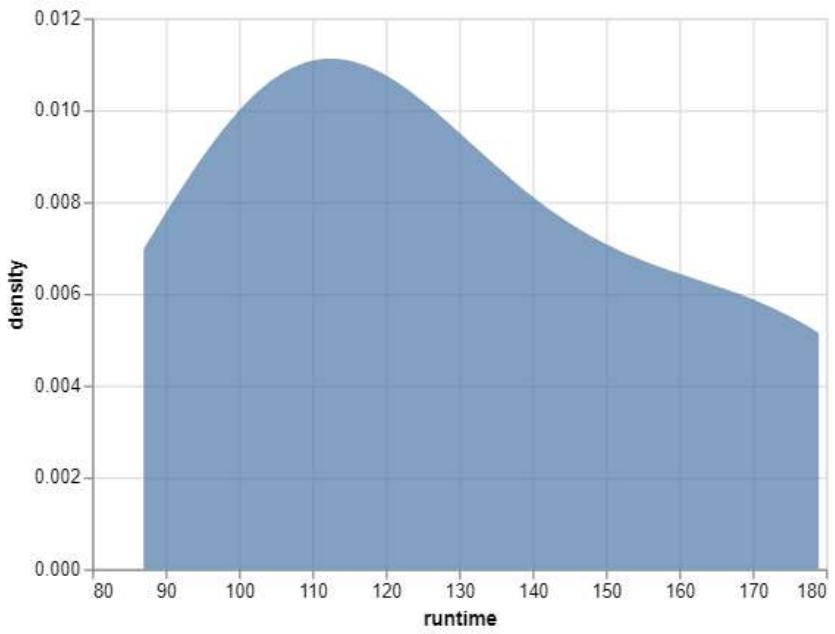
Property	Type	Description
maxsteps	`number`	The maximum number of samples to take along the extent domain for plotting the density. **Default value:** `200`
minsteps	`number`	The minimum number of samples to take along the extent domain for plotting the density. **Default value:** `25`
steps	`number`	The exact number of samples to take along the extent domain for plotting the density. If specified, overrides both minsteps and maxsteps to set an exact number of uniform samples. Potentially useful in conjunction with a fixed extent to ensure consistent sample points for stacked densities

https://github.com/ubco-mds-2022/Data-550

Warning: small data sets

Can you guess how many data points went into this visualization

▶ Click to show the code

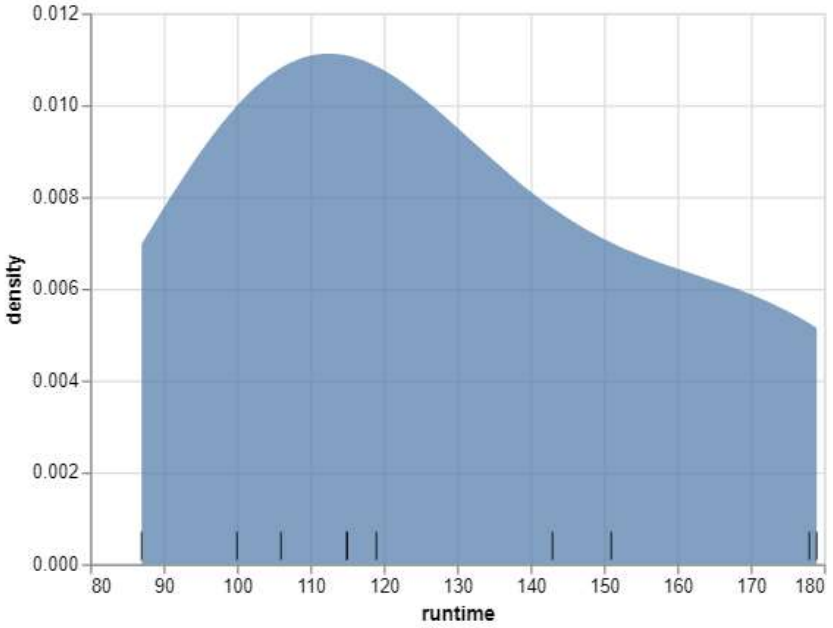


https://github.com/ubco-mds-2022/Data-550

25

“Transparent” plot

```
ticks = alt.Chart(movies[:10]
).mark_tick(
  color='black', yOffset=140
).encode(x='runtime')
density + ticks
```



To be more transparent on the number of data points in our sample, we can always plot ticks to indicate the for each observation (offset to appear on the axis rather than the center of the graph)

https://github.com/ubco-mds-2022/Data-550

Comparing Multiple Distributions

https://github.com/ubco-mds-2022/Data-550

Data

Which genres are the highest grossing world-wide?

```
from vega_datasets import data

movies_extended = data.movies().dropna(subset=["Major_Genre"])
movies_extended.head()
```

	Title	US_Gross	Worldwide_Gross	US_DVD_Sales	Production_Budget	Release_Date	M
1	First Love, Last Rites	10876.0	10876.0	NaN	300000.0	Aug 07 1998	F
2	I Married a Strange Person	203134.0	203134.0	NaN	250000.0	Aug 28 1998	N
3	Let's Talk About Sex	373615.0	373615.0	NaN	300000.0	Sep 11 1998	N

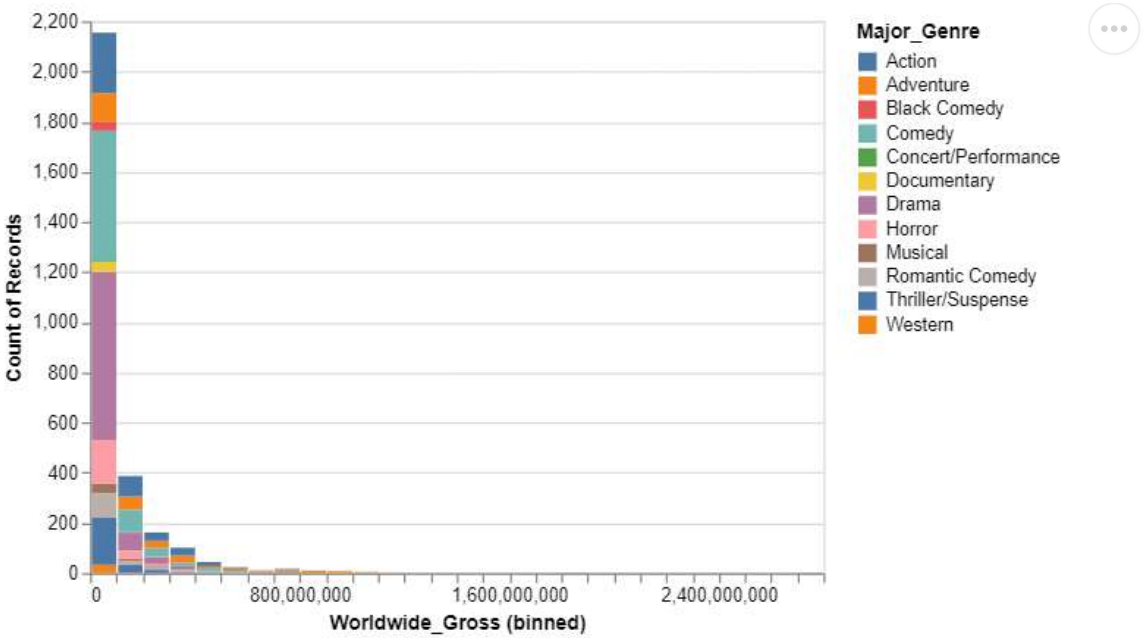
https://github.com/ubco-mds-2022/Data-550

	Title	US_Gross	Worldwide_Gross	US_DVD_Sales	Production_Budget	Release_Date	M
4	Slam	1009819.0	1087521.0	NaN	1000000.0	Oct 09 1998	F
7	Foolish	6026908.0	6026908.0	NaN	1600000.0	Apr 09 1999	F

https://github.com/ubco-mds-2022/Data-550

Stacked histogram (fail)

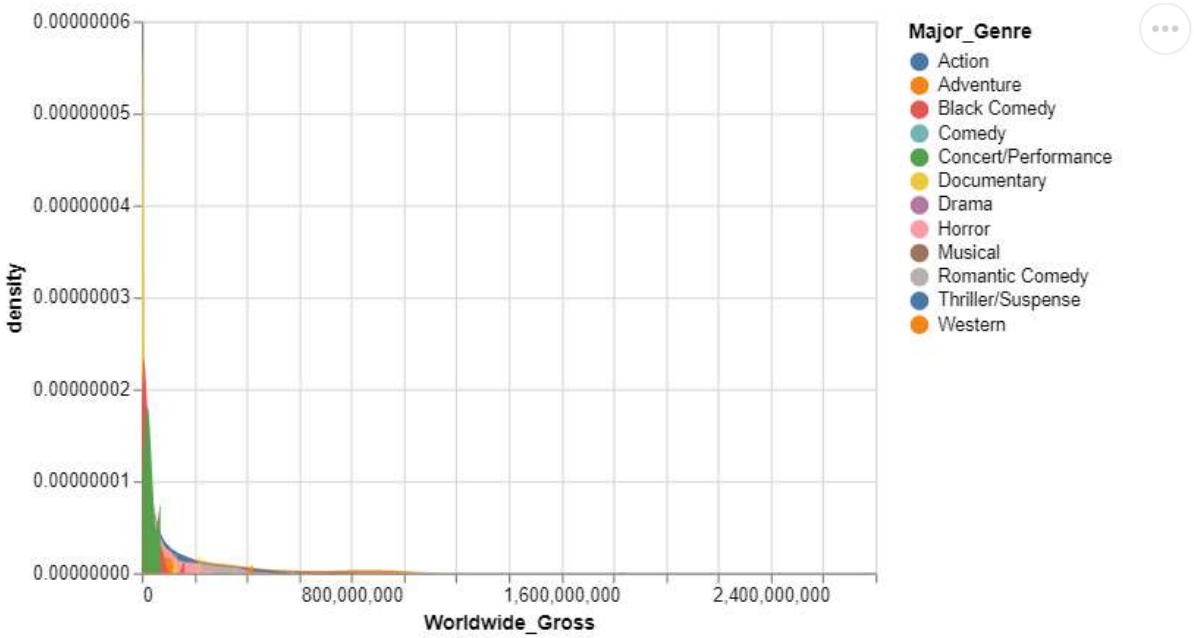
- Show the code



https://github.com/ubco-mds-2022/Data-550

Layered density plot (fail)

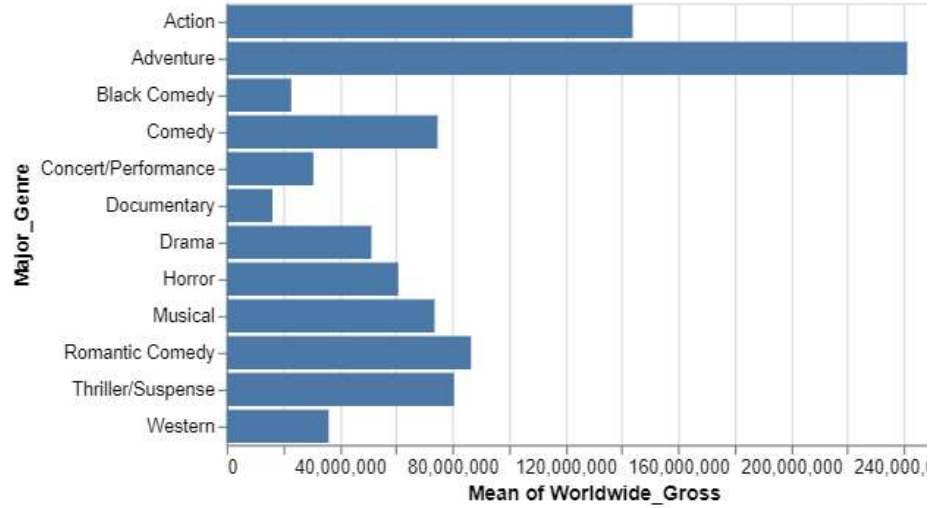
- Show the code



https://github.com/ubco-mds-2022/Data-550

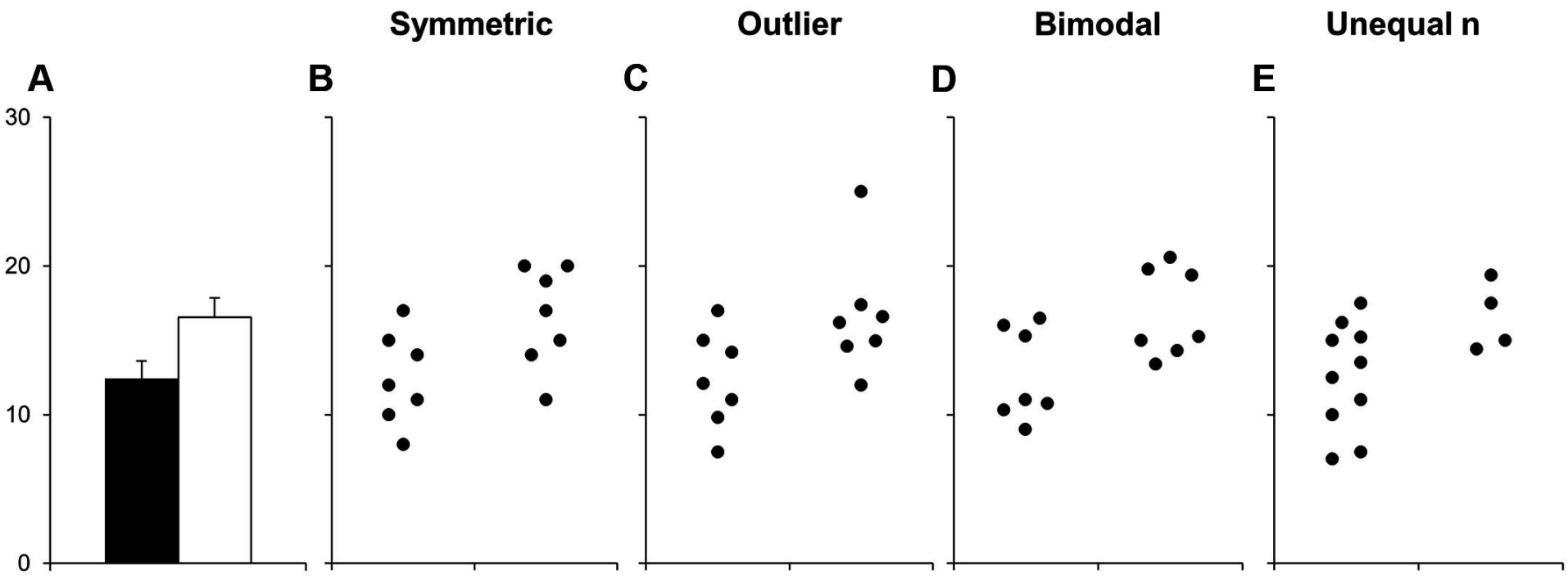
Bar chart

```
alt.Chart(movies_extended).mark_ba
    alt.X('mean(Worldwide_Gross)')
    alt.Y('Major_Genre'))
```



- Plotting one value? *barplot* could work
- Con: hides the variation in the data, which could lead us to arrive at incorrect conclusions

https://github.com/ubco-mds-2022/Data-550



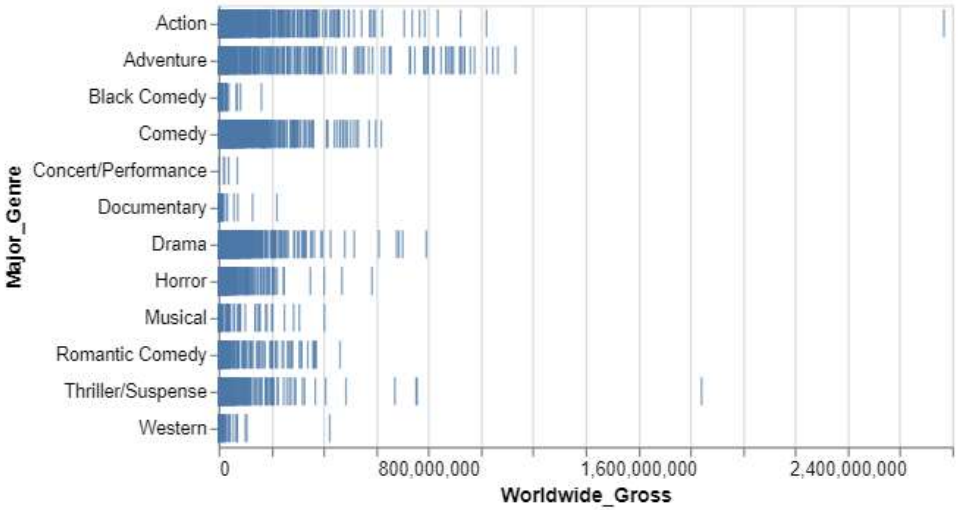
Test		p value		
T-test: Equal var.	0.035	0.050	0.026	0.063
T-test: Unequal var.	0.035	0.050	0.026	0.035
Wilcoxon	0.054	0.073	0.128	0.103

From Beyond Bar and Line Graphs

https://github.com/ubco-mds-2022/Data-550

Rug plot

```
alt.Chart(movies_extended).mark_tick().encode(
    alt.X('Worldwide_Gross'),
    alt.Y('Major_Genre'))
```



https://github.com/ubco-mds-2022/Data-550

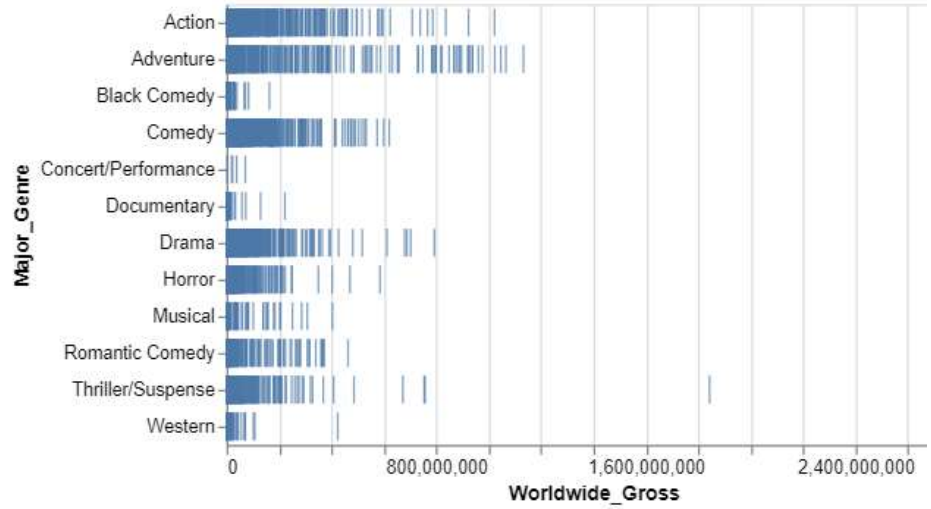
Tooltips

- Interactive `tooltips` can be enabled to facilitate exploration.
- The `tooltip` encoding channel determines tooltip text to show when a user moves the mouse cursor over a mark.
- Let's add a tooltip encoding for the `Title` field, then investigate which movies are grossed highly above the rest.
- Try the following code in lab to answer *What are the highest grossing movies?*

https://github.com/ubco-mds-2022/Data-550

Tooltip Demo

```
alt.Chart(movies_extended).mark_ti
    alt.X('Worldwide_Gross'),
    alt.Y('Major_Genre'),
    alt.Tooltip('Title:N'))
```



While this doesn't function well in my slides, if you view it in a notbook you will that when you hover over a marking the corresponding title will appear.

https://github.com/ubco-mds-2022/Data-550

Comments

- Although this visualization is useful in getting information about the individual movies, this plot is *saturated*
- It is hard to tell exactly how many data points there are in the areas that are completely blue.
- While this could be alleviated somewhat by setting a lower opacity, since these points are so densely positioned it is more effective to use colour for representing the counts.

https://github.com/ubco-mds-2022/Data-550

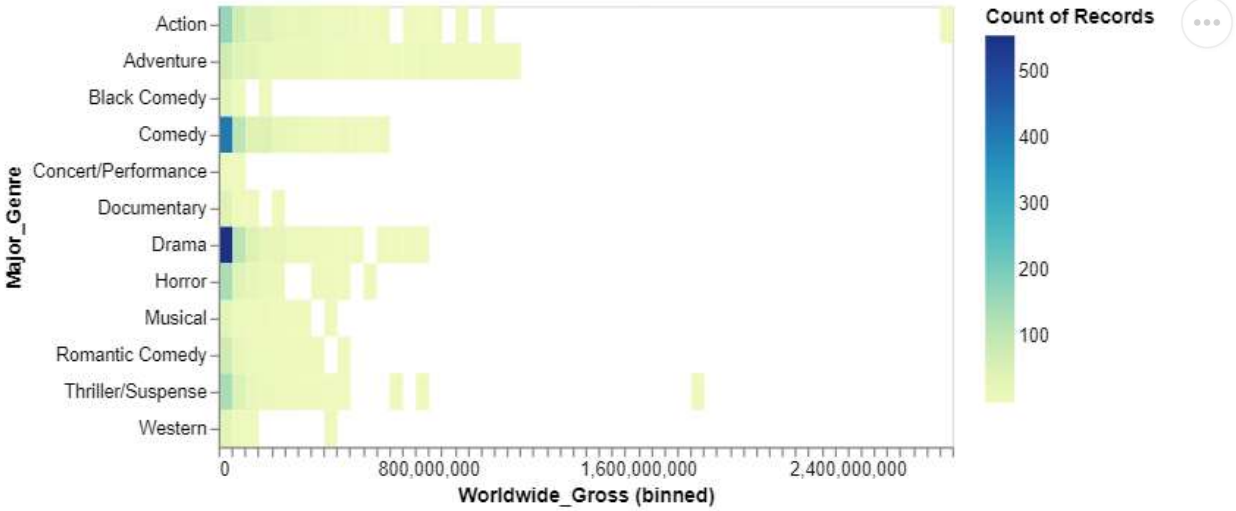
Heatmaps

- Heatmaps can compare multiple distributions without saturation
- A heatmap of the number of observations is very similar to a histogram, but the count is mapped to the colour instead of to the height on the y-axis.

https://github.com/ubco-mds-2022/Data-550

Heatmap Example

```
(alt.Chart(movies_extended).mark_rect().encode(
    alt.X('Worldwide_Gross', bin=alt.Bin(maxbins=100)),
    alt.Y('Major_Genre'),
    alt.Color('count()')))
```



https://github.com/ubco-mds-2022/Data-550

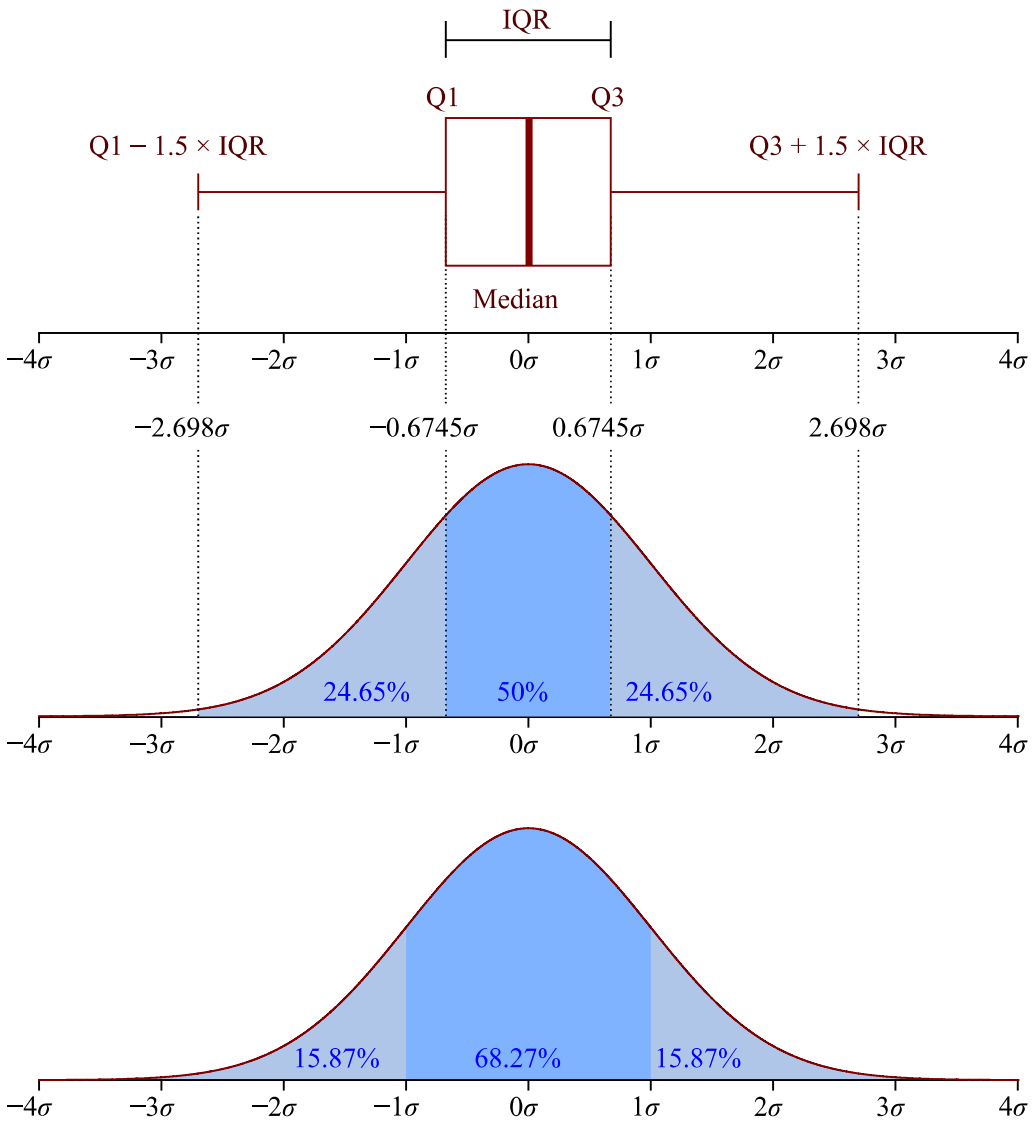
Boxplots

A box plot shows 5 summary statistics:

- the median in the middle and the lower and upper quartile at the edges
- the lines extending from the box are called whiskers and they can represent either:
  - the min and the max (the range) of our data.
  - the furthest points that are still within 1.5 x the IQR from the edges of the box.
- The interquartile range (IQR) is the distance between the edges of the box.

https://github.com/ubco-mds-2022/Data-550

39

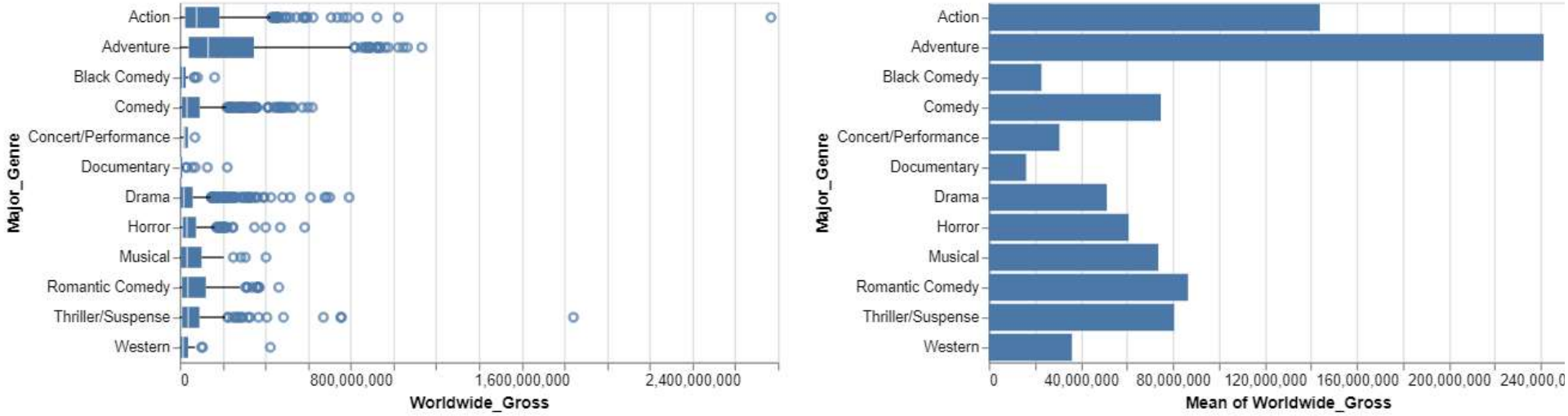


Jhguch at Wikimedia Commons

https://github.com/ubco-mds-2022/Data-550

Boxplot Example

```
1 bar = alt.Chart(movies_extended).mark_bar().encode(
2     alt.X('mean(Worldwide_Gross)'),
3     alt.Y('Major_Genre'))
4
5 box = alt.Chart(movies_extended).mark_boxplot().encode(
6     alt.X('Worldwide_Gross'),
7     alt.Y('Major_Genre'))
8
9 box | bar
```

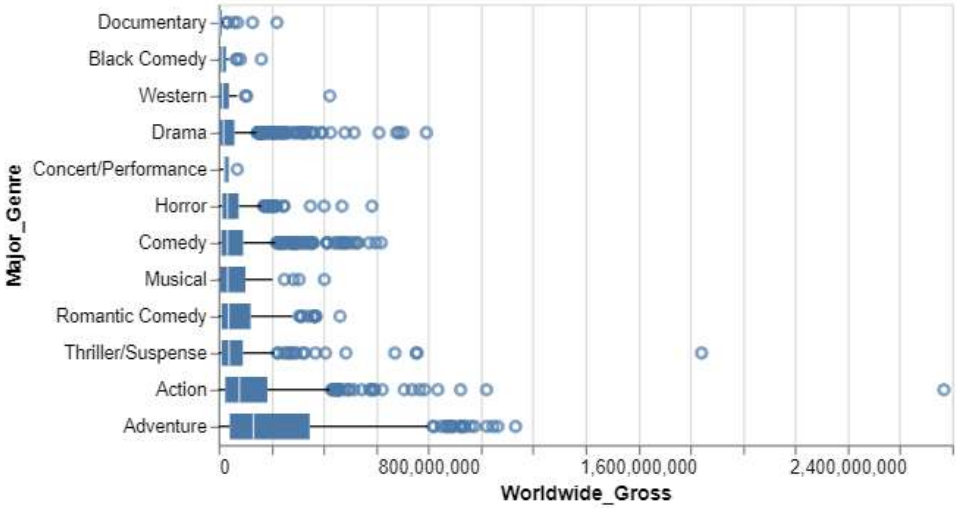


https://github.com/ubco-mds-2022/Data-550

41

Sorted Boxplot

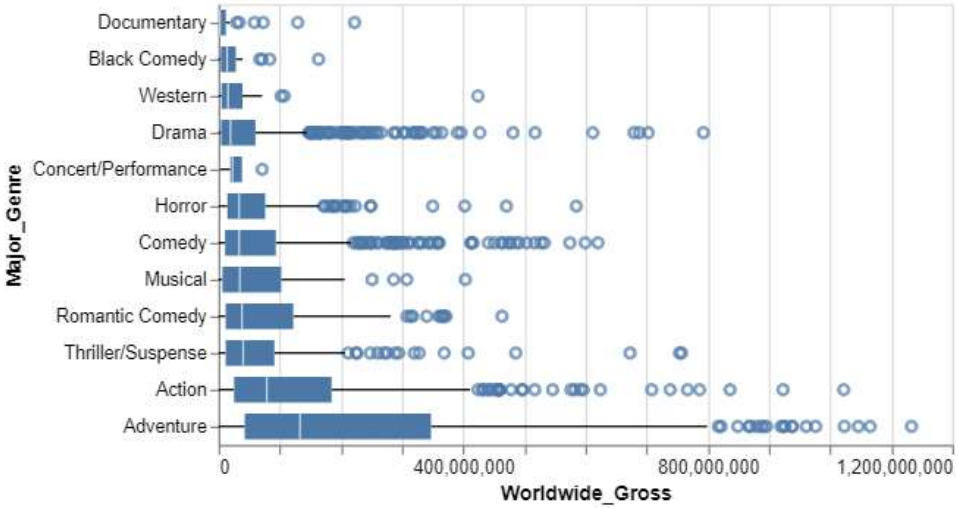
```
genre_order = movies_extended.groupby('Major_Genre')['Worldwide_Gross'].med
alt.Chart(movies_extended).mark_boxplot().encode(
    alt.X('Worldwide_Gross'),
    alt.Y('Major_Genre', sort=genre_order))
```



https://github.com/ubco-mds-2022/Data-550

Zooming

```
filtered_movies = movies_extended[movies_extended['Worldwide_Gross'] < 1_50
alt.Chart(filtered_movies).mark_boxplot().encode(
    alt.X('Worldwide_Gross'),
    alt.Y('Major_Genre', sort=genre_order))
```

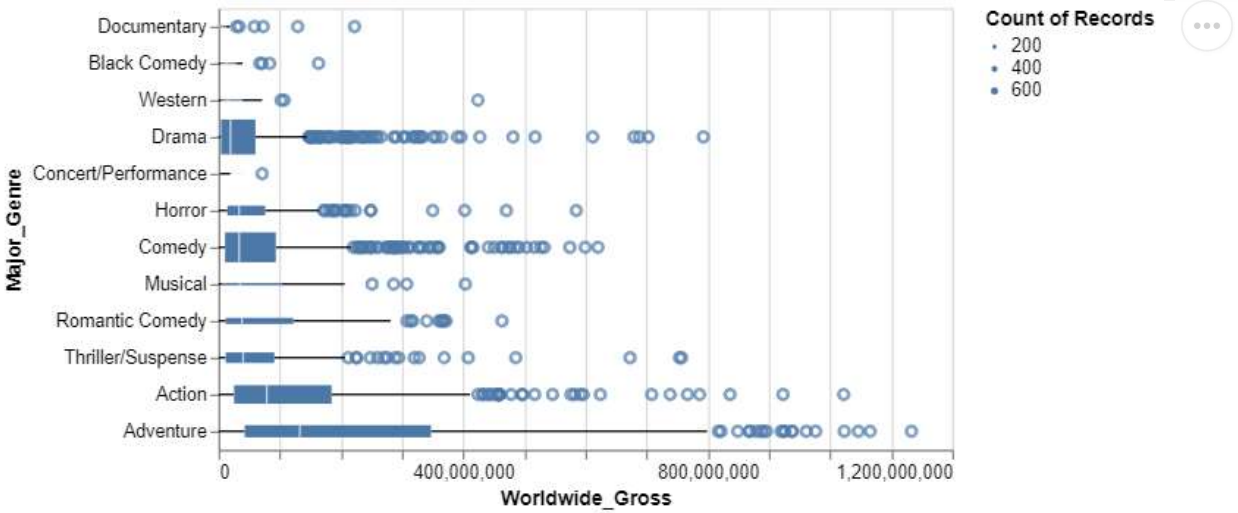


https://github.com/ubco-mds-2022/Data-550

Scaled Boxplots

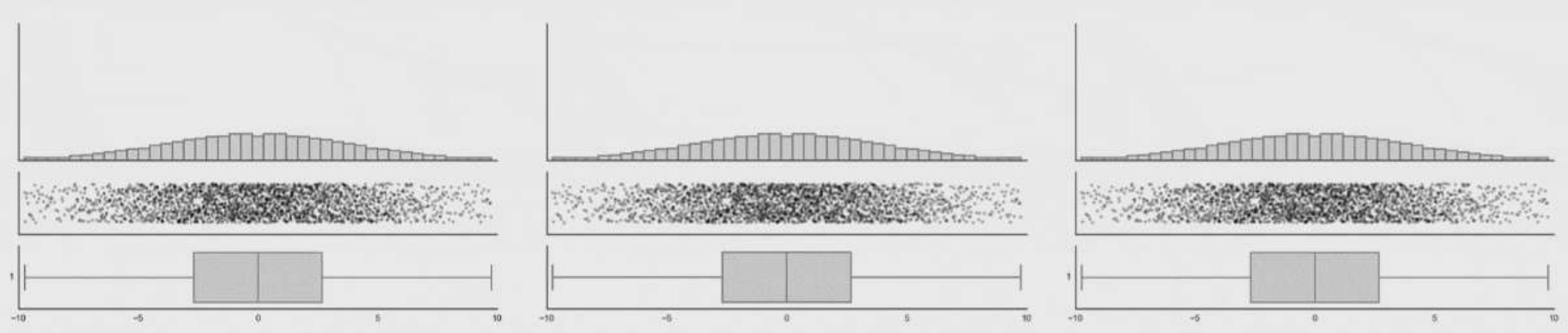
Boxplots can be scaled by the number of observations

```
alt.Chart(filtered_movies).mark_boxplot().encode(
    alt.X('Worldwide_Gross'),
    alt.Y('Major_Genre', sort=genre_order),
    alt.Size('count()'))
```



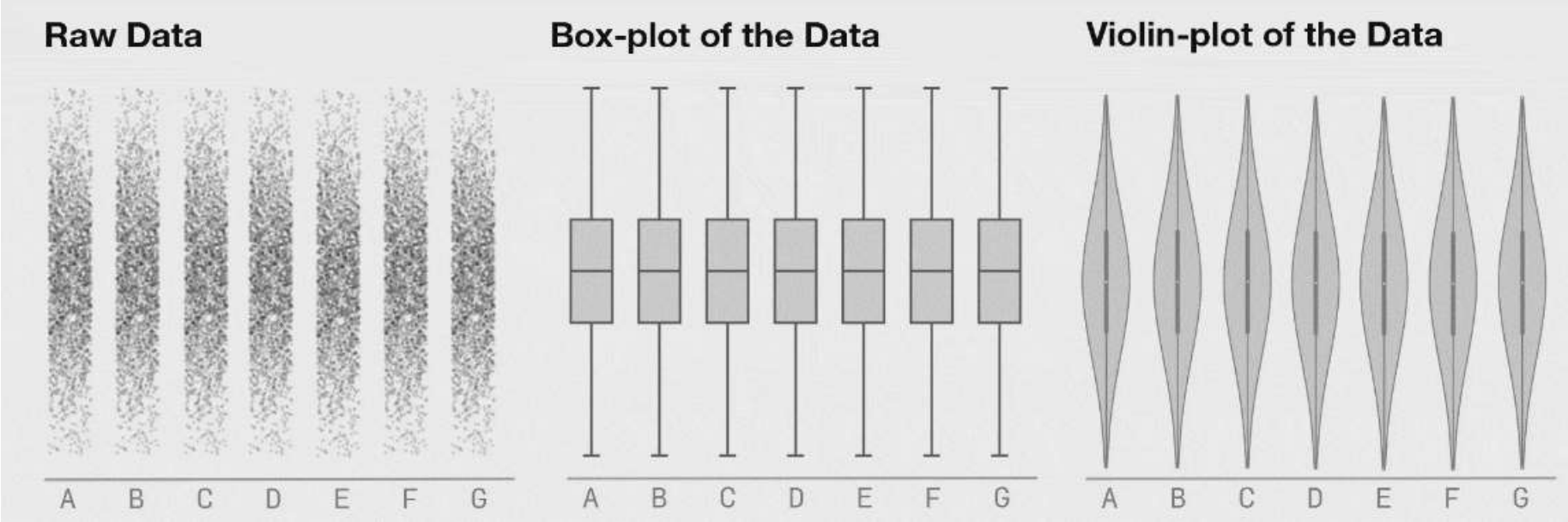
https://github.com/ubco-mds-2022/Data-550

The cons of boxplots



Source of images

https://github.com/ubco-mds-2022/Data-550



https://github.com/ubco-mds-2022/Data-550

https://github.com/ubco-mds-2022/Data-550