Question 🤔 (Answer at practicaldsc.org/q)

Remember that you can always ask questions anonymously at the link above!

How long does Homework 3 feel compared to Homework 2?

- A. Way shorter.
- B. Shorter.
- C. About the same.
- D. Longer.
- E. Way longer.

please answer!

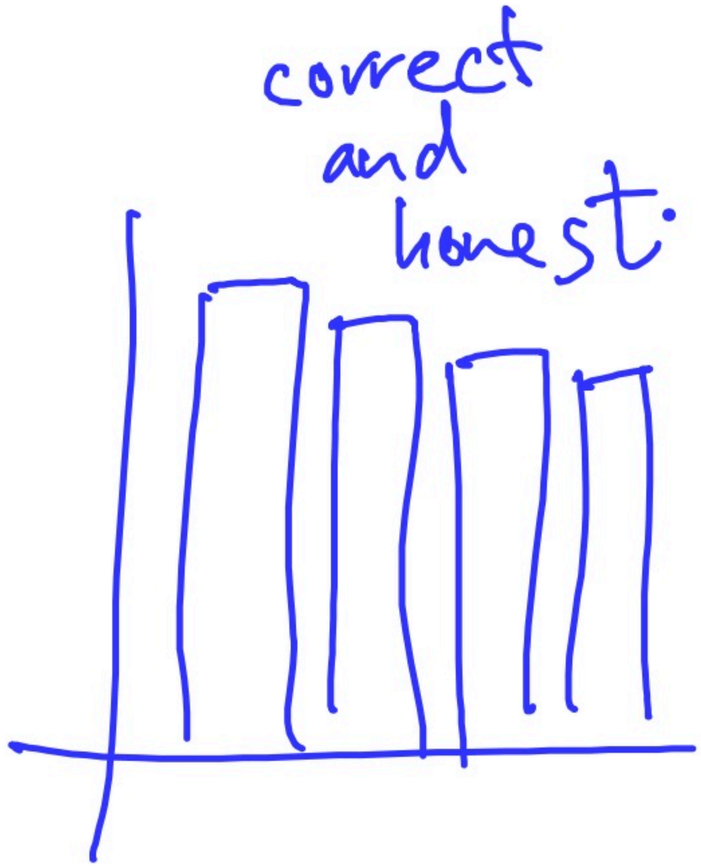
In [2]:
```
# Run this cell to perform the data cleaning steps we implemented last lecture.
def clean_term_column(df):
    return df.assign(
        term=df['term'].str.split().str[0].astype(int)
    )
def clean_date_column(df):
    return (
        df
        .assign(date=pd.to_datetime(df['issue_d'], format='%b-%Y'))
        .drop(columns=['issue_d'])
    )
loans = (
    pd.read_csv('data/loans.csv')
    .pipe(clean_term_column)
    .pipe(clean_date_column)
)
loans
```

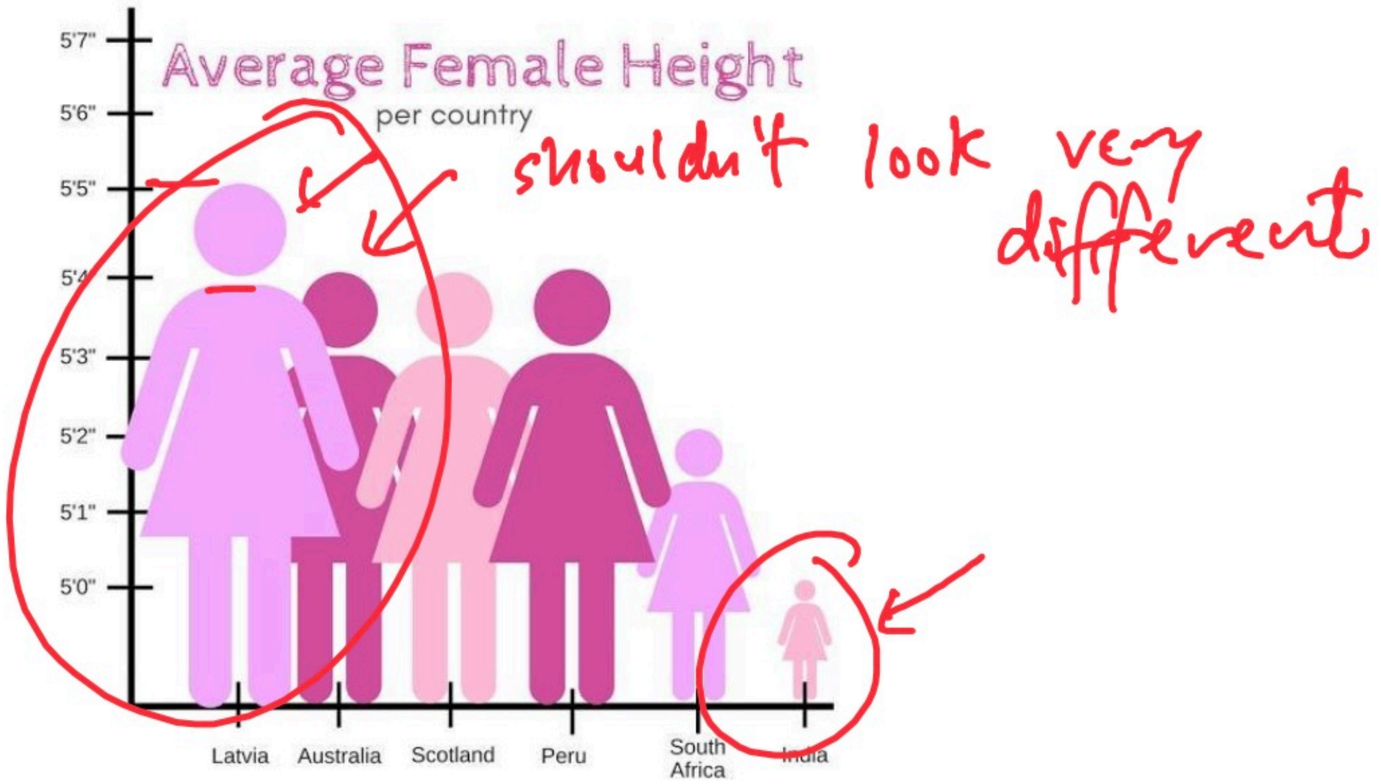
data cleaning steps

Out[2]:

	id	loan_amnt	term	int_rate	...	fico_range_high	hardship_flag	mths_since_last_delinq	date
0	17965023	18000.0	60	16.99	...	704.0	N	72.0	2014-06-01
1	111414087	10000.0	36	16.02	...	684.0	N	6.0	2017-06-01
2	95219557	12800.0	36	7.99	...	709.0	N	66.0	2016-12-01
...	...	...	...	...	...	...	...	...	...
6297	63990101	10800.0	60	18.49	...	679.0	N	39.0	2015-11-01
6298	37641672	15000.0	60	14.31	...	664.0	N	22.0	2014-12-01
6299	50587446	14000.0	60	9.99	...	689.0	N	NaN	2015-06-01

6300 rows × 20 columns

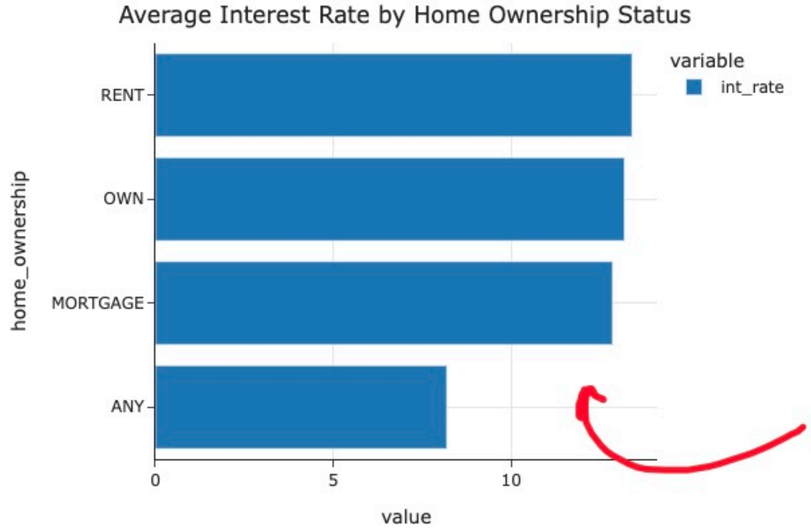




What's wrong with this visualization?

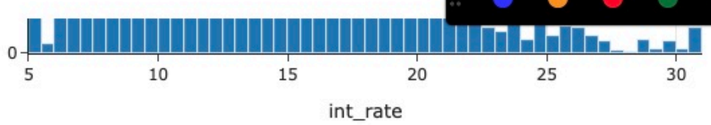
- Example: What is the average `'int_rate'` for each `'home_ownership'` status?

In [36]:
```
(
    loans
    .groupby('home_ownership')
    ['int_rate']
    .mean()
    .plot(kind='barh', title='Average Interest Rate by Home Ownership Status')
)
```



lengths of bars correspond to average interest rates, NOT counts.

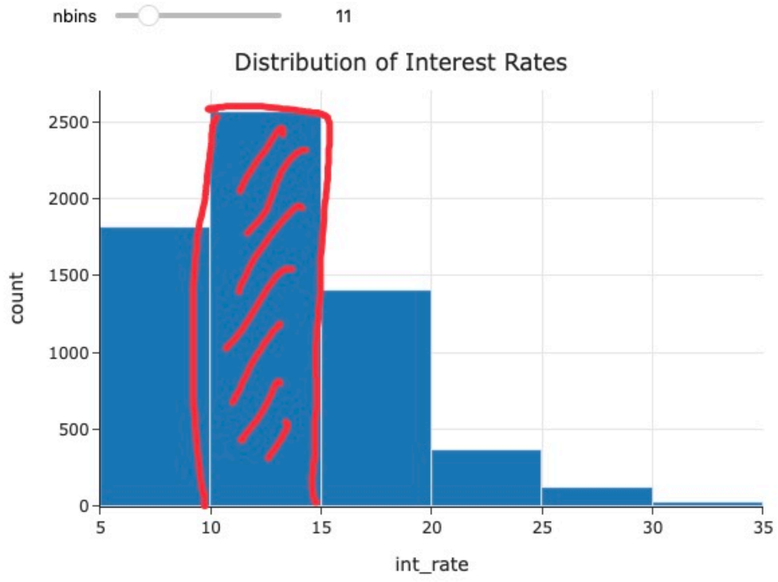
In [ ]:
```
# The "ANY" category seems to be an outlier.
loans['home_ownership'].value_counts()
```



- With fewer bins, we see less detail (and less noise) in the shape of the distribution.

  Play with the slider that appears when you run the cell below!

In [41]:
```
def hist_bins(nbins):
    (
        loans
        .plot(kind='hist', x='int_rate', nbins=nbins, title='Distribution of Interest Rates')
        .show()
    )
interact(hist_bins, nbins=(1, 51));
```



relative areas as proportions

e.g. % between 10-15% = Area / Total Area

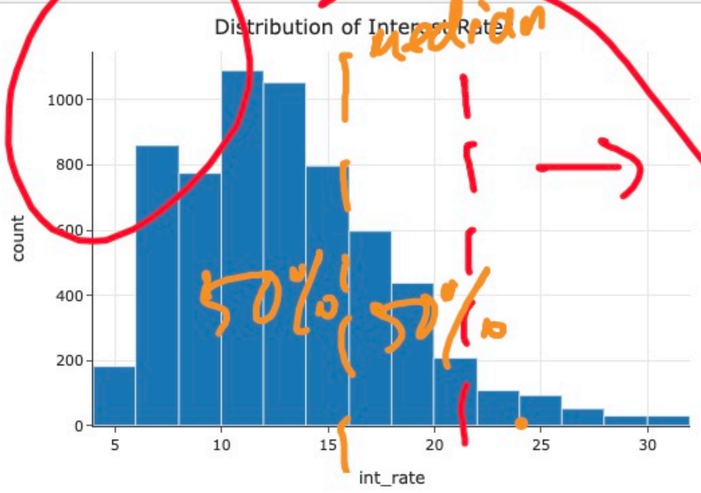
Question 🤔 (Answer at practicaldsc.org/q)

Remember that you can always ask questions anonymously at the link above!

Based on the histogram below, what is the relationship between the mean and median interest rate?

- A. Mean > median.
- B. Mean ≈ median.
- C. Mean < median.

In [42]:
```
(
    loans
    .plot(kind='hist', x='int_rate', title='Distribution of Interest Rates', nbins=20)
)
```



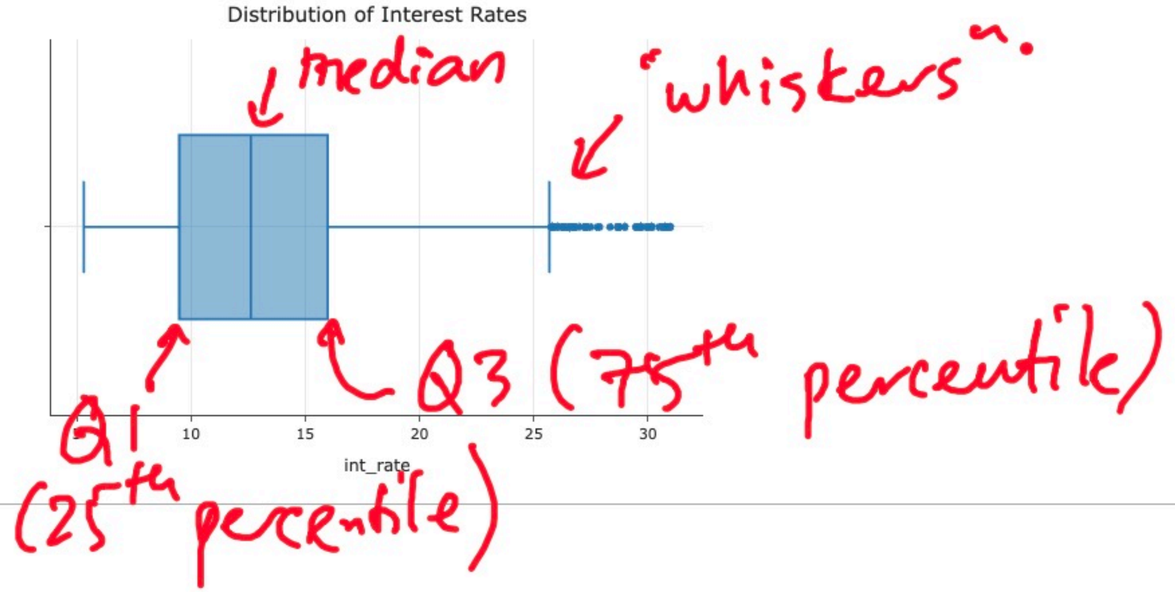
median

50% 50%

mean pulled up by tail

- Example: What is the distribution of `'int_rate'`?

In [43]:
```
(
    loans
    .plot(kind='box', x='int_rate', title='Distribution of Interest Rates')
)
```



In [52]:
```
loans.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6300 entries, 0 to 6299
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   id                      6300 non-null   int64
 1   loan_amnt               6300 non-null   float64
 2   term                    6300 non-null   int64
 3   int_rate                6300 non-null   float64
 4   grade                   6300 non-null   object
 5   sub_grade               6300 non-null   object
 6   emp_title               6300 non-null   object
 7   verification_status     6300 non-null   object
 8   home_ownership          6300 non-null   object
 9   annual_inc              6300 non-null   float64
 10  loan_status             6300 non-null   object
 11  purpose                 6300 non-null   object
 12  desc                    324 non-null    object
 13  addr_state              6300 non-null   object
 14  dti                     6299 non-null   float64
 15  fico_range_low          6300 non-null   float64
 16  fico_range_high         6300 non-null   float64
 17  hardship_flag           6300 non-null   object
 18  mths_since_last_delinq  3120 non-null   float64
 19  date                    6300 non-null   datetime64[ns]
dtypes: datetime64[ns](1), float64(7), int64(2), object(10)
memory usage: 984.5+ KB
```

Most borrowers did not provide a description (`'desc'`).

```
Name: child, Length: 934, dtype: float64
```

- A better idea is to impute missing values with the **mean** of the observed values.

In [79]:
```
heights['child']
```

Out[79]:
```
0      NaN
1     69.2
2     69.0
      ...
931   61.0
932   66.5
933   57.0
Name: child, Length: 934, dtype: float64
```

mean of observed = 67.1

In [80]:
```
heights['child'].mean()
```

Out[80]:
```
67.10339869281046
```

In [81]:
```
mean_imputed = heights['child'].fillna(heights['child'].mean())
mean_imputed
```

Out[81]:
```
0     67.1
1     69.2
2     69.0
      ...
931   61.0
932   66.5
933   57.0
Name: child, Length: 934, dtype: float64
```