In [39]:
```
# Initialize Otter
import otter
grader = otter.Notebook("hw3-diatom.ipynb")
```

PSTAT 100 Homework 3

In [40]:
```
import numpy as np
import pandas as pd
import altair as alt
from sklearn.decomposition import PCA
```

Background: diatoms and paleoclimatology

Diatoms are a type of phytoplankton -- they are photosynthetic algae that function as primary producers in aquatic ecosystems. Diatoms are at the bottom of the food web: they are consumed by filter feeders, like clams, mussels, and many fish, which are in turn consumed by larger organisms like scavengers and predators and, well, us. As a result, changes in the composition of diatom species in marine ecosystems have ripple effects that can dramatically alter overall community structure in any environment of which marine life forms a part.

Diatoms have glass bodies. As a group of organisms, they display a great diversity of body shapes, and many are quite elaborate. The image below, taken from a Scientific American article, shows a small sample of their shapes and structures.

Because they are made of glass, diatoms preserve extraordinarily well over time. When they die, their bodies sink and form part of the sediment. Due to their abundance, there is a sort of steady rain of diatoms forming part of the sedimentation process, which produces sediment layers that are dense with diatoms.

Sedimentation is a long-term process spanning great stretches of time, and the deeper one looks in sediment, the older the material. Since diatoms are present in high density throughout sedimentation layers, and they preserve so well, it is possible to study their presence over longer time spans -- potentially hundreds of thousands of years.

A branch of paleoclimatology is dedicated to studying changes in biological productivity on geologic time scales, and much research in this area has involved studying the relative abundances of diatoms. In this assignment, you'll do just that on a small scale and work with data from sediment cores taken in the gulf of California at the location indicated on the map:

The data is publicly available:

> Barron, J.A., *et al.* 2005. High Resolution Guaymas Basin Geochemical, Diatom, and Silicoflagellate Data. IGBP PAGES/World Data Center for Paleoclimatology Data Contribution Series # 2005-022. NOAA/NGDC Paleoclimatology Program, Boulder CO, USA.

Assignment objectives

In this assignment, you'll use the exploratory techniques we've been discussing in class to analyze the relative abundances of diatom taxa over a time span of 15,000 years. This will involve practicing the following skills.

Acquaint and tidy

- data import
- handling NaNs
- transforming values
- assessing time resolution

Exploratory analysis of individual variables

- visualizing summary statistics
- density histograms and kernel density estimates
- describing variation

Exploratory analysis of multiple variables

- examining correlation structure
- computing and selecting principal components
- interpreting principal component loadings
- using principal components to visualize multivariate data

Communication and critical thinking

- summarizing results in written form
- suggesting next steps

Have fun!

Collaboration

You are encouraged to collaborate with other students on the labs, but are expected to write up your own work for submission. Copying and pasting others' solutions is considered plaigarism and may result in penalties, depending on severity and extent.

If you choose to work with others, please list their names here.

Your name: Marissa Santiago

Collaborators:

---

0. Getting acquainted with the diatom data

In this assignment you'll focus less on tidying and more on exploration -- the data you'll work with are already tidy. So, in this initial part, you'll:

- import the data;
- examine its structure to get acquainted; and
- perform some simple preprocessing transformations to facilitate exploratory analysis.

The data are diatom counts sampled from evenly-spaced depths in a sediment core from the gulf of California. In sediment cores, depth correlates with time before the present -- deeper layers are older -- and depths are typically chosen to obtain a desired temporal resolution. The counts were recorded by sampling material from sediment cores at each depth, and examining the sampled material for phytoplankton cells. For each sample, phytoplankton were identified at the taxon level and counts of diatom taxa were recorded along with the total number of phytoplankton cells identified. Thus:

- The **observational units** are ***sediment samples***.
- The **variables** are ***depth (age), diatom abundance counts, and the total number of identified phytoplankton***. Age is inferred from radiocarbon.
- One **observation** is made at ***each depth*** from 0cm (surface) to 13.71 cm.

The table below provides variable descriptions and units for each column in the dataframe.

Variable	Description	Units
Depth	Depth interval location of sampled material in sediment core	Centimeters (cm)
Age	Radiocarbon age	Thousands of years before present (KyrBP)
A_curv	Abundance of *Actinocyclus curvatulus*	Count (n)

A_octon	Abundance of *Actinocyclus octonarius*	Count (n)
ActinSpp	Abundance of *Actinoptychus* species	Count (n)
A_nodul	Abundance of *Azpeitia nodulifer*	Count (n)
CocsinSpp	Abundance of *Coscinodiscus* species	Count (n)
CyclotSpp	Abundance of *Cyclotella* species	Count (n)
Rop_tess	Abundance of *Roperia tesselata*	Count (n)
StephanSpp	Abundance of *Stephanopyxis* species	Count (n)
Num.counted	Number of diatoms counted in sample	Count (n)

The cell below imports the data.

In [41]:
```
# import diatom data
diatoms_raw = pd.read_csv('data/barron-diatoms.csv')
diatoms_raw.head(5)
```

Out[41]:

	Depth	Age	A_curv	A_octon	ActinSpp	A_nodul	CoscinSpp	CyclotSpp	Rop_tess	Step
0	0.00	1.33	5.0	2.0	32	14.0	21	22.0	1.0	
1	0.05	1.37	8.0	2.0	31	16.0	20	16.0	7.0	
2	0.10	1.42	8.0	6.0	33	18.0	29	7.0	1.0	
3	0.15	1.46	11.0	1.0	21	1.0	12	28.0	25.0	
4	0.20	1.51	11.0	1.0	38	3.0	18	24.0	3.0	

The data are already in tidy format, because each row is an observation (a set of measurements on one sample of sediment) and each column is a variable (one of age, depth, or counts). However, examine rows 3 and 4. These rows illustrate two noteworthy features of the raw data:

1. NaNs are present
2. The number of individuals counted in each sample varies by a lot from sample to sample.

Let's address those before conducting initial explorations.

'Missing' values

The NaNs are an artefact of the data recording -- if *no* diatoms in a particular taxa are observed, a `-` is entered in the table (you can verify this by checking the .csv file). In these cases the value isn't missing, but rather zero. These entries are parsed by pandas as NaNs, but they correspond to a value of 0 (no diatoms observed).

Q0 (a). Filling NaNs

Use `.fill_na()` to replace all NaNs by zeros, and store the result as `diatoms_mod1`. Store rows 4 and 5 (index, not integer location) of the resulting dataframe as `diatoms_mod1_sample` and print it out.

(*Hint*: check the documentation for `fill_na()`.)

In [42]:
```
diatoms_mod1 = diatoms_raw.fillna(0)

# print rows 4 and 5
diatoms_mod1_sample = diatoms_mod1[4:6]
print(diatoms_mod1_sample)
```

```
   Depth   Age  A_curv  A_octon  ActinSpp  A_nodul  CoscinSpp  CyclotSpp  \
4   0.20  1.51    11.0      1.0        38      3.0         18       24.0   
5   0.25  1.55     4.0      9.0        30     10.0         16       14.0   

   Rop_tess  StephanSpp  Num.counted  
4       3.0         0.0          300  
5      16.0         0.0          203  
```

In [43]:
```
grader.check("q0_a")
```

Out[43]:

q0_a passed!

Varying total counts

Since the total number of phytoplankton counted in each sample varies, the raw counts are not directly comparable -- *e.g.*, a count of 18 is actually a *different* abundance in a sample with 200 individuals counted than in a sample with 300 individuals counted.

For exploratory analysis, you'll want the values to be comparable across rows. This can be achieved by a simple transformation so that the values are *relative* abundances: *proportions* of phytoplankton observed from each taxon.

Q0 (b). Counts to proportions

Convert the counts to proportions by dividing by the relevant entry in the `Num.counted` column. There are a few ways to do this, but here's one approach:

1. Set Depth and Age to row indices using `.set_index(...)` and store the result as `diatoms_mod2`.
2. Store the `Num.counted` column from `diatoms_mod2` as `sampsize`.
3. Use `.div(...)` to divide entrywise every column in `diatoms_mod2` by `sampsize` and store the result as `diatoms_mod3`.
4. Drop the `Num.counted` column from `diatoms_mod3` and reset the index; store the result as `diatoms`.

Carry out these steps and print the first four rows of `diatoms`.

(*Hint*: careful with the `axis = ...` argument in `.div(...)`; you may want to look at the documentation.)

In [44]:
```
# set depth, age to indices
diatoms_mod2 = diatoms_mod1.set_index(['Depth','Age'])

# store sample sizes
sampsize = diatoms_mod2['Num.counted']

# divide
diatoms_mod3 = diatoms_mod2.div(sampsize,axis = 0)

# drop num.counted and reset index
diatoms = diatoms_mod3.drop(columns = ['Num.counted']).reset_index()

# print
diatoms.head(4) #230 total rows
```

Out[44]:

	Depth	Age	A_curv	A_octon	ActinSpp	A_nodul	CoscinSpp	CyclotSpp	Rop_tess
0	0.00	1.33	0.024876	0.00995	0.159204	0.069652	0.104478	0.109453	0.004975
1	0.05	1.37	0.040000	0.01000	0.155000	0.080000	0.100000	0.080000	0.035000
2	0.10	1.42	0.040000	0.03000	0.165000	0.090000	0.145000	0.035000	0.005000
3	0.15	1.46	0.055000	0.00500	0.105000	0.005000	0.060000	0.140000	0.125000

In [45]:
```
grader.check("q0_b")
```

Out[45]:

q0_b passed!

Now that the data are ready for exploratory analysis, take a moment to think about what the data represent. They are relative abundances over time; essentially, snapshots of the community composition of diatoms over time, and thus information about how ecological community composition changes.

Before diving in, it will be helpful to resolve two matters:

1. How far back in time do the data go?
2. What is the time resolution of the data?

Q0 (c). Time span

What is the geological time span covered by the data? Compute the minimum and maximum age using `.aggregate(...)` and store as `min_max_age`.

Note: This may be a new function for you, but it's simple: it takes as an argument a list of functions that will be applied to the dataframe (columnwise by default). So for example, to get the mean and variance of each column in `df`, one would use `df.aggregate(['mean', 'var'])`. See the documentation for further examples.

(*Remember*: age is reported as thousands of years before present, so `Age = 2` means 2000 years ago.)

In [46]:
```
#pull out age column
min_max_age = diatoms['Age'].aggregate(['min','max'])
min_max_age
```

Out[46]:
```
min      1.33
max     15.19
Name: Age, dtype: float64
```

In [47]:
```
grader.check("q0_c")
```

Out[47]:

`q0_c` passed!

The most recent value in the dataset is from 1,330 years ago and oldest value was taken 15,190 years ago.

Q0 (d). Time resolution

How are the observations spaced in time?

(i) Make a histogram of the time steps between consecutive sample ages.

Follow these steps:

1. Extract the `Age` column from `diatoms`, sort the values in ascending order, compute the differences between consecutive rows, and store the result as `diffs`.
    - *Hint*: use `.sort_values()` and `.diff()`.
    - *Notice*: that the first difference is NaN, because there is no previous value to compare the first row with. Drop this entry when you store `diffs`.

1. Make a simple count histogram (no need to manually bin or convert to density scale) with bins of width 0.02 (20 years).
    - Label the x axis 'Time step between consecutive sample ages'

In [48]:
```python
# store differences
diffs = diatoms[['Age']].sort_values(
    by = 'Age', ascending = True).diff().dropna().rename(columns={'Age':'dif

# construct histogram
alt.Chart(diffs).mark_bar().encode(
    x = alt.X('Age',
              bin = alt.Bin(step=0.02),
              title = 'Time steps between consecutive sample ages'),
    y = 'count()'
)

diffs.head()
```

Out[48]:

	diff
1	0.04
2	0.05
3	0.04
4	0.05
5	0.04

In [49]:
```python
grader.check("q1_d_i")
```

Out[49]:

q1_d_i passed!

(ii) What is the typical time step in years?

The typical time step is between 40-60 years.

---

1. Exploring diatom taxon abundances

Recall that the first type of exploratory analysis question has to do with exploring variation in each variable; to begin, you'll examine the variation in relative abundance over time for the eight individual taxa.

Here are some initial questions in this spirit that will help you to hone in and develop more focused exploratory questions:

- Which taxa are most and least abundant on average over time?
- Which taxa vary the most over time?

These can be answered by computing simple summary statistics for each column in the diatom data.

Q1 (a). Summary statistics

Use `.aggregate(...)` to find the mean and standard deviation of relative abundances for each taxon. Follow these steps:

1. See Q0 (c) for an explanation of `.aggregate(...)`.
2. Drop the depth and age variables before performing the aggregation.
3. Use `.transpose()` to ensure that the table is rendered in long form (8 rows by 2 columns rather than 2 columns by 8 rows).
4. Store the result as `diatom_summary` and print the dataframe.

In [50]:
```python
diatom_summary = diatoms.drop(
    columns= ['Depth','Age']).aggregate(['mean','std']).transpose()

# print the dataframe
diatom_summary
```

Out[50]:

	mean	std
A_curv	0.028989	0.018602
A_octon	0.018257	0.016465
ActinSpp	0.135900	0.053797
A_nodul	0.072940	0.092677
CoscinSpp	0.085925	0.031795
CyclotSpp	0.070366	0.042423
Rop_tess	0.060448	0.076098
StephanSpp	0.002447	0.007721

In [51]:
```
grader.check("q1_a")
```

Out[51]:

q1_a passed!

It will be easier to determine which taxa are most/least abundant and most variable by displaying this information visually.

Q1 (b). Visualizing summary statistics

Create a plot of the average relative abundances and their variation over time by following these steps:

1. Reset the index of `diatom_summary` so that the taxon names are stored as a column and not an index. Store the result as `plot_df`.

2. Create an Altair chart based on `plot_df` with *no marks* -- just `alt.Chart(...).encode(...)` -- and pass the columnn of taxon names to the `Y` encoding channel with the title 'Taxon' and sorted in descending order of mean relative abundance. Store the result as `base`.

   - *Hint*: `alt.Y(..., sort = {'field': 'column', 'order': 'descending'})` will sort the Y channel by 'column' in descending order.

1. Modify `base` to create a point plot of the average relative abundances for each taxon; store the result as `means`.
   - Average relative abundance (the mean you calculated in Q1 (a)) should appear on the x axis, and taxon on the y axis.
   - Since the `Y` encoding was already specified in `base`, you do not need to add a `Y` encoding at this stage.
   - Give the x axis the title 'Average relative abundance'.

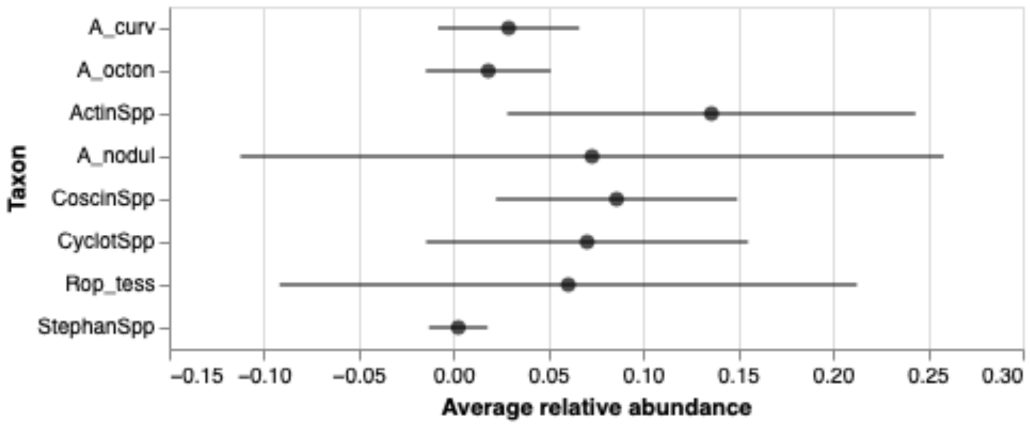
1. Modify `base` to create a plot with bars spanning two standard deviations in either direction from the mean. Store the result as `bars`.
   - First use `base.transform_calculate(...)` to compute `lwr` and `upr` for the positions of the bar endpoints:
     - $\texttt{lwr} = \texttt{mean} - 2 \times \texttt{std}$
     - $\texttt{upr} = \texttt{mean} + 2 \times \texttt{std}$.
   - Then append `.mark_errorbar().encode(...)` to the chain:
     - pass `lwr:Q` to the `X` encoding channel with the title 'Average relative abundance' (to match the point plot)
     - pass `upr:Q` to the `X2` encoding channel (no specific title needed).

1. Layer the plots: `means + bars`.

It may help to have a look at this example. Once you make the plot, answer questions (i) - (iii) below.

In [52]:
```python
# reset index
plot_df = diatom_summary.reset_index()

# create base chart
base = alt.Chart(plot_df).encode(
                    y = alt.Y('index', sort = {'field': 'column', 'order': '

# create point plot
means = base.mark_point(
    filled = True,
    size = 50,
    color = 'black'
).encode(
    x = alt.X('mean',title = 'Average relative abundance'))

# create bar plot
bars = base.transform_calculate(lwr = 'datum.mean-2*datum.std', upr = 'datum
).encode(x= alt.X('lwr:Q' , title = 'Average relative abundance'),x2 = 'upr:

# layer
means + bars
```

Out[52]:



(i) Which taxon is most abundant on average over time?

From the graph, ActionSpp is the most abundant on average over time.

(ii) Which taxon is most rare on average over time?

From the graph, we can see StephanSpp is most rare on average over time.

(iii) Which taxon varies most in relative abundance over time?

From the graph, we can see A_nodul varies most in relative abundance over time.

Now that you have a sense of the typical abundances for each taxon (measured by means) and the variations in abundance (measured by standard deviations), you'll dig in a bit further and examine the variation in abundance of the most variable taxon.

For the next few questions, it may help you to follow code examples from lab 4.

Q1 (c). Distribution of *Azpeitia nodulifer* abundance over time

Here you'll construct a few plots that will help you answer the following key exploratory questions:

- Which values are common?
- Which values are rare?
- How spread out are the values?
- Are values spread evenly or irregularly?

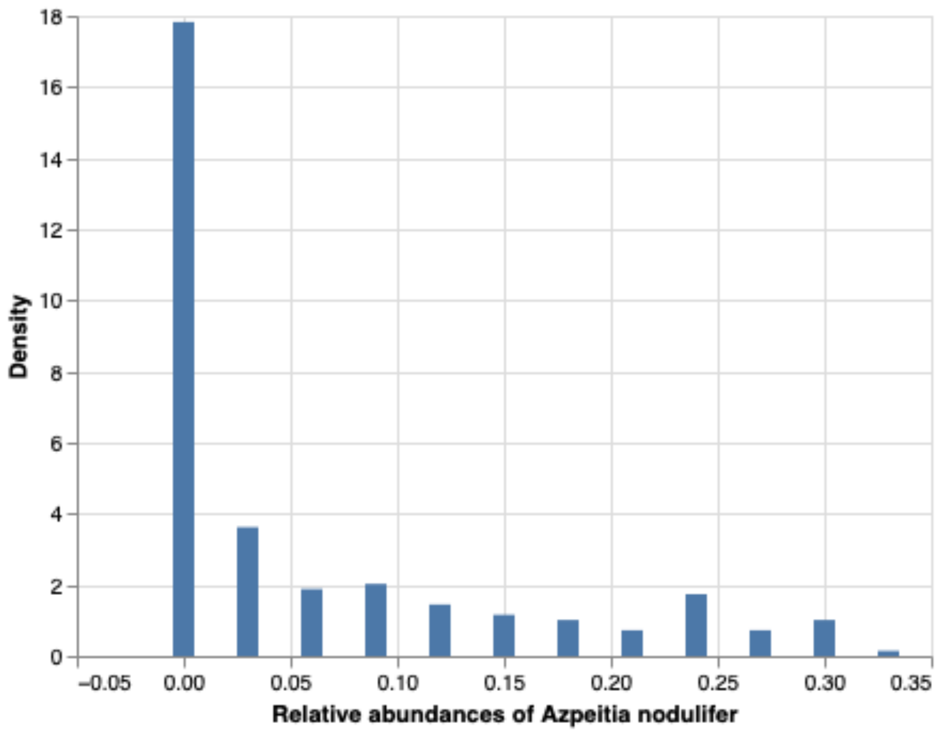
(i) Construct a density scale histogram of the relative abundances of *Azpeitia nodulifer*.

Use the `diatoms` dataframe and a bin width of 0.03 and store the histogram as `hist`.

Hint: It may help to look at *Q0* in **Lab4 on Smoothing**

In [53]:
```
# filter, bin, count, convert scale, and plot
hist = alt.Chart(
    diatoms
).transform_bin(
    'Relative abundances of Azpeitia nodulifer',
    field = 'A_nodul',
    bin = alt.Bin(step = 0.03)
).transform_aggregate(
    Count = 'count()',
    groupby = ['Relative abundances of Azpeitia nodulifer']
).transform_calculate(
    Density = 'datum.Count/(0.03*230)' # divide counts by sample size x binw
).mark_bar(size = 10).encode(
    x = 'Relative abundances of Azpeitia nodulifer:Q',
    y = 'Density:Q'
)

hist
```

Out[53]:



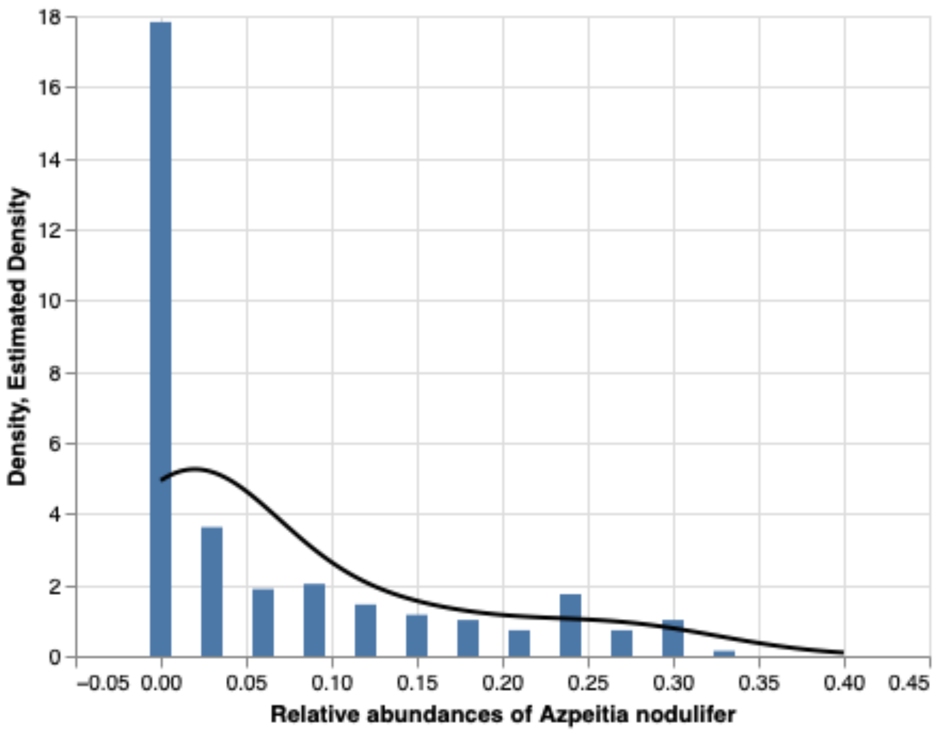
(ii) Construct a kernel density estimate of the distribution.

Create and store the KDE curve as `smooth` and layer it on top of the density histogram from part i.

(*Remember*: experiment with the bandwidth parameter, and find a value that you feel captures the shape best.)

In [54]:
```
smooth = alt.Chart(
    diatoms
).transform_density(
    density = 'A_nodul', # variable to smooth
    as_ = ['Relative abundances of Azpeitia nodulifer', 'Estimated Density']
    bandwidth = 0.05, # how smooth?
    extent = [0, 0.4], # domain on which the smooth is defined
    steps = 500 # for plotting: number of points to generate for plotting li
).mark_line(color = 'black').encode(
    x = 'Relative abundances of Azpeitia nodulifer:Q',
    y = 'Estimated Density:Q'
)

hist + smooth
```

```
Out[54]:
```



(iii) Which values are common?

Value of 0 are most common.

(iv) Which values are rare?

Values of ~0.33 are rare.

(v) How spread out are the values and how are they spread out?

The values are spread out from 0.0 to 0.33. Values are most concentrated around value 0 and seem to spread out towards 0.33.

(vi) How would you describe the shape?

I would describe the graph as skewed right as the data is most concentrated on the left side of the data while the rest of the data slowly spreads to the right.

Comment: 'zero inflation'

There are a disproportionately large number of zeroes, because in many samples no *Azpeitia nodulifer* diatoms were observed. This is a common phenomenon in ecological data, and even has a name: it results in a 'zero inflated' distribution of values. The statistician to identify and name the phenomenon was Diane Lambert, whose highly influential work on the subject (>4k citations) was published in 1992.

Zero inflation can present a variety of challenges. You may have noticed, for example, that there was no bandwidth parameter for the KDE curve that *both* captured the shape of the histogram near zero *and* away from zero -- it either got the height near zero right but was too wiggly, or got the shape away from zero right but was too low near zero.

Conditioning on a climate event

There was a major climate event during the time span covered by the diatom data. The oldest data points in the diatom data correspond to the end of the Pleistocene epoch (ice age), at which time there was a pronounced warming (Late Glacial Interstadial, 14.7 - 12.9 KyrBP) followed by a return to glacial conditions (Younger Dryas, 12.9 - 11.7 KyrBP).

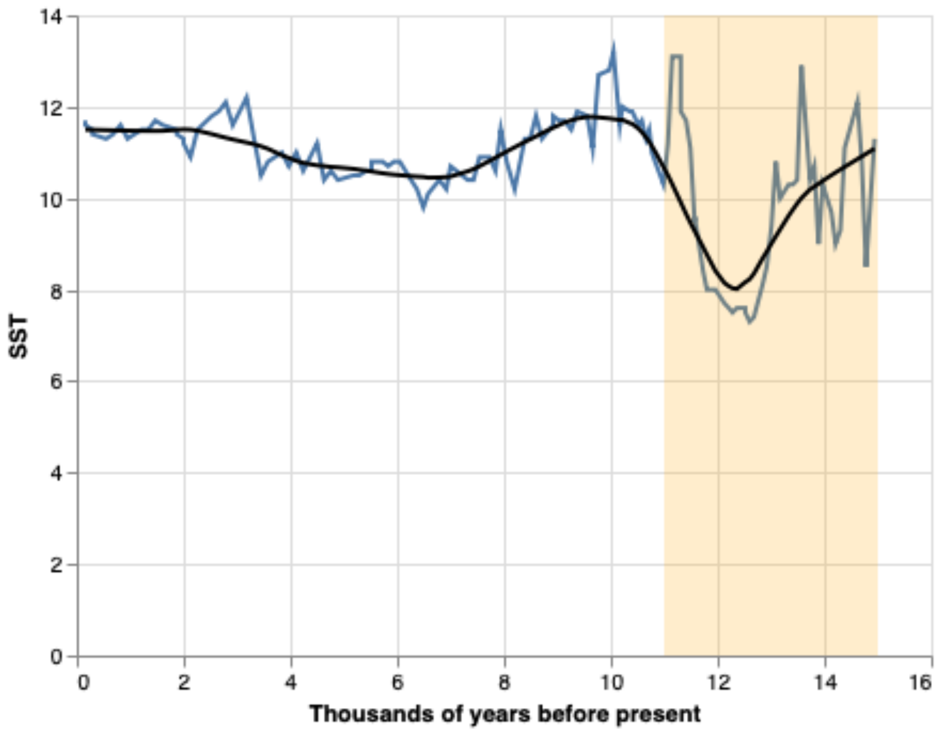
This fluctuation can be seen from temperature reconstructions. Below is a plot of sea surface temperature reconstructions off the coast of Northern California. Data come from the following source:

> Barron *et al.*, 2003. Northern Coastal California High Resolution Holocene/Late Pleistocene Oceanographic Data. IGBP PAGES/World Data Center for Paleoclimatology. Data Contribution Series # 2003-014. NOAA/NGDC Paleoclimatology Program, Boulder CO, USA.

The shaded region indicates the time window with unusually large flucutations in sea surface temperature; this window roughly corresponds to the dates of the climate event.

```
In [55]: # import sea surface temp reconstruction
         seatemps = pd.read_csv('data/barron-sst.csv')

         # line plot of time series
         line = alt.Chart(seatemps).mark_line().encode(
             x = alt.X('Age', title = 'Thousands of years before present'),
             y = 'SST'
         )

         # highlight region with large variations
         highlight = alt.Chart(
             pd.DataFrame(
                 {'SST': np.linspace(0, 14, 100),
                  'upr': np.repeat(11, 100),
                  'lwr': np.repeat(15, 100)}
             )
         ).mark_area(opacity = 0.2, color = 'orange').encode(
             y = 'SST',
             x = alt.X('upr', title = 'Thousands of years before present'),
             x2 = 'lwr'
         )

         # add smooth trend
         smooth = line.transform_loess(
             on = 'Age',
             loess = 'SST',
             bandwidth = 0.2
         ).mark_line(color = 'black')

         # layer
         line + highlight + smooth
```

Out[55]:



Q1 (d). Conditional distributions of relative abundance

Does the distribution of relative abundance of *Azpeitia nodulifer* differ when variation in sea temperatures was higher (before 11KyrBP)?

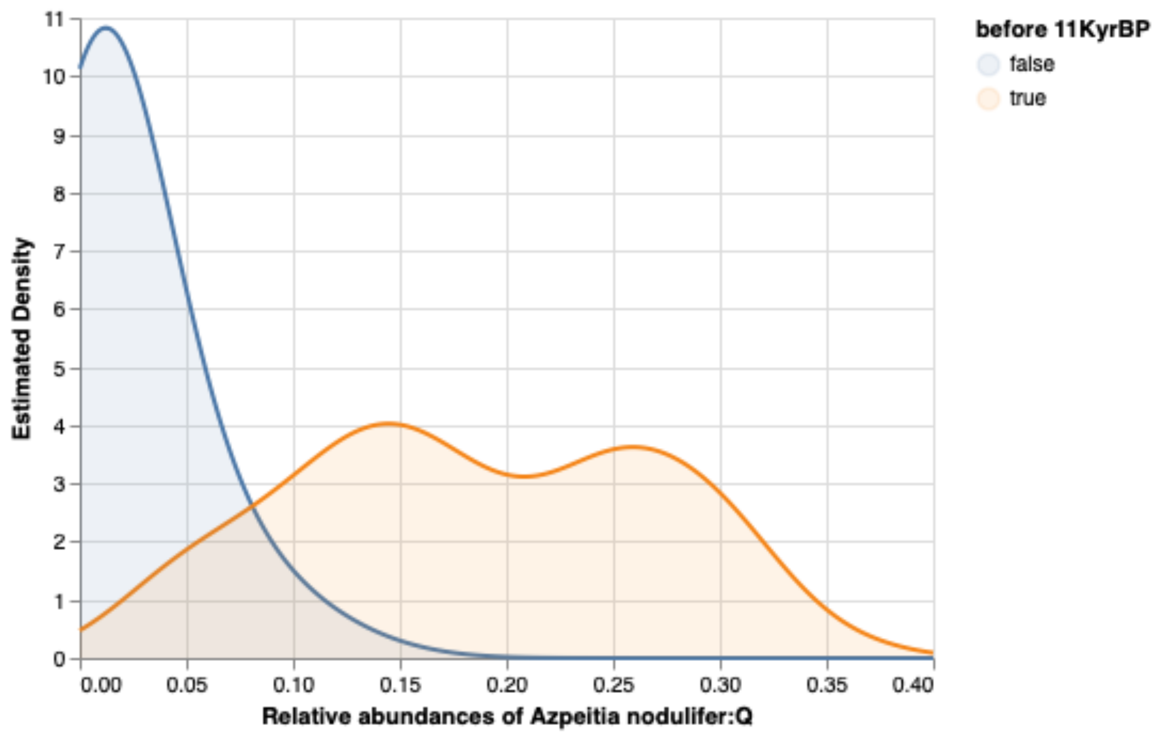
(i) Plot kernel density estimates to show the distribution of relative abundances before and after 11KyrBP.

1. Use `.transform_caluculate(...)` to calculate an indicator variable, `pre_dryas`, that indicates whether `Age` exceeds 11.
2. Use `.transform_density(...)` to compute KDEs separately for observations of relative abundance before and after 11KyrBP.
   - *Hint*: group by `pre_dryas`

1. Plot the KDEs distinguished by color; give the color legend the title 'Before 11KyrBP' and store the plot as `kdes`.
2. Add a shaded area beneath the KDE curves. Adjust the opacity of the area to your liking.

```
In [56]: kdes = mooth = alt.Chart(diatoms).transform_calculate(
             pre_dryas = 'datum.Age > 11'
         ).transform_density(
             density = 'A_nodul', # variable to smooth
             groupby = ['pre_dryas'],
             as_ = ['A_nodul', 'Estimated Density'], # names of outputs
             bandwidth = 0.03, # how smooth?
             extent = [0, 0.4], # domain on which the smooth is defined
             steps = 500 # for plotting: number of points to generate for plotting li
         ).mark_line().encode(
             x = alt.X('A_nodul:Q', title= 'Relative abundances of Azpeitia nodulifer
             y = 'Estimated Density:Q',
             color = alt.Color('pre_dryas:N', legend = alt.Legend(title = 'before 11K
         )

         kdes + kdes.mark_area(opacity = 0.1)
```

Out[56]:



(ii) Describe the variation in relative abundance of *Azpeitia nodulifer* between now and 11,000 years ago.

There is less variation in relative abundance of Azpeitia nodulifer now then 11,000 years ago because the blue curve representing now is much more narrow than the orange curve representing the data 11,000 years ago.

(iii) Describe the variation in relative abundance of *Azpeitia nodulifer* between 11,000 and 14,000 years ago.

The variation in relative abundance of Azpeitia nodulifer between 11,000 and 14,000 years ago is large because the curve is more widespread so it captures more vaariance.

---

2. Visualizing community composition with PCA

So far you've seen that the abundances of one taxon -- *Azpeitia nodulifer* -- change markedly before and after a shift in climate conditions. In this part you'll use PCA to explore variation in community composition *among* all eight taxa.

Throughout this part, it may help you to refer to code examples from lab 5.

Q2 (a). Pairwise correlations in relative abundances

Before carrying out PCA it is a good idea to inspect the correlations between relative abundances directly. Here you'll compute and then visualize the correlation matrix.

(i) Compute the pairwise correlations between relative abundances.

Be sure to remove or set to indices the Depth and Age variables before computing the correlation matrix. Save the matrix as `corr_mx` and print the result.

Hint: See the pandas documentation for `.corr()`.

In [57]:
```
corr_mx = diatoms.drop(columns = ['Depth','Age']).corr()
corr_mx
```

Out[57]:

	A_curv	A_octon	ActinSpp	A_nodul	CoscinSpp	CyclotSpp	Rop_tess
A_curv	1.000000	0.111480	0.390898	-0.446778	0.091222	0.219439	-0.062690
A_octon	0.111480	1.000000	-0.005009	-0.217992	0.049589	0.065249	-0.023047
ActinSpp	0.390898	-0.005009	1.000000	-0.363475	0.306021	-0.055732	-0.343410
A_nodul	-0.446778	-0.217992	-0.363475	1.000000	-0.010920	-0.407338	-0.471941
CoscinSpp	0.091222	0.049589	0.306021	-0.010920	1.000000	-0.266157	-0.341755
CyclotSpp	0.219439	0.065249	-0.055732	-0.407338	-0.266157	1.000000	0.018149
Rop_tess	-0.062690	-0.023047	-0.343410	-0.471941	-0.341755	0.018149	1.000000
StephanSpp	0.151909	-0.041017	0.058494	-0.151409	-0.016332	0.070684	0.032607

In [58]:
```
grader.check("q2_a_i")
```

Out[58]:

q2_a_i passed!

(ii) Visualize the correlation matrix as a heatmap

Have a look at either lab 5 or this example (or both!). Notice that to make a heatmap of a matrix, you'll need to melt it into long format.

1. Melt `corr_mx` to obtain a dataframe with three columns:

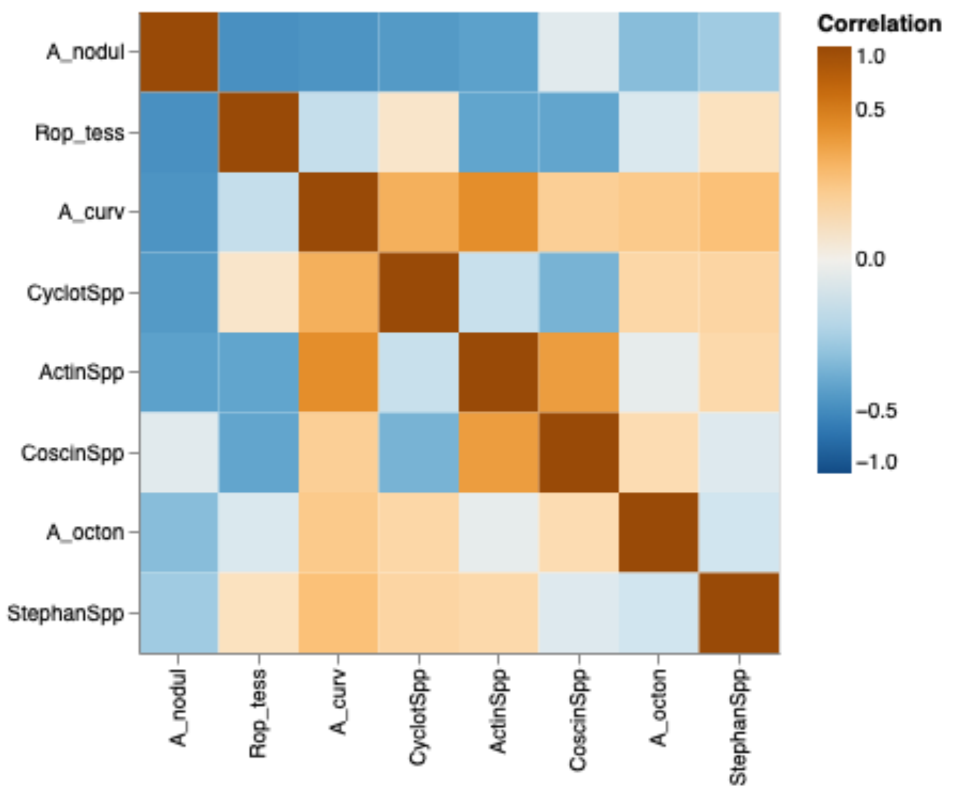
   - `row`, which contains the values of the index of `corr_mx` (taxon names);
   - `column`, which contains the names of the columns of `corr_mx` (also taxon names); and
   - `Correlation`, which contains the values of `corr_mx`.

     Store the result as `corr_mx_long`.

1. Create an Altair chart based on `corr_mx_long` and construct the heatmap by following the examples indicated above.
   - Adjust the color scheme to `blueorange` over the extent (-1, 1) to obtain a diverging color gradient where a correlation of zero is blank (white).
   - Adjust the color legend to indicate the color values corresponding to correlations of 1, 0.5, 0, -0.5, and -1.
   - Sort the rows and columns in ascending order of correlation.

In [59]:
```
# melt corr_mx
corr_mx_long = corr_mx.reset_index().rename(
    columns = {'index': 'row'}
).melt(
    id_vars = 'row',
    var_name = 'col',
    value_name = 'Correlation'
)

# construct plot
alt.Chart(corr_mx_long).mark_rect().encode(
    x = alt.X('col', title = '', sort = {'field': 'Correlation', 'order': 'a
    y = alt.Y('row', title = '', sort = {'field': 'Correlation', 'order': 'a
    color = alt.Color('Correlation',
                      scale = alt.Scale(scheme = 'blueorange', # diverging g
                                        domain = (-1, 1), # ensure white = 0
                                        type = 'sqrt'), # adjust gradient sc
                     legend = alt.Legend(tickCount = 5)) # add ticks to colo
).properties(width = 300, height = 300)
```

Out[59]:



In [60]:
```
grader.check("q2_a_ii")
```

Out[60]:

q2_a_ii passed!

(iii) How does the relative abundance of *Azpeitia nodulifer* seem to covary with the other taxa?

The relative abundance of Azpeitia nodulifer seem to covary negatively with all the other taxa's besides taxa CoscinSpp which doesnt seem to covary at all. We can confirm this because the blue color represents a negative correlation in the legend and the white color represents zero correlation. We can infer he relative abundance of Azpeitia nodulifer means little presence of older taxas and vise versa.

Q2 (b). Computing and selecting principal components

Here you'll perform all of the calculations involved in PCA and check the variance ratios to select an appropriate number of principal components. The parts of this question correspond to the individual steps in this process.

(i) Center and scale the data columns.

For PCA it is usually recommended to center and scale the data; set Depth and Age as indices and center and scale the relative abundances. Store the normalized result as `pcdata`.

In [61]:
```
# helper variable pcdata_raw; set Depth and Age as indices
pcdata_raw = diatoms.set_index(['Depth','Age'])

# center and scale the relative abundances
pcdata = (pcdata_raw - pcdata_raw.mean())/pcdata_raw.std()
```

In [62]:
```
grader.check("q2_b_i")
```

Out[62]:

q2_b_i passed!

(ii) Compute the principal components.

Compute *all 8* principal components. (For this part you do not need to show any specific output.)

In [63]:
```
pca = PCA(n_components = pcdata.shape[1])
pca.fit(pcdata)
```

Out[63]:
```
PCA(n_components=8)
```

In [64]:
```
grader.check("q2_b_ii")
```

Out[64]:

q2_b_ii passed!

(iii) Examine the variance ratios.

Create a dataframe called `pcvars` with the variance information by following these steps:

1. Store the proportion of variance explained (called `.explained_variance_ratio_` in the PCA output) as a dataframe named `pcvars` with just one column named `Proportion of variance explained`.
2. Add a column named `Component` to `pcvars` with the integers 1 through 8 as values (indicating the component number).
3. Add a column named `Cumulative variance explained` to `pcvars` that is the cumulative sum of `Proportion of variance explained`.
   - *Hint*: slice the `Proportion of variance explained` column and use `.cumsum(axis = ...)`.

Print the dataframe `pcvars`.

In [65]:
```
pca.explained_variance_ratio_
# store proportion of variance explained as a dataframe
pcvars = pd.DataFrame({'Proportion of variance explained': pca.explained_var

# add component number as a new column
pcvars['Component'] = np.arange(1,9)

# add cumulative variance explained as a new column
pcvars['Cumulative variance explained'] = pcvars['Proportion of variance exp

# print
pcvars
```

Out[65]:

	Proportion of variance explained	Component	Cumulative variance explained
0	0.255513	1	0.255513
1	0.223354	2	0.478867
2	0.132145	3	0.611012
3	0.122549	4	0.733560
4	0.110833	5	0.844394
5	0.077988	6	0.922382
6	0.067303	7	0.989684
7	0.010316	8	1.000000

In [66]:
```
grader.check("q2_b_iii")
```

Out[66]:

q2_b_iii passed!

(iv) Plot the variance explained by each PC.

Use `pcvars` to construct a dual-axis plot showing the proportion of variance explained (left y axis) and cumulative variance explained (right y axis) as a function of component number (x axis), with points indicating the variance ratios and lines connecting the points.

Follow these steps:

1. Construct a base chart that encodes only `Component` on the `X` channel. Store this as `base`.
2. Make a base layer for the proportion of variance explained that modifies `base` by encoding `Proportion of variance explained` on the `Y` channel. Store the result as `prop_var_base`.
    - Give the `Y` axis title a distinct color of your choosing via `alt.Y(..., axis = alt.Axis(titleColor = ...))`.

1. Make a base layer for the cumulative variance explained that modifies `base` by endocing `Cumulative variance explained` on the `Y` channel. Store the result as `cum_var_base`.
    - Give the `Y` axis title another distinct color of your choosing via `alt.Y(..., axis = alt.Axis(titleColor = ...))`.

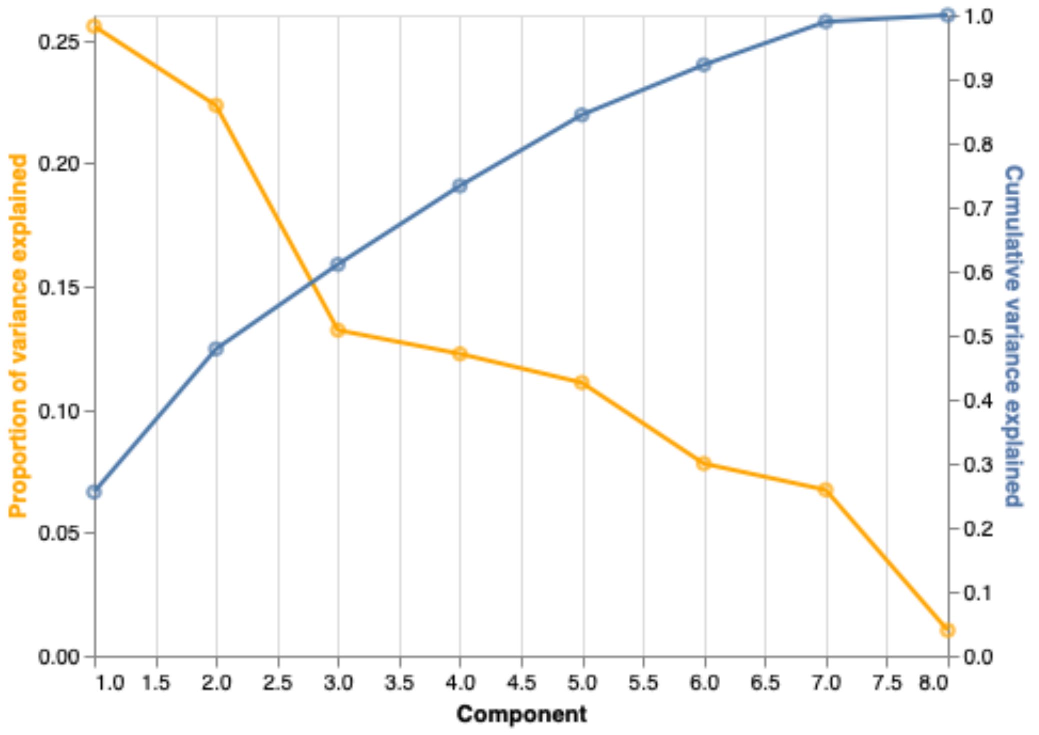
1. Create a plot layer for the proportion of variance explained by combining points (`prop_var_base.mark_point()`) with lines (`prop_var_base.mark_line()`). Store the result as `cum_var`.
    - Apply the color you chose for the axis title to the points and lines.

1. Repeat the previous step for the cumulative variance explained.
    - Apply the color you chose for the axis title to the points and lines.

1. Layer the plots together using `alt.layer(l1, l2).resolve_scale(y = 'independent')`.

In [67]:
```python
# encode component axis only as base layer
base = alt.Chart(pcvars).encode(
    x = 'Component')

# make a base layer for the proportion of variance explained
prop_var_base= base.encode(
    y = alt.Y('Proportion of variance explained',
              axis = alt.Axis(titleColor = 'orange'))
)

# make a base layer for the cumulative variance explained
cum_var_base = base.encode(
    y = alt.Y('Cumulative variance explained', axis = alt.Axis(titleColor =
)

# add points and lines to each base layer
prop_var = prop_var_base.mark_line(stroke = 'orange') + prop_var_base.mark_p
cum_var = cum_var_base.mark_line() + cum_var_base.mark_point()

# layer the layers
var_explained_plot = alt.layer(prop_var, cum_var).resolve_scale(y = 'indepen

# display
var_explained_plot
```

Out[67]:



(v) How many PCs?

How many principal components capture a high proportion of covariation in relative abundances? How much total variation do these explain together?

Around 60% of variation is captured collectively by the first 3 components.

Q2 (c). Interpreting component loadings

Now that you've performed the calculations for PCA, you can move on to the fun part: figuring out what they say about the data!

The first step in this process is to examine the loadings. Each principal component is a linear combination of the relative abundances by taxon, and the loadings tell you *how* that combination is formed; the loadings are the linear combination coefficients, and thus correspond to the weight of each taxon in the corresponding principal component. Some useful points to keep in mind:

- a high loading value (negative or positive) indicates that a variable strongly influences the principal component;
- a negative loading value indicates that
    - increases in the value of a variable *decrease* the value of the principal component
    - and decreases in the value of a variable *increase* the value of the principal component;
- a positive loading value indicates that
    - increases in the value of a variable *increase* the value of the principal component
    - and decreases in the value of a variable *decrease* the value of the principal component;
- similar loadings between two or more variables indicate that the principal component reflects their *average*;
- divergent loadings between two sets of variables indicates that the principal component reflects their *difference*.

(i) Extract the loadings from `pca`.

Store the loadings for the first two principal components (called `.components_` in the PCA output) in a dataframe named `loading_df`. Name the columns `PC1` and `PC2`, and append a column `Taxon` with the corresponding variable names, and print the resulting dataframe.

In [68]:
```
# store the loadings as a data frame with appropriate names
loading_df = pd.DataFrame(pca.components_).transpose().rename(
    columns = {0: 'PC1', 1: 'PC2'} # add entries for each selected component
).loc[:, ['PC1', 'PC2']] # slice just components of interest

# add a column with the variable name Taxon
loading_df['Taxon'] = pcdata_raw.columns

# print
loading_df.head()
```

Out[68]:

	PC1	PC2	Taxon
0	-0.521378	-0.157880	A_curv
1	-0.194520	0.001639	A_octon
2	-0.373815	-0.477144	ActinSpp
3	0.611563	-0.181503	A_nodul
4	-0.041199	-0.548427	CoscinSpp

In [69]:
```
grader.check("q2_c_i")
```

Out[69]:

q2_c_i passed!

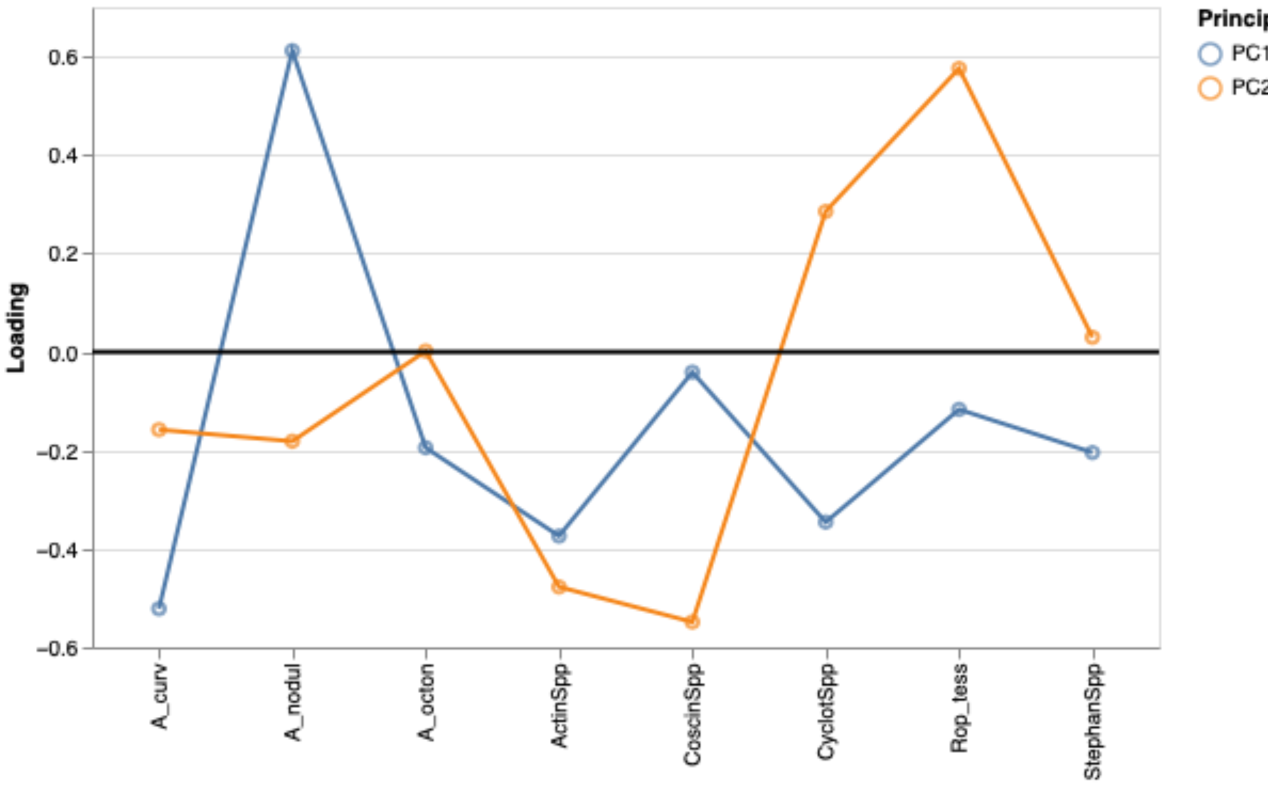
(ii) Loading plots

Construct a line-and-point plot connecting the loadings of the first two principal components. Display the value of the loading on the y axis and the taxa names on the x axis, and show points indicating the loading values. Distinguish the PC's by color, and add lines connecting the loading values for each principal component.

Hint: you will need to first melt `loading_df` to long form with three columns -- the taxon name, the principal component (1 or 2), and the value of the loading.

In [70]:
```
# melt from wide to long
loading_plot_df = loading_df.melt(
    id_vars = 'Taxon',
    var_name = 'Principal Component',
    value_name = 'Loading'
)

# create base layer with encoding
base = alt.Chart(loading_plot_df).encode(
    x = alt.X('Taxon', title = ''),
    y = 'Loading',
    color = 'Principal Component'
)

#  store horizontal line at zero
rule = alt.Chart(pd.DataFrame({'Loading':[0]},index = [0])).mark_rule().encc

# layer points + lines + rule to construct loading plot
loading_plot = base.mark_point() + base.mark_line() + rule

# show
loading_plot.properties(width = 500)
```

Out[70]:



(iii) Interpret the first principal component.

Indicate the following:

- which taxa are up-weighted and which are down-weighted;
- how you would describe the principal component in context (*e.g.*, average abundance among a group, differences in abundances, etc.);
- and how you would interpret a positive value of the PC versus a negative value of the PC in terms of diatom communnity composition.

For principal component 1 the only taxa that is positive is A_nodul meaning this tax is up-weighted while all other taxas are negative so they are down-weighted. An increase in the relative abundance for A_nodul increases the value of the first principle component while for the rest of the taxa, an increase in their value decreases the value of the first principle component. Based on this information the data represents whether or not there is an abundance of Azpeitia nodulifer. Therefore, a positive value of PC1 abundance of Azpeitia nodulifer and a negative value would indicate an abundance of another taxa.

(iv) Interpret the second principal component.

Indicate the following:

- which taxa are up-weighted and which are down-weighted;
- how you would describe the principal component in context (*e.g.*, average abundance among a group, differences in abundances, etc.);
- and how you would interpret a positive value of the PC versus a negative value of the PC in terms of diatom community composition.

For Principal component 2 the taxas that are up-weighted are CycloSpp, Rop_tess, and StephanSpp because they are positive while all other taxas are down-weighted because they are negative. Based on this information we can say this data represents whether or not there is an abundance of different taxas in the community, rather than indicating whether or not there is a single taxon in the community. An increase in the relative abundance for CyclotSpp, Rob_tess or StephanSpp increases the value of the second principle component while for the rest of the Taxa, an increase in their value decreases the value of the second principle component. Amongst the rest of the Taxa, the loadings are all in the negative values and closer to one another which suggests that the principle component reflects their average. A positive value of the PC reflects a higher relative presences of CyclotSpp, Rob_tess and StephanSpp while a negative value of the PC represents a higher relative presence of the other taxa.

Q2 (d). Visualizing community composition

Take a moment to recall that there was a shift in the *A. nodulifer* abundance before and after around 11,000 years ago, which roughly corresponded to a major transition in the earth's climate.

Well, you can now use PCA to investigate whether not just individual abundances but *community composition* may have shifted around that time. To that end, let's think of the principal components as 'community composition indices':

- consider PC1 a nodulifer/non-nodulifer community composition index; and
- consider PC2 a complex community composition index.

A pattern of variation or covariation in the principal components can be thought of as reflecting a particular ecological community composition dynamic -- a way that community composition varies throughout time. Here you'll look for distinct patterns of variation/covariation before and after 11,000 years ago via an exploratory plot of the principal components.

(i) Project the centered and scaled data onto the first two component directions.

This sounds a little more complicated than it is -- all that means is compute the values of the principal components for each data point.

Create a dataframe called `projected_data` containing just the first two principal components as two columns named `PC1` and `PC2`, and two additional columns with the Age and Depth variables.

Print the first four rows of `projected_data`.

In [71]:
```
# project pcdata onto first two components; store as data frame
projected_data = pd.DataFrame(
    pca.fit_transform(pcdata)).iloc[:, 0:2].rename(columns = {0: 'PC1', 1: '

# add index and reset
projected_data.index = pcdata.index
projected_data = projected_data.reset_index()

#project_data[['Depth','Age']] = diatoms[['Depth', 'Age']]

# print first four rows
projected_data.head()
```

Out[71]:

	Depth	Age	PC1	PC2
0	0.00	1.33	-0.294522	-0.633176
1	0.05	1.37	-0.554702	-0.618875
2	0.10	1.42	-0.307745	-2.050236
3	0.15	1.46	-1.771066	1.637274
4	0.20	1.51	-0.292806	0.259430

In [72]:
```
grader.check("q2_d_i")
```

Out[72]:

q2_d_i passed!

(ii) Construct a scatterplot of PC1 and PC2 by age indicator.

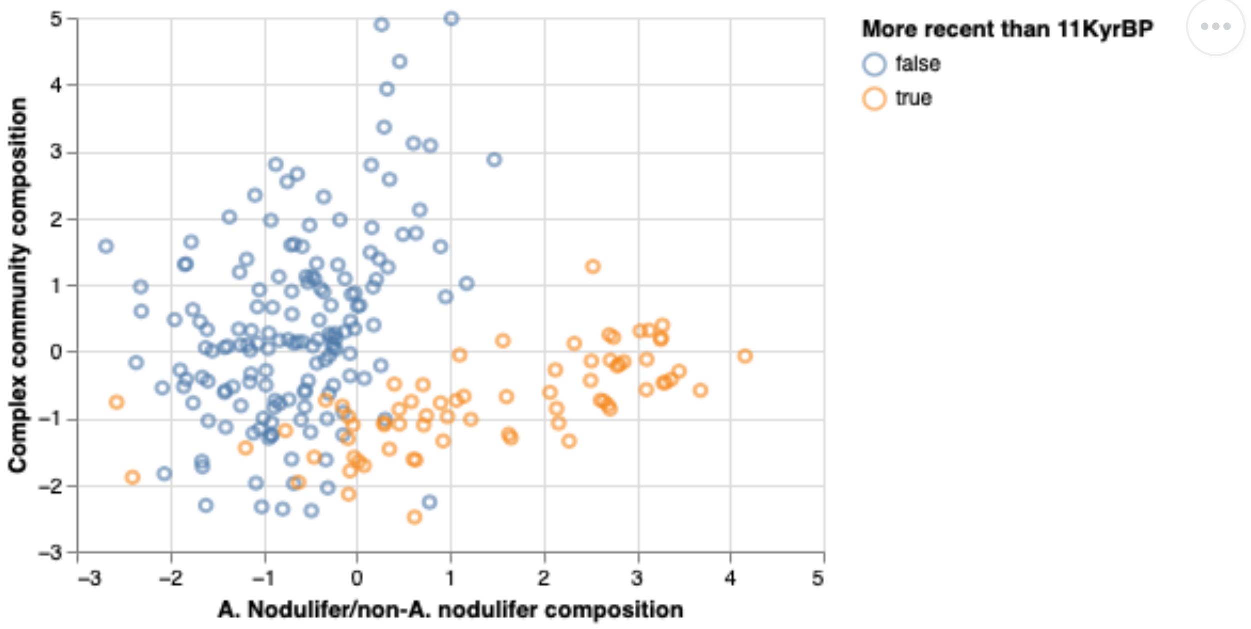
Follow these steps to construct a scatterplot of the principal components.

1. Create an Altair chart based on `projected_data` and use `.transform_calculate(...)` to define a variable `since_11KyrBP` that indicates whether `Age` is older than 11,000 years. Store the result as `base`.
2. Modify `base` to add points with the following encodings.
   - Pass PC1 to the `X` encoding channel and title the axis 'A. Nodulifer/non-A. nodulifer composition'.
   - Pass PC2 to the `Y` encoding channel and title the axis 'Complex community composition'.
   - Pass the variable you created in step 1. to the `color` encoding channel and title it 'More recent than 11KyrBP'. Store the result as `scatter`.

Show the scatterplot once you complete these steps.

In [81]:
```
# base chart
base = alt.Chart(projected_data).transform_calculate(since_11KyrBP = 'datum.

# data scatter
scatter = base.mark_point(opacity = 0.6).encode(
    x = alt.X('PC1:Q', title = 'A. Nodulifer/non-A. nodulifer composition'),
    y = alt.Y('PC2:Q', title = 'Complex community composition'),
    color = alt.Color('since_11KyrBP:N', title = 'More recent than 11KyrBP')
).properties(width = 350, height = 250)

# show
scatter
```

Out[81]:



(iii) Add top and side panels with KDE curves.

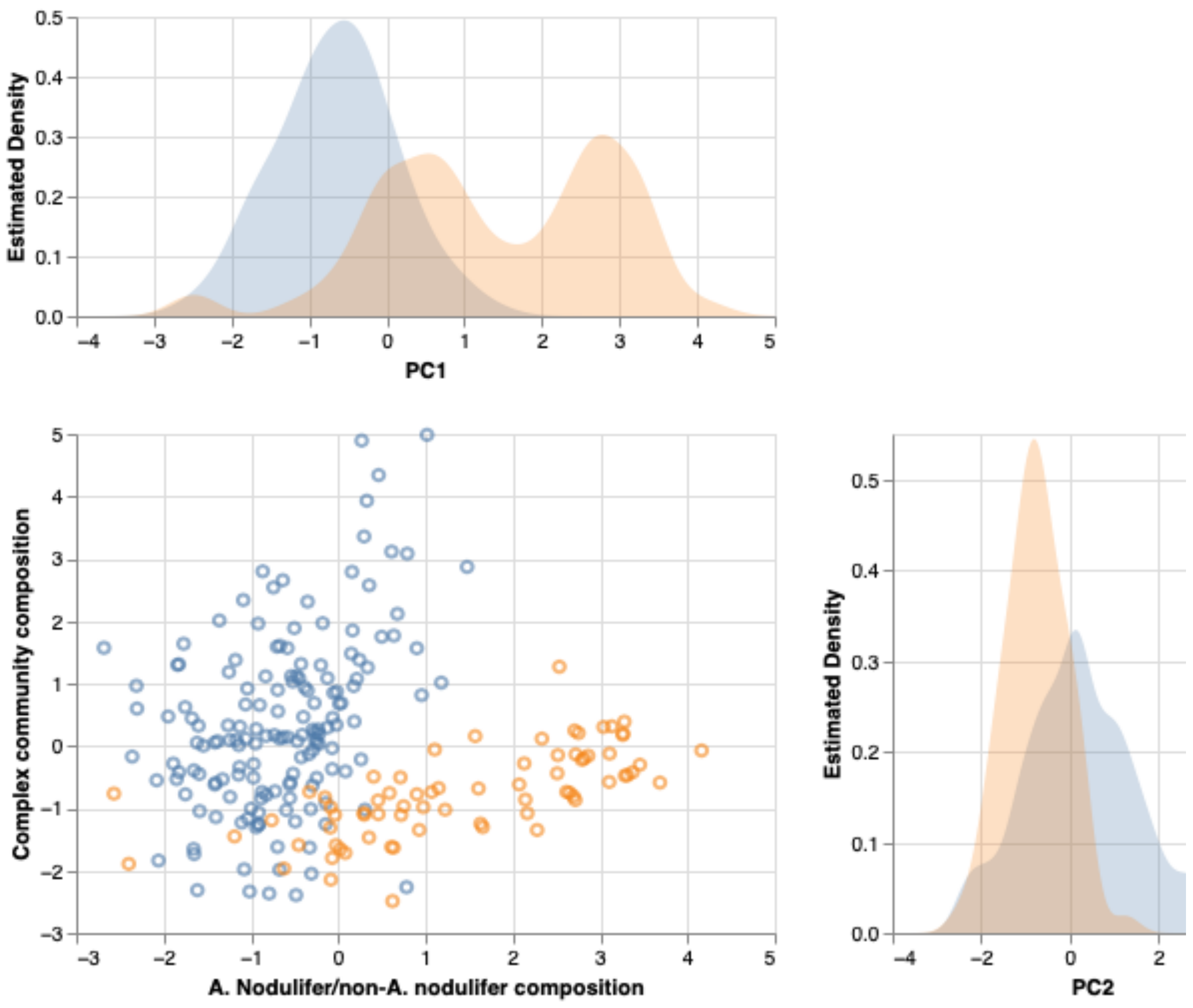
Construct plots of kernel density estimates for each principal component conditional on age being older than 11,000 years:

- modify `base` to create a `top_panel` plot with the KDE curves for PC1, with color corresponding to the age indicator from the `.transform_calculate(...)` step in making the base layer;
- modify `base` again to create a `side_panel` plot with the KDE curves for PC2, rotated 90 degrees relative to the usual orientation (flip the typical axes), and with color corresponding to the age indicator from the `.transform_calculate(...)` step in making the base layer.

Then, resize these panels appropriately (top should be thin, side should be narrow), and use Altair's faceting operators `&` (vertical concatenation) and `|` (horizontal concatenation) to combine them with your scatterplot.

```
In [87]: # construct upper panel (kdes for pc1)
top_panel = base.transform_density(
    density = 'PC1',
    groupby = ['since_11KyrBP'],
    as_ = ['PC1', 'Estimated Density'],
    bandwidth = .3,
    extent = [-4, 5],
    steps = 1000
).mark_area(opacity = 0.25).encode(
    x = 'PC1:Q',
    y = 'Estimated Density:Q',
    color = alt.Color('since_11KyrBP:N')
).properties(width = 350,height = 150)

# construct side panel (kdes for pc2)
side_panel = base.transform_density(
    density = 'PC2',
    groupby = ['since_11KyrBP'],
    as_ = ['PC2', 'Estimated Density'],
    bandwidth = .3,
    extent = [-4, 5],
    steps = 1000
).mark_area(opacity = 0.25).encode(
    x = 'PC2:Q',
    y = 'Estimated Density:Q',
    color = alt.Color('since_11KyrBP:N')
).properties(width = 200, height = 250)



# facet
top_panel & (scatter | side_panel )
```

Out[87]:



3. Communicating results

Take a moment to review and reflect on the results of your analysis in the previous parts. Think about how you would describe succinctly what you've learned from the diatom data.

Q3 (a). Summary

Write a brief paragraph (3-5 sentences) that addresses the following questions by referring to your final plot in Q2 (d).

- How would you characterize the typical ecological community composition of diatom taxa before and after 11,000 years ago?
  - *Hint*: focus on the side and top panels and the typical values of each index in the two time periods.
- Does the variation in ecological community composition over time seem to differ before and after 11,000 years ago?
  - *Hint*: focus on the shape of data scatter.

When age is before 11,000 years ago, the community composition of the diatom taxa has a positive and smaller estimated density of PC1 compared to when age is more recent than 11,000 years ago. On the other hand, when age is more recent that 11,000 years, PC2 has a slight negative value and a high estimated density while PC1 is around the value 0 and has a lower estimated density.

For the variation of PC1, before 11,000 years ago, the spread is much larger than data of more recent years where the spread is much smaller. As for the variation of PC2, the spread for more recent years is larger than the spread of PC1 before 11,000 years ago.

Q3 (b). Further work

What more might you like to know, given what you've learned? Pose a question that your exploratory analysis raises for you.

Answer

For this exploratory analysis, we focused primarily on the covariance between the age and relative abudance over time between different taxa. I am now interested in exploring the relationship between depth and the abundance of these diatoms over time. It is possible that some of the species are more abundant at certain depths than others. I pose the question for further exploration; how abundant are these diatoms at different depths and how does abundance vary at different depth levels?

Submission Checklist

1. Save file to confirm all changes are on disk
2. Run *Kernel > Restart & Run All* to execute all code from top to bottom
3. Save file again to write any new output to disk
4. Select *File > Download as > HTML*.
5. Open in Google Chrome and print to PDF on A3 paper in portrait orientation.
6. Submit to Gradescope

To double-check your work, the cell below will rerun all of the autograder tests.

In [75]:
```
grader.check_all()
```

Out[75]:
```
q0_a results: All test cases passed!

q0_b results: All test cases passed!

q0_c results: All test cases passed!

q1_a results: All test cases passed!

q1_d_i results: All test cases passed!

q2_a_i results: All test cases passed!

q2_a_ii results: All test cases passed!

q2_b_i results: All test cases passed!

q2_b_ii results: All test cases passed!

q2_b_iii results: All test cases passed!

q2_c_i results: All test cases passed!

q2_d_i results: All test cases passed!
```