In [1]:
```
# Initialize Otter
import otter
grader = otter.Notebook("hw3-diatom.ipynb")
```

PSTAT 100 Homework 3

In [2]:
```
import numpy as np
import pandas as pd
import altair as alt
from sklearn.decomposition import PCA
import statsmodels.api as sm
# disable row limit for plotting
alt.data_transformers.disable_max_rows()
# uncomment to ensure graphics display with pdf export
# alt.renderers.enable('mimetype')
```

Out[2]:
```
DataTransformerRegistry.enable('default')
```

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.

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:

- data import and preprocessing
- graphical techniques for visualizing distributions
- multivariate analysis with PCA

Diatom data

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)

Variable	Description	Units
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 [3]:
```
# import diatom data
diatoms_raw = pd.read_csv('data/barron-diatoms.csv')
diatoms_raw.head(5)
```

Out[3]:

	Depth	Age	A_curv	A_octon	ActinSpp	A_nodul	CoscinSpp	CyclotSpp	Rop_tess	Ste
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, and note:

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.

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).

Question 1: 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_examplerows` and display these rows.

In [4]:
```
diatoms_mod1 = diatoms_raw.fillna(0)

# print rows 4 and 5
diatoms_mod1_examplerows = diatoms_mod1.loc[4:5]
print(diatoms_mod1_examplerows)
```

```
   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 [5]:
```
grader.check("q1")
```

Out[5]:

q1 passed! 🚀

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.

Question 2: 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 and check one or two values manually to verify the calculation works as intended.)

In [6]:
```
# 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.divide(sampsize, axis=0)

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

# print
diatoms.head(4)
```

Out[6]:

	Depth	Age	A_curv	A_octon	ActinSpp	A_nodul	CoscinSpp	CyclotSpp	Rop_tess	S
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 [7]:
```
grader.check("q2")
```

Out[7]:

q2 passed! 🎉

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?

Question 3: Time span

What is the geological time span covered by the data? Compute the minimum and maximum age using `.aggregate(...)` and store the result as `min_max_age`. (You will need to use `.aggregate()` to pass the automated tests.)

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 [8]:
```
min_max_age = diatoms['Age'].aggregate(['min', 'max'])
min_max_age
```

Out[8]:
```
min     1.33
max    15.19
Name: Age, dtype: float64
```

In [9]:
```
grader.check("q3")
```

Out[9]:

q3 passed! 💯

Question 4: Time resolution

(i) How are the observations spaced in time?

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`.
2. Make a simple count histogram (no need to manually bin or convert to density scale) with bins of width 0.02 (20 years). Store this as `fig1` and display the figure.
   - Label the x axis 'Time step between consecutive samples'
   - Put the y axis on a square root scale so that bins with one observation are discernible

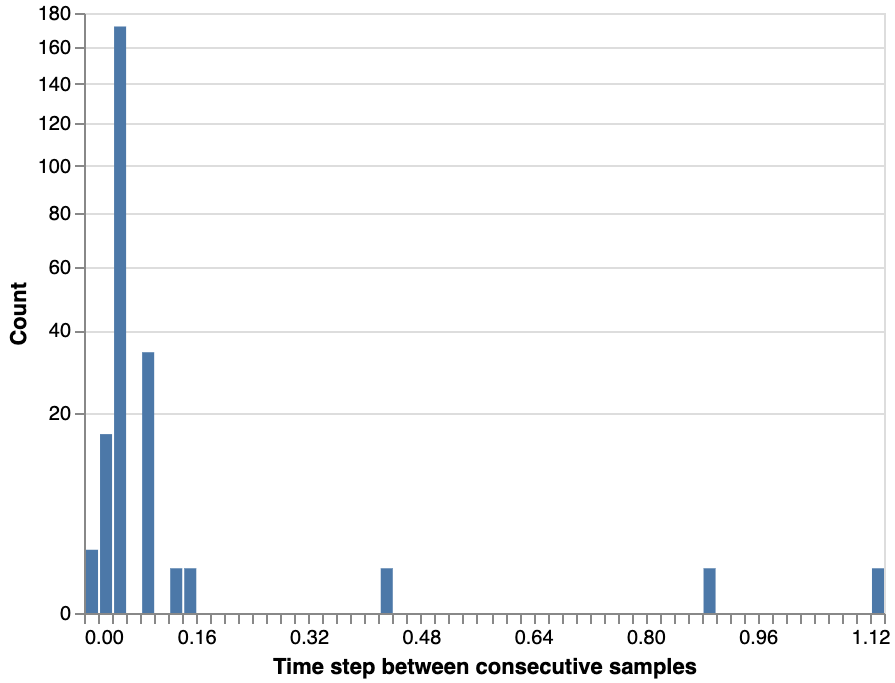
(ii) What is the typical time step in years?

Write your answer based on the count histogram.

Type your answer here, replacing this text.

In [10]:
```
# store differences
diffs = pd.DataFrame({'diff': diatoms.Age.sort_values().diff().loc[1:, ]})

# construct histogram
fig1 = alt.Chart(diffs).mark_bar().encode(
    x=alt.X('diff', bin=alt.Bin(step=0.02), title='Time step between consecu
    y=alt.Y('count()', scale=alt.Scale(type='sqrt'), title='Count')
)

# display
fig1
```

Out[10]:



In [11]:
```
grader.check("q4")
```

Out[11]:

q4 passed! 💯

Univariate explorations

To begin, you'll examine the variation in relative abundance over time for the eight individual taxa, one variable at a time.

Here are some initial questions in this spirit that will help you to hone in and develop more focuesed 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.

Question 5: Summary statistics

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

1. Drop the depth and age variables before performing the aggregation.
2. Use `.transpose()` to ensure that the table is rendered in long form (8 rows by 2 columns rather than 2 columns by 8 rows).
3. Store the resulting dataframe as `diatom_summary` display.

In [12]:
```
diatoms_drop = diatoms.drop(columns=['Depth', 'Age'])
diatom_summary = diatoms_drop.aggregate(['mean', 'std']).transpose()

# print the dataframe
print(diatom_summary)
```

```
                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 [13]:
```
grader.check("q5")
```

Out[13]:

q5 passed! 💯

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

Question 6: Visualizing summary statistics

(i) Create a plot of the average relative abundances and their variation over time.

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.
3. 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'.
4. 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).

5. Layer the plots: `means + bars`. Store the result as `fig2` and display the figure.

It may help to have a look at this example.

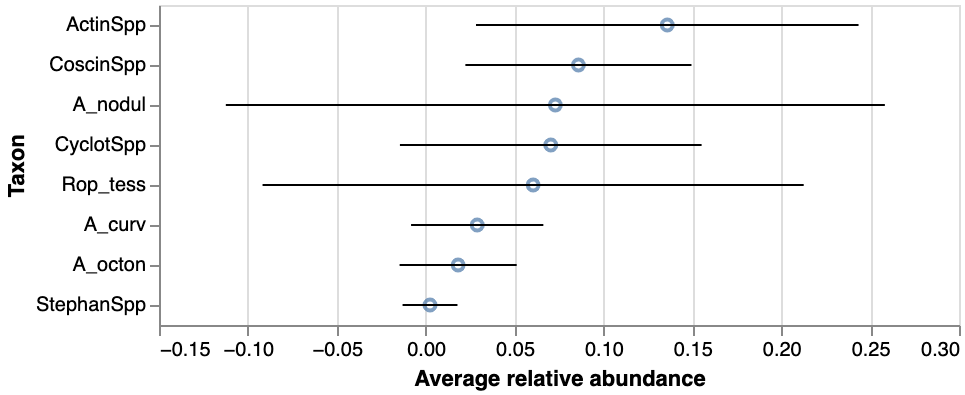
(ii) Based on the figure, answer the following questions.

1. Which taxon is most abundant on average over time?
2. Which taxon is most rare on average over time?
3. Which taxon varies most in relative abundance over time?

Type your answer here, replacing this text.

In [14]:
```
# reset index
plot_df = diatom_summary.reset_index()

# create base chart
base = alt.Chart(plot_df).encode(
    y=alt.Y('index:O', sort={'field': 'mean', 'order': 'descending'}, title=
)

# create point plot
means = base.mark_point().encode(
    x=alt.X('mean:Q', title='Average relative abundance')
)

# create bar plot
bars = base.transform_calculate(
    lwr='datum.mean - 2 * datum.std',
    upr='datum.mean + 2 * datum.std'
).mark_errorbar().encode(
    x=alt.X('lwr:Q', title='Average relative abundance'),
    x2=alt.X2('upr:Q')
)

# layer
fig2 = (means + bars)

# display
fig2
```

Out[14]:



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.

Question 7: Distribution of *Azpeitia nodulifer* abundance over time

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

Use the `diatoms` dataframe and a bin width of 0.03. Be sure to choose an appropriate kernel and smoothing bandwidth. Store the result as `fig3`.

(ii) Answer the following questions in a few sentences.

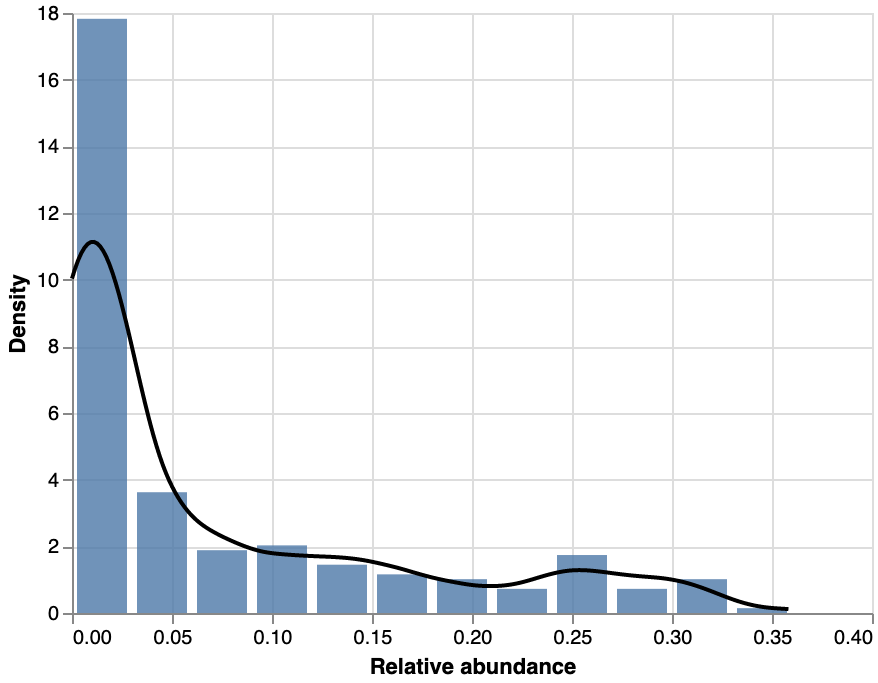
- Which values are common?
- Which values are rare?
- How spread out are the values?
- Are values spread evenly or irregularly?

Type your answer here, replacing this text.

In [15]:
```
# density scale histogram
hist = alt.Chart(diatoms).transform_bin(
    as_ = 'bin',
    field = 'A_nodul',
    bin = alt.Bin(step = 0.03)
).transform_aggregate(
    Count = 'count()',
    groupby = ['bin']
).transform_calculate(
    Density = 'datum.Count/(0.03*230)',
    binshift = 'datum.bin + 0.015'
).mark_bar(size = 25, opacity = 0.8).encode(
    x = alt.X('binshift:Q', title = 'Relative abundance', scale = alt.Scale(
    y = 'Density:Q'
)
```

```
# compute density estimate
kde = sm.nonparametric.KDEUnivariate(diatoms.A_nodul)
kde.fit(kernel = 'triw', bw = 0.05, fft = False, cut = 0)
kde_df = pd.DataFrame({'A_nodul': kde.support, 'Density': kde.density})


# plot kde
smooth = alt.Chart(kde_df).mark_line(color = 'black').encode(x = 'A_nodul',

# layer
fig3 = hist + smooth

# display
fig3
```

Out[15]:



> **Comment:** 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 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 that *both* captured the shape of the histogram near zero *and* away from zero, regardless of the choice of kernel. The key difficulty with zero inflated data is that no common probability distribution fits the data well. This requires the use of mixtures as probability models, which are challenging to incorporate into common statistical models.

There was a transition between geological epochs 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).

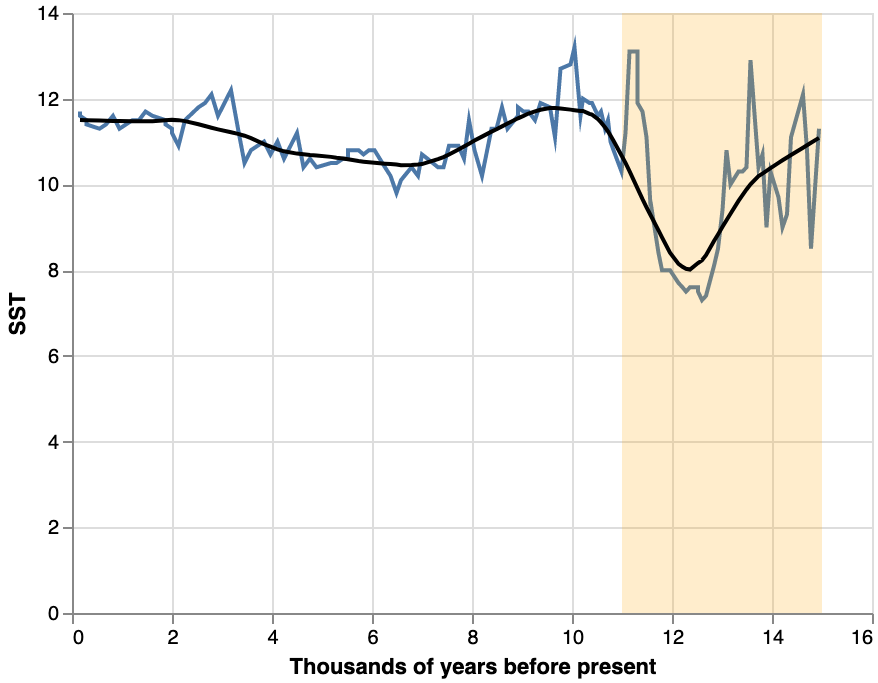
This fluctuation can be seen from temperature reconstructions. Below is a plot of sea surface temperature reconstructions off the coast of Northern California. Data come from the following source:

> 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 [16]:
```
# 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[16]:



Question 8: 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.

Use the Altair implementation of Gaussian KDE:

1. Use `.transform_caluculate(...)` to calculate an indicator variable, `pleistocene`, 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 `pleistocene`
3. Plot the KDEs distinguished by color; give the color legend the title 'Before 11KyrBP' and store the plot as `kdes`.
4. Add a shaded area beneath the KDE curves. Adjust the opacity of the area to your liking.

Store the result as `fig4` and display the figure.

(ii) Does the distribution seem to change between epochs? If so, how?

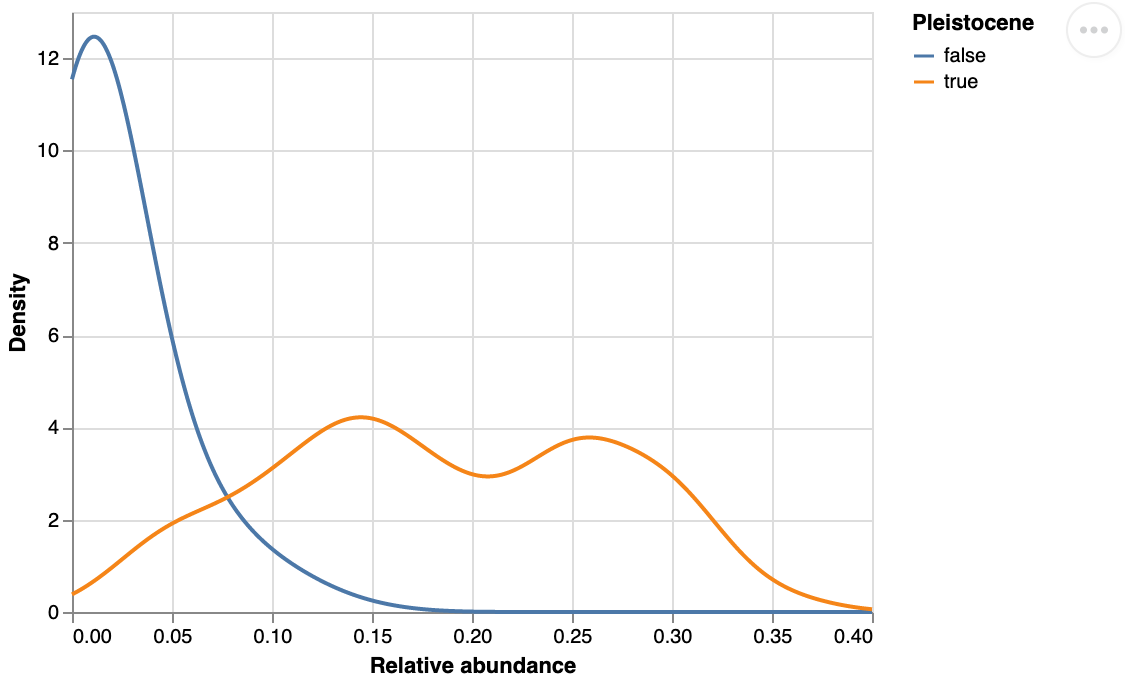
Answer based on the figure in a few sentences.

The relative abundances are low during the end of the pleistcene, but the relative abundance is low. During the shift the abundances are distributed more normally

between 0.025 and 0.375.

In [17]:
```
# Compute KDEs
kdes = alt.Chart(diatoms).transform_calculate(
    pleistocene = 'datum.Age > 11'
).transform_density(
    density = 'A_nodul',
    groupby = ['pleistocene'],
    as_ = ['Relative abundance', 'Density'],
    bandwidth = 0.025,
    extent = [0, 0.4],
    steps = 500
).mark_line(color = 'black').encode(
    x = 'Relative abundance:Q',
    y = 'Density:Q',
    color = alt.Color('pleistocene:N', title = 'Pleistocene')
)


# Create the plot
fig4 = kdes

# Display the figure
fig4
```

Out[17]:



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

compare variation in community composition among *all* eight taxa during the late Pleistocene and Holocene epochs.

Question 9: Pairwise correlations in relative abundances

(i) Compute the pairwise correlations between relative abundances and make a heatmap of the correlation matrix.

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

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`.
2. 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.

(ii) How does *A. nodulifer* seem to vary with the other taxa, if at all?

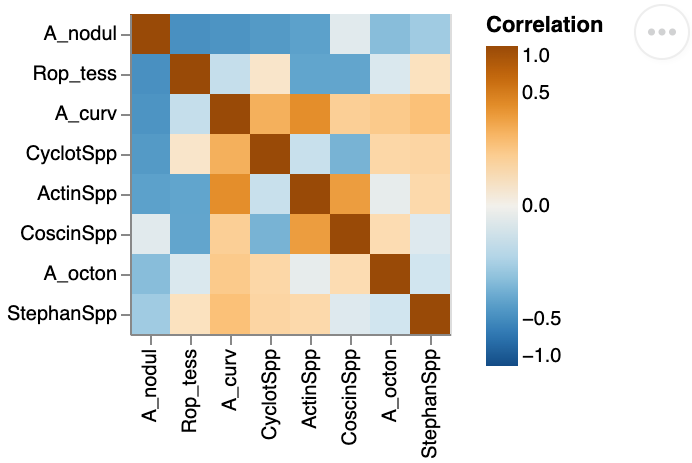
Answer in a few sentences based on the heatmap.

Type your answer here, replacing this text.

In [18]:
```
corr_mx = diatoms.set_index(['Depth', 'Age']).corr()

# 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 heatmap
fig5 = 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',
                                        domain = (-1, 1),
```

```
                                                type = 'sqrt'),
                        legend = alt.Legend(tickCount = 5))
)


# display
fig5
```

Out[18]:



In [19]:
```
grader.check("q9")
```

Out[19]:

q9 passed! 🍀

Question 10: 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`.

(ii) Compute the principal components.

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

(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 = ...)`.

For this part you do not need to show any specific output.

(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 = ...))`.
3. 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 = ...))`.
4. 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.
5. Repeat the previous step for the cumulative variance explained.
    - Apply the color you chose for the axis title to the points and lines.
6. Layer the plots together using `alt.layer(l1, l2).resolve_scale(y = 'independent')`.

Store the result as `fig6` and display the figure.

(v) How many PCs should be used?

Propose an answer based on the variance explained plots and indicate how much total variation your proposed number of components capture jointly.

Type your answer here, replacing this text.

In [20]:
```
## (i) center and scale data

# 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()
```

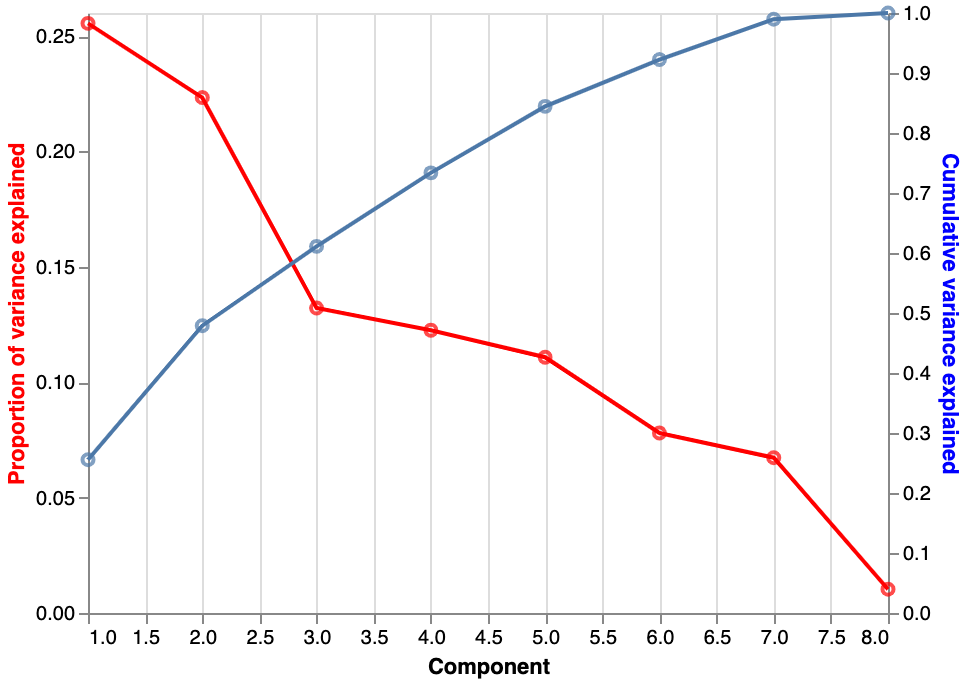
In [21]:
```
## (ii) compute pcs

pca = PCA(8)
pca.fit(pcdata)
```

Out[21]:
```
PCA
PCA(n_components=8)
```

In [22]:
```
# Retrieve variance info
explained_variance_ratio = pca.explained_variance_ratio_

# Store proportion of variance explained as a dataframe
pcvars = pd.DataFrame({'Proportion of variance explained': explained_varianc

# Add component number as a new column
pcvars['Component'] = np.arange(1, len(explained_variance_ratio) + 1)

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

In [23]:
```
# Encode component axis only as the 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='re
)

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

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

# Layer the layers
fig6 = alt.layer(prop_var, cum_var).resolve_scale(y='independent')

# Display the figure
fig6
```

Out[23]:



In [24]:
```
grader.check("q10")
```

Out[24]:

q10 passed! 💯

Now that you've performed the calculations for PCA, you can move on to the fun/difficult 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*.

Question 11: Interpreting component loadings

(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.

(ii) Construct 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. Store the result as `fig7` and display the figure -- you may need to resize for better readability.

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.

(iii) Interpret the first principal component.

In a few sentences, answer the following questions.

1. Which taxa are up-weighted and which are down-weighted in this component?
2. How would you describe the principal component in context (*e.g.*, average abundance among a group, differences in abundances, etc.)?
3. How would you interpret a larger value of the PC versus a smaller value of the PC in terms of diatom communnity composition?

(iv) Interpret the second principal component.

Answer the same questions for component 2.

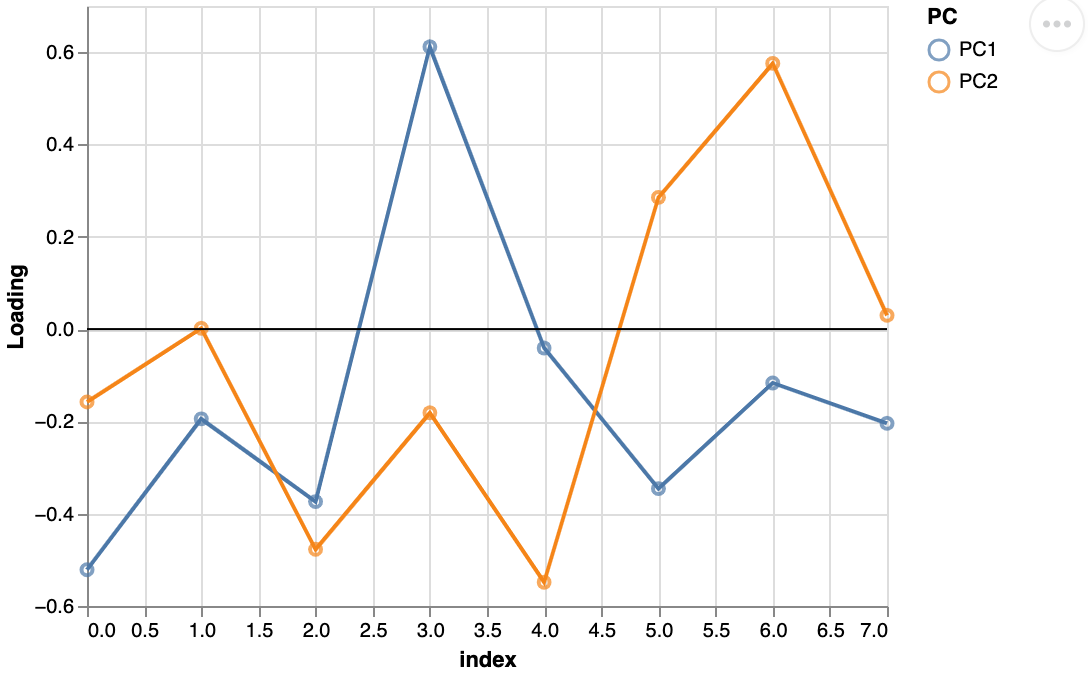
PC1 is weighted up by A. Node and weighted down by others. The principle component is the difference in abundance. THe positive values indicate more amounts of A. Node. In PC2 the up weight is by Cyclotelle and R. Tessel and down weighted by Conscinodiscus and Actinoptychus

In [25]:
```
## (i) retrieve loadings
# store the loadings as a data frame with appropriate names
loading_df = pd.DataFrame(pca.components_).transpose().rename(columns = {0:
```

```
# add a column with the taxon names
loading_df['Taxon'] = pcdata.columns.values
```

In [26]:
```
## (ii) construct loading plots
# melt from wide to long
# Melt from wide to long
loading_plot_df = loading_df.reset_index().melt(
    id_vars='index',
    value_vars=['PC1', 'PC2'],
    var_name='PC',
    value_name='Loading'
)

# Create base layer with encoding
base = alt.Chart(loading_plot_df).encode(
    x='index',
    y='Loading',
    color='PC'
)

# Store horizontal line at zero
rule = alt.Chart(pd.DataFrame({'Loading': [0]})).mark_rule().encode(y='Loadi

# Layer points + lines + rule to construct loading plot
fig7 = alt.layer(
    base.mark_point(),
    base.mark_line(),
    rule
)

# Display the figure
fig7
```

Out[26]:



In [27]:
```
grader.check("q11")
```

Out[27]:

q11 passed! 🙌

Recall that there was a shift in climate around 11,000 years ago, and *A. nodulifer* abundances seemed to differ before and after the shift.

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.

Question 12: Visualizing community composition shift

(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.

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

Construct a scatterplot of the principal components with observations colored according to whether they occurred in the Pleistocene or Holocene epoch. Store the result as `fig8` and display the figure.

(iii) Comment on the plot: does there appear to be any change in community structure?

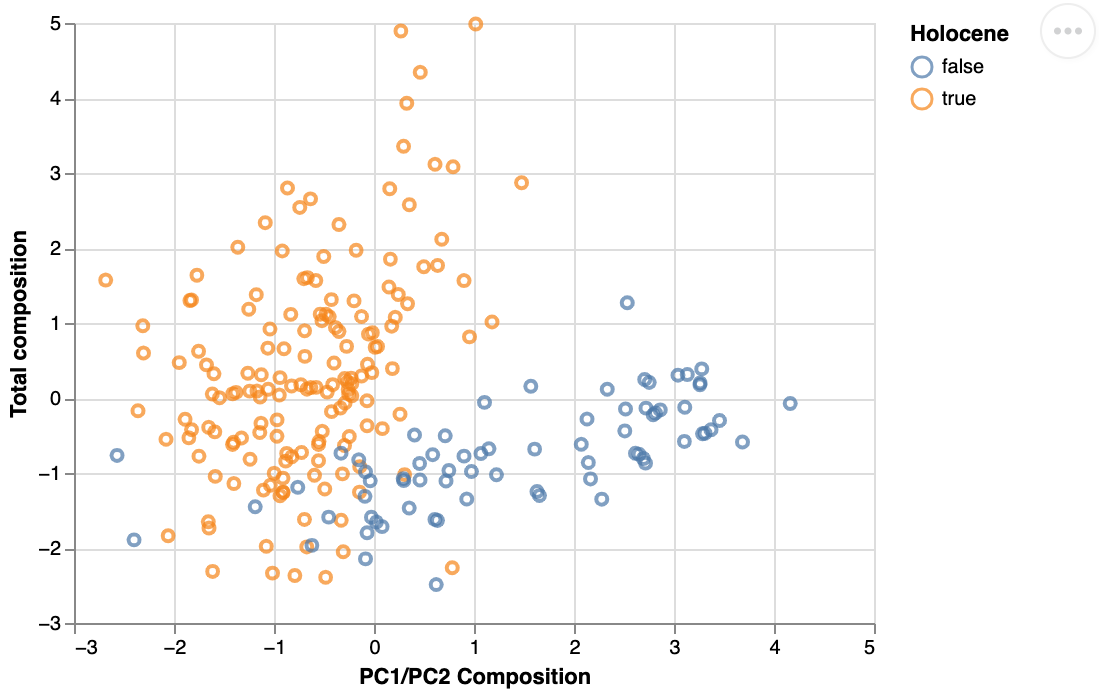
Answer in a few sentences.

There is a change for each of the epochs but the variables are responsable for the changes.

In [28]:
```
## (i) project pcdata onto first two components; store as data frame

# Retrieve principal component scores for PC1 and PC2
projected_data = pd.DataFrame(pca.fit_transform(pcdata)).iloc[:, 0:2].rename

# Adjust index
projected_data.index = pcdata.index
projected_data = projected_data.reset_index()
```

In [29]:
```
## (ii) construct scatterplot
fig8 = alt.Chart(projected_data).transform_calculate(
    holocene='datum.Age < 11'
).mark_point().encode(
    x=alt.X('PC1:Q', title='PC1/PC2 Composition'),
    y=alt.Y('PC2:Q', title='Total composition'),
    color=alt.Color('holocene:N', title='Holocene')
)

# display
fig8
```

Out[29]:



In [30]:
```
grader.check("q12")
```

Out[30]:

q12 passed! 🌈

(Optional) Question 13: Multi-panel visualization

Sometimes it's helpful to see marginal distributions together with a scatterplot. Follow the steps below to create a multi-panel figure with marginal density estimates appended to the projected scatter from the previous question.

1. Create an Altair chart based on `projected_data` and use `.transform_calculate(...)` to define a variable `holocene` 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 'Holocene'.

Store the result as `scatter`.

3. 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.
4. 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.

Store the result as `fig9` and display the figure.

In [31]:
```
# make base layer
...

# data scatter
...

# construct side panel (kdes for pc2)
...

# facet
fig9 = ...

# display
fig9
```

Out[31]:
```
Ellipsis
```

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.

Question 14: Summary

Write a brief paragraph (3-5 sentences) that addresses the following questions by referring to your results above.

- 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.

The typical community is high in abundance of the A. node tax and low abundance of the other ones, but this in the past of 11,000 years. After 11,000 years the tesselata and cyclotella abundance is more popular. The variation used to be controlled by the A. nod now that it is lower the composition is much higher of everything els.e

Question 15: 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

Has there been any major even in the past 11,000 years that caused a ecological chang

Submission

1. Save the notebook.
2. Restart the kernel and run all cells. (**CAUTION**: if your notebook is not saved, you will lose your work.)
3. Carefully look through your notebook and verify that all computations execute correctly and all graphics are displayed clearly. You should see **no errors**; if there are any errors, make sure to correct them before you submit the notebook.
4. Download the notebook as an `.ipynb` file. This is your backup copy.
5. Export the notebook as PDF and upload to Gradescope.

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

In [32]:
```
grader.check_all()
```

Out[32]:
```
q1 results: All test cases passed!

q10 results: All test cases passed!

q11 results: All test cases passed!

q12 results: All test cases passed!

q2 results: All test cases passed!

q3 results: All test cases passed!

q4 results: All test cases passed!

q5 results: All test cases passed!

q9 results: All test cases passed!
```