Protocol

Nondestructive microbial discrimination using single-cell Raman spectra and random forest machine learning algorithm

Raman microspectroscopy is a powerful tool for obtaining biomolecular information from single microbial cells in a nondestructive manner. Here, we detail steps to discriminate prokaryotic species using single-cell Raman spectra acquisitions followed by data preprocessing and random forest model tuning. In addition, we describe the steps required to evaluate the model. This protocol requires minimal preprocessing of Raman spectral data, making it accessible to non-spectroscopists, yet allows intuitive visualization of feature importance.

Publisher’s note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Nanako Kanno, Shingo Kato, Moriya Ohkuma, Motomu Matsui, Wataru Iwasaki, Shinsuke Shigeto

nkanno@kwansei.ac.jp (N.K.)
shigeto@kwansei.ac.jp (S.S.)

Highlights

Accurate discrimination of bacterial and archaeal species at the single-cell level

Detailed description of how to measure and analyze single-cell Raman spectra

Python codes for random forest model construction and features importance extraction

Kanno et al., STAR Protocols 3, 101812
December 16, 2022 © 2022 The Author(s).
https://doi.org/10.1016/j.xpro.2022.101812
Protocol
Nondestructive microbial discrimination using single-cell Raman spectra and random forest machine learning algorithm

Nanako Kanno,1,5,* Shingo Kato,2 Moriya Ohkuma,2 Motomu Matsui,3,4 Wataru Iwasaki,3,4 and Shinsuke Shigeto1,6,*

1Department of Chemistry, School of Science, Kwansei Gakuin University, Sanda, Hyogo 669-1330, Japan
2Japan Collection of Microorganisms, RIKEN BioResource Research Center, Tsukuba, Ibaraki 305-0074, Japan
3Department of Biological Sciences, Graduate School of Science, The University of Tokyo, Bunkyo-ku, Tokyo 113-0032, Japan
4Department of Integrated Biosciences, Graduate School of Frontier Sciences, The University of Tokyo, Kashiwa, Chiba 277-0882, Japan
5Technical contact
6Lead contact
*Correspondence: nkanno@kwansei.ac.jp (N.K.), shigeto@kwansei.ac.jp (S.S.)
https://doi.org/10.1016/j.xpro.2022.101812

SUMMARY
Raman microspectroscopy is a powerful tool for obtaining biomolecular information from single microbial cells in a nondestructive manner. Here, we detail steps to discriminate prokaryotic species using single-cell Raman spectra acquisitions followed by data preprocessing and random forest model tuning. In addition, we describe the steps required to evaluate the model. This protocol requires minimal preprocessing of Raman spectral data, making it accessible to non-spectroscopists, yet allows intuitive visualization of feature importance. For complete details on the use and execution of this protocol, please refer to Kanno et al. (2021).

BEFORE YOU BEGIN
Background
Despite substantial advances in cell culture methods, the vast majority of microorganisms remain uncultured (Rinke et al., 2013). To address this microbial dark matter, culture-independent single-cell approaches have recently been developed and applied to a wide range of microbial systems regardless of whether they are laboratory or environmental samples. Raman microspectroscopy, which is based on inelastic light scattering from molecules, has been shown to be powerful for studying single microbial cells in a nondestructive manner through their molecular fingerprints (i.e., Raman spectra) (Lorenz et al., 2017). Artificial intelligence technology is also being actively used to analyze microbial data, which can often become very large. Combining these two relatively new technologies will enable accurate discrimination of diverse microbial species simply by using Raman spectra of microbial cells (Ho et al., 2019; Lu et al., 2020). We have demonstrated a machine learning-based, nondestructive discrimination of prokaryotic species at the single-cell level. The algorithm employed here, random forest (RF), has the advantage of readily visualizing the importance of the features that contribute to the discrimination. The protocol below describes step-by-step details for confocal Raman microspectroscopic measurements, data preprocessing, and RF model construction involved in microbial discrimination.

Preparation of microbial cells
© Timing: days–weeks (depending on microbial species)
Prepare microbial cultures in your laboratory or receive them from other researchers and/or microbial culture collections (e.g., ATCC, DSMZ, and JCM). The cultivation time and method will depend considerably on each microbial species.

**Note:** This protocol has been tested with six microorganisms (three bacteria and three archaebacteria; see key resources table) but should in principle be applicable to other microorganisms. The protocol for preprocessing of single-cell Raman spectra will be effective even for uncultured microbial cells in environmental samples, while microbial discrimination will require unsupervised approaches as well as the supervised RF algorithm.

### KEY RESOURCES TABLE

| REAGENT or RESOURCE | SOURCE | IDENTIFIER |
|---------------------|--------|------------|
| **Bacterial and virus strains** | | |
| *Escherichia coli* | JCM | JCM 20135 |
| *Bacillus subtilis* | JCM | JCM 1465<sup>T</sup> |
| *Thermus thermophilus* | JCM | JCM 10941<sup>T</sup> |
| *Thermococcus kodakarenensis* | JCM | JCM 12380<sup>T</sup> |
| *Sulfolobus acidocaldarius* | JCM | JCM 8929<sup>T</sup> |
| *Nitrososphaera viennensis* | JCM | JCM 19564<sup>T</sup> |
| **Chemicals, peptides, and recombinant proteins** | | |
| LB broth, Lennox | Nacala: Tesque | Cat# 20066-95 |
| Bacto tryptic soy broth | Becton, Dickinson and Company | Cat# 211825 |
| Bacto peptone | Becton, Dickinson and Company | Cat# 211677 |
| Bacto yeast extract | Becton, Dickinson and Company | Cat# 212750 |
| **Deposited data** | | |
| Raw and analyzed data | (Kanno et al., 2021); Mendeley Data | https://doi.org/10.17632/8cd34fckgt.1 |
| **Software and algorithms** | | |
| Igor Pro 9.0.0.10 | WaveMetrics | | |
| Python 3.7.6 | Python Software Foundation | | |
| scikit-learn version 0.22.1 | (Pedregosa et al., 2011) | | |
| NumPy | Python community project | | |
| pandas | Sponsored Project of NumFOCUS | | |
| matplotlib | Sponsored Project of NumFOCUS | | |
| JupyterLab or Jupyter Notebook | Project Jupyter | | |
| Python script | This paper | | |
| **Others** | | |
| Glass bottom dish | Matsunami Glass Ind., Ltd | Cat# D11130H |
| 1.5 mL microtube | Sansyo Co., Ltd | Cat# STF-15 |

**Alternatives:** This protocol uses the scikit-learn library in Python to perform RF, but other popular programs such as R and MATLAB can also be used. If you do not have Python installed on your computer, you can download it via Anaconda distribution (https://www.anaconda.com/) together with many useful libraries and development environment (e.g., pandas, NumPy, matplotlib, and JupyterLab and Jupyter Notebook) for machine learning.
MATERIALS AND EQUIPMENT

### Phosphate buffer solution (PBS) (1×)

| Reagent                        | Amount         | Final concentration |
|-------------------------------|----------------|---------------------|
| KH2PO4                        | 0.02 g         | 1.47 mM             |
| KCl                           | 0.02 g         | 2.68 mM             |
| Na2HPO4·12H2O (or Na2HPO4)    | 0.29 g (0.115 g) | 8.1 mM             |
| NaCl                          | 0.8 g          | 136.9 mM            |
| Distilled water               | Up to 100 mL   | N/A                 |
| Total                         | 100 mL         | N/A                 |

Adjust pH to 7.4. Autoclave (121°C for 20 min) and store at 4°C.

Note: Autoclaved PBS can be stored at 4°C for more than six months.

Note: Adjust the pH and/or salt concentration of PBS if cell lysis occurs. Alternatively, try another buffer such as bicarbonate buffer and HEPES buffer.

STEP-BY-STEP METHOD DETAILS

Set up your confocal Raman microscope

⊙ Timing: 1–2 h

This section describes the setting of the confocal Raman microscope you use prior to measurements of single microbial cells. We used a laboratory-built confocal Raman microscope with the 632.8 nm light from a He–Ne laser as the Raman excitation source (Figure 1) (Huang et al., 2011, 2012; Matsuda et al., 2019).

Alternatives: A commercial Raman microscope available from Renishaw, HORIBA, Bruker, WITec, etc. can be used as well. We recommend an inverted microscope because it facilitates bright-field observation of cell suspension dropped in a glass bottom dish.

Note: The same protocol can be applied for 532 nm excitation. Modifications may, however, be required in data preprocessing (see step 11 below), because 532 nm excitation often causes stronger cellular autofluorescence than 632.8 nm excitation.

Δ CRITICAL: The following parameters or components of the confocal Raman microscope (Figure 1) are of particular importance because they substantially affect measured Raman spectra: laser wavelength; an objective lens; a pinhole for confocal detection; and a grating of the spectrometer.

Laser wavelength

The choice of the laser wavelength concerns with the resonance Raman effect and laser-induced damage of the cell. 532 nm excitation is most commonly used, but in this protocol, we adopted 632.8 nm excitation, which falls near the optical window of biology.

Objective lens

A phase-contrast objective is generally used to observe prokaryotic cells. We chose an oil-immersion phase-contrast objective (Nikon, 100×, NA 1.3, CFI Plan Fluor DLL). According to the Rayleigh criterion, the lateral (XY) resolution is given by \(0.612\frac{\lambda}{NA}\), where \(\lambda\) is the wavelength of light and NA is the numerical aperture of the objective. Using this relation, the lateral resolution of our Raman microscope is calculated to be 0.3 μm, while experimentally it is estimated to be ~0.45 μm.
Confocal pinhole
A 100-µm pinhole is used to get rid of out-of-focus light; the resulting depth (Z) resolution is estimated to be ~4.5 µm.

Grating
The use of a 600 grooves/mm grating enables us to record the entire spectral window (approximately 660–3,020 cm\(^{-1}\)) covering both fingerprint and CH-stretching regions with a spectral resolution of ~5 cm\(^{-1}\).

1. Wait typically more than 30 min until the laser output and the temperature of the CCD detector (set to ~70°C) are stabilized. Re-align the optical path if necessary.
2. Set the laser power at the sample point to a desired value (3 mW in the present case) by adjusting a variable neutral-density (ND) filter.

   **Note:** The desired value of the laser power depends on purposes. If you do not care about the viability of the measured cell, you could use a higher power (e.g., ~10 mW (Ho et al., 2019)) and improve the signal-to-noise ratio (S/N) of the spectrum.

3. Set the center wavenumber of the grating in order to determine the wavenumber region to be measured.

   **Note:** The calibrated wavenumber region varies slightly from day to day, resulting in a shift in the horizontal axis of the measured spectra. This shift can be corrected for by using emission lines of a standard neon lamp (model: 6032, Newport) recorded on different days. To make the wavenumber regions as close as possible among different measurements, adjust them so that the peak position of a selected Raman band of the reference material comes at roughly the same CCD pixel.

   **Note:** In principle, do not change the grating setting during the experiment.

**Single-cell Raman measurements**

**Timing:** 1.5 h for 40 cells/species

This section outlines steps to acquire Raman spectra of single cells of a microbial species.
4. Prepare a cell-suspension sample (Figure 2A).
   a. Harvest 200 μL culture of each strain in a 1.5 mL microtube and wash it three times with 200 μL of PBS by centrifugation (e.g., 8,000–10,000 × g, 30–60 s) at room temperature.
   b. Resuspend the pellet in 200 μL of PBS.
   c. Either dilute or concentrate the cell suspension for better microscopic observation of individual cells (Figure 2B, left).

   CRITICAL: When the cell density is too high (e.g., 5 × 10⁷ cells/mL), multiple cells may be laser-trapped simultaneously (see Methods video S1). Adjust the cell density so that there are not too many (Figure 2B, right) or too few cells in the field of view. For example, for E. coli cultured for 14–16 h (about 5 × 10⁹ cells/mL) in LB medium, add a 0.5 μL aliquot to 200 μL of PBS, resulting in a cell density of ~1.3 × 10⁷ cells/mL.

d. Transfer 200 μL of the cell suspension to a glass bottom dish (hole diameter, 14 mm).

5. Randomly select a single prokaryotic cell, capture it via the laser trapping technique (see Figure 3A-i; troubleshooting 1) (Xie et al., 2002), and record its Raman spectrum (see Methods video S2).

   Note: We used laser power of 3 mW at the sample point and an exposure time of 30 s per cell. The total timing depends on the exposure time and how many cells you measure for each species. Troubleshooting 2.

   Note: Make sure that the measured spectrum is free from the glass signal (Figure 3B, red). When a cell very close to the cover glass (see Figure 3A-ii) is trapped and measured, the resulting Raman spectrum (Figure 3C, lower trace) shows a Raman band of glass at ~915 cm⁻¹, which is absent in the one measured at the proper height from the cover glass (Figure 3C, upper trace).

6. Record the PBS spectrum (Figure 3B, blue) 10 times for averaging purposes.
Data preprocessing

@Timing: 1 h

This section describes how to preprocess the recorded Raman spectra prior to training and testing in machine learning. The overall workflow is illustrated in Figure 4, which is adapted with permission from (Kanno et al., 2021).

Alternatives: This protocol uses Igor Pro 9.0.0.10 (WaveMetrics) to perform all of the following data preprocessing, but other data analysis software such as Origin (https://www.originlab.com/) can also be used.

7. Remove spiky artifact caused by cosmic ray, if any.
8. Subtract the PBS spectrum (average of 10 spectra) from each spike-free, single-cell Raman spectrum (Figure 4A).
9. Use emission lines of a standard neon lamp to correct for a daily shift of the horizontal axis (Figure 4B).
   a. For each measurement day, fit a particular emission line to the Lorentzian function and determine the wavenumber of its peak (Figure 5).
   b. Delete (typically a few) data points at both ends of each spectrum so that the peak wavenumbers of the above neon emission line match that of a reference spectrum within \( \pm 0.5 \) pixel that is pre-determined from all of the data from different days.
10. Delete the so-called silent region (approximately 1,800–2,775 cm\(^{-1}\)) and retain the fingerprint (660–1,800 cm\(^{-1}\)) and CH-stretching (2,775–3,020 cm\(^{-1}\)) regions (Figure 4C).

Note: No essential Raman bands appear in the silent region with few exceptions (e.g., C=C and C≡N stretching bands (Yamakoshi et al., 2011; Zhao et al., 2017) and overtone and combination bands (Horiue et al., 2020)).
11. Perform baseline subtraction for the fingerprint and CH-stretching regions separately (Figure 4D).
   a. Install the Baseline Fitting package in Igor (see https://www.wavemetrics.com/project/Baselines for details).
   b. Select 'line' from the drop-down list as the baseline function (Figure 6A).

   CRITICAL: What function is used to fit the baseline could significantly affect the performance of machine learning. Other functions than a linear function are available in this package. However, using a more complex function such as higher-order polynomials may often lead to overfitting. To avoid overfitting, linear approximation is a simple and effective means, although it may not be the best.

   c. Drag on the graph and then click the ‘+/-’ button to select/deselect the region(s) to be fitted. A linear fit will immediately be shown as a blue line (Figure 6A).

   Note: Fit only the higher and lower wavenumber edges of each region, as indicated by grey parts in Figure 4D and light-blue parts in the Graph in Figure 6A.

   d. Click ‘Subtract’ for a selected spectrum and ‘All in One’ for all spectra displayed on the Graph, yielding baseline-subtracted Raman spectra (Figure 6B).

12. Perform vector normalization on each region (Figure 4E), by dividing the intensity at each pixel by the square root of the sum of the squared intensities of the spectrum. Troubleshooting 3.

13. Combine the normalized spectra in the fingerprint and CH-stretching regions to obtain a preprocessed single-cell Raman spectrum (Figure 4F).

Tuning hyper-parameters for random forest model construction

© Timing: 1 h
This section details the procedures for optimizing the parameters needed for constructing RF machine learning models using the preprocessed data (40 spectra per species) in Python (Figure 7A). First, a data file to be imported into Python is created. Subsequently, an optimal set of parameters is sought for by using grid search in Python, with a script in the Jupyter Notebook format, which is available on Zenodo (see key resources table). We used Jupyter Notebook in JupyterLab, a web-based interactive development environment.

14. Prepare a data file containing preprocessed single-cell Raman spectra in text format using Microsoft Excel or other software. An example is shown in Figure 8. The columns ‘waves’ and ‘species’ specify the names of spectra and class labels (i.e., microbial species), respectively.

15. Import RandomForestClassifier and GridSearchCV from the scikit-learn package into Python (refer to the Jupyter notebook ‘GridSearch_10models.ipynb’).

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
import pandas as pd
import numpy as np
```

Figure 5. Determining the peak wavenumber of a neon emission line using curve fitting in Igor

(A) Locate cursors (shown/hidden by using the shortcut key ’Ctrl–I’) on the neon emission spectrum displayed as a Graph in order to define the region to be fitted. Then select ‘Curve Fitting’ from the ‘Analysis’ tab.

(B) In the ‘Function and Data’ tab of the ‘Curve Fitting’ window, select the fitting function (Lorentzian function ‘lor’ in the present case) from the drop-down list.

(C) In the ‘Data Options’ tab of the ‘Curve Fitting’ window, click ‘Cursors’ to define the start and end points of the fit. Subsequently, click ‘Do It’.

(D) Fitted results shown in the Graph (left, red line) and in the commandline (right). The peak wavenumber is given as ‘x0’ in the command line.
16. Import the text data file into Python.
   a. Execute the following command. Type the file name in ‘ ’ with the file extension.

   ```python
   df = pd.read_table('dataset.txt')
   ```

   b. Divide the dataset into ‘species’ (i.e., class label) columns and the others.

   ```python
   data_x = df.drop(['species'], axis=1) # others
   data_y = df['species'] # class labels
   ```

17. To perform N-fold cross-validation, split the dataset into N folds.
Note: N differs depending on the size of the dataset. In the case of 10-fold cross validation (N = 10), which we employed, the dataset is split into 10 folds, and 10 patterns of training (9 folds) and test (1 fold) sets are generated and exported as CSV files. You will find the resulting files as ‘train_01.csv’, ‘test_01.csv’, ..., ‘train_10.csv’, and ‘test_10.csv’ in the working directory. The same number indicates a train/test set.

18. Execute grid search for hyper-parameter optimization with M-fold (e.g., M = 5) cross-validation on each training set.

Note: The hyper-parameters, n_estimators and max_features (Figure 7B), that are frequently adopted in the N models are used to search for the optimal parameters that achieve high classification accuracy in the test datasets.

a. Select a model used for grid search.

```python
from sklearn.model_selection import StratifiedKFold

kfold = StratifiedKFold(n_splits=10, random_state=123, shuffle=True)

ranges = range(1,11)

for i, (train_idx, test_idx) in zip(ranges, kfold.split(data_x, data_y)):
    X_train = data_x.iloc[train_idx, :]
    X_test = data_x.iloc[test_idx, :]
    y_train = data_y.iloc[train_idx]
    y_test = data_y.iloc[test_idx]
    train_concat = pd.concat([y_train, X_train], axis=1)
    test_concat = pd.concat([y_test, X_test], axis=1)
    train_concat.to_csv('train_02.f'+'%02.f'+'%i'+'.csv', index=False)
    test_concat.to_csv('test_02.f'+'%02.f'+'%i'+'.csv', index=False)
```

b. Set parameters for the search (see the example script below). The combinations of the numbers given in [] of 'n_estimators' and 'max_features' are attempted.

```python
search_params = {
    'n_estimators': [10, 100, 200, 300, 500, 700, 1000, 1500],
    'max_features': ['sqrt', 20, 30, 50, 60, 70],
    'random_state': [123],
}
```

Note: If you set too many parameters to search, it will take an enormous amount of time. We recommend that you first attempt a broad search, then narrow it down to the finest details, and search again.
c. Define grid search setting.

```python
gs = GridSearchCV(clf_cv, search_params, cv=5, verbose=True, n_jobs=-1)
```

# -1: using all processors

d. Execute grid search for each dataset.

```python
ranges = range(1,11)
for i in ranges:
    train = pd.read_csv('train_'+'%02.f'%(i)+'.csv')
```
e. Display grid search results for 10 sets as shown in Figure 9.
f. Check ‘n_estimators’ and ‘max_features’ in the result of each set and adopt the most frequently employed parameters.

Note: ‘n_estimators’ specifies the number of decision trees, while ‘max_features’ is one of the parameters that restrict the shape of the decision trees and sets an upper limit of the number of features used for node splitting in each decision tree (Figure 7B).

Note: In steps 17 and 18b, a random number needs to be specified (‘random_state’). This ensures that the same result will be obtained whenever the same training samples and parameters are used.

Optional: In addition to ‘max_features’, there are other parameters, such as ‘max_depth’, ‘min_samples_split’, and ‘min_samples_leaf’ in the scikit-learn library, that restrict the shape of the tree for reducing overfitting.

Optional: For simplicity, dimensionality reduction (Guo et al., 2021) is not employed in this protocol, which may be useful in some cases for reducing the number of features used for training.

Evaluation of random forest discrimination model

© Timing: 1 h

19. Using the optimized hyper-parameters, construct a RF model for each combination of training and test sets and evaluate the constructed model using the test set (Figure 7A). For details, refer to ‘RF_10models.ipynb’.

a. Set the hyper-parameters derived from the above grid search.

```
train_x = train.drop(['species', 'waves'], axis=1)
train_y = train['species']
gs.fit(train_x, train_y)
print(gs.best_estimator_)
```
b. Construct a RF model for each training data and evaluate it against the test data.
c. For each train/test set, output the feature importance and class probability in the CSV format and the confusion matrix, and calculate the accuracy score.
d. Calculate the score of feature importance as the total reduction of the criterion brought by that feature (Gini importance) in each validation. Troubleshooting 4.

```python
clf = RandomForestClassifier(max_features='sqrt', n_estimators=200, random_state=123)
```

Note: An average of the scores over N validations (N = 10 in the present case) is used to represent each feature importance. The means of the feature importance (Figure 10A) and class
probability (Figure 10B) are obtained from the exported data using Microsoft Excel or similar spreadsheet software.

e. Display the accuracy for each train/test set.

```python
score_all.pop(0)
print(score_all)
```

f. Display the mean accuracy and standard deviation of 10 models. Troubleshooting 5.

```python
print(np.mean(score_all))
print(np.std(score_all))
```

20. Generate a confusion matrix to visualize the performance of the classification model, in which the sum (or average) of the N patterns of results is displayed in a table layout (Figure 11).

a. Import all of the 10 arrays obtained from the above classification.

```python
model_array_01 = np.load('model_array_01.npy')
model_array_02 = np.load('model_array_02.npy')
model_array_03 = np.load('model_array_03.npy')
model_array_04 = np.load('model_array_04.npy')
model_array_05 = np.load('model_array_05.npy')
model_array_06 = np.load('model_array_06.npy')
model_array_07 = np.load('model_array_07.npy')
model_array_08 = np.load('model_array_08.npy')
```
b. Sum up the 10 arrays.

```python
model_array_09 = np.load('model_array_09.npy')
model_array_10 = np.load('model_array_10.npy')

model_array_sum = model_array_01 + model_array_02 + model_array_03 + model_array_04 +
                   model_array_05 + model_array_06 + model_array_07 + model_array_08 + model_array_09 +
                   model_array_10
```

c. Set up displaying of a confusion matrix. The parameters ‘normalize=True’ and ‘normalize=False’ specify that the values in the matrix are displayed as percentage or as they are, respectively.

```python
import matplotlib.pyplot as plt
%matplotlib inline
import itertools
import numpy as np

def plot_confusion_matrix(cm, classes, normalize=True, title='Confusion matrix',
                          cmap=plt.cm.Blues):
    ...
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting ‘normalize=False’.
    ...
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print("Confusion matrix, without normalization")
    print(cm)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
```
d. Display the confusion matrix.

```python
species=['Escherichia', 'Bacillus', 'Thermus', 'Thermococcus', 'Sulfolobus', 'Nitrososphaera']
plot_confusion_matrix(model_array_sum, species)
```

**Note:** The order of the labels to be included in the ‘species’ should be the same as that of the ‘labels’ above (see step 19b).

**Note:** Recently, a method for visualizing feature importance in convolutional neural network (CNN) has been reported (Lu et al., 2020), but it requires complicated calculations and own coding. In contrast, this protocol using RF can extract feature importance more easily with standard commands available in Python.

---

**Figure 10.** Calculation of mean feature importance and class probability for one train/test set

(A) The importance for each feature (i.e., Raman shift) obtained from one of 10 validation results.

(B) Workflow of calculation of the mean class probability for each true class (i.e., microbial species). Similar calculation is done for other 9 sets.
In constructing each decision tree, a subset of the spectral data sampled by bootstrapping (random resampling of data with replacement) is used, thereby leaving unsampled data (i.e., OOB samples). The performance of each tree can be validated by using these OOB samples. As can be seen from Figure 7C, the OOB error rate in general decreases with increasing the number of trees (‘n_estimators’). This curve is used to derive an optimal value of ‘n_estimators’ (Figure 9); it converges for the ‘n_estimators’ value larger than ~200 in the present case. Detailed instructions on how to display the OOB error rate can be found on the scikit-learn library web site: https://scikit-learn.org/stable/auto_examples/ensemble/plot_ensemble_oob.html?highlight=oob (our script file deposited on Zenodo also contains the code for OOB error rate).

21. Assign the wavenumbers with high feature importance scores obtained in step 19d by referring to one of the many previous papers that show tables of the Raman band assignments for major biological molecules; see, for example, (Carey, 1982; Kuhar et al., 2018; Matsuda et al., 2019; Naumann, 2001).

**EXPECTED OUTCOMES**

A highly accurate discrimination model for microbial cells can be constructed by using single-cell Raman spectra and the RF algorithm. It yields information on the features (Raman shift) that make significant contributions to discrimination between classes. Notable advantages of this protocol are that it can achieve sufficient discrimination accuracy with less computational load and that it is straightforward to extract the importance of features.

Depending on your research purposes, you can change the class label (e.g., from species to a higher or lower taxonomic level), as well as microbial species to be studied. For example, the present RF model was shown to work very well for bacteria/archaea binary classification, where the class of each sample was changed from species to domain. The preprocessed Raman spectral data (Figure 4F) can serve as input data for other types of analysis, such as multivariate analysis (e.g., principal component analysis and clustering analysis) and comparative analysis of peak intensities.
LIMITATIONS
With this protocol, we did not evaluate discrimination accuracies of microbial species whose single-cell Raman spectra are obtained with different Raman microscopes. Therefore, we suggest that all single-cell Raman data be acquired with the same instrument and settings.

Furthermore, we did not take into consideration the influence of different physiological states (e.g., exponential vs. stationary state) on discrimination results. Our preliminary studies showed that the intensities of some Raman bands (those of nucleic acids, in particular) vary significantly with cell’s physiological state and hence affect the discrimination of microbes by RF.

TROUBLESHOOTING
Problem 1
Fail to trap a microbial cell under the microscope.

Potential solution
Increase laser power for more effective trapping. Small particles suspended in the buffer (e.g., some organic/inorganic precipitate) may sometimes be misidentified as microbial cells. Such non-cellular objects tend to be difficult to trap.

Problem 2
Acquired single-cell Raman spectra suffer from very low S/N.

Potential solution
Capture a microbial cell that is not too far from the cover glass. For long rod-shaped microorganisms, trap a cell by focusing the laser at either end of the cell and not around the center. Avoid mechanical shocks to the microscope to prevent trapped cells from escaping. Trap only one cell because trapping multiple cells results in fluctuating cell positions and weak signal intensity. Note that if the size of a microbe is very small as in the case of *Nitrososphaera viennensis*, a coccus with a diameter of 0.6–0.9 μm, its Raman spectrum is very likely to have poor S/N because the number of Raman scatterers within the focal volume is small compared to normal-sized cells.

Problem 3
Fail to perform vector normalization.

Potential solution
It is often difficult to normalize data with very different noise levels. To cope with this problem, perform spectral smoothing prior to linear baseline fitting (Figure 6A). Subtract the baseline from both unsmoothed and smoothed spectra. Then use the norm of the baseline-subtracted, smoothed spectrum to vector-normalize the baseline-subtracted, unsmoothed spectrum.

Problem 4
Features having high importance scores are localized in regions where there are no Raman bands.

Potential solution
This could occur due to inappropriate baseline subtraction or normalization. For the former, adjust the region to be fitted (step 11c and Figure 6A). For the latter, refer to troubleshooting 3.

Overfitting may also cause this issue. RF models may classify microbial cells not by the differences in molecular vibrations characteristic of the cells (‘molecular fingerprints’), but by the features that happen to exist only in certain sample group. In such cases, validate the model using a dataset different from that used to create the model (e.g., data acquired on different measurement days) as a test dataset. If groups are very similar (e.g., between strains), you should be aware of the possibility of overfitting.
Problem 5
Classification accuracy is not high.

Potential solution
Increase the number of decision trees, namely, ‘n_estimators’. A caveat is that the use of too many trees can often lead to overfitting. If the number of Raman spectra differs substantially from class to class, specify the ‘class_weight’ parameter in the Random Forest Classifier.

RESOURCE AVAILABILITY
Lead contact
Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Shinsuke Shigeto (shigeto@kwansei.ac.jp).

Materials availability
This study did not generate new unique reagents.

Data and code availability
The datasets generated during this study are available at Mendeley Data: https://doi.org/10.17632/8cd34fckgt.1.

This paper includes part of the Python code generated during this study. The source code is available at Zenodo: https://doi.org/10.5281/zenodo.7122583.

Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

SUPPLEMENTAL INFORMATION
Supplemental information can be found online at https://doi.org/10.1016/j.xpro.2022.101812.

ACKNOWLEDGMENTS
This work was supported by the Japan Society for the Promotion of Science KAKENHI grant numbers JP19H05679 (Post-Koch Ecology), JP19H05681, and JP19H05689, and in part by the Japan Science and Technology Agency Exploratory Research for Advanced Technology (ERATO) grant number JPMJER1502.

AUTHOR CONTRIBUTIONS
N.K. and S.S. conceived and designed the research. S.S. supervised the project. N.K. performed the experiments and analyzed the data. M.M. and W.I. helped with data analysis. S.K. and M.O. provided the samples. N.K. and S.S. wrote the manuscript. All authors read and approved the final manuscript.

DECLARATION OF INTERESTS
The authors declare no competing interests.

REFERENCES
Carey, P. R. (1982). Biochemical Applications of Raman and Resonance Raman Spectroscopies (Academic Press).
Guo, S., Popp, J., and Bocklitz, T. (2021). Chemometric analysis in Raman spectroscopy from experimental design to machine learning-based modeling. Nat. Protoc. 16, 5426–5459.
Ho, C.-S., Jean, N., Hogan, C. A., Blackmon, L., Jeffrey, S. S., Holodniy, M., Banaei, N., Saleh, A.A.E., Ermon, S., and Dionne, J. (2019). Rapid identification of pathogenic bacteria using Raman spectroscopy and deep learning. Nat. Commun. 10, 4927.
Horie, H., Sasaki, M., Yoshikawa, Y., Toyofuku, M., and Shigeto, S. (2020). Raman spectroscopic signatures of carotenoids and polyenes enable label-free visualization of microbial distributions within pink biofilms. Sci. Rep. 10, 7704.
Huang, C.-K., Ando, M., Hamaguchi, H. O., and Shigeto, S. (2012). Disentangling dynamic changes of multiple cellular components during the yeast cell cycle by in vivo multivariate Raman imaging. Anal. Chem. 84, 5661–5668.
Huang, C. K., Hamaguchi, H. O., and Shigeto, S. (2011). In vivo multimode Raman imaging reveals concerted molecular composition and distribution changes during yeast cell cycle. Chem. Commun. 47, 9423–9425.
Kanno, N., Kato, S., Ohkuma, M., Matsui, M., Iwasaki, W., and Shigeto, S. (2021). Machine learning-assisted single-cell Raman fingerprinting for in situ and nondestructive classification of prokaryotes. iScience 24, 102975.

Kuhar, N., Sil, S., Verma, T., and Umapathy, S. (2018). Challenges in application of Raman spectroscopy to biology and materials. RSC Adv. 8, 25988–25998.

Lorenz, B., Wichmann, C., Stöckel, S., Rosch, P., and Popp, J. (2017). Cultivation-free Raman spectroscopic investigations of bacteria. Trends Microbiol. 25, 413–424.

Lu, W., Chen, X., Wang, L., Lu, H., and Fu, Y.V. (2003). Combination of an artificial intelligence approach and laser tweezers Raman spectroscopy for microbial identification. Anal. Chem. 92, 6288–6296.

Matsuda, A., Sakaguchi, N., and Shigeto, S. (2019). Can cells maintain their bioactivity in ionic liquids? A novel single-cell assessment by Raman microspectroscopy. J. Raman Spectrosc. 50, 768–777.

Naumann, D. (2001). FT-INFRARED and FT-Raman spectroscopy in biomedical research. Appl. Spectrosc. Rev. 36, 239–298.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Rinke, C., Schwientek, P., Szczurba, A., Ivanova, N.N., Anderson, I.J., Cheng, J.-F., Darling, A., Malfatti, S., Swan, B.K., Gies, E.A., et al. (2013). Insights into the phylogeny and coding potential of microbial dark matter. Nature 499, 431–437.

van Manen, H.-J., Kraan, Y.M., Roos, D., and Otto, C. (2004). Intracellular chemical imaging of home-containing enzymes involved in innate immunity using resonance Raman microscopy. J. Phys. Chem. B 108, 18762–18771.

Xie, C., Dinno, M.A., and Li, Y.-q. (2002). Near-infrared Raman spectroscopy of single optically trapped biological cells. Opt. Lett. 27, 249–251.

Yamakoshi, H., Dodo, K., Okada, M., Ando, J., Palonpon, A., Fujita, K., Kawata, S., and Sodeoka, M. (2011). Imaging of EdU, an alkyne-tagged cell proliferation probe, by Raman microscopy. J. Am. Chem. Soc. 133, 6102–6105.

Yasuda, M., Takeda, N., and Shigeto, S. (2019). Inhomogeneous molecular distributions and cytochrome types and redox states in fungal cells revealed by Raman hyperspectral imaging using multivariate curve resolution–alternating least squares. Anal. Chem. 91, 12501–12508.

Zhao, Z., Shen, Y., Hu, F., and Min, W. (2017). Applications of vibrational tags in biological imaging by Raman microscopy. Analyst 142, 4018–4029.