Data Mining and Knowledge Discovery (2019) 33:1821–1852
https://doi.org/10.1007/s10618-019-00647-x

catch22: CAnonical Time-series CHaracteristics

Selected through highly comparative time-series analysis

Carl H. Lubba[1] · Sarab S. Sethi[2] · Philip Knaute[2] · Simon R. Schultz[1] · Ben D. Fulcher[3] · Nick S. Jones[2]

Published online: 9 August 2019

Abstract
Capturing the dynamical properties of time series concisely as interpretable feature vectors can enable efficient clustering and classification for time-series applications across science and industry. Selecting an appropriate feature-based representation of time series for a given application can be achieved through systematic comparison across a comprehensive time-series feature library, such as those in the *hctsa* toolbox. However, this approach is computationally expensive and involves evaluating many similar features, limiting the widespread adoption of feature-based representations of time series for real-world applications. In this work, we introduce a method to infer small sets of time-series features that (i) exhibit strong classification performance across a given collection of time-series problems, and (ii) are minimally redundant. Applying our method to a set of 93 time-series classification datasets (containing over 147,000 time series) and using a filtered version of the *hctsa* feature library (4791 features), we introduce a set of 22 CAnonical Time-series CHaracteristics, *catch22*, tailored to the dynamics typically encountered in time-series data-mining tasks. This dimensionality reduction, from 4791 to 22, is associated with an approximately 1000-fold reduction in computation time and near linear scaling with time-series length, despite an average reduction in classification accuracy of just 7%. *catch22* captures a diverse and interpretable signature of time series in terms of their properties, including linear and non-linear autocorrelation, successive differences, value distributions and outliers, and fluctuation scaling properties. We provide an efficient implementation of *catch22*, accessible from many programming environments, that facilitates feature-

Responsible editor: Eamonn Keogh.

Ben D. Fulcher and Nick S. Jones have contributed equally to this study.

✉ Ben D. Fulcher
ben.fulcher@sydney.edu.au

✉ Nick S. Jones
nick.jones@imperial.ac.uk

Extended author information available on the last page of the article

based time-series analysis for scientific, industrial, financial and medical applications using a common language of interpretable time-series properties.

Keywords Time series · Classification · Clustering · Time-series features

1 Introduction

Time series, ordered sets of measurements over time, enable the study of the dynamics of real-world systems and have become a ubiquitous form of data. Quantitative analysis of time series is vital for countless domain applications, including in industry (e.g., to detect production irregularities), finance (e.g., to identify fraudulent transactions), and medicine (e.g., to diagnose pathological heartbeat patterns). As modern time-series datasets have grown dramatically in size, there is a pressing need for efficient methods to solve problems including time-series visualization, clustering, classification, and anomaly detection.

Many applications involving time series, including common approaches to clustering and classification, are based on a defined measure of similarity between pairs of time series. A straightforward similarity measure—for short, aligned time series of equal length—quantifies whether time-series values are close on average (across time) (Berndt and Clifford 1994; Vlachos et al. 2002; Moon et al. 2001; Faloutsos et al. 1994; Ye and Keogh 2009). This approach does not scale well, often quadratic in both number of time series and series length (Bagnall et al. 2017), due to the necessity to compute distances (often using expensive elastic metrics) between all pairs of objects. An alternative approach involves defining time-series similarity in terms of extracted features that are the output of time-series analysis algorithms (Fulcher and Jones 2014; Fulcher 2018). This approach yields an interpretable summary of the dynamical characteristics of each time series that can then be used as the basis of efficient classification and clustering in a feature space using conventional machine-learning methods.

The number of time-series analysis methods that have been devised to convert a complex time-series data stream into an interpretable set of real numbers is vast, with contributions from a diverse range of disciplinary applications. Some examples include standard deviation, the position of peaks in the Fourier power spectrum, temporal entropy, and many thousands of others (Fulcher 2018; Fulcher et al. 2013). From among this wide range of possible features, selecting a set of features that successfully captures the dynamics relevant to the problem at hand has typically been done manually without quantitative comparison across a variety of potential candidates (Timmer et al. 1993; Nanopoulos et al. 2001; Mörchen 2003; Wang et al. 2006; Bagnall et al. 2012). However, subjective feature selection leaves uncertain whether a different feature set may have optimal performance on a task at hand. Addressing this shortcoming, recent methods have been introduced that take a systematic, data-driven approach involving large-scale comparisons across thousands of time-series features (Fulcher et al. 2013; Fulcher and Jones 2017). This 'highly-comparative' approach involves comparison across a comprehensive collection of thousands of diverse time-series features and has recently been operationalized as the *hctsa* (highly comparative time-series analysis) toolbox (Fulcher et al. 2013; Fulcher and Jones 2017, 2014). *hctsa* has been used

for data-driven selection of features that capture the informative properties in a given dataset in applications ranging from classifying Parkinsonian speech signals (Fulcher et al. 2013) to identifying correlates of mouse-brain dynamics in the structural connectome (Sethi et al. 2017). These applications have demonstrated how automatically constructed feature-based representations of time series can, despite vast dimensionality reduction, yield competitive classifiers that can be applied to new data efficiently (Fulcher and Jones 2014). Perhaps most importantly, the selected features provide interpretable understanding of the differences between classes, and therefore a path towards a deeper understanding of the underlying dynamical mechanisms at play.

Selecting a subset of features from thousands of candidate features is computationally expensive, making the highly-comparative approach unfeasible for some real-world applications, especially those involving large training datasets (Bandara et al. 2017; Shekar et al. 2018; Biason et al. 2017). Furthermore, the *hctsa* feature library requires a Matlab license to run, limiting its widespread adoption. Many more real-world applications of time-series analysis could be tackled using a feature-based approach if a reduced, efficient subset of features, that capture the diversity of analysis approaches contained in *hctsa*, was available.

In this study we develop a data-driven pipeline to distill reduced subsets of the most useful and complementary features for classification from thousands of initial candidates, such as those in the *hctsa* toolbox. Our approach involves scoring the performance of each feature independently according to its classification accuracy across a calibration set of 93 time-series classification problems (Bagnall et al.). We show that the performance of an initial (filtered) pool of 4791 features from *hctsa* (mean class-balanced accuracy across all tasks: 77.2%) can be well summarized by a smaller set of just 22 features (71.7%). We denote this high-performing subset of time-series features as *catch22* (22 CAnonical Time-series CHaracteristics). The *catch22* feature set: (1) computes quickly ($\sim$ 0.5 s/ 10,000 samples, roughly a thousand times faster than the full *hctsa* feature set in Matlab) and empirically scales almost linearly, $\mathcal{O}(N^{1.16})$; (2) provides a low-dimensional summary of time series into a concise set of interpretable characteristics that are useful for classification of diverse real-world time-series; and (3) is implemented in C with wrappers for python, R, and Matlab, facilitating fast time-series clustering and classification. We envisage *catch22* expanding the set of problems—including scientific, industrial, financial, and medical applications—that can be tackled using a common feature-based language of canonical time-series properties.

2 Methods

We here describe the datasets we evaluate features on, the time-series features contained in the *hctsa* toolbox (Fulcher et al. 2013; Fulcher and Jones 2017), and the selection pipeline to generate a reduced feature subset.

2.1 Data

To select a reduced set of useful features, we need to define a measure of usefulness. Here we use a diverse calibration set of time-series classification tasks from the UEA/UCR (University of East Anglia and University of California, Riverside) Time-Series Classification Repository (Bagnall et al. 2017). The number of time series per dataset ranges from 28 ('ECGMeditation') to 33,274 ('ElectricalDevices') adding up to a total of 147,198 time series. Individual time series range in length from 24 samples ('ItalyPowerDemand') to 3750 samples ('HeartbeatBIDMC'), and datasets contain between 2 classes (e.g., 'BeetleFly') and 60 classes ('ShapesAll'). For 85 of the 93 datasets, unbalanced classification accuracies were provided for different shape-based classifiers such as dynamic time warping (DTW) (Berndt and Clifford 1994) nearest neighbor, as well as for hybrid approaches such as COTE (Bagnall et al. 2016). All unbalanced accuracies, a^{ub}, were computed using the fixed training-test split provided by the UCR repository, as the proportion of class predictions that matched the actual class labels:

$$a^{\mathrm{ub}}(y, \hat{y}) = \frac{1}{n_{\mathrm{TS}}} \sum_{l=1}^{n_{\mathrm{TS}}} \mathbb{1}(\hat{y}_l = y_l), \tag{1}$$

where y_l is the actual class, $\hat{y}_l$ is the predicted class, n_{TS} is the number of time series in the test dataset, and $\mathbb{1}$ is the indicator function.

2.2 Time-series features

Our aim is to obtain a data-driven subset of the most useful time-series features by comparing across as diverse a set of time-series analysis algorithms as possible. An ideal starting point for such an exercise is the comprehensive library of over 7500 features provided in the *hctsa* toolbox (Fulcher et al. 2013; Fulcher and Jones 2017). Features included in *hctsa* are derived from a wide conceptual range of algorithms, including measurements of the basic statistics of time-series values (e.g., location, spread, Gaussianity, outlier properties), linear correlations (e.g., autocorrelation, power spectral features), stationarity (e.g., StatAv, sliding window measures, prediction errors), entropy (e.g., auto-mutual information, Approximate Entropy, Lempel-Ziv complexity), methods from the physical nonlinear time-series analysis literature (e.g., correlation dimension, Lyapunov exponent estimates, surrogate data analysis), linear and nonlinear model parameters, fits, and predictive power [e.g., from autoregressive moving average (ARMA), Gaussian Process, and generalized autoregressive conditional heteroskedasticity (GARCH) models], and others (e.g., wavelet methods, properties of networks derived from time series, etc.) (Fulcher et al. 2013; Fulcher and Jones 2017). Features were calculated in Matlab 2017a (a product of The MathWorks, Natick, MA, USA) using *hctsa* v0.97. For each dataset, each feature was linearly rescaled to the unit interval.

We performed an initial filtering of all 7658 *hctsa* features based on their characteristics and general applicability. Because the vast majority of time series in the UEA/UCR repository are z-score normalized,[1] we first removed the 766 features that are sensitive to the mean and variance of the distribution of values in a time series based on keywords assigned through previous work (Fulcher et al. 2013), resulting in a set of 6892 features. We note that on specific tasks with non-normalized data, features of the raw value distribution (such as mean, standard deviation, and higher moments) can lead to significant performance gains and that for some applications, this initial preselection is undesirable (Dau et al. 2018). Given a suitable collection of datasets in which the raw value distributions contain information about class differences, our pipeline can easily skip this preselection. We next excluded the features that frequently outputted special values, which indicate that an algorithm is not suitable for the input data, or that it did not evaluate successfully. Algorithms that produced special-valued outputs on at least one time series in more than 80% of our datasets were excluded: a total of 2101 features (across datasets, minimum: 655, maximum: 3427, mean: 1327), leaving a remaining set of 4791 features. This relatively high number of features lost during preselection reflects our strict exclusion criterion for requiring real-valued outputs across a diverse range of input data, and the restricted applicability of many algorithms (e.g., that require a minimum length of input data, require positive-valued data, or cannot deal with data repeated identical values). For example, the datasets with the most special-valued features are 'ElectricDevices' (3427 special-valued features), which contains 96-sample time series with many repeated values (e.g., some time series contain just 10 unique values), and 'ItalyPowerDemand' (2678 special-valued features), which consists of very short (24-sample) time series. The 4791 features that survived the preselection gave real-valued outputs on at least 90% of the time series of all datasets, and 90% of them succeeded on at least 99% of time series.

2.3 Performance-based selection

In contrast to typical feature-selection algorithms, which search for combinations of features that perform well together (and might therefore include features that have low individual performance), our procedure involves a pre-filtration to identify features that individually possess discriminatory power across a diverse range of real-world data, before subsequently identifying those that exhibit complementary behavior. To this end we used the pipeline depicted in Fig. 1, which evaluates the univariate classification performance of each feature on each task, combines feature-scores across tasks, and then selects a reduced set of the most useful features across a two-step filtering process which involves: (1) *performance filtering*: select features that perform best across all tasks, and (2) *redundancy minimization*: reduce redundancy between features. The method is general and is easily extendable to different sets of classification tasks, or to different initial pools of features. All analysis was performed in Python 2.7 using `scikit-learn` and code to repro-

[1] With the notable exception of four unnormalized datasets: 'AALTDChallenge', 'ElectricDeviceOn', 'ECGMeditation', 'HeartbeatBIDMC'.

duce all of our analyses is accessible on GitHub (https://github.com/chlubba/op_importance).

2.4 Quantifying feature performance

Our pipeline (Fig. 1) requires a method to score the performance of individual features across classification tasks. We scored each feature by its ability to distinguish the labeled classes in each of our $M = 93$ classification tasks and then computed a combined performance score for that feature across all tasks. Classification was performed using a decision tree with stratified cross validation with N_{CV} folds. The number of folds, N_{CV}, was chosen separately for each task according to:

$$N_{\mathrm{CV}} = \min\left\{10, \max\left[2, \min_{k=1}^{N_c}\left(\sum_{l=1}^{N_{\mathrm{TS}}} \mathbb{1}(y_l = k)\right)\right]\right\}, \tag{2}$$

where N_{TS} is the number of time series, N_c is the number of classes, and y_l is the class-label of the lth time series.

For feature i ($i = 1, \ldots, 4791$) on classification task j ($j = 1, \ldots, M$), we computed the mean class-balanced classification accuracy across folds $a_{i,j}$ as a performance score.

$$a_{i,j}(y, \hat{y}, w) = \frac{1}{\sum_{l=1}^{N_{\mathrm{TS}}} w_l} \sum_{l=1}^{N_{\mathrm{TS}}} \mathbb{1}(\hat{y}_l = y_l) w_l, \tag{3}$$

where the weights for each time series w_l compensate for imbalances in the number of samples per class, $w_l = 1/\sum_{m=1}^{N_{\mathrm{TS}}} \mathbb{1}(y_m = y_l)$. To combine scores across tasks, we computed a normalized accuracy of the jth task by dividing raw feature accuracies, $a_{i,j}$, by the mean accuracy across all features on that task, $\bar{a}_j$, as follows:

$$a^{\mathrm{n}}_{i,j} = \frac{a_{i,j}}{\bar{a}_j}. \tag{4}$$

This allowed us to quantify the performance of each feature on a given task relative to the performances of other features in our library.

Finally, the combined feature-accuracy-score across all tasks, $a_i^{\mathrm{n,c}}$, was calculated as the mean over normalized accuracies, $a^{\mathrm{n}}_{i,j}$, on our $M = 93$ tasks:

$$a_i^{\mathrm{n,c}} = \frac{1}{M} \sum_{j=1}^{M} a^{\mathrm{n}}_{i,j}. \tag{5}$$

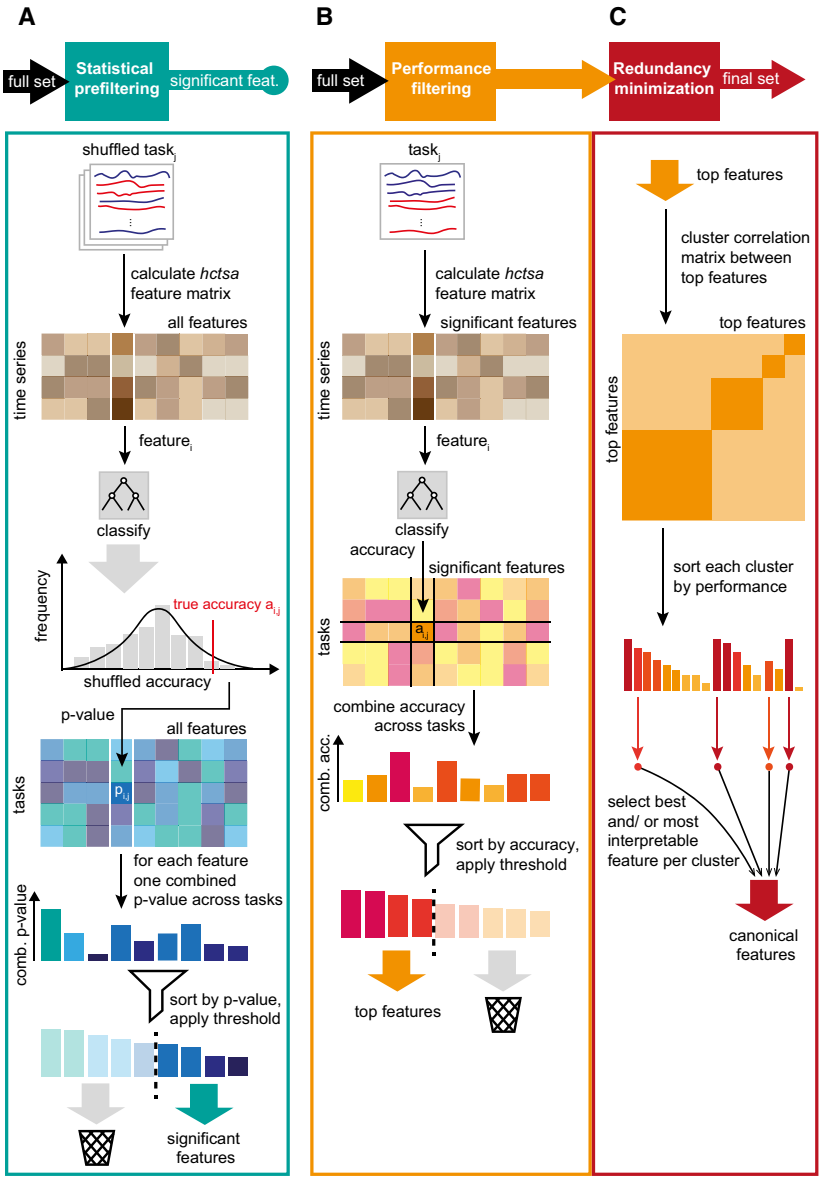

Fig. 1 Given a set of classification tasks, our pipeline selects a reduced set of high-performing features while minimizing inter-feature redundancy. **a** *Statistical prefiltering* We identified features with performance consistent with that of random number generators. To this end, we derived null accuracy distributions for each feature on each task by classifying repeatedly on shuffled class labels. *P*-values from those null distributions were combined across datasets to identify features with performance consistent with random-number generators. **b** *Performance filtering* We selected an intermediate set of top features by ranking and thresholding the combined accuracy over all datasets. **c** *Redundancy minimization* Top performing features were clustered into groups with similar performance across tasks to minimize redundancy between the final set of canonical features. We selected a single representative feature per cluster to yield a canonical feature set

2.5 Statistical prefiltering

Given the size and diversity of features in *hctsa*, we first wanted to determine whether some features exhibited performance consistent with chance on this set of classification tasks. To estimate a p-value for each feature on each classification task, we generated null accuracy distributions, $a^s_{i,j}$ (superscript s indicating 'shuffeled'), using a permutation-based procedure that involved repeated classification on randomly shuffled class labels, shown in Fig. 1a. The computational expense of estimating $\sim$440,000 p-values using permutation testing, one for each of the 4791 features on each of the 93 problems, scales linearly with the number of repetitions. To limit computation time to within reasonable bounds, we fitted a Gaussian approximation, estimated from 1000 null samples for each each feature-task combination, $a^s_{i,j}$, and used it to estimate p-values to a resolution beyond the limits of traditional permutation testing with this many null samples (i.e., 0.001). From visual inspection, the distributions were mostly unimodal and approximately normally distributed and, as expected, had higher variance on datasets with fewer time series.

The p-values for a given feature across all classification tasks were combined using Fisher's method (Fisher 1925) and corrected for multiple hypothesis testing across features using the Holm-Bonferroni method (Holm 1979) at a significance level of 0.05.

2.6 Selecting a canonical set of features

From the features that performed significantly better than chance, we selected a subset of β high-performing features by ranking them by their combined normalized accuracy $a^{\mathrm{n,c}}$ (Fig. 1b), comparing values in the range $100 \leq \beta \leq 1000$. As shown in Fig. 1c, we then aimed to reduce the redundancy in these top-performing features, defining redundancy in terms of patterns of performance across classification tasks. To achieve this, we used hierarchical clustering on the Pearson correlation distance, $d_{ij} = 1 - r_{ij}$ between the M-dimensional performance vectors of normalized accuracies a^{n} of features i and j. Clustering was performed using complete linkage at a threshold $\gamma = 0.2$ to form clusters of similarly performing features, that are all inter-correlated by $r_{ij} > 1-\gamma$ (for all i, j in the cluster). We then selected a single feature to represent each cluster, comparing two different methods: (i) the feature with the highest normalized accuracy combined across tasks, and (ii) manual selection of representative features to favour interpretability (while also taking into account computational efficiency and classification performance).

2.7 Overall classification performance

To evaluate the classification performance of different feature sets, and compare our feature-based classification to alternative time-series classification methods, we used two different accuracy measures. Comparisons between different sets of *hctsa*-features were based on the mean class-balanced accuracy across M tasks and N_{CV} cross-validation folds:

$$a_{\text{tot}} = \frac{1}{M}\sum_{j=1}^{M}\frac{1}{N_{\text{CV},j}}\sum_{k=1}^{N_{\text{CV},j}} a_{j,k}. \tag{6}$$

When comparing our feature sets to existing methods we used the mean unbalanced classification accuracy across tasks as in Eq. (1) on the given train-test split to match the metric used for the accuracies supplied with the UEA/UCR repository:

$$a_{\text{tot}}^{\text{ub}} = \frac{1}{N_{\text{tasks}}}\sum_{j=1}^{N_{\text{tasks}}} a_j^{\text{ub}}. \tag{7}$$

2.8 Execution times and scaling

One of the merits of a small canonical feature set for time-series characterization is that it is quick to compute. To compare the execution time of different feature sets, we used a benchmark set of 40 time series from different sources, including simulated dynamical systems, financial data, medical recordings, meteorology, astrophysics, and bird sounds (see Sect. 5.3 for a complete list). To estimate how features scale with time-series length, we generated multiple versions for each of our 40 reference time series of different lengths from 50 to 10,000 samples. Lengths were adapted by either removing points after a certain sample count or by up-sampling of the whole time series to the desired length. Execution times were obtained on a 2.2 GHz Intel Core i7, using single-threaded execution (although we note that feature calculation can be parallelized straightforwardly).

2.9 Selecting the two most informative features from a small subset

For the purpose of quickly analyzing a dataset visually in feature space, it can be helpful to identify the two features that, taken together, are the most informative to distinguish between time-series classes of the selected dataset. To this end, we used sequential forward selection (Whitney 1971; Fulcher and Jones 2014) that first selects a single feature which achieves the best mean class-balanced accuracy across cross-validation folds and then iterates over the remaining features to select the one that, combined with the first feature, reaches the best accuracy.

3 Results

We present results of using our pipeline to obtain a canonical set of 22 time-series features from an initial pool of 4791 candidates. We name our set *catch22* (22 CAnonical Time-series CHaracteristics), which approximates the classification performance of the initial feature pool to 90% and computes in less than 0.5 s on 10,000 samples.

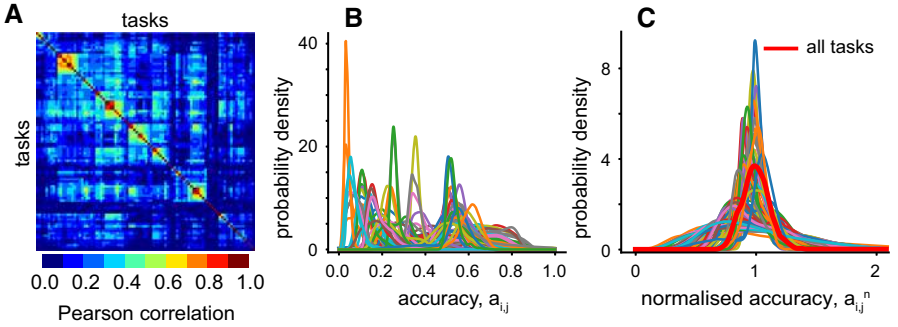

Fig. 2 Normalization of feature accuracies allows comparison of performance scores across a diverse set of 93 classification tasks. **a** A 93×93 matrix of Pearson correlation coefficients between the 4791-dimensional accuracy vectors of pairs of tasks, reordered according to hierarchical linkage clustering. **b** Each line shows the smoothed distribution over feature-accuracies on one classification task. Differences in task difficulty are reflected by a wide range of accuracies. **c** The accuracies plotted in **b** were normalized by the mean accuracy of each task, as in Eq. (4). The red line shows the distribution of normalized and combined accuracies across all tasks, Eq. (5) (Color figure online)

3.1 Performance diversity across classification tasks

We first analyze the 93 classification tasks, which are highly diverse in their properties (see Sect. 2.1) and difficulty, as shown in Fig. 2. We characterized the similarity of two tasks in terms of the types of features that perform well on them, as the Pearson correlation coefficient between accuracies of all features, shown in Fig 2a. The figure reveals a diversity of performance signatures across tasks: for some groups of tasks, similar types of features contribute to successful classification, whereas very different types of features are required for other tasks. The 93 tasks also vary markedly in their difficulty, as judged by the distribution of accuracies, $a_{i,j}$, across tasks, shown in Fig. 2b. We normalized feature accuracies across tasks by dividing them by the mean accuracy of the task at hand, Eq. (4), yielding normalized accuracies, $a^{\mathrm{n}}_{i,j}$, that were comparable across tasks, shown in Fig. 2c. Note that this normalized accuracy scores features relative to all other features on a given task. The red line in Fig. 2c shows the distribution of the mean normalized accuracies across tasks $a_i^{n,c}$, Eq. (5).

3.2 Features with performance consistent with chance

To detect whether some features in *hctsa* exhibit a combined performance across classification tasks that is consistent with the performance of a random-number generator, we used a permutation-testing based procedure (described in Sect. 2.5). At a significance level $p < 0.05$, 145 of the 4791 features (or 3%) exhibited chance-level performance. These 145 features (listed in "Appendix" Table 3) were mostly related to quantifying complex dynamics in longer time series, such as nonlinear time-series analysis methods, long-range automutual information; properties that are not meaningful for the short, shape-based time-series patterns that dominate the UEA/UCR database.

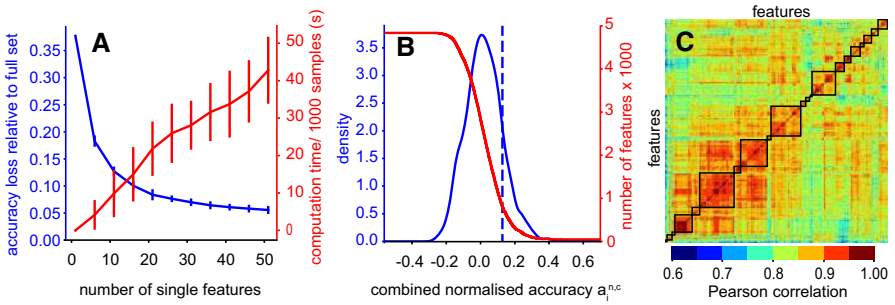

Fig. 3 The mean classification performance of the full feature set can be well approximated (to within 10%) by as few as 20 features. **a** While computation time is observed to rise linearly with increasing number of single selected features, the relative difference in accuracy to the full set of 4791 features starts to saturate at around 20 features. The performance of our final set of features is not highly sensitive to the size of our intermediate set of top-performing features, β. Error bars signify standard deviation over both relative loss in accuracy and computation time for different numbers of top features ($\beta = 100, 200, \ldots, 1000$), which were clustered to obtain single features (see Methods Sect. 2.6). **b** We select the number of top features from a relative threshold on the combined normalized accuracy across tasks shown as a dashed blue vertical line, yielding a set of 710 high-performing features. **c** High-performing features were clustered on performance-correlation distances using hierarchical complete linkage clustering, using a distance threshold γ of of 0.2, yielding 22 clusters (Color figure online)

3.3 Top-performing features

As a second step in our pipeline, we ranked features by their combined accuracy across tasks and then selected a subset of β best performers. How important is the choice of β? Fig. 3a shows how the relative difference in classification accuracy between full and reduced set $(a_{\text{tot,full}} - a_{\text{tot,subset}})/a_{\text{tot,full}}$ (blue line) and computation time (red line) evolve when increasing the number of clusters (1–50) into which the top performing features are grouped. The relative classification accuracy difference saturated at under 10% for between 20 and 30 selected features showing that this modest number of estimators covers most of diversity of the full set. Error bars signify the standard deviation over accuracies and computation times when starting from different numbers of top performers $\beta = 100, 200, 300, \ldots, 1000$. Their tightness demonstrates that the accuracy of the final feature subset was not highly sensitive to the value of β. Computation time is more variable. To obtain a reduced set of high-performing features, we used a threshold on the combined normalized accuracy $a^{n,c}$ of one standard deviation above the mean, $a_{\text{th}} = \overline{a^{\text{n,c}}} + \sigma_{a^{\text{n,c}}}$, shown in Fig. 3b, yielding a set of 710 features.

3.4 A canonical feature set, *catch22*

We reduced inter-feature redundancy in *hctsa* (Fulcher et al. 2013), by applying hierarchical complete linkage clustering based on the correlation distances between performance vectors of the set of 710 high-performing features, as shown in Fig. 3c. Clustering at a distance threshold $\gamma = 0.2$ (see Sect. 2.6) yielded 22 clusters of similarly-performing features, where the correlation of performance vectors between

all pairs of features within each cluster was greater than 0.8. Different values of γ correspond to different penalties for redundancy; e.g., higher values ($\gamma > 0.4$) group all features into a single cluster, whereas low values would form many more clusters and increase the size and complexity of computing the resulting canonical feature set. We found $\gamma = 0.2$ to represent a good compromise that yields a resulting set of 22 clusters that matches the saturation of performance observed between 20 and 30 features (Fig. 3a).

We next aimed to capture the behavior of each of the 22 clusters as a single feature with the most representative behavior of its cluster. We first achieved this automatically: selecting the feature with the highest combined normalized accuracy $a_{\mathrm{i}^{n,c}}$ from each cluster. When classifying our tasks with this set of 22 best estimators, it reached an overall class-balanced accuracy over folds and tasks a^{tot}, Eq. (7), of $\sim$70%, compared to $\sim$77% using the full set. However, it is desirable for our 22 features to be as fast and easily interpretable as possible. For 6 of the 22 clusters, the top-performing feature was relatively complicated to compute and only offered a small improvement in performance relative to simpler features with similar performance in the same cluster. In these cases, we manually selected a simpler and more interpretable feature, yielding a final canonical set of 22 features which we call *catch22* (CAnonical Time-series CHaracteristics). The 22 features that make up *catch22* are described in Table 1. The *catch22* features reflect the diverse and interdisciplinary literature of time-series analysis methods that have been developed to date (Fulcher et al. 2013), simultaneously probing different types of structure in the data, including properties of the distribution of values in the time series, its linear and non-linear autocorrelation, predictability, scaling of fluctuations, and others.

Using the diverse canonical *catch22* feature set, the mean class-balanced accuracy across all datasets, a^{tot}, of *catch22* was $\sim$72%, a small reduction relative to the $\sim$77% achieved when computing all 4791 features and very similar to the $\sim$70% of the 22 highest-ranked features in each cluster. See Fig. 4a for a dataset-by-dataset scatter. The change in mean accuracy across folds and tasks, a^{tot}, from using the 22 features of *catch22* instead of all 4791 features cote the properties of a given dataset, but there was an average reduction in class-balanced accuracy (mean across folds) of 7.5% relative to the full set accuracy (77.2% full vs. 71.7% canonical, 7.5 percentage points). For some difficult problems, the increased computational expense of the full set of 4791 features yields a large boost in classification accuracy (accuracy of *catch22* lower by a relative difference of 37% for the dataset 'EthanolLevel'; 50.2% full vs. 31.8% *catch22*). The reduced set gave better mean performance in only a small number of cases: e.g., for 'ECGMeditation' with 60% full versus 81.2% *catch22*; given that this dataset contained just 28 time series and had a high standard deviation in accuracies of the full set between folds (35.3%), the performance might not be significantly increased.

How does the performance of the data-driven features, *catch22*, compare to other small feature sets proposed in the literature? One popular collection of features is the manually-curated `tsfeatures` package (Hyndman et al. 2019) of which certain features were used for forecasting (Bandara et al. 2017), anomaly detection (Hyndman et al. 2016), and clustering (Williams 2014). While not being explicitly optimized for classification and clustering, its widespread adoption demonstrates its versatility

Table 1 The *catch22* feature set spans a diverse range of time-series characteristics representative of the diversity of interdisciplinary methods for time-series analysis

hctsa feature name	Description
Distribution	
DN_HistogramMode_5	Mode of z-scored distribution (5-bin histogram)
DN_HistogramMode_10	Mode of z-scored distribution (10-bin histogram)
Simple temporal statistics	
SB_BinaryStats_mean_longstretch1	Longest period of consecutive values above the mean
DN_OutlierInclude_p_001_mdrmd	Time intervals between successive extreme events above the mean
DN_OutlierInclude_n_001_mdrmd	Time intervals between successive extreme events below the mean
Linear autocorrelation	
CO_f1ecac	First $1/e$ crossing of autocorrelation function
CO_FirstMin_ac	First minimum of autocorrelation function
SP_Summaries_welch_rect_area_5_1	Total power in lowest fifth of frequencies in the Fourier power spectrum
SP_Summaries_welch_rect_centroid	Centroid of the Fourier power spectrum
FC_LocalSimple_mean3_stderr	Mean error from a rolling 3-sample mean forecasting
Nonlinear autocorrelation	
CO_trev_1_num	Time-reversibility statistic, $\langle (x_{t+1} - x_t)^3 \rangle_t$
CO_HistogramAMI_even_2_5	Automutual information, $m = 2, \tau = 5$
IN_AutoMutualInfoStats_40_gaussian_fmmi	First minimum of the automutual information function
Successive differences	
MD_hrv_classic_pnn40	Proportion of successive differences exceeding 0.04σ (Mietus 2002)
SB_BinaryStats_diff_longstretch0	Longest period of successive incremental decreases
SB_MotifThree_quantile_hh	Shannon entropy of two successive letters in equiprobable 3-letter symbolization
FC_LocalSimple_mean1_tauresrat	Change in correlation length after iterative differencing
CO_Embed2_Dist_tau_d_expfit_meandiff	Exponential fit to successive distances in 2-d embedding space

Table 1 continued

hctsa feature name	Description
Fluctuation Analysis	
`SC_FluctAnal_2_dfa_50_1_2_logi_prop_r1`	Proportion of slower timescale fluctuations that scale with DFA (50% sampling)
`SC_FluctAnal_2_rsrangefit_50_1_logi_prop_r1`	Proportion of slower timescale fluctuations that scale with linearly rescaled range fits
Others	
`SB_TransitionMatrix_3ac_sumdiagcov`	Trace of covariance of transition matrix between symbols in 3-letter alphabet
`PD_PeriodicityWang_th0_01`	Periodicity measure of (Wang et al. 2007)

Features in *catch22* capture time-series properties of the distribution of values in the time series, linear and nonlinear temporal autocorrelation properties, scaling of fluctuations, and others

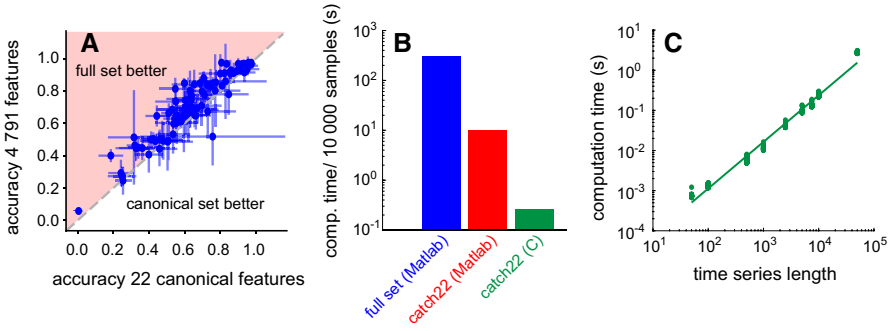

Fig. 4 The *catch22* set of 22 features approximates the classification performance of all 4791 features despite a dramatic reduction in computation time. **a** Each point represents one dataset in its balanced accuracy based on the *catch22* feature set (x-axis) and the full set of 4791 features (y-axis). Error bars signify standard deviation across cross-validation folds. *catch22* performs only a relative 7.5% worse than the full set of 4791 features: 71.7% versus 77.2% mean class-balanced accuracy across tasks a^{tot} as defined in Eq. (6). **b** Bars represents average over serial computation times for each of our 40 reference time series at a length of 10,000 samples using the full set of 4791 features, *catch22* in Matlab and *catch22* in **c**. From full set in Matlab to *catch22* in **c**, computation time decreases from ∼300 s to less than 0.5 s. **c** Each dot shows computation time for one of the 40 reference time series adjusted to different lengths for the C-implemented *catch22* set. The linear fit in the logarithmic plot reveals an almost linear scaling, with a scaling exponent of 1.16. See Sect. 2.8 for a description of the data

in characterizing time series and makes it an interesting candidate to compare with *catch22*. We classified all datasets based on the 16 default features of `tsfeatures` (version 1.0.0) listed in "Appendix" Sect. 5.4. Reassuringly, the class-balanced accuracies of both feature sets were very similar across the generic UEA/UCR datasets, with a Pearson correlation coefficient $r = 0.93$ (Fig. 5). The mean accuracy across tasks and folds, a^{tot}, was slightly higher for *catch22* (71.7%) than `tsfeatures` (69.4%). Our pipeline is general, and can select informative subsets of features for

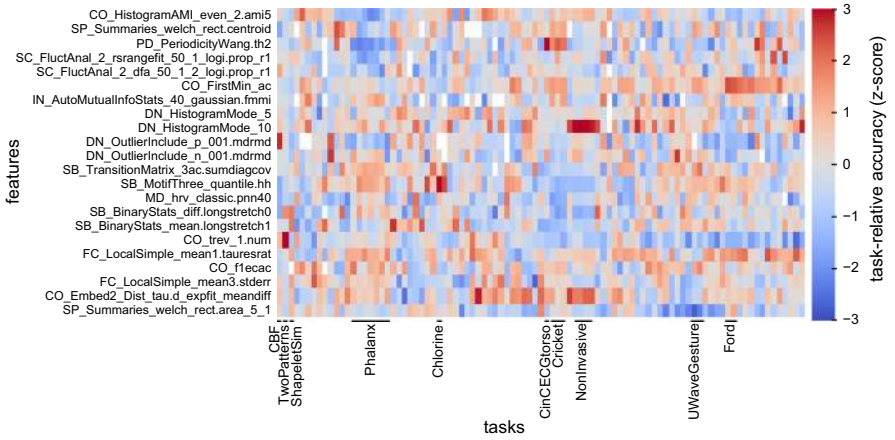

Fig. 5 Our automatically selected *catch22* feature set performs as well as the standard feature set for simple time series contained in the *tsfeatures* package. Class-balanced accuracy is shown for *tsfeatures* and *catch22*, error bars indicate standard deviation across folds. A gray dashed equality line is annotated, and particular datasets with the greatest differences in accuracy are highlighted as red circles and labeled

any collection of problems; e.g., for a more complex set of time-series classification tasks, our pipeline may yield estimators of more distinctive and complex dynamics.

How diverse are the features in *catch22*? Fig. 6 displays the class-balanced accuracies of each of the *catch22* features (rows) on each task (columns), z-normalized by task. Some groups of tasks recruit the same types of features for classification (reflected by groups of columns with similar patterns). Patterns across rows capture the characteristic performance signature of each feature, and are visually very different, reflecting the diversity of features that make up *catch22*. This diversity is key to being able to probe the different types of temporal structure required to capture specific differences between labeled classes in different time-series classification tasks in the UEA/UCR repository. Feature-performances often fit the known dynamics in the data, e.g., for the two datasets 'FordA' and 'FordB' in which manual inspection reveals class differences in the low frequency content, the most successful feature is '`CO_FirstMin_ac`' which finds the first minimum in the autocorrelation function. In some datasets, high performance can be attained using just a single feature, e.g., in 'ChlorineConcentration' ('`SB_motifThree_quantile.hh`', 52.3% versus 67.5% class-balanced mean accuracy over folds *a* for *catch22* vs. all features) and 'TwoPatterns' ('`CO_trev_1.num`', 73.4% vs. 88.1%).

3.5 Computation time and complexity

The classification performance using all 4791 features is well approximated by the 22 features in *catch22*, but how much computational effort does it save? To minimize execution time and make our condensed subset accessible from all major ecosystems used by the data-mining community, we implemented all *catch22* features in C and wrapped them for R, Python and Matlab. All code is accessible on GitHub (https://

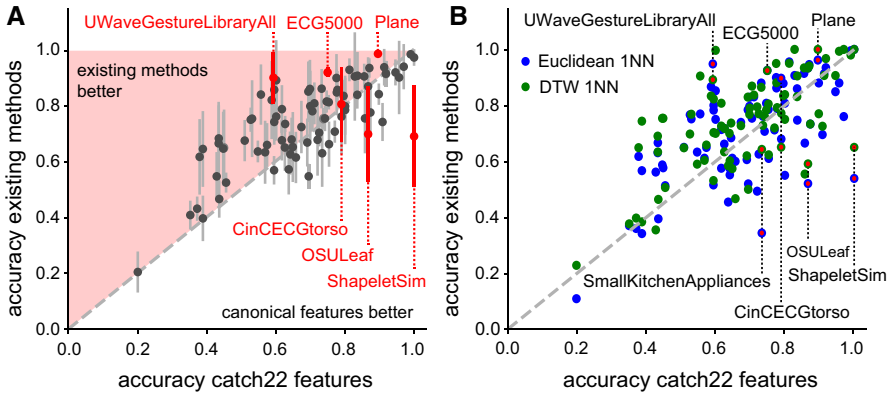

Fig. 6 The canonical features in *catch22* are sufficiently diverse to enable high performance across diverse classification tasks. The matrix shows class-balanced accuracies, z-scored per task (column), truncated at ± 3, and was reordered by hierarchical linkage clustering based on correlation distance in both columns (93 classification tasks) and rows (22 features). Similar columns are visible for datasets of the same type. The *catch22* features each show strengths and weaknesses, and their diversity allows them to complement each other across a range of tasks (Color figure online)

github.com/chlubba/catch22). Using this C-implementation, the *catch22* feature set can be computed sequentially on all 93 datasets of the UEA/UCR repository in less than 15 min on an Intel Core i7. On average, the features for each dataset were calculated within 9.4 s, the slowest being 'StarLightCurves' with 97 s due to its many (9236) relatively long (1024 samples) time series. The 27 quickest datasets stayed below 1 s in computation time; the three quickest, 'BirdChicken', 'Coffee', and 'BeetleFly' took less than 0.25 s.

While time series contained in the UEA/UCR repository are usually short, with an average length of 500 samples, real-world recordings can be substantially longer. Therefore, to understand how the computation times of our feature set scale with time-series lengths above those available in the UEA/UCR repository, we used a set of 40 reference time series from diverse sources (described in Sect. 2.8) to evaluate execution times of all *hctsa*- and the *catch22*-features for longer time series. Figure 4b shows execution times of different feature sets as a mean over our 40 reference time series at length 10,000. The Matlab implementation of *catch22* accelerates computation time by a factor of ~ 30 compared to the full set of 4791 from ~ 300 to ~ 10 s. The C-implementation of *catch22* again reduces execution time by a factor of approximately 30 compared to the Matlab implementation to ~ 0.3 s at 10,000 samples, signifying an approximately 1000-fold acceleration compared to the full *hctsa* feature set in Matlab. The C-version of *catch22* exhibits near-linear computational complexity, $\mathcal{O}(N^{1.16})$, as shown in Fig. 4c. Features varied markedly in their execution time, ranging from (C-implemented) `DN_HistogramMode_10` (< 0.1 ms for our 10,000-sample reference series) to `PD_PeriodicityWang_th0_01` (79 ms), with the latter representing approximately one third of the total computation time for *catch22*. A further acceleration by a factor of 3 could be achieved through paralleliza-

tion, limited by the slowest feature `PD_PeriodicityWang_th0_01` which takes up one third of the overall computation time.

3.6 Performance comparison

Compared to conventional shape-based time-series classifiers, that use distinguishing patterns in the time domain as the basis for classification (Fulcher and Jones 2014; Fulcher 2018), feature-based representations can reduce high-dimensional time series down to a compact and interpretable set of important numbers, constituting a dramatic reduction in dimensionality. While this computationally efficient representation of potentially long and complex streams of data is appealing, important information may be lost in the process, resulting in poorer performance than alternative methods that learn classification rules on the full time-series object. To investigate this, we compared the classification performance of *catch22* (using a decision tree classifier as for every classification, see Sect. 2.4) to that of 36 other classifiers (accuracies obtained from the UEA/UCR repository Bagnall et al. 2017) including shape-based approaches like Euclidean or DTW nearest neighbor, ensembles of different elastic distance metrics (Lines and Bagnall 2015), interval methods, shapelets (Ye and Keogh 2009), dictionary based classifiers, or complex transformation ensemble classifiers that combine multiple time-series representations (COTE) (Bagnall et al. 2016). All comparisons are based on (class unbalanced) classification accuracies $a_{\mathrm{tot}}^{\mathrm{ub}}$ on a fixed train-test split obtained from UEA/UCR classification repository. As shown in Fig. 7a, most datasets exhibit similar performance between the alternative methods and *catch22*, with a majority of datasets exhibiting better performance using existing algorithms than *catch22*. However, despite drastic dimensionality reduction, our feature-based approach outperforms the existing methods on a range of datasets, some of which are labeled in Fig. 7a. To better understand the strengths and weaknesses of our low-dimensional feature-based representation of time series, we compared it directly to two of the most well-studied and purely shape-based time-series classification methods: Euclidean-1NN and DTW-1NN ('DTW-R1-1NN' in the UEA/UCR repository), as shown in Fig. 7b. There is an overall high correlation in performance across datasets, with a range of average performance (unbalanced classification rate on the given train-test partition $a_{\mathrm{tot}}^{\mathrm{ub}}$): *catch22* (69%), Euclidean 1-NN (71%), and DTW 1-NN (74%). The most interesting datasets are those for which one of the two approaches (shape-based or feature-based) markedly outperforms the other, as in these cases there is a clear advantage to tailoring your classification method to the structure of the data (Fulcher 2018); selected examples are annotated in Fig. 7b. We next investigate the characteristics of time-series datasets that make them better suited to different classification approaches.

3.7 Characteristics of datasets that favor feature- or shape-based representations

There is no single representation that is best for all time-series datasets, but rather, the optimal representation depends on the structure of the dataset and the questions being asked of it (Fulcher 2018). In this section we characterize the properties of

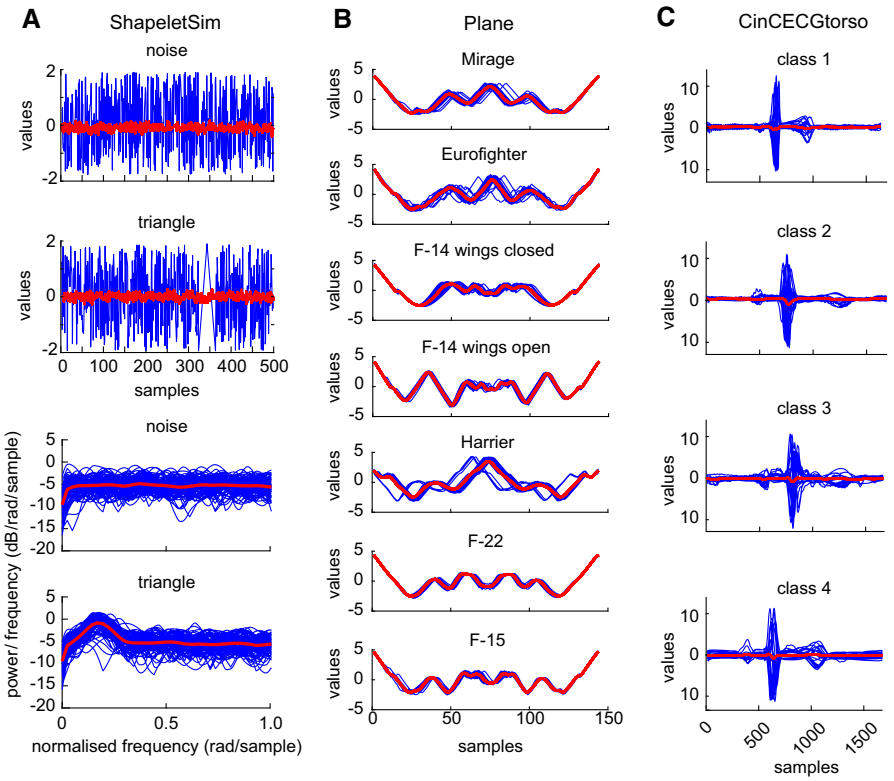

Fig. 7 Despite massive dimensionality reduction to 22 features, the ***catch22*** representation often achieves similar or better performance on time-series classification tasks. **a** Classification accuracy is plotted from using the feature-based *catch22* representation versus the performance of a range of existing methods across the 93 tasks in the UEA/UCR repository. Each dot represents the mean accuracy of alternative classifiers on a given dataset; error bars show the standard deviation over the 36 considered other methods containing simple full sequence shape-based approaches, over ensembles, shapelets, intervals, to complex transformation ensembles. An equality gray-dashed line is plotted, and regions in which *catch22* or other methods perform better are labeled. **b** The two purely shape-based classifiers, Euclidean (blue circles) and DTW (green circles) 1 nearest-neighbor, are compared against *catch22* features and a classification tree. All accuracies are unbalanced, as Eq. (1), and evaluated on the fixed train-test split provided in the UEA/UCR repository (Color figure online)

selected datasets that show a strong preference for either feature-based or shape-based classification, as highlighted in Fig. 7.

One striking example, is that of 'ShapeletSim', where the two labeled classes are much more accurately distinguished using the *catch22* feature-based representation (unbalanced accuracy a^{ub} of 100%) than by all but two existing methods [BOSS (Schäfer 2015) and Fast Shapelets (Rakthanmanon and Keogh 2013)] with a mean and standard deviation over all other classifiers of 69.0 ± 18.7% (DTW-1NN 65%, Euc-1NN 53.9%). To understand the discrepancy, we visualized the data in the time-domain, as shown in Fig. 8a (upper), where one example time series and the mean in each class are plotted, revealing no consistent time-domain shape across the 100

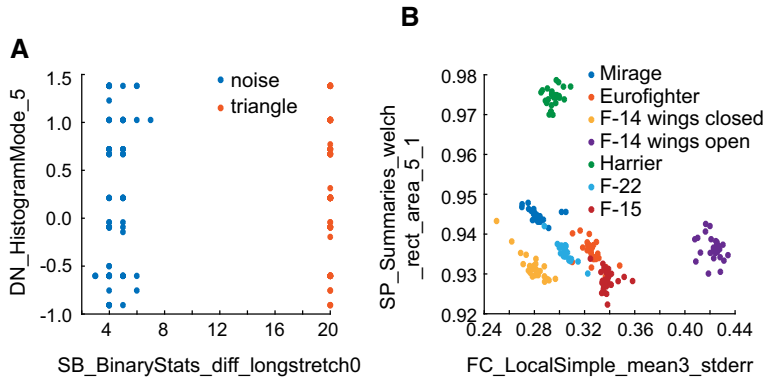

Fig. 8 Differences in the frequency domain are better picked up by features; subtle differences in shape are better detected by shape-based methods. Each subplot represents a class, blue lines show individual time series, red lines show an average over all time series in one class. **a** In the time domain (upper two plots), we show one example time series of the 'ShapeletSim' dataset (blue) and the average across all time series (red) for each class. The lower two plots display the Welch spectra of all time series individually in blue and the average over single time-series spectra in red. The mean spectra of the two classes differ visibly while there is no reliable difference in the time domain. **b** The individual (blue) and averaged (red) time series of the dataset 'Plane' should favor shape-based comparisons because of the highly reliable and aligned shapes in each class. **c** For the dataset 'CincECGtorso', all four classes can be well distinguished by their temporal offsets (Color figure online)

instances of each class. However, the two classes of time series are clearly distinguished by their frequency content, as shown in the corresponding Welch power spectra in Fig. 8a (lower). The features in *catch22* capture the temporal autocorrelation properties of each time series in various ways, facilitating an efficient representation to successfully capture class differences in 'ShapeletSim'; these differences cannot be captured straightforwardly from the time series' shape. In general, datasets without reliable shape differences between classes pose problems for time-domain distance metrics; consequently, the *catch22* feature-based representation often yields superior classification performance. Examples are 'USOLeaf' (86.7 *catch22* vs. 69.5±13.3% others; DTW-1NN 59.1%, Euc-1NN 52.1%), and 'SmallKitchenAppliances' (73.3% vs. 63.3 ± 12.1%; DTW-1NN 64.3%, Euc-1NN 34.4%).

An example of a dataset that is well-suited to shape-based classification is the seven-class 'Plane' dataset, shown in Fig. 8b. Apart from a minority of anomalous instances in e.g., the 'Harrier' class, each class has a subtle but robust shape, and these shapes are phase-aligned, allowing shape-based classifiers to accurately capture class differences. Despite being visually well-suited to shape-based classification, *catch22* captures the class differences with only a small reduction in accuracy a^{ub} (89.5%) compared to the shape-based classifiers (99.2 ± 1.4% over all given classifiers; DTW-1NN 100%, Euc-1NN 96.1%), demonstrating that feature-based representations can be versatile in capturing differences in time-series shape, despite a substantial reduction in dimensionality.

As a final example we consider the four classes of the 'CinCECGtorso' dataset, which are similarly accurately classified by our *catch22* feature-based method (78.9%) and the average existing classifier (81.3 ± 13.3%). Interestingly, when comparing

selected shape-based classifiers in Fig. 7, Euclidean-1NN (89.7%) outperforms the more complex DTW-1NN (65.1%). This difference in performance is due to the subtle differences in shape (particularly temporal offset of the deviation from zero) between the four classes, as shown in Fig. 8b. Simple time-domain distance metrics like Euclidean-1NN will capture these important differences well, whereas elastic distance measures like DTW shadow the informative temporal offsets. Converting to our feature-based representation discards most of the phase-information but still leads to a high classification accuracy.

3.8 Informative features provide understanding

Concise, low-dimensional summaries of time series, that exploit decades of interdisciplinary methods development for time-series analysis, are perhaps most important for scientists because they provide a means to understand class differences. Often a researcher will favor a method that provides interpretable understanding that can be used to motivate new solutions to a problem, even if it involves a small drop in classification accuracy relative to an opaque, black-box method. To demonstrate the ability of *catch22* to provide understanding into class difference, we projected all datasets into a two-dimensional feature space determined using sequential forward selection (Whitney 1971) as described in Sect. 2.9. Two examples are shown in Fig. 9. In the dataset 'ShapeletSim' (Fig. 9a), the simple feature, `SB_BinaryStats_diff_longstretch0`, clearly distinguishes the two classes. This simple measure quantifies the length of the longest continued descending increments in the data which enables a perfect separation of the two classes because time series of the 'triangle' class vary on a slower timescale than 'noise' time series.

In the most accurate two-dimensional feature space for the 7-class 'Plane' dataset, shown in Fig. 9b, each class occupies a distinctive part of the space. The first feature, `FC_LocalSimple_mean3_stderr` captures variability in residuals for local 3-sample mean predictions of the next datapoint applied to through time, while the second feature, `SP_Summaries_welch_rect_area_5_1`, captures the proportion of low-frequency power in the time series. We discover, e.g., that time series of 'F-14 wings open' are less predictable from a 3-sample running mean than other planes, and that time series of 'Harrier' planes exhibit a greater proportion of low-frequency power than other types of planes. Thus, in cases when both shape-based and feature-based methods exhibit comparable performance (unbalanced accuracies a^{ub} on given split: 89.5% by *catch22* vs. 99.1% mean over other classifiers), the ability to understand class differences can be a major advantage of the feature-based approach.

4 Discussion

Feature-based representations of time-series can distill complex time-varying dynamical patterns into a small set of interpretable characteristics that can be used to represent the data for applications like classification and clustering. Most importantly, features connect the data analyst to deeper theory, allowing interpretation of the properties

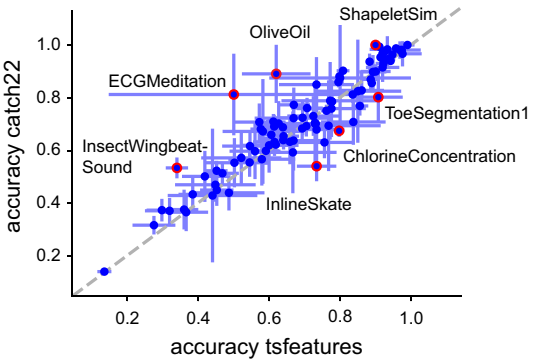

Fig. 9 Class differences can be interpreted using feature-based representations of time series. We plot a projection of time series into an informative two-dimensional feature space (estimated from *catch22* using sequential forward selection, see Sect. 2.9), where each time series is a point in the space and colored by its class label. Plots are shown for two datasets: **a** 'ShapeletSim', and **b** 'Plane'; in both cases, all labeled classes are clearly distinguished in the space. In 'ShapeletSim', `SB_BinaryStats_diff_longstretch0`, which calculates the length of the longest run of consecutive decreases in the time series. The two features selected for the 'Plane' dataset are the local predictability measure, `FC_LocalSimple_mean3_stderr`, and the low-frequency power estimate, `SP_Summaries_welch_rect_area_5_1` (Color figure online)

of the data that facilitate successful performance. While large feature libraries have helped to overcome the limitations of manual, subjective duration of time-series features, they are inefficient and computationally expensive. Overcoming this limitation, here we introduce a methodology to generate small, canonical subsets of features that each display high classification performance across a given ensemble of tasks, and exhibit complementary performance characteristics with each other. We apply the method to a set of 93 classification tasks from the UEA/UCR repository, showing how a large library of 4791 features can be reduced to a canonical subset of just 22 features, *catch22*, which displays similar classification accuracy as the full set (relative reduction of 7.5% on average, 77.2% vs. 71.7%), computes quickly ($<$ 0.5 s/10,000 samples), scales approximately linearly with time-series length ($\mathcal{O}(N^{1.16})$), and allows the investigator to learn and understand what types of dynamical properties distinguish the labeled classes of their dataset. Compared to shape-based methods like dynamic time warping (DTW), *catch22* gives comparable, and often superior classification performance, despite substantial dimensionality reduction. Using case studies, we explain why some datasets are better suited to shape-based classification (e.g., there are characteristic aligned shapes within each class), while others are better suited to feature-based classification (e.g., where classes do not have a characteristic, temporally aligned shape, but have characteristic dynamical properties that are encapsulated in one or more time-series features).

While some applications may be able to justify the computational expense of searching across a large feature library such as *hctsa* (Fulcher and Jones 2014, 2017), the availability of an efficient, reduced set of features, as *catch22*, will make the advantages of feature-based time-series classification and clustering more widely accessible. As an example application *catch22* is being used in the self-organizing time-series database

for data-driven interdisciplinary collaboration *CompEngine* to assess the similarity of recordings (Fulcher et al. 2019). Unlike the Matlab-based *hctsa*, *catch22* does not require a commercial license to run, computes efficiently, and scales approximately linearly with time-series length in the cases we tested. This makes it straightforwardly applicable to much longer time series than are typically considered in the time-series classification literature, e.g., for a 10,000-sample time series, *catch22* computes in 0.5 s. As well as being suitable for long recordings, feature-based representations do not require all time series to be the same length (unlike conventional shape-based classifiers), opening up the feature-based approach to new types of datasets – and indeed new types of analyses. Even though *catch22* is selected here based on classification performance , the availability and ease of computing *catch22* opens up applications to areas including feature-based time-series modeling, forecasting, anomaly detection, motif discovery, and others. To facilitate its adoption, we provide an efficient C-implementation of *catch22*, with wrappers for Matlab, Python, and R.

We have shown that the most useful representation of a time series varies widely across datasets, with some problems better suited to feature-based classification, and others better suited to shape-based classification. The 22 features selected here are tailored to the properties of the UEA/UCR datasets (which are typically short and phase-aligned), but the method we present here is general and could be used to generate reduced feature sets tailored to any application domain of interest that allows individual features to be assigned performance scores. For example, given a different set of classification datasets where key class differences are the result of subtle variations in dynamical properties in long streams of time-series data, we would obtain a canonical set that might include features of long-range automutual information or measures the nonlinear time-series analysis literature: very different features to the relatively simple measures contained in *catch22*. As new time-series datasets are added to the UEA/UCR repository, that better capture the diversity of time-series data studied across industry and science, our feature reduction method could be rerun to extract new canonical feature sets that reflect the types of time-series properties that are important to measure in the new data. Note that hybrid methods such as COTE (Bagnall et al. 2016), which are not limited to a single time-series representation but can adapt to the problem at hand, consistently outperform both the shape-based existing classifiers and our features at the price of a much higher computational effort. Given its computational efficiency, *catch22* could be incorporated straightforwardly in these ensemble-based frameworks of multiple representations. Here we excluded features that are sensitive to the location and spread of the data distribution, to ensure a fair comparison to shape-based methods which use normalized data; but for many real-world applications these could be highly relevant and should therefore be retained to allow improvements in classification accuracy. Our selection pipeline is agnostic to the classification task collection used and can in principle be generalized beyond classification tasks to different time-series analyses as well. Here we score the performance of each feature on a given task as the classification accuracy, but this metric could be adapted to allow application to regression problems (correlation of a feature with the exogenous target variable), forecasting problems (prediction error), and clustering problems (separation of known clusters). The proposed method has the advantage of identifying individually informative estimators and transparently grouping features

into similarly performing clusters for enhanced interpretability. Still, other approaches for selecting feature subsets exist, such as sequential forward selection or LASSO, and it would be interesting to compare the results of alternative pipelines building on these existing selection techniques with our results in future work.

In conclusion, here we present *catch22*, a concise, accessible feature-based summary of an interdisciplinary time-series analysis literature for use in time-series classification tasks. We hope that the ability to readily leverage feature-based representations of time series—and to generate new reduced feature sets tailored to specific domain problems—will aid diverse applications involving time series.

Information sharing statement

The C-implementation of our canonical features along with their wrapped versions for R, Python and Matlab can be accessed on GitHub repository https://github.com/chlubba/catch22.
The selection pipeline is accessible on https://github.com/chlubba/op_importance.

Acknowledgements CL thanks Engineering and Physical Sciences Research Council (EPSRC) Grant EP/L016737/1 and Galvani Bioelectronics. SSS is supported by the Natural Environment Research Council through the Science and Solutions for a Changing Planet DTP. BDF is supported by the National Health and Medical Research Council (NHMRC) Grant, 1089718. NJ thanks EPSRC Grants EP/N014529/1 and EP/K503733/1. We further thank the authors behind the UEA/UCR time series classification repository for the valuable data that made this project possible. Funding was provided by Natural Environment Research Council.

5 Appendix

5.1 Manually replaced features

Table 2 lists the best five features of each cluster in which a feature has been manually exchanged. The full list of 710 features grouped by cluster can be accessed as a "Appendix" file.

Table 2 The five best features of each cluster with manual replacement

hctsa feature name	z-scored combined accuracy
Cluster 3	
EX_MovingThreshold_01_01_meankickf	1.44
PH_Walker_prop_01_res_ac1	1.39
SP_Summaries_welch_rect_area_5_1	**1.18**
PH_Walker_prop_11_w_std	1.17
IN_AutoMutualInfoStats_40_gaussian_pextrema	1.13
Cluster 6	
MF_arfit_1_8_sbc_meanA	2.57
FC_LocalSimple_mean3_stderr	**2.26**
FC_LocalSimple_mean4_stderr	2.25
NL_embed_PCA_1_10_std	2.19
NL_embed_PCA_1_10_perc_1	2.17
Cluster 10	
SB_TransitionMatrix_2ac_sumdiagcov	2.51
SB_TransitionMatrix_2ac_T3	2.25
SB_TransitionMatrix_2ac_ondiag	2.18
SB_TransitionMatrix_2ac_T1	2.07
SB_BinaryStats_mean_longstretch1	**2.05**
Cluster 17	
SB_TransitionMatrix_3ac_maxeigcov	2.65
SB_TransitionMatrix_3ac_stdeigcov	2.56
SB_TransitionMatrix_3ac_stdeig	2.53
SB_TransitionMatrix_5ac_stdeigcov	2.53
SB_TransitionMatrix_3ac_sumdiagcov	**2.46**
Cluster 18	
SB_TransitionMatrix_41_ondiag	1.83
SB_TransitionMatrix_51_ondiag	1.80
EN_SampEn_5_03_sampen1	1.54
CO_HistogramAMI_quantiles_10_1	1.51
MD_hrv_classic_pnn40	**1.50**
Cluster 19	
PH_Walker_prop_01_sw_propcross	1.99
IN_AutoMutualInfoStats_40_gaussian_fmmi	**1.92**
SB_TransitionMatrix_31_ondiag	1.52
PH_Walker_prop_01_sw_taudiff	1.47
CO_Embed2_Basic_1_updiag01	1.40

The feature in bold was used instead of the top feature

5.2 Insignificant features

The features listed in Table 3 were found to exhibit a classification performance across tasks consistent with a random-number generator.

Table 3 The 145 features listed here exhibited classification performance consistent with a random-number generator

p-value	name
1.000	WL_coeffs_db3_4.wb99m
1.000	WL_coeffs_db3_3.wb99m
1.000	WL_coeffs_db3_2.wb99m
1.000	WL_coeffs_db3_1.wb99m
1.000	SY_LocalGlobal_unicg500.iqr
1.000	SP_Summaries_welch_rect.linfitloglog_hf_sigrat
1.000	SP_Summaries_pgram_hamm.linfitloglog_mf_sigrat
1.000	SP_Summaries_pgram_hamm.linfitloglog_hf_sigrat
1.000	SP_Summaries_pgram_hamm.linfitloglog_all_sigrat
1.000	SP_Summaries_fft_logdev.logstd
1.000	SP_Summaries_fft_logdev.logiqr
1.000	SP_Summaries_fft_logdev.linfitsemilog_all_sigrat
1.000	SP_Summaries_fft_logdev.linfitsemilog_all_sigma
1.000	SP_Summaries_fft_logdev.linfitsemilog_all_sea1
1.000	SP_Summaries_fft_logdev.linfitsemilog_all_a2
1.000	SP_Summaries_fft_logdev.linfitloglog_mf_sigrat
1.000	SP_Summaries_fft_logdev.linfitloglog_mf_sigma
1.000	SP_Summaries_fft_logdev.linfitloglog_mf_sea1
1.000	SP_Summaries_fft_logdev.linfitloglog_mf_a2
1.000	SP_Summaries_fft_logdev.linfitloglog_lf_sigrat
1.000	SP_Summaries_fft_logdev.linfitloglog_hf_sigrat
1.000	SP_Summaries_fft_logdev.linfitloglog_hf_sigma
1.000	SP_Summaries_fft_logdev.linfitloglog_hf_sea1
1.000	SP_Summaries_fft_logdev.linfitloglog_hf_a2
1.000	SP_Summaries_fft_logdev.linfitloglog_all_sigrat
1.000	SP_Summaries_fft_logdev.linfitloglog_all_sigma
1.000	SP_Summaries_fft_logdev.linfitloglog_all_sea1
1.000	SD_TSTL_surrogates_1_100_3_tc3.stdsurr
1.000	SD_TSTL_surrogates_1_100_3_tc3.normpatponmax
1.000	SD_TSTL_surrogates_1_100_3_tc3.meansurr
1.000	SD_TSTL_surrogates_1_100_3_tc3.kspminfromext
1.000	SD_TSTL_surrogates_1_100_3_tc3.ksphereonmax
1.000	SD_TSTL_surrogates_1_100_3_tc3.ksiqrsfrommode
1.000	SD_TSTL_surrogates_1_100_2_trev.meansurr

Table 3 continued

p-value	name
1.000	NL_TSTL_acp_1_001_025_10_05.macpfdrop_8
1.000	NL_TSTL_acp_1_001_025_10_05.macpfdrop_7
1.000	NL_TSTL_acp_1_001_025_10_05.macpfdrop_6
1.000	NL_TSTL_acp_1_001_025_10_05.macpfdrop_5
1.000	NL_TSTL_acp_1_001_025_10_05.macpfdrop_3
1.000	NL_TSTL_LargestLyap_n1_01_001_3_1_4.vse_rmsres
1.000	NL_TSTL_LargestLyap_n1_01_001_3_1_4.vse_meanabsres
1.000	NL_MS_nlpe_fnn_mi.maxonmean
1.000	NL_MS_nlpe_fnn_mi.acmnd0
1.000	NL_MS_nlpe_fnn_mi.ac3n
1.000	NL_MS_nlpe_2_mi.maxonmean
1.000	MS_shannon_4_1t10.stdent
1.000	MS_shannon_4_1t10.maxent
1.000	MS_shannon_3_1t10.stdent
1.000	MS_shannon_3_1t10.maxent
1.000	MS_shannon_2_1t10.maxent
1.000	MF_armax_3_1_05_1.lastimprovement
1.000	MF_armax_3_1_05_1.ac3n
1.000	MF_armax_2_2_05_1.lastimprovement
1.000	MF_StateSpace_n4sid_3_05_1.ac3n
1.000	MF_StateSpace_n4sid_3_05_1.ac2n
1.000	MF_GP_LocalPrediction_covSEiso_covNoise_5_3_10_beforeafter.minstderr_run
1.000	MF_GP_LocalPrediction_covSEiso_covNoise_5_3_10_beforeafter.maxstderr_run
1.000	MF_GP_LocalPrediction_covSEiso_covNoise_10_3_20_randomgap.minstderr_run
1.000	MF_GP_LocalPrediction_covSEiso_covNoise_10_3_20_randomgap.maxstderr_run
1.000	MF_GP_LocalPrediction_covSEiso_covNoise_10_3_20_frombefore.minstderr_run
1.000	MF_CompareTestSets_y_ar_4_rand_25_01_1.ac1s_iqr
1.000	CO_AddNoise_ac_kraskov1_4.ami_at_15
0.997	MS_shannon_4_1t10.meanent
0.994	NL_TSTL_acp_1_001_025_10_05.macpfdrop_9
0.979	MF_armax_2_2_05_1.ac3n
0.963	SP_Summaries_fft_logdev.linfitloglog_lf_sigma
0.834	MS_shannon_2t10_2.stdent
0.830	TSTL_delaytime_01_1.stdtau
0.830	MF_steps_ahead_ar_best_6.ac1_1
0.810	SP_Summaries_welch_rect.linfitloglog_mf_sigrat
0.810	SP_Summaries_fft_logdev.linfitloglog_lf_sea1
0.781	NL_TSTL_acp_1_001_025_10_05.macpfdrop_1
0.712	SP_Summaries_welch_rect.linfitloglog_hf_sigma

Table 3 continued

p-value	name
0.703	SP_Summaries_welch_rect.linfitloglog_lf_sigrat
0.703	MF_StateSpaceCompOrder_8.maxdiffaic
0.699	NL_TSTL_acp_1_001_025_10_05.macpfdrop_2
0.699	NL_MS_nlpe_fnn_mi.p3_5
0.697	PH_ForcePotential_dblwell_3_001_01.finaldev
0.608	NL_MS_nlpe_fnn_mi.ac2n
0.561	NL_MS_nlpe_fnn_mi.meane
0.561	NL_MS_nlpe_2_mi.ac3n
0.511	CO_AddNoise_ac_kraskov1_4.ami_at_10
0.484	CO_AddNoise_ac_kraskov1_4.ami_at_20
0.432	WL_coeffs_db3_max.wb99m
0.375	NL_MS_nlpe_fnn_mi.p5_5
0.354	SP_Summaries_fft_logdev.linfitloglog_hf_a1
0.354	NL_MS_nlpe_fnn_mi.p2_5
0.300	NL_MS_nlpe_2_mi.p4_5
0.295	SP_Summaries_welch_rect.linfitloglog_hf_sea1
0.286	NL_TSTL_acp_1_001_025_10_05.stdmacpfdiff
0.286	CO_StickAngles_y.ratmean_p
0.278	CO_AddNoise_ac_std1_10.ami_at_10
0.261	MF_armax_2_2_05_1.ac2n
0.227	NL_MS_nlpe_2_mi.acmnd0
0.223	NL_MS_nlpe_fnn_mi.ac3
0.222	MF_CompareTestSets_y_ar_best_uniform_25_01_1.ac1s_iqr
0.222	CO_AddNoise_1_std1_10.pdec
0.218	CO_AddNoise_1_kraskov1_4.ami_at_15
0.183	CO_AddNoise_ac_std1_10.ami_at_15
0.172	Y_LocalGlobal_unicg100.iqr
0.170	MF_armax_2_2_05_1.p5_5
0.166	SP_Summaries_fft_logdev.linfitloglog_all_a2
0.152	Y_SlidingWindow_lil_ent10_1
0.145	PH_ForcePotential_dblwell_2_005_02.finaldev
0.140	MF_StateSpace_n4sid_2_05_1.ac3n
0.133	Y_LocalGlobal_unicg50.iqr
0.129	NL_MS_nlpe_fnn_mi.acsnd0
0.129	CO_AddNoise_ac_std1_10.pdec
0.127	CO_AddNoise_1_kraskov1_4.ami_at_20
0.127	MS_shannon_2t10_2.medent
0.124	MF_CompareTestSets_y_ss_best_uniform_25_01_1.ac1s_iqr
0.119	SP_Summaries_welch_rect.linfitloglog_all_sigrat

Table 3 continued

p-value	name
0.105	NL_MS_nlpe_2_mi.ac2n
0.104	NL_TSTL_acp_1_001_025_10_05.macpfdrop_4
0.096	MS_shannon_2t10_3.stdent
0.090	MS_shannon_3_2
0.088	MS_shannon_3_1t10.meanent
0.083	MF_GP_LocalPrediction_covSEiso_covNoise_10_3_20_frombefore.maxerrbar
0.080	SP_Summaries_fft_logdev.logarea_3_3
0.080	FC_LoopLocalSimple_mean.sws_stdn
0.077	SP_Summaries_fft_logdev.logarea_5_5
0.074	MS_shannon_2t10_4.stdent
0.074	MF_armax_3_1_05_1.ac2n
0.069	SP_Summaries_fft_logdev.logarea_4_4
0.068	ST_LocalExtrema_n25.minabsmin
0.068	NL_MS_nlpe_2_mi.p3_5
0.068	MS_shannon_4_1t10.medent
0.068	MF_FitSubsegments_arma_2_2_uniform_25_01.q_2_std
0.068	SP_Summaries_fft_logdev.q25
0.061	NL_MS_nlpe_2_mi.p2_5
0.060	MF_FitSubsegments_arma_2_2_uniform_25_01.q_1_max
0.060	MS_shannon_2_1t10.stdent
0.059	NL_TSTL_acp_mi_1__10.iqracpf_1
0.059	NL_MS_nlpe_fnn_mi.ac1n
0.059	MF_FitSubsegments_arma_2_2_uniform_25_01.q_2_min
0.059	MF_CompareTestSets_y_ss_2_uniform_25_01_1.ac1s_iqr
0.059	NL_MS_nlpe_2_mi.p5_5
0.056	MF_StateSpace_n4sid_2_05_1.ac2n
0.056	DN_SimpleFit_gauss2_hsqrt.resAC2
0.056	CO_AddNoise_ac_std1_10.ami_at_20
0.056	SP_Summaries_fft_logdev.logarea_2_2
0.055	MF_FitSubsegments_arma_2_2_uniform_25_01.q_2_max
0.054	NL_TSTL_acp_mi_1__10.macpfdrop_4
0.050	Y_LocalGlobal_unicg20.iqr
0.050	NL_MS_nlpe_2_mi.ac2

5.3 Time series for computation time evaluation

A selection of 40 time series was obtained from the dataset '1000 Empirical Time series' (Fulcher 2017) (Table 4).

Table 4 40 empirical time series selected for evaluating the computation times of features

ID	Name	Keywords
1	NS_beta_L10000_a1_b3_2.dat	Synthetic, noise, beta
25	ST_M5a_N10000_a-0.01_b-0.6_c0_d0_x0_1_2.dat	Synthetic, stochastic, SDE,M5
53	SY_rwalk_L10000_20.dat	Synthetic, randomwalk
75	FL_ddp_L300_N5000_IC_0.1_0.1_y.dat	Synthetic,dynsys,ddp
106	FL_lorenz_L250_N10000_IC_0_-0.01_9.1_y.dat	Synthetic, dynsys, chaos, lorenz
125	FL_shawvdp_L300_N5000_IC_1.3_0.1_x.dat	Synthetic, dynsys, shawvdp
158	MP_burgers_L300_IC_-0.2_0.1_x.dat	Synthetic, map, burgers
175	MP_chirikov_L300_IC_0.2_6_y.dat	Synthetic, map, chirikov
211	MP_henon_L1000_a1.4_b0.3_IC_0_0.9.dat	Synthetic, map, henon
225	MP_holmescubic_L300_IC_1.7_0_y.dat	Synthetic, map, holmescubic
263	MP_lorenz3d_L300_IC_0.51_0.5_-1_x.dat	Synthetic, map, chaos, lorenz3d
275	MP_pinchers_L5000_s2.1c_0.55.dat	Synthetic, map, pinchers
316	MP_spence_L5000_x0_0.27.dat	Synthetic, map, spence
325	MP_tent_L5000_A1.88.dat	Synthetic, map, tent, chaos
369	SY_MA_L500_p8_7.dat	Synthetic, MA, MA8
375	SY_MIX_p0.3_L5000_5.dat	Synthetic, MIXP, MIX0.3
421	FI_yahoo_HL_KLSE.dat	Finance, yahoo, opening
425	FI_yahoo_HL_z28_SPMIB.dat	Finance, yahoo, opening
474	FL_dblscroll_L1000_N5000_IC_0.01_0.01_0_z.dat	Synthetic, dynsys, chaos, dblscroll
475	FL_dblscroll_L200_N10000_IC_0.01_0.01_0_z.dat	Synthetic, dynsys, chaos, dblscroll
525	FL_moorespiegel_L250_N1000_IC_0.1_0_0_x.dat	Synthetic, dynsys, chaos, moorespiegel
526	FL_moorespiegel_L250_N5000_IC_0.1_0_0_x.dat	Synthetic, dynsys, chaos, moorespiegel
575	FL_simpqcf_L1000_N10000_IC_-0.9_0_0.5_z.dat	Synthetic, dynsys, chaos, simpqcf
579	FL_simpqcf_L2000_N1000_IC_-0.9_0_0.5_z.dat	Synthetic, dynsys, chaos, simpqcf
625	MP_Lozi_iii.dat	Synthetic, map, chaos, lozi
631	MP_freitas_nlma_L500_a-3.92_b-3.1526_1.dat	Synthetic, map, nonlinear, freita
675	TR_arge030_rf.dat	Treerings
684	AS_s2.2_f4_b8_l8800_58925.dat	Sound, animalsounds
725	FI_yahoo_Op_IFL.L_LOGR.dat	Finance, logr
737	Ich_recflow.dat	Meteorology, riverflow, reconstructed, UK
775	SF_E_5.dat	SantaFe, astronomy
789	SPIDR_hpidmsp_F15_meas.dat	Space, hpidmsp
825	SPIDR_meanDelay_ACE_hrly.dat	Space, magneticfield
842	SPIDR_vostok_L6600_Sep2002_vostok.dat	Space, vostok
875	t_osaka_rf.dat	Meteorology, temperature

Table 4 continued

ID	Name	Keywords
894	MD_chfdb_chf07_seg039_SNIP_9574-17073.dat	Medical, physionet, ecg, chfdb, snip
925	MD_tremordb_g12ren.dat	Medical, tremor, physionet, lowamp, dbson, medon, gpi
947	MUS_Tetra-Sync_1364s_F0.02_b8.dat	Sound, music, downed
976	MD_nsrdb_nsr19088_seg007_SNIP_5659-15658.dat	Medical, physionet, ecg, nsrdb, snip
1000	MD_mghdb_mgh79_PAP_SNIP_9047-15146.dat	Medical, physionet, mghdb, snip, pulmonaryarterialpressure

5.4 Performance comparison with *tsfeatures*

The list of the 16 default features from *tsfeatures* that we used for a performance comparison with *catch22* are in Table 5.

Table 5 The 16 features of `tsfeatures` we used for classification

`frequency`	`nperiods`	`seasonal_period`	`trend`
`spike`	`linearity`	`curvature`	`e_acf1`
`e_acf10`	`entropy`	`x_acf1`	`x_acf10`
`diff1_acf1`	`diff1_acf10`	`diff2_acf1`	`diff2_acf10`

References

Bagnall A, Davis LM, Hills J, Lines J (2012) Transformation based ensembles for time series classification. In: Proceedings of the 2012 SIAM international conference on data mining, pp 307–318. ISBN 978-1-61197-232-0

Bagnall A, Lines J, Hills J, Bostrom A (2016) Time-series classification with COTE: the collective of transformation-based ensembles. In: 2016 IEEE 32nd international conference on data engineering, ICDE, vol 27, no 9, pp 1548–1549, 2016. ISSN 10414347. https://doi.org/10.1109/ICDE.2016.7498418

Bagnall A, Lines J, Bostrom A, Large J, Keogh E (2017) The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Min Knowl Discov 31(3):606–660. ISSN 1573756X. https://doi.org/10.1007/s10618-016-0483-9

Bagnall A, Lines J, Vickers W, Keogh E The UEA & UCR time series classification repository. http://www.timeseriesclassification.com/

Bandara K, Bergmeir C, Smyl S (2017) Forecasting across time series databases using long short-term memory networks on groups of similar series: a clustering approach. arXiv. ISSN 13578170. https://doi.org/10.1002/pdi.718. http://arxiv.org/abs/1710.03222

Berndt D, Clifford J (1994) Using dynamic time warping to find patterns in time series. In: Workshop on knowledge knowledge discovery in databases, vol 398, pp 359–370. ISBN 0-929280-73-3

Biason A, Pielli C, Rossi M, Zanella A, Zordan D, Kelly M, Zorzi M (2017) EC-CENTRIC: an energy- and context-centric perspective on IoT systems and protocol design. IEEE Access 5:6894–6908. ISSN 21693536. https://doi.org/10.1109/ACCESS.2017.2692522

Dau HA, Bagnall A, Kamgar K, Yeh CM, Zhu Y (2018) UCR time series archive 2018. arXiv

Faloutsos C, Ranganathan M, Manolopoulos Y (1994) Fast subsequence matching in time-series databases. In: SIGMOD '94 proceedings of the 1994 ACM SIGMOD international conference on management of data, pp 419–429

Fisher RA (1925) Statistical methods for research workers. ISBN 978-1614271666. 52, 281–302

Fulcher BD (2017) 1000 empirical time series

Fulcher BD (2018) Feature-based time-series analysis. In: Dong G, Liu H (eds) Feature engineering for machine learning and data analytics, chap 4, pp 87–116. CRC Press

Fulcher BD, Jones NS (2014) Highly comparative feature-based time-series classification. IEEE Trans Knowl Data Eng 26(12):3026–3037. ISSN 10414347. https://doi.org/10.1109/TKDE.2014.2316504

Fulcher BD, Jones NS (2017) hctsa: a computational framework for automated time-series phenotyping using massive feature extraction. Cell Syst 5(5):527–531. ISSN 24054720. https://doi.org/10.1016/j.cels.2017.10.001

Fulcher BD, Little MA, Jones NS (2013) Highly comparative time-series analysis: the empirical structure of time series and their methods. J R Soc Interface 10(83):20130048. ISSN 1742-5662. https://doi.org/10.1098/rsif.2013.0048

Fulcher BD, Lubba CH, Sethi S, Jones NS (2019) CompEngine: a self-organizing, living library of time-series data (in submission)

Holm S (1979) A simple sequentially rejective multiple test procedure. Scand J Stat 6:65–70. ISSN 03036898. https://doi.org/10.2307/4615733

Hyndman RJ, Wang E, Laptev N (2016) Large-scale unusual time series detection. In: Proceedings—15th IEEE international conference on data mining workshop, ICDMW 2015, pp 1616–1619. ISSN 2375-9259. https://doi.org/10.1109/ICDMW.2015.104

Hyndman RJ, Wang E, Kang Y, Talagala T, Taieb SB (2019) tsfeatures: time series feature extraction. https://github.com/robjhyndman/tsfeatures

Lines J, Bagnall A (2015) Time series classification with ensembles of elastic distance measures. Data Min Knowl Discov 29(3):565–592. ISSN 13845810. https://doi.org/10.1007/s10618-014-0361-2

Mietus JE (2002) The pNNx files: re-examining a widely used heart rate variability measure. Heart 88(4):378–380. ISSN 00070769. https://doi.org/10.1136/heart.88.4.378

Moon Y-S, Whang K-Y, Loh W-K (2001) Duality-based subsequence matching in time-series databases. In: Proceedings 17th international conference on data engineering, pp 263–272. ISSN 1063-6382. https://doi.org/10.1109/ICDE.2001.914837

Mörchen F (2003) Time series feature extraction for data mining using DWT and DFT. Technical Report, 33

Nanopoulos A, Alcock RJ, Manolopoulos Y (2001) Feature-based classification of time-series data. Int J Comput Res 10(3):

Rakthanmanon T, Keogh E (2013) Fast shapelets: a scalable algorithm for discovering time series shapelets. In: Proceedings of the 2013 SIAM international conference on data mining, pp 668–676. ISSN 1063-4266. https://doi.org/10.1137/1.9781611972832.74. https://doi.org/10.1137/1.9781611972832.74

Schäfer P (2015) The BOSS is concerned with time series classification in the presence of noise. Data Min Knowl Discov 29(6):1505–1530. ISSN 13845810. https://doi.org/10.1007/s10618-014-0377-7

Sethi SS, Zerbi V, Wenderoth N, Fornito A, Fulcher BD (2017) Structural connectome topology relates to regional BOLD signal dynamics in the mouse brain. Chaos 27(4). ISSN 10541500. https://doi.org/10.1063/1.4979281

Shekar AK, Pappik M, Iglesias Sánchez P, Müller E (2018) Selection of relevant and non-redundant multivariate ordinal patterns for time series classification. In: Larisa S, Joaquin V, George P, Michelangelo C (eds) Discovery science. Springer International Publishing, Cham, pp 224–240 (ISBN 978-3-030-01771-2)

Timmer J, Gantert C, Deuschl G, Honerkamp J (1993) Characteristics of hand tremor time series. Biol Cybern 70(1):75–80. ISSN 03401200. https://doi.org/10.1007/BF00202568

Vlachos M, Kollios G, Gunopulos D (2002) Discovering similar multidimensional trajectories. In: Data mining and knowledge discovery, p 673. ISBN 978-3-319-23519-6. https://doi.org/10.1007/978-3-319-23519-6_1401-2

Wang X, Smith K, Hyndman R (2006) Characteristic-based clustering for time series data. Data Min Knowl Discov 13(3):335–364. ISSN 13845810. https://doi.org/10.1007/s10618-005-0039-x

Wang X, Wirth A, Wang L (2007) Structure-based statistical features and multivariate time series clustering. In: Proceedings—IEEE international conference on data mining, ICDM, pp 351–360. ISSN 15504786. https://doi.org/10.1109/ICDM.2007.103

Whitney AW (1971) A direct method of nonparametric measurement selection. IEEE Trans Comput 20(September):1100–1103

Williams J (2014) Clustering household electricity use profiles. In: MLSDA '13 Proceedings of workshop on machine learning for sensory data analysis (December 2013), pp 19–26. https://doi.org/10.1145/2542652.2542656

Ye L, Keogh E (2009) Time series shapelets. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining—KDD '09, p 947. https://doi.org/10.1145/1557019.1557122

Affiliations

Carl H. Lubba[1] · Sarab S. Sethi[2] · Philip Knaute[2] · Simon R. Schultz[1] · Ben D. Fulcher[3] · Nick S. Jones[2]

Carl H. Lubba
c.lubba15@imperial.ac.uk

Sarab S. Sethi
s.sethi16@imperial.ac.uk

Philip Knaute
philip.knaute@gmail.com

Simon R. Schultz
s.schultz@imperial.ac.uk

1 Department of Bioengineering, Imperial College London, South Kensington, London SW7 2AZ, UK

2 Department of Mathematics, Imperial College London, South Kensington, London SW7 2AZ, UK

3 School of Physics, Faculty of Science, The University of Sydney, Camperdown, NSW 2006, Australia