Statistical Learning for Process Data

Session 2: Introduction to ProcData Package

Xueying Tang

Department of Mathematics
University of Arizona

June 4, 2021

Agenda

- Overview of ProcData package

Agenda

- Overview of ProcData package
- Installation

Agenda

- Overview of ProcData package
- Installation
- Data input/output, structure, and processing
- Feature extraction
- Sequence models

ProcData Overview

- ProcData provides tools for processing and analyzing process data.

ProcData Overview

- ProcData provides tools for processing and analyzing process data. It includes
  - an S3 class `"proc"` for response processes

ProcData Overview

- ProcData provides tools for processing and analyzing process data. It includes
  - an S3 class `"proc"` for response processes
  - functions for data processing

ProcData Overview

- ProcData provides tools for processing and analyzing process data. It includes
  - an S3 class `"proc"` for response processes
  - functions for data processing
  - state-of-the-art methods for process data analysis

ProcData Overview

- ProcData provides tools for processing and analyzing process data. It includes
  - an S3 class `"proc"` for response processes
  - functions for data processing
  - state-of-the-art methods for process data analysis
    - feature extraction methods
    - sequence models

ProcData Overview

Category	Functions
Data input/output	`read.seqs`, `write.seqs`
Data structure	`proc`, `print.proc`, `summary.proc`, `cc_data`
Data processing	`remove_repeat`, `remove_action`, `replace_action`, `combine_actions`
Feature extraction	`seq2feature_mds`, `chooseK_mds`, `seq2feature_seq2seq`†, `chooseK_seq2seq`†
Sequence model	`seqm`†, `predict.seqm`†

Table: Summary of `ProcData`. Functions marked by † depend on `keras`.

Installation

1. Install `ProcData` and dependent `R` packages from CRAN

```
install.packages("ProcData", dependencies=T)
```

   or from GitHub:

```
devtools::install_github("xytangtang/ProcData",
                         dependencies=T)
```

Installation

1. Install `ProcData` and dependent `R` packages from CRAN

   ```
   install.packages("ProcData", dependencies=T)
   ```

   or from GitHub:

   ```
   devtools::install_github("xytangtang/ProcData",
                            dependencies=T)
   ```

2. Install dependent Python libraries

   ```
   library(keras)
   install_keras()
   ```

Installation

1. Install `ProcData` and dependent `R` packages from CRAN

   ```
   install.packages("ProcData", dependencies=T)
   ```

   or from GitHub:

   ```
   devtools::install_github("xytangtang/ProcData",
                            dependencies=T)
   ```

2. Install dependent Python libraries

   ```
   library(keras)
   install_keras()
   ```

   - If this step is skipped or fails, calling functions marked by † in the table on Page 4 will lead to an error.
   - Getting help for installation: `https://github.com/xytangtang/ProcData/issues`

Data Input/Output

File Style "single"

	A	B	C	D	E	F
1	Country	Gender	Age	ID	Action	Time
2	US	F	18	101	Start,CHECK_A,End	0,6803,8774
3	US	M	35	102	Start,OPT1_1,OPT2_1,RUN,CHECK_D,End	0,18263,21010,22034,42015,44132
4	US	F	31	103	Start,OPT1_3,OPT2_2,RUN,CHECK_A,End	0,71910,75087,81733,105105,129596
5	JP	M	22	104	Start,OPT1_1,OPT2_2,RUN,OPT1_3,OPT2_2,	0,81733,95466,98860,105105,106560,111(
6	JP	F	40	105	Start,CHECK_C,End	0,3325,3568
7						

File Style "multiple"

	A	B	C	D	E	F
1	Country	Gender	Age	ID	Action	Time
2	US	F	18	101	Start	0
3	US	F	18	101	CHECK_A	6803
4	US	F	18	101	End	8774
5	US	M	35	102	Start	0
6	US	M	35	102	OPT1_1	18263
7	US	M	35	102	OPT2_1	21010
8	US	M	35	102	RUN	22034
9	US	M	35	102	CHECK_D	42015
10	US	M	35	102	End	44132

Data Input/Output

- Two file styles: single or multiple

Data Input/Output

- Two file styles: single or multiple
- `read.seqs`: read process data from a CSV file as a `proc` object

Data Input/Output

- Two file styles: single or multiple
- `read.seqs`: read process data from a CSV file as a `proc` object
- `write.seqs`: write a `proc` object to a CSV file.

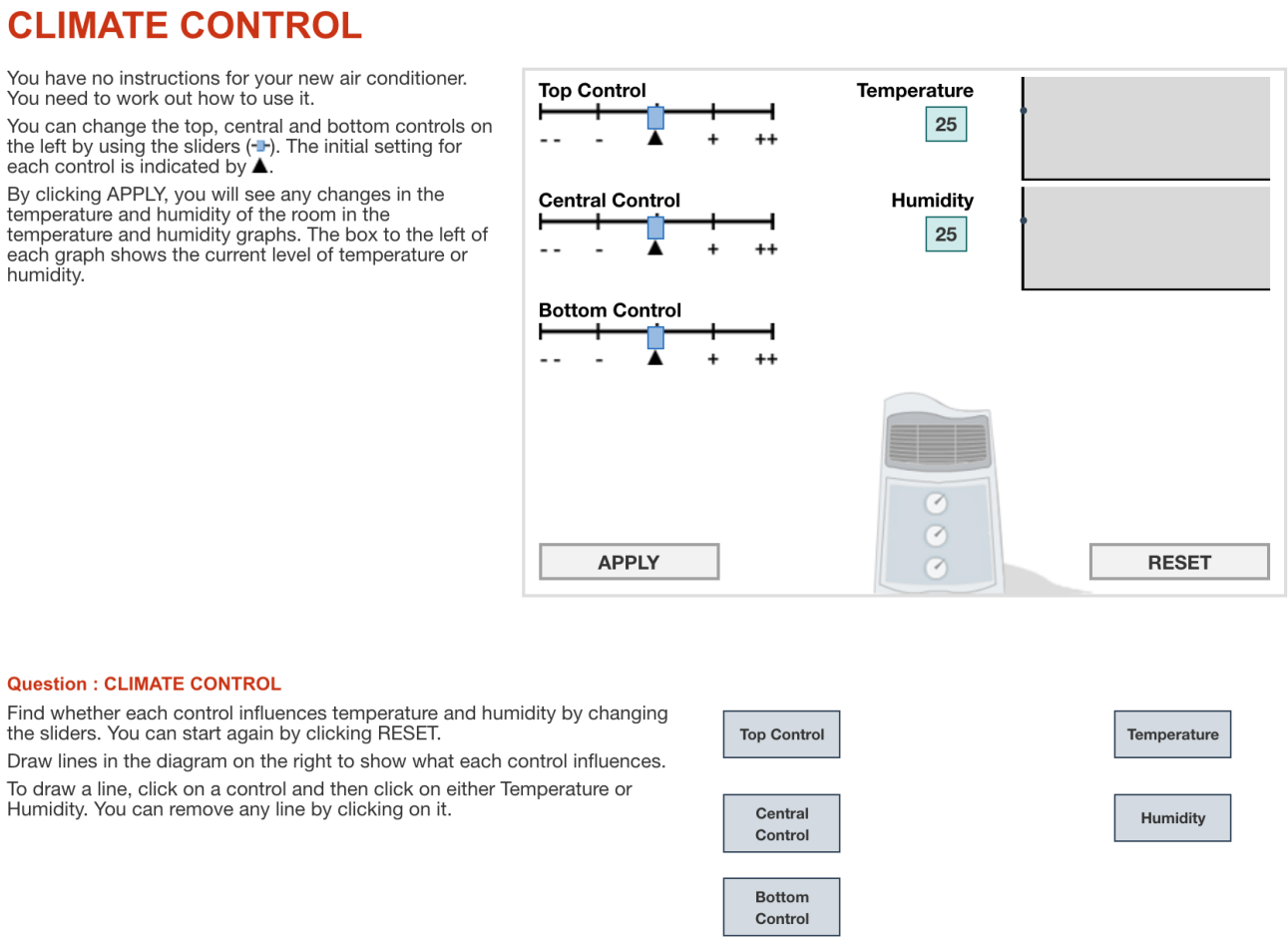
Data Input/Output

- Two file styles: single or multiple
- `read.seqs`: read process data from a CSV file as a `proc` object
- `write.seqs`: write a `proc` object to a CSV file.

Code demonstration: chuck 1 in `procdata_demo.R`

Climate Control Dataset

`ProcData` contains a dataset of PISA climate control item.



Climate Control Dataset

ProcData contains a dataset of PISA climate control item.

- Item interface: `https://www.oecd.org/pisa/test-2012/testquestions/question3/`

Climate Control Dataset

ProcData contains a dataset of PISA climate control item.

- Item interface: `https://www.oecd.org/pisa/test-2012/testquestions/question3/`
- Load dataset in R

```
data(cc_data)
```

Climate Control Dataset

ProcData contains a dataset of PISA climate control item.

- Item interface: `https://www.oecd.org/pisa/test-2012/testquestions/question3/`
- Load dataset in R

  ```
  data(cc_data)
  ```

- `cc_data` is a list of two elements
  - `seqs`: a `proc` object of 16,763 processes
  - `responses`: binary item responses

Code demonstration: chunk 2 in `procdata_demo.R`

Data Structure

Response process:

Action:	Start,	Click_CS,	...,	Next,	Next_OK
Time:	0.0,	2.9,	...,	60.4,	62.2

- S3 class `proc` is defined in ProcData for response processes.

Data Structure

Response process:

Action:	Start,	Click_CS,	...,	Next,	Next_OK
Time:	0.0,	2.9,	...,	60.4,	62.2

- S3 class `proc` is defined in ProcData for response processes.
- A `proc` object of n processes is a list of two elements.
  - `action_seqs`: a list of n action sequences (character vectors).
  - `time_seqs`: a list of n time sequences (numeric vectors).
- Names of `action_seqs` and `time_seqs` (if not `NULL`) are IDs of respondents.

Data Structure – Methods for `proc`

- `print.proc` prints response processes in an easy-to-read format

Data Structure – Methods for `proc`

- `print.proc` prints response processes in an easy-to-read format
- `summary.proc` gives a list of quantities summarized from a set of response processes.
  - number of processes
  - number of distinct actions
  - action set
  - total response time
  - ...

Data Structure – Methods for `proc`

- `print.proc` prints response processes in an easy-to-read format
- `summary.proc` gives a list of quantities summarized from a set of response processes.
  - number of processes
  - number of distinct actions
  - action set
  - total response time
  - ...

Code demonstration: chuck 3 in `procdata_demo.R`

Data Processing

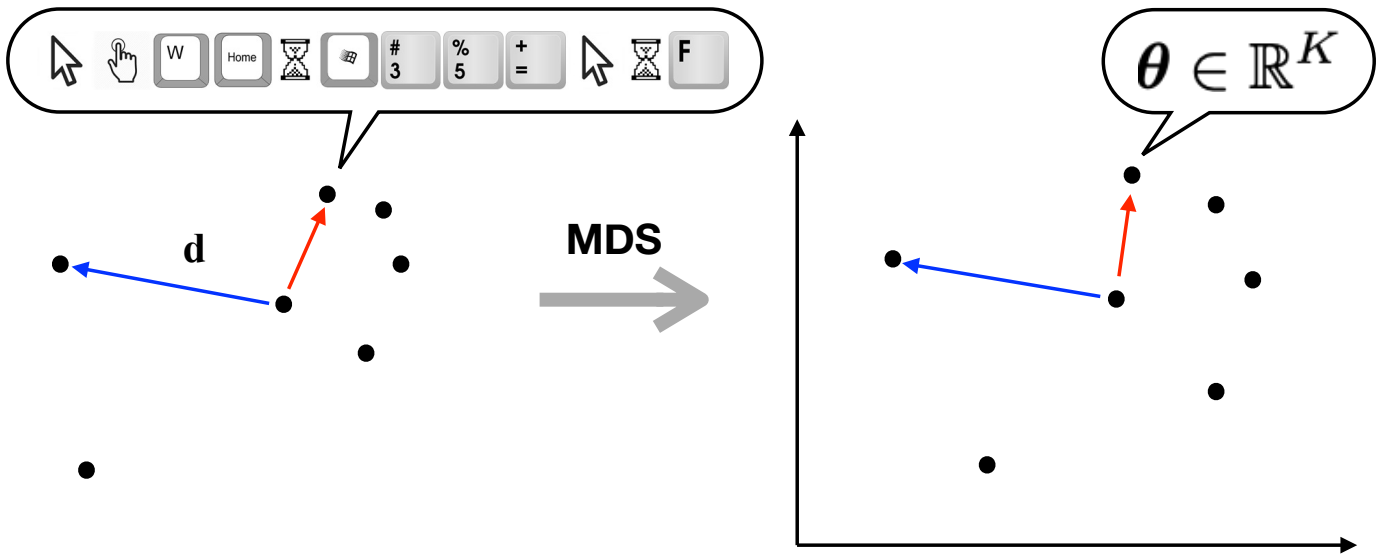
- `sub_seqs`: subset `proc` objects
- `remove_repeat`: remove consecutive repeated actions
- `remove_action`: remove a set of actions
- `replace_action`: rename an action
- `combine_actions`: combine several consecutive actions into a single action

Code demonstration: chunk 4 in `procdata_demo.R`

Feature Extraction

- Goal: transform a response process $\boldsymbol{s}$ into $\boldsymbol{\theta} \in \mathbb{R}^K$ without losing too much information
- Two methods:
  - Multidimensional Scaling
  - Sequence-to-sequence autoencoder

Feature Extraction – Multidimensional Scaling



Feature Extraction – Multidimensional Scaling

- $d_{ij} = d(\boldsymbol{s}_i, \boldsymbol{s}_j)$: dissimilarity between action sequences $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$.

Feature Extraction – Multidimensional Scaling

- $d_{ij} = d(\boldsymbol{s}_i, \boldsymbol{s}_j)$: dissimilarity between action sequences $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$.
- Minimize

$$F(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n) = \sum_{i<j} (d_{ij} - \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|)^2$$

  - $\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_j\|$: distance between latent features $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$.

Feature Extraction – Multidimensional Scaling

```
seq2feature_mds(seqs = NULL, K = 2, method = "auto",
        dist_type = "oss_action", pca = TRUE,
        return_dist = FALSE, ...)
```

- `seq2feature_mds` extracts MDS features from response processes.

Feature Extraction – Multidimensional Scaling

```
seq2feature_mds(seqs = NULL, K = 2, method = "auto",
        dist_type = "oss_action", pca = TRUE,
        return_dist = FALSE, ...)
```

- `seq2feature_mds` extracts MDS features from response processes.
- `seqs` can be a `proc` object or a dissimilarity matrix.

Feature Extraction – Multidimensional Scaling

```
seq2feature_mds(seqs = NULL, K = 2, method = "auto",
        dist_type = "oss_action", pca = TRUE,
        return_dist = FALSE, ...)
```

- `seq2feature_mds` extracts MDS features from response processes.
- `seqs` can be a `proc` object or a dissimilarity matrix.
- `K` specifies feature dimension.
  - use `chooseK_mds` to select the dimension by cross-validation.

Feature Extraction – Multidimensional Scaling

```
seq2feature_mds(seqs = NULL, K = 2, method = "auto",
        dist_type = "oss_action", pca = TRUE,
        return_dist = FALSE, ...)
```

- `seq2feature_mds` extracts MDS features from response processes.
- `seqs` can be a `proc` object or a dissimilarity matrix.
- `K` specifies feature dimension.
  - use `chooseK_mds` to select the dimension by cross-validation.
- If `pca=TRUE`, principal components of features are returned.

Feature Extraction – Multidimensional Scaling

```
seq2feature_mds(seqs = NULL, K = 2, method = "auto",
        dist_type = "oss_action", pca = TRUE,
        return_dist = FALSE, ...)
```

- `seq2feature_mds` extracts MDS features from response processes.
- `seqs` can be a `proc` object or a dissimilarity matrix.
- `K` specifies feature dimension.
  - use `chooseK_mds` to select the dimension by cross-validation.
- If `pca=TRUE`, principal components of features are returned.
- If `return_dist=TRUE`, the full dissimilarity matrix is returned.

Feature Extraction – Multidimensional Scaling

```
seq2feature_mds(seqs = NULL, K = 2, method = "auto",
        dist_type = "oss_action", pca = TRUE,
        return_dist = FALSE, ...)
```

- Dissimilarity measure: Order-based Symbol Similarity (OSS; Gómez-Alonso and Valls, 2008)
- Customized dissimilarity measures can be used by computing the dissimilarity matrix outside `seq2feature_mds` and then passing the matrix to the function through `seqs`.

Feature Extraction – Multidimensional Scaling

```
seq2feature_mds(seqs = NULL, K = 2, method = "auto",
        dist_type = "oss_action", pca = TRUE,
        return_dist = FALSE, ...)
```

- Optimization method is specified via `method`.

Feature Extraction – Multidimensional Scaling

```
seq2feature_mds(seqs = NULL, K = 2, method = "auto",
        dist_type = "oss_action", pca = TRUE,
        return_dist = FALSE, ...)
```

- Optimization method is specified via `method`.
- `method="small"`: classical MDS via `cmdscale`.
  - Requires large memory if sample size is large.

Feature Extraction – Multidimensional Scaling

```
seq2feature_mds(seqs = NULL, K = 2, method = "auto",
        dist_type = "oss_action", pca = TRUE,
        return_dist = FALSE, ...)
```

- Optimization method is specified via `method`.
- `method="small"`: classical MDS via `cmdscale`.
  - Requires large memory if sample size is large.
- `method="large"`: an algorithm for large datasets (Paradis 2018)

Feature Extraction – Multidimensional Scaling

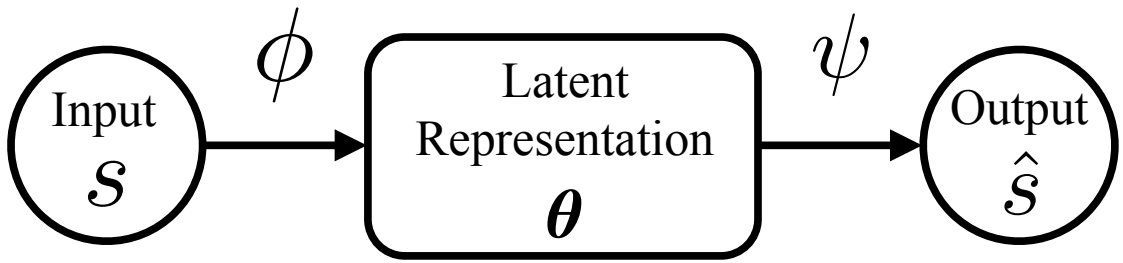
```
seq2feature_mds(seqs = NULL, K = 2, method = "auto",
        dist_type = "oss_action", pca = TRUE,
        return_dist = FALSE, ...)
```

- Optimization method is specified via `method`.
- `method="small"`: classical MDS via `cmdscale`.
  - Requires large memory if sample size is large.
- `method="large"`: an algorithm for large datasets (Paradis 2018)
- `method="auto"`: use `method="small"` if sample size < 5000 and `method="large"` otherwise.

Code demonstration: chunk 5 in `procdata_demo.R`

Feature Extraction – Seq2Seq Autoencoder

Autoencoder: neural network that reproduces input in its output

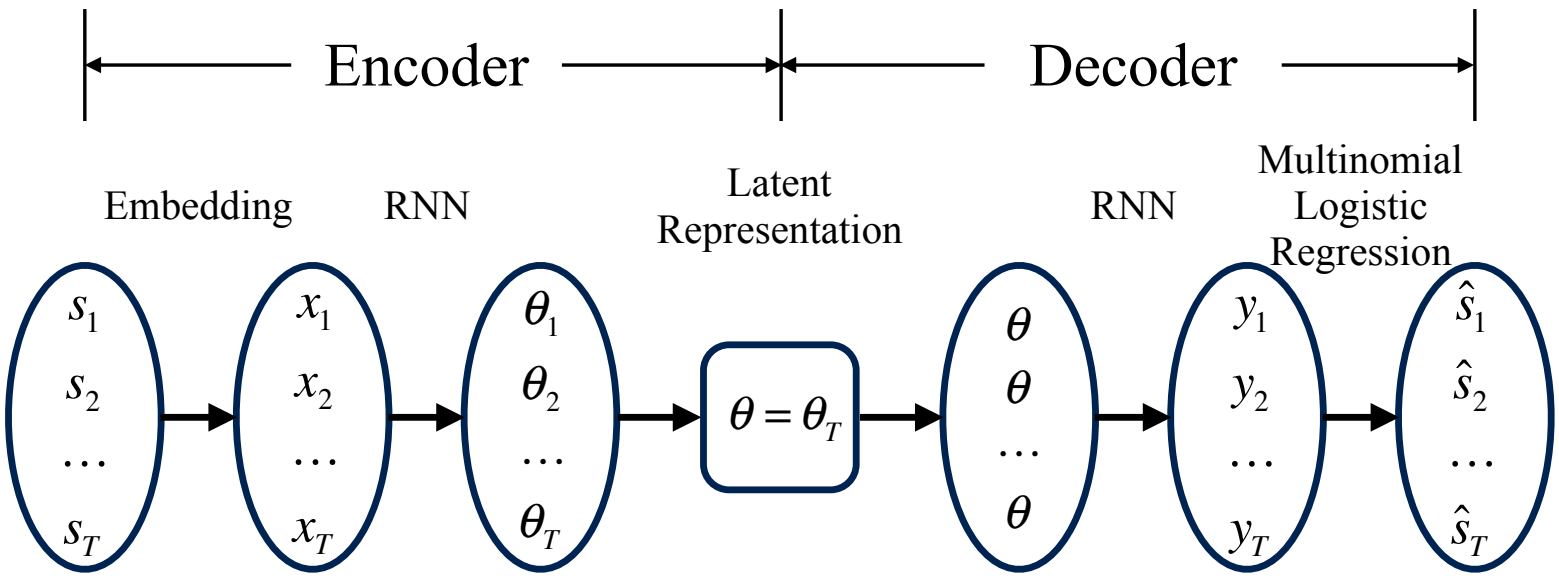


- $\phi(\cdot;\eta)$: encoder; $\psi(\cdot;\xi)$: decoder; features: $\boldsymbol{\theta} = \phi(\boldsymbol{s};\eta)$
- Parameters are estimated by minimizing the average difference between $\hat{\boldsymbol{s}}$ and $\boldsymbol{s}$

$$F(\eta,\xi) = \frac{1}{n}\sum_{i=1}^{n} L(\boldsymbol{s}_i, \hat{\boldsymbol{s}}_i)$$

Feature Extraction – Seq2Seq Autoencoder

Action sequence autoencoder



Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type = "action",
    rnn_type = "lstm", pca = TRUE, n_epoch = 50,
    step_size = 1e-04, optimizer_name = "adam",
    samples_train, samples_valid, ...)
```

Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type = "action",
    rnn_type = "lstm", pca = TRUE, n_epoch = 50,
    step_size = 1e-04, optimizer_name = "adam",
    samples_train, samples_valid, ...)
```

- `seq2feature_seq2seq` extracts `K` features from response processes given in the `proc` object `seqs`.
  - `K` can be selected by `chooseK_seq2seq`.

Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type = "action",
    rnn_type = "lstm", pca = TRUE, n_epoch = 50,
    step_size = 1e-04, optimizer_name = "adam",
    samples_train, samples_valid, ...)
```

- `seq2feature_seq2seq` extracts `K` features from response processes given in the `proc` object `seqs`.
  - `K` can be selected by `chooseK_seq2seq`.
- `rnn_type` specifies the structure of the encoder RNN and the decoder RNN (`"lstm"` or `"gru"`).

Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type = "action",
    rnn_type = "lstm", pca = TRUE, n_epoch = 50,
    step_size = 1e-04, optimizer_name = "adam",
    samples_train, samples_valid, ...)
```

- `seq2feature_seq2seq` extracts `K` features from response processes given in the `proc` object `seqs`.
  - `K` can be selected by `chooseK_seq2seq`.
- `rnn_type` specifies the structure of the encoder RNN and the decoder RNN (`"lstm"` or `"gru"`).
- If `pca=TRUE`, principal components of extracted features are returned.

Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type = "action",
    rnn_type = "lstm", pca = TRUE, n_epoch = 50,
    step_size = 1e-04, optimizer_name = "adam",
    samples_train, samples_valid, ...)
```

- The minimizer of the objective function $F(\eta, \xi)$ is obtained by stochastic approximation.

Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type = "action",
    rnn_type = "lstm", pca = TRUE, n_epoch = 50,
    step_size = 1e-04, optimizer_name = "adam",
    samples_train, samples_valid, ...)
```

- The minimizer of the objective function $F(\eta, \xi)$ is obtained by stochastic approximation.
- `n_epoch` specifies the number of epochs to run.

Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type = "action",
    rnn_type = "lstm", pca = TRUE, n_epoch = 50,
    step_size = 1e-04, optimizer_name = "adam",
    samples_train, samples_valid, ...)
```

- The minimizer of the objective function $F(\eta, \xi)$ is obtained by stochastic approximation.
- `n_epoch` specifies the number of epochs to run.
- `step_size` specifies the (baseline) step size.

Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type = "action",
    rnn_type = "lstm", pca = TRUE, n_epoch = 50,
    step_size = 1e-04, optimizer_name = "adam",
    samples_train, samples_valid, ...)
```

- The minimizer of the objective function $F(\eta, \xi)$ is obtained by stochastic approximation.
- `n_epoch` specifies the number of epochs to run.
- `step_size` specifies the (baseline) step size.
- `optimizer_name` specifies the algorithms.
  - Available options are `"sgd"`, `"rmsprop"`, `"adadelta"`, and `"adam"`.

Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type = "action",
    rnn_type = "lstm", pca = TRUE, n_epoch = 50,
    step_size = 1e-04, optimizer_name = "adam",
    samples_train, samples_valid, ...)
```

- Validation-based early stopping is used to prevent overfitting.

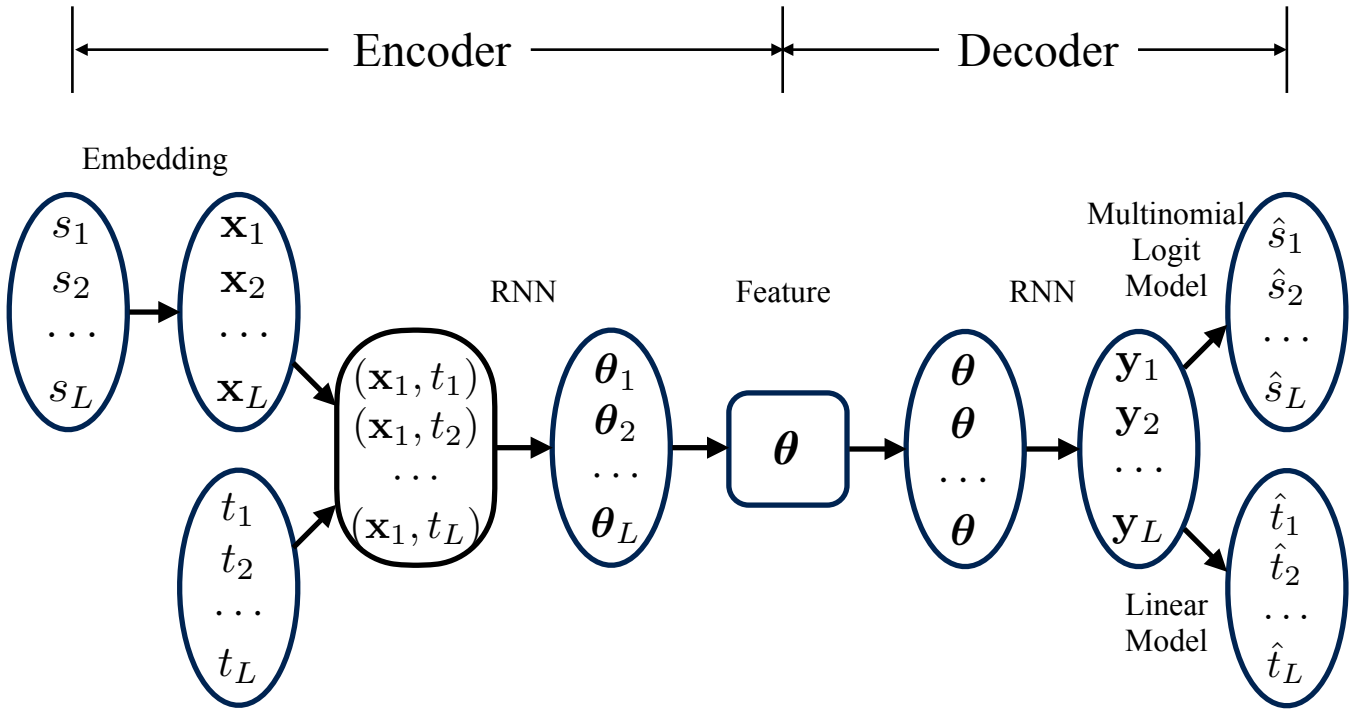
Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type = "action",
    rnn_type = "lstm", pca = TRUE, n_epoch = 50,
    step_size = 1e-04, optimizer_name = "adam",
    samples_train, samples_valid, ...)
```

- Validation-based early stopping is used to prevent overfitting.
- `samples_train` and `samples_valid` specifies the indices of observations in the training and validation set.

Feature Extraction – Seq2Seq Autoencoder

Action-time sequence autoencoder



Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type, rnn_type,
      n_epoch, step_size, optimizer_name,
      cumulative=FALSE, log=TRUE, weights=c(1,0.5),
      samples_train, samples_valid, ...)
```

Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type, rnn_type,
      n_epoch, step_size, optimizer_name,
      cumulative=FALSE, log=TRUE, weights=c(1,0.5),
      samples_train, samples_valid, ...)
```

- `ae_type="both"`

Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type, rnn_type,
      n_epoch, step_size, optimizer_name,
      cumulative=FALSE, log=TRUE, weights=c(1,0.5),
      samples_train, samples_valid, ...)
```

- `ae_type="both"`
- If `cumulative=FALSE`, time between two consecutive actions will be used instead of the timestamp sequence.

Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type, rnn_type,
      n_epoch, step_size, optimizer_name,
      cumulative=FALSE, log=TRUE, weights=c(1,0.5),
      samples_train, samples_valid, ...)
```

- `ae_type="both"`
- If `cumulative=FALSE`, time between two consecutive actions will be used instead of the timestamp sequence.
- If `log=TRUE`, the logarithm of the time sequence will be used.

Feature Extraction – Seq2Seq Autoencoder

```
seq2feature_seq2seq(seqs, K, ae_type, rnn_type,
      n_epoch, step_size, optimizer_name,
      cumulative=FALSE, log=TRUE, weights=c(1,0.5),
      samples_train, samples_valid, ...)
```

- `ae_type="both"`
- If `cumulative=FALSE`, time between two consecutive actions will be used instead of the timestamp sequence.
- If `log=TRUE`, the logarithm of the time sequence will be used.
- `weights` specifies the weights of action sequence difference and time sequence difference in the loss function.

Code demonstration: chunk 6 in `procdata_demo.R`

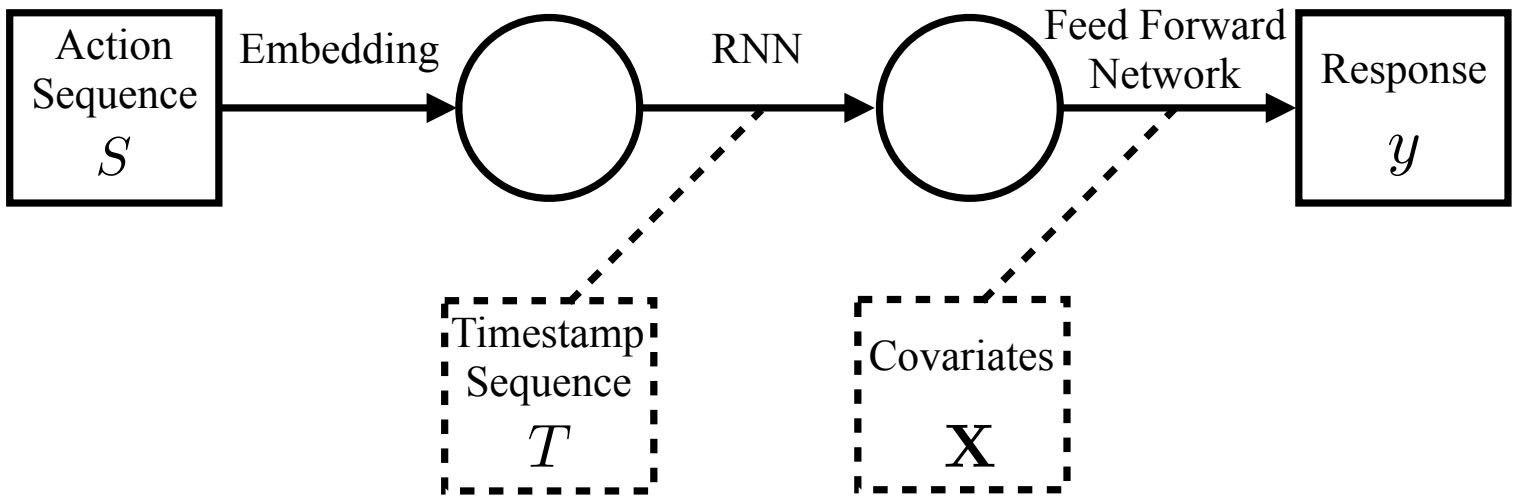
Sequence Model

Processes	Covariates	Response variable
	31	✓
	24	✗
	45	✓
	37	✗
	29	?

Sequence models can be used for

- predicting response variables based on processes and covariates
- extracting features related to the response variable

Sequence Model



Sequence Model

```
seqm(seqs, response, covariates, response_type,
     actions, max_len, rnn_type, include_time,
     time_interval, log_time, K_emb, K_rnn,
     n_hidden, K_hidden, index_valid, n_epoch,
     optimizer_name, step_size, ...)
```

Sequence Model

```
seqm(seqs, response, covariates, response_type,
     actions, max_len, rnn_type, include_time,
     time_interval, log_time, K_emb, K_rnn,
     n_hidden, K_hidden, index_valid, n_epoch,
     optimizer_name, step_size, ...)
```

- `seqs`: response processes
- `response`: response variable
- `covariates`: covariates
- `response_type`: type of response variable `"binary"` or `"scale"`.

Sequence Model

```
seqm(seqs, response, covariates, response_type,
     actions, max_len, rnn_type, include_time,
     time_interval, log_time, K_emb, K_rnn,
     n_hidden, K_hidden, index_valid, n_epoch,
     optimizer_name, step_size, ...)
```

- `actions`: action set; it contains all actions appearing in `seqs` by default.
- `max_len`: maximum process length; it is the length of the longest process in `seqs` by default.
- `rnn_type`: structure of RNN: `"lstm"` or `"gru"`.

Sequence Model

```
seqm(seqs, response, covariates, response_type,
     actions, max_len, rnn_type, include_time,
     time_interval, log_time, K_emb, K_rnn,
     n_hidden, K_hidden, index_valid, n_epoch,
     optimizer_name, step_size, ...)
```

- `include_time`: if TRUE, time sequence is included in the model.
- `time_interval`: if TRUE, time between two consecutive actions is used.
- `log_time`: if TRUE, logarithm of time sequences is used in the model.

Sequence Model

```
seqm(seqs, response, covariates, response_type,
     actions, max_len, rnn_type, include_time,
     time_interval, log_time, K_emb, K_rnn,
     n_hidden, K_hidden, index_valid, n_epoch,
     optimizer_name, step_size, ...)
```

- `K_emb`: embedding dimension
- `K_rnn`: dimension of the output of RNN
- `n_hidden`: the number of layers in the feed forward neural network.
- `K_hidden`: a numeric vector specifying the dimension of each layer in the feed forward network.

Sequence Model

```
seqm(seqs, response, covariates, response_type,
     actions, max_len, rnn_type, include_time,
     time_interval, log_time, K_emb, K_rnn,
     n_hidden, K_hidden, index_valid, n_epoch,
     optimizer_name, step_size, ...)
```

- `index_valid`: indices of processes or the proportion of processes in the validation set.
- `n_epoch`: the number of epochs to be run when optimizing the objective function.
- `optimizer_name`: name of the optimizer
- `step_size`: (baseline) step size of updates during optimization

Sequence Model

`seqm` returns an object of class `seqm`. It is a list containing

- human-readable model structure
- the fitted model to be used by `predict.seqm`
- trace of values of objective function in the training process.

Sequence Model

Predictions and features can be obtained by the `predict` method for sequence models

```
predict(object, new_seqs, new_covariates = NULL,
        type = "response", ...)
```

- `object`: an object of class `seqm`
- `new_seqs`: an object of class `"proc"` with which to predict.
- `new_covariates`: a covariate matrix with which to predict.
- `type`: a string specifying whether to predict responses (`"response"`) or to extract features (`"feature"`) or both (`"both"`).

Code demonstration: chunk 7 in `procdata_demo.R`

Report bugs at
`https://github.com/xytangtang/ProcData/issues`.

Thank you!

xytang@math.arizona.edu