Modal Symbolic Learning: Day 3

Interpretable gesture recognition on NATOPS

Logisets

In [1]:

```julia
using Pkg
Pkg.activate(".")
Pkg.instantiate()
Pkg.update()
Pkg.status()
```

```
  Activating project at `~/Desktop/modal-symbolic-learning-course`
    Updating registry at `~/.julia/registries/General`
    Updating git-repo `https://github.com/JuliaRegistries/General.git`
  No Changes to `~/Desktop/modal-symbolic-learning-course/Project.toml`
  No Changes to `~/Desktop/modal-symbolic-learning-course/Manifest.toml`
Status `~/Desktop/modal-symbolic-learning-course/Project.toml`
  [acdeb78f] Catch22 v0.4.5
  [a93c6f00] DataFrames v1.6.1
  [864edb3b] DataStructures v0.18.15
  [7806a523] DecisionTree v0.12.4
  [7073ff75] IJulia v1.24.2
  [6a3955dd] ImageFiltering v0.7.8
  [033835bb] JLD2 v0.4.38
  [23992714] MAT v0.10.6
⌅ [add582a8] MLJ v0.19.5
  [c6f25543] MLJDecisionTreeInterface v0.4.0
  [e54bda2e] ModalDecisionTrees v0.3.6
  [91a5bcdd] Plots v1.39.0
  [7b3b3b3f] Sole v0.3.1
  [b002da8f] SoleLogics v0.6.14
  [4249d9c7] SoleModels v0.5.6
  [2913bbd2] StatsBase v0.34.2
  [9a3f8284] Random
Info Packages marked with ⌅ have new versions available but compatibility co
nstraints restrict them from upgrading. To see why use `status --outdated`
```

In [2]:

```julia
# Import libraries for statistics & Machine Learning
using Random
using DataFrames
using MLJ
using Plots
using StatsBase
```

In [3]:

```julia
# Import the Sole framework
using Sole
```

```
# Load an example time-series classification dataset as a tuple (DataFrame,
X_df, y = Sole.load_arff_dataset("NATOPS");
```

In [4]:

```
countmap(y)
```

Out[4]:

```
Dict{CategoricalArrays.CategoricalValue{String, UInt32}, Int64} with 6 entr
ies:
  "Spread wings"   => 60
  "I have command" => 60
  "Not clear"      => 60
  "Lock wings"     => 60
  "All clear"      => 60
  "Fold wings"     => 60
```

In [5]:

```
X_df
```

Out[5]: 360×24 DataFrame *335 rows omitted*

Row	X[Hand tip l]	Y[Hand tip l]	Z[Hand tip l]	X[Hand tip r]	Y[Hand tip r]	Z[Hand tip r]	X[
	Array…	Array…	Array…	Array…	Array…	Array…	Ar
1	[-0.519771, -0.52758, -0.531415, -0.517159, -0.510312, -0.518154, -0.50362, -0.485176, -0.466677, -0.444535 … -0.45501, -0.458937, -0.465048, -0.471251, -0.470015, -0.464627, -0.462666, -0.460253, -0.459572, -0.456737]	[-2.14011, -2.18043, -2.18425, -2.16547, -2.16635, -2.18836, -2.17162, -2.15248, -2.08072, -2.00607 … -2.17597, -2.1638, -2.17779, -2.17766, -2.17848, -2.16689, -2.15667, -2.13474, -2.13435, -2.13855]	[-0.957224, -0.970778, -0.970232, -0.960666, -0.962437, -0.970488, -0.966847, -0.96441, -0.972943, -0.979085 … -1.04234, -1.03616, -1.03756, -1.03275, -1.02525, -1.03115, -1.02558, -1.01884, -1.01701, -1.01059]	[0.675893, 0.699281, 0.673774, 0.700096, 0.765257, 0.980454, 1.43803, 1.78334, 2.08495, 2.32037 … 0.755717, 0.778103, 0.755128, 0.751274, 0.742517, 0.743311, 0.786792, 0.730863, 0.730482, 0.732217]	[-2.31794, -2.36398, -2.48698, -2.3176, -2.34228, -2.34828, -2.24596, -1.8102, -1.28214, -0.703666 … -2.45044, -2.33026, -2.44767, -2.43509, -2.44371, -2.42475, -2.25219, -2.38539, -2.38603, -2.35704]	[-0.254602, -0.246883, -0.252635, -0.235782, -0.13363, 0.051243, 0.078424, 0.274688, 0.335957, 0.390646 … -0.210761, -0.181256, -0.213764, -0.206785, -0.222643, -0.214863, -0.169845, -0.20958, -0.202703, -0.201438]	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
2	[-0.489753, -0.48607, -0.484529, -0.492771, -0.492031, -0.493076, -0.491979, -0.493256, -0.493156, -0.487527 … -0.400825, -0.414617, -0.407231, -0.397206, -0.366296, -0.354333, -0.371938, -0.386065, -0.408146, -0.415736]	[-1.55293, -1.54966, -1.55206, -1.55821, -1.556, -1.56055, -1.55812, -1.5648, -1.56414, -1.56731 … -1.6062, -1.62319, -1.61939, -1.6173, -1.58341, -1.5697, -1.55188, -1.54089, -1.52865, -1.52388]	[-0.907814, -0.911305, -0.92587, -0.921268, -0.928352, -0.928697, -0.932141, -0.930564, -0.933592, -0.932622 … -0.989828, -0.990365, -0.998319, -0.994962, -0.994991, -0.983351, -0.976952, -0.975923, -0.963954, -0.953944]	[0.632831, 0.633167, 0.637368, 0.640823, 0.635858, 0.63401, 0.634496, 0.637154, 0.640618, 0.643018 … 0.558287, 0.447356, 0.452128, 0.525122, 0.651756, 0.77637, 0.948441, 1.09432, 1.30458, 1.42438]	[-1.61526, -1.61763, -1.62374, -1.61861, -1.62068, -1.62244, -1.62164, -1.6257, -1.62654, -1.62966 … 1.56275, 1.58349, 1.59581, 1.60302, 1.55387, 1.53016, 1.47453, 1.47069, 1.45205, 1.39396]	[-0.63772, -0.637168, -0.644338, -0.651686, -0.653233, -0.654332, -0.651011, -0.6489, -0.654768, -0.653883 … 0.526364, 0.534895, 0.553634, 0.564454, 0.478762, 0.47897, 0.444671, 0.328608, 0.29968, 0.242647]	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
3	[-0.521346, -0.518394, -0.522321, -0.519893, -0.521016, -0.521524, -0.523362, -0.511653, -0.512519, -0.511312 … -0.514448, -0.518708,	[-1.72326, -1.72407, -1.72326, -1.72352, -1.72479, -1.72389, -1.7244, -1.76782, -1.76903, -1.76877 … -1.79175, -1.77926,	[-0.581362, -0.578159, -0.586091, -0.582611, -0.583196, -0.582819, -0.580284, -0.57613, -0.576047, -0.575067 … -0.64696, -0.640021,	[0.480245, 0.413413, 0.425131, 0.420865, 0.481781, 0.483458, 0.415258, 0.429159, 0.449354, 0.476563 … 0.71045, 0.665733,	[-1.72509, -1.79325, -1.77693, -1.78382, -1.72083, -1.72458, -1.80616, -1.77722, -1.78057, -1.79041 … -1.57885, -1.64564,	[-0.749465, -0.814978, -0.79228, -0.801608, -0.754548, -0.74575, -0.806902, -0.788115, -0.775095, -0.768625 … -1.16744, -0.986366,	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0

Row	X[Hand tip l] Array…	Y[Hand tip l] Array…	Z[Hand tip l] Array…	X[Hand tip r] Array…	Y[Hand tip r] Array…	Z[Hand tip r] Array…	X[Ar
	-0.521672, -0.525064, -0.514835, -0.507935, -0.521132, -0.517193, -0.517363, -0.51327]	-1.77711, -1.77828, -1.77587, -1.76419, -1.77843, -1.77882, -1.77992, -1.77987]	-0.630712, -0.616814, -0.62093, -0.621969, -0.621552, -0.624645, -0.621028, -0.626299]	0.621122, 0.557295, 0.519791, 0.48524, 0.481703, 0.488414, 0.485208, 0.479489]	-1.68675, -1.81935, -1.76374, -1.76348, -1.78189, -1.7849, -1.78435, -1.78333]	-0.849024, -0.767521, -0.725116, -0.723884, -0.743611, -0.736042, -0.731239, -0.733958]	-0 -0 -0 -0 -0 -0 -0 -0
4	[-0.57022, -0.562064, -0.565967, -0.562913, -0.567557, -0.566175, -0.566748, -0.561748, -0.55966, -0.556271 … -0.530846, -0.535016, -0.537207, -0.533389, -0.530497, -0.532508, -0.522586, -0.53489, -0.534332, -0.54071]	[-1.91196, -1.90369, -1.90527, -1.90405, -1.90318, -1.90619, -1.90959, -1.89934, -1.8948, -1.89346 … -1.87427, -1.87535, -1.88059, -1.8954, -1.89976, -1.89333, -1.90898, -1.91169, -1.92236, -1.92444]	[-0.753404, -0.748702, -0.747062, -0.7541, -0.751551, -0.749891, -0.75006, -0.748899, -0.745352, -0.74102 … -0.704626, -0.713649, -0.720423, -0.721149, -0.720037, -0.727544, -0.718666, -0.731909, -0.73111, -0.727761]	[0.459493, 0.464525, 0.461903, 0.455969, 0.460419, 0.465137, 0.445696, 0.458416, 0.4603, 0.46256 … 2.09097, 1.91878, 1.58165, 1.21182, 0.941954, 0.708641, 0.537249, 0.464884, 0.459635, 0.46293]	[-1.90089, -1.87507, -1.89495, -1.89809, -1.87756, -1.87972, -1.9182, -1.88876, -1.8717, -1.86988 … -0.790038, -1.22432, -1.5668, -1.69141, -1.78663, -1.85248, -1.9406, -1.96856, -1.96701, -1.9625]	[-0.764456, -0.766048, -0.757716, -0.756718, -0.767963, -0.767328, -0.754985, -0.757794, -0.759314, -0.755372 … -0.437201, -0.560395, -0.625109, -0.675635, -0.628565, -0.654884, -0.668321, -0.737166, -0.740219, -0.737878]	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
5	[-0.624417, -0.626031, -0.625388, -0.62798, -0.624838, -0.623534, -0.626624, -0.626658, -0.622853, -0.622373 … -0.606563, -0.611505, -0.614609, -0.607108, -0.598554, -0.621197, -0.625199, -0.644386, -0.657226, -0.663721]	[-1.84287, -1.84026, -1.84688, -1.84182, -1.84628, -1.84354, -1.84273, -1.83752, -1.83289, -1.83472 … -1.68283, -1.72178, -1.77294, -1.80126, -1.81198, -1.87223, -1.89073, -1.89526, -1.9043, -1.91686]	[-0.789348, -0.786501, -0.768675, -0.779753, -0.775049, -0.77593, -0.770693, -0.771605, -0.773377, -0.76946 … -0.831481, -0.841451, -0.848442, -0.851784, -0.846179, -0.850705, -0.837824, -0.816053, -0.801157, -0.795484]	[0.58095, 0.57809, 0.579865, 0.577963, 0.576101, 0.576345, 0.575145, 0.579263, 0.579383, 0.579958 … 2.07734, 2.11504, 2.1128, 1.91689, 1.5704, 1.18571, 0.803449, 0.617248, 0.555628, 0.519571]	[-1.83512, -1.83411, -1.83304, -1.83161, -1.82641, -1.82692, -1.82371, -1.81809, -1.81299, -1.81521 … 0.210206, -0.240879, -0.761203, -1.25598, -1.64153, -1.92075, -2.01471, -1.99813, -1.98928, -2.0021]	[-0.748908, -0.753321, -0.749488, -0.758251, -0.764208, -0.764563, -0.768688, -0.772309, -0.774509, -0.774836 … 0.212435, 0.104328, -0.043032, -0.286689, -0.430668, -0.572304, -0.638792, -0.682752, -0.718812, -0.761999]	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
6	[-0.502501, -0.502525, -0.499415, -0.501144, -0.502677, -0.501937,	[-2.17556, -2.15613, -2.18516, -2.19291, -2.15844, -2.14539,	[-1.09413, -1.07683, -1.09008, -1.09044, -1.07624, -1.06987,	[0.631689, 0.624567, 0.638725, 0.640064, 0.617619, 0.609287,	[-2.39645, -2.35991, -2.39196, -2.35874, -2.39011, -2.38913,	[-0.174365, -0.166227, -0.164783, -0.171156, -0.171868, -0.168856,	[-(-0 -0 -0 -0 -0

Row	X[Hand tip l] Array…	Y[Hand tip l] Array…	Z[Hand tip l] Array…	X[Hand tip r] Array…	Y[Hand tip r] Array…	Z[Hand tip r] Array…	X[Ar
	-0.500699, -0.501717, -0.501963, -0.504734 … -0.43365, -0.436541, -0.447761, -0.456823, -0.460775, -0.467277, -0.464943, -0.469757, -0.468361, -0.469486]	-2.17194, -2.14695, -2.13512, -2.12584 … -2.05511, -2.07088, -2.07849, -2.09125, -2.10468, -2.12407, -2.13097, -2.16563, -2.1762, -2.19243]	-1.07743, -1.07267, -1.0659, -1.0586 … -0.994042, -1.00038, -1.00443, -1.00719, -1.01067, -1.02072, -1.02934, -1.0372, -1.04505, -1.04952]	0.623102, 0.614398, 0.60629, 0.591307 … 1.1434, 0.936468, 0.81356, 0.748232, 0.70633, 0.715475, 0.701832, 0.708491, 0.711467, 0.724143]	-2.4044, -2.3628, -2.33339, -2.39623 … -2.29177, -2.33778, -2.32335, -2.31428, -2.39301, -2.30433, -2.44085, -2.4951, -2.50935, -2.53032]	-0.167825, -0.166135, -0.168217, -0.164987 … 0.248703, 0.315796, 0.30116, 0.181476, 0.109893, 0.069707, 0.072417, 0.054247, 0.056288, 0.068829]	-0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
7	[-0.488461, -0.489463, -0.487539, -0.495673, -0.498767, -0.492156, -0.492845, -0.484968, -0.482085, -0.480355 … -0.493495, -0.492836, -0.49895, -0.503093, -0.509026, -0.513016, -0.515636, -0.523701, -0.519121, -0.512226]	[-2.17242, -2.18203, -2.18057, -2.18011, -2.16312, -2.16706, -2.1655, -2.16417, -2.16289, -2.16507 … -2.20638, -2.24703, -2.26315, -2.27173, -2.2962, -2.3206, -2.28483, -2.26507, -2.30022, -2.30896]	[-0.968068, -0.970886, -0.972168, -0.964309, -0.968031, -0.964959, -0.965357, -0.96689, -0.961591, -0.971308 … -0.946976, -0.954752, -0.957002, -0.954402, -0.956824, -0.959061, -0.951237, -0.939096, -0.95456, -0.970996]	[0.56396, 0.595508, 0.563289, 0.562872, 0.569912, 0.59887, 0.597455, 0.59935, 0.588259, 0.616466 … 1.90101, 1.68372, 1.45597, 1.22308, 1.03946, 0.896663, 0.789375, 0.750815, 0.748015, 0.753018]	[-2.39541, -2.32961, -2.40599, -2.4037, -2.38496, -2.30025, -2.29899, -2.29647, -2.34951, -2.34849 … -1.68622, -2.01573, -2.27071, -2.44548, -2.54653, -2.61704, -2.58102, -2.42942, -2.45797, -2.3098]	[-0.189166, -0.156892, -0.183036, -0.188968, -0.182562, -0.155315, -0.159691, -0.162753, -0.167263, -0.126656 … 0.428323, 0.353664, 0.283833, 0.227649, 0.192813, 0.144374, 0.100455, 0.056708, 0.020836, -0.010608]	[-0 -0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
8	[-0.468105, -0.410602, -0.473909, -0.475146, -0.465564, -0.459415, -0.408703, -0.407192, -0.406746, -0.471503 … -0.403425, -0.38908, -0.388014, -0.376936, -0.386189, -0.383457, -0.379303, -0.379167, -0.35105, -0.37192]	[-1.86535, -1.89011, -1.87105, -1.87014, -1.86305, -1.86513, -1.88585, -1.88192, -1.88197, -1.86255 … -1.70318, -1.71049, -1.71516, -1.72376, -1.72854, -1.75506, -1.78602, -1.83142, -1.82577, -1.85042]	[-0.697004, -0.708269, -0.681783, -0.685562, -0.700491, -0.698824, -0.712602, -0.714313, -0.709702, -0.683408 … -0.694703, -0.696871, -0.685235, -0.698665, -0.689877, -0.695257, -0.711887, -0.716584, -0.716449, -0.711233]	[0.51303, 0.535447, 0.526609, 0.529012, 0.516169, 0.514988, 0.515586, 0.504798, 0.504606, 0.531836 … 1.64002, 1.79941, 1.99117, 2.07635, 2.11726, 2.06097, 1.95247, 1.74608, 1.46703, 1.22156]	[-1.89671, -1.86846, -1.87776, -1.86998, -1.90863, -1.90658, -1.89158, -1.91002, -1.9078, -1.88441 … 0.980896, 0.775941, 0.411292, 0.105297, -0.240116, -0.606455, -0.99533, -1.35875, -1.59084, -1.78522]	[-0.72422, -0.706672, -0.716476, -0.716994, -0.720904, -0.726877, -0.722421, -0.730964, -0.734788, -0.696653 … -0.583728, -0.670627, -0.697702, -0.731501, -0.745341, -0.771108, -0.766031, -0.755091, -0.675288, -0.632553]	[-0 -0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0

Row	X[Hand tip l]	Y[Hand tip l]	Z[Hand tip l]	X[Hand tip r]	Y[Hand tip r]	Z[Hand tip r]	X
	Array…	Array…	Array…	Array…	Array…	Array…	Ar
9	[-0.568195, -0.572936, -0.571337, -0.577742, -0.562071, -0.563401, -0.56426, -0.56251, -0.567891, -0.568983 … -0.572375, -0.573012, -0.570897, -0.571727, -0.572124, -0.56665, -0.568694, -0.576087, -0.577822, -0.576184]	[-1.79059, -1.78162, -1.78303, -1.77291, -1.79029, -1.79301, -1.7906, -1.79415, -1.78852, -1.78192 … -1.71373, -1.712, -1.71488, -1.71719, -1.724, -1.73081, -1.73278, -1.74117, -1.74922, -1.7542]	[-0.629271, -0.631997, -0.629313, -0.636701, -0.635508, -0.634023, -0.639724, -0.637628, -0.638034, -0.642679 … -0.59198, -0.595464, -0.590844, -0.597421, -0.599602, -0.605654, -0.595905, -0.590594, -0.592675, -0.593041]	[0.574572, 0.572953, 0.58065, 0.576566, 0.576827, 0.576204, 0.578404, 0.575839, 0.571202, 0.578351 … 1.11163, 1.28867, 1.47037, 1.62137, 1.73751, 1.76605, 1.68489, 1.51916, 1.30435, 1.10295]	[-1.82092, -1.82256, -1.8185, -1.81975, -1.81878, -1.81891, -1.81806, -1.82086, -1.82313, -1.8185 … 1.05829, 0.920416, 0.710814, 0.407749, 0.04714, -0.35929, -0.776797, -1.13484, -1.41145, -1.55159]	[-0.617017, -0.615382, -0.612141, -0.614752, -0.611139, -0.616751, -0.615291, -0.615829, -0.618155, -0.617481 … -0.349956, -0.491946, -0.653769, -0.814871, -0.892301, -0.952159, -0.969561, -0.944127, -0.865372, -0.795585]	[- -0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
10	[-0.517579, -0.515374, -0.517325, -0.516505, -0.514786, -0.513077, -0.518725, -0.520816, -0.519732, -0.521663 … -0.470503, -0.478416, -0.476322, -0.483407, -0.494542, -0.497023, -0.503007, -0.511892, -0.516592, -0.506665]	[-1.73887, -1.74072, -1.7397, -1.73405, -1.73453, -1.73561, -1.7393, -1.71604, -1.72597, -1.73219 … -1.72644, -1.73147, -1.73613, -1.75607, -1.76061, -1.78371, -1.80105, -1.82082, -1.83392, -1.84493]	[-0.693497, -0.691899, -0.687666, -0.692099, -0.686091, -0.6872, -0.693279, -0.690005, -0.68333, -0.693432 … -0.746989, -0.737197, -0.732215, -0.722234, -0.737401, -0.737225, -0.742772, -0.742642, -0.749105, -0.749828]	[0.514507, 0.526694, 0.549404, 0.506781, 0.550714, 0.548035, 0.542946, 0.536573, 0.54797, 0.515168 … 2.05311, 2.01937, 2.0615, 2.14439, 2.14045, 2.15822, 2.0852, 1.93771, 1.70293, 1.46727]	[-1.83433, -1.81353, -1.76559, -1.82393, -1.76616, -1.76641, -1.7819, -1.79043, -1.76682, -1.8396 … 0.671155, 0.405484, 0.170111, -0.095538, -0.391807, -0.683817, -0.992354, -1.32681, -1.56162, -1.79568]	[-0.711129, -0.706901, -0.695049, -0.725259, -0.690311, -0.68963, -0.702482, -0.705599, -0.698506, -0.715266 … -0.295706, -0.343539, -0.356921, -0.385912, -0.405628, -0.440152, -0.444337, -0.468822, -0.478764, -0.456275]	[- -0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
11	[-0.631494, -0.629032, -0.630474, -0.628314, -0.625873, -0.620084, -0.622708, -0.615488, -0.604842, -0.609029 … -0.614349, -0.623253, -0.624818, -0.61368,	[-1.98071, -1.98581, -1.98407, -1.98487, -1.98305, -1.98431, -1.98929, -1.98398, -2.00128, -1.99469 … -1.96031, -1.97189, -1.97785, -1.98624,	[-0.747038, -0.74841, -0.747703, -0.75014, -0.754128, -0.759453, -0.75647, -0.766852, -0.762574, -0.767539 … -0.729413, -0.732674, -0.739823, -0.731681,	[0.856043, 0.857518, 0.851185, 0.859634, 0.858619, 0.854354, 0.905673, 1.16508, 1.42501, 1.71238 … 1.209, 1.07754, 0.957434, 0.851525, 0.757334,	[-2.03258, -2.0345, -2.03402, -2.03329, -2.03401, -2.034, -2.02314, -1.8548, -1.60643, -0.912175 … -1.78783, -1.91655, -2.02219, -2.06767,	[-0.865939, -0.864099, -0.862266, -0.86024, -0.852186, -0.85945, -0.842965, -0.9574, -1.05019, -1.05552 … -0.972505, -0.913282, -0.893715, -0.822636, -0.784828,	[- -0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0

Row	X[Hand tip l] Array…	Y[Hand tip l] Array…	Z[Hand tip l] Array…	X[Hand tip r] Array…	Y[Hand tip r] Array…	Z[Hand tip r] Array…	X[Ar
	-0.615468, -0.620129, -0.619546, -0.617533, -0.623453, -0.620921]	-1.98397, -1.97893, -1.97566, -1.97111, -1.96347, -1.97802]	-0.733116, -0.739864, -0.741084, -0.742643, -0.748294, -0.737384]	0.744572, 0.701962, 0.779988, 0.778182, 0.778619]	-2.06231, -2.04269, -2.01127, -1.73125, -1.7347, -1.73614]	-0.733635, -0.782048, -0.759991, -0.75973, -0.756357]	-0 -0 -0 -0 -0 -0
12	[-0.628575, -0.621757, -0.631781, -0.634901, -0.628957, -0.637745, -0.635805, -0.629445, -0.632091, -0.635745 … -0.591855, -0.582347, -0.588977, -0.598174, -0.599671, -0.619432, -0.621313, -0.628482, -0.633884, -0.627249]	[-1.64959, -1.65482, -1.6445, -1.65008, -1.65215, -1.64652, -1.64085, -1.65269, -1.64934, -1.64933 … -1.65049, -1.66718, -1.6695, -1.67345, -1.67695, -1.6821, -1.68837, -1.69011, -1.68936, -1.69211]	[-0.826129, -0.820825, -0.825739, -0.822362, -0.825972, -0.82397, -0.82843, -0.827873, -0.833039, -0.834004 … -0.850629, -0.836348, -0.83736, -0.830908, -0.825183, -0.81275, -0.804439, -0.805276, -0.816357, -0.813985]	[0.63093, 0.630534, 0.619773, 0.626157, 0.621542, 0.621468, 0.613933, 0.637572, 0.767764, 1.01477 … 1.33375, 1.03323, 0.808823, 0.693519, 0.658938, 0.589302, 0.627916, 0.67699, 0.70274, 0.72096]	[-1.73747, -1.73717, -1.7373, -1.73898, -1.7427, -1.74377, -1.73731, -1.74702, -1.75181, -1.73828 … -1.70125, -1.79771, -1.87956, -1.89704, -1.86995, -1.96147, -1.91128, -1.85101, -1.8577, -1.83568]	[-0.701359, -0.702162, -0.706464, -0.704693, -0.705164, -0.697885, -0.693803, -0.691132, -0.704814, -0.670501 … -0.380183, -0.407226, -0.445143, -0.422019, -0.475656, -0.479682, -0.47908, -0.507252, -0.51364, -0.539279]	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
13	[-0.635968, -0.613303, -0.636138, -0.635273, -0.636906, -0.633466, -0.633318, -0.628543, -0.627045, -0.624422 … -0.61882, -0.614184, -0.626042, -0.633471, -0.634397, -0.639003, -0.645128, -0.654244, -0.651518, -0.649729]	[-1.96703, -1.98867, -1.96746, -1.9631, -1.96258, -1.96333, -1.96385, -1.9627, -1.96265, -1.96134 … -1.84131, -1.85966, -1.85407, -1.83433, -1.84256, -1.83536, -1.82502, -1.82088, -1.82251, -1.80934]	[-0.661007, -0.672837, -0.656478, -0.658991, -0.65884, -0.659275, -0.655378, -0.658712, -0.654345, -0.651957 … -0.577366, -0.578097, -0.565833, -0.580407, -0.577185, -0.581951, -0.598315, -0.612136, -0.617067, -0.631079]	[0.630916, 0.629885, 0.629919, 0.630639, 0.629542, 0.63015, 0.630143, 0.630747, 0.630963, 0.629108 … 2.06203, 2.03448, 1.9007, 1.64908, 1.37607, 1.14622, 0.918773, 0.727559, 0.594801, 0.538287]	[-2.01777, -2.01644, -2.01636, -2.01498, -2.01403, -2.01435, -2.01093, -2.01459, -2.01451, -2.01158 … -0.037665, -0.489951, -0.950234, -1.36542, -1.63791, -1.74946, -1.86018, -1.92242, -1.96731, -1.97118]	[-0.713973, -0.720726, -0.720042, -0.715272, -0.72197, -0.720044, -0.719547, -0.715168, -0.712008, -0.717159 … -0.679075, -0.752105, -0.784518, -0.804115, -0.727146, -0.627818, -0.604224, -0.602402, -0.59308, -0.614281]	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	
349	[-0.616689, -0.614287, -0.616339, -0.611394, -0.615198, -0.652625,	[-1.94958, -1.94994, -1.94992, -1.95348, -1.93972, -2.09082,	[-0.755485, -0.760594, -0.755267, -0.755398, -0.751462, -0.75515,	[0.664354, 0.657678, 0.663926, 0.653829, 0.672965, 0.729495,	[-2.00496, -2.03505, -2.01516, -2.05454, -2.06293, -2.0968,	[-0.547348, -0.552963, -0.548251, -0.563437, -0.571891, -0.557266,	[-(-0 -0 -0 -0 -0

Row	X[Hand tip l] Array…	Y[Hand tip l] Array…	Z[Hand tip l] Array…	X[Hand tip r] Array…	Y[Hand tip r] Array…	Z[Hand tip r] Array…	X[Ar
	-0.758759, -0.926013, -1.16653, -1.45786 … -0.634068, -0.638292, -0.638414, -0.633326, -0.629956, -0.624907, -0.628131, -0.623601, -0.621252, -0.619131]	-2.07774, -2.07786, -2.03293, -1.95337 … -2.01936, -2.01745, -2.01658, -2.01221, -2.01132, -2.00823, -1.99968, -1.99785, -1.99715, -1.99688]	-0.740546, -0.726338, -0.712166, -0.657644 … -0.700128, -0.702082, -0.69888, -0.70226, -0.701131, -0.708299, -0.705904, -0.705396, -0.706151, -0.706053]	0.949406, 1.34799, 1.67983, 1.83766 … 0.595432, 0.637802, 0.679586, 0.636176, 0.597256, 0.593704, 0.595663, 0.594957, 0.59227, 0.590825]	-1.96131, -1.51774, -0.880203, -0.12666 … -2.20064, -1.99626, -1.91715, -1.99687, -2.19972, -2.19755, -2.19723, -2.19504, -2.19353, -2.19285]	-0.713681, -0.837412, -0.903952, -0.789053 … -0.568336, -0.569397, -0.532565, -0.553386, -0.551853, -0.553435, -0.551775, -0.55471, -0.553742, -0.553014]	-0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
350	[-0.596139, -0.596019, -0.594476, -0.591199, -0.59016, -0.587625, -0.589226, -0.59427, -0.614432, -0.712729 … -0.575696, -0.57971, -0.582036, -0.585876, -0.576416, -0.57689, -0.578217, -0.57949, -0.578794, -0.575571]	[-1.67946, -1.68074, -1.67967, -1.68043, -1.68028, -1.67404, -1.68079, -1.67452, -1.66515, -1.63637 … -1.6783, -1.66726, -1.66698, -1.66767, -1.6741, -1.67276, -1.67021, -1.66882, -1.66735, -1.66801]	[-0.41818, -0.410148, -0.404465, -0.404289, -0.397016, -0.404511, -0.393309, -0.403948, -0.41825, -0.471417 … -0.358823, -0.375101, -0.372223, -0.371538, -0.370928, -0.37131, -0.378007, -0.373494, -0.373474, -0.379876]	[0.496632, 0.49866, 0.496747, 0.50127, 0.493619, 0.49165, 0.497987, 0.49167, 0.516722, 0.582021 … 0.430446, 0.429438, 0.434283, 0.434753, 0.433317, 0.432706, 0.429335, 0.431152, 0.432418, 0.429916]	[-1.6367, -1.63157, -1.63873, -1.63019, -1.63165, -1.63523, -1.62979, -1.63167, -1.59549, -1.48439 … -1.70408, -1.7138, -1.70738, -1.70491, -1.69697, -1.70152, -1.71372, -1.70201, -1.70447, -1.70052]	[-0.739211, -0.742347, -0.733166, -0.738768, -0.734405, -0.734872, -0.736452, -0.734485, -0.784128, -0.965686 … -0.707477, -0.698281, -0.703048, -0.703093, -0.706035, -0.702741, -0.694049, -0.696337, -0.688899, -0.695393]	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
351	[-0.937442, -1.02995, -0.985338, -0.828794, -0.719483, -0.764852, -0.790658, -0.906044, -1.06143, -1.24269 … -1.11567, -0.915111, -0.690979, -0.683455, -0.758403, -0.722961, -0.661048, -0.674346, -0.687475, -0.68439]	[-2.10495, -1.96835, -1.95082, -2.09426, -2.13928, -2.08863, -2.23229, -2.28766, -2.27454, -2.06905 … -1.9192, -1.99392, -1.94095, -2.02532, -2.04187, -2.00124, -1.94335, -1.91709, -1.91269, -1.90738]	[-0.445069, -0.419463, -0.494451, -0.546546, -0.592364, -0.584375, -0.554674, -0.464802, -0.479882, -0.533597 … -0.758341, -0.621436, -0.620878, -0.632757, -0.6223, -0.690714, -0.73709, -0.740402, -0.738781, -0.740575]	[0.593855, 0.611397, 0.602737, 0.584278, 0.563043, 0.581611, 0.588664, 0.595967, 0.858367, 1.31364 … 0.827272, 0.658031, 0.608874, 0.625905, 0.608621, 0.608801, 0.613776, 0.596233, 0.588538, 0.590146]	[-2.01059, -2.01198, -2.019, -2.02334, -2.07292, -2.11766, -2.14359, -2.12517, -2.2062, -2.03789 … -2.15671, -1.94299, -1.94304, -1.95421, -1.96479, -1.95365, -1.95082, -1.93788, -1.94164, -1.88977]	[-0.460162, -0.438999, -0.433038, -0.449091, -0.514482, -0.526686, -0.511987, -0.479525, -0.090289, 0.001343 … -0.13886, -0.521114, -0.532044, -0.553127, -0.572179, -0.632077, -0.637166, -0.669985, -0.66622, -0.659614]	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0

Row	X[Hand tip l] Array…	Y[Hand tip l] Array…	Z[Hand tip l] Array…	X[Hand tip r] Array…	Y[Hand tip r] Array…	Z[Hand tip r] Array…	X[Ar
352	[-0.595625, -0.593605, -0.588495, -0.584801, -0.59971, -0.595148, -0.59403, -0.591524, -0.591834, -0.591661 … -0.631753, -0.613123, -0.604525, -0.578524, -0.539126, -0.486437, -0.4675, -0.453359, -0.445844, -0.454189]	[-1.93897, -1.93641, -1.93761, -1.91826, -1.97002, -1.97009, -1.966, -1.94677, -1.94209, -1.93865 … -2.05116, -2.02755, -1.97911, -1.93749, -1.94224, -1.95526, -1.95181, -1.94552, -1.94973, -1.96401]	[-0.683665, -0.694508, -0.682787, -0.701766, -0.747765, -0.744019, -0.744061, -0.72129, -0.711405, -0.707665 … -0.40507, -0.479012, -0.569021, -0.649807, -0.70869, -0.75495, -0.780949, -0.791049, -0.774547, -0.751589]	[0.676463, 0.682937, 0.682541, 0.685074, 0.686095, 0.681547, 0.680452, 0.681045, 0.684939, 0.676113 … 0.660947, 0.637387, 0.590117, 0.57134, 0.574835, 0.595796, 0.619714, 0.625665, 0.638661, 0.6339]	[-1.8211, -1.82324, -1.82337, -1.82025, -1.87301, -1.86164, -1.8572, -1.83805, -1.83235, -1.82618 … -1.90219, -1.88858, -1.87824, -1.87888, -1.88442, -1.89448, -1.90057, -1.90804, -1.90383, -1.91231]	[-0.693638, -0.693881, -0.685873, -0.687236, -0.726534, -0.726996, -0.728512, -0.706081, -0.696437, -0.698008 … -0.650799, -0.709445, -0.729737, -0.740974, -0.743894, -0.735067, -0.731009, -0.713037, -0.708452, -0.695166]	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
353	[-0.466582, -0.469372, -0.43454, -0.432809, -0.480658, -0.500828, -0.532364, -0.552448, -0.453876, -0.439888 … -0.715931, -0.734048, -0.696927, -0.643663, -0.575416, -0.492768, -0.422277, -0.389059, -0.394768, -0.411497]	[-1.71874, -1.72861, -1.58691, -1.58583, -1.70418, -1.64664, -1.61578, -1.6056, -1.52025, -1.30873 … -1.81162, -1.98329, -1.972, -1.93991, -1.8876, -1.85807, -1.83981, -1.81063, -1.75395, -1.65299]	[-1.03662, -1.03794, -0.949864, -0.954459, -1.04864, -1.078, -1.1047, -1.15505, -1.40861, -1.56395 … -0.820564, -0.976599, -1.03893, -1.08675, -1.09675, -1.09772, -1.07486, -1.03899, -1.04458, -1.0622]	[0.59468, 0.594129, 0.593619, 0.602656, 0.641599, 0.735446, 0.882214, 1.15693, 1.48269, 1.78098 … 0.40946, 0.618935, 0.637069, 0.640586, 0.635214, 0.636883, 0.636803, 0.632184, 0.609569, 0.601727]	[-1.84908, -1.84819, -1.85054, -1.85237, -1.85198, -1.90322, -1.86753, -1.73999, -1.51691, -1.05561 … -1.63182, -1.88926, -1.90622, -1.89375, -1.86071, -1.83122, -1.80992, -1.7971, -1.7638, -1.75888]	[-0.365797, -0.368545, -0.357051, -0.344981, -0.34665, -0.378178, -0.396243, -0.545071, -0.604943, -0.635584 … -0.569574, -0.114756, -0.139989, -0.165661, -0.214669, -0.267517, -0.306533, -0.329637, -0.362193, -0.385993]	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
354	[-0.500404, -0.502824, -0.504771, -0.505733, -0.505021, -0.510656, -0.477738, -0.455724, -0.450382, -0.43636 … -0.62565, -0.617651, -0.588326, -0.552376, -0.529874,	[-1.89209, -1.88986, -1.89188, -1.88755, -1.88602, -1.92025, -1.89595, -1.88366, -1.85715, -1.8259 … -1.95207, -1.95965, -1.95948, -1.98069, -1.98885,	[-0.667846, -0.671957, -0.666312, -0.67298, -0.671848, -0.673591, -0.705428, -0.734485, -0.761981, -0.783711 … -0.707524, -0.715147, -0.718432, -0.710816,	[0.604001, 0.608896, 0.619072, 0.620853, 0.622834, 0.630271, 0.618775, 0.598823, 0.585954, 0.59969 … 0.562149, 0.525271, 0.513633, 0.528847, 0.554589,	[-1.73447, -1.74005, -1.74104, -1.73596, -1.73738, -1.7684, -1.76486, -1.75967, -1.71397, -1.62112 … -1.75231, -1.76314, -1.76987, -1.79338,	[-0.852622, -0.847307, -0.850225, -0.849992, -0.843751, -0.848163, -0.854097, -0.872344, -0.9205, -0.982214 … -0.955112, -0.961629, -0.94402, -0.937481,	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0

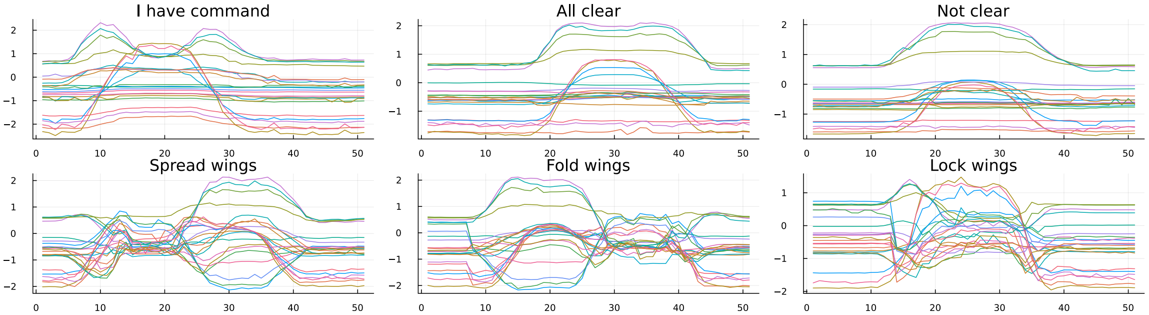
Row	X[Hand tip l] Array…	Y[Hand tip l] Array…	Z[Hand tip l] Array…	X[Hand tip r] Array…	Y[Hand tip r] Array…	Z[Hand tip r] Array…	X[A
	-0.553498, -0.572458, -0.588936, -0.604985, -0.611643]	-1.97063, -1.99709, -2.00647, -2.00631, -2.00625]	-0.693616, -0.713655, -0.704543, -0.704291, -0.710759, -0.712378]	0.577905, 0.581103, 0.576899, 0.575532, 0.577343]	-1.80963, -1.81876, -1.82579, -1.82698, -1.80892, -1.808]	-0.934921, -0.935206, -0.925465, -0.917304, -0.927874, -0.930375]	-0 -0 -0 -0 -0 -0
355	[-0.686893, -0.690966, -0.710514, -0.771405, -0.865657, -1.10153, -1.28476, -1.3701, -1.17664, -0.759236 … -0.590119, -0.586947, -0.58872, -0.586027, -0.586456, -0.587513, -0.590223, -0.588903, -0.591829, -0.591138]	[-2.04375, -2.05011, -2.07035, -2.08768, -2.11075, -2.08723, -2.01236, -1.89968, -1.62723, -1.3901 … -2.01395, -2.0175, -2.01779, -2.00809, -2.00814, -2.00896, -2.01059, -2.01857, -2.0202, -2.01936]	[-0.763731, -0.739648, -0.709582, -0.698519, -0.695876, -0.734551, -0.780351, -0.911202, -1.30088, -1.56614 … -0.750251, -0.749495, -0.755412, -0.75954, -0.761369, -0.763274, -0.76296, -0.76822, -0.767053, -0.773619]	[0.619419, 0.641105, 0.677359, 0.750174, 0.969919, 1.55344, 1.8599, 2.083, 1.98471, 1.68652 … 0.736513, 0.739569, 0.743421, 0.743809, 0.735055, 0.766925, 0.749379, 0.748856, 0.740557, 0.744968]	[-2.08314, -2.08161, -2.08674, -2.08547, -2.02839, -1.70898, -1.29341, -0.711759, 0.182653, 0.622182 … -1.97317, -1.97663, -1.97623, -1.97845, -2.14337, -2.07043, -1.98182, -1.9928, -2.13751, -2.12822]	[-0.63965, -0.621357, -0.591876, -0.534126, -0.468331, -0.174424, -0.017619, 0.10863, 0.149411, 0.097924 … -0.510329, -0.509272, -0.517616, -0.512647, -0.534151, -0.5225, -0.516237, -0.519779, -0.529337, -0.530785]	[-(-0 -0 -0 -0 -0 -0 -0 -1 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
356	[-0.525938, -0.516073, -0.5177, -0.516002, -0.517101, -0.531324, -0.598619, -0.632816, -0.641388, -0.624213 … -0.506511, -0.53477, -0.536864, -0.537779, -0.545924, -0.540783, -0.544388, -0.550136, -0.546228, -0.545995]	[-1.69259, -1.70519, -1.70824, -1.72114, -1.72487, -1.74597, -1.73602, -1.70956, -1.66064, -1.55737 … -1.7578, -1.67215, -1.66308, -1.6627, -1.64937, -1.66342, -1.66382, -1.66711, -1.67062, -1.66981]	[-0.514372, -0.521267, -0.51823, -0.507091, -0.489786, -0.500235, -0.559973, -0.659927, -0.761855, -0.891289 … -0.536724, -0.524299, -0.517065, -0.522152, -0.517777, -0.526417, -0.521008, -0.531884, -0.52938, -0.531231]	[0.385693, 0.391897, 0.387426, 0.391813, 0.416009, 0.49509, 0.833951, 1.08184, 1.34254, 1.6102 … 0.473226, 0.439757, 0.435853, 0.444632, 0.418755, 0.410171, 0.416042, 0.413618, 0.409138, 0.410571]	[-1.71975, -1.72145, -1.72375, -1.72402, -1.73314, -1.73277, -1.60902, -1.50969, -1.29886, -0.787043 … -1.7279, -1.71693, -1.71674, -1.7201, -1.72104, -1.72351, -1.74602, -1.75108, -1.74972, -1.74926]	[-0.715464, -0.711937, -0.712161, -0.714993, -0.70302, -0.716358, -0.748026, -0.79191, -0.858587, -0.998692 … -0.702175, -0.710662, -0.711929, -0.700321, -0.716329, -0.719247, -0.716009, -0.72007, -0.723947, -0.720406]	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
357	[-0.440887, -0.452221, -0.447185, -0.451468, -0.451852, -0.460196, -0.454885, -0.456595,	[-1.90444, -1.91917, -1.91652, -1.92718, -1.92486, -1.93992, -1.93326, -1.931,	[-1.07556, -1.08468, -1.08069, -1.08455, -1.08098, -1.08955, -1.08634, -1.08236,	[0.601969, 0.607711, 0.600509, 0.620354, 0.619002, 0.601562, 0.59883, 0.585152,	[-2.15266, -2.17978, -2.19577, -2.043, -2.03998, -2.21929, -2.21141, -2.21772,	[-0.432073, -0.435872, -0.42879, -0.406766, -0.410107, -0.432601, -0.431501, -0.438689,	[-(-0 -0 -0 -0 -0 -0 -0

Row	X[Hand tip l] Array…	Y[Hand tip l] Array…	Z[Hand tip l] Array…	X[Hand tip r] Array…	Y[Hand tip r] Array…	Z[Hand tip r] Array…	X[Ar
	-0.453205, -0.469354 … 0.181187, -0.131194, -0.541423, -0.777776, -0.953652, -1.04438, -0.984894, -0.891926, -0.788926, -0.698249]	-1.89117, -1.90504 … -1.18755, -1.39716, -1.36595, -1.59362, -1.67972, -1.74415, -1.83087, -1.86974, -1.92518, -1.86742]	-1.07372, -1.08595 … -1.38186, -1.4358, -1.35391, -1.36938, -1.34895, -1.24551, -1.14865, -1.07423, -1.03839, -1.01313]	0.58419, 0.583967 … 1.82576, 2.01202, 1.99811, 2.14004, 1.9395, 1.61937, 1.20757, 0.874289, 0.691181, 0.562766]	-2.21474, -2.23002 … 0.350891, 0.023507, -0.423043, -0.812878, -1.24957, -1.70558, -2.05941, -2.229, -2.24742, -2.19937]	-0.431665, -0.434617 … -0.097338, -0.017904, 0.118915, 0.111467, 0.145836, 0.084069, 0.021073, -0.070298, -0.066992, -0.291861]	-0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
358	[-0.647672, -0.653511, -0.642305, -0.6383, -0.637352, -0.641916, -0.641393, -0.641784, -0.644001, -0.646141 … -0.224099, -0.500692, -0.682501, -0.758069, -0.765982, -0.72369, -0.645501, -0.624031, -0.62637, -0.628032]	[-1.6173, -1.61051, -1.60491, -1.59955, -1.60063, -1.60052, -1.60037, -1.60233, -1.61603, -1.62083 … -1.3884, -1.479, -1.63708, -1.65932, -1.664, -1.6762, -1.69198, -1.6986, -1.69814, -1.69324]	[-0.505743, -0.499972, -0.497074, -0.500385, -0.494025, -0.496815, -0.49508, -0.503109, -0.513043, -0.515419 … -0.97989, -0.827817, -0.673086, -0.504513, -0.400298, -0.340097, -0.315575, -0.317331, -0.321467, -0.327659]	[0.500621, 0.496023, 0.498986, 0.498838, 0.504341, 0.500749, 0.503862, 0.504645, 0.505147, 0.504536 … 0.293131, 0.443811, 0.581705, 0.631393, 0.575838, 0.527186, 0.534738, 0.529456, 0.518023, 0.517005]	[-1.61356, -1.61504, -1.61717, -1.61682, -1.61745, -1.61724, -1.6164, -1.6166, -1.61653, -1.61602 … -0.727617, -1.1854, -1.55158, -1.64122, -1.65639, -1.67336, -1.67886, -1.68227, -1.6856, -1.68569]	[-0.682743, -0.680679, -0.674725, -0.676847, -0.673253, -0.674156, -0.676122, -0.674299, -0.673238, -0.675257 … -1.2196, -1.11147, -0.889893, -0.700757, -0.638817, -0.619261, -0.614219, -0.615918, -0.62318, -0.623297]	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0
359	[-0.476117, -0.4705, -0.474443, -0.475678, -0.475745, -0.476801, -0.466286, -0.470162, -0.466426, -0.456026 … -0.310213, -0.498413, -0.660566, -0.792202, -0.859166, -0.837351, -0.769369, -0.684986, -0.545003, -0.53447]	[-1.70846, -1.7099, -1.70912, -1.70614, -1.70891, -1.70891, -1.70919, -1.70437, -1.71174, -1.71172 … -1.01014, -1.24986, -1.4634, -1.57394, -1.71292, -1.80086, -1.85209, -1.84221, -1.8798, -1.78985]	[-0.5028, -0.509458, -0.508224, -0.509525, -0.508446, -0.505009, -0.504267, -0.49798, -0.500755, -0.489398 … -1.27009, -1.1431, -1.00582, -0.793894, -0.569638, -0.399036, -0.319151, -0.279739, -0.295245, -0.332945]	[0.379579, 0.380022, 0.381541, 0.37959, 0.387175, 0.383584, 0.39284, 0.397443, 0.395351, 0.395874 … 0.733999, 0.871185, 0.973005, 0.972278, 0.867967, 0.770433, 0.654358, 0.552439, 0.431852, 0.378889]	[-1.62798, -1.63086, -1.63214, -1.63089, -1.62406, -1.63349, -1.62449, -1.60594, -1.59181, -1.58358 … 0.25072, -0.203643, -0.645084, -1.02025, -1.43843, -1.60047, -1.74759, -1.79286, -1.89259, -1.82064]	[-0.840636, -0.834709, -0.83162, -0.833223, -0.83019, -0.823642, -0.817607, -0.812333, -0.80052, -0.792545 … -1.65719, -1.75717, -1.73933, -1.55679, -1.24865, -1.06737, -0.932671, -0.84789, -0.79027, -0.771634]	[-(-0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0

Row	X[Hand tip l]	Y[Hand tip l]	Z[Hand tip l]	X[Hand tip r]	Y[Hand tip r]	Z[Hand tip r]	X
	Array…	Array…	Array…	Array…	Array…	Array…	A
360	[-0.553245, -0.551704, -0.548044, -0.544929, -0.546446, -0.546651, -0.554797, -0.580788, -0.634431, -0.679319 … -0.605506, -0.606554, -0.662612, -0.709257, -0.727722, -0.721847, -0.687693, -0.655425, -0.627145, -0.613324]	[-1.69493, -1.69349, -1.69951, -1.7085, -1.71117, -1.71164, -1.70638, -1.68895, -1.66512, -1.63018 … -1.76228, -1.75493, -1.74816, -1.73299, -1.72764, -1.72671, -1.725, -1.72086, -1.71081, -1.70149]	[-0.756008, -0.762863, -0.760376, -0.756736, -0.756973, -0.758584, -0.766189, -0.790081, -0.83163, -0.879984 … -0.568478, -0.590416, -0.601258, -0.618041, -0.641417, -0.648827, -0.660898, -0.674366, -0.687214, -0.689185]	[0.569726, 0.570003, 0.573506, 0.572164, 0.621714, 0.696098, 0.876118, 1.12203, 1.47091, 1.64837 … 0.503982, 0.51867, 0.550847, 0.56638, 0.586101, 0.585824, 0.581762, 0.575169, 0.572896, 0.568447]	[-1.8093, -1.80937, -1.80944, -1.80935, -1.8126, -1.85915, -1.8061, -1.67266, -1.36509, -1.061 … -1.84879, -1.81557, -1.82067, -1.82043, -1.8186, -1.8199, -1.81979, -1.8188, -1.81805, -1.81838]	[-0.448892, -0.45103, -0.445198, -0.449493, -0.440023, -0.397585, -0.392426, -0.419542, -0.481797, -0.566341 … -0.39616, -0.374028, -0.372259, -0.369527, -0.37349, -0.374199, -0.375562, -0.382328, -0.383967, -0.382463]	[-0 -0 -0 -0 -0 -0 -0 -0 -0 -0 … -0 -0 -0 -0 -0 -0 -0 -0 -0 -0

```
In [6]:  # Let's inspect an instance for each class.
         plot(map(i->plot(collect(X_df[i,:]), labels=nothing,title=y[i]), 1:30:180)..
```

Out[6]:



```
In [48]: # Obtain the graph structure (i.e., the "frame") of the first instance.
         # It is a "dimensional" frame, of all intervals in [1,51].
         fr = SoleLogics.frame(X_df, 1)
```

```
Out[48]: SoleLogics.FullDimensionalFrame{1, Interval{Int64}}((51,))
```

```
In [50]: # Enumerate all worlds
         allworlds(fr)
```

```
Out[50]: Base.Generator{Base.Iterators.Zip{Tuple{Base.Iterators.Filter{SoleLogics.va
         r"#128#129", Base.Iterators.ProductIterator{Tuple{UnitRange{Int64}, UnitRan
         ge{Int64}}}}}}, IterTools.var"#1#2"{DataType}}(IterTools.var"#1#2"{DataTyp
         e}(Interval{Int64}), zip(Base.Iterators.Filter{SoleLogics.var"#128#129", Ba
         se.Iterators.ProductIterator{Tuple{UnitRange{Int64}, UnitRange{Int64}}}}(So
         leLogics.var"#128#129"(), Base.Iterators.ProductIterator{Tuple{UnitRange{In
         t64}, UnitRange{Int64}}}((1:51, 2:52)))))
```

In [56]:

```julia
using SoleLogics: Interval
# Enumerate the intervals that are "Later" than [1,10]
collect(accessibles(fr, Interval(1,10), IA_L))
```

Out[56]:

```
1326-element Vector{Interval{Int64}}:
 Interval{Int64}(1, 2)
 Interval{Int64}(1, 3)
 Interval{Int64}(2, 3)
 Interval{Int64}(1, 4)
 Interval{Int64}(2, 4)
 Interval{Int64}(3, 4)
 Interval{Int64}(1, 5)
 Interval{Int64}(2, 5)
 Interval{Int64}(3, 5)
 Interval{Int64}(4, 5)
 Interval{Int64}(1, 6)
 Interval{Int64}(2, 6)
 Interval{Int64}(3, 6)
 ⋮
 Interval{Int64}(40, 52)
 Interval{Int64}(41, 52)
 Interval{Int64}(42, 52)
 Interval{Int64}(43, 52)
 Interval{Int64}(44, 52)
 Interval{Int64}(45, 52)
 Interval{Int64}(46, 52)
 Interval{Int64}(47, 52)
 Interval{Int64}(48, 52)
 Interval{Int64}(49, 52)
 Interval{Int64}(50, 52)
 Interval{Int64}(51, 52)
```

In [11]:

```julia
# Remember that features are computed on each world
# Let's compute the minimum of the first variable on an arbitrary interval,
feature = UnivariateMin(1)
```

Out[11]:

```
UnivariateMin: min[V1]
```

In [12]:

```julia
Sole.featvalue(feature, X_df, 1, Interval(10,30))
```

Out[12]:

```
-0.444535
```

In [72]:

```julia
for i in allworlds(Sole.frame(X_df, 1))
    Sole.featvalue(feature, X_df, 1, i)
end
```

In [14]:

```julia
# Dataset -> multi-modal Kripke Logiset
X = scalarlogiset(X_df)
```

Out[14]:

```
SupportedLogiset with 1 support (343.08 MBs)
├ worldtype:                   Interval{Int64}
├ featvaltype:                 Float64
├ featuretype:                 SoleModels.AbstractUnivariateFeature
├ frametype:                   SoleLogics.FullDimensionalFrame{1, Interval
{Int64}}
├ # instances:                 360
├ usesfullmemo:                true
├[BASE] UniformFullDimensionalLogiset of channel size (51,) (342.91 MBs)
│ ├ size × eltype:              (51, 51, 360, 48) × Float64
│ └ features:                   48 -> SoleModels.AbstractUnivariateFeature
["max[V1]", "min[V1]", "max[V2]", "min[V2]", "...", "min[V22]", "max[V23]",
"min[V23]", "max[V24]", "min[V24]"]
└[SUPPORT 1] FullMemoset (0 memoized values, 174.42 KBs))
```

In [15]:

```julia
# Remember that atoms are *scalar conditions on features*
# Let's check one on an interval of the first instance
p = Atom(ScalarCondition(feature, >, -0.5))
check(p, X, 1, Interval(10,30))
```

Out[15]:

```
true
```

In [16]:

```julia
# Of course I can check any formula
p = Atom(ScalarCondition(UnivariateMin(1), >, -0.5))
q = Atom(ScalarCondition(UnivariateMin(2), <=, 10))
φ = p ∨ q
println(syntaxstring(φ))

check(φ, X, 1, Interval(10,30))
```

```
min[V1] > -0.5 ∨ min[V2] ≤ 10
```

Out[16]:

```
true
```

In [17]:

```julia
boxlater = box(IA_L)
```

Out[17]:

```
BoxRelationalConnective{SoleLogics._IA_L}
syntaxstring: [L]
```

In [18]:

```julia
lateralwaysφ = boxlater(φ)
```

Out[18]:

```
SyntaxBranch{BoxRelationalConnective{SoleLogics._IA_L}}: [L](min[V1] > -0.5
∨ min[V2] ≤ 10)
```

In [19]:

```julia
check(lateralwaysφ, X, 1, Interval(10,30))
```

Out[19]:

```
true
```

In [75]:

```julia
# Generate a random HS formula with scalar conditions on features, and check
features = [UnivariateMin(i_variable) for i_variable in 1:ncol(X_df)]
alpha = [Atom(ScalarCondition(feat, >, thresh)) for feat in features for thr

HS_connectives = SoleLogics.diamondsandboxes(SoleLogics.IARelations)
propo_connectives = SoleLogics.BASE_PROPOSITIONAL_CONNECTIVES

println("Propositional connectives: $(join(syntaxstring.(propo_connectives),
println("HS connectives: $(join(syntaxstring.(HS_connectives), ", "))")

propo_weights = fill(1/length(propo_connectives), length(propo_connectives))
HS_weights = fill(1/length(HS_connectives), length(HS_connectives))

connectives = vcat(propo_connectives, HS_connectives)

opweights = vcat(propo_weights, HS_weights)

treeheight = 3
φ2 = randformula(Random.MersenneTwister(30), treeheight, alpha, connectives;
println()
println("Random formula:")
println(syntaxstring(φ2))

# Check on the first instance
check(φ2, X, 1, Interval(10,30))
```

```
Propositional connectives: ¬, ∧, ∨, →
HS connectives: ⟨A⟩, [A], ⟨L⟩, [L], ⟨B⟩, [B], ⟨E⟩, [E], ⟨D⟩, [D], ⟨O⟩, [O],
⟨A̅⟩, [A̅], ⟨L̅⟩, [L̅], ⟨B̅⟩, [B̅], ⟨E̅⟩, [E̅], ⟨D̅⟩, [D̅], ⟨O̅⟩, [O̅]

Random formula:
¬((min[V8] > 0.3 → min[V13] > 0.2) → min[V18] > 0.6 ∨ min[V16] > 0.2)
```

Out[75]:

```
false
```

In [77]:

```
# Let's check a formula on all instances
check_mask = check(φ2, X, Interval(10,30))
println(check_mask)
```

```
Bool[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1,
0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1,
0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
```

In [22]:

```
# It holds on part of the instances
sum(check_mask)
```

Out[22]:

```
98
```

In [23]:

```
# Let's ask whether the formula holds *all* intervals, instead of checking i
println("Applying the universal global operator: ", SoleLogics.globalbox)
println()

universal_φ = globalbox(φ2)
```

```
println("Formula: ", syntaxstring(universal_φ))
check_mask = check(universal_φ, X)

# It holds on no instance... Too restrictive?
sum(check_mask)
```

```
Applying the universal global operator: [G]

Formula: [G]¬((min[V8] > 0.3 → min[V13] > 0.2) → min[V18] > 0.6 ∨ min[V16] >
0.2)
```

Out[23]: 0

In [24]:

```
# Let's ask whether there exists any interval where the formula holds
println("Applying the existential global operator: ", SoleLogics.globaldiamc
println()

existential_φ = globaldiamond(φ2)
println("Formula: ", syntaxstring(existential_φ))
check_mask = check(existential_φ, X)

# It holds on more instances!
sum(check_mask)
```

```
Applying the existential global operator: ⟨G⟩

Formula: ⟨G⟩¬((min[V8] > 0.3 → min[V13] > 0.2) → min[V18] > 0.6 ∨ min[V16] >
0.2)
```

Out[24]: 127

In [25]:

```
# Question: does it lead to a good rule?

println(syntaxstring(existential_φ))

println()
println(SoleLogics.experimentals.formula2natlang(existential_φ))
```

```
⟨G⟩¬((min[V8] > 0.3 → min[V13] > 0.2) → min[V18] > 0.6 ∨ min[V16] > 0.2)

∃ interval where (¬(whenever whenever min[V8] > 0.3 holds, also min[V13] >
0.2 holds, also (min[V18] > 0.6) or (min[V16] > 0.2)))
```

In [26]:

```
neg_existential_φ = normalize(¬ existential_φ;
    profile = :readability,
    remove_implications = true,
    allow_atom_flipping = true
)

println()
print("Formula:\n\t")
println(syntaxstring(neg_existential_φ))

println()
print("Natural language translation:\n\t")
println(SoleLogics.experimentals.formula2natlang(neg_existential_φ))

println()
```

```julia
print("Using abbrevations for feature-test operator pairs:\n\t")
println(SoleLogics.experimentals.formula2natlang(neg_existential_φ; use_feat
```

```
Formula:
	[G](min[V18] > 0.6 ∨ (min[V8] > 0.3 ∧ min[V13] ≤ 0.2) ∨ min[V16] >
0.2)

Natural language translation:
	∀ intervals ((min[V18] > 0.6) or (((min[V8] > 0.3) and (min[V13] ≤
0.2)) or (min[V16] > 0.2)))

Using abbrevations for feature-test operator pairs:
	∀ intervals ((V18 ≻ 0.6) or (((V8 ≻ 0.3) and (V13 ⤓ 0.2)) or (V16 ≻
0.2)))
```

In [27]:

```julia
countmap(y)
```

Out[27]:

```
Dict{CategoricalArrays.CategoricalValue{String, UInt32}, Int64} with 6 entr
ies:
  "Spread wings"   => 60
  "I have command" => 60
  "Not clear"      => 60
  "Lock wings"     => 60
  "All clear"      => 60
  "Fold wings"     => 60
```

In [28]:

```julia
println(neg_existential_φ)
countmap(y[(!).(check_mask)])
```

```
SyntaxBranch{BoxRelationalConnective{GlobalRel}}: [G](min[V18] > 0.6 ∨ (min
[V8] > 0.3 ∧ min[V13] ≤ 0.2) ∨ min[V16] > 0.2)
```

Out[28]:

```
Dict{CategoricalArrays.CategoricalValue{String, UInt32}, Int64} with 6 entr
ies:
  "Spread wings"   => 2
  "I have command" => 60
  "Not clear"      => 59
  "Lock wings"     => 50
  "All clear"      => 59
  "Fold wings"     => 3
```

(Pure) Decision Trees

In [29]:

```julia
# A first decision tree:
mydecisiontree = DecisionTree(neg_existential_φ, "I have command", "Spread w
printmodel(mydecisiontree; use_feature_abbreviations = true)

# Accuracy
y_preds = SoleModels.apply(mydecisiontree, X)
println("Random chance: $(60/length(y))")
println("Accuracy: $(sum(y .== y_preds)/length(y))")
```

```
▣ [G]((V18 ≻ 0.6) ∨ ((V8 ≻ 0.3) ∧ (V13 ⤓ 0.2)) ∨ (V16 ≻ 0.2))
├✔ I have command
└✘ Spread wings

Random chance: 0.16666666666666666
Accuracy: 0.32777777777777778
```

In [30]:
```
# Obtain a *partitioning* set of rules
listrules(mydecisiontree)
```

Out[30]:
```
2-element Vector{Rule{String, A, SoleModels.ConstantModel{String}} where A
<:Formula}:
 ▣ [G]((min[V18] > 0.6) ∨ ((min[V8] > 0.3) ∧ (min[V13] ≤ 0.2)) ∨ (min[V16]
> 0.2))  ➡  I have command

 ▣ ¬[G]((min[V18] > 0.6) ∨ ((min[V8] > 0.3) ∧ (min[V13] ≤ 0.2)) ∨ (min[V16]
> 0.2))  ➡  Spread wings
```

In [31]:
```
# DecisionTree's can be composed
mydecisiontree2 = DecisionTree(globaldiamond(φ), mydecisiontree, "Not clear"
printmodel(mydecisiontree2; use_feature_abbreviations = true)

# Obtain a *partitioning* set of rules.
listrules(mydecisiontree2)
```

```
▣ ⟨G⟩((V1 ≻ -0.5) ∨ (V2 ⤓ 10))
├✔ [G]((V18 ≻ 0.6) ∨ ((V8 ≻ 0.3) ∧ (V13 ⤓ 0.2)) ∨ (V16 ≻ 0.2))
│├✔ I have command
│└✘ Spread wings
└✘ Not clear
```

Out[31]:
```
3-element Vector{Rule{String, A, SoleModels.ConstantModel{String}} where A
<:Formula}:
 ▣ ⟨G⟩((min[V1] > -0.5) ∨ (min[V2] ≤ 10)) ∧ [G]((min[V18] > 0.6) ∨ ((min[V
8] > 0.3) ∧ (min[V13] ≤ 0.2)) ∨ (min[V16] > 0.2))  ➡  I have command

 ▣ ⟨G⟩((min[V1] > -0.5) ∨ (min[V2] ≤ 10)) ∧ ¬[G]((min[V18] > 0.6) ∨ ((min[V
8] > 0.3) ∧ (min[V13] ≤ 0.2)) ∨ (min[V16] > 0.2))  ➡  Spread wings

 ▣ ¬⟨G⟩((min[V1] > -0.5) ∨ (min[V2] ≤ 10))  ➡  Not clear
```

Exercise 1: a first decision tree algorithm

Inspired by the previous experiment, write a function that inductively learns a decision tree by generating random formulas of bounded height. The procedure should build the tree in a recursive, CART-like fashion where each call should:

- Check if the dataset is pure (that is, if all labels are the same);
- Find a formula that splits *well-enough* (define your own metric) the logiset;
- Split the logiset into instances that satisfy the formula, and those that do not;
- Recourse on the sub-logisets.

In [32]:

```julia
# Generate a random decision tree
function randdecisiontree(rng, X, y, depth = 0; treeheight = 2)
    if depth == 0
        return SoleModels.ConstantModel(mode(y))
    else
        check_mask = nothing
        print("depth: $depth")
        while (
            φ = randformula(rng, treeheight, alpha, connectives; opweights =
            φ = globaldiamond(φ);
            check_mask = check(φ, X);
            allequal(check_mask)
        )
            print(".")
        end
        println()
        Xleft = slicedataset(X, check_mask)
        Xright = slicedataset(X, (!).(check_mask))
        depthleft, depthright = (rand(rng, Bool) ? (depth-1, rand(rng, 0:(de
        dtleft = randdecisiontree(rng, Xleft, y[check_mask], depthleft; tree
        dtright = randdecisiontree(rng, Xright, y[(!).(check_mask)], depthri
        return DecisionTree(φ, dtleft, dtright)
    end
end

println("Learning a decision tree...")
rng = Random.Xoshiro(1)
mydecisiontree2 = @time randdecisiontree(rng, X, y, 3)

println()
printmodel(mydecisiontree2)

println()
y_preds = SoleModels.apply(mydecisiontree2, X)
println("Accuracy: $(sum(y .== y_preds)/length(y))")
```

```
Learning a decision tree...
depth: 3
depth: 2......
depth: 1........
 97.801960 seconds (1.03 G allocations: 131.200 GiB, 13.74% gc time, 14.60%
compilation time: <1% of which was recompilation)

▣ ⟨G⟩¬((min[V17] > 0.8) ∨ (min[V16] > 0.0))
├✔ Spread wings
└✘ ⟨G⟩(((min[V20] > 0.2) → (min[V16] > 0.9)) → (min[V19] > 0.7) ∨ (min[V5] >
0.7))
 ├✔ I have command
 └✘ ⟨G⟩(((min[V7] > 0.9) ∨ (min[V23] > 0.0)) ∧ [O̅](min[V20] > 0.1))
  ├✔ All clear
  └✘ Not clear


Accuracy: 0.46111111111111114
```

In [33]:

```
# Note: while checking the formulas, some values were *memoized*
println(X)
```

```
SupportedLogiset with 1 support (586.12 MBs)
├ worldtype:                   Interval{Int64}
├ featvaltype:                 Float64
├ featuretype:                 SoleModels.AbstractUnivariateFeature
├ frametype:                   SoleLogics.FullDimensionalFrame{1, Interval{I
nt64}}
├ # instances:                 360
├ usesfullmemo:                true
├[BASE] UniformFullDimensionalLogiset of channel size (51,) (342.91 MBs)
│ ├ size × eltype:              (51, 51, 360, 48) × Float64
│ └ features:                   48 -> SoleModels.AbstractUnivariateFeature
["max[V1]", "min[V1]", "max[V2]", "min[V2]", "...", "min[V22]", "max[V23]",
"min[V23]", "max[V24]", "min[V24]"]
└[SUPPORT 1] FullMemoset (24672 memoized values, 243.21 MBs))
```

CART-like learning of Modal Decision Trees

In [34]:

```
# Downsize the temporal axis by moving averages
X_df_small = broadcast(x->movingwindow(mean, x; nwindows = 10, relative_over
```

Out[34]: 360×24 DataFrame *335 rows omitted*

Row	X[Hand tip l]	Y[Hand tip l]	Z[Hand tip l]	X[Hand tip r]	Y[Hand tip r]	Z[Hand tip r]
	Array…	**Array…**	**Array…**	**Array…**	**Array…**	**Array…**
1	[-0.520732, -0.478074, -0.407314, -0.395833, -0.398731, -0.424229, -0.433961, -0.440746, -0.462338, -0.462312]	[-2.17083, -2.10124, -1.82028, -1.6872, -1.71429, -1.91497, -2.10415, -2.1442, -2.17028, -2.15161]	[-0.965304, -0.968721, -0.932382, -0.89114, -0.908892, -0.961698, -1.02626, -1.04462, -1.03579, -1.0214]	[0.749126, 1.65307, 1.8441, 0.981466, 1.45814, 1.77602, 1.00715, 0.763208, 0.753621, 0.744364]	[-2.36284, -1.55874, 0.496972, 1.36516, 0.89391, -0.954975, -2.24809, -2.37978, -2.42388, -2.37485]	[-0.178715, 0.210692, 0.433228, 0.362036, 0.36309, 0.308746, 0.0660514, -0.174854, -0.209064, -0.203512]
2	[-0.489705, -0.490986, -0.482879, -0.477844, -0.461825, -0.405518, -0.372019, -0.368619, -0.388233, -0.383752]	[-1.5549, -1.56325, -1.56785, -1.56909, -1.56434, -1.52118, -1.48789, -1.53443, -1.59792, -1.54974]	[-0.920551, -0.93175, -0.936862, -0.944511, -0.953147, -0.959415, -0.97225, -0.970391, -0.989504, -0.974853]	[0.635676, 0.638875, 0.645802, 0.646596, 0.783325, 1.4228, 1.70323, 1.07318, 0.590275, 1.03331]	[-1.61973, -1.62598, -1.62972, -1.63145, -1.59547, -0.82092, 0.780895, 1.43152, 1.56559, 1.47921]	[-0.646413, -0.652875, -0.656965, -0.651708, -0.645447, -0.589263, -0.175096, 0.331561, 0.51838, 0.37889]
3	[-0.520749, -0.515206, -0.499155, -0.496912, -0.510951, -0.528283, -0.523068, -0.516337, -0.516789, -0.515288]	[-1.7238, -1.74921, -1.77769, -1.78882, -1.75818, -1.73677, -1.74756, -1.78149, -1.77938, -1.77618]	[-0.582373, -0.578241, -0.590816, -0.631495, -0.647523, -0.651549, -0.648375, -0.654838, -0.633536, -0.622737]	[0.450816, 0.478307, 0.774727, 1.02737, 0.78464, 0.703484, 0.840466, 0.848916, 0.61853, 0.489974]	[-1.75408, -1.75602, -1.47291, -0.138697, 0.972938, 1.08465, 0.667588, -0.729587, -1.66866, -1.77695]	[-0.776438, -0.776525, -1.12074, -1.63391, -1.14135, -0.960213, -1.35022, -1.54776, -0.947025, -0.732308]
4	[-0.565816, -0.562449, -0.562236, -0.562989, -0.543475, -0.532236, -0.533084, -0.53331, -0.533284, -0.532587]	[-1.90572, -1.89993, -1.89647, -1.90282, -1.84865, -1.79916, -1.80988, -1.83257, -1.88177, -1.91009]	[-0.750785, -0.74669, -0.74164, -0.741961, -0.724922, -0.708688, -0.702197, -0.695951, -0.714125, -0.726171]	[0.461241, 0.459243, 0.512718, 1.37894, 1.87828, 1.02844, 1.00621, 1.96535, 1.51937, 0.595882]	[-1.88771, -1.88345, -1.86923, -1.24722, 0.630626, 1.41282, 1.45273, 0.495731, -1.31749, -1.91297]	[-0.763371, -0.7594, -0.766832, -0.706061, -0.17522, -0.0489641, -0.11657, -0.235314, -0.562423, -0.694505]
5	[-0.625365, -0.62478, -0.626322, -0.638601, -0.58534, -0.542019, -0.59497, -0.608185, -0.609227, -0.635047]	[-1.84361, -1.83872, -1.83276, -1.84772, -1.72935, -1.73062, -1.79696, -1.72503, -1.76103, -1.88189]	[-0.779209, -0.772209, -0.771932, -0.82567, -0.874757, -0.880812, -0.855176, -0.840179, -0.842859, -0.824567]	[0.578219, 0.577887, 0.608473, 1.40044, 1.78643, 0.621851, 0.359037, 1.40534, 1.84719, 0.875334]	[-1.8312, -1.81922, -1.8094, -1.43453, 0.578385, 1.57481, 1.59355, 1.16925, -0.704143, -1.92775]	[-0.756457, -0.770078, -0.773788, -0.476888, 0.185892, 0.0419027, -0.109731, 0.139695, -0.105265, -0.634221]
6	[-0.5017, -0.502625,	[-2.16893, -2.14364,	[-1.08293, -1.06866,	[0.626992, 0.60828,	[-2.38105, -2.37943,	[-0.169543, -0.166098,

Row	X[Hand tip l] Array…	Y[Hand tip l] Array…	Z[Hand tip l] Array…	X[Hand tip r] Array…	Y[Hand tip r] Array…	Z[Hand tip r] Array…
	-0.503367, -0.471004, -0.419453, -0.402451, -0.403706, -0.423374, -0.447107, -0.466767]	-2.13725, -2.00944, -1.78124, -1.71999, -1.83412, -2.02508, -2.08005, -2.149]	-1.06542, -1.03596, -0.949373, -0.926886, -0.961193, -1.00315, -1.00412, -1.03209]	0.830428, 1.69707, 1.34372, 0.950025, 1.36353, 1.64604, 0.922088, 0.71129]	-2.2983, -0.988672, 0.875532, 1.23389, 0.721278, -1.12535, -2.28633, -2.44549]	0.00183614 0.399289, 0.448594, 0.309239, 0.305136, 0.310196, 0.212328, 0.0718968]
7	[-0.49201, -0.488344, -0.483127, -0.472051, -0.444949, -0.437359, -0.450799, -0.477055, -0.499801, -0.515454]	[-2.17422, -2.16156, -2.14621, -2.0967, -1.99033, -1.90827, -1.91822, -2.04961, -2.25151, -2.29598]	[-0.96807, -0.966641, -0.967986, -0.961942, -0.920407, -0.900009, -0.903213, -0.921315, -0.951938, -0.955296]	[0.575735, 0.605823, 1.16715, 1.87748, 1.38845, 1.1885, 1.38638, 1.89547, 1.4654, 0.829557]	[-2.36999, -2.33107, -2.03327, -0.367611, 0.729929, 0.835735, 0.658218, -0.524872, -2.12489, -2.4903]	[-0.17599, -0.144256, 0.155189, 0.525673, 0.51622, 0.434747, 0.488408, 0.538916, 0.305844, 0.0840963]
8	[-0.45879, -0.432315, -0.439952, -0.408236, -0.352797, -0.380338, -0.397032, -0.402198, -0.390485, -0.375181]	[-1.87081, -1.87505, -1.87234, -1.86466, -1.80721, -1.73796, -1.71578, -1.70684, -1.7205, -1.7962]	[-0.695322, -0.704867, -0.701925, -0.712508, -0.728154, -0.709393, -0.70851, -0.70076, -0.693678, -0.706881]	[0.522543, 0.517137, 0.552456, 1.13388, 1.91961, 1.51238, 1.05461, 1.26092, 1.87949, 1.7609]	[-1.88802, -1.89932, -1.90077, -1.62142, -0.103876, 1.04566, 1.31712, 1.22717, 0.366593, -1.09612]	[-0.718691, -0.718763, -0.698726, -0.783592, -0.810682, -0.435188, -0.314745, -0.446456, -0.674679, -0.724235]
9	[-0.56928, -0.565574, -0.569954, -0.575982, -0.608574, -0.615468, -0.573287, -0.581618, -0.573085, -0.572927]	[-1.78524, -1.78883, -1.78899, -1.78455, -1.77083, -1.73434, -1.71116, -1.70382, -1.71689, -1.7387]	[-0.632802, -0.63893, -0.64226, -0.64208, -0.627324, -0.609535, -0.602673, -0.596906, -0.595214, -0.596245]	[0.576295, 0.575658, 0.577578, 0.606472, 1.30923, 1.76698, 1.0738, 0.88752, 1.42591, 1.51915]	[-1.8199, -1.82026, -1.82229, -1.81573, -1.27671, 0.21299, 1.16409, 1.14555, 0.557396, -0.864472]	[-0.61453, -0.616358, -0.617909, -0.632956, -0.774777, -0.618792, -0.173941, -0.170232, -0.62764, -0.903184]
10	[-0.515774, -0.518321, -0.518788, -0.505651, -0.464753, -0.444865, -0.437798, -0.452522, -0.480842, -0.504954]	[-1.73725, -1.73062, -1.73317, -1.74312, -1.75999, -1.75117, -1.7448, -1.73029, -1.74591, -1.80751]	[-0.689742, -0.689256, -0.684004, -0.682732, -0.719373, -0.745056, -0.757458, -0.750336, -0.736565, -0.743162]	[0.532689, 0.541907, 0.541979, 0.779612, 1.8104, 1.91173, 1.47501, 1.71758, 2.07368, 1.9153]	[-1.79499, -1.78343, -1.79152, -1.78287, -0.883476, 0.604672, 1.1762, 0.998638, 0.129588, -1.12535]	[-0.703047, -0.699836, -0.698641, -0.631636, -0.45419, -0.198334, -0.118912, -0.23771, -0.358709, -0.448996]
11	[-0.627545, -0.615972, -0.609384, -0.599799, -0.602157, -0.6048,	[-1.9838, -1.98538, -1.9441, -1.87498, -1.8487, -1.84779,	[-0.751145, -0.763987, -0.76812, -0.731808, -0.716519, -0.712603,	[0.856225, 1.25459, 1.55259, 0.706536, 0.40093, 0.612677,	[-2.03373, -1.5449, 0.495725, 1.47787, 1.47538, 1.42904,	[-0.860697, -0.971815, -0.960895, -0.553449, -0.432449, -0.511147,

Row	X[Hand tip l]	Y[Hand tip l]	Z[Hand tip l]	X[Hand tip r]	Y[Hand tip r]	Z[Hand tip r]
	Array…	Array…	Array…	Array…	Array…	Array…
	-0.607613, -0.613584, -0.617851, -0.619508]	-1.87435, -1.92973, -1.97342, -1.97519]	-0.722911, -0.732238, -0.733759, -0.740397]	1.3455, 1.52789, 0.99686, 0.756776]	0.722227, -0.948349, -1.93153, -1.88639]	-0.966358, -1.13706, -0.879679, -0.762765]
12	[-0.630619, -0.633912, -0.64359, -0.569823, -0.520622, -0.570373, -0.581052, -0.594079, -0.59592, -0.621672]	[-1.64961, -1.64951, -1.63573, -1.57181, -1.61494, -1.64919, -1.63388, -1.64014, -1.66701, -1.6865]	[-0.824166, -0.830622, -0.845946, -0.877665, -0.926133, -0.909464, -0.885836, -0.867469, -0.834718, -0.812998]	[0.625067, 0.811823, 1.81693, 1.54755, 0.874367, 0.97882, 1.78935, 1.94519, 0.966987, 0.662808]	[-1.73957, -1.71566, -0.588143, 1.05948, 1.28049, 1.26524, 0.863765, -0.794015, -1.80679, -1.88118]	[-0.702955, -0.692936, -0.389187, 0.189794, 0.295625, 0.251317, 0.196087, -0.122127, -0.423752, -0.499098]
13	[-0.631842, -0.630124, -0.650948, -0.663173, -0.674231, -0.64216, -0.651306, -0.65166, -0.626698, -0.64567]	[-1.9687, -1.96269, -1.92603, -1.9263, -1.89756, -1.83041, -1.74282, -1.77122, -1.8432, -1.82594]	[-0.661238, -0.654904, -0.659684, -0.66026, -0.635695, -0.626698, -0.616492, -0.586879, -0.576312, -0.602956]	[0.630175, 0.629933, 0.633313, 1.03425, 2.05444, 1.56638, 1.01793, 1.60553, 1.74025, 0.883618]	[-2.01565, -2.01294, -2.0096, -1.79676, -0.189262, 1.13343, 1.27704, 0.802655, -0.836524, -1.85141]	[-0.718671, -0.717171, -0.717451, -0.821834, -0.495228, -0.0665926, -0.0791353, -0.382212, -0.711979, -0.628158]
⋮	⋮	⋮	⋮	⋮	⋮	⋮
349	[-0.621089, -1.0449, -1.89761, -0.751027, 0.71488, 0.58048, -0.499629, -0.722987, -0.634486, -0.624496]	[-1.97224, -1.99991, -1.68478, -1.17514, -0.189116, -0.369091, -1.54352, -2.06749, -2.01538, -2.00185]	[-0.755559, -0.707114, -0.602131, -1.24964, -0.981653, -1.02043, -1.1991, -0.731488, -0.700776, -0.705489]	[0.673708, 1.27972, 1.3514, 0.924717, 0.616268, 0.79357, 1.17613, 0.728068, 0.62562, 0.594113]	[-2.04491, -1.15343, 0.839272, 1.0693, 1.06356, 0.916949, -0.78875, -2.11422, -2.07714, -2.19599]	[-0.556859, -0.697077, -0.0376414, 0.118583, -0.220732, -0.525724, -1.0853, -0.641972, -0.556918, -0.553088]
350	[-0.592603, -0.644958, -0.714828, 0.0109829, 0.220784, 0.0286745, -0.586435, -0.59565, -0.579942, -0.577563]	[-1.6791, -1.65559, -1.28133, -0.522509, -0.26778, -0.627524, -1.54413, -1.66844, -1.67047, -1.67021]	[-0.406435, -0.437962, -0.857345, -1.07072, -0.861038, -1.03574, -0.670029, -0.346703, -0.368963, -0.374515]	[0.49643, 0.534467, 0.756529, 0.448627, 0.204009, 0.185264, 0.379267, 0.435372, 0.432218, 0.431474]	[-1.63401, -1.54634, -0.344291, 0.884682, 0.840474, 0.358094, -1.34456, -1.69257, -1.7048, -1.7032]	[-0.737128, -0.839457, -1.20095, -0.651531, -0.31886, -0.886655, -1.07123, -0.712738, -0.703423, -0.697242]
351	[-0.877644, -0.998648, -1.01135, 0.505568, 0.804174, 0.755874, 0.284785, -1.05657,	[-2.05772, -2.13632, -1.27547, -0.394201, -0.276477, -0.287154, -0.58105, -1.40029,	[-0.513711, -0.554783, -1.3324, -1.40401, -0.666086, -0.593085, -1.09237, -1.33981,	[0.589487, 0.884399, 1.60208, 0.651455, 0.209562, 0.145298, 0.614592, 1.36465,	[-2.04258, -2.04654, -0.378628, 1.00288, 0.775746, 0.789032, 0.585767, -1.1264,	[-0.47041, -0.29761, -0.169514, -0.153843, -0.0097968, 0.0351534, -0.287731, -0.462847,

Row	X[Hand tip l]	Y[Hand tip l]	Z[Hand tip l]	X[Hand tip r]	Y[Hand tip r]	Z[Hand tip r]
	Array…	Array…	Array…	Array…	Array…	Array…
	-0.874824, -0.698104]	-1.95797, -1.95394]	-0.694231, -0.711644]	0.715105, 0.601019]	-1.98809, -1.93976]	-0.428546, -0.63954]
352	[-0.592897, -0.594365, -0.543591, 0.0304071, 0.50631, 0.531192, 0.484547, -0.329009, -0.585854, -0.474409]	[-1.94523, -1.95243, -1.6631, -0.435107, 0.365465, 0.358099, -0.0695414, -1.70034, -1.99241, -1.95143]	[-0.709085, -0.727124, -0.903469, -1.25895, -0.924595, -0.940715, -1.0953, -0.811169, -0.566758, -0.760296]	[0.682443, 0.681739, 0.746074, 0.681111, 0.365984, 0.316107, 0.3465, 0.586969, 0.616176, 0.614762]	[-1.8371, -1.84451, -1.19749, 0.760556, 1.33244, 1.27185, 0.892469, -1.27799, -1.89337, -1.90061]	[-0.70236, -0.710797, -0.908023, -0.620313, -0.117529, -0.251424, -0.676503, -0.923287, -0.703015, -0.721104]
353	[-0.464131, -0.472453, 0.057958, 0.222224, -0.0170001, -0.238523, -1.34832, -0.936844, -0.641723, -0.447631]	[-1.66182, -1.49547, -0.714124, -0.408666, -0.444767, -0.508523, -1.28476, -1.82242, -1.86169, -1.80051]	[-1.01759, -1.29342, -1.61227, -0.946614, -0.745826, -0.882679, -1.32525, -0.730512, -0.948798, -1.06918]	[0.627022, 1.23101, 1.59713, 0.774379, 0.557754, 1.12653, 1.62131, 0.626525, 0.577916, 0.625397]	[-1.85923, -1.49325, 0.281258, 1.16438, 1.1825, 0.940152, -0.914969, -1.60244, -1.77432, -1.80361]	[-0.3602, -0.505338, -0.439313, -0.0947241, -0.112955, -0.143548, 0.316931, -0.139101, -0.262971, -0.31109]
354	[-0.504902, -0.462323, -0.186389, 0.341665, 0.366952, 0.052739, -0.636105, -0.602012, -0.581194, -0.576899]	[-1.89461, -1.86227, -1.34841, -0.175019, 0.121401, -0.92737, -1.93102, -1.95566, -1.96668, -1.99593]	[-0.670756, -0.736532, -1.07898, -1.3155, -0.979174, -1.08504, -0.696026, -0.641559, -0.706322, -0.70654]	[0.617654, 0.608126, 0.520928, 0.298708, 0.081962, 0.198168, 0.550441, 0.58417, 0.549558, 0.573895]	[-1.74288, -1.68431, -0.38712, 1.18138, 1.16441, 0.175368, -1.54509, -1.79407, -1.78324, -1.81635]	[-0.848677, -0.917827, -1.25423, -0.694945, -0.446276, -1.13442, -1.15183, -0.879169, -0.94173, -0.928524]
355	[-0.804495, -0.970834, 0.445412, 0.970814, 0.212064, -0.869522, -0.67696, -0.59732, -0.587643, -0.589344]	[-2.07498, -1.75156, -0.646479, 0.0447481, -0.523043, -1.65123, -2.05628, -2.00938, -2.01221, -2.0143]	[-0.723651, -1.09007, -1.26815, -0.807578, -1.33863, -1.26457, -0.733744, -0.742166, -0.755587, -0.766082]	[0.868569, 1.63956, 1.00085, 0.663644, 1.33199, 1.57372, 0.786366, 0.731459, 0.742744, 0.747623]	[-2.01239, -0.580455, 0.981621, 1.17123, 0.596439, -1.28156, -2.1046, -2.08664, -2.01255, -2.07569]	[-0.504961, -0.0351897, -0.0831729, -0.145221, -0.498779, -0.685788, -0.509209, -0.520414, -0.517038, -0.525465]
356	[-0.52069, -0.590235, -0.219191, 0.550082, 0.618486, 0.522613, -0.219627, -0.54662, -0.529615, -0.545576]	[-1.71633, -1.65114, -1.06446, -0.265112, 0.0323169, -0.220663, -1.23225, -1.78249, -1.69342, -1.66403]	[-0.508497, -0.699385, -1.20871, -1.10408, -0.931363, -1.09811, -1.03041, -0.556592, -0.522102, -0.526283]	[0.411321, 1.06068, 1.35426, 0.723047, 0.606123, 0.763531, 1.03972, 0.695936, 0.446751, 0.413049]	[-1.72581, -1.29421, 0.231613, 0.947112, 0.979502, 0.992799, -0.0676264, -1.57954, -1.72572, -1.74011]	[-0.712322, -0.839832, -0.949662, -0.589259, -0.504341, -0.783981, -1.30793, -0.885706, -0.711157, -0.719335]
357	[-0.450635, -0.460649,	[-1.92201, -1.91714,	[-1.08267, -1.08277,	[0.608518, 0.593674,	[-2.13841, -2.19632,	[-0.424368, -0.430631,

Row	X[Hand tip l]	Y[Hand tip l]	Z[Hand tip l]	X[Hand tip r]	Y[Hand tip r]	Z[Hand tip r]
	Array…	Array…	Array…	Array…	Array…	Array…
	-0.523194, -0.148305, 0.833755, 1.02523, 1.06324, 0.759938, -0.400821, -0.893672]	-1.90514, -1.22424, -0.334661, -0.193376, -0.160412, -0.576072, -1.41386, -1.81952]	-1.15257, -1.58571, -1.05491, -0.449945, -0.40653, -0.842404, -1.33734, -1.14481]	0.798462, 1.62198, 1.16085, 0.469548, 0.536862, 1.16147, 1.86881, 1.14911]	-2.16177, -0.991775, 0.739527, 0.922981, 0.979557, 0.838403, -0.455372, -1.94839]	-0.372836, -0.196953, -0.0406683, 0.137708, 0.0407201, -0.074632, 0.0317846, -0.0296955]
358	[-0.643509, -0.642832, -0.681704, -0.602261, 0.247589, 0.577098, 0.513836, 0.346947, -0.495814, -0.668934]	[-1.60557, -1.60981, -1.61058, -1.29489, -0.644471, -0.333255, -0.448054, -0.729249, -1.50573, -1.68703]	[-0.499002, -0.504605, -0.571579, -0.979253, -0.984472, -0.709164, -0.717957, -0.815709, -0.667301, -0.337071]	[0.499926, 0.503723, 0.616953, 1.24145, 0.921731, 0.401024, 0.312462, 0.270012, 0.454514, 0.533708]	[-1.61621, -1.6172, -1.5053, -0.253119, 0.885004, 0.826736, 0.787488, 0.377166, -1.21379, -1.67703]	[-0.677067, -0.673809, -0.739933, -0.802048, -0.30956, -0.109487, -0.178386, -0.529695, -0.875833, -0.622449]
359	[-0.474881, -0.468528, -0.463785, -0.824964, -0.806782, 0.294937, 0.513915, 0.232681, -0.577393, -0.705058]	[-1.70857, -1.7017, -1.67589, -1.60821, -0.929662, -0.193123, 0.0128294, -0.42005, -1.37775, -1.81295]	[-0.507244, -0.498243, -0.486741, -0.552751, -1.11043, -1.27671, -1.132, -1.29456, -0.940712, -0.365959]	[0.381915, 0.392676, 0.40139, 0.861814, 1.20212, 0.641535, 0.477999, 0.603016, 0.839808, 0.609323]	[-1.6299, -1.6065, -1.58939, -1.39375, 0.183108, 0.957773, 0.928694, 0.821823, -0.575078, -1.71543]	[-0.832337, -0.809786, -0.79043, -0.924113, -0.874998, -0.537489, -0.556103, -1.10424, -1.50916, -0.943081]
360	[-0.548503, -0.607269, -0.630892, 0.0779301, 0.765495, 0.783736, 0.0203699, -0.709682, -0.674494, -0.672193]	[-1.70321, -1.67196, -1.4457, -0.58088, -0.268049, -0.223267, -0.603229, -1.53005, -1.74832, -1.71875]	[-0.75859, -0.815361, -1.02025, -1.05648, -0.672829, -0.626758, -1.08714, -0.894006, -0.603934, -0.666984]	[0.600535, 1.1625, 1.42205, 0.737176, 0.594218, 0.653958, 0.92972, 0.825932, 0.548429, 0.578367]	[-1.8182, -1.46588, 0.046472, 0.836447, 0.817326, 0.784752, 0.239143, -1.36072, -1.83095, -1.81892]	[-0.438703, -0.483525, -0.509236, -0.18074, -0.109449, -0.101898, -0.533919, -0.686273, -0.384316, -0.378668]

In [35]:

```
features = [maximum, minimum]
X_small = scalarlogiset(X_df_small, features)
```

Out[35]:

```
SupportedLogiset with 1 support (13.36 MBs)
├ worldtype:                   Interval{Int64}
├ featvaltype:                 Float64
├ featuretype:                 SoleModels.AbstractUnivariateFeature
├ frametype:                   SoleLogics.FullDimensionalFrame{1, Interval
{Int64}}
├ # instances:                 360
├ usesfullmemo:                true
├[BASE] UniformFullDimensionalLogiset of channel size (10,) (13.19 MBs)
│ ├ size × eltype:              (10, 10, 360, 48) × Float64
│ └ features:                   48 -> SoleModels.AbstractUnivariateFeature
["max[V1]", "min[V1]", "max[V2]", "min[V2]", "...", "min[V22]", "max[V23]",
"min[V23]", "max[V24]", "min[V24]"]
└[SUPPORT 1] FullMemoset (0 memoized values, 174.42 KBs))
```

In [78]:

```
model = ModalDecisionTree(; relations = :IA, features = [minimum, maximum])
```

Out[78]:

```
ModalDecisionTree(
  max_depth = nothing,
  min_samples_leaf = 4,
  min_purity_increase = 0.002,
  max_purity_at_leaf = Inf,
  max_modal_depth = nothing,
  relations = :IA,
  features = nothing,
  conditions = Function[minimum, maximum],
  featvaltype = Float64,
  initconditions = nothing,
  downsize = ModalDecisionTrees.MLJInterface.var"#downsize#43"(),
  print_progress = false,
  rng = Random._GLOBAL_RNG(),
  display_depth = nothing,
  min_samples_split = nothing,
  n_subfeatures = identity,
  post_prune = false,
  merge_purity_threshold = nothing,
  feature_importance = :split)
```

In [37]:

```
using ModalDecisionTrees


# Bind a machine learning algorithm to logiset & labels
mach = machine(model, X_df_small, y)
# mach = machine(ModalDecisionTree(; relations = (x)->[globalrel, IARelation

# Train!
@time fit!(mach; rows=train_idxs);

# Compute accuracy
yhat = predict_mode(mach; rows=test_idxs)
MLJ.accuracy(yhat, y[test_idxs])
```

```
[ Info: Precomputing logiset...
[ Info: Training machine(ModalDecisionTree(max_depth = nothing, …), …).
 48.424346 seconds (107.45 M allocations: 6.261 GiB, 6.06% gc time, 93.42% c
ompilation time: 1% of which was recompilation)
```

```
Out[37]:  0.7986111111111112
```

```
In [38]: # Show the restricted MDT learned
         printmodel(report(mach).rawmodel_full; hidemodality = true)
```

```
{1} SimpleDecision(⟨G⟩min[V1] ≥ 0.1815317142857143)          All clear : 16/
72 (conf = 0.2222)
✔ {1} SimpleDecision(⟨G⟩min[V1] < -1.6262652857142856)       Lock wings : 1
4/34 (conf = 0.4118)
│✔ {1} SimpleDecision(⟨B⟩max[V4] < 0.3800692857142858)       Fold wings : 1
2/20 (conf = 0.6000)
││✔ Spread wings : 8/8 (conf = 1.0000)
││✘ Fold wings : 12/12 (conf = 1.0000)
│✘ Lock wings : 14/14 (conf = 1.0000)
✘ {1} SimpleDecision(⟨G⟩min[V5] ≥ 0.7817771428571428)        All clear : 16/
38 (conf = 0.4211)
 ✔ I have command : 13/13 (conf = 1.0000)
 ✘ {1} SimpleDecision(⟨G⟩min[V11] ≥ 0.26558299999999996)     All clear : 16/
25 (conf = 0.6400)
  ✔ Not clear : 5/5 (conf = 1.0000)
  ✘ {1} SimpleDecision(⟨G⟩min[V10] ≥ 1.073351875)            All clear : 16/
20 (conf = 0.8000)
   ✔ {1} SimpleDecision(⟨=⟩min[V3] ≥ -0.473409875)           All clear : 15/
16 (conf = 0.9375)
   │✔ All clear : 3/4 (conf = 0.7500)
   │✘ All clear : 12/12 (conf = 1.0000)
   ✘ Not clear : 3/4 (conf = 0.7500)
```

```
In [39]: # Show its *pure* version
         printmodel(report(mach).solemodel_full; show_metrics = true, hidemodality =
```

```
▣ (⟨G⟩(min[V1] ≥ 0.1815317142857143))
├✔ (⟨G⟩((min[V1] ≥ 0.1815317142857143) ∧ ⟨G⟩(min[V1] < -1.626265285714285
6)))
│├✔ (⟨G⟩((min[V1] ≥ 0.1815317142857143) ∧ ⟨G⟩((min[V1] < -1.626265285714285
6) ∧ ⟨B⟩(max[V4] < 0.3800692857142858))))
││├✔ Spread wings : (ninstances = 8, confidence = 1.0, coverage = 1.0)
││└✘ Fold wings : (ninstances = 12, confidence = 1.0, coverage = 1.0)
│└✘ Lock wings : (ninstances = 14, confidence = 1.0, coverage = 1.0)
└✘ (⟨G⟩(min[V5] ≥ 0.7817771428571428))
 ├✔ I have command : (ninstances = 13, confidence = 1.0, coverage = 1.0)
 └✘ (⟨G⟩(min[V11] ≥ 0.26558299999999996))
  ├✔ Not clear : (ninstances = 5, confidence = 1.0, coverage = 1.0)
  └✘ (⟨G⟩(min[V10] ≥ 1.073351875))
   ├✔ (⟨G⟩((min[V10] ≥ 1.073351875) ∧ (min[V3] ≥ -0.473409875)))
   │├✔ All clear : (ninstances = 4, confidence = 0.75, coverage = 1.0)
   │└✘ All clear : (ninstances = 12, confidence = 1.0, coverage = 1.0)
   └✘ Not clear : (ninstances = 4, confidence = 0.75, coverage = 1.0)
```

```
In [40]: simplified_restricted_tree = ModalDecisionTrees.prune(report(mach).rawmodel_

         puretree = ModalDecisionTrees.translate(simplified_restricted_tree)
         printmodel(puretree; threshold_digits = 2, use_feature_abbreviations = true,
         println("# Leaves: ", SoleModels.nsubmodels(puretree))
         println("# Classes: ", length(unique(y)))
```

```
▣ (⟨G⟩X[Hand tip l] ≥ 0.18)
├✔ (⟨G⟩(X[Hand tip l] ≥ 0.18 ∧ ⟨G⟩X[Hand tip l] ↓ -1.63))
│├✔ (⟨G⟩(X[Hand tip l] ≥ 0.18 ∧ ⟨G⟩(X[Hand tip l] ↓ -1.63 ∧ ⟨B⟩X[Hand tip r]
< 0.38)))
││├✔ Spread wings
││└✘ Fold wings
│└✘ Lock wings
└✘ (⟨G⟩Y[Hand tip r] ≥ 0.78)
 ├✔ I have command
 └✘ (⟨G⟩Y[Elbow r] ≥ 0.27)
  ├✔ Not clear
  └✘ (⟨G⟩X[Elbow r] ≥ 1.07)
   ├✔ All clear
   └✘ Not clear

# Leaves: 12
# Classes: 6
```

In [41]:
```
# Print leaf rules + their training performances
ruleset = listrules(puretree)
printmodel.(ruleset; show_metrics = true, threshold_digits = 2, use_feature_
```

```
▣ (⟨G⟩(V1 ≥ 0.18 ∧ ⟨G⟩(V1 ↓ -1.63 ∧ ⟨B⟩V4 < 0.38)))  ➡  Spread wings : (nins
tances = 8, confidence = 1.0, coverage = 0.11)
▣ ⟨G⟩(V1 ≥ 0.18 ∧ ⟨G⟩V1 ↓ -1.63) ∧ [G](V1 ≥ 0.18 → [G](V1 ↓ -1.63 → [B]V4 ⤒
0.38))  ➡  Fold wings : (ninstances = 12, confidence = 1.0, coverage = 0.17)
▣ ⟨G⟩V1 ≥ 0.18 ∧ [G](V1 ≥ 0.18 → [G]V1 ≥ -1.63)  ➡  Lock wings : (ninstance
s = 14, confidence = 1.0, coverage = 0.19)
▣ ⟨G⟩V5 ≥ 0.78 ∧ [G]V1 ↓ 0.18  ➡  I have command : (ninstances = 13, confide
nce = 1.0, coverage = 0.18)
▣ ⟨G⟩V11 ≥ 0.27 ∧ [G]V1 ↓ 0.18 ∧ [G]V5 ↓ 0.78  ➡  Not clear : (ninstances =
5, confidence = 1.0, coverage = 0.07)
▣ ⟨G⟩V10 ≥ 1.07 ∧ [G]V1 ↓ 0.18 ∧ [G]V5 ↓ 0.78 ∧ [G]V11 ↓ 0.27  ➡  All clear
: (ninstances = 16, confidence = 0.94, coverage = 0.22)
▣ [G]V1 ↓ 0.18 ∧ [G]V5 ↓ 0.78 ∧ [G]V11 ↓ 0.27 ∧ [G]V10 ↓ 1.07  ➡  Not clear
: (ninstances = 4, confidence = 0.75, coverage = 0.06)
```

In [79]:
```
# Visualize a rule
println("IF\n\t", SoleLogics.experimentals.formula2natlang(antecedent(rulese
println("THEN\n\t", consequent(ruleset[4]))
```

```
IF
        (∃ interval where (min[Y[Hand tip r]] ≥ 0.78)) and (∀ intervals (min
[X[Hand tip l]] < 0.18))
THEN
        ▣ I have command
```

In [43]:
```
for (i_rule, rule) in enumerate(ruleset)
    println()
    println("[$i_rule]")
    antd = antecedent(rule)
    println(SoleLogics.experimentals.formula2natlang(antd; threshold_digits
end
```

```
[1]
∃ interval where ((V1 ≥ 0.18) and (∃ interval where ((V1 ↓ -1.63) and (∃ pre
fix interval where (V4 < 0.38)))))


[2]
(∃ interval where ((V1 ≥ 0.18) and (∃ interval where (V1 ↓ -1.63)))) and (∀
intervals (whenever V1 ≥ 0.18 holds, also ∀ intervals (whenever V1 ↓ -1.63 h
olds, also ∀ prefix intervals (V4 ⤒ 0.38))))


[3]
(∃ interval where (V1 ≥ 0.18)) and (∀ intervals (whenever V1 ≥ 0.18 holds, a
lso ∀ intervals (V1 ≥ -1.63)))


[4]
(∃ interval where (V5 ≥ 0.78)) and (∀ intervals (V1 ↓ 0.18))


[5]
((∃ interval where (V11 ≥ 0.27)) and (∀ intervals (V1 ↓ 0.18))) and (∀ inter
vals (V5 ↓ 0.78))


[6]
(((∃ interval where (V10 ≥ 1.07)) and (∀ intervals (V1 ↓ 0.18))) and (∀ inte
rvals (V5 ↓ 0.78))) and (∀ intervals (V11 ↓ 0.27))


[7]
(((∀ intervals (V1 ↓ 0.18)) and (∀ intervals (V5 ↓ 0.78))) and (∀ intervals
(V11 ↓ 0.27))) and (∀ intervals (V10 ↓ 1.07))
```

Exercise 2: improving the natural language translation

Rule 4 and 5 talk about properties holding on either *any* or *all* intervals. There is probably a shorter way of phrasing such formulas (which, remember, are aliases for `⟨G⟩(min[*] >= *)`, `[G](min[*] < *)`, etc).

Improve the recursive function `formula2natlang` so that rules 4 and 5 are translated into a simpler natural language sentence. This can be done by adding methods for the function covering specific cases. The function is defined in the module module `SoleLogics.experimentals` and is extended in `SoleModels.experimentals` and, since `ScalarCondition`s do not exist at the purely logical level, such an extension should be done in `SoleModels.experimentals`.

Feel free to submit your solution by opening a pull request! 😃

Further note: formulas that only make use of modal connectives that `⟨G⟩` or `[G]`, as well as features that are `min[⋅]` or `max[⋅]` can probably be expressed via propositional logic formulas that talk about the minimum or the maximum of a variable throughout the whole series.

In [44]:

```
# Print rules + their *test* performances

# Sprinkle the model with the test instances!
# Note: I'm testing it on the original dataset, not the small one.
predictions, tree_test = report(mach).sprinkle(X_df[test_idxs,:], y[test_idx

# Extract ruleset and print its metrics
ruleset_test = listrules(tree_test)

# printmodel.(ruleset_test; show_metrics = true, threshold_digits = 2, varia
printmodel.(ruleset_test; use_feature_abbreviations = true, show_metrics = t
```

```
Applying tree... 100%|████████████████████████████████████████| Time: 0:00:02
▣ ((⟨G⟩(V1 ≥ 0.18 ∧ ⟨G⟩(V1 ↓ -1.63 ∧ ⟨B⟩V4 < 0.38)))  ➡  Spread wings : (nins
tances = 57, confidence = 0.91, coverage = 0.2)
▣ ⟨G⟩(V1 ≥ 0.18 ∧ ⟨G⟩V1 ↓ -1.63) ∧ [G](V1 ≥ 0.18 → [G](V1 ↓ -1.63 → [B]V4 ⤒
0.38))  ➡  Fold wings : (ninstances = 48, confidence = 0.9, coverage = 0.17)
▣ ⟨G⟩V1 ≥ 0.18 ∧ [G](V1 ≥ 0.18 → [G]V1 ≥ -1.63)  ➡  Lock wings : (ninstance
s = 41, confidence = 1.0, coverage = 0.14)
▣ ⟨G⟩V5 ≥ 0.78 ∧ [G]V1 ↓ 0.18  ➡  I have command : (ninstances = 56, confide
nce = 0.84, coverage = 0.19)
▣ ⟨G⟩V11 ≥ 0.27 ∧ [G]V1 ↓ 0.18 ∧ [G]V5 ↓ 0.78  ➡  Not clear : (ninstances =
14, confidence = 1.0, coverage = 0.05)
▣ ⟨G⟩V10 ≥ 1.07 ∧ [G]V1 ↓ 0.18 ∧ [G]V5 ↓ 0.78 ∧ [G]V11 ↓ 0.27  ➡  All clear
: (ninstances = 47, confidence = 0.51, coverage = 0.16)
▣ [G]V1 ↓ 0.18 ∧ [G]V5 ↓ 0.78 ∧ [G]V11 ↓ 0.27 ∧ [G]V10 ↓ 1.07  ➡  Not clear
: (ninstances = 25, confidence = 0.48, coverage = 0.09)
```

Exercise 3: tree transplanting

The above cell shows that the last two rules extracted from the learned tree have low confidences. Furthermore, these were extracted from the rightmost branch in the decision tree. This shows, that the *Modal CART* algorithm, once it reached that branch node, was not able to find a sufficiently good condition of the kind $\langle G \rangle p$ for telling apart the instances that reached this branch.

Note that it is, however, possible that a different splitting condition existed, and that the algorithm might have just failed; it is heuristic, and not not optimal, afterall!

In such cases, we can gather the training and testing instances that reached the leaves, use another symbolic learning algorithm for extracting symbolic model (e.g., rules, decision trees/lists, etc.), and, if its performances are good enough, ultimately substitute the affected branch with it.

Use the `randdecisiontree` routine above, or the `ModalDecisionTree` itself (perhaps with different learning hyperparameter values) for re-learning, and transplanting a new tree for these instances?

The cell below computes the indices of the training and test instances to use. Additionally, it shows that the class distribution of these instances is between the classes "All clear" and "Not clear", which only differ in the orientation of the thumb (see pictures here). Would it help the algorithm if we, say, create a new variable that is the difference between *Y[Thumb r]* and *Y[Hand tip r]*?

In [45]:

```julia
affected_rules = ruleset[[end-1,end]]

# Take the problematic rules and retrain a tree on their covered instances
check_mask = fill(false, ninstances(X))
for r in affected_rules
    check_mask = (|).(check_mask, check(first(values(modforms(antecedent(r)))
end
train_idxs2 = filter(i->check_mask[i], train_idxs)
test_idxs2 = filter(i->check_mask[i], test_idxs)

println("# Training instances:  $(length(train_idxs2))")
println(countmap(y[train_idxs2]))
println("# Testing instances:  $(length(test_idxs2))")
println(countmap(y[test_idxs2]))
```

```
# Training instances:  17
Dict{CategoricalArrays.CategoricalValue{String, UInt32}, Int64}("Not clear"
=> 4, "All clear" => 13)
# Testing instances:  72
Dict{CategoricalArrays.CategoricalValue{String, UInt32}, Int64}("Not clear"
=> 35, "All clear" => 37)
```

In [46]:

```julia
# # TODO
# # Change the initial condition for the learning:
# # learn a formula to be interpreted at the *smallest, centered interval* c
# mach2 = machine(ModalDecisionTree(; initconditions=:start_at_center, relat

# # Train model
# @time fit!(mach2; rows=train_idxs2)

# predictions, tree2_test = report(mach2).sprinkle(X_df[test_idxs2,:], y[tes

# # Extract ruleset, join rules with same outcome (by means of v), and print
# ruleset2_test = listrules(tree2_test)
# ruleset2_test = joinrules(ruleset2_test)
# printmodel.(ruleset2_test; show_metrics = true, threshold_digits = 2, vari
```

Exercise 4: Land Cover Classification

`ModalDecisionTrees.jl` can also handle images! In which case, they use a 2D logic of rectangles instead of a 1D logic of intervals.

Apply ModalDecisionTrees to Indian Pines, a benchmark dataset for Land Cover Classification with 16 classes. The dataset consists of a hyperspectral image (i.e., 200 color channels instead of the typical 3 RGB channels) where many pixels have been labelled as belonging to one of the classes.

Sketch of the idea:

- Load the image and the ground truths. The package MAT.jl can be helpful;
- From the 145×145 image provided, sample a (small) number m of 3×3 patches for each class, and label each patch with the class label for the central pixel;
- Gather the ground truths into a vector of strings `y`, and the samples into a Julia `DataFrame` `X` with 200 columns, $16m$ rows and 3×3 *matrices* in the cells;
- Use MLJ to train a ModalDecisionTree on `X` and `y`, similarly to the above case.

Suggestion: since the formulas are desirably rotation-invariant, ask the algorithm to use *topological* relations instead of *directional* relations. Rrefer to the doc to know more.