Package 'Rmach'

August 6, 2024

Title Provides machine learning algorythm

Version 2.0.0.0

Description Provides these algorythms: coefficient finder for regression functions

License GPL (==3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports stringr

Contents

`best_model` *best_model*

Description

Returns the best input models. The coefficient of the best model can be found with the poly_model function

Usage

```
best_model(
  inpt_datf,
  Degree,
  Coeff_v = NA,
  Powers = NA,
  Mth_symb,
  Numrtr_v = NA
)
```

Arguments

inpt_datf	is the input dataframe, first column for the x values and second column for the y values
Degree	is a vector containing all the degrees. Each degree represents how many coefficients the model has.
Coeff_v	is a list containing the vector containing the coefficients for each model. The first value of each coefficient vector is always the constant, so it is not linked to any math symbol
Powers	is a list containing all the values associated with the math symbols of mth_symb list for each model. Because you can have multiple models in the function, so Powers is separated with the "-" separator between the different powers values for each model like in the examples
Mth_symb	is a list containing the vector of the different math symbols linked to the coefficients from the second value
Numrtr_v	is a list containing the different numerator values for each math symbol for each model, see examples

Examples

```
print(best_model(inpt_datf=data.frame(mtcars$wt, mtcars$mpg), Degree=c(2, 2), Coeff_v=c("

[1] 2

print(best_model(inpt_datf=data.frame(mtcars$wt, mtcars$mpg), Degree=c(2, 2), Coeff_v=c("

[1] 1

print(best_model(inpt_datf=data.frame(mtcars$wt, mtcars$mpg), Degree=c(2, 2), Coeff_v=c("

[1] 1

print(best_model(inpt_datf=data.frame(mtcars$wt, mtcars$mpg), Degree=c(2, 2), Coeff_v=c("

#' [1] 1
```

calcall *calcall*

Description

Takes a formula as a character as an input and makes the calculation. Accepts also variables, in this case the part of the formula that contains the variable wont be calculated, but the others part will be as usual.

Usage

```
calcall(inpt)
```

Arguments

inpt	is the input formula as a character

Examples

```
print(calcall(inpt="ze+(yu*((fgf)-(-12+8-Y+4-T+4+97+a)+tt))"))

[1] "ze+(yu*(fgf-(-4-Y+4-T+101+a)+tt))"

print(calcall(inpt="ze+(yu*((fgf)-(-12+8-7+3-67+4+97+1)+tt))"))

[1] "ze+(yu*(fgf-27+tt))"

print(calcall(inpt="ze+(yu*((fgf)+(12*3/2+4)+tt))"))

[1] "ze+(yu*(fgf+22+tt))"

print(calcall(inpt="1+3*2+(-2/-3*-3*((fgf)-(--12-6)+2))+5-3*5"))

[1] "7+(-2*(fgf-4))+20"

print(calcall(inpt="1+3*2+(-2/-3*-3*((fgf)-(--12-6)+2))+(-log_e_1_e_2+t+2^3)+m-log_e_1_e_

[1] "7+(-2*(fgf-4))+(-2+t+8)+m+6-m-12+(e_ii-8+log_im_4-67)-4+(y+2)"

print(calcall("(6+4*-(4-5))+3/3"))

[1] "11"

print(calcall(inpt="1+3*2+(-2/-3*-3*((fgf)-(--12-6)+2))+(-log_e_1_e_2+t+2^3)+m-log_e_1_e_

[1] "7+(-2*(fgf-4))+(-2+t+8)+m+6-m-16"

print(calcall(inpt="(log_5_Z-2-6+5)+-6+2"))

[1] "(log_5_Z-3)-4"

print(calcall(inpt="m--2+-5"))

[1] "m-3"
```

```
print(calcall(inpt="(-2-6)+-6+2"))

[1] "-12"

print(calcall(inpt="m-6"))

[1] "m-6"

print(calcall(inpt="--6"))

[1] "6"
```

`calcall_var` *calcall_var*

Description

Does the same thing as calcall function but calculates the formula that have variables. The values of the variables have to be given in a list of vectors, see examples.

Usage

```
calcall_var(inpt, var_name_v, var_val_l)
```

Arguments

`inpt`	is the input formula, with the variables
`var_name_v`	is the vector that contains the variables name in the order of apparition in the formula. If the variable appears multiple times in the formula, it has to be specified in this vector, see examples.
`var_val_l`	is the list containing the vectors containing the values of each variable, for each point you want to calculate. The vectors has to be given in the same order has the variable in var_name_v.

Examples

```
print(calcall_var(inpt="(6+m*-(4-imp))+3/jp", var_name_v=c("m", "imp", "jp"),
                  var_val_l=list(

                                  c(1:6),

                                  c(3, 4, 2, 5, 6, 1),

                                  c(6:1))))

 [1] "5.5"  "6.6"  "0.75" "11"   "17.5" "-9"

print(calcall_var(inpt="(6+m*-(4-imp))+3/jp+jp", var_name_v=c("m", "imp", "jp", "jp"),
                   var_val_l=list(

                                   c(1:6),
```

```
                                    c(3, 4, 2, 5, 6, 1),
                                    c(6:1))))
[1] "11.5" "11.6" "4.75" "14" "19.5" "-8"
```

`datf_folder` *datf_folder*

Description

Folds a dataframe, see examples.

Usage

```
datf_folder(inpt_datf)
```

Arguments

`inpt_datf`	is the input dataframe

Examples

```
print(datf_folder(inpt_datf = iris))
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	6.9	3.1	4.9	1.5	versicolor
3	4.7	3.2	1.3	0.2	setosa
4	5.1	3.5	1.4	0.3	setosa
5	7.2	3.0	5.8	1.6	virginica
6	5.8	2.7	5.1	1.9	virginica
7	5.4	3.0	4.5	1.5	versicolor
8	6.7	3.1	5.6	2.4	virginica
9	6.0	3.0	4.8	1.8	virginica
10	5.4	3.4	1.5	0.4	setosa
11	6.9	3.1	5.4	2.1	virginica
12	5.8	2.7	5.1	1.9	virginica
13	6.4	3.1	5.5	1.8	virginica
14	5.7	2.6	3.5	1.0	versicolor
15	5.4	3.9	1.7	0.4	setosa
16	5.7	2.8	4.1	1.3	versicolor
17	5.1	3.7	1.5	0.4	setosa
18	4.4	3.0	1.3	0.2	setosa
19	5.7	3.8	1.7	0.3	setosa
20	5.1	3.8	1.5	0.3	setosa
21	5.4	3.4	1.7	0.2	setosa
22	6.7	3.1	4.7	1.5	versicolor
23	6.0	3.4	4.5	1.6	versicolor
24	6.9	3.1	4.9	1.5	versicolor
25	4.8	3.4	1.9	0.2	setosa
26	5.8	2.7	5.1	1.9	virginica

27	5.0	3.4	1.6	0.4	setosa
28	5.8	2.8	5.1	2.4	virginica
29	6.3	2.3	4.4	1.3	versicolor
30	4.7	3.2	1.6	0.2	setosa
31	4.8	3.0	1.4	0.3	setosa
32	5.4	3.4	1.5	0.4	setosa
33	6.1	2.6	5.6	1.4	virginica
34	6.1	3.0	4.6	1.4	versicolor
35	6.0	2.2	4.0	1.0	versicolor
36	5.0	3.2	1.2	0.2	setosa
37	5.5	3.5	1.3	0.2	setosa
38	5.8	2.8	5.1	2.4	virginica
39	6.2	3.4	5.4	2.3	virginica
40	5.1	3.4	1.5	0.2	setosa
41	5.0	3.5	1.3	0.3	setosa
42	4.5	2.3	1.3	0.3	setosa
43	4.9	3.6	1.4	0.1	setosa
44	5.0	3.5	1.6	0.6	setosa
45	5.7	3.0	4.2	1.2	versicolor
46	6.4	2.8	5.6	2.1	virginica
47	6.2	3.4	5.4	2.3	virginica
48	4.6	3.2	1.4	0.2	setosa
49	6.4	3.2	5.3	2.3	virginica
50	5.5	4.2	1.4	0.2	setosa
51	7.7	3.0	6.1	2.3	virginica
52	5.9	3.0	4.2	1.5	versicolor
53	6.5	3.0	5.5	1.8	virginica
54	5.4	3.9	1.7	0.4	setosa
55	6.5	2.8	4.6	1.5	versicolor
56	5.8	2.6	4.0	1.2	versicolor
57	5.7	2.8	4.5	1.3	versicolor
58	4.9	2.4	3.3	1.0	versicolor
59	6.7	3.1	5.6	2.4	virginica
60	6.1	3.0	4.9	1.8	virginica
61	5.8	2.8	5.1	2.4	virginica
62	5.9	3.0	4.2	1.5	versicolor
63	5.2	4.1	1.5	0.1	setosa
64	6.9	3.1	4.9	1.5	versicolor
65	5.6	2.9	3.6	1.3	versicolor
66	5.4	3.4	1.7	0.2	setosa
67	5.6	3.0	4.5	1.5	versicolor
68	5.8	2.7	4.1	1.0	versicolor
69	6.2	2.2	4.5	1.5	versicolor
70	6.2	2.2	4.5	1.5	versicolor
71	5.9	3.2	4.8	1.8	versicolor
72	6.1	2.8	4.0	1.3	versicolor
73	6.3	2.5	4.9	1.5	versicolor
74	5.0	3.0	1.6	0.2	setosa
75	4.6	3.4	1.4	0.3	setosa
76	6.4	3.2	5.3	2.3	virginica
77	6.7	3.1	4.7	1.5	versicolor
78	5.5	4.2	1.4	0.2	setosa
79	6.0	2.9	4.5	1.5	versicolor
80	5.4	3.9	1.7	0.4	setosa
81	5.5	3.5	1.3	0.2	setosa
82	6.3	3.3	6.0	2.5	virginica
83	5.8	2.7	3.9	1.2	versicolor

84	6.0	2.7	5.1	1.6	versicolor
85	6.8	2.8	4.8	1.4	versicolor
86	6.1	3.0	4.6	1.4	versicolor
87	6.7	3.1	4.7	1.5	versicolor
88	5.1	3.8	1.6	0.2	setosa
89	6.8	2.8	4.8	1.4	versicolor
90	6.9	3.2	5.7	2.3	virginica
91	6.0	3.4	4.5	1.6	versicolor
92	6.1	3.0	4.6	1.4	versicolor
93	5.8	2.6	4.0	1.2	versicolor
94	5.0	2.3	3.3	1.0	versicolor
95	5.7	3.0	4.2	1.2	versicolor
96	5.7	3.0	4.2	1.2	versicolor
97	5.7	2.9	4.2	1.3	versicolor
98	6.4	2.8	5.6	2.2	virginica
99	5.1	3.4	1.5	0.2	setosa
100	5.7	2.8	4.1	1.3	versicolor
101	6.5	2.8	4.6	1.5	versicolor
102	4.8	3.4	1.9	0.2	setosa
103	4.4	2.9	1.4	0.2	setosa
104	5.1	2.5	3.0	1.1	versicolor
105	7.4	2.8	6.1	1.9	virginica
106	7.6	3.0	6.6	2.1	virginica
107	4.9	2.5	4.5	1.7	virginica
108	7.3	2.9	6.3	1.8	virginica
109	4.8	3.4	1.9	0.2	setosa
110	5.7	4.4	1.5	0.4	setosa
111	6.5	3.2	5.1	2.0	virginica
112	6.9	3.2	5.7	2.3	virginica
113	5.9	3.2	4.8	1.8	versicolor
114	7.1	3.0	5.9	2.1	virginica
115	5.8	2.8	5.1	2.4	virginica
116	4.8	3.4	1.9	0.2	setosa
117	4.3	3.0	1.1	0.1	setosa
118	6.6	2.9	4.6	1.3	versicolor
119	5.1	2.5	3.0	1.1	versicolor
120	6.0	2.2	5.0	1.5	virginica
121	5.1	3.4	1.5	0.2	setosa
122	6.3	2.7	4.9	1.8	virginica
123	6.7	3.3	5.7	2.1	virginica
124	6.1	2.6	5.6	1.4	virginica
125	5.0	3.3	1.4	0.2	setosa
126	7.2	3.2	6.0	1.8	virginica
127	6.2	2.8	4.8	1.8	virginica
128	6.1	3.0	4.9	1.8	virginica
129	5.0	3.4	1.6	0.4	setosa
130	6.2	2.2	4.5	1.5	versicolor
131	7.4	2.8	6.1	1.9	virginica
132	6.6	2.9	4.6	1.3	versicolor
133	6.7	3.3	5.7	2.1	virginica
134	6.3	3.3	4.7	1.6	versicolor
135	5.7	2.9	4.2	1.3	versicolor
136	7.2	3.6	6.1	2.5	virginica
137	6.5	3.0	5.5	1.8	virginica
138	6.4	3.1	5.5	1.8	virginica
139	5.5	4.2	1.4	0.2	setosa
140	5.8	2.7	5.1	1.9	virginica

```
141          5.0         2.0          3.5         1.0 versicolor
142          6.9         3.1          5.1         2.3  virginica
143          5.8         2.7          5.1         1.9  virginica
144          6.8         3.2          5.9         2.3  virginica
145          6.7         3.3          5.7         2.5  virginica
146          5.1         3.3          1.7         0.5     setosa
147          5.1         3.8          1.9         0.4     setosa
148          6.5         3.0          5.2         2.0  virginica
149          4.6         3.6          1.0         0.2     setosa
150          5.9         3.0          5.1         1.8  virginica
```

`individual_cloning` *individual_cloning*

Description

Allow to generate individuals with the same label as those existig and having as values at variables, a value generated with a normal distribution having as parameters the mean for the variable A for the individual I and the same goes for the standard deviation, see examples.

Usage

```
individual_cloning(inpt_datf, col_vars = c(), label_var, hmn)
```

Arguments

`inpt_datf`	is the input dataset as a dataframe
`col_vars`	is a vector containing the colnames or the column numbers of the variables
`label_var`	is a either the colnames or the column number of the label variable
`hmn`	is how many of new individual from the same label will be generated

Examples

```
datf <- iris
datf[, 5] <- as.character(datf[, 5])
datf <- individual_cloning(inpt_datf = datf, col_vars = c(1:4), label_var = 5, hmn = 3)
print(datf)
nrow(datf)
nrow(iris)
```

```
     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1        5.100000    3.500000     1.400000   0.2000000     setosa
2        4.900000    3.000000     1.400000   0.2000000     setosa
3        4.700000    3.200000     1.300000   0.2000000     setosa
4        4.600000    3.100000     1.500000   0.2000000     setosa
5        5.000000    3.600000     1.400000   0.2000000     setosa
6        5.400000    3.900000     1.700000   0.4000000     setosa
7        4.600000    3.400000     1.400000   0.3000000     setosa
8        5.000000    3.400000     1.500000   0.2000000     setosa
9        4.400000    2.900000     1.400000   0.2000000     setosa
10       4.900000    3.100000     1.500000   0.1000000     setosa
11       5.400000    3.700000     1.500000   0.2000000     setosa
```

12	4.800000	3.400000	1.600000	0.2000000	setosa
13	4.800000	3.000000	1.400000	0.1000000	setosa
14	4.300000	3.000000	1.100000	0.1000000	setosa
15	5.800000	4.000000	1.200000	0.2000000	setosa
16	5.700000	4.400000	1.500000	0.4000000	setosa
17	5.400000	3.900000	1.300000	0.4000000	setosa
18	5.100000	3.500000	1.400000	0.3000000	setosa
19	5.700000	3.800000	1.700000	0.3000000	setosa
20	5.100000	3.800000	1.500000	0.3000000	setosa
21	5.400000	3.400000	1.700000	0.2000000	setosa
22	5.100000	3.700000	1.500000	0.4000000	setosa
23	4.600000	3.600000	1.000000	0.2000000	setosa
24	5.100000	3.300000	1.700000	0.5000000	setosa
25	4.800000	3.400000	1.900000	0.2000000	setosa
26	5.000000	3.000000	1.600000	0.2000000	setosa
27	5.000000	3.400000	1.600000	0.4000000	setosa
28	5.200000	3.500000	1.500000	0.2000000	setosa
29	5.200000	3.400000	1.400000	0.2000000	setosa
30	4.700000	3.200000	1.600000	0.2000000	setosa
31	4.800000	3.100000	1.600000	0.2000000	setosa
32	5.400000	3.400000	1.500000	0.4000000	setosa
33	5.200000	4.100000	1.500000	0.1000000	setosa
34	5.500000	4.200000	1.400000	0.2000000	setosa
35	4.900000	3.100000	1.500000	0.2000000	setosa
36	5.000000	3.200000	1.200000	0.2000000	setosa
37	5.500000	3.500000	1.300000	0.2000000	setosa
38	4.900000	3.600000	1.400000	0.1000000	setosa
39	4.400000	3.000000	1.300000	0.2000000	setosa
40	5.100000	3.400000	1.500000	0.2000000	setosa
41	5.000000	3.500000	1.300000	0.3000000	setosa
42	4.500000	2.300000	1.300000	0.3000000	setosa
43	4.400000	3.200000	1.300000	0.2000000	setosa
44	5.000000	3.500000	1.600000	0.6000000	setosa
45	5.100000	3.800000	1.900000	0.4000000	setosa
46	4.800000	3.000000	1.400000	0.3000000	setosa
47	5.100000	3.800000	1.600000	0.2000000	setosa
48	4.600000	3.200000	1.400000	0.2000000	setosa
49	5.300000	3.700000	1.500000	0.2000000	setosa
50	5.000000	3.300000	1.400000	0.2000000	setosa
51	7.000000	3.200000	4.700000	1.4000000	versicolor
52	6.400000	3.200000	4.500000	1.5000000	versicolor
53	6.900000	3.100000	4.900000	1.5000000	versicolor
54	5.500000	2.300000	4.000000	1.3000000	versicolor
55	6.500000	2.800000	4.600000	1.5000000	versicolor
56	5.700000	2.800000	4.500000	1.3000000	versicolor
57	6.300000	3.300000	4.700000	1.6000000	versicolor
58	4.900000	2.400000	3.300000	1.0000000	versicolor
59	6.600000	2.900000	4.600000	1.3000000	versicolor
60	5.200000	2.700000	3.900000	1.4000000	versicolor
61	5.000000	2.000000	3.500000	1.0000000	versicolor
62	5.900000	3.000000	4.200000	1.5000000	versicolor
63	6.000000	2.200000	4.000000	1.0000000	versicolor
64	6.100000	2.900000	4.700000	1.4000000	versicolor
65	5.600000	2.900000	3.600000	1.3000000	versicolor
66	6.700000	3.100000	4.400000	1.4000000	versicolor
67	5.600000	3.000000	4.500000	1.5000000	versicolor
68	5.800000	2.700000	4.100000	1.0000000	versicolor

```
69       6.200000    2.200000     4.500000   1.5000000 versicolor
70       5.600000    2.500000     3.900000   1.1000000 versicolor
71       5.900000    3.200000     4.800000   1.8000000 versicolor
72       6.100000    2.800000     4.000000   1.3000000 versicolor
73       6.300000    2.500000     4.900000   1.5000000 versicolor
74       6.100000    2.800000     4.700000   1.2000000 versicolor
75       6.400000    2.900000     4.300000   1.3000000 versicolor
76       6.600000    3.000000     4.400000   1.4000000 versicolor
77       6.800000    2.800000     4.800000   1.4000000 versicolor
78       6.700000    3.000000     5.000000   1.7000000 versicolor
79       6.000000    2.900000     4.500000   1.5000000 versicolor
80       5.700000    2.600000     3.500000   1.0000000 versicolor
81       5.500000    2.400000     3.800000   1.1000000 versicolor
82       5.500000    2.400000     3.700000   1.0000000 versicolor
83       5.800000    2.700000     3.900000   1.2000000 versicolor
84       6.000000    2.700000     5.100000   1.6000000 versicolor
85       5.400000    3.000000     4.500000   1.5000000 versicolor
86       6.000000    3.400000     4.500000   1.6000000 versicolor
87       6.700000    3.100000     4.700000   1.5000000 versicolor
88       6.300000    2.300000     4.400000   1.3000000 versicolor
89       5.600000    3.000000     4.100000   1.3000000 versicolor
90       5.500000    2.500000     4.000000   1.3000000 versicolor
91       5.500000    2.600000     4.400000   1.2000000 versicolor
92       6.100000    3.000000     4.600000   1.4000000 versicolor
93       5.800000    2.600000     4.000000   1.2000000 versicolor
94       5.000000    2.300000     3.300000   1.0000000 versicolor
95       5.600000    2.700000     4.200000   1.3000000 versicolor
96       5.700000    3.000000     4.200000   1.2000000 versicolor
97       5.700000    2.900000     4.200000   1.3000000 versicolor
98       6.200000    2.900000     4.300000   1.3000000 versicolor
99       5.100000    2.500000     3.000000   1.1000000 versicolor
100      5.700000    2.800000     4.100000   1.3000000 versicolor
101      6.300000    3.300000     6.000000   2.5000000  virginica
102      5.800000    2.700000     5.100000   1.9000000  virginica
103      7.100000    3.000000     5.900000   2.1000000  virginica
104      6.300000    2.900000     5.600000   1.8000000  virginica
105      6.500000    3.000000     5.800000   2.2000000  virginica
106      7.600000    3.000000     6.600000   2.1000000  virginica
107      4.900000    2.500000     4.500000   1.7000000  virginica
108      7.300000    2.900000     6.300000   1.8000000  virginica
109      6.700000    2.500000     5.800000   1.8000000  virginica
110      7.200000    3.600000     6.100000   2.5000000  virginica
111      6.500000    3.200000     5.100000   2.0000000  virginica
112      6.400000    2.700000     5.300000   1.9000000  virginica
113      6.800000    3.000000     5.500000   2.1000000  virginica
114      5.700000    2.500000     5.000000   2.0000000  virginica
115      5.800000    2.800000     5.100000   2.4000000  virginica
116      6.400000    3.200000     5.300000   2.3000000  virginica
117      6.500000    3.000000     5.500000   1.8000000  virginica
118      7.700000    3.800000     6.700000   2.2000000  virginica
119      7.700000    2.600000     6.900000   2.3000000  virginica
120      6.000000    2.200000     5.000000   1.5000000  virginica
121      6.900000    3.200000     5.700000   2.3000000  virginica
122      5.600000    2.800000     4.900000   2.0000000  virginica
123      7.700000    2.800000     6.700000   2.0000000  virginica
124      6.300000    2.700000     4.900000   1.8000000  virginica
125      6.700000    3.300000     5.700000   2.1000000  virginica
```

```
126      7.200000    3.200000     6.000000   1.8000000  virginica
127      6.200000    2.800000     4.800000   1.8000000  virginica
128      6.100000    3.000000     4.900000   1.8000000  virginica
129      6.400000    2.800000     5.600000   2.1000000  virginica
130      7.200000    3.000000     5.800000   1.6000000  virginica
131      7.400000    2.800000     6.100000   1.9000000  virginica
132      7.900000    3.800000     6.400000   2.0000000  virginica
133      6.400000    2.800000     5.600000   2.2000000  virginica
134      6.300000    2.800000     5.100000   1.5000000  virginica
135      6.100000    2.600000     5.600000   1.4000000  virginica
136      7.700000    3.000000     6.100000   2.3000000  virginica
137      6.300000    3.400000     5.600000   2.4000000  virginica
138      6.400000    3.100000     5.500000   1.8000000  virginica
139      6.000000    3.000000     4.800000   1.8000000  virginica
140      6.900000    3.100000     5.400000   2.1000000  virginica
141      6.700000    3.100000     5.600000   2.4000000  virginica
142      6.900000    3.100000     5.100000   2.3000000  virginica
143      5.800000    2.700000     5.100000   1.9000000  virginica
144      6.800000    3.200000     5.900000   2.3000000  virginica
145      6.700000    3.300000     5.700000   2.5000000  virginica
146      6.700000    3.000000     5.200000   2.3000000  virginica
147      6.300000    2.500000     5.000000   1.9000000  virginica
148      6.500000    3.000000     5.200000   2.0000000  virginica
149      6.200000    3.400000     5.400000   2.3000000  virginica
150      5.900000    3.000000     5.100000   1.8000000  virginica
151      4.601009    3.727368     1.268078   0.3122136     setosa
210      4.613076    3.989209     1.555392   0.2953775     setosa
310      4.722235    3.602591     1.479940   0.2471369     setosa
513      5.660667    2.449398     4.241485   1.5317590 versicolor
511      5.987887    3.016099     3.690411   1.5357972 versicolor
512      5.803584    2.828602     4.024589   1.2767213 versicolor
1013     6.851160    3.287923     5.157840   1.5365199  virginica
1011     7.119751    3.460045     4.990113   1.2895762  virginica
1012     7.370573    3.140464     5.680828   1.8674812  virginica
[1] 159
[1] 150
```

`individual_equalizer_max`
individual_equalizer_max

Description

Remove the individual that are in exess according to a given value, see examples

Usage

```
individual_equalizer_max(inpt_datf, label_var, hmn)
```

Examples

```
print(individual_equalizer_max(inpt_datf = datf, label_var = 5, hmn = 15))
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.0	3.2	1.2	0.2	setosa
2	5.5	3.5	1.3	0.2	setosa
3	4.9	3.6	1.4	0.1	setosa
4	4.4	3.0	1.3	0.2	setosa
5	5.1	3.4	1.5	0.2	setosa
6	5.0	3.5	1.3	0.3	setosa
7	4.5	2.3	1.3	0.3	setosa
8	4.4	3.2	1.3	0.2	setosa
9	5.0	3.5	1.6	0.6	setosa
10	5.1	3.8	1.9	0.4	setosa
11	4.8	3.0	1.4	0.3	setosa
12	5.1	3.8	1.6	0.2	setosa
13	4.6	3.2	1.4	0.2	setosa
14	5.3	3.7	1.5	0.2	setosa
15	5.0	3.3	1.4	0.2	setosa
16	6.0	3.4	4.5	1.6	versicolor
17	6.7	3.1	4.7	1.5	versicolor
18	6.3	2.3	4.4	1.3	versicolor
19	5.6	3.0	4.1	1.3	versicolor
20	5.5	2.5	4.0	1.3	versicolor
21	5.5	2.6	4.4	1.2	versicolor
22	6.1	3.0	4.6	1.4	versicolor
23	5.8	2.6	4.0	1.2	versicolor
24	5.0	2.3	3.3	1.0	versicolor
25	5.6	2.7	4.2	1.3	versicolor
26	5.7	3.0	4.2	1.2	versicolor
27	5.7	2.9	4.2	1.3	versicolor
28	6.2	2.9	4.3	1.3	versicolor
29	5.1	2.5	3.0	1.1	versicolor
30	5.7	2.8	4.1	1.3	versicolor
31	7.7	3.0	6.1	2.3	virginica
32	6.3	3.4	5.6	2.4	virginica
33	6.4	3.1	5.5	1.8	virginica
34	6.0	3.0	4.8	1.8	virginica
35	6.9	3.1	5.4	2.1	virginica
36	6.7	3.1	5.6	2.4	virginica
37	6.9	3.1	5.1	2.3	virginica
38	5.8	2.7	5.1	1.9	virginica
39	6.8	3.2	5.9	2.3	virginica
40	6.7	3.3	5.7	2.5	virginica
41	6.7	3.0	5.2	2.3	virginica
42	6.3	2.5	5.0	1.9	virginica
43	6.5	3.0	5.2	2.0	virginica
44	6.2	3.4	5.4	2.3	virginica
45	5.9	3.0	5.1	1.8	virginica

`individual_equalizer_min`

individual_equalizer_min

Description

Allow to increase the number of inividual from any label to a certain point based on the individual_cloning function from the same package (Rmach)

Usage

```
individual_equalizer_min(inpt_datf, col_vars = c(), label_var, untl)
```

Arguments

inpt_datf	is the input dataset as a dataframe
col_vars	is a vector containing the colnames or the column numbers of the variables
label_var	is a either the colnames or the column number of the label variable
untl	is how many individual from the same label the dataset has to have, at minimum

Examples

```
datf <- iris
datf[, 5] <- as.character(datf[, 5])
datf <- individual_equalizer_min(inpt_datf = datf, col_vars = c(1:4), label_var = 5, untl
print(datf)
nrow(datf)
nrow(iris)
```

```
     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1        5.100000    3.500000     1.400000  0.20000000     setosa
2        4.900000    3.000000     1.400000  0.20000000     setosa
3        4.700000    3.200000     1.300000  0.20000000     setosa
4        4.600000    3.100000     1.500000  0.20000000     setosa
5        5.000000    3.600000     1.400000  0.20000000     setosa
6        5.400000    3.900000     1.700000  0.40000000     setosa
7        4.600000    3.400000     1.400000  0.30000000     setosa
8        5.000000    3.400000     1.500000  0.20000000     setosa
9        4.400000    2.900000     1.400000  0.20000000     setosa
10       4.900000    3.100000     1.500000  0.10000000     setosa
11       5.400000    3.700000     1.500000  0.20000000     setosa
12       4.800000    3.400000     1.600000  0.20000000     setosa
13       4.800000    3.000000     1.400000  0.10000000     setosa
14       4.300000    3.000000     1.100000  0.10000000     setosa
15       5.800000    4.000000     1.200000  0.20000000     setosa
16       5.700000    4.400000     1.500000  0.40000000     setosa
17       5.400000    3.900000     1.300000  0.40000000     setosa
18       5.100000    3.500000     1.400000  0.30000000     setosa
19       5.700000    3.800000     1.700000  0.30000000     setosa
20       5.100000    3.800000     1.500000  0.30000000     setosa
21       5.400000    3.400000     1.700000  0.20000000     setosa
22       5.100000    3.700000     1.500000  0.40000000     setosa
23       4.600000    3.600000     1.000000  0.20000000     setosa
24       5.100000    3.300000     1.700000  0.50000000     setosa
25       4.800000    3.400000     1.900000  0.20000000     setosa
26       5.000000    3.000000     1.600000  0.20000000     setosa
27       5.000000    3.400000     1.600000  0.40000000     setosa
28       5.200000    3.500000     1.500000  0.20000000     setosa
29       5.200000    3.400000     1.400000  0.20000000     setosa
30       4.700000    3.200000     1.600000  0.20000000     setosa
31       4.800000    3.100000     1.600000  0.20000000     setosa
32       5.400000    3.400000     1.500000  0.40000000     setosa
33       5.200000    4.100000     1.500000  0.10000000     setosa
34       5.500000    4.200000     1.400000  0.20000000     setosa
35       4.900000    3.100000     1.500000  0.20000000     setosa
```

36	5.000000	3.200000	1.200000	0.20000000	setosa
37	5.500000	3.500000	1.300000	0.20000000	setosa
38	4.900000	3.600000	1.400000	0.10000000	setosa
39	4.400000	3.000000	1.300000	0.20000000	setosa
40	5.100000	3.400000	1.500000	0.20000000	setosa
41	5.000000	3.500000	1.300000	0.30000000	setosa
42	4.500000	2.300000	1.300000	0.30000000	setosa
43	4.400000	3.200000	1.300000	0.20000000	setosa
44	5.000000	3.500000	1.600000	0.60000000	setosa
45	5.100000	3.800000	1.900000	0.40000000	setosa
46	4.800000	3.000000	1.400000	0.30000000	setosa
47	5.100000	3.800000	1.600000	0.20000000	setosa
48	4.600000	3.200000	1.400000	0.20000000	setosa
49	5.300000	3.700000	1.500000	0.20000000	setosa
50	5.000000	3.300000	1.400000	0.20000000	setosa
51	7.000000	3.200000	4.700000	1.40000000	versicolor
52	6.400000	3.200000	4.500000	1.50000000	versicolor
53	6.900000	3.100000	4.900000	1.50000000	versicolor
54	5.500000	2.300000	4.000000	1.30000000	versicolor
55	6.500000	2.800000	4.600000	1.50000000	versicolor
56	5.700000	2.800000	4.500000	1.30000000	versicolor
57	6.300000	3.300000	4.700000	1.60000000	versicolor
58	4.900000	2.400000	3.300000	1.00000000	versicolor
59	6.600000	2.900000	4.600000	1.30000000	versicolor
60	5.200000	2.700000	3.900000	1.40000000	versicolor
61	5.000000	2.000000	3.500000	1.00000000	versicolor
62	5.900000	3.000000	4.200000	1.50000000	versicolor
63	6.000000	2.200000	4.000000	1.00000000	versicolor
64	6.100000	2.900000	4.700000	1.40000000	versicolor
65	5.600000	2.900000	3.600000	1.30000000	versicolor
66	6.700000	3.100000	4.400000	1.40000000	versicolor
67	5.600000	3.000000	4.500000	1.50000000	versicolor
68	5.800000	2.700000	4.100000	1.00000000	versicolor
69	6.200000	2.200000	4.500000	1.50000000	versicolor
70	5.600000	2.500000	3.900000	1.10000000	versicolor
71	5.900000	3.200000	4.800000	1.80000000	versicolor
72	6.100000	2.800000	4.000000	1.30000000	versicolor
73	6.300000	2.500000	4.900000	1.50000000	versicolor
74	6.100000	2.800000	4.700000	1.20000000	versicolor
75	6.400000	2.900000	4.300000	1.30000000	versicolor
76	6.600000	3.000000	4.400000	1.40000000	versicolor
77	6.800000	2.800000	4.800000	1.40000000	versicolor
78	6.700000	3.000000	5.000000	1.70000000	versicolor
79	6.000000	2.900000	4.500000	1.50000000	versicolor
80	5.700000	2.600000	3.500000	1.00000000	versicolor
81	5.500000	2.400000	3.800000	1.10000000	versicolor
82	5.500000	2.400000	3.700000	1.00000000	versicolor
83	5.800000	2.700000	3.900000	1.20000000	versicolor
84	6.000000	2.700000	5.100000	1.60000000	versicolor
85	5.400000	3.000000	4.500000	1.50000000	versicolor
86	6.000000	3.400000	4.500000	1.60000000	versicolor
87	6.700000	3.100000	4.700000	1.50000000	versicolor
88	6.300000	2.300000	4.400000	1.30000000	versicolor
89	5.600000	3.000000	4.100000	1.30000000	versicolor
90	5.500000	2.500000	4.000000	1.30000000	versicolor
91	5.500000	2.600000	4.400000	1.20000000	versicolor
92	6.100000	3.000000	4.600000	1.40000000	versicolor

93	5.800000	2.600000	4.000000	1.20000000	versicolor
94	5.000000	2.300000	3.300000	1.00000000	versicolor
95	5.600000	2.700000	4.200000	1.30000000	versicolor
96	5.700000	3.000000	4.200000	1.20000000	versicolor
97	5.700000	2.900000	4.200000	1.30000000	versicolor
98	6.200000	2.900000	4.300000	1.30000000	versicolor
99	5.100000	2.500000	3.000000	1.10000000	versicolor
100	5.700000	2.800000	4.100000	1.30000000	versicolor
101	6.300000	3.300000	6.000000	2.50000000	virginica
102	5.800000	2.700000	5.100000	1.90000000	virginica
103	7.100000	3.000000	5.900000	2.10000000	virginica
104	6.300000	2.900000	5.600000	1.80000000	virginica
105	6.500000	3.000000	5.800000	2.20000000	virginica
106	7.600000	3.000000	6.600000	2.10000000	virginica
107	4.900000	2.500000	4.500000	1.70000000	virginica
108	7.300000	2.900000	6.300000	1.80000000	virginica
109	6.700000	2.500000	5.800000	1.80000000	virginica
110	7.200000	3.600000	6.100000	2.50000000	virginica
111	6.500000	3.200000	5.100000	2.00000000	virginica
112	6.400000	2.700000	5.300000	1.90000000	virginica
113	6.800000	3.000000	5.500000	2.10000000	virginica
114	5.700000	2.500000	5.000000	2.00000000	virginica
115	5.800000	2.800000	5.100000	2.40000000	virginica
116	6.400000	3.200000	5.300000	2.30000000	virginica
117	6.500000	3.000000	5.500000	1.80000000	virginica
118	7.700000	3.800000	6.700000	2.20000000	virginica
119	7.700000	2.600000	6.900000	2.30000000	virginica
120	6.000000	2.200000	5.000000	1.50000000	virginica
121	6.900000	3.200000	5.700000	2.30000000	virginica
122	5.600000	2.800000	4.900000	2.00000000	virginica
123	7.700000	2.800000	6.700000	2.00000000	virginica
124	6.300000	2.700000	4.900000	1.80000000	virginica
125	6.700000	3.300000	5.700000	2.10000000	virginica
126	7.200000	3.200000	6.000000	1.80000000	virginica
127	6.200000	2.800000	4.800000	1.80000000	virginica
128	6.100000	3.000000	4.900000	1.80000000	virginica
129	6.400000	2.800000	5.600000	2.10000000	virginica
130	7.200000	3.000000	5.800000	1.60000000	virginica
131	7.400000	2.800000	6.100000	1.90000000	virginica
132	7.900000	3.800000	6.400000	2.00000000	virginica
133	6.400000	2.800000	5.600000	2.20000000	virginica
134	6.300000	2.800000	5.100000	1.50000000	virginica
135	6.100000	2.600000	5.600000	1.40000000	virginica
136	7.700000	3.000000	6.100000	2.30000000	virginica
137	6.300000	3.400000	5.600000	2.40000000	virginica
138	6.400000	3.100000	5.500000	1.80000000	virginica
139	6.000000	3.000000	4.800000	1.80000000	virginica
140	6.900000	3.100000	5.400000	2.10000000	virginica
141	6.700000	3.100000	5.600000	2.40000000	virginica
142	6.900000	3.100000	5.100000	2.30000000	virginica
143	5.800000	2.700000	5.100000	1.90000000	virginica
144	6.800000	3.200000	5.900000	2.30000000	virginica
145	6.700000	3.300000	5.700000	2.50000000	virginica
146	6.700000	3.000000	5.200000	2.30000000	virginica
147	6.300000	2.500000	5.000000	1.90000000	virginica
148	6.500000	3.000000	5.200000	2.00000000	virginica
149	6.200000	3.400000	5.400000	2.30000000	virginica

150	5.900000	3.000000	5.100000	1.80000000	virginica
151	5.119546	3.240896	1.659373	0.25516050	setosa
152	4.902088	4.003746	1.228617	0.35778383	setosa
153	4.834331	3.698540	1.547812	0.33339113	setosa
154	5.134884	3.180819	1.588032	0.18761885	setosa
155	5.488401	3.298369	1.683031	0.18180736	setosa
156	4.758992	3.086108	1.434159	0.25348240	setosa
157	4.817610	3.052438	1.470246	0.18414810	setosa
158	5.372952	3.815612	1.344489	0.12705451	setosa
159	5.203751	3.331928	1.384586	0.26145797	setosa
160	5.154693	4.326639	1.585445	0.10767788	setosa
161	4.651867	2.915629	1.333128	0.24085761	setosa
162	4.703818	3.295307	1.524695	0.53200346	setosa
163	5.299254	3.127387	1.436154	0.32571756	setosa
164	4.576459	3.690579	1.500380	0.24860844	setosa
165	4.821700	3.891746	1.277726	0.34434218	setosa
166	5.195495	2.693142	1.518095	0.11628275	setosa
167	4.751171	4.076332	1.437831	0.29611751	setosa
168	4.895746	3.340168	1.505157	0.32204518	setosa
169	5.084452	2.649230	1.253577	0.34230634	setosa
170	4.994526	3.283612	1.466568	0.10785695	setosa
171	4.914249	3.713116	1.456736	0.13825711	setosa
172	5.168494	3.384539	1.391309	0.36352904	setosa
173	4.868237	3.608825	1.580430	0.16346689	setosa
174	4.922010	3.812630	1.385674	0.17966376	setosa
175	4.782539	3.520596	1.166369	0.19443475	setosa
176	4.999012	2.953373	1.276890	0.04813659	setosa
177	4.237476	3.501651	1.603897	-0.02137016	setosa
178	4.161835	2.900175	1.340508	0.31471652	setosa
179	5.326641	2.690628	1.367918	0.30229792	setosa
180	5.144879	2.889594	1.627228	0.29699450	setosa
181	5.032020	3.092995	1.262743	0.13014888	setosa
182	4.912576	4.102884	1.592814	0.46510333	setosa
183	4.886276	3.643501	1.362697	0.45850332	setosa
184	5.067843	3.644076	1.284018	0.11802271	setosa
185	4.870130	3.261045	1.387769	0.24945158	setosa
186	4.203276	3.532647	1.759381	0.22793382	setosa
187	5.147728	2.949748	1.344759	0.14613345	setosa
188	5.044451	3.821792	1.690910	0.27432788	setosa
189	5.144534	3.260319	1.486522	0.15193060	setosa
190	4.749463	3.242690	1.558031	0.29964703	setosa
191	5.012355	4.056773	1.568806	0.28175520	setosa
192	5.286178	3.657418	1.556329	0.25865612	setosa
193	4.739473	3.599081	1.361732	0.11096506	setosa
194	4.763743	3.719912	1.532282	0.23680057	setosa
195	4.352927	3.606171	1.443575	0.22201153	setosa
196	5.420318	3.234039	1.257110	0.29332868	setosa
197	5.032471	4.002458	1.149330	0.14118440	setosa
198	4.679526	3.634655	1.503754	0.19732104	setosa
199	4.655581	2.890624	1.538909	0.10855489	setosa
200	5.432263	3.587195	1.448039	0.15201721	setosa
201	5.030955	3.620666	1.379309	0.22296525	setosa
202	5.117052	3.640415	1.680914	0.22426164	setosa
203	4.206403	3.577511	1.579905	0.34627623	setosa
204	5.345245	3.207691	1.351151	0.10816533	setosa
205	5.287934	3.630390	1.494184	0.31610331	setosa
206	4.371540	3.674677	1.483436	0.12756906	setosa

207	4.458787	3.512193	1.499114	0.35598241	setosa
208	4.694526	4.189214	1.065203	0.32728599	setosa
209	5.199256	3.164026	1.523074	-0.00277085	setosa
210	4.857067	3.279462	1.431379	0.28051926	setosa
211	5.120333	3.079011	1.256199	0.26650341	setosa
212	5.526492	3.715932	1.385397	0.10935802	setosa
213	4.255062	3.442076	1.032584	0.22553491	setosa
214	5.547997	3.899931	1.805604	0.14245435	setosa
215	5.056086	3.556886	1.485842	0.25052054	setosa
216	4.602273	3.582194	1.637627	0.10750785	setosa
217	5.707143	3.272366	1.495331	0.24957136	setosa
218	4.529437	3.295707	1.370119	0.22733484	setosa
219	4.815724	3.274761	1.264803	0.19839835	setosa
220	5.219331	3.678528	1.534777	0.31111961	setosa
221	7.469393	3.164987	4.045576	1.29571858	versicolor
222	6.435686	2.759148	4.152976	1.38660991	versicolor
223	6.004909	2.617229	3.374965	1.50931533	versicolor
224	5.998960	2.889328	4.787927	0.99816956	versicolor
225	6.066878	2.738260	4.317450	1.35360632	versicolor
226	6.558577	3.004756	3.518091	1.10350572	versicolor
227	5.226591	2.937582	4.211646	1.82617395	versicolor
228	6.519901	3.085536	4.666132	1.47417398	versicolor
229	6.212108	2.297953	3.256134	1.57999643	versicolor
230	6.234065	2.904038	3.899946	1.61728632	versicolor
231	5.891353	2.871663	3.585063	1.15322879	versicolor
232	5.495659	2.332178	4.373762	1.40539853	versicolor
233	5.484945	3.186158	5.022759	1.03428734	versicolor
234	5.002003	2.716631	4.221475	1.13953629	versicolor
235	6.289043	3.017459	3.910062	1.38286708	versicolor
236	5.700736	3.131150	4.960207	1.14958223	versicolor
237	5.500216	3.190272	4.253273	1.18245190	versicolor
238	6.445503	2.960724	4.621510	1.21795268	versicolor
239	5.889688	2.752965	4.360846	1.21917725	versicolor
240	5.217994	2.727503	4.018054	1.19655177	versicolor
241	5.628761	2.782079	4.503714	1.21105694	versicolor
242	5.922639	2.647391	3.774616	1.48842237	versicolor
243	6.021925	2.565549	3.937052	1.45849084	versicolor
244	6.330301	2.687627	4.026615	0.94678432	versicolor
245	6.304311	2.635169	3.998727	1.45603553	versicolor
246	6.663896	3.297885	3.907486	1.16979322	versicolor
247	5.376404	2.885587	3.866554	1.05112744	versicolor
248	4.695327	2.578715	3.943357	1.16919180	versicolor
249	6.278448	3.381682	3.893139	1.31728551	versicolor
250	5.808922	2.342279	4.329488	1.36901786	versicolor
251	6.257850	3.299147	4.763327	1.45358673	versicolor
252	5.397398	2.181731	5.237967	1.63885805	versicolor
253	6.318406	3.370869	4.403785	1.71528585	versicolor
254	6.030213	2.934996	5.690094	1.18095022	versicolor
255	6.322254	2.643724	4.712019	1.30067547	versicolor
256	5.483814	3.540120	3.935919	1.36104088	versicolor
257	4.923149	2.834738	3.978205	1.09514320	versicolor
258	5.102353	3.275399	4.167623	1.69802624	versicolor
259	6.503755	2.772905	4.500401	1.10261134	versicolor
260	6.024940	2.379938	3.663719	1.24096925	versicolor
261	6.155505	2.960939	4.628437	1.63876689	versicolor
262	6.547596	2.753326	3.814345	1.50055748	versicolor
263	7.340028	3.049036	4.128880	1.43704378	versicolor

264	6.771703	2.744679	3.755760	1.35657812	versicolor
265	6.526113	3.315310	4.723554	1.13676188	versicolor
266	5.737681	2.732723	4.619607	1.20118401	versicolor
267	5.118896	3.053538	5.153921	1.24286955	versicolor
268	6.557536	2.506483	3.775426	1.25665234	versicolor
269	6.773637	3.056770	3.907444	1.48359009	versicolor
270	5.231083	2.716242	3.701491	1.43445828	versicolor
271	6.373044	2.810367	3.823155	1.48776176	versicolor
272	6.689764	2.329003	4.315204	1.20003129	versicolor
273	5.909787	2.877026	3.921463	1.44035219	versicolor
274	5.985060	3.408963	4.312826	1.14822888	versicolor
275	5.720711	3.047025	4.502301	1.30692891	versicolor
276	6.075586	2.625810	3.462166	1.13883320	versicolor
277	5.979742	3.037604	4.337108	1.17174718	versicolor
278	5.944742	3.187138	4.131605	1.40617115	versicolor
279	5.377366	2.850410	4.848731	1.31109047	versicolor
280	5.911520	2.601061	3.978657	1.19677413	versicolor
281	6.299276	3.083130	3.767828	1.21669672	versicolor
282	6.508117	2.717810	4.400327	1.15816277	versicolor
283	5.564065	2.991926	3.244794	0.97614826	versicolor
284	5.636803	3.041730	3.675623	1.52144698	versicolor
285	6.249670	2.545928	4.021866	1.48874150	versicolor
286	5.779178	3.126088	4.456842	1.35907598	versicolor
287	5.056560	3.158496	4.029340	1.09487926	versicolor
288	6.256082	2.754099	3.546839	1.10515518	versicolor
289	6.727157	3.127967	4.478930	1.36983039	versicolor
290	6.644075	2.156546	4.073352	1.24130902	versicolor
291	6.086009	2.661626	6.272420	1.43328200	virginica
292	6.415624	3.507285	4.970803	2.29244152	virginica
293	7.783730	3.194127	6.263952	2.12710505	virginica
294	6.714708	2.207256	4.695838	1.57280728	virginica
295	6.892027	3.146945	5.963832	2.03720894	virginica
296	6.384602	2.842640	5.424208	1.34455702	virginica
297	7.151880	2.761441	5.193842	2.65759524	virginica
298	7.000909	3.538284	5.949645	2.37981867	virginica
299	6.267784	3.471146	5.832588	1.97858577	virginica
300	6.684294	3.095409	5.918461	1.79584906	virginica
301	6.653542	3.193293	5.478747	2.02974253	virginica
302	6.932936	2.532998	5.398907	2.58686242	virginica
303	6.171339	3.401070	5.778270	2.14575174	virginica
304	6.321461	3.238482	5.728325	1.77370288	virginica
305	6.939597	3.105226	5.153168	2.30218152	virginica
306	4.983468	2.869016	5.249331	2.33602954	virginica
307	7.057275	3.000195	5.368063	2.29811745	virginica
308	5.648449	3.022504	4.670324	2.44199827	virginica
309	7.023223	3.038748	6.549980	1.74164740	virginica
310	6.621430	2.928325	4.114293	1.65060008	virginica
311	5.947210	2.572431	6.035025	1.67473550	virginica
312	6.720834	2.791217	4.373968	1.80139289	virginica
313	7.277691	3.013233	6.057093	2.41664038	virginica
314	6.036578	3.034487	5.680667	2.14347484	virginica
315	7.523033	2.906421	5.746571	2.19174990	virginica
316	6.148008	3.219150	5.385260	2.29487465	virginica
317	6.653134	3.286357	5.439343	2.01415643	virginica
318	7.665406	2.418833	4.912548	2.04701493	virginica
319	6.962181	3.122207	5.926113	2.14427668	virginica
320	6.968055	3.394053	5.176526	2.28774948	virginica

```
321     8.433217    3.190685     6.154875  1.86645175  virginica
322     5.865485    3.206422     6.182362  2.06380350  virginica
323     6.357587    3.105502     6.086674  2.22194560  virginica
324     7.000027    3.093890     5.694556  1.95490517  virginica
325     5.329756    3.313431     7.114499  1.82374316  virginica
326     7.063835    2.978432     6.702789  1.97846514  virginica
327     6.643032    3.331938     5.319034  1.98032475  virginica
328     5.812732    2.605752     4.698275  2.04751518  virginica
329     5.922603    2.951062     4.789723  1.86828922  virginica
330     6.534338    3.077621     4.735738  1.96590508  virginica
331     6.566409    2.869386     5.256565  2.30887779  virginica
332     5.873025    2.576689     5.399706  1.51365277  virginica
333     6.436762    2.807203     5.237271  1.70436243  virginica
334     6.700115    2.741499     6.361120  2.57743789  virginica
335     6.800498    2.964161     6.726096  2.01077453  virginica
336     6.817689    3.044292     5.651350  1.64623491  virginica
337     6.589657    2.978472     6.011304  2.51979646  virginica
338     8.263734    3.121411     5.285361  1.93618630  virginica
339     7.027356    2.891612     5.821978  1.92039311  virginica
340     4.943241    2.503378     5.732430  1.80385345  virginica
341     7.071175    2.628713     6.012994  2.06170238  virginica
342     6.074115    3.436504     5.791817  1.23968953  virginica
343     6.853310    2.681229     5.643604  1.21275207  virginica
344     6.254123    3.365158     5.832863  2.67274454  virginica
345     6.511558    2.738037     5.355683  1.85846301  virginica
346     5.842295    3.300082     4.540820  2.12329402  virginica
347     6.423004    3.294433     6.394560  1.76478497  virginica
348     5.833874    3.222916     5.861218  1.69319220  virginica
349     6.478021    3.028388     6.606609  2.06623919  virginica
350     7.784342    2.902471     5.142493  1.91602616  virginica
351     6.775815    3.445127     5.519265  2.13719655  virginica
352     7.014933    2.715428     6.798085  2.04147119  virginica
353     7.689606    2.506295     5.531764  1.88075834  virginica
354     7.506985    2.788839     5.837837  2.47057469  virginica
355     7.242421    2.782457     6.390016  1.66938074  virginica
356     6.400116    2.353697     4.388649  2.24717026  virginica
357     7.384851    3.077118     5.716925  2.36297064  virginica
358     6.892294    3.466955     4.959172  2.13813060  virginica
359     5.904443    3.286340     4.911794  1.90991134  virginica
360     6.292600    2.938076     5.710938  2.61396630  virginica
[1] 360
[1] 150
```

`individual_route` *individual_route*

Description

From a time serie, allow to get the most common route for each individual at a given depth (time - 1). Access the frequency value as an element from the output vector and the value itself (the path) as a name of its element, see examples.

Usage

```
individual_route(inpt_datf, col_target, id_col, untl_last = 2)
```

Arguments

inpt_datf	is the input time serie as a dataframe
col_target	is the column name or number that refers to the value of each individual
id_col	is the column name or number that refers to the individual (ids)
untl_last	is the depth value

Examples

```
datf_test <- data.frame("id" = c(1, 1, 1, 2, 2, 3, 3, 3, 4, 4, 5, 5, 5, 5),
                        "city" = c("A", "C", "B", "B", "A", "C", "A", "C", "A", "C", "B"

print(individual_route(inpt_datf = datf_test,
                       col_target = "city",
                       id_col = "id",
                       untl_last = 2))

AC CA BA
 2  1  2

print(individual_route(inpt_datf = datf_test,
                       col_target = "city",
                       id_col = "id",
                       untl_last = 3))

ACB  AC CAC  BA BAA
  1   2   1   2   1
```

---

knn_Rmach *knn_Rmach*

Description

KNN algorythm, see example

Usage

```
knn_Rmach(train, test, k, col_vars_train = c(), col_vars_test = c(), class_col)
```

Arguments

train	is a dataframe with the known individual and their variadbles and classification columns
test	is a dataframe with the new individuals with ich e do not know the class, only the variables
k	is the number of neighbours
col_vars_train	is a vector containing the column names or column numbers of the variables in train, if empty all column are considered as a variable apart from the last one that is considered as the classification column

col_vars_test
: is a vector containing the column names or column numbers of the variables in test, if empty all column are considered as a variable

class_col
: is the column name or column number of the classification column in train

Examples

```
cur_ids <- round(runif(n = 45, min = 1, max = 150))

vec <- knn_Rmach(train = iris[-cur_ids,],
          test = iris[cur_ids, 1:4],
           col_vars_train = c(1:4),
          col_vars_test = c(1:4),
          class_col = 5,
          k = 3
       )

sum(vec == iris[cur_ids, 5]) / 45

[1] 0.9555556
```

---

knn_Rmach_cross_validation_k

knn_Rmach_cross_validation_k

Description

Allow to perform knn with cross validation for the optimal value of k neighbours used, see examples and parameters. The result outputed is a vector containing the ratio of correct label found divided by the total number of unique individuals in the current dataset where the training occurred. So, higher is better.

Usage

```
knn_Rmach_cross_validation_k(
  inpt_datf,
  train_prop,
  knn_v = c(),
  n_fold = 5,
  col_vars = c(),
  class_col
)
```

Arguments

inpt_datf
: is the input dataset as a ddataframe

train_prop
: is the training proportion

knn_v
: is a vector containing the values of k neighbours to test

n_fold
: is the number of fold used for each value of k, the higher this value is, he more accurate the result will be but the higher the amount of time it will takes

col_vars	is a vector containing the column names or numbers of the variables in the input dataframe
class_col	is the column names or number of the variable to predict in the input dataframe

Examples

```
iris[, 5] <- as.character(iris[, 5])
print(knn_Rmach_cross_validation_k(
        inpt_datf = iris,
        col_vars = c(1:4),
        n_fold = 5,
        knn_v = c(3, 5, 7, 9, 11),
        class_col = 5,
        train_prop = 0.7
))

[1] 0.9333333 0.9200000 0.9333333 0.9466667 0.9288889

# here the optimal k value is 9
```

---

knn_Rmach_cross_validation_train

knn_Rmach_cross_validation_train

Description

Allow to perform knn with cross validation for the optimal value of k neighbours used, see examples and parameters. The result outputed is a vector containing the ratio of correct label found divided by the total number of individuals in the current dataset where the training occurred. So, higher is better.

Usage

```
knn_Rmach_cross_validation_train(
  inpt_datf,
  train_prop_v = c(),
  k,
  n_fold = 5,
  col_vars = c(),
  class_col
)
```

Arguments

inpt_datf	is the input dataset as a ddataframe
n_fold	is the number of fold used for each value of k, the higher this value is, he more accurate the result will be but the higher the amount of time it will takes
col_vars	is a vector containing the column names or numbers of the variables in the input dataframe
class_col	is the column names or number of the variable to predict in the input dataframe
train_prop	is the training proportion
knn_v	is a vector containing the values of k neighbours to test

Examples

```
iris[, 5] <- as.character(iris[, 5])
print(knn_Rmach_cross_validation_train(
        inpt_datf = iris,
        col_vars = c(1:4),
        n_fold = 15,
        k = 7,
        class_col = 5,
        train_prop_v = c(0.7, 0.75, 0.8)
))

[1] 0.4057143 0.3273810 0.2400000

# here the optimal training proportion is 0.7
```

`poly_model` *Rmach poly_model*

Description

Take a datasets of x and y values and a function tha could fit all the data with the missing coefficients, and returns a list containing the coefficients that fit the best the data for a given function, as a vector for the first index, and at the second index, the actual sum of difference between each data point and the function at the same x values.

Usage

```
poly_model(
  inpt_datf,
  degree,
  twk_val = NA,
  sensi_val = twk_val,
  coeff_v = NA,
  powers = NA,
  mth_symb = c("x"),
  numrtr_v = NA
)
```

Arguments

`inpt_datf`	is the input data as a dataframe, first column is the x values and the second is the y values
`degree`	is how many coefficients will be involved (each coefficient multiplies either an x to the power of something, an exponential of something or a base something logarithm for a something value)
`twk_val`	is the value used for finding the best coefficients, it is directly linked to the accuracy of the coefficients, see the description for more information. Defaults to (max(yval) - min(yval)) / n

`sensi_val` is the value from which two variations of a coefficient brings a so small accuracy contribution that the algorythm does not continue to find better coefficients. For example, if i set sensi_val = 0.001, so if coefficients alpha1 and beta1 brings a total difference between the function and the actual data of 10.8073 and then the algorythm find alpha2 and beta1 that brings a total difference equal to 10.8066, so the algorythm will stop running. But the coefficients returned will still be the best, that is alpha2 and beta1

`coeff_v` is a vector containing the original coefficients for the function, so the closest those are from the best one, the fastest the algorythm will compute the best coefficients. The first value of coeff is always the constant.

`powers` is a vector containing the exponent, or related value to mth_symb. powers can be a vector if those values are constants or it could be a list of vectors the length of observed individuals, if those values varies like in the examples. Notthat if you use variables in powers (list), each values of a vector from this list has to be at the exact same x coordinates of each observed individuals in the input dataframe. Ex: datf <- data.frame("x"=c(4, 4, 3, 2, 1, 1), "y"=c(1:6)), so vector(s) from powers that contain varying value must be of length 4. Also, the values are not ascendly sorted, don't worry values are ascendly sorted under the hood, so fill your powers vectors in the intuitive ascendly way

`mth_symb` is a vector containing the elemnts linked to the coefficients from the second element. It can be x, e (exp(x)) or log-X (log(x)-base), and their reverse like 1/x. If the numerator varies the element should be entered like tis list/x, list/e or list/log-base. See numrtr_v for the values related to list

`numrtr_v` is a vector containing the values for the numerator related to mth_symb if on element is like this: list/x or list/e

Examples

```
print(poly_model(inpt_datf=data.frame(mtcars$wt, mtcars$mpg), degree=2, coeff_v=c(32.5, -

                 numrtr_v=NA))

[[1]]
[1] 33.234375 -4.265625

[[2]]
[1] 74.78275

print(poly_model(inpt_datf=data.frame(mtcars$wt, mtcars$mpg), degree=2, coeff_v=c(32.5, -

                 numrtr_v=NA))

[[1]]
[1] 31.765625 -3.734375

[[2]]
[1] 80.36228

print(poly_model(inpt_datf=data.frame(mtcars$wt, mtcars$mpg), degree=2, coeff_v=c(32.5, -

                 numrtr_v=NA))

[[1]]
[1] 32.5 -3.0
```

```
[[2]]
[1] 1.067436e+24

print(poly_model(inpt_datf=data.frame(mtcars$wt, mtcars$mpg), degree=2, coeff_v=c(32.5, -

                 numrtr_v=list(c(length(mtcars$wt):1))))
[[1]]
[1] 19.28125 -0.06250

[[2]]
[1] 35839.44

print(poly_model(inpt_datf=data.frame(mtcars$wt, mtcars$mpg), degree=2, coeff_v=c(32.5, -

                 numrtr_v=NA))

[[1]]
[1] 27.359375 -8.140625

[[2]]
[1] 160.2263

print(poly_model(inpt_datf=data.frame(mtcars$wt, mtcars$mpg), degree=1, coeff_v=c(32.5),

                 numrtr_v=NA))

[[1]]
[1] 19.28125

[[2]]
[1] 148.7625

print(poly_model(inpt_datf=data.frame(mtcars$wt, mtcars$mpg), degree=2, coeff_v=c(32.5, -

                  numrtr_v=NA))

[[1]]
[1]  0.921875 -5.203125  2.000000

[[2]]
[1] 455.6017
```

`sample_Rmach-class` *v_Rmach_fold*

Description

Allow to create uniform sampling dataset for cross validation, train and test, see examples and variables

Arguments

`inpt_datf` is the input dataframe

`train_prop`	is the training proportion
`n_fold`	is the number of distinc pair of training and test dataset that will be outputed

Examples

```
lst_test <- v_Rmach_fold(inpt_datf = iris[1:25,],
             train_prop = 0.7,
             n_fold = 4)

print(lst_test)

$sample1
An object of class "sample_Rmach"
Slot "train":
     Sepal.Length Sepal.Width Petal.Length Petal.Width Species test_status
24            5.1         3.3          1.7         0.5  setosa           0
18            5.1         3.5          1.4         0.3  setosa           0
12            4.8         3.4          1.6         0.2  setosa           0
19            5.7         3.8          1.7         0.3  setosa           0
20            5.1         3.8          1.5         0.3  setosa           0
5             5.0         3.6          1.4         0.2  setosa           0
4             4.6         3.1          1.5         0.2  setosa           0
23            4.6         3.6          1.0         0.2  setosa           0
18.1          5.1         3.5          1.4         0.3  setosa           0
1             5.1         3.5          1.4         0.2  setosa           0
7             4.6         3.4          1.4         0.3  setosa           0
14            4.3         3.0          1.1         0.1  setosa           0
7.1           4.6         3.4          1.4         0.3  setosa           0
4.1           4.6         3.1          1.5         0.2  setosa           0
19.1          5.7         3.8          1.7         0.3  setosa           0
9             4.4         2.9          1.4         0.2  setosa           0
8             5.0         3.4          1.5         0.2  setosa           0
16            5.7         4.4          1.5         0.4  setosa           0

Slot "test":
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species test_status
7           4.6         3.4          1.4         0.3  setosa           1
12          4.8         3.4          1.6         0.2  setosa           1
8           5.0         3.4          1.5         0.2  setosa           1
14          4.3         3.0          1.1         0.1  setosa           1
11          5.4         3.7          1.5         0.2  setosa           1
25          4.8         3.4          1.9         0.2  setosa           1
23          4.6         3.6          1.0         0.2  setosa           1

Slot "train_ids":
 [1] 24 18 12 19 20  5  4 23 18  1  7 14  7  4 19  9  8 16

Slot "test_ids":
[1]  7 12  8 14 11 25 23


$sample2
An object of class "sample_Rmach"
Slot "train":
     Sepal.Length Sepal.Width Petal.Length Petal.Width Species test_status
20            5.1         3.8          1.5         0.3  setosa           0
8             5.0         3.4          1.5         0.2  setosa           0
```

```
2             4.9         3.0          1.4         0.2  setosa           0
11            5.4         3.7          1.5         0.2  setosa           0
22            5.1         3.7          1.5         0.4  setosa           0
13            4.8         3.0          1.4         0.1  setosa           0
24            5.1         3.3          1.7         0.5  setosa           0
2.1           4.9         3.0          1.4         0.2  setosa           0
7             4.6         3.4          1.4         0.3  setosa           0
2.2           4.9         3.0          1.4         0.2  setosa           0
22.1          5.1         3.7          1.5         0.4  setosa           0
22.2          5.1         3.7          1.5         0.4  setosa           0
24.1          5.1         3.3          1.7         0.5  setosa           0
22.3          5.1         3.7          1.5         0.4  setosa           0
3             4.7         3.2          1.3         0.2  setosa           0
3.1           4.7         3.2          1.3         0.2  setosa           0
11.1          5.4         3.7          1.5         0.2  setosa           0
6             5.4         3.9          1.7         0.4  setosa           0

Slot "test":
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species test_status
8           5.0         3.4          1.5         0.2  setosa           1
12          4.8         3.4          1.6         0.2  setosa           1
1           5.1         3.5          1.4         0.2  setosa           1
11          5.4         3.7          1.5         0.2  setosa           1
2           4.9         3.0          1.4         0.2  setosa           1
18          5.1         3.5          1.4         0.3  setosa           1
20          5.1         3.8          1.5         0.3  setosa           1

Slot "train_ids":
 [1] 20  8  2 11 22 13 24  2  7  2 22 22 24 22  3  3 11  6

Slot "test_ids":
[1]  8 12  1 11  2 18 20


$sample3
An object of class "sample_Rmach"
Slot "train":
     Sepal.Length Sepal.Width Petal.Length Petal.Width Species test_status
5             5.0         3.6          1.4         0.2  setosa           0
14            4.3         3.0          1.1         0.1  setosa           0
16            5.7         4.4          1.5         0.4  setosa           0
4             4.6         3.1          1.5         0.2  setosa           0
16.1          5.7         4.4          1.5         0.4  setosa           0
15            5.8         4.0          1.2         0.2  setosa           0
3             4.7         3.2          1.3         0.2  setosa           0
18            5.1         3.5          1.4         0.3  setosa           0
25            4.8         3.4          1.9         0.2  setosa           0
23            4.6         3.6          1.0         0.2  setosa           0
4.1           4.6         3.1          1.5         0.2  setosa           0
24            5.1         3.3          1.7         0.5  setosa           0
20            5.1         3.8          1.5         0.3  setosa           0
7             4.6         3.4          1.4         0.3  setosa           0
19            5.7         3.8          1.7         0.3  setosa           0
21            5.4         3.4          1.7         0.2  setosa           0
23.1          4.6         3.6          1.0         0.2  setosa           0
11            5.4         3.7          1.5         0.2  setosa           0
```

```
Slot "test":
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species test_status
18          5.1         3.5          1.4         0.3  setosa           1
21          5.4         3.4          1.7         0.2  setosa           1
5           5.0         3.6          1.4         0.2  setosa           1
12          4.8         3.4          1.6         0.2  setosa           1
14          4.3         3.0          1.1         0.1  setosa           1
2           4.9         3.0          1.4         0.2  setosa           1
8           5.0         3.4          1.5         0.2  setosa           1

Slot "train_ids":
 [1]  5 14 16  4 16 15  3 18 25 23  4 24 20  7 19 21 23 11

Slot "test_ids":
[1] 18 21  5 12 14  2  8


$sample4
An object of class "sample_Rmach"
Slot "train":
     Sepal.Length Sepal.Width Petal.Length Petal.Width Species test_status
18            5.1         3.5          1.4         0.3  setosa           0
18.1          5.1         3.5          1.4         0.3  setosa           0
13            4.8         3.0          1.4         0.1  setosa           0
7             4.6         3.4          1.4         0.3  setosa           0
18.2          5.1         3.5          1.4         0.3  setosa           0
2             4.9         3.0          1.4         0.2  setosa           0
19            5.7         3.8          1.7         0.3  setosa           0
9             4.4         2.9          1.4         0.2  setosa           0
23            4.6         3.6          1.0         0.2  setosa           0
15            5.8         4.0          1.2         0.2  setosa           0
16            5.7         4.4          1.5         0.4  setosa           0
15.1          5.8         4.0          1.2         0.2  setosa           0
8             5.0         3.4          1.5         0.2  setosa           0
9.1           4.4         2.9          1.4         0.2  setosa           0
10            4.9         3.1          1.5         0.1  setosa           0
14            4.3         3.0          1.1         0.1  setosa           0
11            5.4         3.7          1.5         0.2  setosa           0
12            4.8         3.4          1.6         0.2  setosa           0

Slot "test":
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species test_status
9           4.4         2.9          1.4         0.2  setosa           1
13          4.8         3.0          1.4         0.1  setosa           1
4           4.6         3.1          1.5         0.2  setosa           1
19          5.7         3.8          1.7         0.3  setosa           1
22          5.1         3.7          1.5         0.4  setosa           1
11          5.4         3.7          1.5         0.2  setosa           1
5           5.0         3.6          1.4         0.2  setosa           1

Slot "train_ids":
 [1] 18 18 13  7 18  2 19  9 23 15 16 15  8  9 10 14 11 12

Slot "test_ids":
[1]  9 13  4 19 22 11  5
```

Index