# Package 'IndRSA'

July 20, 2021

**Title** What the Package Does (One Line, Title Case)

**Version** 0.0.0.9000

**Description** What the package does (one paragraph).

**License** `use_mit_license()`, `use_gpl3_license()` or friends to pick a license

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.1.1.9001

## R topics documented:

---

`aictab_ind` *Perform model selection over all individuals*

---

## Description

Perform AIC model selection over all individuals by adding up likelihood of individual model (based code partly taken from package AICcmodavg)

## Usage

```
aictab_ind(mod_ls, cutoff = 0, K = NULL)
```

## Arguments

| | |
|---|---|
| `mod_ls` | A list of list of model generated by rsf_ind |
| `cutoff` | A cutoff value to exclude individuals with bad fit, default = -1 indicating model that did not converge will be excluded. Values > 0 will exclude based on coefficient values |

## Value

A AIC model selection table

## Examples

```
data(goats)
ls1<-list()
ls1[[1]]<-as.formula(STATUS~ELEVATION+SLOPE+ET+ASPECT+HLI+TASP)
ls1[[2]]<-as.formula(STATUS~ET+ASPECT+HLI+TASP)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
aictab_ind(out)
```

## bad_fit — *Identify potential individual with bad fits based on coefficients and model convergence*

### Description

Identify potential individual with bad fits based on coefficients and model convergence

### Usage

```
bad_fit(mod_ls, cutoff = 1000)
```

### Arguments

| | |
|---|---|
| `mod_ls` | A list of list of model generated by rsf_ind |
| `cutoff` | Value a coeffiecient may take to indicate bad fit (default=1000) |

### Value

A list with first element giving individuals with bad fits based on coefficients and second element containing individuals with bad fit based on convergence

### Examples

```
data(goats)
ls1<-list()
ls1[[1]]<-as.formula(STATUS~ELEVATION+SLOPE+ET+ASPECT+HLI+TASP)
ls1[[2]]<-as.formula(STATUS~ET+ASPECT+HLI+TASP)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
bad_fit(out, cutoff=20) #None
```

## bad_fit1 — *Identify potential individual with bad fits based on coefficients and model convergence for a specific model*

### Description

Identify potential individual with bad fits based on coefficients and model convergence

### Usage

```
bad_fit1(mod_ls, cutoff = 1000, m = 1)
```

### Arguments

| | |
|---|---|
| `mod_ls` | A list of list of model generated by rsf_ind |
| `cutoff` | Value a coeffiecient may take to indicate bad fit (default=1000) |
| `m` | model number (based on number in list of formula provided to rsf_ind) |

## Value

A list with first element giving individuals with bad fits based on coefficients and second element containing individuals with bad fit based on convergence

## Examples

```
data(goats)
ls1<-list()
ls1[[1]]<-as.formula(STATUS~ELEVATION+SLOPE+ET+ASPECT+HLI+TASP)
ls1[[2]]<-as.formula(STATUS~ET+ASPECT+HLI+TASP)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
bad_fit(out, cutoff=20, m=1) #None
```

---

## cmodel *Remove elements from glm object to save space*

---

## Description

Remove elements from glm objects (Taken from: https://www.r-bloggers.com/trimming-the-fat-from-glm-models-in-r/)

## Usage

```
cmodel(cm)
```

## Arguments

| | |
|---|---|
| cm | A glm object |

## Value

A glm object

---

## cmodel_ssf *Remove elements from clogit object to save space*

---

## Description

Remove elements from glm objects

## Usage

```
cmodel_ssf(cm)
```

## Arguments

| | |
|---|---|
| cm | A coxph object |

## Value

A coxph object

## eval_ratio — *Evaluate ratio of used and random locations of individuals in a RSF table*

### Description

Evaluate ratio of used and random locations of individuals in a RSF table

### Usage

```
eval_ratio(id, value)
```

### Arguments

| | |
|---|---|
| `id` | A vector of individual for each observation |
| `value` | A vector indicating if each observation is used (=1) or random(=0) |

### Value

A list indicating the range in ratio, range in random locations, and range in used location.

## files2list — *Convert a folder with individual files to object provided by ssf_ind or rsf_ind*

### Description

This function convert a folder of individual files (for example if analysis was ran on a high performance cluster) to the same format provided by ssf_ind or rsf_ind. This facilitate the use of other functions such as aictab_ind and pop_avg.

### Usage

```
files2list(lsfiles, cleanModel = F)
```

### Arguments

| | |
|---|---|
| `lsfiles` | The list of files to be imported, for example using the dir() command. |
| `cleanModel` | Whether the cleanModel function should be applied. . |

### Value

A list of the same format as ssf_ind or rsf_ind.

## `goats` *goats - Mountain goats data set*

### Description

GPS collar data of mountain goats (Oreamnos americanus) from Lele and Keim (2006).

### Usage

```
goats
```

### Format

A data frame with 19014 rows and 8 variables

STATUS a numeric vector, 1: used, 0: available

ID a numeric vector, individuals

ELEVATION a numeric vector (m)

SLOPE a numeric vector (degrees, steep)

ET a numeric vector, access to escape terrain (distance from steep slopes, m)

ASPECT a numeric vector (degrees)

HLI a numeric vector, heat load index (0-1)

TASP a numeric vector, transformed aspect

## `ind_coef` *Extract individual coefficients*

### Description

Extract individual coefficients.

### Usage

```
ind_coef(m = 1, mod_ls, cutoff = 0, id_year = F)
```

### Arguments

| | |
|---|---|
| `m` | model number (based on number in list of formula provided to rsf_ind) |
| `mod_ls` | A list of list of model generated by rsf_ind |
| `cutoff` | A cutoff value to exclude individuals with bad fit, default = -1 indicating model that did not converge will be excluded. Values > 0 will exclude based on coefficient |
| `id_year` | Whether id_year (instead of individual) are provided. Individual and year needs to be separated by an underscore for the function to work properly. |

### Value

A table of individual coefficients

## Examples

```
data(goats)
ls1<-list()
ls1[[1]]<-as.formula(STATUS~ELEVATION+SLOPE+ET+ASPECT+HLI+TASP)
ls1[[2]]<-as.formula(STATUS~ET+ASPECT+HLI+TASP)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
ind_coef(m=1, out)
```

---

`ind_se` *Extract individual standard errors*

---

## Description

Extract individual standard errors

## Usage

```
ind_se(m = 1, mod_ls, cutoff = 0, id_year = F)
```

## Arguments

| | |
|---|---|
| `m` | model number (based on number in list of formula provided to rsf_ind) |
| `mod_ls` | A list of list of model generated by rsf_ind |
| `cutoff` | A cutoff value to exclude individuals with bad fit, default = -1 indicating model that did not converge will be excluded. Values > 0 will exclude based on coefficient |
| `id_year` | Whether id_year (instead of individual) are provided. Individual and year needs to be separated by an underscore for the function to work properly. |

## Value

A table of individual standard errors for each coefficients

## Examples

```
data(goats)
ls1<-list()
ls1[[1]]<-as.formula(STATUS~ELEVATION+SLOPE+ET+ASPECT+HLI+TASP)
ls1[[2]]<-as.formula(STATUS~ET+ASPECT+HLI+TASP)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
ind_se(m=1, out)
```

## kfoldRSF *Perform kfold cross-validation on a RSF output.*

### Description

Perform kfold cross-validation on a RSF output. Similar to what is recommended in Boyce 2002. Function developped with Mathieu Basille

### Usage

```
kfoldRSF(
  mod,
  k = 5,
  nrepet = 10,
  nbins = 10,
  jitter = TRUE,
  random = TRUE,
  method = method,
  x = m,
  form_ls = ls,
  reproducible = TRUE
)
```

### Arguments

| | |
|---|---|
| `mod` | A RSF model (glm or glmer) |
| `k` | number of fold (default = 5) |
| `nrepet` | Number of repetitions (default =10) |
| `nbins` | Number of bins (default =10) |
| `jitter` | Logical, whether to add some random noise to the predictions (useful when the model is fitted on categorical variables, which can produces error in the ranking process). |
| `reproducible` | Logical, whether to use a fixed seed for each repetition. |

### Value

A data frame with the correlations (`cor`) and the type of value (`type`).

## kfold_ind *Perform kfold cross-validation at the individual level .*

### Description

Perform kfold cross-validation at the individual level and return histogram, mean kfold accros individual and min/max value

### Usage

```
kfold_ind(
  m = 1,
  mod_ls,
  ls = ls,
  cutoff = 0,
  k = 5,
  nrepet = 5,
  nbins = 10,
  grph = T
)
```

### Arguments

| | |
|---|---|
| `m` | model number (based on number in list of formula provided to rsf_ind) |
| `mod_ls` | A list of list of model generated by rsf_ind |
| `cutoff` | A cutoff value to exclude individuals with bad fit, default = -1 indicating model that did not converge will be excluded. Values > 0 will exclude based on coefficient |
| `k` | number of fold (default = 5) |
| `nrepet` | Number of repetitions (default =10) |
| `nbins` | Number of bins (default =10) |
| `jitter` | Logical, whether to add some random noise to the predictions (useful when the model is fitted on categorica variables, which can produces error in the ranking process). |
| `reproducible` | Logical, whether to use a fixed seed for each repetition. |

### Value

A data frame with the correlations (`cor`) and the type of value (`type`).

### Examples

```
data(goats)
ls1<-list()
ls1[[1]]<-as.formula(STATUS~ELEVATION+SLOPE+ET+ASPECT+HLI+TASP)
ls1[[2]]<-as.formula(STATUS~ET+ASPECT+HLI+TASP)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
kfold_ind(m=1, out, ls=ls1)
```

---

## pop_avg — *Extract population average of top model and extract individual coefficients*

---

### Description

Extract population average of top model and extract individual coefficients. Population average can be calculated based on bootstrap (Prokopenko et al. 2016 JAppEco) or weighted based on standard errors (Murtaugh 2007 Ecology)

## Usage

```
pop_avg(
  m = 1,
  mod_ls,
  cutoff = 0,
  nudge = 0.01,
  method = "murtaugh",
  nboot = 1000,
  id_year = F
)
```

## Arguments

| | |
|---|---|
| `m` | model number (based on number in list of formula provided to rsf_ind) |
| `mod_ls` | A list of list of model generated by rsf_ind |
| `cutoff` | A cutoff value to exclude individuals with bad fit, default = -1 indicating model that did not converge will be excluded. Values > 0 will exclude based on coefficient |
| `method` | If = "boot", population average is based on bootstrap, if = "murtaugh" based on standard errors weighting. See Prokopenko et al 2016 or Murtaugh 2007 for details. |
| `nboot` | Number of bootstrap iterations, default = 1000. Only applicable if method = "boot". |
| `id_year` | Whether id_year (instead of individual) are provided. Individual and year needs to be separated by an underscore for the function to work properly. |

## Value

A list containing a table population average with confidence intervals and a table of individual coefficients

## Examples

```
data(goats)
ls1<-list()
ls1[[1]]<-as.formula(STATUS~ELEVATION+SLOPE+ET+ASPECT+HLI+TASP)
ls1[[2]]<-as.formula(STATUS~ET+ASPECT+HLI+TASP)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
pop_avg(m=1, out, method="murtaugh")
```

---

`resample_rsf` *Resample a RSF table to keep constant ratio of used/random locations across individuals*

---

## Description

Resample a RSF table to keep constant ratio of used/random locations across individuals. Resampling is done with replacement.

## Usage

```
resample_rsf(data, id = "Id_Year", value = "Value", ratio = 3)
```

## Arguments

| | |
|---|---|
| `data` | The RSF dataset to resample |
| `id` | A vector of individual for each observation |
| `value` | A vector indicating if each observation is used (=1) or random(=0) |
| `ratio` | The ratio of random:used location (default =3, meaning 3 random locations for each used location) |

## Value

A RSF dataset

---

`rf` *Extraction of proximity from random forest classification*

---

## Description

Apply random forest classification

## Usage

```
rf(coef, ntree = 10000, ...)
```

## Arguments

| | |
|---|---|
| `coef` | A matrix of model coefficient (from function ind_coef) |

## Value

A proximity matrix

## Examples

```
data(goats)
ls1<-list()
ls1[[1]]<-as.formula(STATUS~ELEVATION+SLOPE+ET+ASPECT+HLI+TASP)
ls1[[2]]<-as.formula(STATUS~ET+ASPECT+HLI+TASP)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
coef<-ind_coef(m=1, out)
prox<-rf(coef)
```

## rm_bad_fit — *Remove potential individual with bad fits based on coefficients*

### Description

Remove potential individual with bad fits based on coefficients

### Usage

```
rm_bad_fit(mod_ls, cutoff = 1000)
```

### Arguments

| | |
|---|---|
| `mod_ls` | A list of list of model generated by rsf_ind |
| `cutoff` | Value a coeffiecient may take to indicate bad fit (default=1000) |

### Value

A list excluding individual with bad fits.

## rm_bad_fit1 — *Remove potential individual with bad fits based on coefficients for a specific model*

### Description

Remove potential individual with bad fits based on coefficients

### Usage

```
rm_bad_fit1(mod_ls, cutoff = 1000, m = 1)
```

### Arguments

| | |
|---|---|
| `mod_ls` | A list of list of model generated by rsf_ind |
| `cutoff` | Value a coeffiecient may take to indicate bad fit (default=1000) |
| `m` | model number (based on number in list of formula provided to rsf_ind) |

### Value

A list excluding individual with bad fits.

## `rm_conv_fit` *Remove potential individual with bad fits based on model convergence*

### Description

Remove potential individual with bad fits based on model convergence

### Usage

```
rm_conv_fit(mod_ls)
```

### Arguments

| | |
|---|---|
| `mod_ls` | A list of list of model generated by rsf_ind |

### Value

A list excluding individual with bad fits.

## `rm_conv_fit1` *Remove potential individual with bad fits based on model convergence for a specific model*

### Description

Remove potential individual with bad fits based on model convergence

### Usage

```
rm_conv_fit1(mod_ls, m = 1)
```

### Arguments

| | |
|---|---|
| `m` | model number (based on number in list of formula provided to rsf_ind) |

### Value

A list excluding individual with bad fits.

## rsf_ind — *Apply a list of candidate models to multiple individuals*

### Description

Apply rsf_mod to each individual of a dataset

### Usage

```
rsf_ind(id, data, form_ls, cleanModel = F, method = "glm.fit")
```

### Arguments

| | |
|---|---|
| `id` | A vector indicating the individuals |
| `data` | The dataset containing all data |
| `form_ls` | A list of formulas for the different candidate models |
| `method` | Weither typical glm or bias-reduction glm should be fitted (default="glm.fit) (see package brglm) |
| `cleamModel` | Whether the model should be "cleaned" to save memory space (default = F) |

### Value

A list of list of glm objects

### Examples

```
data(goats)
ls1<-list()
ls1[[1]]<-as.formula(STATUS~ELEVATION+SLOPE+ET+ASPECT+HLI+TASP)
ls1[[2]]<-as.formula(STATUS~ET+ASPECT+HLI+TASP)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
```

## rsf_mod — *Apply a list of candidate RSF models to a single individual*

### Description

Apply a list of candidate RSF models to a single individual

### Usage

```
rsf_mod(sub, form_ls, cleanModel = F, method = method)
```

### Arguments

| | |
|---|---|
| `sub` | A subset of data from a single individual |
| `form_ls` | A list of formulas for the different candidate models |
| `method` | Weither typical glm or bias-reduction glm should be fitted (see package brglm) |
| `cleamModel` | Whether the model should be "cleaned" to save memory space |

### Value

A list of glm objects

---

`simu_avg` *Un-weighted population average based on simulated coefficients*

---

### Description

Calculate population average for each replicate of simulated coefficients

### Usage

```
simu_avg(simu)
```

### Arguments

| | |
|---|---|
| `simu` | An array of simulated individual coefficients based on their uncertainties(output of simu_coefs) |

### Value

A matrix of population averages (one value for each covariates and replicates)

### Examples

```
data(goats)
ls1<-list()
ls1[[1]]<-as.formula(STATUS~ELEVATION+SLOPE+ET+ASPECT+HLI+TASP)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
coef<-ind_coef(m=1, out)
se<-ind_se(m=1, out)
simu<-simu_coefs(coef, se, n=100)
avg<-simu_avg(simu)
colnames(avg)<-names(coef)
head(avg)
apply(avg, 2, quantile, na.rm=T) #Show variation around estimate of each covariate
colMeans(avg) #Calculate average population average
```

---

`simu_coefs` *Simulate normally-distributed individual coefficients from RSF/SSF based on their standard errors*

---

### Description

Simulate individual coefficients based on standard errors (to propagate uncertainty)

### Usage

```
simu_coefs(coef, se, n = 1000)
```

## Arguments

| | |
|---|---|
| `coef` | A matrix of individual coefficients (output of ind_se) |
| `n` | Number of random coefficients to generate for each individual (default=1000) |

## Value

An array of individual coefficients

## Examples

```
data(goats)
ls1<-list()
ls1[[1]]<-as.formula(STATUS~ELEVATION+SLOPE+ET+ASPECT+HLI+TASP)
ls1[[2]]<-as.formula(STATUS~ET+ASPECT+HLI+TASP)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
coef<-ind_coef(m=1, out)
se<-ind_se(m1, out)
simu<-simu_coefs(coef, se, n=100)
```

---

`simu_cons2` *Temporal consistency (with TWO time periods) based on simulated coefficients*

---

## Description

Calculate temporal consistency for each replicate of simulated coefficients

## Usage

```
simu_cons2(simu, col1, col2)
```

## Arguments

| | |
|---|---|
| `simu` | An array of simulated individual coefficients based on their uncertainties(output of simu_coefs) |
| `col1` | Column of the variable of interest for the 1st temporal period |
| `col2` | Column of the variable of interest for the 2nd temporal period |

## Value

A matrix of consistency (one value for each covariates and replicates)

## Examples

```
data(goats)
goats$Season<-c("1", "2")
ls1<-list()
ls1[[1]]<-as.formula(STATUS~(ELEVATION+SLOPE+ET+ASPECT+HLI+TASP):Season)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
coef<-ind_coef(m=1, out)
se<-ind_se(m=1, out)
simu<-simu_coefs(coef, se, n=100)
```

```
head(coef)
cons_elevation<-simu_cons2(simu, 3, 4) #Calculate specialization for elevation covariate
quantile(cons_elevation) #Show variation around estimate of elevation covariate
mean(cons_elevation) #Calculate average consistency for elevation covariate
```

---

## simu_cons3 *Temporal consistency (with THREE time periods) based on simulated coefficients*

---

### Description

Calculate temporal consistency for each replicate of simulated coefficients

### Usage

```
simu_cons3(simu, col1, col2, col3)
```

### Arguments

| | |
|---|---|
| simu | An array of simulated individual coefficients based on their uncertainties(output of simu_coefs) |
| col1 | Column of the variable of interest for the 1st temporal period |
| col2 | Column of the variable of interest for the 2nd temporal period |
| col3 | Column of the variable of interest for the 3rd temporal period |

### Value

A matrix of consistency (one value for each covariates and replicates)

### Examples

```
data(goats)
goats$Season<-c("1", "2", "3")
ls1<-list()
ls1[[1]]<-as.formula(STATUS~(ELEVATION+SLOPE+ET+ASPECT+HLI+TASP):Season)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
coef<-ind_coef(m=1, out)
se<-ind_se(m=1, out)
simu<-simu_coefs(coef, se, n=100)
head(coef)
cons_elevation<-simu_cons3(simu, 3, 4,5) #Calculate specialization for elevation covariate
quantile(cons_elevation) #Show variation around estimate of elevation covariate
mean(cons_elevation) #Calculate average consistency for elevation covariate
```

## simu_rev2 *Temporal reversal (with TWO time periods) based on simulated coefficients*

### Description

Calculate temporal reversal for each replicate of simulated coefficients

### Usage

```
simu_rev2(simu, col1, col2)
```

### Arguments

| | |
|---|---|
| simu | An array of simulated individual coefficients based on their uncertainties(output of simu_coefs) |
| col1 | Column of the variable of interest for the 1st temporal period |
| col2 | Column of the variable of interest for the 2nd temporal period |

### Value

A matrix of reversal (one value for each covariates and replicates)

### Examples

```
data(goats)
goats$Season<-c("1", "2")
ls1<-list()
ls1[[1]]<-as.formula(STATUS~(ELEVATION+SLOPE+ET+ASPECT+HLI+TASP):Season)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
coef<-ind_coef(m=1, out)
se<-ind_se(m=1, out)
simu<-simu_coefs(coef, se, n=100)
head(coef)
rev_elevation<-simu_rev2(simu, 3, 4) #Calculate specialization for elevation covariate
quantile(rev_elevation) #Show variation around estimate of elevation covariate
mean(rev_elevation) #Calculate average reversal for elevation covariate
```

## simu_rev3 *Temporal reversal (with THREE time periods) based on simulated coefficients*

### Description

Calculate temporal reversal for each replicate of simulated coefficients

### Usage

```
simu_rev3(simu, col1, col2, col3)
```

### Arguments

| | |
|---|---|
| `simu` | An array of simulated individual coefficients based on their uncertainties(output of simu_coefs) |
| `col1` | Column of the variable of interest for the 1st temporal period |
| `col2` | Column of the variable of interest for the 2nd temporal period |
| `col3` | Column of the variable of interest for the 3rd temporal period |

### Value

A matrix of reversal (one value for each covariates and replicates)

### Examples

```
data(goats)
goats$Season<-c("1", "2", "3")
ls1<-list()
ls1[[1]]<-as.formula(STATUS~(ELEVATION+SLOPE+ET+ASPECT+HLI+TASP):Season)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
coef<-ind_coef(m=1, out)
se<-ind_se(m=1, out)
simu<-simu_coefs(coef, se, n=100)
head(coef)
rev_elevation<-simu_rev3(simu, 3, 4,5) #Calculate specialization for elevation covariate
quantile(rev_elevation) #Show variation around estimate of elevation covariate
mean(rev_elevation) #Calculate average reversal for elevation covariate
```

---

## `simu_sd` *Individual variation (Heterogeneity) based on simulated coefficients*

---

### Description

Calculate heterogeneity for each replicate of simulated coefficients

### Usage

```
simu_sd(simu)
```

### Arguments

| | |
|---|---|
| `simu` | An array of simulated individual coefficients based on their uncertainties(output of simu_coefs) |

### Value

A matrix of heterogeneity (one value for each covariates and replicates)

## Examples

```
data(goats)
ls1<-list()
ls1[[1]]<-as.formula(STATUS~ELEVATION+SLOPE+ET+ASPECT+HLI+TASP)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
coef<-ind_coef(m=1, out)
se<-ind_se(m=1, out)
simu<-simu_coefs(coef, se, n=100)
sd<-simu_sd(simu)
colnames(sd)<-names(coef)
head(sd)
apply(sd, 2, quantile, na.rm=T) #Show variation around estimate of each covariate
colMeans(sd) #Calculate average heterogeneity for each covariate
```

---

`simu_spe` *Specialization based on simulated coefficients*

---

## Description

Calculate specialization for each replicate of simulated coefficients

## Usage

```
simu_spe(simu)
```

## Arguments

| | |
|---|---|
| `simu` | An array of simulated individual coefficients based on their uncertainties(output of simu_coefs) |

## Value

A matrix of specialization (one value for each covariates and replicates)

## Examples

```
data(goats)
ls1<-list()
ls1[[1]]<-as.formula(STATUS~ELEVATION+SLOPE+ET+ASPECT+HLI+TASP)
out<-rsf_ind(goats$ID, data=goats, form_ls=ls1)
coef<-ind_coef(m=1, out)
se<-ind_se(m=1, out)
simu<-simu_coefs(coef, se, n=100)
spe<-simu_spe(simu)
colnames(spe)<-names(coef)
head(spe)
apply(spe, 2, quantile, na.rm=T) #Show variation around estimate of each covariate
colMeans(spe) #Calculate average specialization for each covariate
```

## ssf_ind — *Apply a list of SSF candidate models to multiple individuals*

### Description

Apply ssf_mod to each individual of a dataset

### Usage

```
ssf_ind(id, data, form_ls, cleanModel = T, method = "approximate")
```

### Arguments

| | |
|---|---|
| `id` | A vector indicating the individuals |
| `data` | The dataset containing all data |
| `form_ls` | A list of formulas for the different candidate models |
| `method` | Whether exact or approximate ML should be performed (see package survival) |
| `cleamModel` | Whether the model should be "cleaned" to save memory space (default = F) |

### Value

A list of of coxph objects

## ssf_mod — *Apply a list of candidate SSF models to a single individual*

### Description

Apply a list of candidate SSF models to a single individual

### Usage

```
ssf_mod(sub, form_ls, cleanModel = F, method = "approximate")
```

### Arguments

| | |
|---|---|
| `sub` | A subset of data from a single individual |
| `form_ls` | A list of formulas for the different candidate models |
| `method` | Whether exact or approximate ML should be performed (see package survival) |
| `cleamModel` | Whether the model should be "cleaned" to save memory space |

### Value

A list of coxph objects

# Index