Package ‘AMPLE’
September 16, 2019

Title  Common Code for the 'AMPED' and 'PIMPLE' 'shiny' Apps for Stakeholder Engagement and Harvest Strategies
Version  0.0.2
Maintainer  Finlay Scott <finlays@spc.int>
Description  A collection of common code for the 'AMPED' and 'PIMPLE' 'shiny' applications for stakeholder engagement in developing fisheries harvest strategies. The package is not that useful on its own but provides common functionality including plots, stochasticity options, life history options and a simple surplus production model.
License  GPL-3
Depends  shiny (>= 1.3.2)
Imports  ggplot2 (>= 3.1.1), dplyr (>= 0.8.1), tidyr (>= 0.8.3), wesanderson (>= 0.3.6), shinyjs (>= 1.0)
Encoding  UTF-8
LazyData  true
RoxygenNote  6.1.1
Suggests  testthat (>= 2.1.0)
NeedsCompilation  no
Author  Finlay Scott [aut, cre]
Repository  CRAN
Date/Publication  2019-09-16 04:10:03 UTC

R topics documented:

 AMPED modules ................................................................. 2
 amped_maintainer_and_licence ........................................... 4
 Comparison plots ............................................................ 4
 create_stock ................................................................. 7
 current_pi_table ............................................................ 9
 Front page plots ............................................................ 11
 get_time_periods .......................................................... 14
 project ............................................................... 15

Index  17
AMPED modules

Modules for common features of the AMPED Shiny apps.

Description

stoch_params_setterUI() is the interface for the stochasticity options.
stoch_params_setter() does the setting.
set_stoch_params() Sets up default values. Only exported to get the examples to work.
stock_params_setterUI() is the interface for the stock options (e.g. life history and exploitation status).
stock_params_setter() does the setting.
get_lh_params() Sets up default values. Only exported to get the examples to work.
mp_params_setterUI() is the interface for the HCR options
mp_params_setter() does the setting.
mp_params_switcheroo() Sets up default values. Only exported to get the examples to work.

Usage

stoch_params_setterUI(id, show_biol_prod_sigma = TRUE,
                      show_biol_est_sigma = TRUE, show_biol_est_bias = TRUE,
                      init_prod_sigma = 0, init_est_sigma = 0, init_est_bias = 0,
                      show_var = TRUE)

stoch_params_setter(input, output, session)

set_stoch_params(input)

stock_params_setterUI(id)

stock_params_setter(input, output, session)

get_lh_params(input)

mp_params_setterUI(id, mp_visible = NULL,
                    title = "Select the type of HCR you want to test. ",
                    init_thresh_max_catch = 140, init_thresh_below = 0.5,
                    init_constant_catch = 50, init_constant_effort = 1,
                    input_label = "HCR Type")

mp_params_setter(input, output, session)

mp_params_switcheroo(input)
AMPED modules

Arguments

- **id**: Shiny magic id
- **show_biol_prod_sigma**: Show the biological productivity variability option.
- **show_biol_est_sigma**: Show the estimation variability option.
- **show_biol_est_bias**: Show the estimation bias option.
- **init_prod_sigma**: Default value for biological productivity variability.
- **init_est_sigma**: Default value for estimation variability.
- **init_est_bias**: Default value for estimation bias.
- **show_var**: Show the variability options.
- **input**: The UI input.
- **output**: Shiny magic.
- **session**: Shiny magic.
- **mp_visible**: Which HCR types to show.
- **title**: The title.
- **init_thresh_max_catch**: Initial value of the maximum catch for the catch threshold HCR.
- **init_thresh_belbow**: Initial value of the belbow for the catch threshold HCR.
- **init_constant_catch**: Initial value of constant catch for the constant catch HCR.
- **init_constant_effort**: Initial value of constant effort for the constant effort HCR.
- **input_label**: The label of the menu.

Value

A taglist

A list of stochasticity options.

A taglist

A list of stock options.

A taglist

A list of HCR options.

Examples

```r
## Not run:
# Put something like this in the Shiny apps UI code
mp_params_setterUI("mpparams", mp_visible=c("Threshold catch", "Constant catch", "Threshold effort", "Constant effort"))
```
# And then something like this in the Shiny server code
get_mp_params <- callModule(mp_params_setter, "mpparams")

## End(Not run)

amed_maintainer_and_licence

*Maintainer, licence and SPC about information.*

**Description**

Show the maintainer and licence for use in the AMPED PIMPLE applications. Also show the 'About SPC' information.

**Usage**

amed_maintainer_and_licence()

pimple_maintainer_and_licence()

spc_about()

**Value**

A shiny.tag for use in Shiny apps

**Examples**

amed_maintainer_and_licence()

pimple_maintainer_and_licence()

spc_about()

---

**Comparison plots**

*pitable*

**Description**

pitable() is not a plot but a table comparing PIs across HCRs and periods. Only pass in 1 time period at a time.

plot_majuro() plots a Majuro plot.

quantile_plot() plots times series of indicators for each HCR.

myboxplot() plots box plots or bar charts of the median values of the indicators for each HCR.

myradar() plots a radar chart in each time period, scaled either by the maximum or by ranking.

hcrplot() plots the shape of each HCR.

hcr_histo_plot() plots histograms of the HCR outputs in each time period.
Comparison plots

Usage

```r
pitable(dat, percentile_range = c(20, 80))

plot_majuro(dat, percentile_range = c(20, 80), hcr_choices, stock_params)

quantile_plot(dat, hcr_choices, wormdat = NULL, alpha20_80 = 0.6, 
linetype_worm = 1, percentile_range = c(20, 80), 
colour_worm = "black", size_worm = 0.3, add_start_line = TRUE, 
time_period_lines = TRUE, short_term = 2016:2024, 
medium_term = 2025:2033, long_term = 2034:2042, 
last_plot_year = 2042, show_spaghetti = FALSE)

myboxplot(dat, hcr_choices, plot_type = "median_bar")

myradar(dat, hcr_choices, scaling = "scale", polysize = 2)

hcr_plot(hcr_choices, hcr_shape, hcr_points, lrp, trp, blacklinesize = 4, 
linesize = 3, pointsize = 4.2, stroke = 3)

hcr_histo_plot(hcr_choices, histodat)
```

Arguments

dat The data.frame with the data to be plotted.

percentile_range A vector of length with minimum and maximum percentile range to plot.

hcr_choices The names of the HCRs to plot.

stock_params A vector of life history and stochastic parameters.

wormdat Data set of the spaghetti (worms).

alpha20_80 Alpha of the ribbons.

linetype_worm Line type of the spaghetti.

colour_worm Colour of the spaghetti.

size_worm Thickness of the spaghetti.

add_start_line Add a line to the start of the projection (TRUE / FALSE).

time_period_lines Add a lines to show the time periods (TRUE / FALSE).

short_term Year range for the short-term.

medium_term Year range for the medium-term.

long_term Year range for the long-term.

last_plot_year Last year to plot.

show_spaghetti Show the spaghetti (worms) (TRUE / FALSE).

plot_type Either median_bar or box.

scaling Either scale or rank.
polysize  The thickness of the radar.
hcr_shape  The shape parameters of each HCR.
hcr_points  Optionally show the bits of the HCR that triggered (currently not used).
lrp  The limit reference point.
trp  The target reference point.
blacklinesize  Size of the underlying black lines.
linesize  Size of the coloured lines.
pointsize  Size of the points (currently not used).
stroke  Mmmmm, stroking.
histodat  Data for the histograms.

Value

A data.frame to be shown as a table.
A ggplot2 plot object.
A ggplot2 plot object.
A ggplot2 plot object.
A ggplot2 plot object.
A ggplot2 plot object.
A ggplot2 plot object.

Examples

# Set up all the bits for a projection - should be done inside a Shiny app
# Management procedure bits - should come from Shiny app inputs
input_mp <- list(
  blim_belbow = c(0.2, 0.5),
  cmin_cmax = c(10, 140),
  constant_catch_level = 50,
  constant_effort_level = 1,
  emin_emax = c(0.1, 0.5),
  hcr_type = "threshold_catch"
)
mp_params <- mp_params_switcheroo(input_mp)

# Stochasticity bits - should come from Shiny app inputs
input_stoch <- list(
  biol_est_bias = 0,
  biol_est_sigma = 0.2,
  biol_prod_sigma = 0.2,
  show_var <- TRUE)
stoch_params <- set_stoch_params(input_stoch)

# Life history bits - should come from Shiny app inputs
input_lh <- list(
create_stock

Functions for creating and filling stock objects.

Description

A stock is made up of biomass, hcr_ip, hcr_op, effort and catch. create_stock() makes the empty reactive stock object for AMPED apps. It has the elements but they are empty.

reset_stock() clears out an existing stock and refills the initial period depending on the stock status option (over, fully, underexploited).

Usage

create_stock()
reset_stock(stock, stock_params, mp_params, app_params, initial_biomass, nyears, niters)

**Arguments**

stock A list with elements biomass, hcr_ip, hcr_op, effort and catch.
stock_params A vector of life history and stochastic parameters.
mp_params A vector of management procedure parameters.
app_params A vector of application parameters.
initial_biomass The initial biomass of the population.
nyears The numbers of years to set the stock up for.
niters The numbers of iters to set the stock up for.

**Value**

A reactiveValues object with bits for the stock.
A filled stock object (a reactiveValues object with bits for the stock).

**Examples**

# Making a NULL stock
create_stock()

# Resetting a stock
# Management procedure bits - should come from Shiny app inputs
input_mp <- list(
  blim_below = c(0.2, 0.5),
  cmin_cmax = c(10, 140),
  constant_catch_level = 50,
  constant_effort_level = 1,
  emin_emax = c(0.1, 0.5),
  hcr_type = "threshold_catch"
)
mp_params <- mp_params_switcheroo(input_mp)

# Stochasticity bits - should come from Shiny app inputs
input_stoch <- list(
  biol_est_bias = 0,
  biol_est_sigma = 0,
  biol_prod_sigma = 0,
  show_var <- TRUE
)
stoch_params <- set_stoch_params(input_stoch)

# Life history bits - should come from Shiny app inputs
input_lh <- list(
  stock_history = "fully",
  stock_lh = "medium"
)
lh_params <- get_lh_params(input_lh)
# Stitch together and make other parameters - should be inside an Shiny app
stock_params <- c(stoch_params, lh_params)
app_params <- list(initial_year = 2009, last_historical_timestep = 10)

# Make the null stock and fill it up
# In a Shiny app use the create_stock() function but cannot do here so make an equivalent object
stock <- list(biomass = NULL, hcr_ip = NULL, hcr_op = NULL, effort = NULL, catch = NULL)

# Reset the stock
stock <- reset_stock(stock = stock, stock_params = stock_params, mp_params = mp_params,
                     app_params = app_params, initial_biomass = stock_params$b0, nyears = 20, niters = 10)

---

**current_pi_table**

Routines for calculating and displaying various performance indicators.

**Description**

current_pi_table() takes the processed indicators and formats them into a table.
replicate_table() shows the final values for SB/SBF=0, catch and relative CPUE for each replicate.
get_projection_pis() get the performance indicators for the Introduction to Projections AMPED app
get_summaries() gets the current indicators including SB/SBF=0, catch and the probability of being above the limit reference point.

**Usage**

current_pi_table(dat, app_params, years, percentile_range = c(20, 80),
                 piname_choice = c("SB/SBF=0", "Prob. SB>LRP", "Catch", "Relative CPUE",
                                  "Catch variability", "Catch stability", "Relative effort",
                                  "Relative effort variability", "Relative effort stability",
                                  "Proximity to TRP"))

replicate_table(stock, app_params, stock_params, percentile_range = c(20, 80))

get_projection_pis(stock, stock_params, app_params, current_timestep)

get_summaries(stock, stock_params, app_params, quantiles)

**Arguments**

dat          A data.frame with the 20th, 80th and 50th percentile value of each indicator.
app_params   A vector of application parameters.
years        A character vector of the years in the simulation.
percentile_range          A vector of length with minimum and maximum percentile range to plot.
piname_choice | A character vector of the indicator names to be included in the table.
stock | A list with elements biomass, hcr_ip, hcr_op, effort and catch.
stock_params | A vector of life history and stochasticity parameters.
current_timestep | The current timestep.
quantiles | The quantiles to calculate the indicators over.

Value

Various data.frames with summaries in different formats.

Examples

# Set up all the bits for a projection - should be done inside a Shiny app
# Management procedure bits - should come from Shiny app inputs
input_mp <- list(
  blim_belbow = c(0.2, 0.5),
  cmin_cmax = c(10, 140),
  constant_catch_level = 50,
  constant_effort_level = 1,
  emin_emax = c(0.1, 0.5),
  hcr_type = "threshold_catch")
mp_params <- mp_params_switcheroo(input_mp)

# Stochasticity bits - should come from Shiny app inputs
input_stoch <- list(
  biol_est_bias = 0,
  biol_est_sigma = 0,
  biol_prod_sigma = 0,
  show_var <- TRUE)
stoch_params <- set_stoch_params(input_stoch)

# Life history bits - should come from Shiny app inputs
input_lh <- list(
  stock_history = "fully",
  stock_lh = "medium")
vh_params <- get_lh_params(input_lh)

# Stitch together and make other parameters - should be inside an Shiny app
stock_params <- c(stock_params, lh_params)
app_params <- list(initial_year = 2009, last_historical_timestep = 10)

# Make the null stock and fill it up
# In a Shiny app use the create_stock() function but cannot do here so make an equivalent stock
stock <- list(biomass = NULL, hcr_ip = NULL, hcr_op = NULL, effort = NULL, catch = NULL)
stock <- reset_stock(stock = stock, stock_params = stock_params, mp_params = mp_params, 
app_params = app_params, initial_biomass = stock_params$b0, nyears = 20, niters = 10)
# Finally project over the timesteps
out <- project(stock, timesteps = c(11,20), stock_params = stock_params, mp_params = mp_params, 
app_params = app_params)
# Get the summaries
Front page plots

pisums <- get_summaries(stock=out, stock_params=stock_params, app_params=app_params,
  quantiles=c(0.01, 0.05, 0.20, 0.5, 0.80, 0.95, 0.99))
# Get the current PI table in a neat format from one of the summary tables
current_pi_table(dat=pisums$periodqs, app_params=app_params,
  years=dimnames(stock$biomass)$year,
  piname_choice=c("SB/SBF=0", "Prob. SB>LRP", "Catch"))

# Get the PIs for the Introduction to Projections app
get_projection_pis(stock=out, stock_params=stock_params, app_params=app_params, current_timestep=15)

Front page plots  Plots for the front page of the Introduction to HCRs and other AMPED apps.

Description

plot_hcr() plots the shape of the HCR.
plot_biomass() plots time series of 'true' and observed depletion (SB/SBF=0).
plot_catch() plots time series of catches.
plot_projection() plots time series of depletion, catch and relative effort for the Introduction to Projections app.
plot_hcr_intro_arrow() draws an arrow between the depletion plot and the HCR plot for the Introduction to HCRs app.
plot_metric_with_histo() is the generic plotting routine for plotting time series of metrics with histograms of the final time step.
plot_kobe_majuro_projections() plots a Kobe or Majuro plot for the Introduction to Projections app.
plot_yieldcurve_projections() plots a yield curve (catch against effort) for the Introduction to Projections app.

Usage

plot_hcr(stock, stock_params, mp_params, app_params, timestep = NULL,
  show_last = TRUE)

plot_biomass(stock, stock_params, mp_params, timestep = NULL,
  show_last = TRUE, max_spaghetti_iters = 50, quantiles,
  nspaghetti = 5, add_grid = TRUE, xlab = "Year",
  ghost_col = "grey", last_col = "blue", ylim = c(0, 1), ...)

plot_catch(stock, stock_params, mp_params, app_params = NULL,
  timestep = NULL, show_last = TRUE, max_spaghetti_iters = 50,
  quantiles, nspaghetti = 5, add_grid = TRUE, xlab = "Year",
  ghost_col = "grey", true_col = "black", ...)

plot_projection(stock, stock_params, mp_params, app_params = NULL,
Front page plots

show_last = TRUE, max_spaghetti_iters = 50, quantiles, nspaghetti = 5, add_grid = TRUE, ...)

plot_hcr_intro_arrow(stock, timestep)

plot_metric_with_histo(stock, stock_params, mp_params, metric, app_params = NULL, show_last = TRUE, percentile_range = c(20, 80))

plot_kobe_majuro_projections(stock, stock_params, choice = "kobe")

plot_yieldcurve_projections(stock, stock_params, app_params)

Arguments

stock          A list with elements biomass, hcr_ip, hcr_op, effort and catch.
stock_params   A vector of life history and stochasticy parameters.
mp_params      A vector of management procedure parameters.
app_params     A vector of application parameters.
timestep       The current timestep (optional).
show_last      Show the previous iters as ghosts (optional).
max_spaghetti_iters
               The number of iterations to show as spaghetti before ribbons are shown.
quantiles      Quantiles of the ribbons.
nspaghetti     The number of spaghetti iterations to plot on top of the ribbons.
add_grid       Add a grid.
xlab           The x-label.
ghost_col      Colours of the ghost iterations.
last_col       Colours of the last iteration.
ylim           Y limits
...            Other arguments to pass to the plot() function.
true_col       Colour of the current iteration.
metric         The name of the metric to plot (catch, biomass or relcpue).
percentile_range
               A vector of length with minimum and maximum percentile range to plot.
choice         Either kobe or majuro.

Value

A plot
A plot
A plot
A plot
# Set up all the bits for a projection - should be done inside a Shiny app
# Management procedure bits - should come from Shiny app inputs
input_mp <- list(
  blim_bellow = c(0.2, 0.5),
  cmin_cmax = c(10, 140),
  constant_catch_level = 50,
  constant_effort_level = 1,
  emin_emax = c(0.1, 0.5),
  hcr_type = "threshold_catch")
mp_params <- mp_params_switcheroo(input_mp)

# Stochasticity bits - should come from Shiny app inputs
input_stoch <- list(
  biol_est_bias = 0,
  biol_est_sigma = 0.2,
  biol_prod_sigma = 0.2,
  show_var <- TRUE)
stoch_params <- set_stoch_params(input_stoch)

# Life history bits - should come from Shiny app inputs
input_lh <- list(
  stock_history = "fully",
  stock_lh = "medium")
lh_params <- get_lh_params(input_lh)

# Stitch together and make other parameters - should be inside an Shiny app
stock_params <- c(stoch_params, lh_params)
app_params <- list(initial_year = 2009, last_historical_timestep = 10)

# Make the null stock and fill it up
# In a Shiny app use the create_stock() function but cannot do here so make an equivalent object
stock <- list(biomass = NULL, hcr_ip = NULL, hcr_op = NULL, effort = NULL, catch = NULL)
stock <- reset_stock(stock = stock, stock_params = stock_params, mp_params = mp_params,
  app_params = app_params, initial_biomass = stock_params$b0, nyears = 20, niter = 100)

# Finally project over the timesteps
stock <- project(stock, timesteps = c(11,20), stock_params = stock_params,
  mp_params = mp_params, app_params = app_params)

# The plots
# Plot the HCR
plot_hcr(stock=stock, stock_params=stock_params, mp_params=mp_params, app_params=app_params)
# Plot biomass timeseries
plot_biomass(stock=stock, stock_params=stock_params, mp_params=mp_params, quantiles=c(0.2,0.8))
# Plot catch timeseries
plot_catch(stock=stock, stock_params=stock_params, mp_params=mp_params, quantiles=c(0.2,0.8))
# Plot the projection (biomass, catch and rel cpue)
plot_projection(stock=stock, stock_params=stock_params, mp_params=mp_params,
    app_params=app_params, quantiles=c(0.2,0.8))

# The arrow connecting the HCR to the biomass
plot_hcr_intro_arrow(stock=stock, timestep=15)

# Time series with a histogram on the end
plot_metric_with_histo(stock=stock, stock_params=stock_params, mp_params=mp_params,
    metric="catch", app_params=app_params)

# Kobe or Majuro plot
# Just a few iters for efficiency
stock2 <- lapply(stock, '[',1:5,)
plot_kobe_majuro_projections(stock=stock2, stock_params=stock_params, choice="kobe")

# Yield curve
plot_yieldcurve_projections(stock=stock2, stock_params=stock_params, app_params=app_params)

---

get_time_periods

Get timesteps for the short-, medium- and long term time periods.

Description

Used when calculating the performance indicators over the three time periods.

Usage

get_time_periods(app_params, nyears)

Arguments

app_params A vector of application parameters.
nyears The number of years.

Value

A list of the timesteps that make up each time period.

Examples

app_params <- list(initial_year = 2009, last_historical_timestep = 10)
nyears <- 40
get_time_periods(app_params=app_params, nyears=nyears)
**project**  
*Projects the stock over the timesteps given and updates the biomass, HCR ip / op and catches*

**Description**

project() uses a simple biomass dynamic model where the catches or fishing effort are set every timestep by the harvest control rule.

get_hcr_op() evaluates the harvest control rule in a single year (timestep) for multiple iterations.

**Usage**

```r
project(stock, timesteps, stock_params, mp_params, app_params,  
  max_releffort = 10)

get_hcr_op(stock, stock_params, mp_params, yr)
```

**Arguments**

- `stock` A list with elements biomass, hcr_ip, hcr_op, effort and catch.
- `timesteps` The timesteps to project over. A vector of length 2 (start and end).
- `stock_params` A vector of life history and stochasticity parameters.
- `mp_params` A vector of management procedure parameters.
- `app_params` A vector of application parameters.
- `max_releffort` The maximum relative effort.
- `yr` Evaluate the HCR in a particular year (timestep).

**Value**

- A stock object (a reactiveValues object with bits for the stock)
- A vector of outputs from the HCR.

**Examples**

```r
# Set up all the bits for a projection - should be done inside a Shiny app
# Management procedure bits - should come from Shiny app inputs
input_mp <- list(
  blim_belbelow = c(0.2, 0.5),
  cmin_cmax = c(10, 140),
  constant_catch_level = 50,
  constant_effort_level = 1,
  emin_emax = c(0.1, 0.5),
  hcr_type = "threshold_catch")
mp_params <- mp_params_switcheroo(input_mp)

# Stochasticity bits - should come from Shiny app inputs
```
input_stoch <- list(
    biol_est_bias = 0,
    biol_est_sigma = 0,
    biol_prod_sigma = 0,
    show_var <- TRUE)
stoch_params <- set_stoch_params(input_stoch)

# Life history bits - should come from Shiny app inputs
input_lh <- list(
    stock_history = "fully",
    stock_lh = "medium")
lh_params <- get_lh_params(input_lh)

# Stitch together and make other parameters - should be inside an Shiny app
stock_params <- c(stoch_params, lh_params)
app_params <- list(initial_year = 2009, last_historical_timestep = 10)

# Make the null stock and fill it up
# In a Shiny app use the create_stock() function but cannot do here so make an equivalent object
stock <- list(biomass = NULL, hcr_ip = NULL, hcr_op = NULL, effort = NULL, catch = NULL)
stock <- reset_stock(stock = stock, stock_params = stock_params, mp_params = mp_params,
    app_params = app_params, initial_biomass = stock_params$b0, nyears = 20, niters = 10)

# Finally project over the timesteps
out <- project(stock, timesteps = c(11,20), stock_params = stock_params, mp_params = mp_params,
    app_params = app_params)

# Or just get the HCR op in a single timestep
hcr_op <- get_hcr_op(stock=stock, stock_params=stock_params, mp_params=mp_params, yr=10)
Index

AMPED modules, 2
amped_maintainer_and_licence, 4

Comparison plots, 4
create_stock, 7
current_pi_table, 9

Front page plots, 11
get_hcr_op (project), 15
get_lh_params (AMPED modules), 2
get_projection_pis (current_pi_table), 9
get_summaries (current_pi_table), 9
get_time_periods, 14

hcr_histo_plot (Comparison plots), 4
hcr_plot (Comparison plots), 4

mp_params_setter (AMPED modules), 2
mp_params_setterUI (AMPED modules), 2
mp_params_switcheroo (AMPED modules), 2
myboxplot (Comparison plots), 4
myradar (Comparison plots), 4

pimple_maintainer_and_licence
  (amped_maintainer_and_licence), 4
pitable (Comparison plots), 4
plot_biomass (Front page plots), 11
plot_catch (Front page plots), 11
plot_hcr (Front page plots), 11
plot_hcr_intro_arrow (Front page plots), 11
plot_kobe_majuro_projections (Front page plots), 11
plot_majuro (Comparison plots), 4
plot_metric_with_histo (Front page plots), 11
plot_projection (Front page plots), 11
plot_yieldcurve_projections (Front page plots), 11

project, 15
quantile_plot (Comparison plots), 4
replicate_table (current_pi_table), 9
reset_stock (create_stock), 7

set_stoch_params (AMPED modules), 2
spc_about
  (amped_maintainer_and_licence), 4
stoch_params_setter (AMPED modules), 2
stoch_params_setterUI (AMPED modules), 2
stock_params_setter (AMPED modules), 2
stock_params_setterUI (AMPED modules), 2