r4Casal2

C. Marsh & A. Dunn

2023-07-21

Contents

Chapter 1

Welcolme to r4Casal2

This book demonstrates functionality of the `r4Casal2` R package. This is an R package that works with the `Casal2` base R-library found here, although it is advised to use the R-library that is included with the Casal2 binary and usermanual you acquired. The `Casal2` base R-library is responsible for reading in output and interacting with Casal2 configuration files. The `r4Casal2` R package has been built for summarising and visualising objects read in from the base `Casal2` R-library.

This repository is a clone of Casal2 of https://github.com/NIWAFisheriesModelling/r4Casal2. It is intended to have additional bug fixes and enhancements for potential inclusion into the original NIWA codebase.

All functions in this package should be documented using the `roxygen` syntax with input parameters available using the `?` query. For example `?get_fisheries`. To get a list of functions and general info on the package you can use `library(help="r4Casal2")` or see Section 2 for another list

```
library(r4Casal2)
library(Casal2)
library(knitr)
library(ggplot2)
library(dplyr)
library(reshape2)
library(tidyr)
```

The core functionality of `r4Casal2` are its accessor functions. These are functions that will return a specific object from a range of Casal2 objects in long format that are `ggplot`, `dplyr` friendly. Most accessors start with `get_` and should be self explanatory. There are some plotting functions, but I have found that I often want to custom ggplots and so mainly have custom plots. The accessors are coded to deal with three types of outpus. These are;

- `extract.mpd()` where Casal2 has been run with default report style. These objects are of class `casal2MPD` which are set by the Casal2 base function
- `extract.mpd()` where Casal2 has been run with tabular reports `casal2 --tabular` or `casal2 -t`. These objects are of class `casal2TAB` which are set by the Casal2 base function
- `list` this is a list of `casal2MPD` which is a useful format for comparing MPD runs see Section 5

Chapter 2

A list of key functions in the `r4Casal2` package

2.1 Accessor functions

- `get_derived_quantities()` (or in shorthand form `get_dqs()`). These will return all the derived quantities for a model output.
- `get_selectivities` will return a data frame with all the selectivity reports for a model output.
- `get_selectivities_by_year` will return a data frame with all the reports of type `selectivity_by_year` from a model output.
- `get_catchabilities` will return a data frame with all the catchability reports for a model output.
- `get_fisheries` will return a data frame with information from an `instantaneous_mortality` process for a model output.
- `get_BH_recruitment` will return a data frame with information from a `recruitment_beverton_holt` process for a model output.
- `get_abundance_observations` will return a data frame with information from an `abundance` or `biomass` observation for a model output.
- `get_composition_observations` will return a data frame with information from an `proportion_at_length`, `proportion_at_age`, `process_removals_by_age` and `process_removals_by_length` observation for a model output.
- `get_composition_mean_bin` will return a data frame with information from an `proportion_at_length`, `proportion_at_age`,

process_removals_by_age and process_removals_by_length summarised as the mean length or mean age.

- get_tag_recapture_observations will return a data frame with information from an tag_recapture_by_length_for_growth, tag_recapture_by_length and tag_recapture_by_age observation for a model output.
- get_partition will return a data frame with partition data from partition report.
- get_initial_partition will return a data frame with initial partition initialisation_partition report.
- get_profile Will return a data frame for a profile report.
- get_estimated_values Will return a data frame for a estimate_value report.
- get_transformed_parameters Will return a data frame for a parameter_transformations report.
- get_timevarying_parameters Will return a data frame for a time_varying report.
- get_simulated_age_resids Will reformat simulated data read in by the read.simulated.data function.
- get_projections will return a data frame of all projection reports from a model output.
- get_growth will return a data frame of all age_length report from a model output.
- get_covariance will return a data frame of all covariance_matrix report from a model output.

2.2 Other useful functions

- aggregate_objective_report This reformats an objective function report to be "similar" to CASALs output.
- create_simulation_reports This will create a range of @report.type=simulated_observation Casal2 reports that can help set up simulations. See Section 7 on why you want to do this.
- build_assessment_bookdown This will create a bookdown template for an assessment model MPD run.
- summarise_config Will summarize input files see Section 3
- calculate_composition_stage_two_weights Calculates the stage-two weights using Francis [2011] TA1.8 method.

- `get_high_correlations` Returns index of parameters that have high correlations from MPD. This requires the Casal2 model to have reported the `correlation_matrix`
- `run_automatic_reweighting` Automatically apply iterative reweighting methods for a Casal2 model
- `extract_reweighted_mpds` extract all the reweighted mpds that are created by `run_automatic_reweighting`. Useful to then plot the effect of reweighting
- `error_value_table` Create a data.frame of all observations from a casal2 mpd run outlining likelihood type, observation type and error value by year and observation.
- `summarise_estimated_parameters` If a model reports `estimate_summary` this function will extract two data frames that can be used to assess starting values and estimated values along with prior assumptions.
- `plot_profile` Will plot profiles for reports that have been run with `casal2 -p` format.

Chapter 3

Summarise configuration inputs

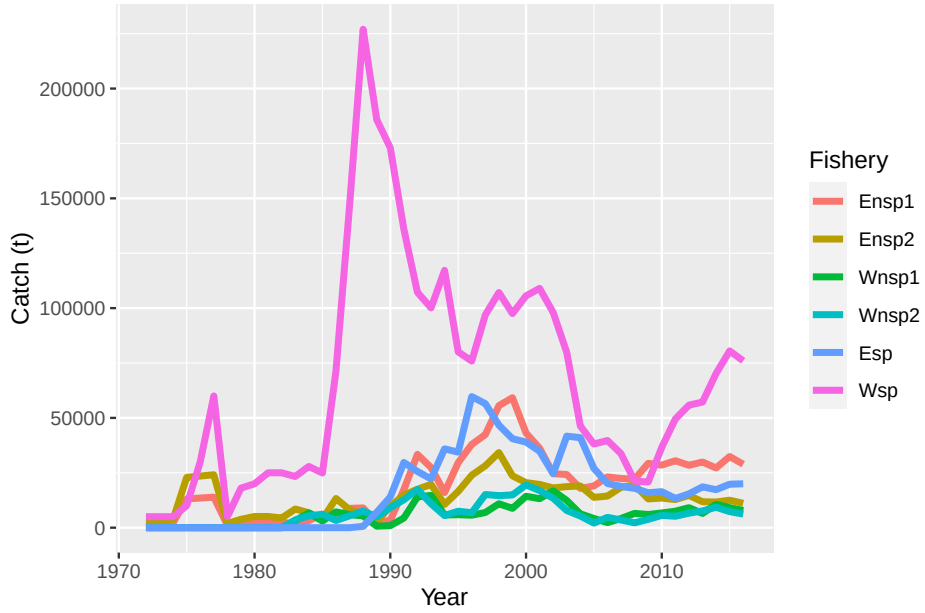
The `r4Casal2` has some functions that summarise a set of input files and returns a summary of the key model attributes. It can be difficult to know all the working parts in a Casal2 model. This is compounded when users make tweaks during an assessment and so the initial assumptions will not correspond to the final assumptions. The key function is `summarise_config`

3.1 Example files

```
config_dir = system.file("extdata", "TestModelComplex", package = "r4Casal2", mustWork = TRUE)
## This function is the key function will read a Casal config file and report useful information
## should be used when describing model structures and assumptions
## as well as validation.
summary = summarise_config(config_dir, config_file = "config.csl2", quiet = T)
names(summary)
```

```
##  [1] "category_df"           "estimate_df"           "full_category_df"      "method_df"
##  [6] "time_step_df"          "time_step_df_just_lab" "obs_year_df"           "model_years"
## [11] "model_length_bins"     "M_by_category"         "model_block"
```

```
ggplot(summary$catch_df, aes(x = year, y = catch, col = fishery)) +
  geom_line(size = 1.5) +
  labs(x = "Year", y = "Catch (t)", col = "Fishery")
```




```
ggplot(summary$obs_year_df, aes(x = year, y = observation, col = observation, size = a
  geom_point() +
  guides(colour = "none", size = "none")
```

```
## Warning: Removed 509 rows containing missing values (`geom_point()`).
```

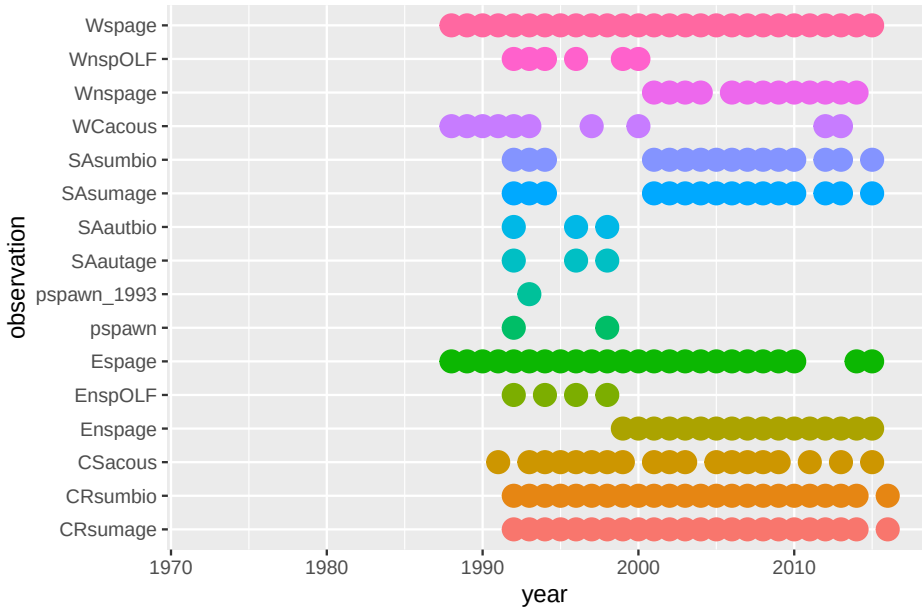


Table 3.1: (#tab:$annual_cycle$)Annual cycle

Time-step	Processes (type)
Oct_Nov	Wrtn (transition_category), Ertn (transition_category), Instant_mortality (mortality_instantaneous
Dec_Mar	recruit_W (recruitment_beverton_holt), recruit_E (recruitment_beverton_holt), Instant_mortality
Apr_Jun	Whome (transition_category), Instant_mortality (mortality_instantaneous)
End_Jun	Wspmg (transition_category), Espmg (transition_category)
Jul_Sep	Ageing (ageing), Instant_mortality (mortality_instantaneous), SSB_E (derived-quantity 0.5), SSB_

Table 3.2: Category information

Category	AgeLength	LengthWeight	Distribution
male.west.sa	age_size_W_male (von_bertalanffy)	Length_weight (basic)	normal
male.east.cr	age_size_E_male (von_bertalanffy)	Length_weight (basic)	normal
male.west.cr	age_size_W_male (von_bertalanffy)	Length_weight (basic)	normal
male.west.wc	age_size_W_male (von_bertalanffy)	Length_weight (basic)	normal
male.east.cs	age_size_E_male (von_bertalanffy)	Length_weight (basic)	normal
female.west.sa	age_size_W_female (von_bertalanffy)	Length_weight (basic)	normal
female.east.cr	age_size_E_female (von_bertalanffy)	Length_weight (basic)	normal
female.west.cr	age_size_W_female (von_bertalanffy)	Length_weight (basic)	normal
female.west.wc	age_size_W_female (von_bertalanffy)	Length_weight (basic)	normal
female.east.cs	age_size_E_female (von_bertalanffy)	Length_weight (basic)	normal

```
kable(x = summary$time_step_df, caption = "Annual cycle")
```

```
kable(x = summary$full_category_df, caption = "Category information")
```

```
kable(x = summary$estimate_df, caption = "Estimate summary")
```

Table 3.3: Estimate summary

label	same	prior	lower_bound
CSacousq	-	lognormal	0.01
WCacousq	-	lognormal	0.01
CRsumq	-	lognormal	0.016
SAsumq	-	lognormal	0.020
SAautq	-	lognormal	0.020
CR_process_error	-	uniform	0.0
SA_process_error	-	uniform	0.0
B0_E_with_total_log_b0_prior	-	uniform	12.6
B0_W_with_proportion_prior	-	beta	0.11
YCS_E	-	lognormal	0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.0
YCS_W	-	lognormal	0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.0
M_male_x0	-	uniform	5.1
M_male_y0	-	uniform	0.01
M_male_y1	-	uniform	0.5
M_male_y2	-	uniform	0.5
M_female_x0	-	uniform	5.1
M_female_y0	-	uniform	0.01
M_female_y1	-	uniform	0.5
M_female_y2	-	uniform	0.5
sel_Whome	-	uniform	0.01 0.01 0.01 0 0 0 0 1
sel_Espmg_male	-	uniform	0 0 0 0 0 0 0 0
sel_Wspmg_male	-	uniform	0 0 0 0 0 0 0 0
sel_Espmg_female	-	uniform	0 0 0 0 0 0 0 0.6
sel_Wspmg_female	-	uniform	0 0 0 0 0 0 0 0.6
Enspsl_mu	-	uniform	64
Enspsl_s_l	-	uniform	4
Enspsl_s_r	-	uniform	4
Wnspsl_mu	-	uniform	64
Wnspsl_s_l	-	uniform	4
Wnspsl_s_r	-	uniform	4
Espsl_a50	-	uniform	6
Espsl_ato95	-	uniform	4
Wspsl_shift_param	-	normal_by_stdev	-10.24
CRsl_mu	-	uniform	64
CRsl_s_l	-	uniform	4
CRsl_s_r	-	uniform	4
SAsl_mu	-	uniform	64
SAsl_s_l	-	uniform	4
SAsl_s_r	-	uniform	4

Chapter 4

MPD summaries

4.1 Single Model Output

```
file_name = system.file("extdata","SimpleTestModel",
                        "estimate.log", package = "r4Casal2", mustWork = TRUE)
mpd = extract.mpd(file = file_name)
```

4.1.1 Model Convergence

There are a range of appproaches for checking your model has converged. The approaches we will be working through include checking the hessian is positive definite, checking parameters are not running to bounds and reestimate with random starting locations.

When estimating models in Casal2, it is recommended to have the following report included

```
@report covariance_matrix
type covariance_matrix
## or the Hessian
@report hessian_matrix
type hessian_matrix
```

When estimation is complete and you have read in the Casal2 output using `Casal2::extract.mpd()`.

```
# file name
mpd_file_name = system.file("extdata", "PosteriorPredictiveChecks","estimate.log",
                            package = "r4Casal2", mustWork = TRUE)
# read in output
mpd = extract.mpd(file = mpd_file_name)
```

Table 4.1: (#tab:$correlated_params$)Correlated Parameters

correlation	row_param	col_param
-0.974	process[Recruitment].b0	catchability[chatTANq].q
0.830	selectivity[chatTANSel].mu	selectivity[chatTANSel].sigma_1
0.951	selectivity[eastFSel].mu	selectivity[eastFSel].sigma_1
0.925	selectivity[westFSel].mu	selectivity[westFSel].sigma_1
0.810	process[Recruitment].ycs_values{1978}	process[Recruitment].ycs_values{1979}
0.818	process[Recruitment].ycs_values{1979}	process[Recruitment].ycs_values{1980}
0.827	process[Recruitment].ycs_values{1980}	process[Recruitment].ycs_values{1981}
0.831	process[Recruitment].ycs_values{1981}	process[Recruitment].ycs_values{1982}
0.835	process[Recruitment].ycs_values{1982}	process[Recruitment].ycs_values{1983}

```
# is covariance symmetric
isSymmetric(mpd$covar$covariance_matrix)
# is hessian invertable
is_matrix_invertable(mpd$hess$hessian_matrix)
# check high correlations
correlation_matrix = cov2cor(mpd$covar$covariance_matrix)
corr_df = get_high_correlations(correlation_matrix = correlation_matrix, max_correlati
                      labels = names(mpd$estimate_value$values))
corr_df$correlation = round(corr_df$correlation, 3)
```

```
kable(x = corr_df[,c("correlation", "row_param", "col_param")],
      caption = "Correlated Parameters")
```

You will want to try remove high correlations from the covariance to help estimation and MCMC simulations. We recommend you explore parameter transformations to remove high correlations or alternative parameterisations.

Once these are satisfied you will have more confidence in your standard errors, in addition to being able to run MCMC run mode.

Another useful convergence diagnostic is re-estimating Casal2 with difference starting locations. The function used for this is `?generate.starting.pars`. This will read a Casal2 config file that contains all the `@estimate` definitions are generate a bunch of random starting values in the format of useable for `-i` in Casal2. Below is some example R code of running Casal2 from R with randomly generated starting values.

```
working_dir = "Directory to Casal output"
## generate starting values
start_pars = generate.starting.pars(path = working_dir,
                                    estimation_csl2_file  = "estimation.csl2",
                                    par_file_name = "starting_pars.out")
## re-run Casal2
```

```
current_dir = getwd()
setwd(working_dir)
system2(command = "casal2", args = "-e -o multi_start_pars.par -i starting_pars.out",
        stdout = "multi_start.log",
        stderr = "multi_start.err", wait=TRUE)
system2(command = "casal2", args = "-r -i starting_pars.out",
        stdout = "multi_start_init.log",
        stderr = "multi_start_init.err", wait=TRUE)
setwd(current_dir)

## read in jitter_start run
multi_est = extract.mpd("multi_start.log", path = working_dir)
multi_run = extract.mpd("multi_start_init.log", path = working_dir)
## check if any didn't converge

## plot SSBS
ssb_df = get_derived_quantities(multi_est)
ggplot(ssb_df, aes(x = years, y = values, col = par_set, linetype = par_set)) +
  geom_line(size = 1.5) +
  labs(x = "Years",y = "SSB", linetype = "Starting\nvalues", col = "Starting\nvalues")
ssb_df = get_derived_quantities(multi_run)
ggplot(ssb_df, aes(x = years, y = values, col = par_set, linetype = par_set)) +
  geom_line(size = 1.5) +
  labs(x = "Years",y = "SSB", linetype = "Starting\nvalues", col = "Starting\nvalues")

## get aggregated objective functions
obj = aggregate_objective_report(model = multi_est)
head(obj)
```

A useful diagnostic that is encouraged to explore is to run Casal2 as an age-structured population model (ASPM) [Minte-Vera et al., 2017, Carvalho et al., 2021]. This is useful for asking the following question "do we need to know the variability in recruitment to get the "correct" trends in relative abundance and the absolute scale of the model?" The diagnostic is run following these steps 1. Estimate the full integrated model 2. Fix selectivity parameters at MPD values from step 2 3. Turn off recruitment variability i.e., assume R_0 for all years 4. fit the model (ASPM) to the indices of abundance only. Just estimating R_0, q etc. 5. fit the above model again but estimate YCS parameters (ASPMdev) 6. fit the model again with the recruitment deviates set equal to the MPD values from the integrated model (ASPMfix)

This runs will help you explore the fit and production assumptions in the data. The idea is to understand what parameters are informing what population signals and how perhaps identify misspecifications. For more information on this diagnostic we recommend users read Minte-Vera et al. [2017].

4.1.2 Data Weighting

Some pseudo r code to help with data weighting according to Francis [2011] with multinomial data.

```
working_dir = "Directory to Casal output"
reweight_folder = "Reweight"
## Don't always want to re-run this code
if(FALSE) {
  weights = run_automatic_reweighting(config_dir = working_dir,
                                      config_filename = "config.csl2",
                                      weighting_folder_name = reweight_folder,
                                      mpd_file_name = "estimate.log",
                                      n_loops = 3,
                                      approximate_single_year_obs = T)
  saveRDS(weights, file = file.path(working_dir, reweight_folder, "Weights.RDS"))
}
## get reweighted MPDs to observe the effect
MPD_list = extract_reweighted_mpds(file.path(working_dir,reweight_folder))

## plot SSBs
plot_derived_quantities(MPD_list)
plot_fishery(MPD_list, quantity = "exploitation")
plot_recruitment(MPD_list, quantity = "standardised_recruitment_multipliers")
```

4.1.3 Model quantities

Fishing Pressures

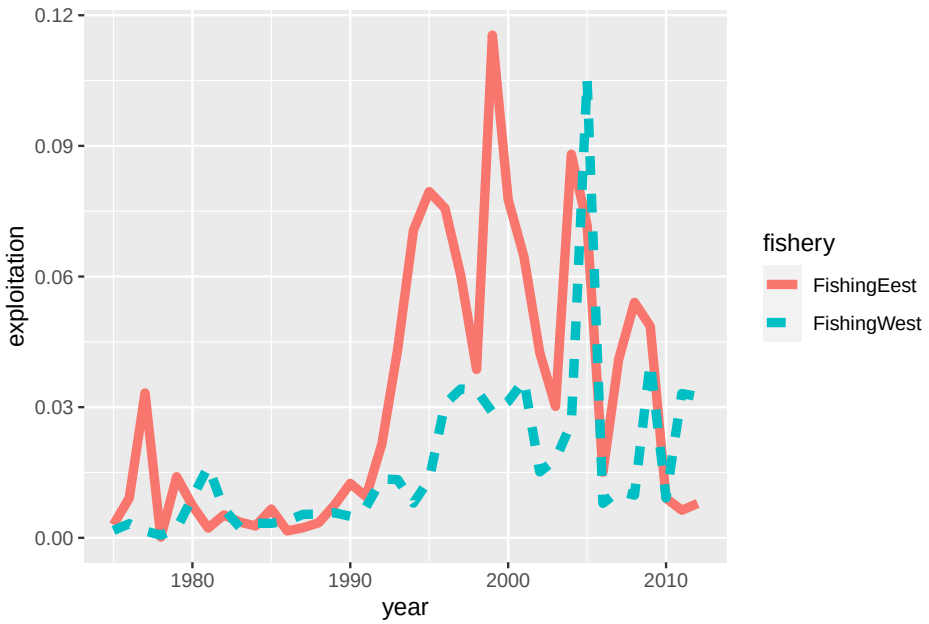
Below illustrates code to plot fishing pressure, but you can also easily adapt the code to plot catches. One thing to note, is Casal2 will report both `exploitation_rate` (actual a proportion also called harvest rate) and `fishing_pressures`. For models that only have a single fishery per time-step and area these quantities will be the same. If there are multiple fisheries interacting with the partition in the same time-step then these quantities will differ. Fishing pressure is the maximum exploitation applied to the partition for that time-step. See the Casal2 user manual for more detail on the difference. `exploitation_rate` reported is just

$$\frac{catch}{vulnerable}$$

Some R-code used to summarise fishing pressures.

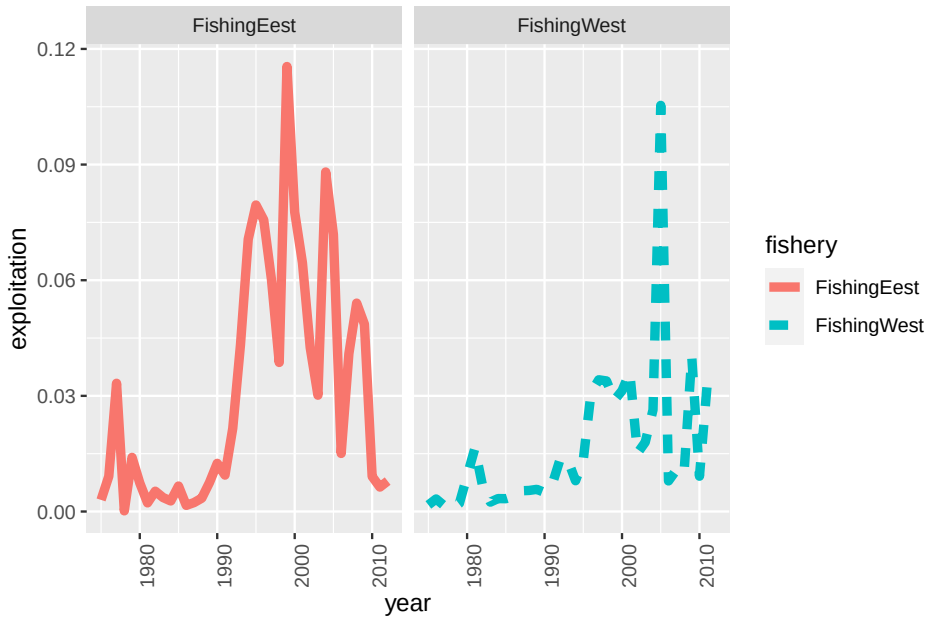
```
file_name = system.file("extdata", "SimpleTestModel", "estimate.log",
                        package = "r4Casal2", mustWork = TRUE)
mpd = extract.mpd(file = file_name)
# Report labels
```

```
# names(mpd)
# plot fishing pressures
fishery_info = get_fisheries(mpd)
head(fishery_info)
# Note this will print both fishing pressure and exploitation
my_plot = ggplot(fishery_info, aes(x = year, y = exploitation, col = fishery, linetype = fishery)
                  geom_line(size =2)
my_plot
```

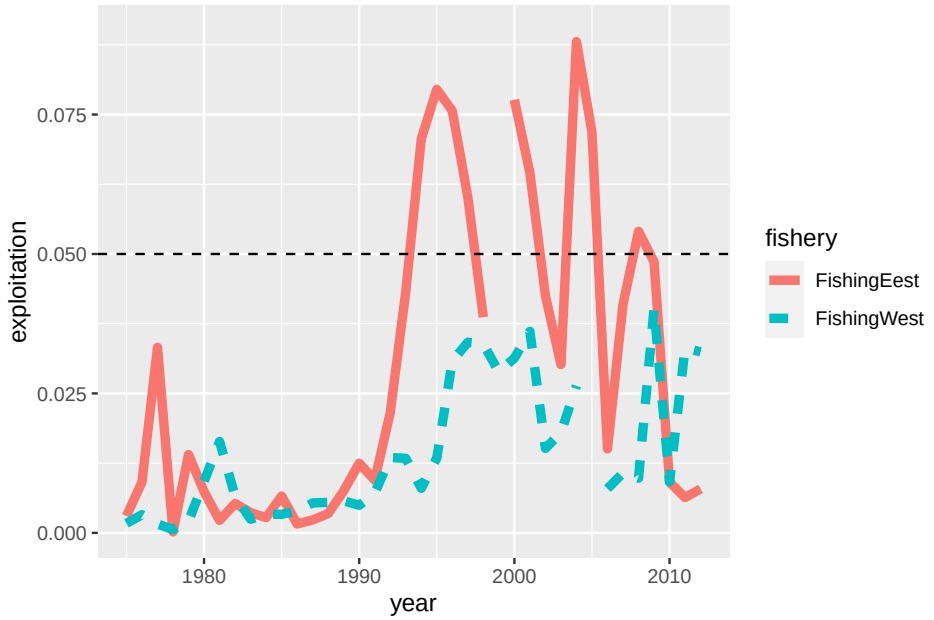


Flexibility using standard ggplot functions

```
# you can add adjust it as you please, for example if you want 3 panels for each fishery
my_plot +
  facet_wrap(~fishery) +
  theme(axis.text.x = element_text(angle = 90))
```



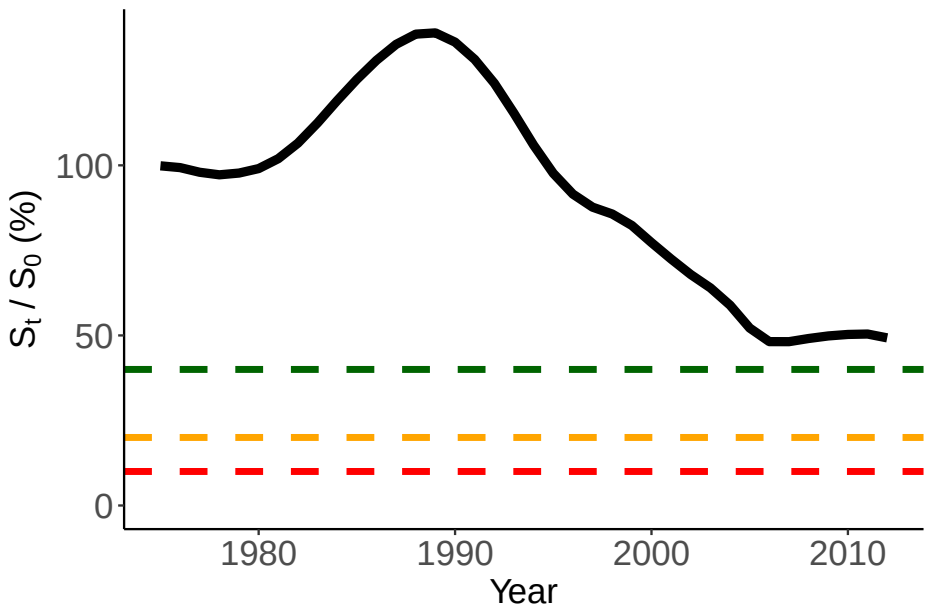

```
# Adjust ylim and add a reference limit
my_plot + ylim(0,0.09) + geom_hline(yintercept = 0.05, linetype = "dashed")
```



SSBs {-}

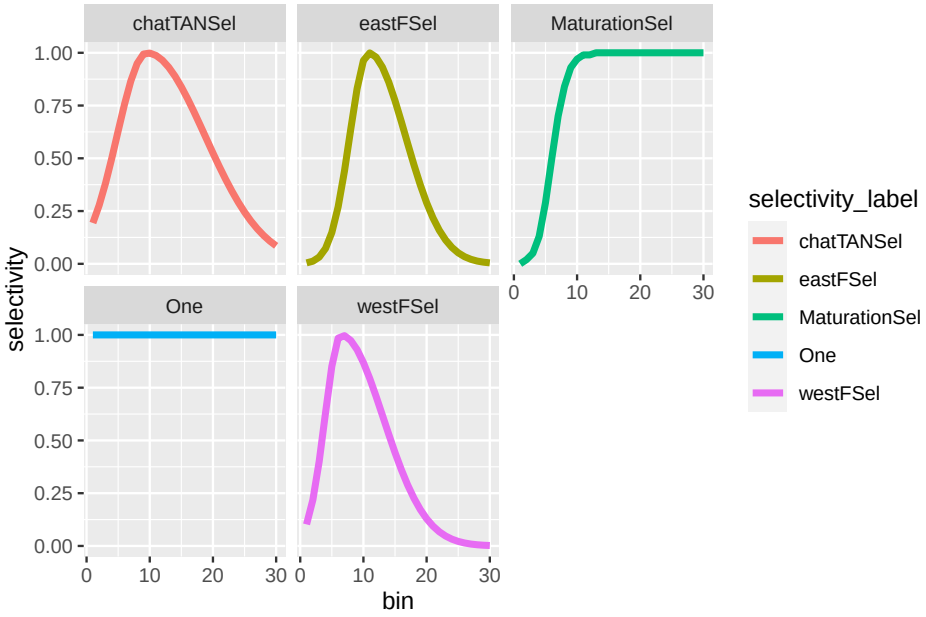
```
# get SSB and recruit
ssb_df = get_dqs(mpd)
```

```
ssb_df$model_year = ssb_df$years
recruit_df = get_BH_recruitment(mpd)
# merge these two data frames
joint_df = right_join(x = ssb_df, y = recruit_df, by = c("model_year", "par_set"))
## for multi-stock models, you will need to also join by stock id
joint_df$percent_b0 = joint_df$values / joint_df$b0 * 100
## plot percent B0
ggplot(joint_df, aes(x = years, y = percent_b0)) +
  geom_line(size = 2) +
  ylim(0,NA) +
  labs(x = "Year", y = expression(paste(S[t], " / ", S[0], " (%)"))) +
  geom_hline(yintercept = 40, col = "darkgreen", linetype = "dashed", size = 1.5) +
  geom_label(x = 1900, y = 40, label = "Target") +
  geom_hline(yintercept = 20, col = "orange", linetype = "dashed", size = 1.5) +
  geom_label(x = 1900, y = 20, label = "Soft") +
  geom_hline(yintercept = 10, col = "red", linetype = "dashed", size = 1.5) +
  geom_label(x = 1900, y = 10, label = "Hard")+
  ggtitle("Percent B0") +
  theme_classic() +
  theme(legend.position = "bottom",
        axis.text = element_text(size = 16),
        axis.title = element_text(size = 16),
        strip.text = element_text(size=16),
        title=element_blank(),
        legend.text = element_text(size=16))
```



4.1.3.1 Plotting selectivities

```
selectivity_df = get_selectivities(model = mpd)
ggplot(selectivity_df, aes(x = bin, y = selectivity, col = selectivity_label)) +
  geom_line(size = 1.5) +
  facet_wrap(~selectivity_label)
```



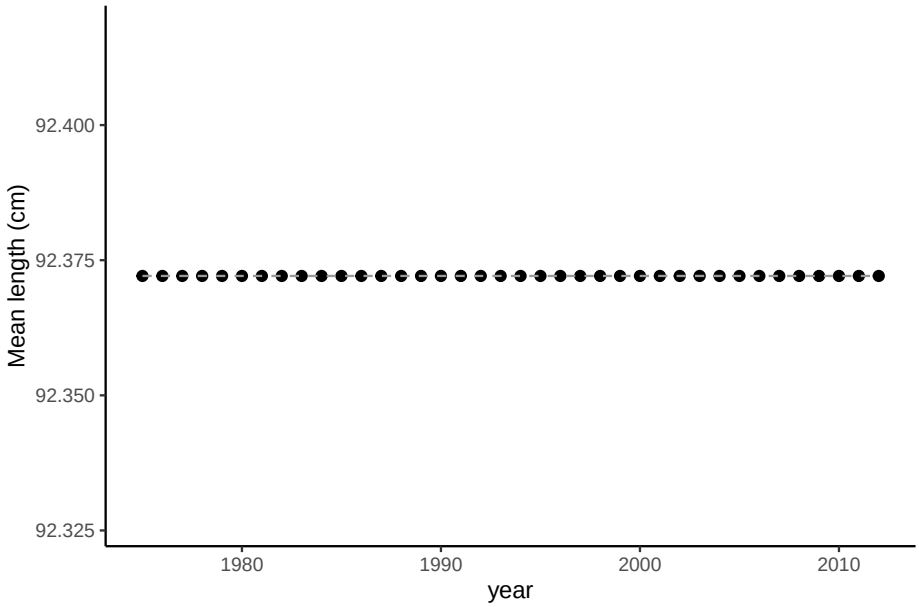
Growth {-}

```
growth_df = get_growth(mpd)
head(growth_df)
```

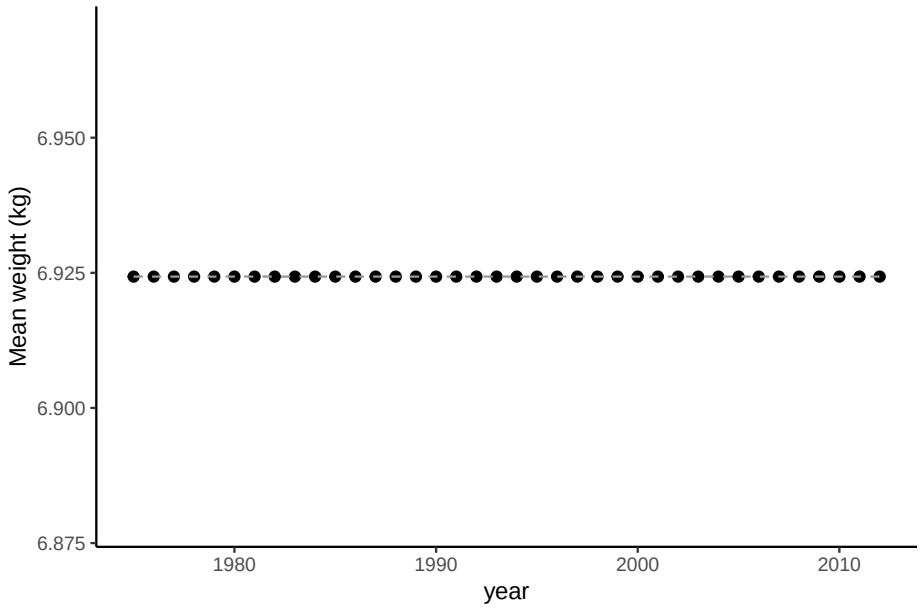
```
##   age year time_step cvs_by_age mean_length_at_age mean_weight_at_age            lab
## 1   1 1975     step1        0.1              29.6178        0.000143136 age_length_ste
## 2   2 1975     step1        0.1              44.7450        0.000555816 age_length_ste
## 3   3 1975     step1        0.1              55.5000        0.001128550 age_length_ste
## 4   4 1975     step1        0.1              63.8482        0.001789010 age_length_ste
## 5   5 1975     step1        0.1              70.5854        0.002488050 age_length_ste
## 6   6 1975     step1        0.1              76.1440        0.003192290 age_length_ste
```

```
## if time-varying or type data summarise mean length and weight by year
avg_summarises = growth_df %>% group_by(year) %>% summarise(
  mean_length = mean(mean_length_at_age),
  mean_weight = mean(mean_weight_at_age))

ggplot(avg_summarises, aes(x = year, y = mean_length)) +
  geom_point(size =2) +
  geom_line(col = "gray60", linetype = "dashed") +
```

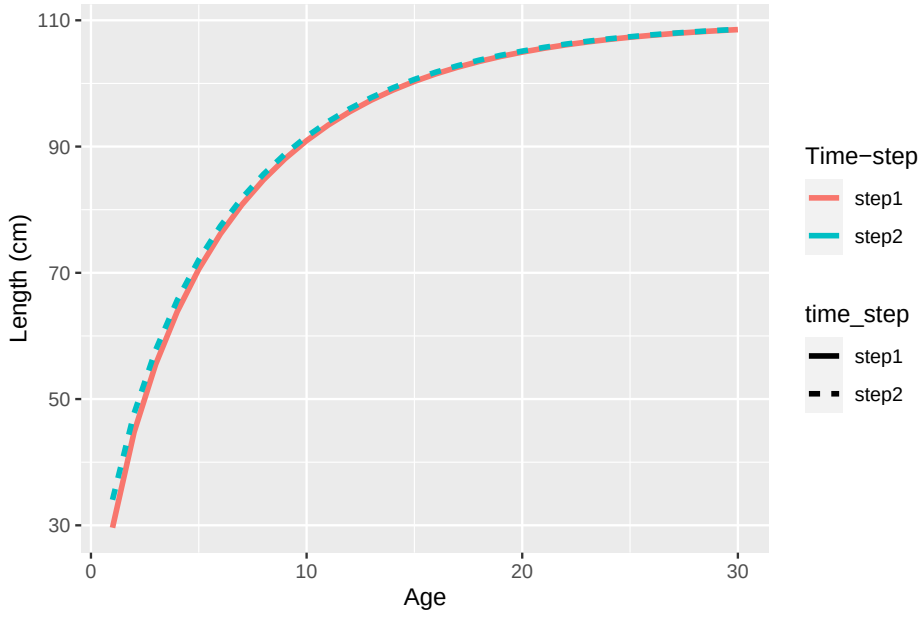
```
  ylab("Mean length (cm)") +
  theme_classic()
```




```
ggplot(avg_summarises, aes(x = year, y = mean_weight * 1000)) +
  geom_point(size =2) +
  geom_line(col = "gray60", linetype = "dashed") +
  ylab("Mean weight (kg)") +
  theme_classic()
```

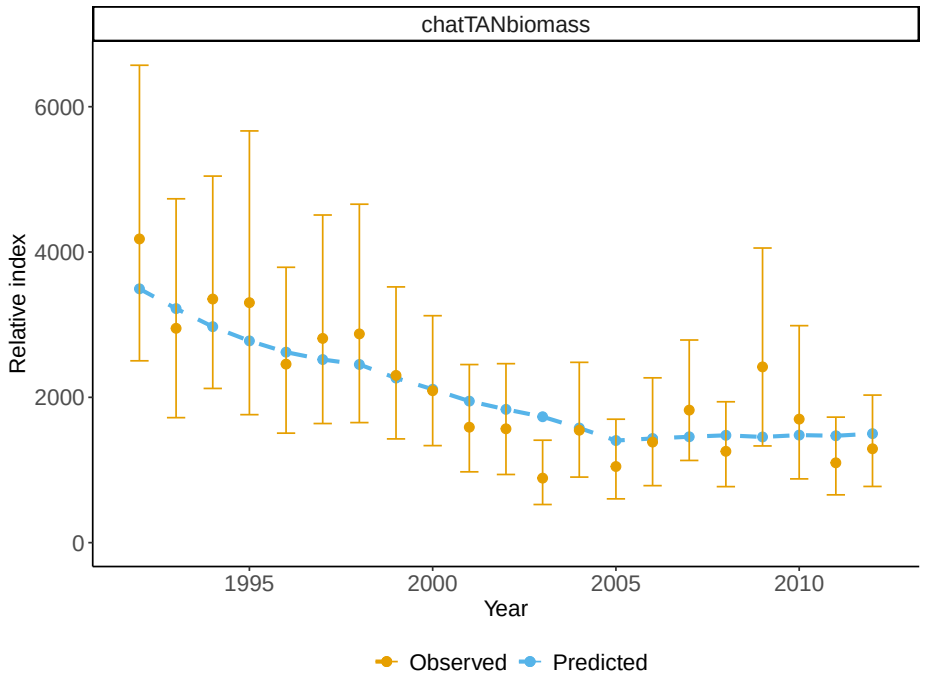



```
## if not time-varying just pick one year to plot
ggplot(growth_df %>% filter(year == 1990)) +
  geom_line(aes(x = age, y = mean_length_at_age, col = time_step, linetype = time_step)
  labs(x = "Age", y = "Length (cm)", col = "Time-step")
```



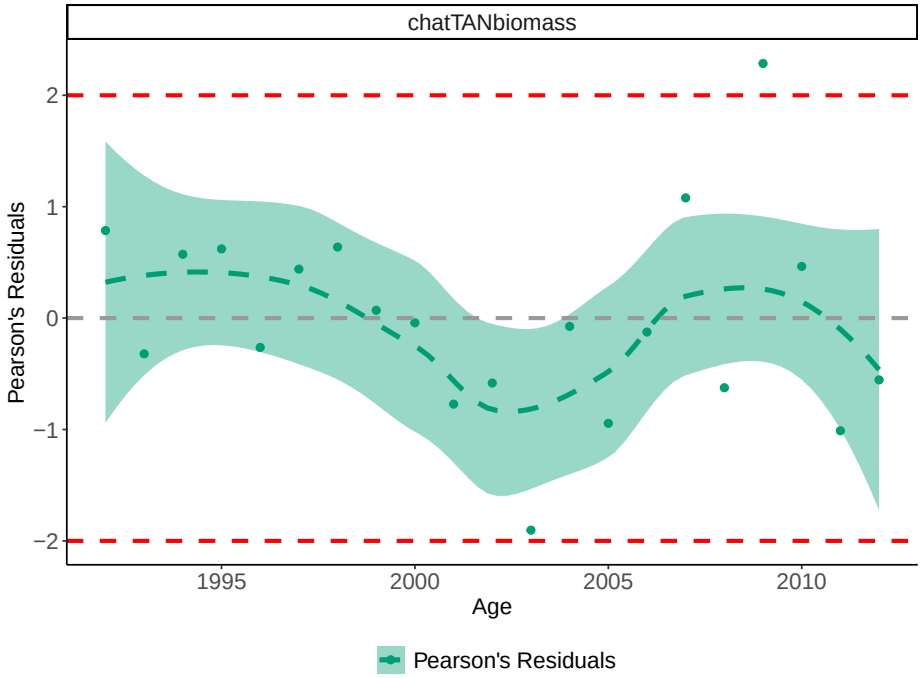
4.1.4 Plotting Fits

```
## define a palette
obs_palette <- c("#E69F00", "#56B4E9","#009E73")
names(obs_palette) = c("Observed", "Predicted","Pearson's Residuals")
## get abundance data frames
abundance_obs = get_abundance_observations(mpd)
abundance_obs_label = unique((abundance_obs$observation_label))

## plot observed vs fitted
ggplot(abundance_obs, aes(x = year)) +
    geom_line(aes(y = expected, col = "Predicted"), size = 1.2,
              linetype = "dashed") +
    geom_point(aes(y = expected, col = "Predicted"), size = 2.5) +
    geom_errorbar(aes(ymin = L_CI, ymax = U_CI, col = "Observed"),
                  width=.5, position = position_dodge(width=0.9)) +
    geom_point(aes(y = observed, col = "Observed"), size = 2.5) +
    labs(x = "Year", y = "Relative index", color = "") +
    ylim(0,NA) +
    theme_classic() +
    facet_wrap(~observation_label, ncol = 1, scales = "free_y") +
    theme(legend.position = "bottom",
          axis.text = element_text(size = 14),
          axis.title = element_text(size = 14),
          strip.text = element_text(size=14),
          legend.text = element_text(size=14)) +
    scale_color_manual(values = obs_palette[1:2])
```




```
## Plot pearsons residuals
ggplot(abundance_obs, aes(x = year)) +
  geom_point(aes(y = pearsons_residuals, col = "Pearson's Residuals"),
             size = 2) +
  geom_smooth(aes(y = pearsons_residuals, col = "Pearson's Residuals",
                  fill = "Pearson's Residuals"), size = 1.5, alpha = 0.4,
              linetype = "dashed") +
  labs(x = "Age", y = "Pearson's Residuals", fill = "", col = "") +
  geom_hline(yintercept = 0, col = "#999999", linetype = "dashed",
             size = 1.2) +
  geom_hline(yintercept = c(-2,2), col = "red", linetype = "dashed",
             size = 1.2) +
  facet_wrap(~observation_label, ncol = 4) +
  #ylim(3,-3) +
  theme_classic() +
  facet_wrap(~observation_label, ncol = 1, scales = "free_y") +
  list(theme(legend.position = "bottom",
        axis.text = element_text(size = 14),
        axis.title = element_text(size = 14),
        strip.text = element_text(size=14),
        legend.text = element_text(size=14)),
  scale_color_manual(values = obs_palette[3]),
  scale_fill_manual(values = obs_palette[3]))
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
## complete a Wald-Wolfowitz Runs Test
## are residuals a random sequence
runs_test_residuals(abundance_obs$pearsons_residuals)
```

```
## $siglim
## [1] -2.340038  2.340038
##
## $p.runs
## [1] 0.556
```

```
## seem fine i.e. not rejecting Null hypothesis
## H0: "The order of the data is random "
```

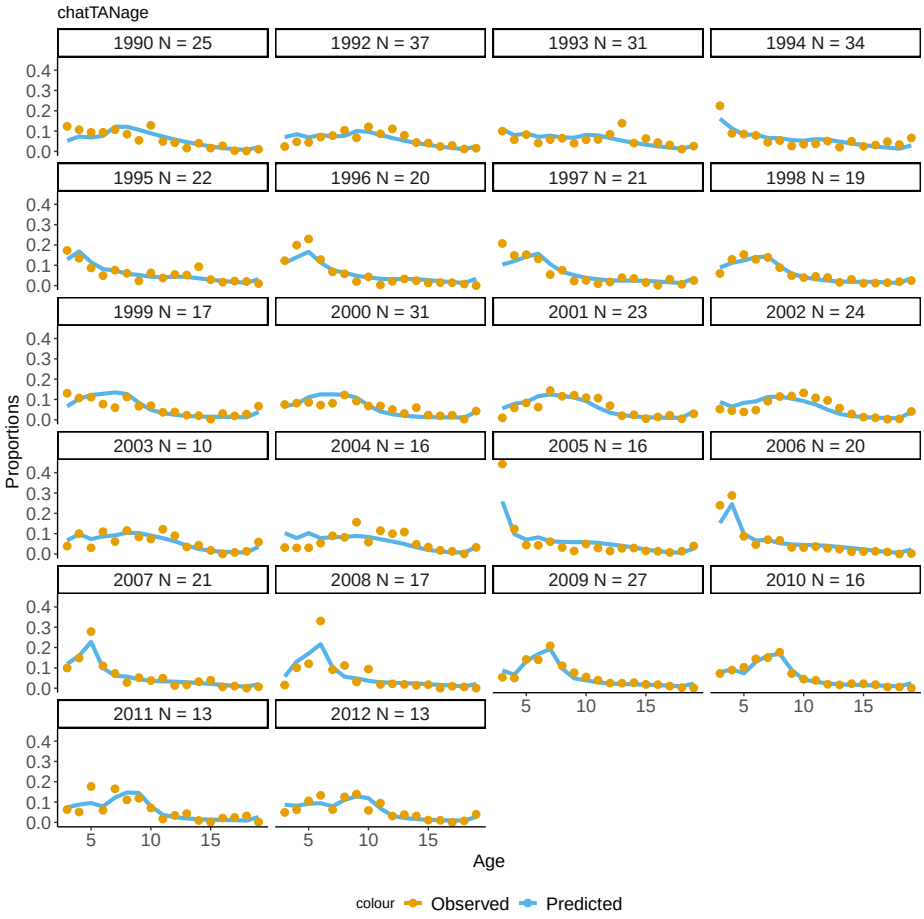
Age composition

```
## define a palette
comp_obs = get_composition_observations(mpd)
unique(comp_obs$observation_label)
```

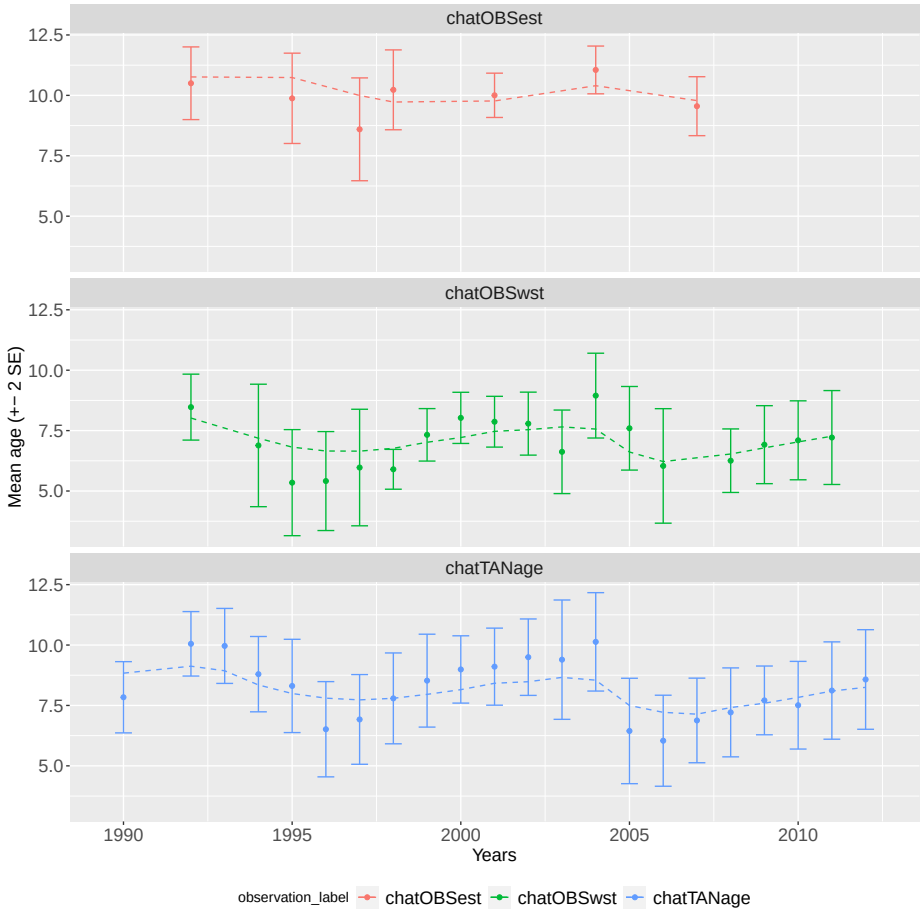
```
## [1] "chatTANage" "chatOBSwst" "chatOBSest"
```

```
## just plot one of them
## and the first 12 years
this_obs = comp_obs %>% filter(observation_label == "chatTANage")
```

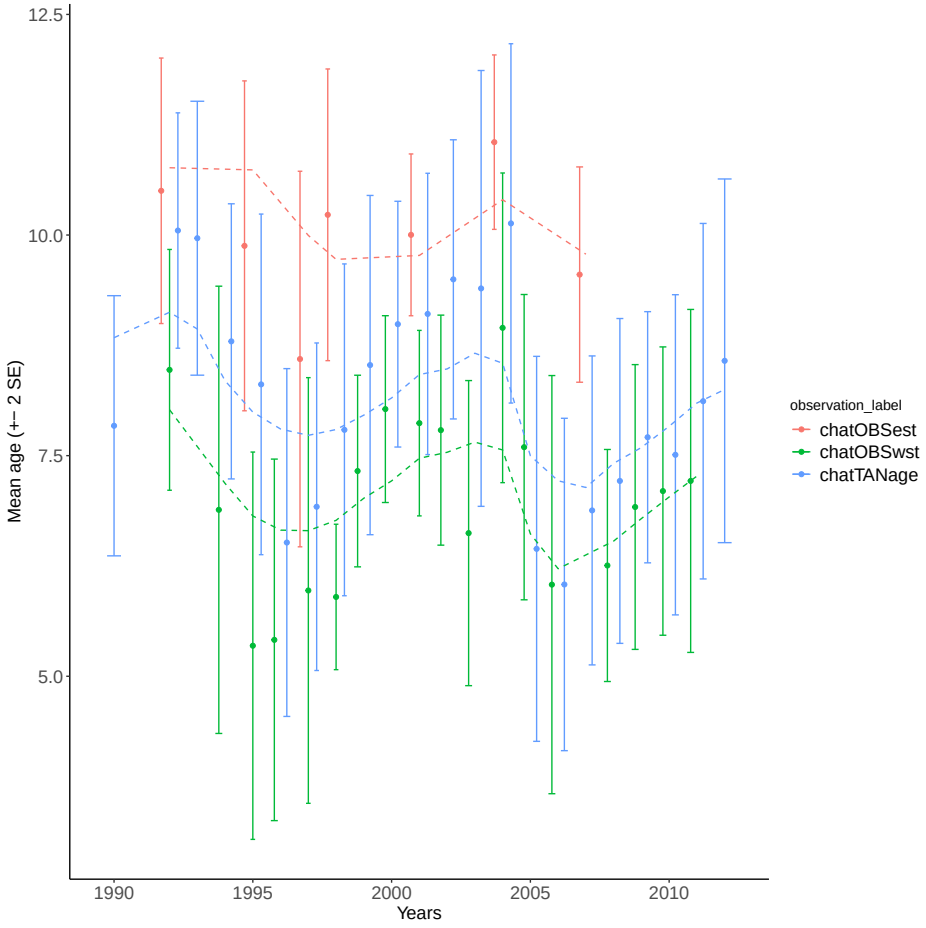
```
years_to_plot = unique(this_obs$year)[1:12]
## calculate effective N for plot
this_obs = this_obs %>% group_by(year, observation_label) %>%
  mutate(Nassumed = mean(adjusted_error))
  this_obs$label = paste0(this_obs$year, " N = ", round(this_obs$Nassumed,1))

ggplot(this_obs, aes(x = age)) +
  geom_line(aes(y = expected, col = "Predicted"), size = 1.5) +
  geom_point(aes(y = observed, col = "Observed"), size = 2.5) +
  labs(x = "Age", y = "Proportions") +
  facet_wrap(~label, ncol = 4) +
  ggtitle("chatTANage") +
  theme_classic() +
  theme(legend.position = "bottom",
        axis.text = element_text(size = 14),
        axis.title = element_text(size = 14),
        strip.text = element_text(size=14),
        legend.text = element_text(size=14)) +
  scale_color_manual(values = obs_palette[1:2])
```




```
mean_age_df = get_composition_mean_bin(mpd)
n_obs = unique(mean_age_df$observation_label)
## create a nice plot with separate axis
ggplot(mean_age_df, aes(x = year, col = observation_label)) +
  geom_errorbar(aes(ymin = Oy-2*SEy, ymax = Oy + 2*SEy), width=.5,
                position = position_dodge(width=0)) +
  geom_point(aes(y = Oy), position = position_dodge(width=0.9)) +
  geom_line(aes(y = Ey), linetype = "dashed") +
  labs(x = "Years",y = "Mean age (+- 2 SE)") +
  facet_wrap(~observation_label, nrow = length(n_obs))  +
  theme(legend.position = "bottom",
        axis.text = element_text(size = 14),
        axis.title = element_text(size = 14),
        strip.text = element_text(size=14),
        legend.text = element_text(size=14))
```



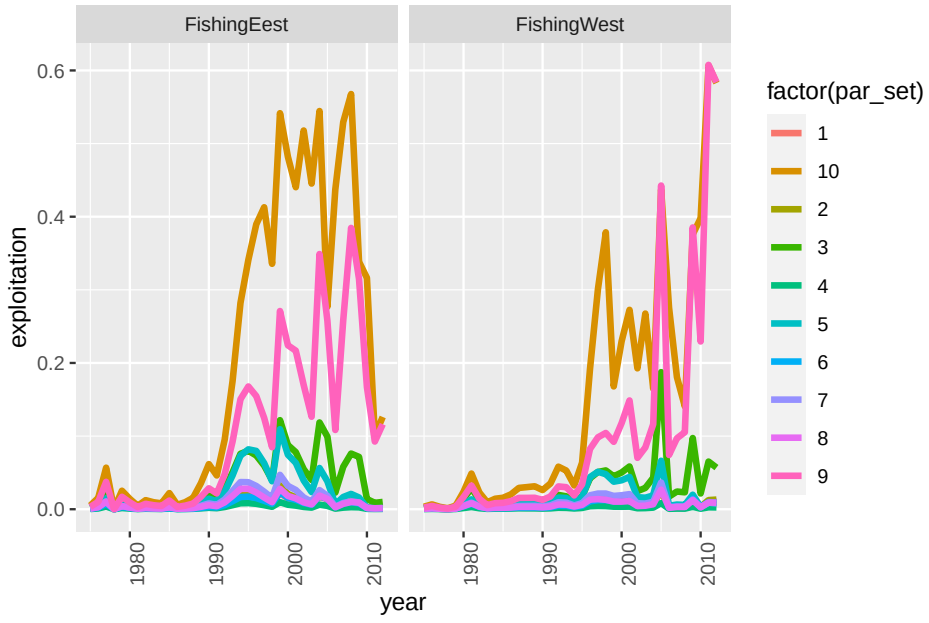

```
## a nice plot with overlapping mean age
ggplot(mean_age_df, aes(x = year, col = observation_label)) +
  geom_errorbar(aes(ymin = Oy-2*SEy, ymax = Oy + 2*SEy), width=.5,
                position = position_dodge(width=0.9)) +
  geom_point(aes(y = Oy), position = position_dodge(width=0.9)) +
  geom_line(aes(y = Ey), linetype = "dashed") +
  labs(x = "Years",y = "Mean age (+- 2 SE)") +
  theme_classic() +
  theme(legend.position = "right",
        axis.text = element_text(size = 14),
        axis.title = element_text(size = 14),
        strip.text = element_text(size=14),
        legend.text = element_text(size=14))
```



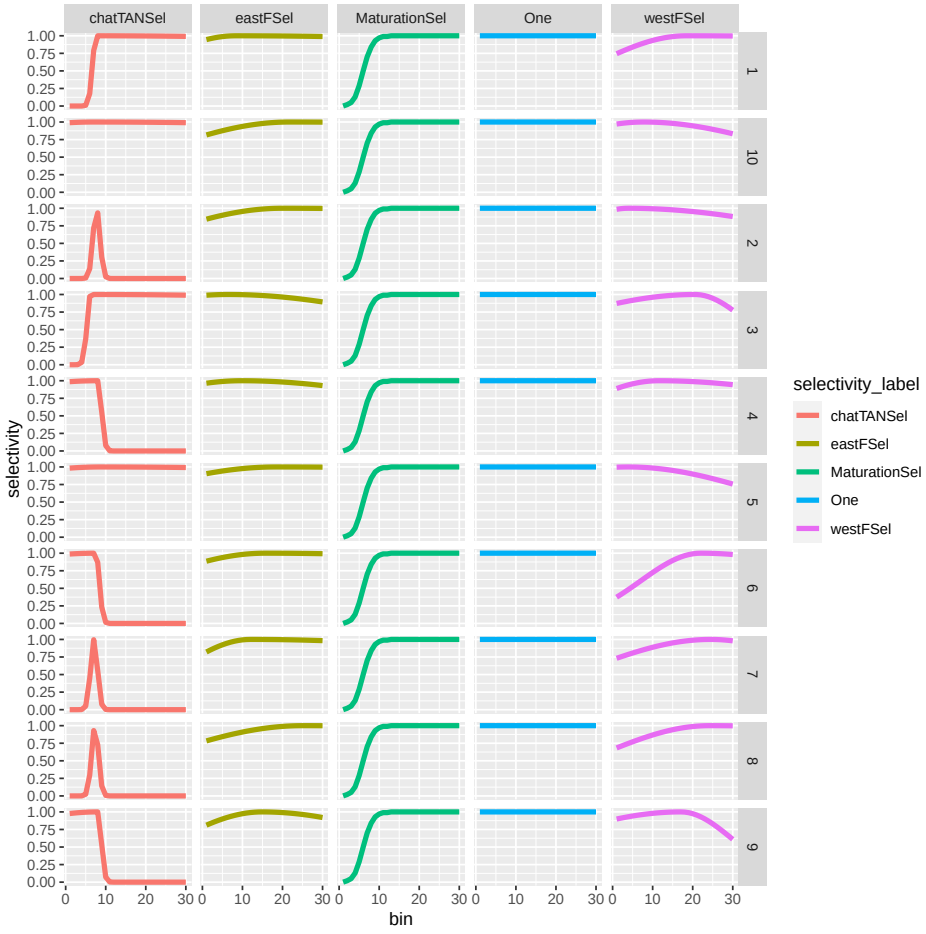
4.2 Multiple Casal2 runs with -i or -s

```
file_name = system.file("extdata", "SimpleTestModel","multi_run.log",
                        package = "r4Casal2", mustWork = TRUE)
mpd = extract.mpd(file = file_name)
# Report labels
# names(mpd)
# plot fishing pressures
fishery_df = get_fisheries(model = mpd)
my_plot = ggplot(fishery_df, aes(x = year, y = exploitation, col = factor(par_set))) +
  geom_line(size = 1.4) +
  facet_wrap(~fishery)
my_plot = my_plot + theme(axis.text.x = element_text(angle = 90))
# this will generate a generic ggplot
```

```
print(my_plot)
```

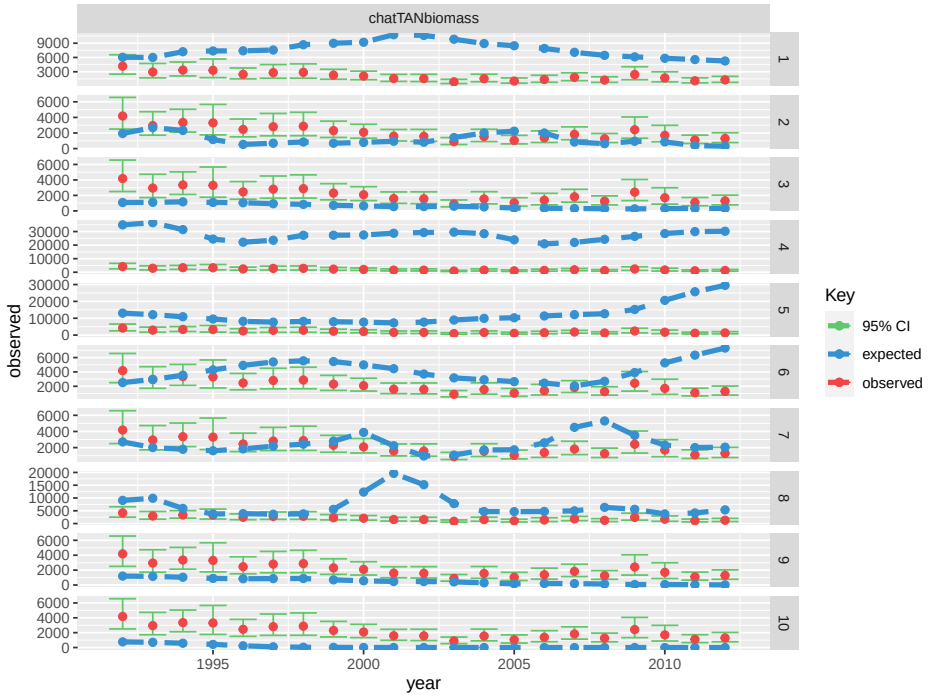



```
selectivity_df = get_selectivities(mpd)
selectivity_df$par_set = factor(selectivity_df$par_set, ordered = T)
ggplot(selectivity_df, aes(x = bin, y = selectivity, col = selectivity_label, line_typ
  geom_line(size = 1.5) +
  facet_grid(par_set ~ selectivity_label)
```




```
my_plot = plot_relative_index(model = mpd, report_labels = c("chatTANbiomass"), plot.it = T, plot
my_plot
```

Plotting Fits


chatTANbiomass
observed
year
1995
2000
2005
2010
Key
95% CI
expected
observed

Chapter 5

Comparing multiple MPD runs

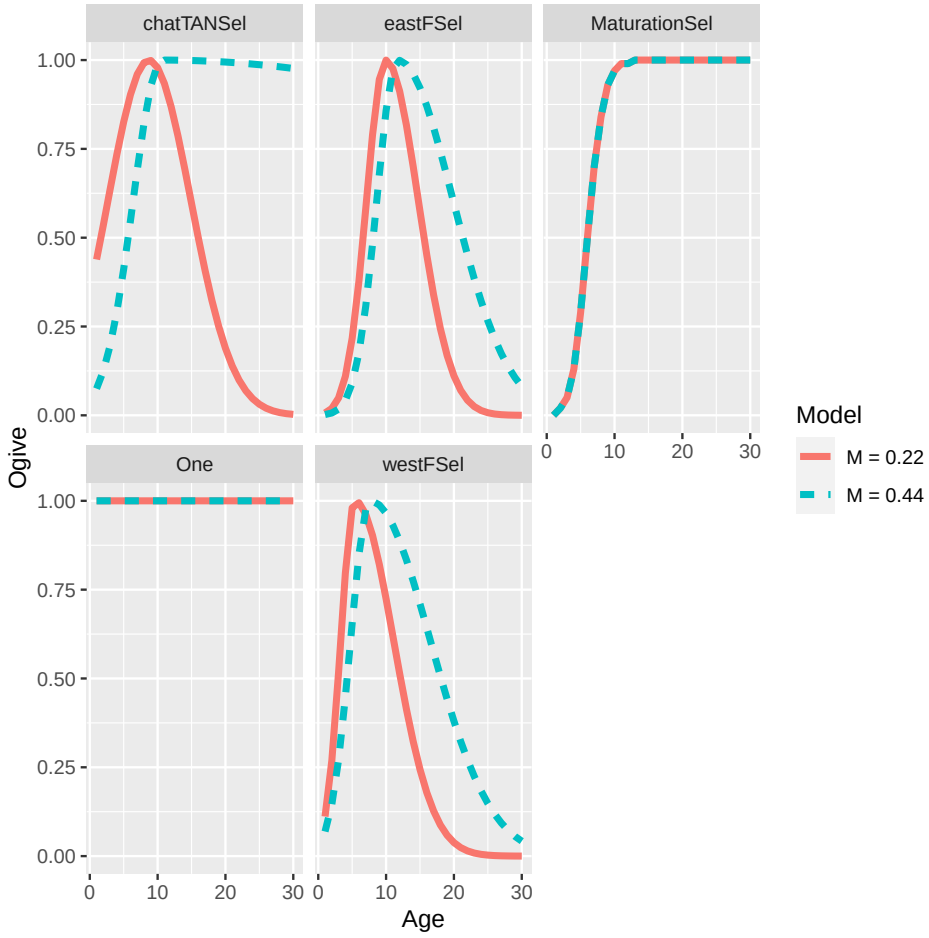
5.1 Read in models

```
file_name_low_m = system.file("extdata", "SimpleTestModel" ,"LowM.log",
                              package = "r4Casal2", mustWork = TRUE)
low_m_mpd = extract.mpd(file = file_name_low_m)
file_name_high_m = system.file("extdata", "SimpleTestModel", "highM.log",
                               package = "r4Casal2", mustWork = TRUE)
high_m_mpd = extract.mpd(file = file_name_high_m)
## create a named list
models = list("M = 0.22" = low_m_mpd, "M = 0.44" = high_m_mpd)
```

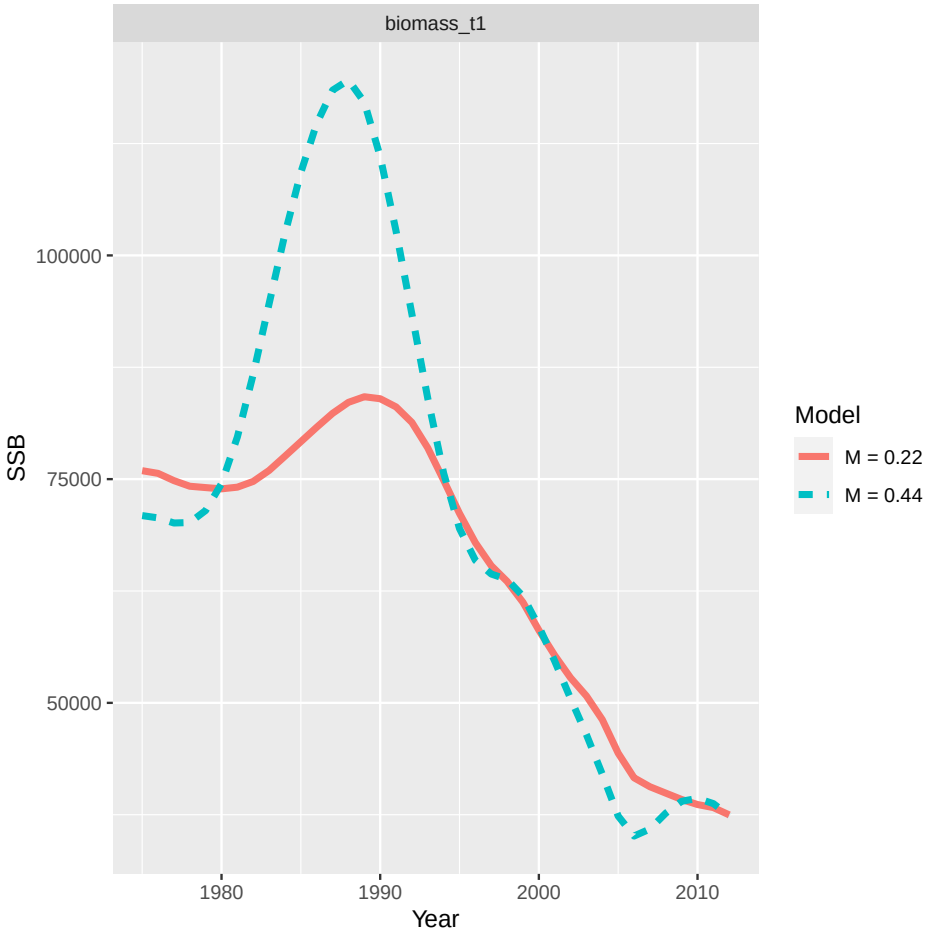
5.2 Compare model outputs

5.2.1 selectivities

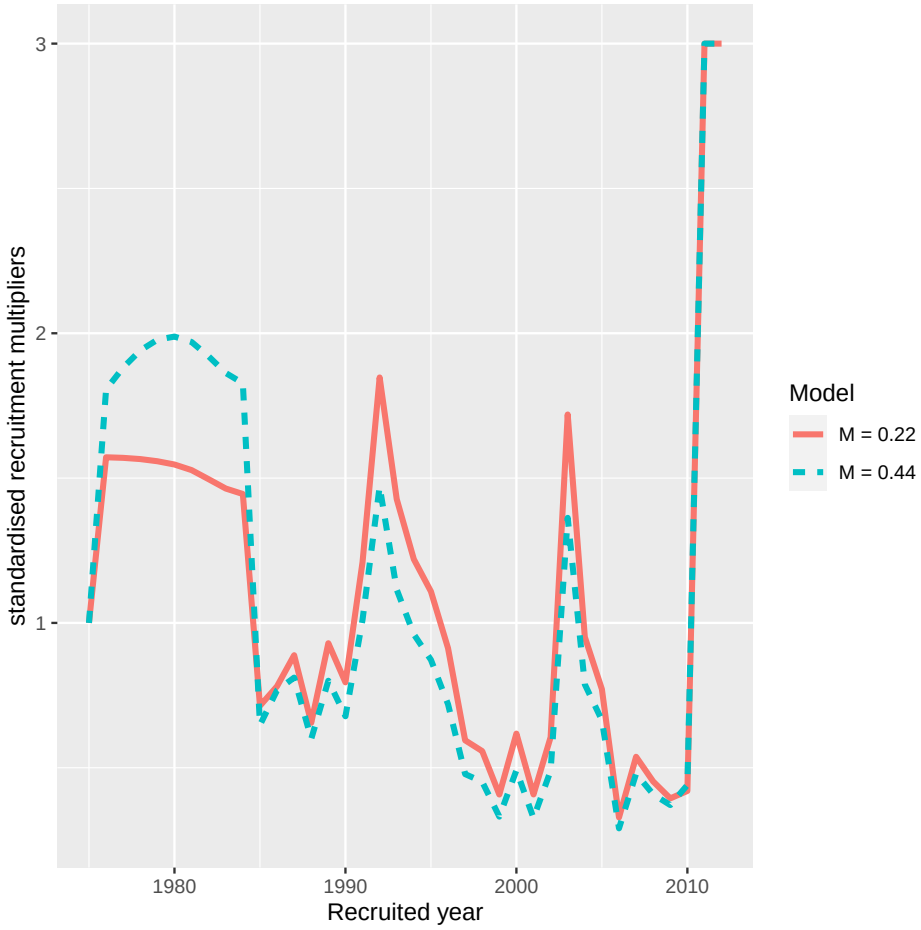
```
selectivity_df = get_selectivities(models)
ggplot(selectivity_df, aes(x = bin, y = selectivity, col = model_label, linetype = model_label))
  geom_line(size = 1.5) +
  facet_wrap(~selectivity_label) +
  labs(x = "Age", y = "Ogive", col = "Model", linetype = "Model")
```



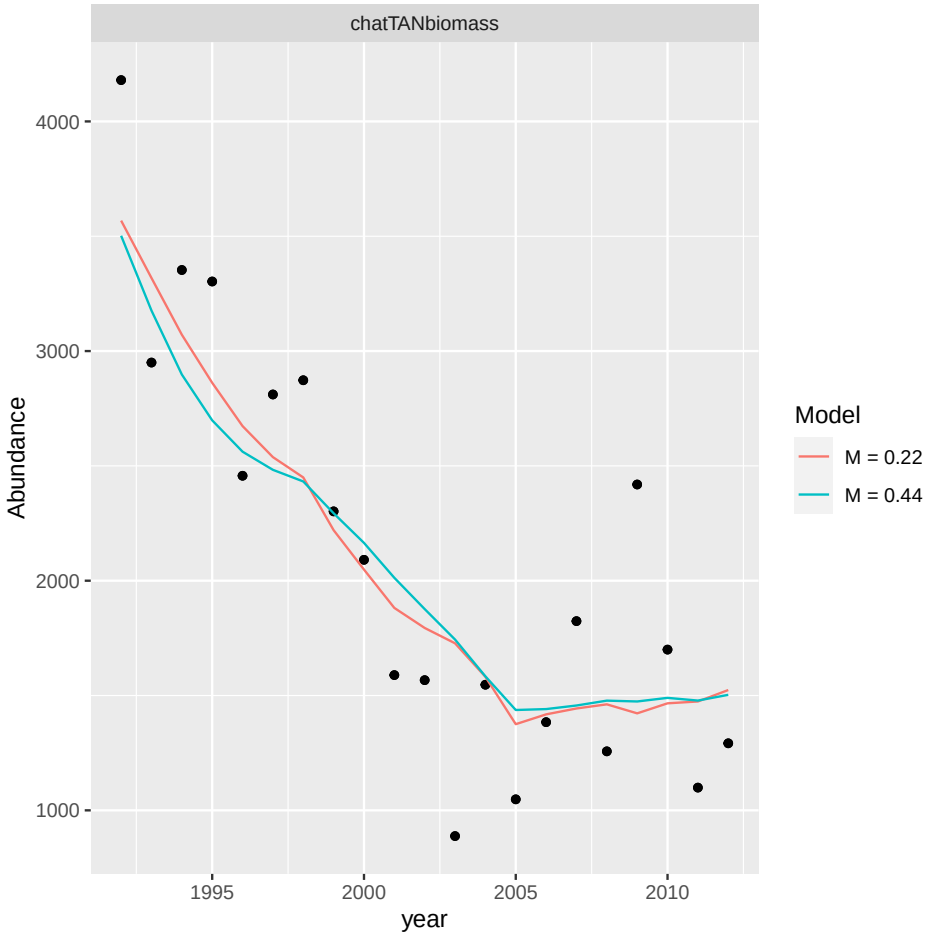

```
dq_df = get_dqs(models)
dq_df$years = as.numeric(dq_df$years)
ggplot(dq_df, aes(x = years, y = values, col = model_label, linetype = model_label)) +
  geom_line(size = 1.5) +
  facet_wrap(~dq_label) +
  labs(x = "Year", y = "SSB", col = "Model", linetype = "Model")
```




```
recruit_df = get_BH_recruitment(models)
ggplot(recruit_df, aes(x = model_year, y = standardised_recruitment_multipliers, col = model_labe
  geom_line(size = 1.3) +
  labs(x = "Recruited year", y = "standardised recruitment multipliers", col = "Model", linetype
```




```
abundance_obs_df = get_abundance_observations(models)
ggplot(abundance_obs_df, aes(x = year)) +
  geom_point(aes(y = observed), size = 1.4) +
  geom_line(aes(y = expected, col = model_label)) +
  labs(x = "year", y = "Abundance", col = "Model", linetype = "Model") +
  facet_wrap(~observation_label, scales = "free")
```



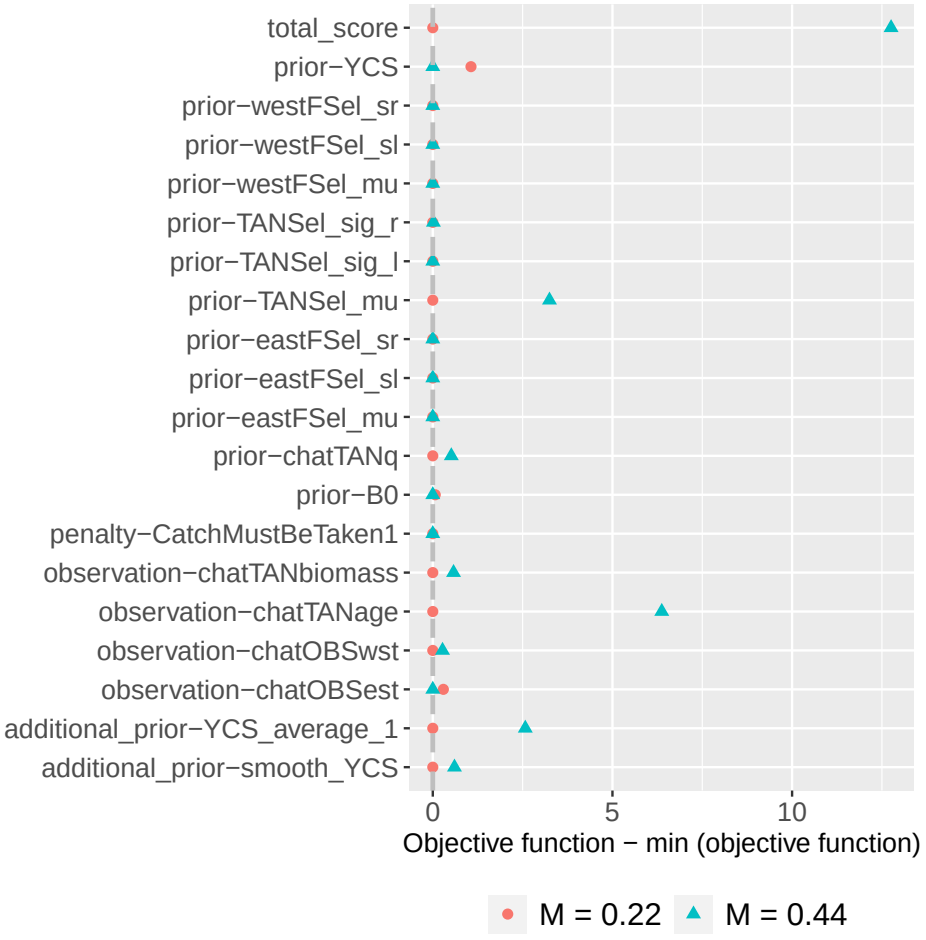
5.2.5 Compare objective function

```
cas2_obj = get_objective_function(models)
compar_obj = cas2_obj %>% pivot_wider(values_from = negative_loglik, names_from = model_label, id
head(compar_obj, n = 10)
```

```
## # A tibble: 10 x 3
##    component                   `M = 0.22`  `M = 0.44`
##    <chr>                            <dbl>       <dbl>
##  1 observation-chatTANbiomass -20.2       -19.7
##  2 observation-chatTANage     339.        346.
##  3 observation-chatOBSwst     227.        227.
##  4 observation-chatOBSest     127.        127.
##  5 penalty-CatchMustBeTaken1    0           0
##  6 prior-B0                    11.2        11.2
```

```
##  7 prior-chatTANq              -2.32       -1.80
##  8 prior-TANSel_mu              0.256       3.50
##  9 prior-TANSel_sig_l           0.0000171   0.0000502
## 10 prior-TANSel_sig_r           0.0000148   0.0177
```

```
## rescale objective score so the model with the best fit (lowest score)
## will have zero for a given component and others will have be reference from that
obj_df = cas2_obj %>% group_by(component) %>%
  mutate(rescaled_obj = negative_loglik - min(negative_loglik, na.rm = T))
## plot it for each component
ggplot(obj_df, aes(x = rescaled_obj, y = component, col = model_label, shape = model_la
  geom_point(size = 2) +
  labs(x = "Objective function - min (objective function)", y = "") +
  geom_vline(xintercept = 0, col = "gray", linetype = "dashed", size = 1) +
  theme(legend.position = "bottom",
        axis.text = element_text(size = 12),
        axis.title = element_text(size = 12),
        strip.text = element_text(size=16),
        title=element_blank(),
        legend.text = element_text(size=14))
```


total_score
prior−YCS
prior−westFSel_sr
prior−westFSel_sl
prior−westFSel_mu
prior−TANSel_sig_r
prior−TANSel_sig_l
prior−TANSel_mu
prior−eastFSel_sr
prior−eastFSel_sl
prior−eastFSel_mu
prior−chatTANq
prior−B0
penalty−CatchMustBeTaken1
observation−chatTANbiomass
observation−chatTANage
observation−chatOBSwst
observation−chatOBSest
additional_prior−YCS_average_1
additional_prior−smooth_YCS
0
5
10
Objective function − min (objective function)
M = 0.22
M = 0.44

Chapter 6

MCMC

Casal2 MCMC estimation for a single model should be done using "multiple chains". A chain in this case is a separate MCMC run, ideally starting from a different set of starting locations and with a different seed number. It is advised to create at least three chains per model. Most of the MCMC diagnostics in this package are designed for multiple chains. These include calculating Rhats (within and between chain variation in parameters) [Vehtari et al., 2021] and effective sample sizes which is a measure of efficiency in your mcmc sampler.

6.1 Read in models

```
## extra packages
library(purrr)
library(bayesplot)

cas2_file_dir = system.file("extdata", "multi_chain_mcmc", package = "r4Casal2", mustWork = TRUE)
mcmc_1 = extract.mcmc(path = cas2_file_dir, samples.file = "samples.4", objectives.file = "object
mcmc_2 = extract.mcmc(path = cas2_file_dir, samples.file = "samples.5", objectives.file = "object
mcmc_3 = extract.mcmc(path = cas2_file_dir, samples.file = "samples.6", objectives.file = "object
## assign chain label
mcmc_1$chain = "1"
mcmc_2$chain = "2"
mcmc_3$chain = "3"
## Remove burn-in iterations
mcmc_post_1 = mcmc_1 %>% filter(state == "mcmc")
mcmc_post_2 = mcmc_2 %>% filter(state == "mcmc")
mcmc_post_3 = mcmc_3 %>% filter(state == "mcmc")
## combine
mcmc_all = rbind(mcmc_1, mcmc_2, mcmc_3)
```

```
mcmc_non_burn_in = mcmc_all %>% filter(state == "mcmc")
n_posterior_samples =  nrow(mcmc_post_1) + nrow(mcmc_post_2) + nrow(mcmc_post_3)
## do some modifying so we have just parameters available
## TODO: change parameter labels so they are not so large and easier to read on figure
pars = colnames(mcmc_post_1[,12:(ncol(mcmc_non_burn_in) - 1)])
iters = max(nrow(mcmc_post_1), nrow(mcmc_post_2), nrow(mcmc_post_3))
bayes_array = array(dim = c(iters, 3, length(pars)), dimnames = list(1:iters, 1:3, pars
bayes_array[1:nrow(mcmc_post_1),1,] = as.matrix(mcmc_post_1[,12:(ncol(mcmc_non_burn_in
bayes_array[1:nrow(mcmc_post_2),2,] = as.matrix(mcmc_post_2[,12:(ncol(mcmc_non_burn_in
bayes_array[1:nrow(mcmc_post_3),3,] = as.matrix(mcmc_post_3[,12:(ncol(mcmc_non_burn_in
## cut off at min
min_cutoff = min(nrow(mcmc_post_1), nrow(mcmc_post_2), nrow(mcmc_post_3))
bayes_array = bayes_array[1:min_cutoff, ,]
```

6.2 Diagnostics

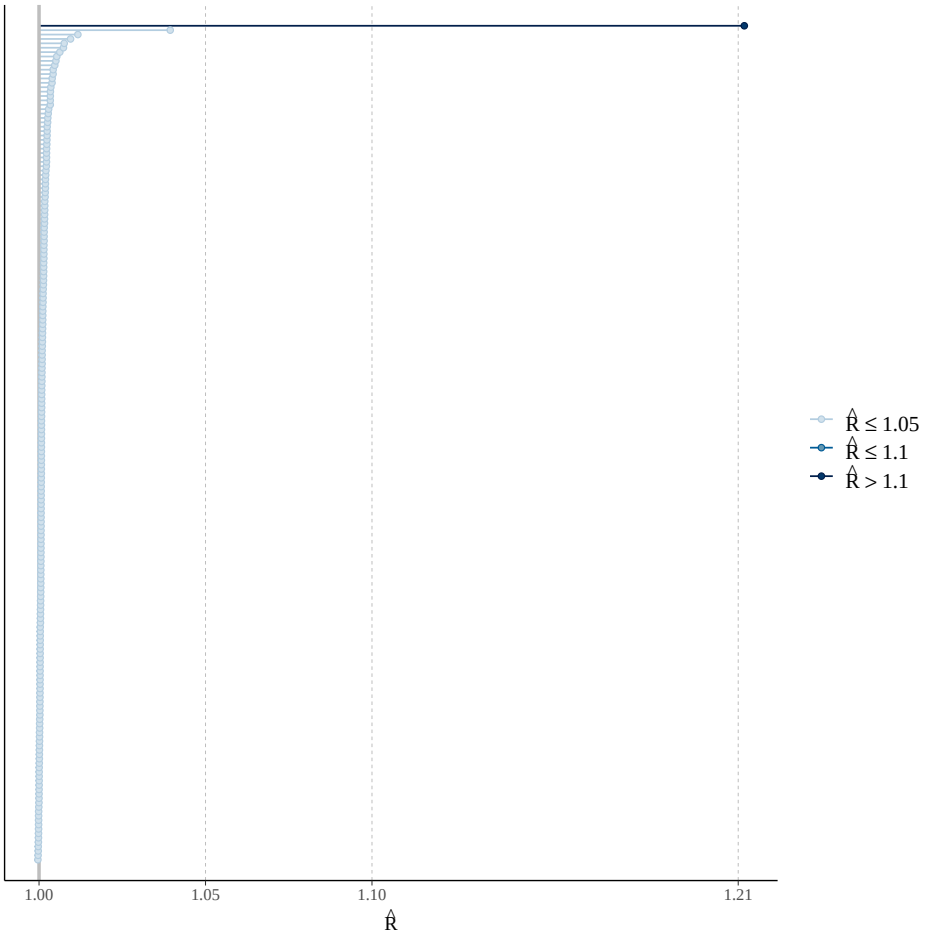
```
## get Rhats
rhats = apply(bayes_array, MARGIN = 3, Rhat)
## get effective sample sizes
n_eff_bulk = apply(bayes_array, MARGIN = 3, ess_bulk)
n_eff_tail = apply(bayes_array, MARGIN = 3, ess_tail)
## TODO: need to think about what is a good general rule of thumb.
## I was thinking you would want n_eff of at least 200.
```

The `Rhat` function produces R-hat convergence diagnostic, which compares the between- and within-chain estimates for model parameters and other univariate quantities of interest. Chains that have not mixed well (i.e., the between-and within-chain estimates don't agree) will result in R-hat larger than 1.1. The `ess_bulk` function produces an estimated Bulk Effective Sample Size (bulk-ESS) using rank normalized draws. Bulk-ESS is useful measure for sampling efficiency in the bulk of the distribution (related e.g. to efficiency of mean and median estimates), and is well defined even if the chains do not have finite mean or variance. The `ess_tail` function produces an estimated Tail Effective Sample Size (tail-ESS) by computing the minimum of effective sample sizes for 5% and 95% quantiles. Tail-ESS is useful measure for sampling efficiency in the tails of the distribution (related e.g. to efficiency of variance and tail quantile estimates).

Once you have calculated these quantities you can use `bayesplot` plotting functions. Need to work on changing parameter labels from the Casal2 model. They make some of these plots difficult to read

```
mcmc_rhat(rhats)
```

```
## Warning: Dropped 6 NAs from 'new_rhat(rhat)'.
```




```
#mcmc_neff(n_eff_bulk)
#mcmc_neff(n_eff_tail)
```

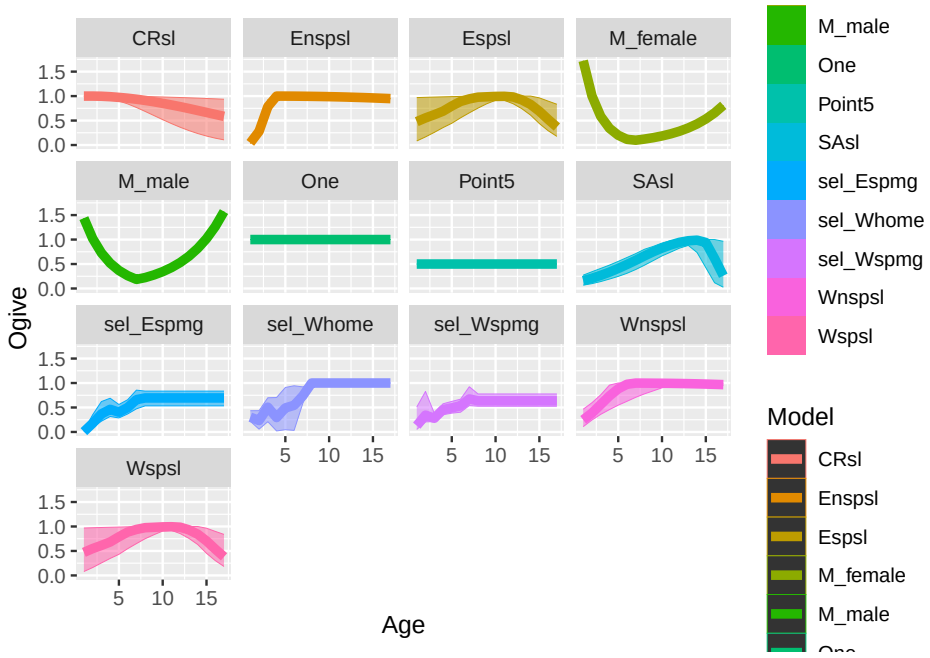
6.3 Plotting quantities

```
## This function helps create 95% CIs for quantities
p <- c(0.025, 0.5, 0.975) ## confidence intervals
p_names <- map_chr(p, ~paste0(.x*100, "%"))
p_funs <- map(p, ~partial(quantile, probs = .x, na.rm = TRUE)) %>%
  rlang::set_names(nm = c("low", "mid", "upp"))
#p_funs

## Bring in derived quantities
cas2_file_name = system.file("extdata", "tabular.log", package = "r4Casal2", mustWork = TRUE)
```

```
cas2_tab = extract.tabular(file = cas2_file_name, quiet = T)
## cut off burn-in the first 50 samples
cas2_tab = burn.in.tabular(cas2_tab, Row = 50)
```

6.3.1 selectivities

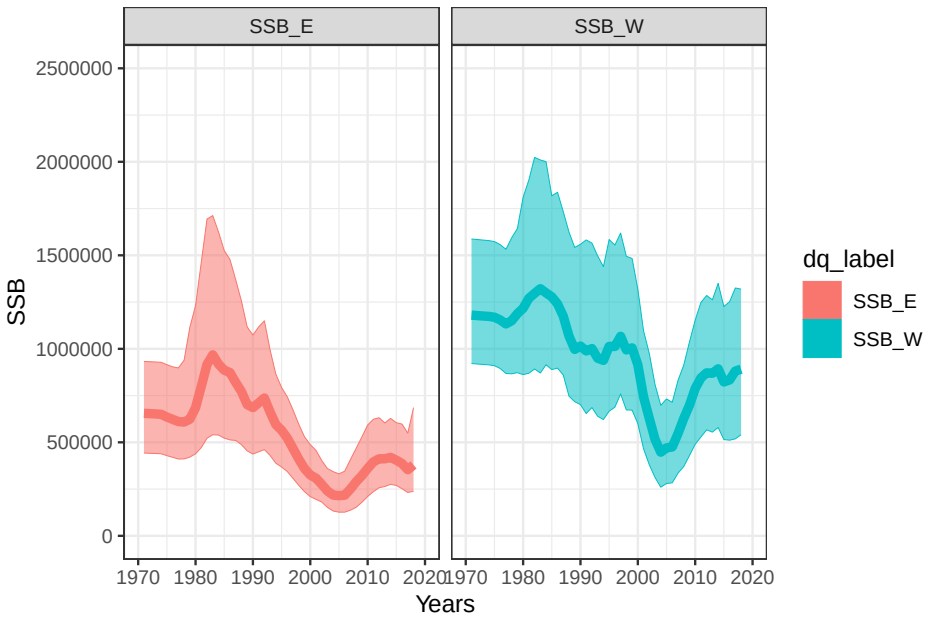
```
selectivity_df = get_selectivities(cas2_tab)
quantile_selectivity_df = selectivity_df %>%
  group_by(bin, selectivity_label) %>%
  summarize_at(vars(selectivity), p_funs)

ggplot(quantile_selectivity_df, aes(x = bin)) +
  geom_ribbon(aes(ymax  = low, ymin = upp, alpha = 0.5, col = selectivity_label, fill =
  geom_line(aes(y = mid, col = selectivity_label, group = selectivity_label), size =2,
  facet_wrap(~selectivity_label) +
  labs(x = "Age", y = "Ogive", col = "Model", linetype = "Model")
```



6.3.2 Derived quantities

```
# plot Ssbs
ssbs = get_derived_quantities(model = cas2_tab)

## getting values for  SSB_E
```

```
## getting values for  SSB_W
#ssbs_mpd = get_derived_quantities(model = mpd)
#head(ssbs)
#ssbs_mpd$years = as.numeric(ssbs_mpd$years)

ssbs$years[ssbs$years == "initialisation_phase_1"] = 1971

quantile_ssb_df = ssbs %>%
  group_by(years, dq_label) %>%
  summarize_at(vars(values), p_funs)

quantile_ssb_df$years = as.numeric(quantile_ssb_df$years)
##
quant_ssb_plot = ggplot(quantile_ssb_df, aes(x = years)) +
  geom_ribbon(aes(ymax  = low, ymin = upp, alpha = 0.5, col = dq_label, fill = dq_label), lwd=0)
  theme_bw() +
  geom_line(aes(y = mid, col = dq_label, group = dq_label), size =2, alpha = 1) +
  xlab("Years") +
  ylab("SSB") +
  ylim(0, 2500000) +
  xlim(1970, 2020) +
  scale_alpha(guide = 'none') +
  #geom_line(data = ssbs_mpd, aes(x = years, y = values), inherit.aes = F, col = "black", size =
  facet_wrap(~dq_label)
quant_ssb_plot
```



Chapter 7

Posterior Predictive Checks

7.1 Introduction

This vignette demonstrates how to use Casal2s simulation mode with `r4Casal2` R functions to generate posterior predictive checks for goodness of fit measures. In terms of assessment workflow, this falls in the Diagnostic component.

The following vignette uses the Casal2 model embedded into this R package. If you want to see where this is on you system paste the following line of code into your R console `system.file("extdata", "PosteriorPredictiveChecks", package = "r4Casal2")`

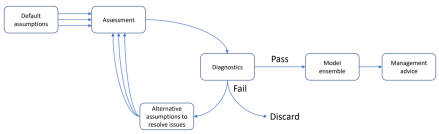


Figure 7.1: Assessment Process

7.2 Estimation

Before looking at data goodness of fit you should be checking if the model has converged. We assume that the estimated model has satisfied this criteria i.e. invertable covariance, acceptable gradient (close to zero) and global minima (as opposed to local - try re-estimating with jittered start values to test for this).

```
library(DHARMa)
library(mvtnorm) ## if simulating obs at MPD
```

```
mpd_file_name = system.file("extdata", "PosteriorPredictiveChecks","estimate.log",
                            package = "r4Casal2", mustWork = TRUE)
mpd = extract.mpd(file = mpd_file_name)
```

```
## WARNING: The output file was generated with a different version than the R libray be
## This may cause compatibility issues. Please update the R package to be consistent wi
## The output was generated with Casal2 v(c) 2
## The Casal2 R package is compatible with Casal2 v23.06
```

```
# Report labels
names(mpd)
```

```
##  [1] "header"               "obj"                  "estimate_summary"     "covar"
##  [6] "estimate_value"       "biomass_t1"           "chatOBSest"           "chatOBSws
## [11] "chatTANbiomass"       "Recruitment"          "Ageing"               "instant_r
## [16] "westFSel"             "eastFSel"             "chatTANSel"           "One"
```

```
# is covariance symmetric
isSymmetric(mpd$covar$covariance_matrix)
```

```
## [1] TRUE
```

```
# is hessian invertable
is_matrix_invertable(mpd$hess$hessian_matrix)
```

```
## [1] TRUE
```

7.3 Simulations

The first thing you should do is add reports of type `simulated_observation` for each observation in your Casal2 configuration files. The helper function `?create_simulation_reports` will automatically generate a casal2 compatible reports to a file named `simulated_reports.csl2` containing all simulated observations in your configuration files. If you use this function you will need to then add an include statement into your Casal2 config files i.e. `!include simulated_reports.csl2` before running casal2 in simulation mode `casal2 -s 1`.

```
config_dir = system.file("extdata", "PosteriorPredictiveChecks", package = "r4Casal2",
config_files = "Observation.csl2"
## this will create the file 'simulated_reports.csl2' in config_dir
## in addition to creating the directory labelled output_folder
obs = create_simulation_reports(config_dir = config_dir, config_files = config_files,
                                output_folder = "simulated_observations")
## append include statement
if(FALSE)
  cat("!include simulated_reports.csl2", file =
```

```
        file.path(config_dir, "config.csl2"), append = T)
## run Casal2 -s
# system(paste0("cd ", config_dir, "; casal2 -s 100 -i pars" ))
```

If you don't have these reports in your configuration files, Casal2 will not save simulated observations. Tips when specifying this report class

1. Save each simulated observation into a separate `file_name`
2. Create a directory to save simulated data sets in.
3. Have the report label the same as the `file_name` (see example below)
4. Avoid the use of periods/dots (".") in `file_name`

An example report structure would look like

```
@report sim_chatTANage
type simulated_observation
observation chatTANage
file_name simulated_observations/sim_chatTANage
```

There are three variants of simulations you can conduct in Casal2, and these depend on if you are in MPD or MCMC estimation phase. If you are evaluating a MPD run, there are two variants and depend if you want to account for parameter uncertainty or not. If you don't want parameter uncertainty, then you need to run the following Casal2 command to produce 100 sets of simulations `casal2 -s 100 -i mpd_pars.log > simulate.log`. If you want to account for parameter uncertainty then you can use a multivariant normal distribution with mean equal to MPD and resulting covariance to produce a set of simulations, example below.

```
n_sims = 1000
## NOTE: might have issue with bounds assuming normal dist
sims = rmvnorm(n = n_sims, mean = as.numeric(mpd$estimate_value$values),
               sigma = mpd$covar$covariance_matrix)
dim(sims)
```

```
## [1] 1000   46
```

```
colnames(sims) = names(mpd$estimate_value$values)
## save simulated pars in the same directory as your
## CSL files
if(FALSE)
  write.table(sims, file = "mpd_mvnorm_pars.csl2", quote = F, row.names = F, col.names = T)
# run
# casal2 -s 1 -i mpd_mvnorm_pars.csl2 > simulate.log
```

7.4 Summarising simulated data in R

Assuming you have saved all the simulated observations as separate files in a standalone folder.

```
sim_dir = system.file("extdata", "PosteriorPredictiveChecks","simulated_observations",
                      package = "r4Casal2", mustWork = TRUE)

## created from running simulations and reading them in with
# sim_vals = read.simulated.data(dir = sim_dir, mean_age = F)
# saveRDS(sim_vals, file.path("sim_vals.RDS"))
sim_vals = readRDS(file.path(sim_dir, "sim_vals.RDS"))

# check no trouble with files
sim_vals$failed_files
```

```
## logical(0)
```

```
#
names(sim_vals$sim_obs)
```

```
## [1] "sim_chatOBSest"     "sim_chatOBSwst"     "sim_chatTANage"     "sim_chatTANbioma
```

```
sim_dir_alt = system.file("extdata", "PosteriorPredictiveChecks","simulated_observatio
                          package = "r4Casal2", mustWork = TRUE)

#sim_vals_alt = read.simulated.data(dir = sim_dir_alt, mean_age = F)
#saveRDS(sim_vals_alt, file.path("sim_vals_alt.RDS"))
sim_vals_alt = readRDS(file.path(sim_dir_alt, "sim_vals_alt.RDS"))

# check no trouble with reading in files
sim_vals_alt$failed_files
```

```
## logical(0)
```

```
#
names(sim_vals_alt$sim_obs)
```

```
## [1] "sim_chatOBSest"     "sim_chatOBSwst"     "sim_chatTANage"     "sim_chatTANbioma
```

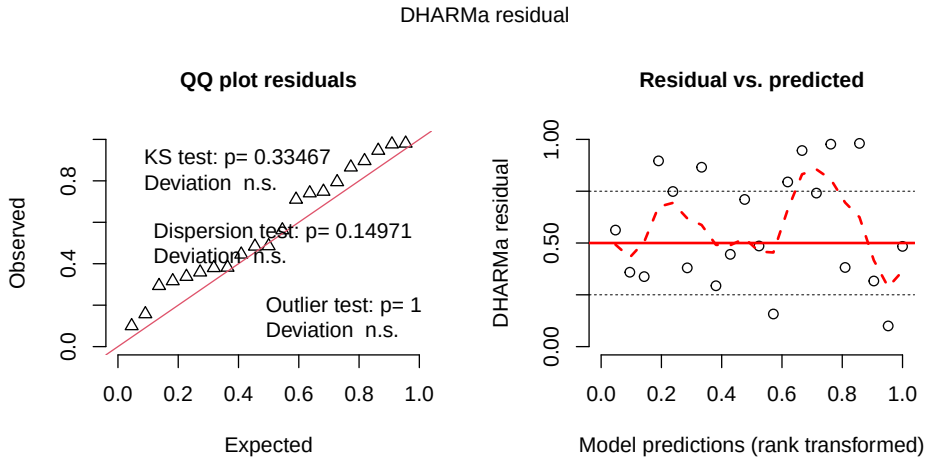
7.5 Posterior predictive checks

Once simulated data has been read into the R environment, we want to compare where the observed values fall relative to the posterior predictive distributions. We recommend using the DHarma r package for this. To interpret P-values or understand the test-statistics that DHARMa does copy this into your R console `vignette("DHARMa", package="DHARMa")` (Assuming you have installed this package).

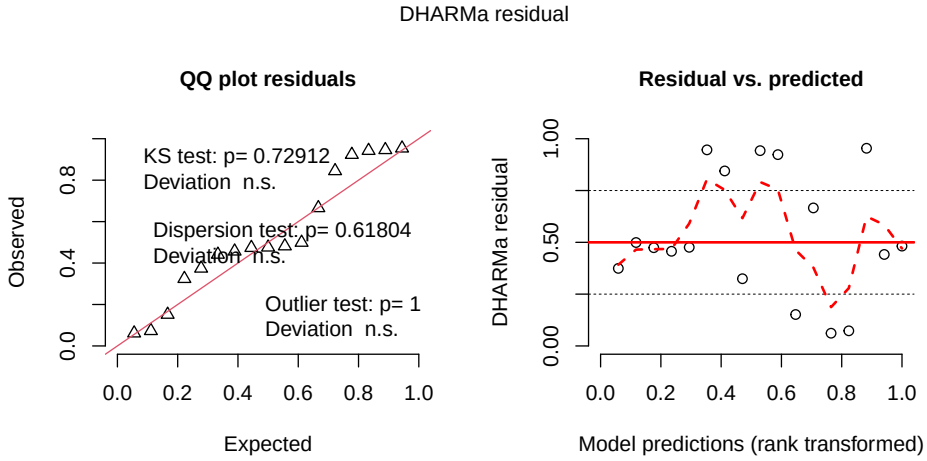
```
## Create DHARMa objects and P-values
DHARMaResbio = createDHARMa(simulatedResponse = sim_vals$sim_obs$sim_chatTANbiomass,
  observedResponse = mpd$chatTANbiomass$Values$observed,
  fittedPredictedResponse = mpd$chatTANbiomass$Values$expected, integerResponse = F)
## Create DHARMa objects and P-values
## for AF
mpd$chatTANage$Values$numbers_at_age = mpd$chatTANage$Values$observed * mpd$chatTANage$Values$err
year = 1999
obs = mpd$chatTANage$Values$numbers_at_age[mpd$chatTANage$Values$year == year]
DHARMaResAF = createDHARMa(simulatedResponse = sim_vals$sim_obs$sim_chatTANage[[as.character(year
  fittedPredictedResponse = NULL, integerResponse = F)
```

```
## No fitted predicted response provided, using the mean of the simulations
```

```
## use DHARMa functions
plot(DHARMaResbio, quantreg = F)
```




```
plot(DHARMaResAF, quantreg = F)
```

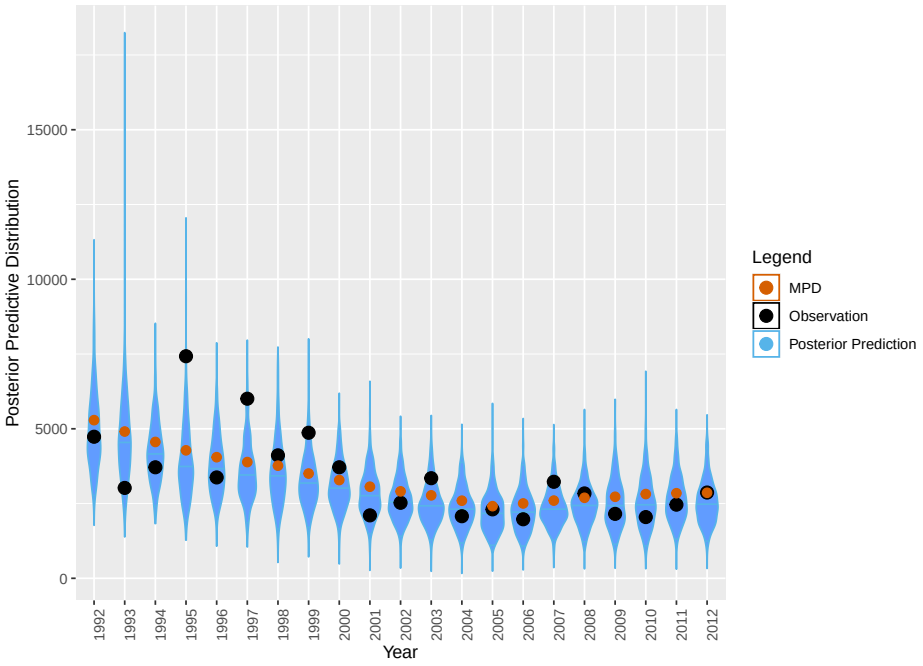


Examples of custom plots

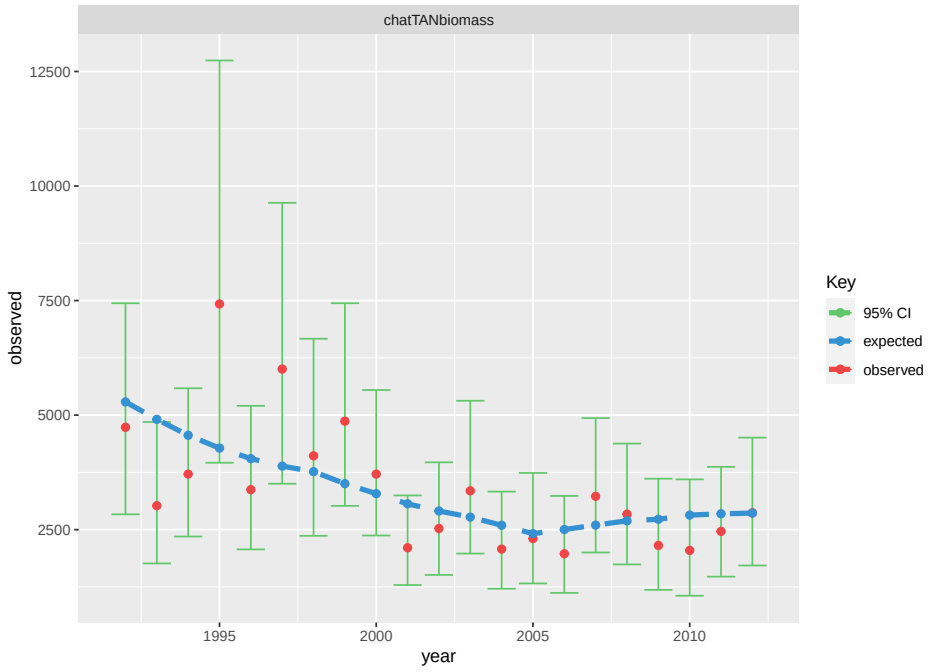
```
sim_chatTANbiomass = sim_vals$sim_obs$sim_chatTANbiomass
rownames(sim_chatTANbiomass) = mpd$chatTANbiomass$Values$year
DHARMaResbio_quant_resids = createDHARMa(simulatedResponse = sim_chatTANbiomass,
                                observedResponse = mpd$chatTANbiomass$Values$observed,
                                fittedPredictedResponse = mpd$chatTANbiomass$Values$expecte
## convert from uniform variable -> standard normal
norm_quant_resids = qnorm(DHARMaResbio_quant_resids$scaledResiduals)
# formal tests
#dispersion_test = testDispersion(DHARMaResbio_quant_resids)
#zero_inflat_test = testZeroInflation(DHARMaResbio_quant_resids)
temporal_correlation = testTemporalAutocorrelation(simulationOutput = DHARMaResbio_quan
temporal_correlation
```

```
##
##  Durbin-Watson test
##
## data:  simulationOutput$scaledResiduals ~ 1
## DW = 1.9861, p-value = 0.9743
## alternative hypothesis: true autocorrelation is not 0
```

```
## plot posterior predictive checks
bioplt = plot_abundance_predictive_dist(sim_data = sim_chatTANbiomass,
                                obs = data.frame(obs = mpd$chatTANbiomass$Values$observed
                                                 year = mpd$chatTANbiomass$Values$year,
                                                 mpd_fit = mpd$chatTANbiomass$Values$expe
                                lab = "", plot_type = "violin")
```




```
bioplt= bioplt + theme(axis.text.x = element_text(angle = 90)) +
  ggtitle("MPD Predictive distributions")
## qq plot for simulated quantile residuals.
norm_resid = ggplot(data.frame(resids = norm_quant_resids), aes(sample = resids)) +
  stat_qq(size = 3) +
  stat_qq_line(col = "steelblue", size = 1.5, linetype = "dashed") +
  labs(x = "Theoretical", y = "Sample")
## Plot normalised residuals generated from Casal2
normalised_resids = ggplot(mpd$chatTANbiomass$Values, aes(sample = normalised_residuals)) +
  stat_qq(size = 3) +
  stat_qq_line(col = "steelblue", size = 1.5, linetype = "dashed") +
  labs(x = "Theoretical", y = "Sample")
## Plot fits obs
fits = plot_relative_index(model = mpd, report_label = "chatTANbiomass") + ggtitle("")
##
fits
```

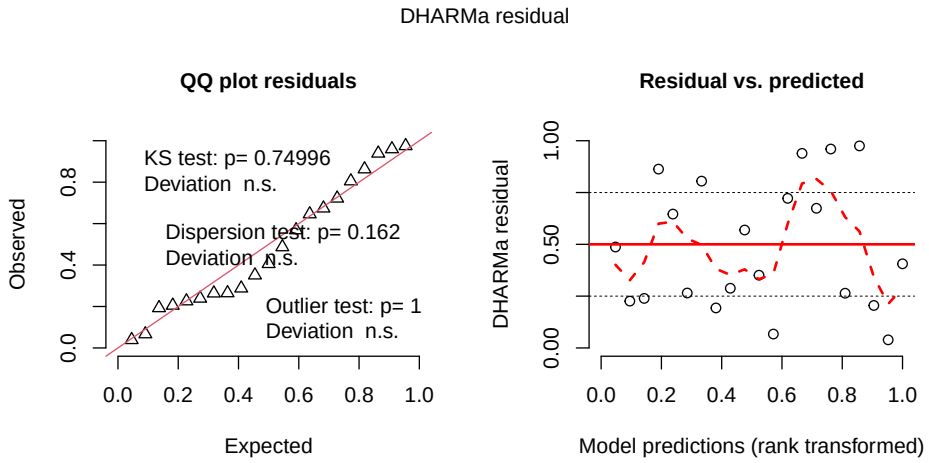


Look at Predictive checks when generated with variability in parameter estimates as well.

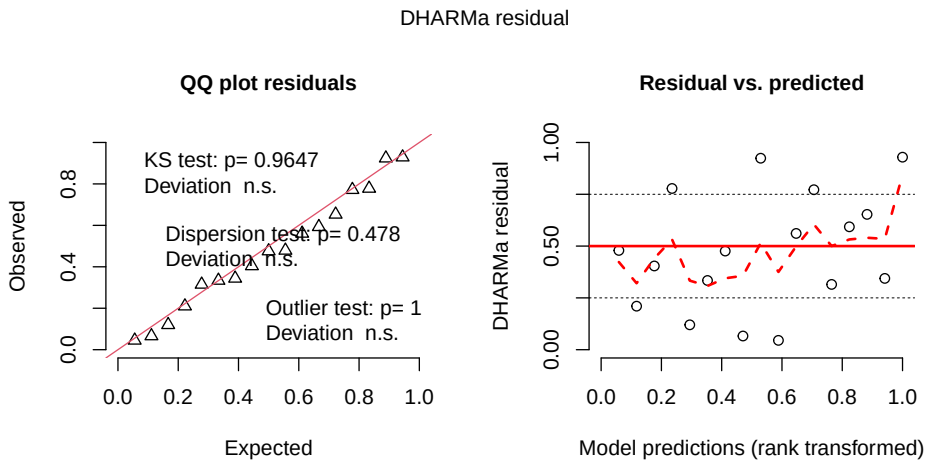
```
## Create DHARMa objects and P-values
DHARMaResbio = createDHARMa(simulatedResponse = sim_vals_alt$sim_obs$sim_chatTANbiomass
  observedResponse = mpd$chatTANbiomass$Values$observed,
  fittedPredictedResponse = mpd$chatTANbiomass$Values$expected, integerResponse = F)
## Create DHARMa objects and P-values
## for AF
year = 2000
obs = mpd$chatTANage$Values$numbers_at_age[mpd$chatTANage$Values$year == year]

DHARMaResAF = createDHARMa(simulatedResponse = sim_vals_alt$sim_obs$sim_chatTANage[[as
  fittedPredictedResponse = NULL, integerResponse = T)
```

```
## No fitted predicted response provided, using the mean of the simulations
```

```
plot(DHARMaResbio, quantreg = F)
```

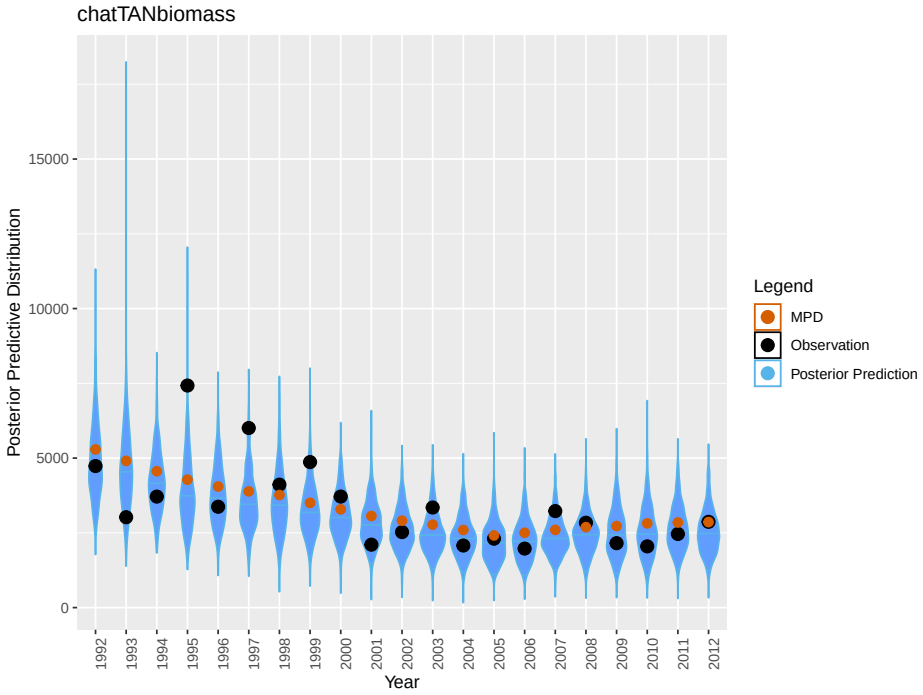



```
plot(DHARMaResAF, quantreg = F)
```

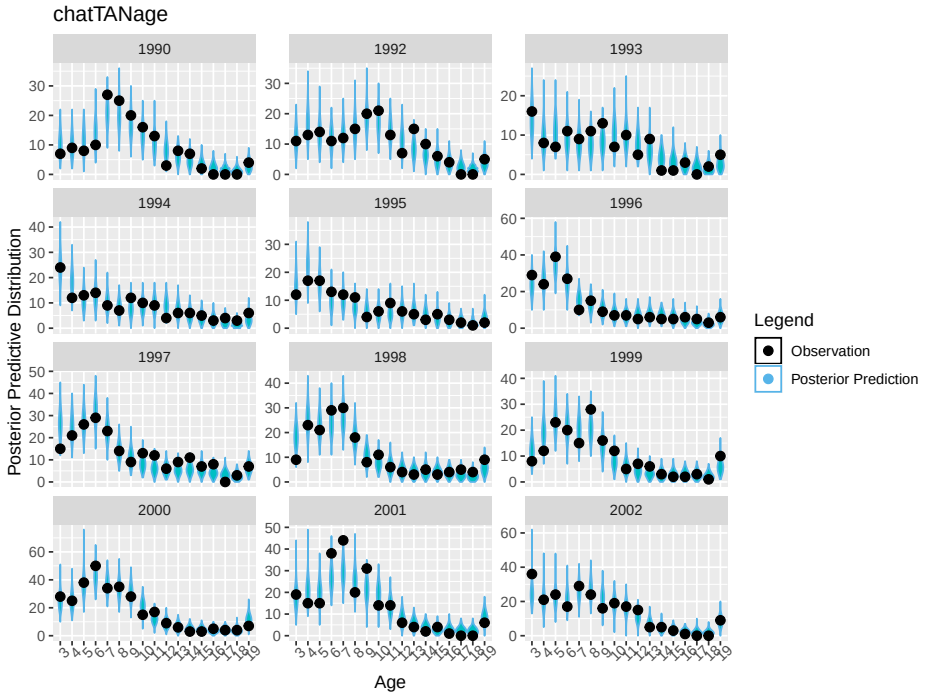


Some other visualizations plots

```
#########################
## boxplot predictive distribution vs observation
legend = c("Posterior Prediction" = "#56B4E9", "Observation" = "black" , "MPD" = "#D55E00")

sim_data = sim_vals$sim_obs$sim_chatTANbiomass
rownames(sim_data) = mpd$chatTANbiomass$Values$year

bioplt = plot_abundance_predictive_dist(sim_data = sim_data,
    obs = data.frame(obs = mpd$chatTANbiomass$Values$observed,
    year = mpd$chatTANbiomass$Values$year,
    mpd_fit = mpd$chatTANbiomass$Values$expected),
    lab = "chatTANbiomass", plot_type = "violin")
```




```
chat_sim_vals = get_simulated_age_resids(sim_vals$sim_obs$sim_chatTANage, mpd$chatTANag
obs_years = unique(chat_sim_vals$mpd_df$year)
years_to_plot = obs_years[1:12]
pppAFplt_1 =  ggplot(chat_sim_vals$full_simulated_values %>%
    filter(year %in% years_to_plot), aes(x = factor(age, ordered = T), y = simulated_va
  geom_violin(aes(color = "Posterior Prediction", fill = "Posterior Prediction"),adjust
  scale_color_manual(values = legend) +
  guides(fill = "none") +
  theme(axis.text.x = element_text(angle = 45)) +
  labs(colour = "Legend", x = "Age", y = "Posterior Predictive Distribution") +
  geom_point(data = chat_sim_vals$mpd_df %>% filter(year %in% years_to_plot), aes(x = a
  facet_wrap(~year, scales = "free_y", ncol = 3)    +
  ggtitle("chatTANage")
pppAFplt_1
```




```
OM_file = system.file("extdata", "PosteriorPredictiveChecks","OM","OM_vary.log", package = "r4Cas
OM_run = extract.mpd(file = OM_file)
```

```
## WARNING: The output file was generated with a different version than the R libray being used t
## This may cause compatibility issues. Please update the R package to be consistent with the ver
## The output was generated with Casal2 v(c) 2
## The Casal2 R package is compatible with Casal2 v23.06
##
## loading a Casal2 output from a multi parameter input format
## loading a Casal2 output from a multi parameter input format
```

```
## plot SSBs
my_plot = r4Casal2::plot_derived_quantities(model = list(OM = OM_run, EM = mpd))
my_plot = my_plot + xlab("SSB") + ylim(0, 120000)
```

```
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```

7.6 PIT residuals

Pearson residuals for multinomial distributed random variables can be difficult to interpret for a lot of reasons, data-weighting for mean age to standardised residuals = 1 [Francis, 2011], sparsity can create funny patterns etc. An alternative is to use randomised quantile PIT residuals [Warton et al., 2017, Dunn

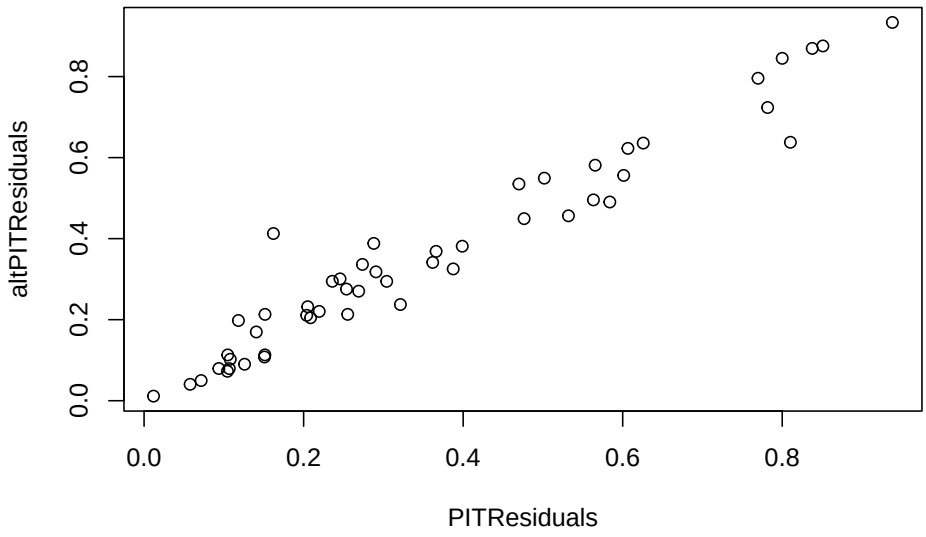
and Smyth, 1996]

Assuming we have data denoted by y which has cumulative distribution function $F(y;\theta), u = F(y;\theta) \sim Uniform(0,1)$. For discrete variables the following adjustment can be made

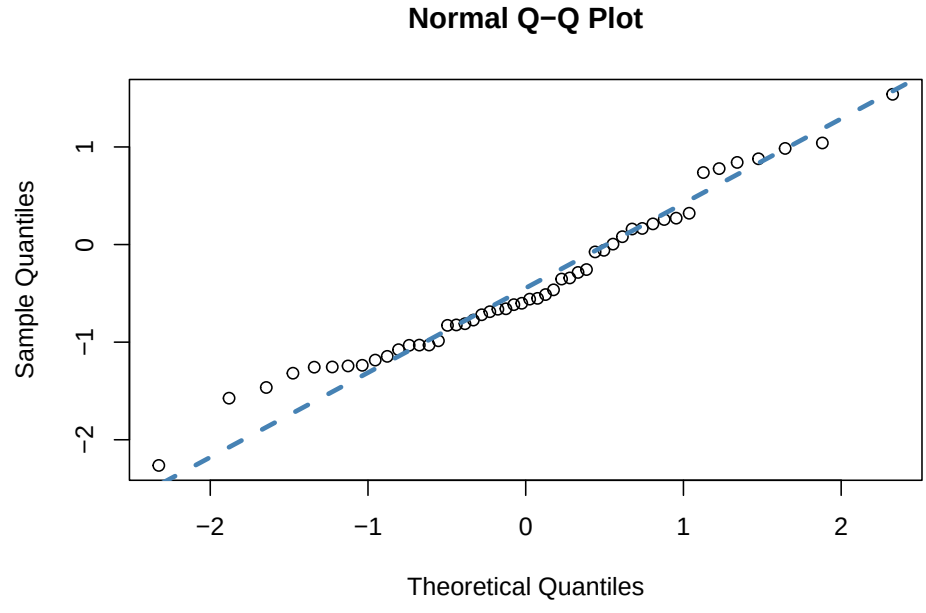
$$u_i = q_i F(y;\theta) + (1 - q_i)F(y^-;\theta) \tag{7.1}$$

where, q_i is a standard uniform random variable and y^- is the previous allowable value for y. they do not behave like residuals in the usual sense they are centred around a value of 0.5 rather than a value of 0, and are bounded between 0 and 1.

```
# hello
set.seed(123)
n = 50
nsims = 1000
x = rnorm(n, 5, 3)
beta0 = 3
beta1 = 1.2
true_beta1 = 1
y = rpois(n, beta0 + true_beta1 * x)
y_sim = matrix(nrow = n, ncol = nsims)
for(i in 1:nsims)
  y_sim[,i] = rpois(n, beta0 + beta1 * x)
## calculate PIT
PITResiduals = rep(NA, n)
altPITResiduals = rep(NA, n)
for (i in 1:n){
  minSim <- mean(y_sim[i,] < y[i])
  maxSim <- mean(y_sim[i,] <= y[i])
  if (minSim == maxSim) {
    PITResiduals[i] = minSim
  } else {
    PITResiduals[i] = runif(1, minSim, maxSim)
  }
  qi = runif(1)
  lower_limit = mean(y_sim[i,] < y[i])
  cum_ecdf = mean(y_sim[i,] <= y[i])
  altPITResiduals[i] = qi * cum_ecdf + (1 - qi) * lower_limit
}
plot(PITResiduals, altPITResiduals)
```




```
# It is assumed that PIT residuals are from uniform(0,1)
# most people transform them to normal distribution for testing
# for familiarity more than anything.
normal_transformed = qnorm(PITResiduals)
qqnorm(normal_transformed)
qqline(normal_transformed, col = "steelblue", lwd = 3, lty = 2)
```




```
shapiro.test(normal_transformed)
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  normal_transformed
## W = 0.97169, p-value = 0.2707
```

```
# From the output, the p-value > 0.05 implying that the distribution
# of the data are not significantly different from normal distribution.
# In other words, we can assume the normality.
```

Chapter 8

Presenting Models using Bookdown

Recently there has been a lot of development with respect to using R Shiny apps to present stock assessment models to technical working groups. A problem with Shiny is it requires you to host the app somewhere which can encounter permission and confidentiality issues. An approach that I believe is worth exploring is using the R package `library(bookdown)`. This package allows users to bundle results into an html book that can be accessed locally by opening the html files within a web browser. This would allow it for easy distribution of HTML files to the necessary parties/stakeholders, in addition to it being easily archived by administrators (which is difficult with shiny apps).

Often when presenting Casal2 models we present a suite of models. This means each presentation will be bespoke to the problem as you may want to emphasis different model characteristics. That being said, this chapter will walk through an example that I have recently been working on, with the aim of providing a template for others to use.

The main function that has been created for this task is `?build_assessment_bookdown`.

Bibliography

Felipe Carvalho, Henning Winker, Dean Courtney, Maia Kapur, Laurence Kell, Massimiliano Cardinale, Michael Schirripa, Toshihide Kitakado, Dawit Yemane, Kevin R. Piner, Mark N. Maunder, Ian Taylor, Chantel R. Wetzel, Kathryn Doering, Kelli F. Johnson, and Richard D. Methot. A cookbook for using model diagnostics in integrated stock assessments. *Fisheries Research*, 240:105959, 2021. ISSN 0165-7836. doi: https://doi.org/10.1016/j.fishres.2021.105959. URL https://www.sciencedirect.com/science/article/pii/S0165783621000874.

Peter K Dunn and Gordon K Smyth. Randomized quantile residuals. *Journal of Computational and Graphical Statistics*, 5(3):236–244, 1996.

RIC Chris Francis. Data weighting in statistical fisheries stock assessment models. *Canadian Journal of Fisheries and Aquatic Sciences*, 68(6):1124–1138, 2011.

Carolina V. Minte-Vera, Mark N. Maunder, Alexandre M. Aires da Silva, Keisuke Satoh, and Koji Uosaki. Get the biology right, or use size-composition data at your own risk. *Fisheries Research*, 192:114–125, 2017.

Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. Rank-normalization, folding, and localization: An improved r for assessing convergence of mcmc (with discussion). *Bayesian analysis*, 16(2):667–718, 2021.

David I Warton, Loïc Thibaut, and Yi Alice Wang. The pit-trap—a "model-free" bootstrap procedure for inference about regression models with discrete, multivariate responses. *PloS one*, 12(7):e0181790, 2017.