Sardine MSE Results Summary

Robert Wildermuth

3/25/2022

```
# run GetSumryOutput and CalcPerformance to get performance metrics

mseDir <- "J:/Desiree/Sardine/SardineScenarios"
# mseDir <- "C:/Users/r.wildermuth/Documents/FutureSeas/SardineScenarios"
termYr <- 2068

scenarios <- c("constGrow2001OM_constGrow2005EM_RandRecHCR0",
               "constGrow2001OM_constGrow2005EM_RandRecHCR1",
               "constGrow2001OM_constGrow2005EM_RandRecHCR2",
               "constGrow2001OM_constGrow2005EM_RandRecHCR3",
               #"constGrow2001OM_constGrow2005EM_RandRecHCR4",
               "constGrow2001OM_constGrow2005EM_RandRecHCR5",
               "constGrow2001OM_constGrow2005EM_RandRecHCR6",
               "constGrow2001OM_constGrow2005EM_RandRecHCR7",
               "constGrow2001OM_constGrow2005EM_RandRecHCR8")#,

               # "constGrow2001OM_constGrow2005EM_ARRecHCR0",
               # "constGrow2001OM_constGrow2005EM_ARRecHCR1",
               # "constGrow2001OM_constGrow2005EM_ARRecHCR2",
               # "constGrow2001OM_constGrow2005EM_ARRecHCR3",
               # "constGrow2001OM_constGrow2005EM_ARRecHCR4",
               # "constGrow2001OM_constGrow2005EM_ARRecHCR5",
               # "constGrow2001OM_constGrow2005EM_ARRecHCR6",
               # "constGrow2001OM_constGrow2005EM_ARRecHCR7",
               # "constGrow2001OM_constGrow2005EM_ARRecHCR8")#,

               # "constGrow2001OM_constGrow2005EM_SSTRecHCR0",
               # "constGrow2001OM_constGrow2005EM_SSTRecHCR1",
               # "constGrow2001OM_constGrow2005EM_SSTRecHCR2",
               # "constGrow2001OM_constGrow2005EM_SSTRecHCR3",
               # "constGrow2001OM_constGrow2005EM_SSTRecHCR4",
               # "constGrow2001OM_constGrow2005EM_SSTRecHCR5",
               # "constGrow2001OM_constGrow2005EM_SSTRecHCR6",
               # "constGrow2001OM_constGrow2005EM_SSTRecHCR7",
               # "constGrow2001OM_constGrow2005EM_SSTRecHCR8",
               #
               # "constGrow2001OM_constGrow2005EM_ccPDORecHCR0",
               # "constGrow2001OM_constGrow2005EM_ccPDORecHCR1",
               # "constGrow2001OM_constGrow2005EM_ccPDORecHCR2",
               # "constGrow2001OM_constGrow2005EM_ccPDORecHCR3",
               # "constGrow2001OM_constGrow2005EM_ccPDORecHCR4",
               # "constGrow2001OM_constGrow2005EM_ccPDORecHCR5",
```

```
              # "constGrow2001OM_constGrow2005EM_ccPDORecHCR6",
              # "constGrow2001OM_constGrow2005EM_ccPDORecHCR7",
              # "constGrow2001OM_constGrow2005EM_ccPDORecHCR8",
              #
              # "constGrow2001OM_constGrow2005EM_PDORecHCR0",
              # "constGrow2001OM_constGrow2005EM_PDORecHCR1",
              # "constGrow2001OM_constGrow2005EM_PDORecHCR2",
              # "constGrow2001OM_constGrow2005EM_PDORecHCR3",
              # "constGrow2001OM_constGrow2005EM_PDORecHCR4",
              # "constGrow2001OM_constGrow2005EM_PDORecHCR5",
              # "constGrow2001OM_constGrow2005EM_PDORecHCR6",
              # "constGrow2001OM_constGrow2005EM_PDORecHCR7",
              # "constGrow2001OM_constGrow2005EM_PDORecHCR8",
              #
              # "constGrow2001OM_constGrow2005EM_MICERecHCR0",
              # "constGrow2001OM_constGrow2005EM_MICERecHCR1",
              # "constGrow2001OM_constGrow2005EM_MICERecHCR2",
              # "constGrow2001OM_constGrow2005EM_MICERecHCR3",
              # "constGrow2001OM_constGrow2005EM_MICERecHCR4",
              # "constGrow2001OM_constGrow2005EM_MICERecHCR5",
              # "constGrow2001OM_constGrow2005EM_MICERecHCR6",
              # "constGrow2001OM_constGrow2005EM_MICERecHCR7",
              # "constGrow2001OM_constGrow2005EM_MICERecHCR8")


  smryOutputList <- GetSumryOutput(dirSSMSE = mseDir,
                                   scenarios = scenarios)
```

```
## Parsed with column specification:
## cols(
##   Value.SSB = col_double(),
##   Value.Recr = col_double(),
##   Value.SPRratio = col_double(),
##   Value.F = col_double(),
##   Value.Bratio = col_double(),
##   Value.ForeCatch = col_double(),
##   Value.OFLCatch = col_double(),
##   Value.lnSPB = col_double(),
##   year = col_double(),
##   model_run = col_character(),
##   iteration = col_double(),
##   scenario = col_character()
## )
## Parsed with column specification:
## cols(
##   Value.SSB = col_double(),
##   Value.Recr = col_double(),
##   Value.SPRratio = col_double(),
##   Value.F = col_double(),
##   Value.Bratio = col_double(),
##   Value.ForeCatch = col_double(),
##   Value.OFLCatch = col_double(),
##   Value.lnSPB = col_double(),
```

```
##   year = col_double(),
##   model_run = col_character(),
##   iteration = col_double(),
##   scenario = col_character()
## )
## Parsed with column specification:
## cols(
##   Value.SSB = col_double(),
##   Value.Recr = col_double(),
##   Value.SPRratio = col_double(),
##   Value.F = col_double(),
##   Value.Bratio = col_double(),
##   Value.ForeCatch = col_double(),
##   Value.OFLCatch = col_double(),
##   Value.lnSPB = col_double(),
##   year = col_double(),
##   model_run = col_character(),
##   iteration = col_double(),
##   scenario = col_character()
## )
## Parsed with column specification:
## cols(
##   Value.SSB = col_double(),
##   Value.Recr = col_double(),
##   Value.SPRratio = col_double(),
##   Value.F = col_double(),
##   Value.Bratio = col_double(),
##   Value.ForeCatch = col_double(),
##   Value.OFLCatch = col_double(),
##   Value.lnSPB = col_double(),
##   year = col_double(),
##   model_run = col_character(),
##   iteration = col_double(),
##   scenario = col_character()
## )
## Parsed with column specification:
## cols(
##   Value.SSB = col_double(),
##   Value.Recr = col_double(),
##   Value.SPRratio = col_double(),
##   Value.F = col_double(),
##   Value.Bratio = col_double(),
##   Value.ForeCatch = col_double(),
##   Value.OFLCatch = col_double(),
##   Value.lnSPB = col_double(),
##   year = col_double(),
##   model_run = col_character(),
##   iteration = col_double(),
##   scenario = col_character()
## )
## Parsed with column specification:
## cols(
##   Value.SSB = col_double(),
##   Value.Recr = col_double(),
```

```
##   Value.SPRratio = col_double(),
##   Value.F = col_double(),
##   Value.Bratio = col_double(),
##   Value.ForeCatch = col_double(),
##   Value.OFLCatch = col_double(),
##   Value.lnSPB = col_double(),
##   year = col_double(),
##   model_run = col_character(),
##   iteration = col_double(),
##   scenario = col_character()
## )
## Parsed with column specification:
## cols(
##   Value.SSB = col_double(),
##   Value.Recr = col_double(),
##   Value.SPRratio = col_double(),
##   Value.F = col_double(),
##   Value.Bratio = col_double(),
##   Value.ForeCatch = col_double(),
##   Value.OFLCatch = col_double(),
##   Value.lnSPB = col_double(),
##   year = col_double(),
##   model_run = col_character(),
##   iteration = col_double(),
##   scenario = col_character()
## )
## Parsed with column specification:
## cols(
##   Value.SSB = col_double(),
##   Value.Recr = col_double(),
##   Value.SPRratio = col_double(),
##   Value.F = col_double(),
##   Value.Bratio = col_double(),
##   Value.ForeCatch = col_double(),
##   Value.OFLCatch = col_double(),
##   Value.lnSPB = col_double(),
##   year = col_double(),
##   model_run = col_character(),
##   iteration = col_double(),
##   scenario = col_character()
## )
```

```
  # saveRDS(smryOutputList,
  #         file = file.path(mseDir, "serverRandRec_ARRec_allHCRs_results.RDS"))
# smryOutputList <- readRDS(file.path(mseDir, "serverRandRec_ARRec_allHCRs_results.RDS"))

performanceList <- CalcPerformance(smryOutputList)
```

```
## `summarise()` has grouped output by 'iteration'. You can override using the
## `.groups` argument.
```

```
## `summarise()` has grouped output by 'model_run', 'iteration'. You can override
## using the `.groups` argument.
```

```
## `summarise()` has grouped output by 'model_run', 'iteration', 'scenario'. You
## can override using the `.groups` argument.
## `summarise()` has grouped output by 'model_run', 'iteration', 'scenario'. You
## can override using the `.groups` argument.
## `summarise()` has grouped output by 'year', 'model_run', 'iteration'. You can
## override using the `.groups` argument.
## `summarise()` has grouped output by 'model_run', 'iteration'. You can override
## using the `.groups` argument.
## `summarise()` has grouped output by 'model_run', 'iteration'. You can override
## using the `.groups` argument.
## `summarise()` has grouped output by 'model_run', 'iteration'. You can override
## using the `.groups` argument.
```

```
metricsTbl <- performanceList$perfomanceMetrics

metricsTbl
```

```
## # A tibble: 160 x 20
## # Groups:   iteration [20]
##    iteration scenario    nonconvg  nYrs frqNonConvg model_run  yrsN closuresFreq
##        <int> <chr>          <int> <dbl>       <dbl> <chr>     <int>        <dbl>
##  1         8 constGrow2~        1    50        0.02 constGro~    50         0.12
##  2        13 constGrow2~        1    50        0.02 constGro~    50         0.22
##  3        17 constGrow2~        1    50        0.02 constGro~    50         0.48
##  4        17 constGrow2~        1    50        0.02 constGro~    50         0.36
##  5        20 constGrow2~        1    50        0.02 constGro~    50         0.32
##  6        20 constGrow2~        1    50        0.02 constGro~    50         0.32
##  7         1 constGrow2~       NA    NA       NA    constGro~    50         0.06
##  8         1 constGrow2~       NA    NA       NA    constGro~    50         0.48
##  9         1 constGrow2~       NA    NA       NA    constGro~    50         0.08
## 10         1 constGrow2~       NA    NA       NA    constGro~    50         0.1
## # ... with 150 more rows, and 12 more variables: collapseFreq <dbl>,
## #   bonanzaFreq <dbl>, meanB1plus <dbl>, meanCollapseSever <dbl>,
## #   closure <lgl>, rebuildLengthMax <int>, bonanza <lgl>,
## #   bonanzaLengthMax <int>, meanCatch <dbl>, sdCatch <dbl>, minAge <dbl>,
## #   minLen <dbl>
```

#Performance Metrics Plot comparisons across HCR and Recruitment scenario

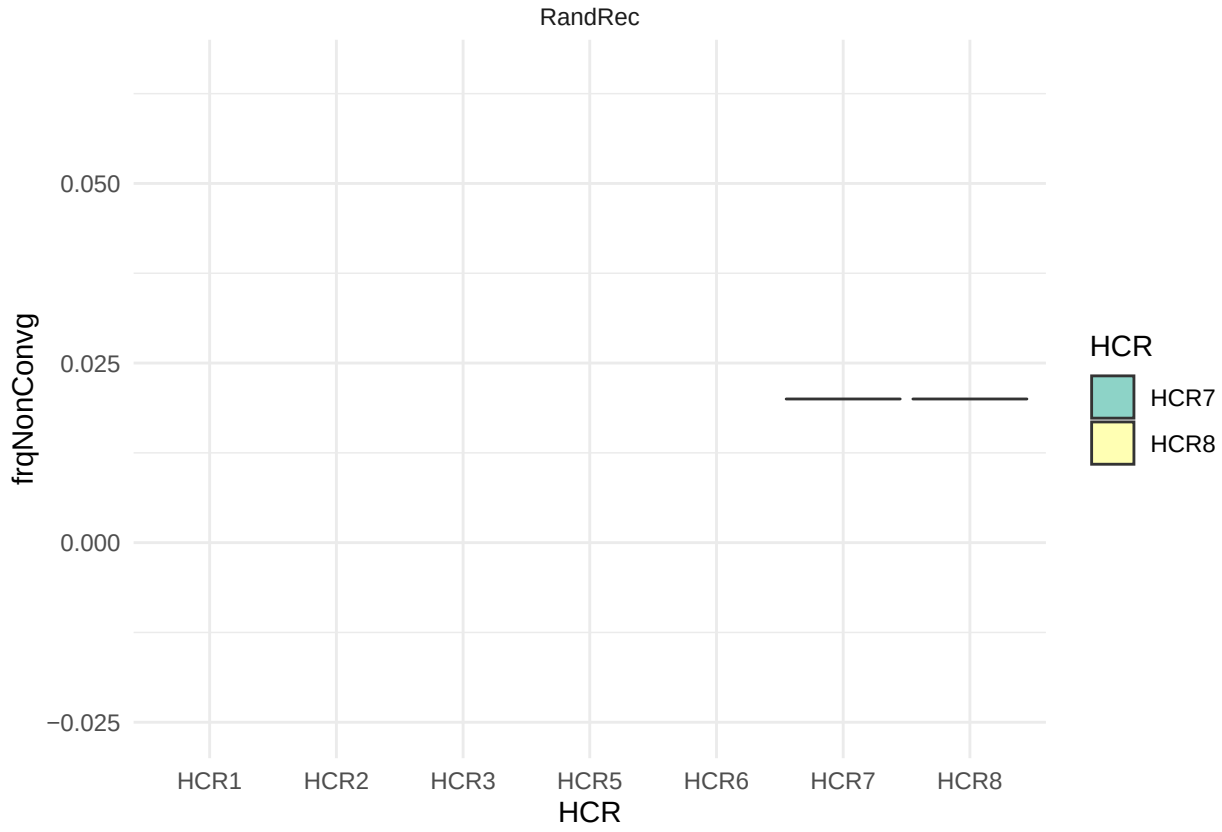
Convergence

```
# parse out HCR and recruitment scenario
metricsTbl <- metricsTbl %>% mutate(HCR = sub(pattern = ".*Rec","", scenario),
                                    recScen = sub(pattern = "HCR.*","", scenario)) %>%
                mutate(recScen = sub(pattern = ".*EM_","", recScen))

hcrPal <- brewer.pal(10, "Set3")[-2]

# plot convergence frequency
metricsTbl %>% filter(HCR != "HCR0") %>%
  ggplot(aes(x = HCR, y = frqNonConvg)) +
```

```
  geom_violin(aes(fill = HCR), draw_quantiles = c(0.1, 0.5, 0.9)) +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_brewer(palette = "Set3")
```

```
## Warning: Removed 134 rows containing non-finite values (stat_ydensity).
```

```
## Warning: Groups with fewer than two data points have been dropped.
## Groups with fewer than two data points have been dropped.
```



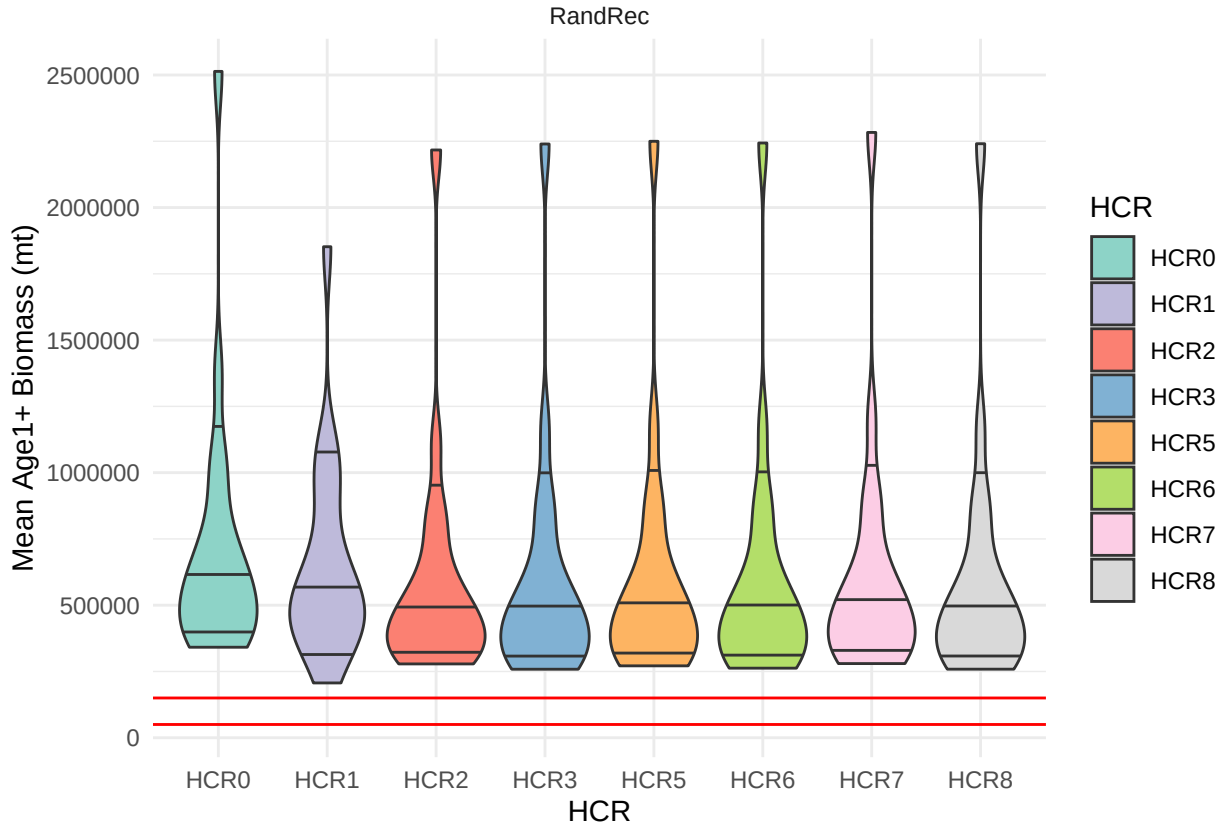

```
metricsTbl %>% group_by(scenario, recScen, HCR) %>%
  summarize(Nnonconvrg = sum(nonconvg, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'scenario', 'recScen'. You can override
## using the `.groups` argument.
```

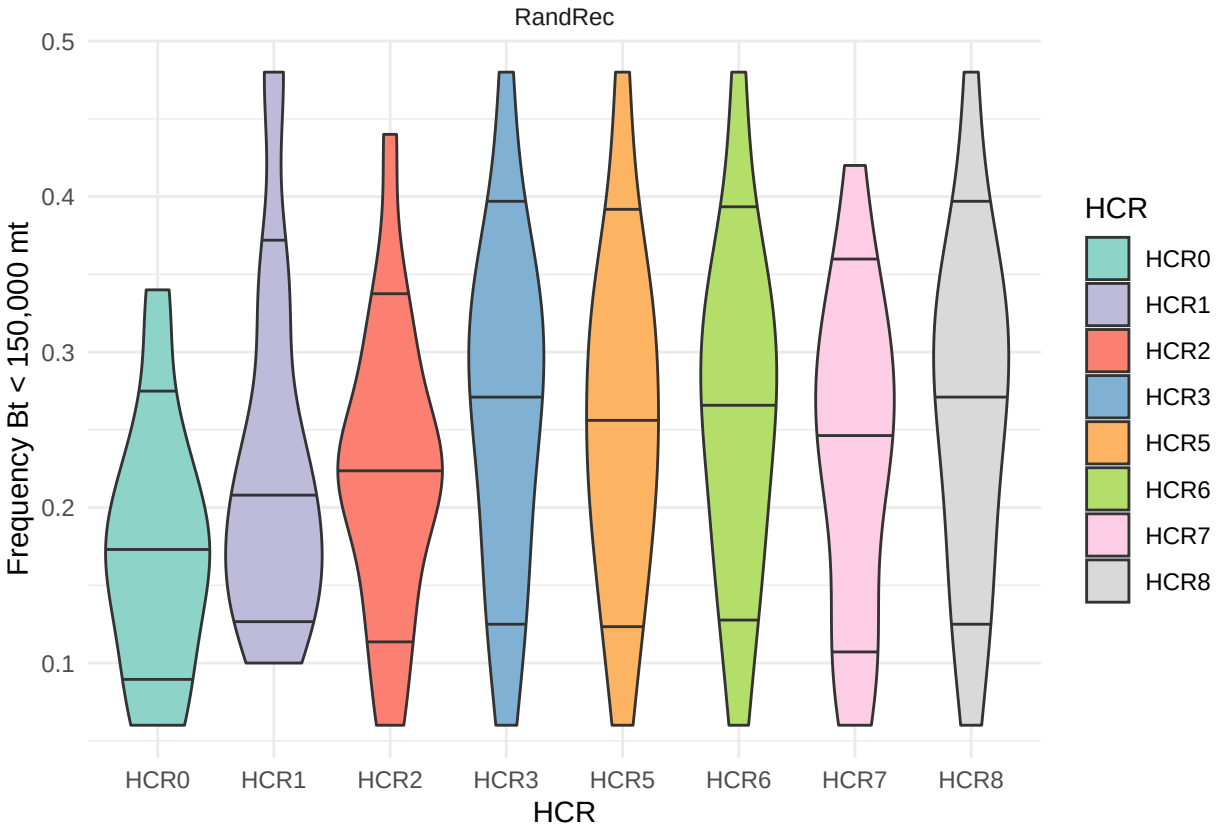
```
## # A tibble: 8 x 4
## # Groups:   scenario, recScen [8]
##   scenario                                      recScen HCR   Nnonconvrg
##   <chr>                                         <chr>   <chr>      <int>
## 1 constGrow2001OM_constGrow2005EM_RandRecHCR0 RandRec HCR0           0
## 2 constGrow2001OM_constGrow2005EM_RandRecHCR1 RandRec HCR1           1
## 3 constGrow2001OM_constGrow2005EM_RandRecHCR2 RandRec HCR2           0
```

```
## 4 constGrow2001OM_constGrow2005EM_RandRecHCR3 RandRec HCR3           0
## 5 constGrow2001OM_constGrow2005EM_RandRecHCR5 RandRec HCR5           0
## 6 constGrow2001OM_constGrow2005EM_RandRecHCR6 RandRec HCR6           1
## 7 constGrow2001OM_constGrow2005EM_RandRecHCR7 RandRec HCR7           2
## 8 constGrow2001OM_constGrow2005EM_RandRecHCR8 RandRec HCR8           2
```

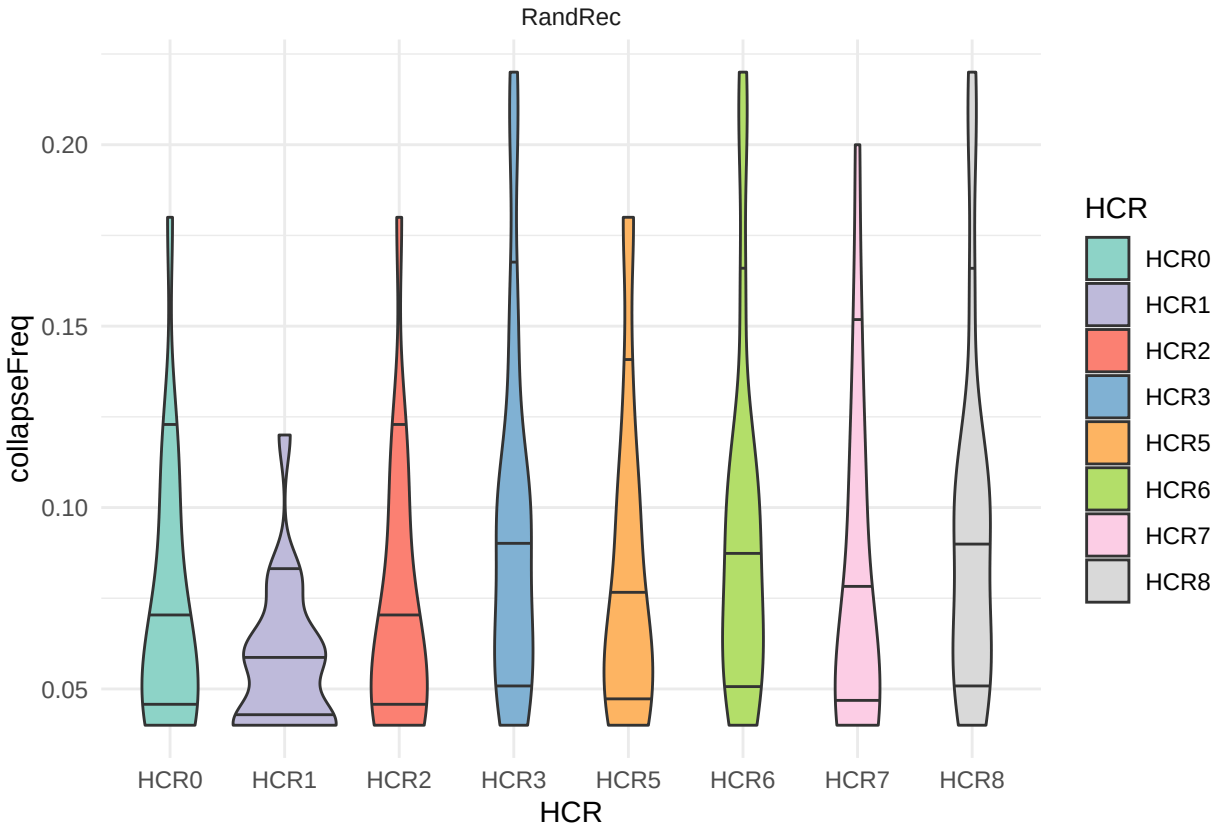
```r
# plot mean biomass
metricsTbl %>% filter(recScen == "RandRec", HCR != "HCR4") %>%
  ggplot(aes(x = HCR, y = meanB1plus)) +
  geom_hline(yintercept = c(50000, 150000), color = "red") +
  geom_violin(aes(fill = HCR), draw_quantiles = c(0.1, 0.5, 0.9)) +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal) +
  labs(y = "Mean Age1+ Biomass (mt)", x = "HCR")
```



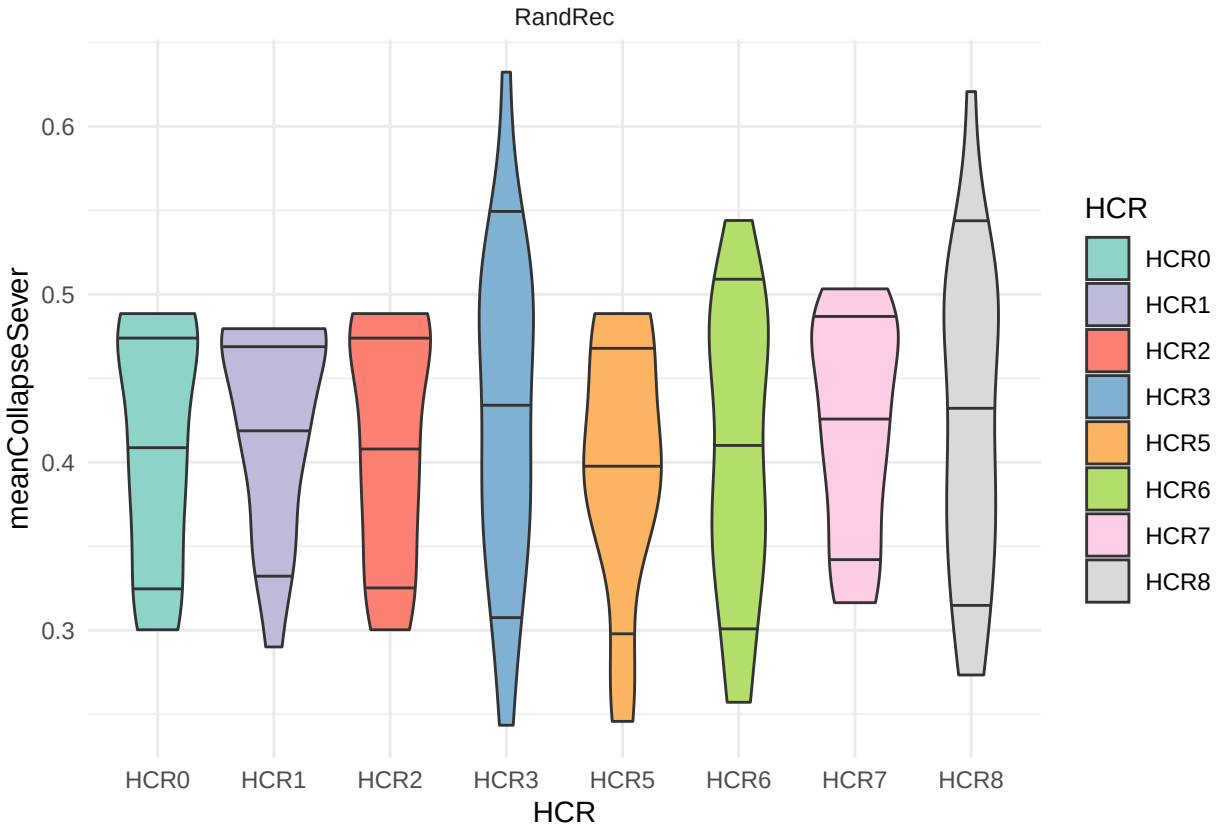

```r
# plot closure frequency
metricsTbl %>% filter(recScen == "RandRec", HCR != "HCR4") %>%
  ggplot(aes(x = HCR, y = closuresFreq)) +
  geom_violin(aes(fill = HCR), draw_quantiles = c(0.1, 0.5, 0.9)) +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal) +
  labs(y = "Frequency Bt < 150,000 mt", x = "HCR")
```




```
# plot collapse frequency
metricsTbl %>% ggplot(aes(x = HCR, y = collapseFreq)) +
  geom_violin(aes(fill = HCR), draw_quantiles = c(0.1, 0.5, 0.9)) +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal)
```



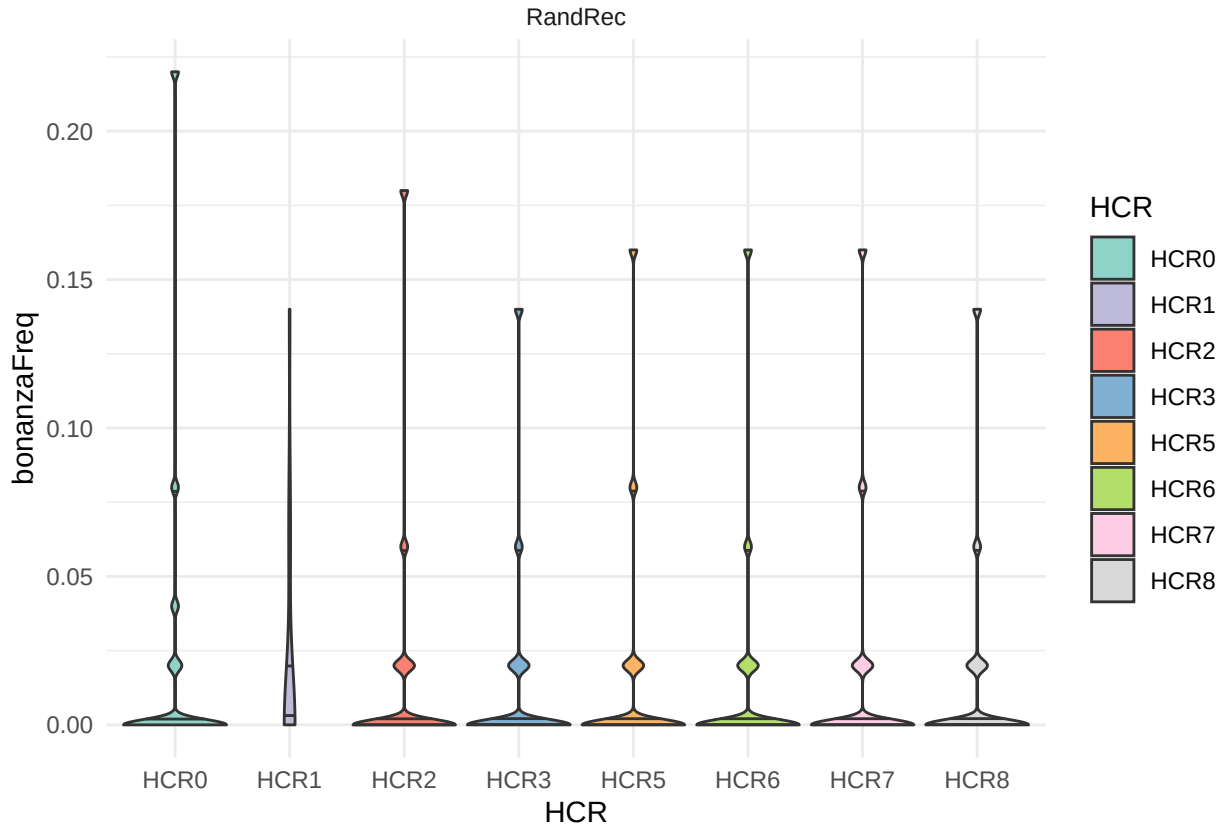

```
# plot collapse severity
metricsTbl %>% ggplot(aes(x = HCR, y = meanCollapseSever)) +
  geom_violin(aes(fill = HCR), draw_quantiles = c(0.1, 0.5, 0.9)) +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal)
```



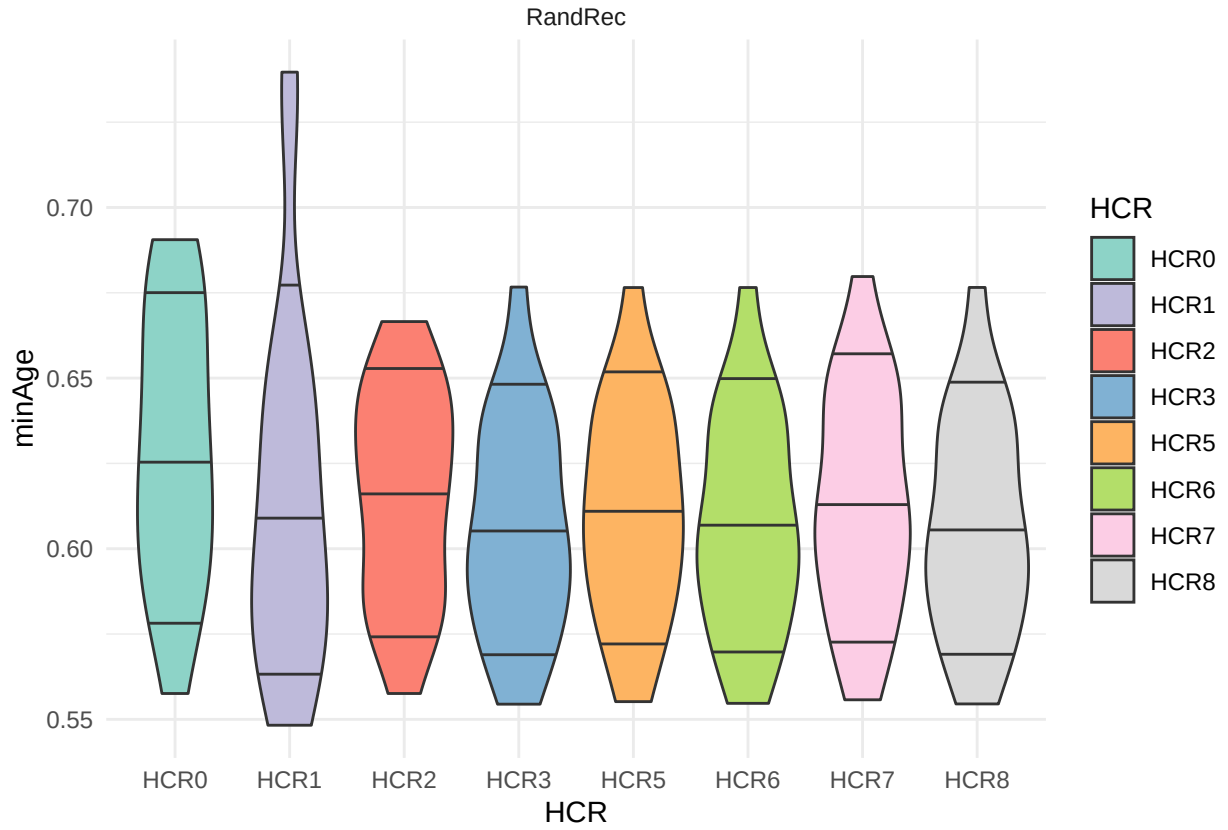

```
# plot bonanza frequency
metricsTbl %>% ggplot(aes(x = HCR, y = bonanzaFreq)) +
  geom_violin(aes(fill = HCR), draw_quantiles = c(0.1, 0.5, 0.9)) +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
```

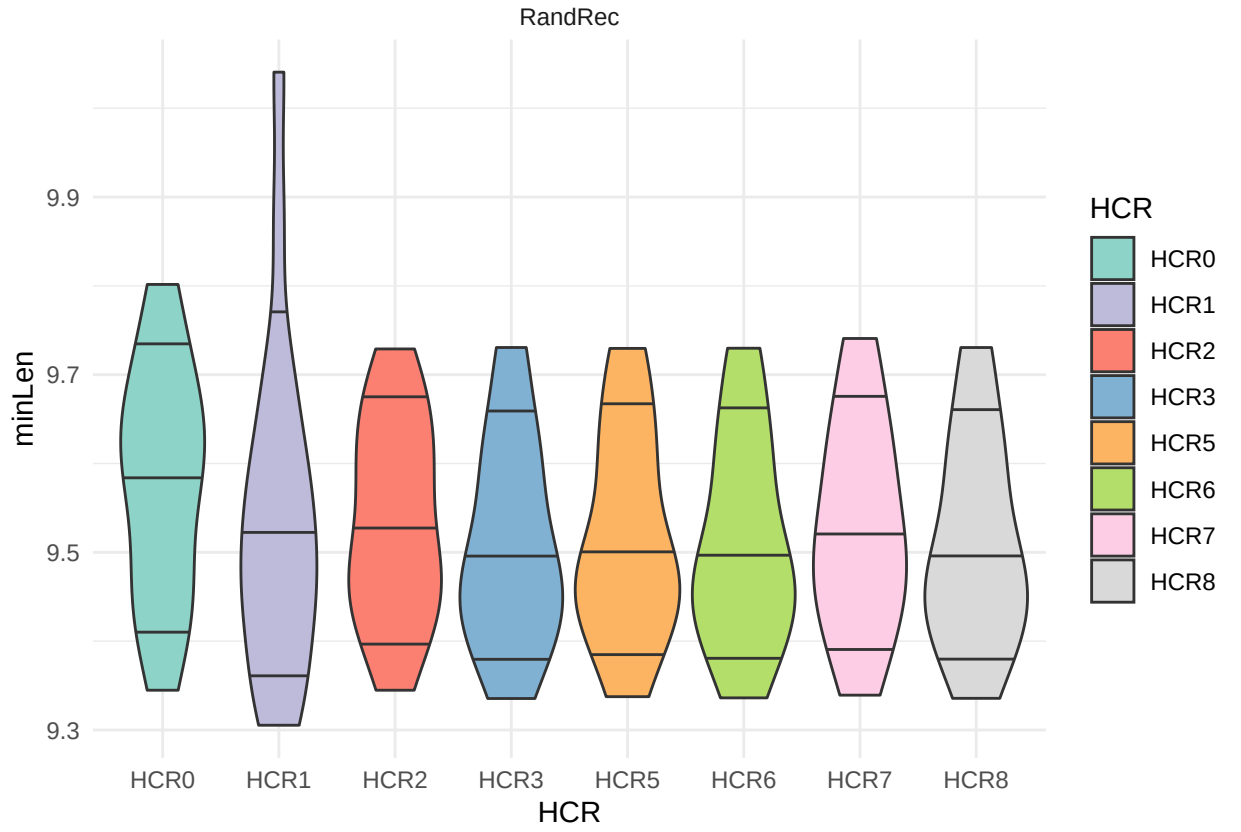
```
## collapsing to unique 'x' values
```



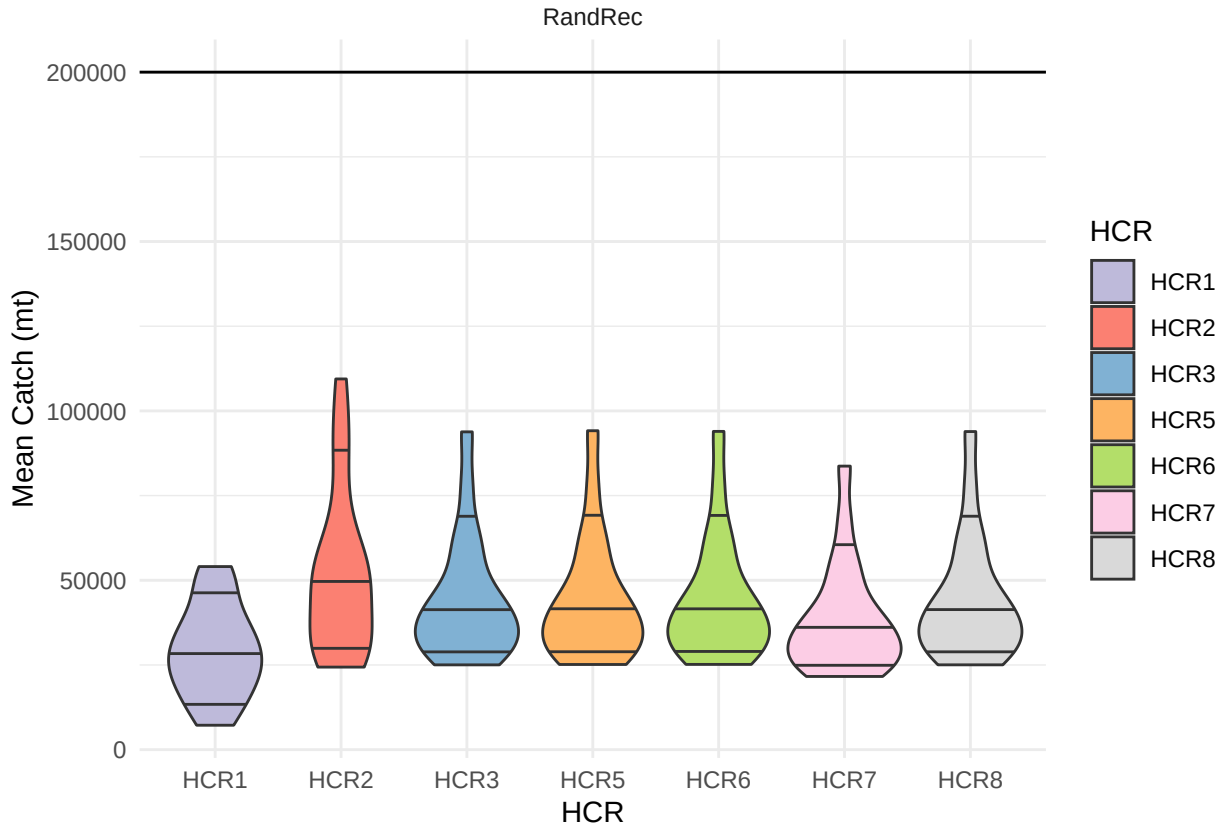

```r
# plot mean minimum age
metricsTbl %>% ggplot(aes(x = HCR, y = minAge)) +
  geom_violin(aes(fill = HCR), draw_quantiles = c(0.1, 0.5, 0.9)) +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal)
```



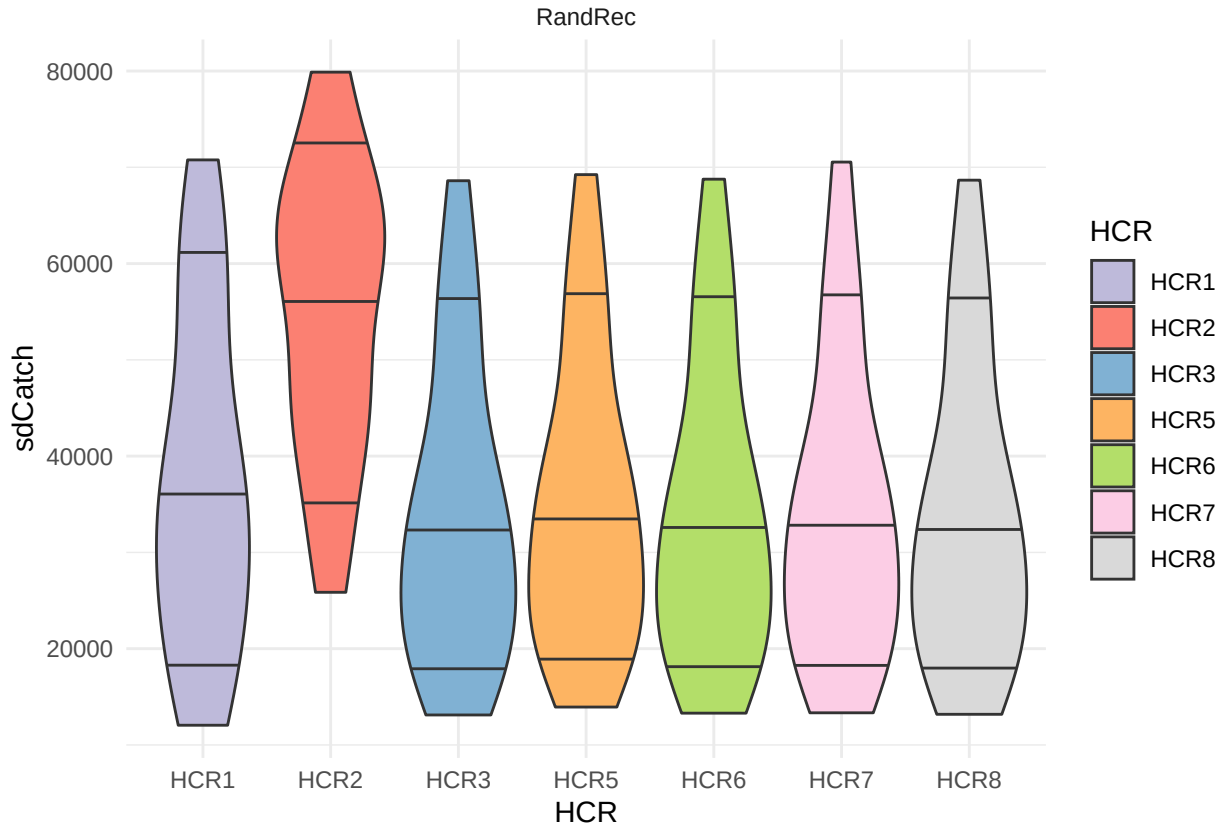

```
# plot mean minimum length
metricsTbl %>% ggplot(aes(x = HCR, y = minLen)) +
  geom_violin(aes(fill = HCR), draw_quantiles = c(0.1, 0.5, 0.9)) +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal)
```




```
# plot mean catch
metricsTbl %>%  filter(HCR != "HCR0") %>%
  filter(recScen == "RandRec", HCR != "HCR4") %>%
  ggplot(aes(x = HCR, y = meanCatch)) +
  geom_hline(yintercept = 200000) +
  geom_violin(aes(fill = HCR), draw_quantiles = c(0.1, 0.5, 0.9)) +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal[-1]) +
  labs(y = "Mean Catch (mt)", x = "HCR")
```

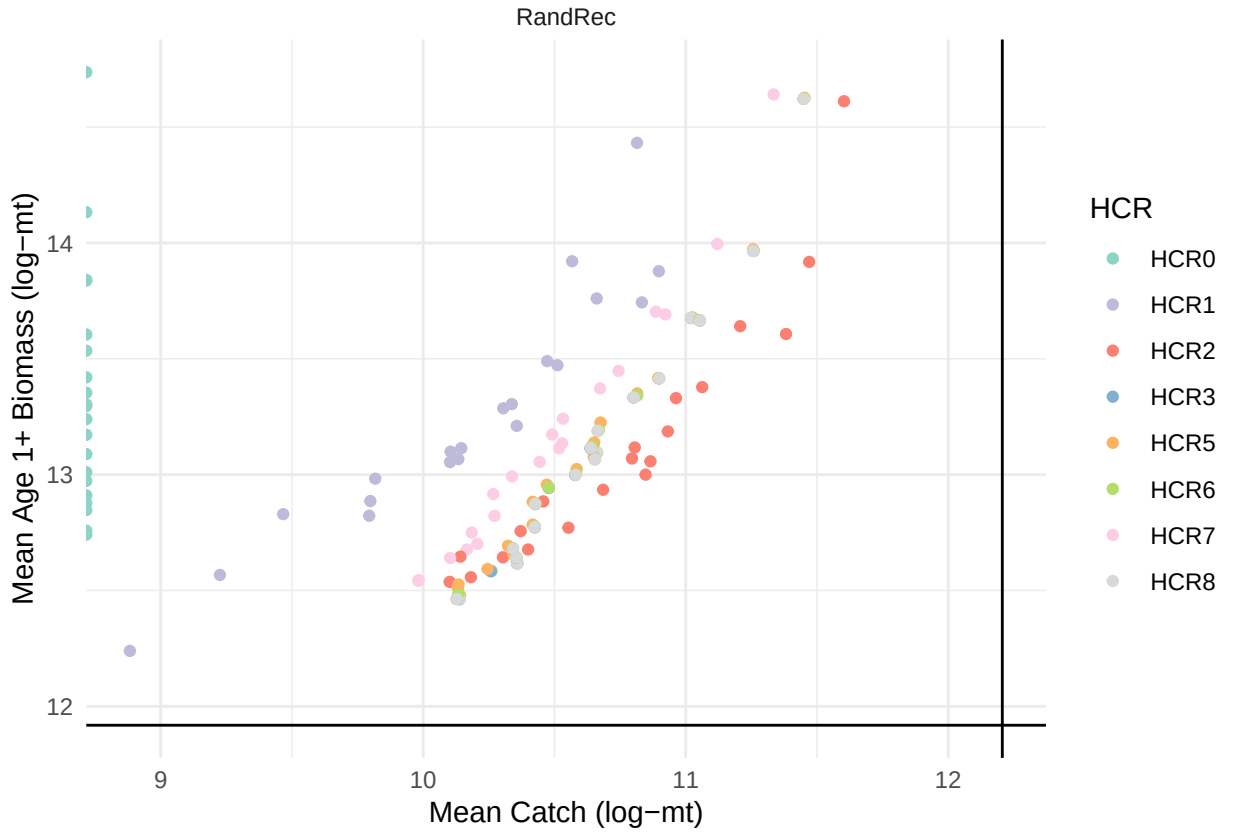



```
# plot catch sd
metricsTbl %>%  filter(HCR != "HCR0") %>%
  ggplot(aes(x = HCR, y = sdCatch)) +
  geom_violin(aes(fill = HCR), draw_quantiles = c(0.1, 0.5, 0.9)) +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal[-1])
```

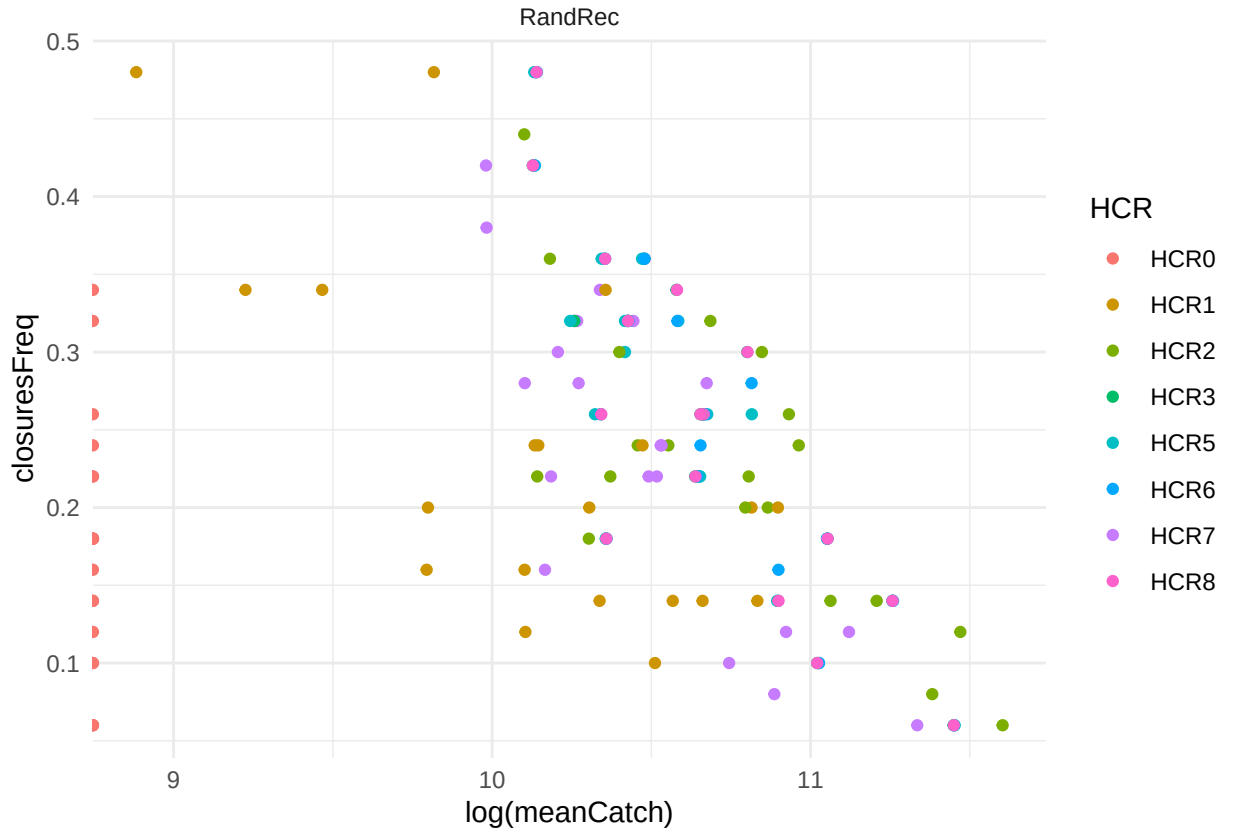


Tradeoffs Plot tradeoffs among metrics

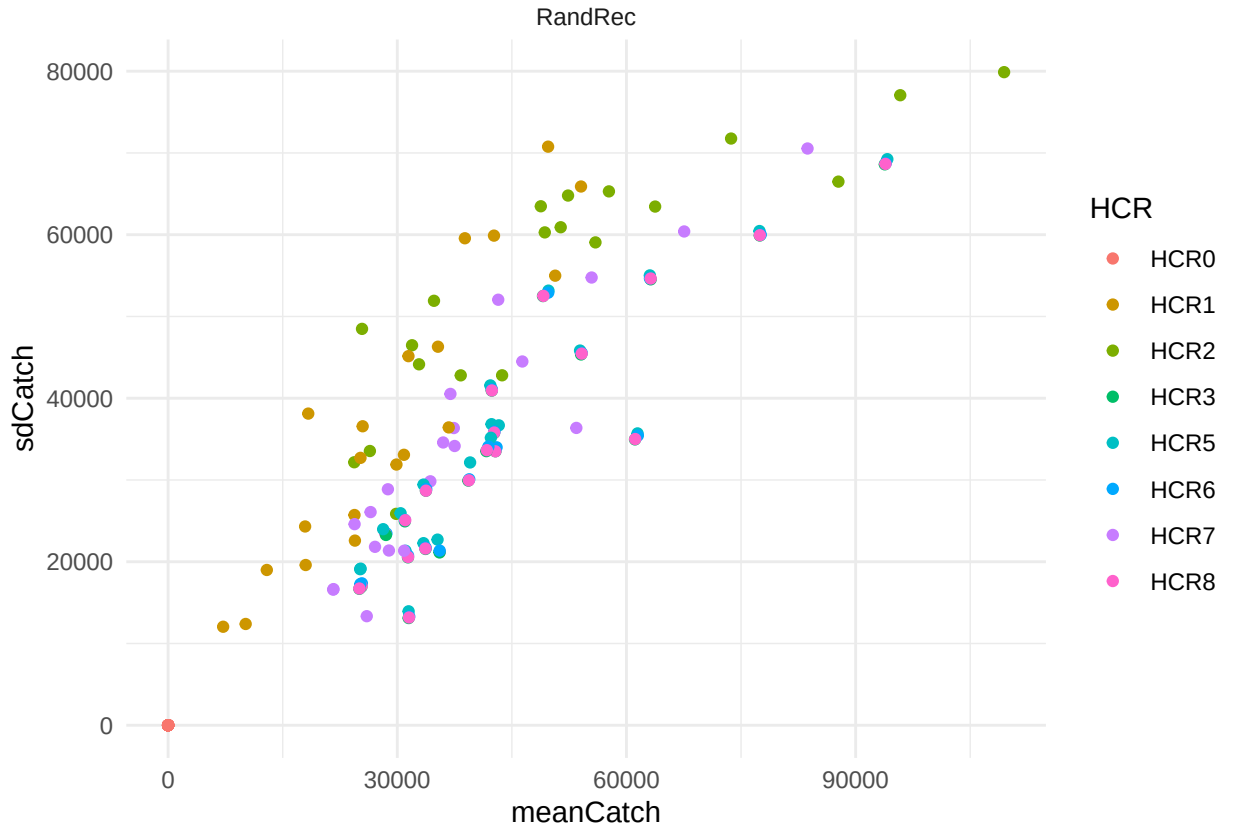
```
metricsTbl %>% filter(recScen == "RandRec", HCR != "HCR4") %>%
  ggplot(aes(x = log(meanCatch), y = log(meanB1plus), color = HCR)) +
  geom_vline(xintercept = log(200000)) +
  geom_hline(yintercept = log(150000)) +
  geom_point() +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_color_manual(values = hcrPal) +
  labs(y = "Mean Age 1+ Biomass (log-mt)", x = "Mean Catch (log-mt)")
```



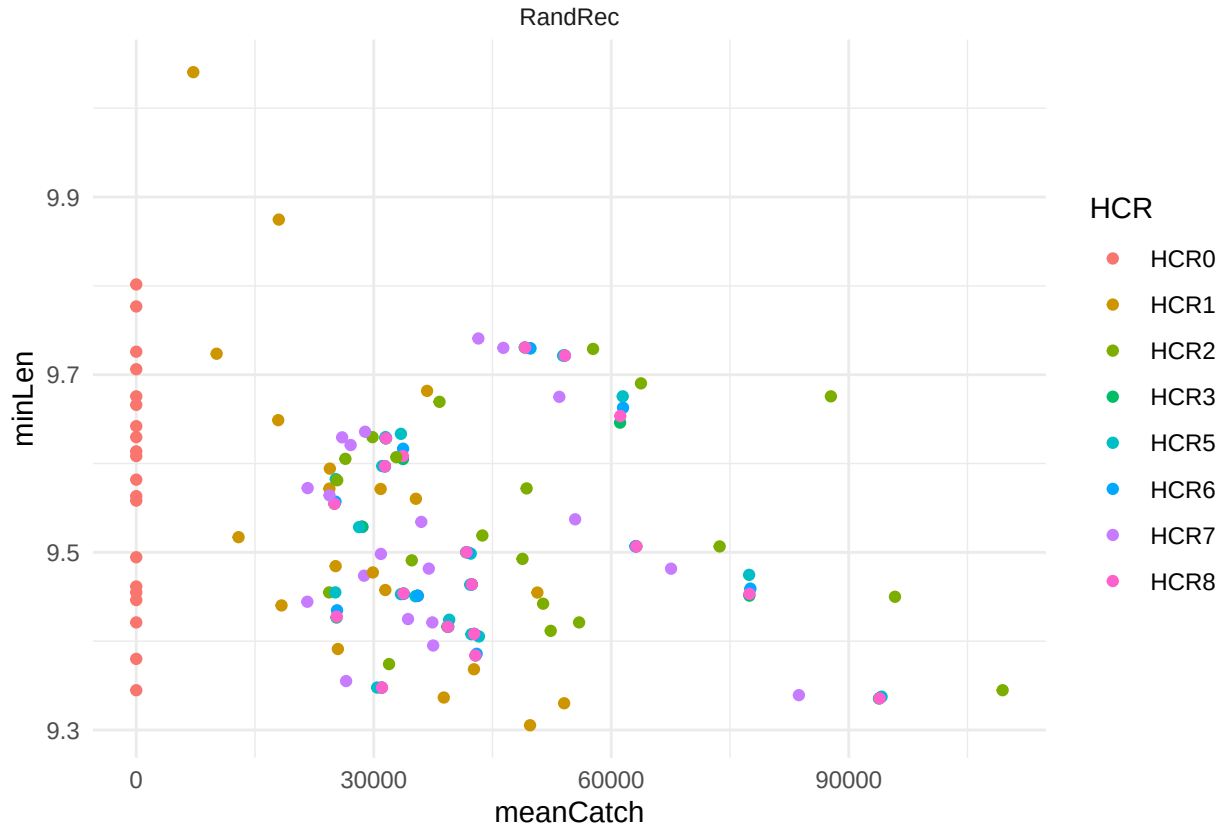

```
metricsTbl %>% ggplot(aes(x = log(meanCatch), y = closuresFreq, color = HCR)) +
  geom_point() +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal)
```




```
metricsTbl %>% ggplot(aes(x = meanCatch, y = sdCatch, color = HCR)) +
  geom_point() +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal)
```




```
metricsTbl %>% ggplot(aes(x = meanCatch, y = minLen, color = HCR)) +
  geom_point() +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal)
```



Timeseries

Plot timeseries of age1+ biomass, catch, and recruitment

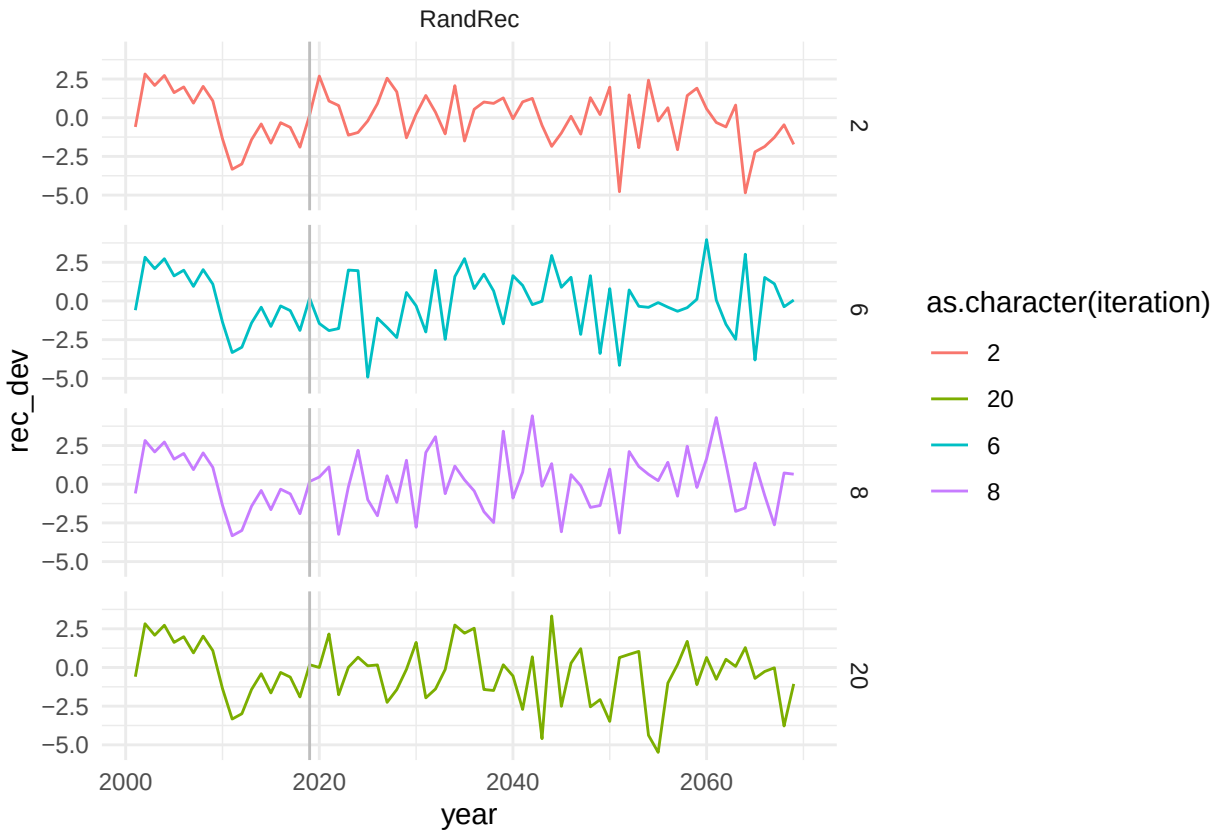
```
# get terminal estimates of these values for timeseries plots
termTS <- CalcTermTS(smryOutputList) %>%
             mutate(HCR = sub(pattern = ".*Rec","", scenario),
                              recScen = sub(pattern = "HCR.*","", scenario)) %>%
             mutate(recScen = sub(pattern = ".*EM_","", recScen))
```

```
## `summarise()` has grouped output by 'year', 'model_run', 'iteration'. You can
## override using the `.groups` argument.
```

```
omName <- grep("_OM", smryOutputList$tsSmry$model_run,
                fixed = TRUE, value = TRUE)[1]

# look at recruitment deviations
termTS %>% filter(model_run == omName,
                  HCR == "HCR0", iteration %in% sample(iteration, size = 4)) %>%
  ggplot(aes(x = year, y = rec_dev)) +
  geom_line(aes(linetype = as.character(iteration), color = as.character(iteration))) +
  scale_linetype_manual(values = rep("solid", 50)) +
  guides(linetype = "none") +
```

```
  facet_grid(rows = vars(iteration), cols = vars(recScen)) +
  theme_minimal() +
  geom_vline(xintercept = 2019, color = "gray")
```

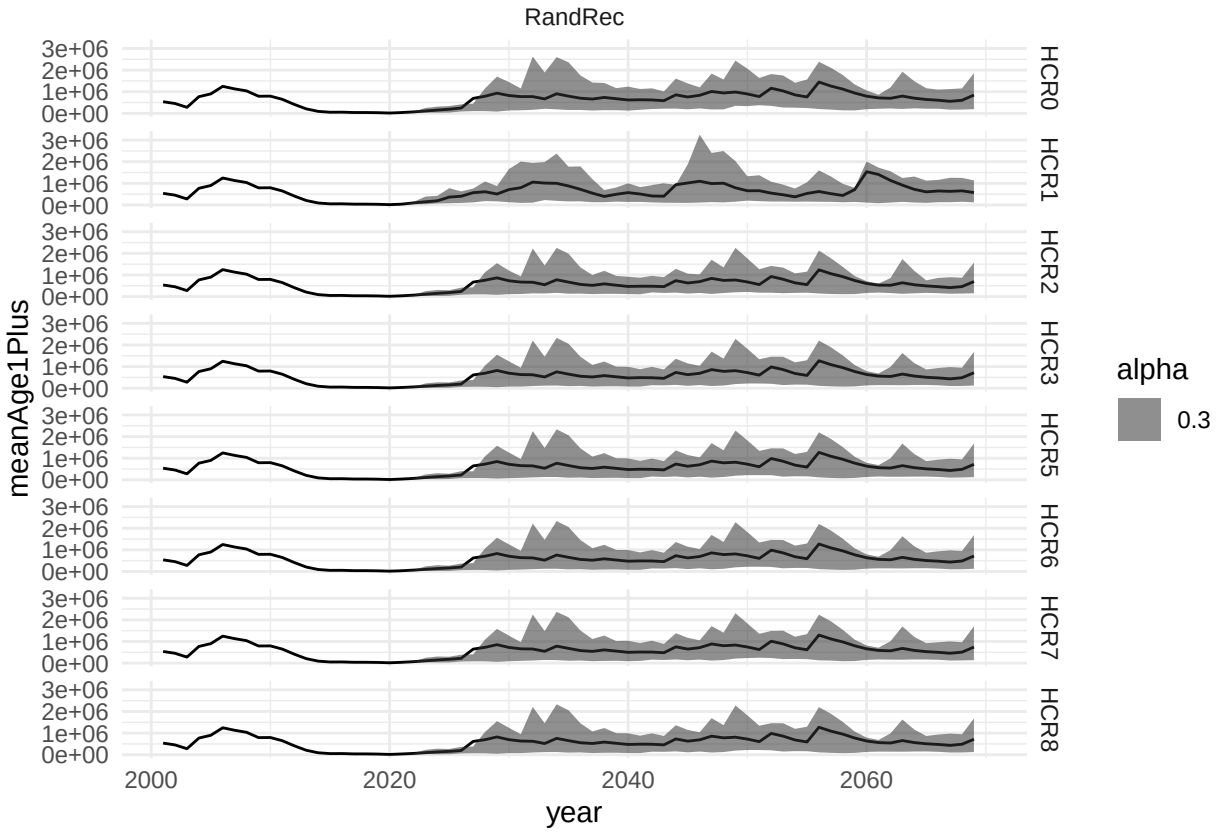
```
## Warning: Removed 4 row(s) containing missing values (geom_path).
```




```
# plot summary of biomass trajectories
termTS %>% filter(model_run == omName) %>%
  select(Bio_smry, year, model_run, iteration, scenario, HCR, recScen) %>%
  group_by(year, scenario, HCR, recScen) %>%
  summarize(meanAge1Plus = mean(Bio_smry),
            lowAge1Plus = quantile(Bio_smry, probs = 0.1, na.rm = TRUE),
            hiAge1Plus = quantile(Bio_smry, probs = 0.9, na.rm = TRUE)) %>%
  ggplot(aes(x = year, y = meanAge1Plus)) +
  geom_line() +
  geom_ribbon(aes(ymin = lowAge1Plus, ymax = hiAge1Plus, alpha = 0.3)) +
  facet_grid(rows = vars(HCR), cols = vars(recScen)) +
  theme_minimal()
```

```
## `summarise()` has grouped output by 'year', 'scenario', 'HCR'. You can override
## using the `.groups` argument.
```

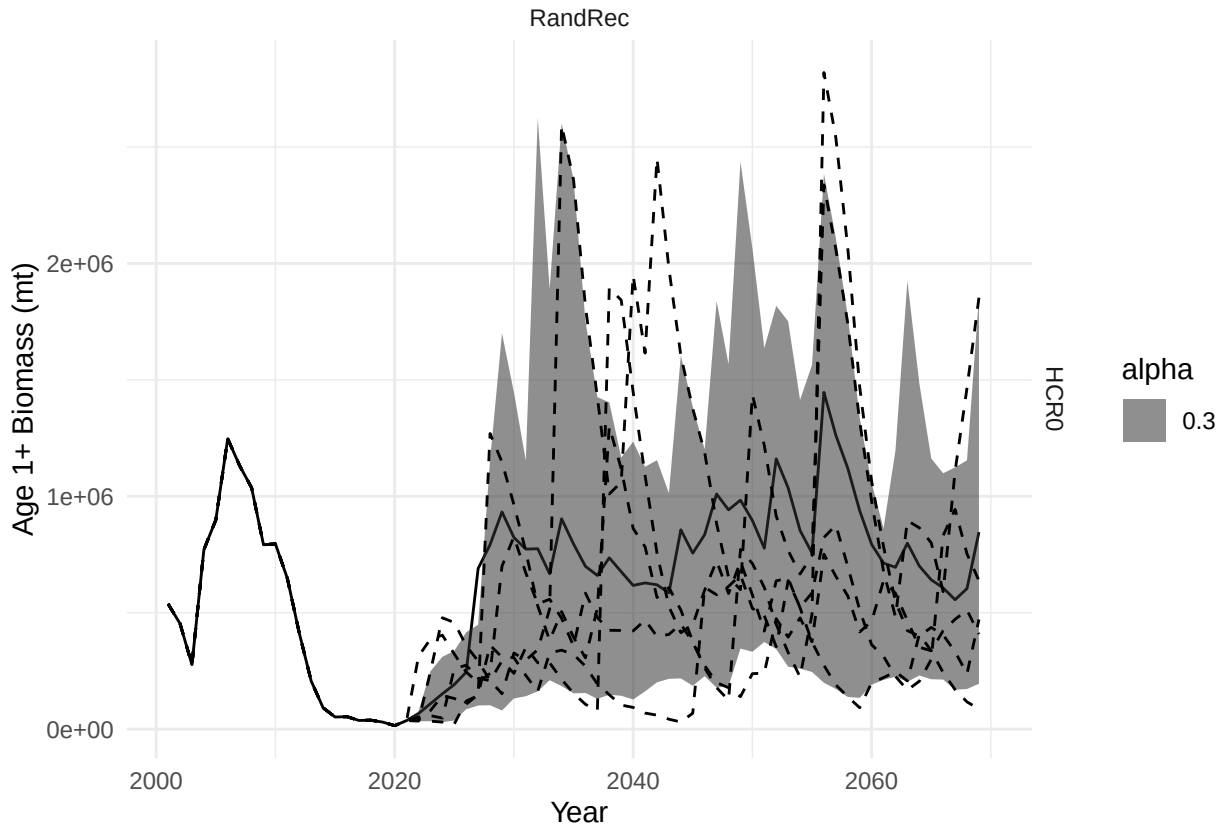
```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```




```
# timeseries plot for presentation
termTS %>% filter(model_run == omName, HCR == "HCR0") %>%
  select(Bio_smry, year, model_run, iteration, scenario, HCR, recScen) %>%
  group_by(year, scenario, HCR, recScen) %>%
  summarize(meanAge1Plus = mean(Bio_smry),
            lowAge1Plus = quantile(Bio_smry, probs = 0.1, na.rm = TRUE),
            hiAge1Plus = quantile(Bio_smry, probs = 0.9, na.rm = TRUE)) %>%
  ggplot(aes(x = year, y = meanAge1Plus)) +
  geom_line() +
  geom_ribbon(aes(ymin = lowAge1Plus, ymax = hiAge1Plus, alpha = 0.3)) +
  facet_grid(rows = vars(HCR), cols = vars(recScen)) +
  theme_minimal() +
  labs(x = "Year", y = "Age 1+ Biomass (mt)") +
  # add example trajectories
  geom_line(data = subset(termTS, model_run == omName &
                            HCR == "HCR0" &
                            iteration %in% sample(iteration, size = 5)),
            mapping = aes(x = year, y = Bio_smry, linetype = as.character(iteration))) +
  scale_linetype_manual(values = rep("dashed", 5)) +
  scale_color_manual(values = c("blue")) +
  guides(linetype = FALSE)
```

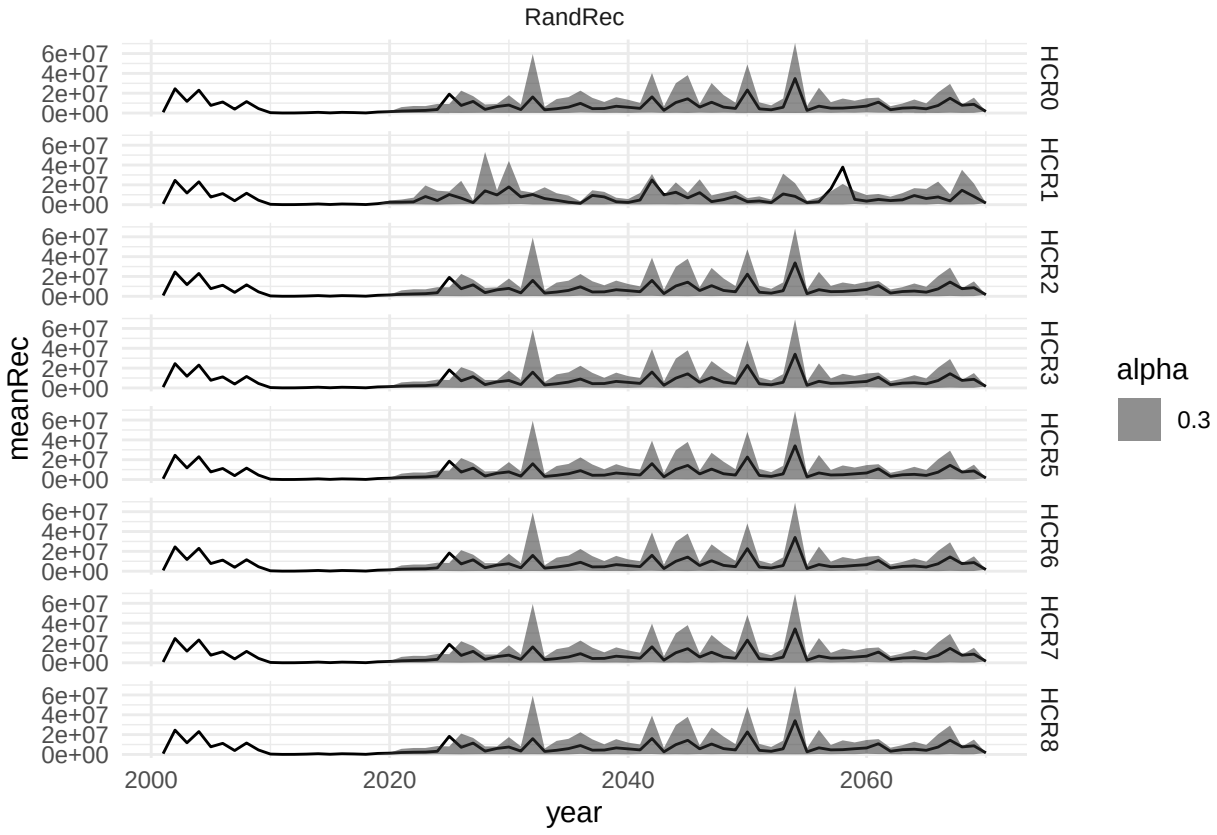
```
## `summarise()` has grouped output by 'year', 'scenario', 'HCR'. You can override
## using the `.groups` argument.

## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
```

```
## "none")‘ instead.

## Warning: Removed 1 row(s) containing missing values (geom_path).

## Warning: Removed 5 row(s) containing missing values (geom_path).
```




```
# plot summary of recruitment trajectories
termTS %>% filter(model_run == omName) %>%
  select(Value.Recr, year, model_run, iteration, scenario, HCR, recScen) %>%
  group_by(year, scenario, HCR, recScen) %>%
  summarize(meanRec = mean(Value.Recr),
            lowRec = quantile(Value.Recr, probs = 0.1, na.rm = TRUE),
            hiRec = quantile(Value.Recr, probs = 0.9, na.rm = TRUE)) %>%
  ggplot(aes(x = year, y = meanRec)) +
  geom_line() +
  geom_ribbon(aes(ymin = lowRec, ymax = hiRec, alpha = 0.3)) +
  facet_grid(rows = vars(HCR), cols = vars(recScen)) +
  theme_minimal()
```

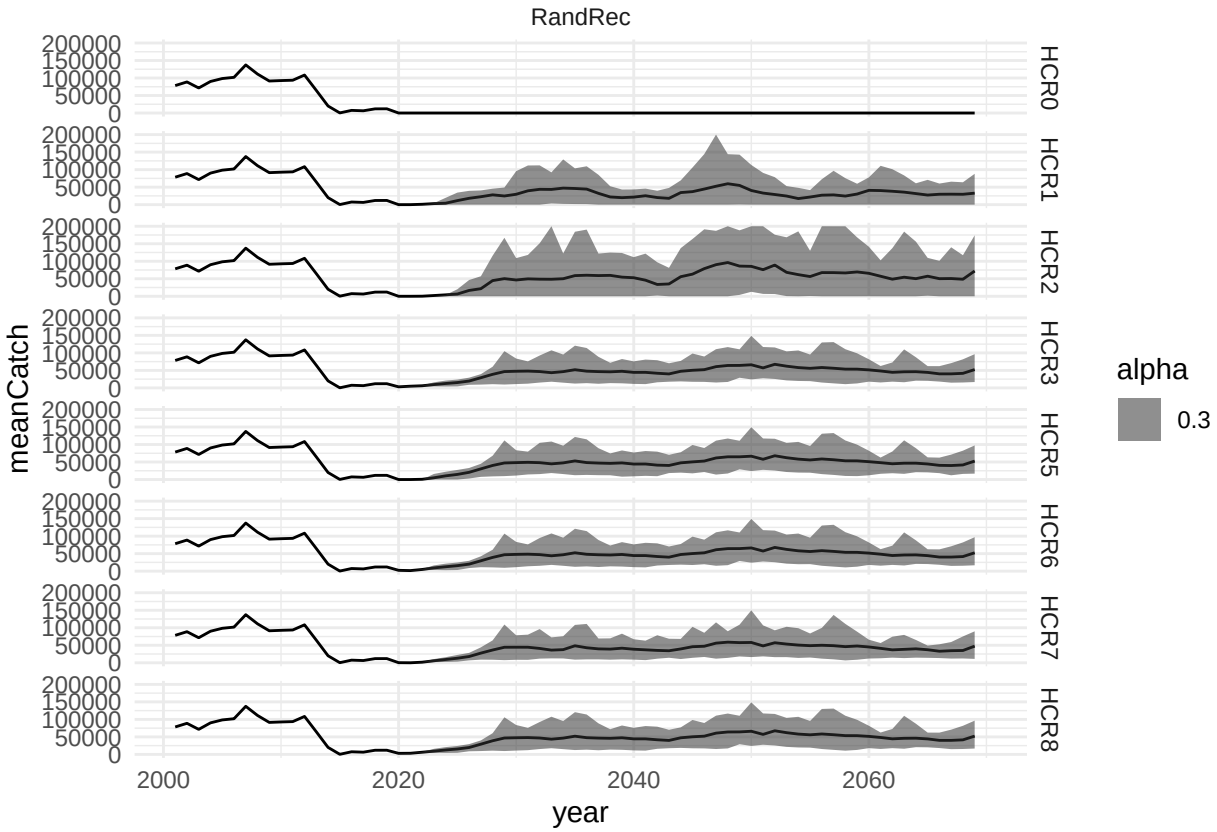
```
## ‘summarise()‘ has grouped output by ’year’, ’scenario’, ’HCR’. You can override
## using the ‘.groups‘ argument.
```




```
# plot summary of catch trajectories
termTS %>% filter(model_run == omName) %>%
  select(totCatch, year, model_run, iteration, scenario, HCR, recScen) %>%
  group_by(year, scenario, HCR, recScen) %>%
  summarize(meanCatch = mean(totCatch),
            lowCatch = quantile(totCatch, probs = 0.1, na.rm = TRUE),
            hiCatch = quantile(totCatch, probs = 0.9, na.rm = TRUE)) %>%
  ggplot(aes(x = year, y = meanCatch)) +
  geom_line() +
  geom_ribbon(aes(ymin = lowCatch, ymax = hiCatch, alpha = 0.3)) +
  facet_grid(rows = vars(HCR), cols = vars(recScen)) +
  theme_minimal()
```

```
## `summarise()` has grouped output by 'year', 'scenario', 'HCR'. You can override
## using the `.groups` argument.
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

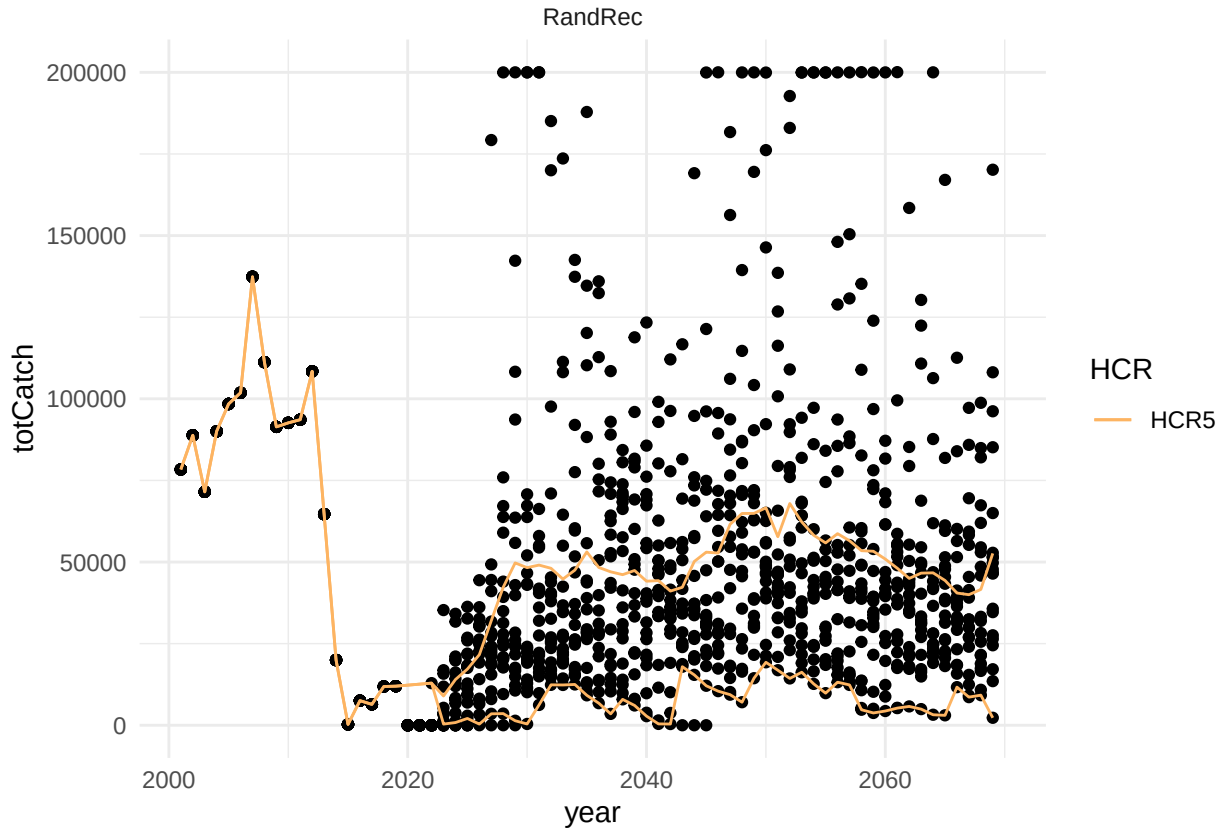



```
# plot catch levels for HCR5 (Pikitch rule)
meanCat <- termTS %>% filter(model_run == omName, HCR == "HCR5") %>%
  select(totCatch, year, model_run, iteration, scenario, HCR, recScen) %>%
  # try non-zero catch summary
  filter(totCatch > 0) %>%
  group_by(year, scenario, HCR, recScen) %>%
  summarize(meanCatch = mean(totCatch),
            # lowCatch = quantile(totCatch, probs = 0.1, na.rm = TRUE),
            hiCatch = quantile(totCatch, probs = 0.9, na.rm = TRUE),
            minCatch = min(totCatch))
```

```
## `summarise()` has grouped output by 'year', 'scenario', 'HCR'. You can override
## using the `.groups` argument.
```

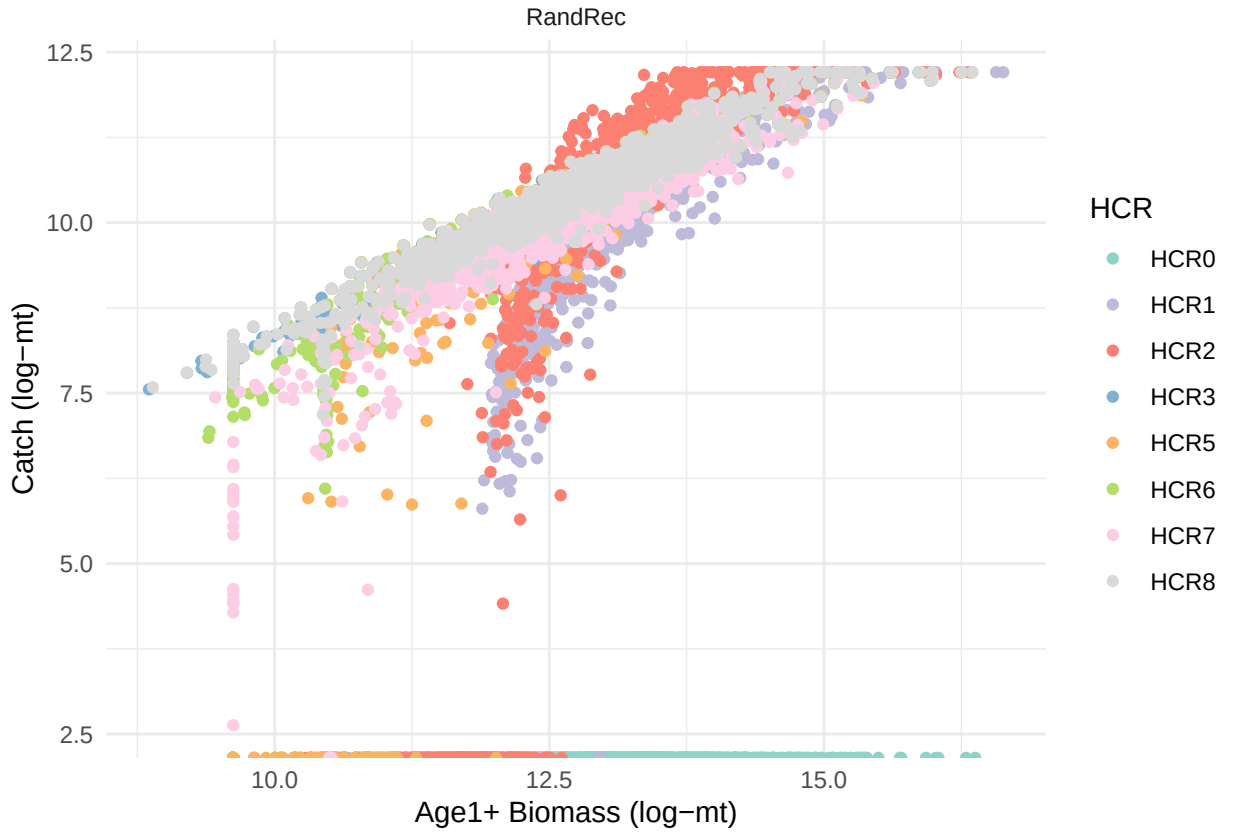
```
termTS %>% filter(model_run == omName, HCR == "HCR5") %>%
  ggplot(aes(x = year, y = totCatch)) +
  geom_point() +
  geom_line(data = meanCat, aes(y = meanCatch, color = HCR)) +
  geom_line(data = meanCat, aes(y = minCatch, color = HCR)) +
  facet_wrap(~recScen) +
  scale_color_manual(values = hcrPal[5]) +
  theme_minimal()
```

```
## Warning: Removed 20 rows containing missing values (geom_point).
```



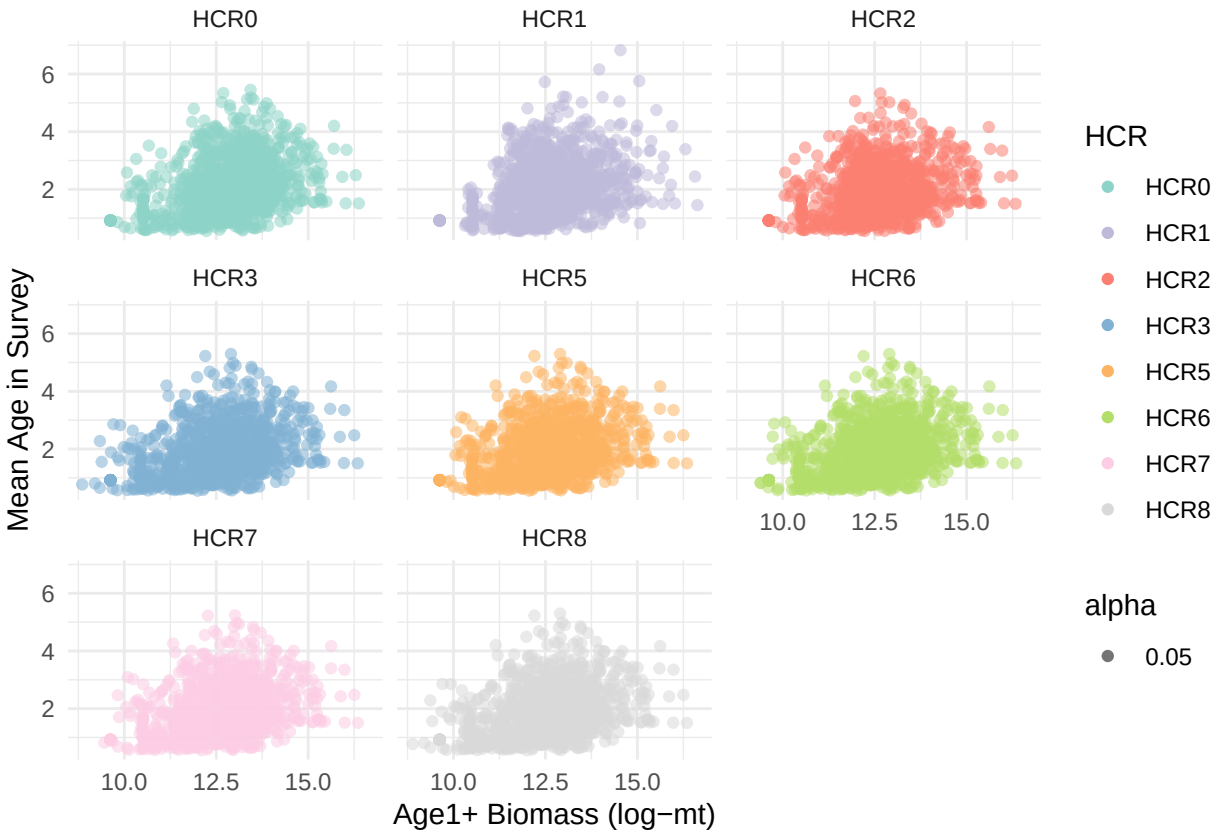

```
termTS %>% filter(model_run == omName, year > 2019) %>%
  ggplot(aes(x = log(Bio_smry), y = log(totCatch), color = HCR)) +
  geom_point() +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_color_manual(values = hcrPal) +
  labs(y = "Catch (log-mt)", x = "Age1+ Biomass (log-mt)")
```

```
## Warning: Removed 160 rows containing missing values (geom_point).
```




```
# compare to mean length
  # find columns for directory parsing
  dirsCol <- ncol(str_split(smryOutputList$ageComp$resDir,
                            pattern = "/", simplify = TRUE))

fleet4Age <- smryOutputList$ageComp %>%
                # need to parse out iteration, model run and scenario
                mutate(model_run = str_split(resDir, pattern = "/",
                                             simplify = TRUE)[, dirsCol],
                       iteration = as.integer(str_split(resDir, pattern = "/",
                                             simplify = TRUE)[, dirsCol-1]),
                       scenario = str_split(resDir, pattern = "/",
                                            simplify = TRUE)[, dirsCol-2]) %>%
                filter(grepl(omName, model_run, fixed = TRUE), Yr > 2019,
                       Seas == 1, Fleet == 4)

termTS %>% filter(model_run == omName, year > 2019,
                  recScen == "RandRec") %>%
  left_join(y = fleet4Age, by = c("model_run", "iteration", "scenario", "year" = "Yr")) %>%
  ggplot(aes(x = log(Bio_smry), y = All_exp_mean, color = HCR, alpha = 0.05)) +
  geom_point() +
  facet_wrap(~HCR) +
  theme_minimal() +
  scale_color_manual(values = hcrPal) +
  labs(y = "Mean Age in Survey", x = "Age1+ Biomass (log-mt)")
```

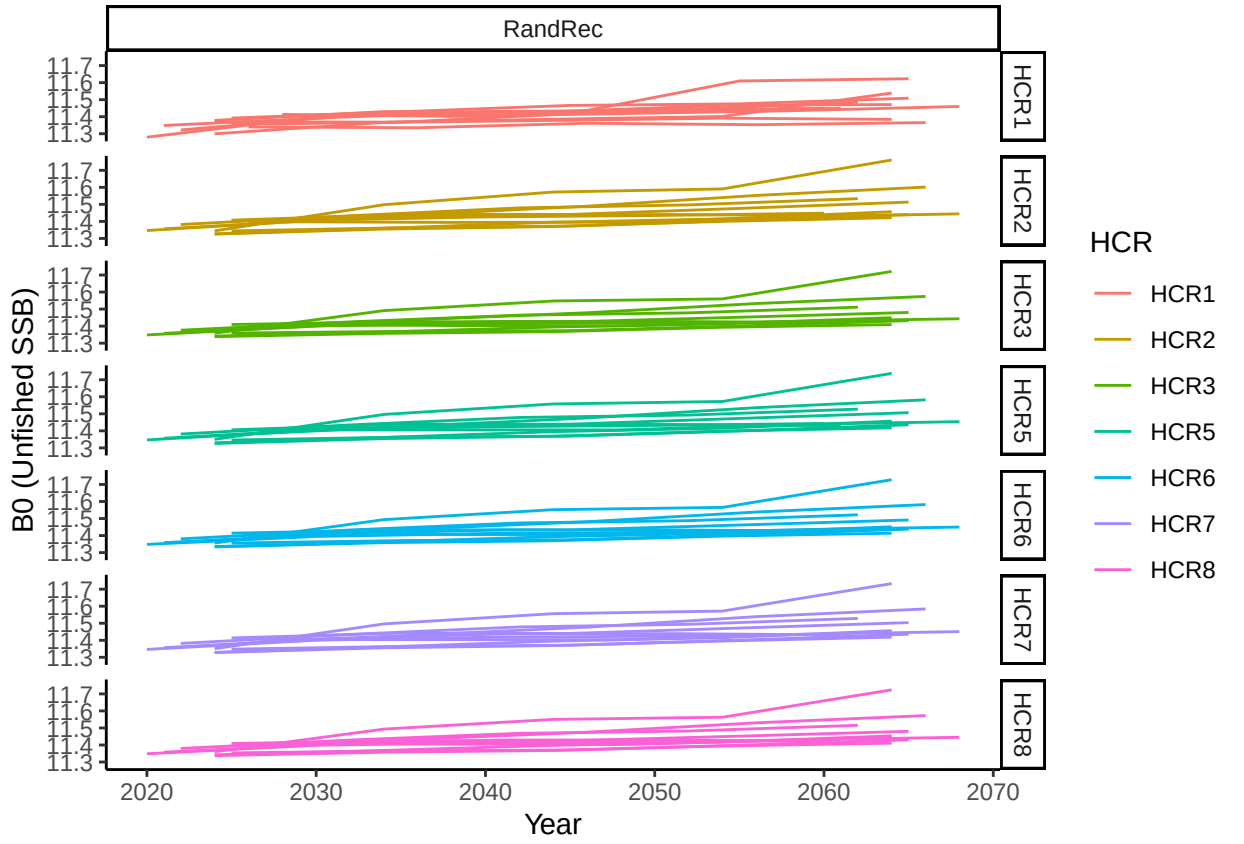
```
## Warning: Removed 160 rows containing missing values (geom_point).
```



Convergence and Estimation Error

```
convrgCheck <- smryOutputList$sclSmry %>%
                  select(max_grad, model_run, iteration, scenario) %>%
                  mutate(emYear = as.numeric(regmatches(model_run,
                                                        gregexpr("[[:digit:]]+",
                                                                 model_run))),
                         HCR = sub(pattern = ".*Rec","", scenario),
                         recScen = sub(pattern = "HCR.*","", scenario)) %>%
                  mutate(recScen = sub(pattern = ".*EM_","", recScen))

# Look at timeseries of B0 and account for non-convergence
smryOutputList$sclSmry %>% mutate(emYear = as.numeric(regmatches(model_run,
                                                        gregexpr("[[:digit:]]+",
                                                                 model_run))),
                         HCR = sub(pattern = ".*Rec","", scenario),
                         recScen = sub(pattern = "HCR.*","", scenario)) %>%
  mutate(recScen = sub(pattern = ".*EM_","", recScen))  %>%
  filter(model_run != omName, HCR != "HCR0", max_grad < 0.01,
         iteration == sample(1:20, size = 10)) %>%
  ggplot(aes(x = emYear, y = log(SSB_Unfished))) +
```

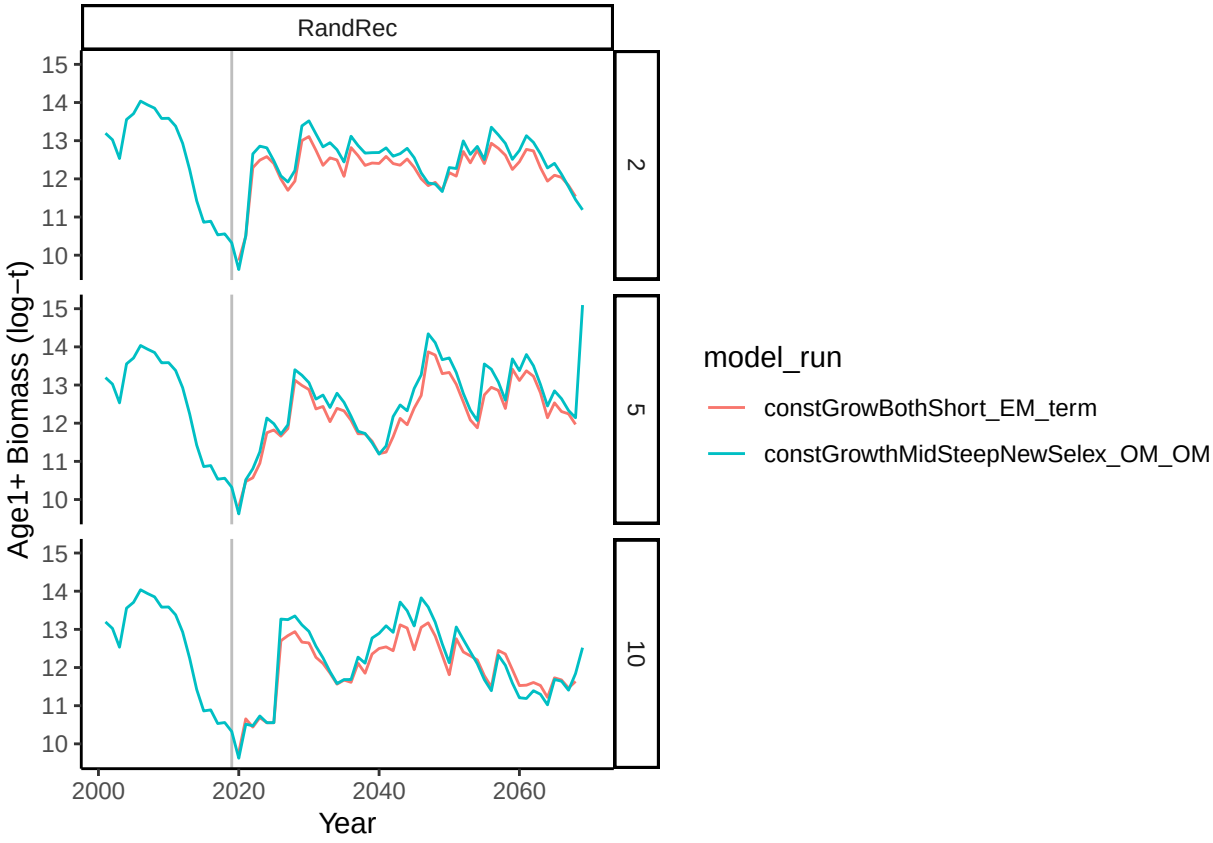
```
  geom_line(ggplot2::aes(linetype = as.character(iteration), color = HCR))+
    #ggplot2::scale_color_manual(values = c("#D65F00", "blue")) +
    ggplot2::scale_linetype_manual(values = rep("solid", 100)) +
  ggplot2::guides(linetype = "none") +
    ggplot2::facet_grid(rows = vars(HCR), cols = vars(recScen)) +
    ggplot2::theme_classic() +
  #geom_rug(data = convrgCheck, mapping = aes(x = emYear),
  #                      sides = "b", inherit.aes = FALSE) +
  labs(x = "Year", y = "B0 (Unfished SSB)")
```




```
set.seed(seed = 413)
exIters <- sample(termTS$iteration, size = 3)
exHCR <- "HCR2"
convrgCheckEx <- convrgCheck %>% filter(iteration %in% exIters, HCR == exHCR)%>%
                  filter(max_grad > 0.01)

termTS %>% filter(iteration %in% exIters,
                  HCR == exHCR,
                  model_run != "constGrowBothShort_EM_init") %>%
  ggplot2::ggplot(aes(x = year, y = log(Bio_smry))) +
    ggplot2::geom_vline(xintercept = 2019, color = "gray") +

    ggplot2::geom_line(ggplot2::aes(linetype = as.character(iteration), color = model_run))+
    #ggplot2::scale_color_manual(values = c("#D65F00", "blue")) +
    ggplot2::scale_linetype_manual(values = rep("solid", 50)) +
```

```
    ggplot2::guides(linetype = "none") +
    ggplot2::facet_grid(rows = vars(iteration), cols = vars(recScen)) +
    ggplot2::theme_classic() +
  geom_rug(data = convrgCheckEx, mapping = aes(x = emYear),
                        sides = "b", inherit.aes = FALSE) +
  labs(x = "Year", y = "Age1+ Biomass (log-t)")
```

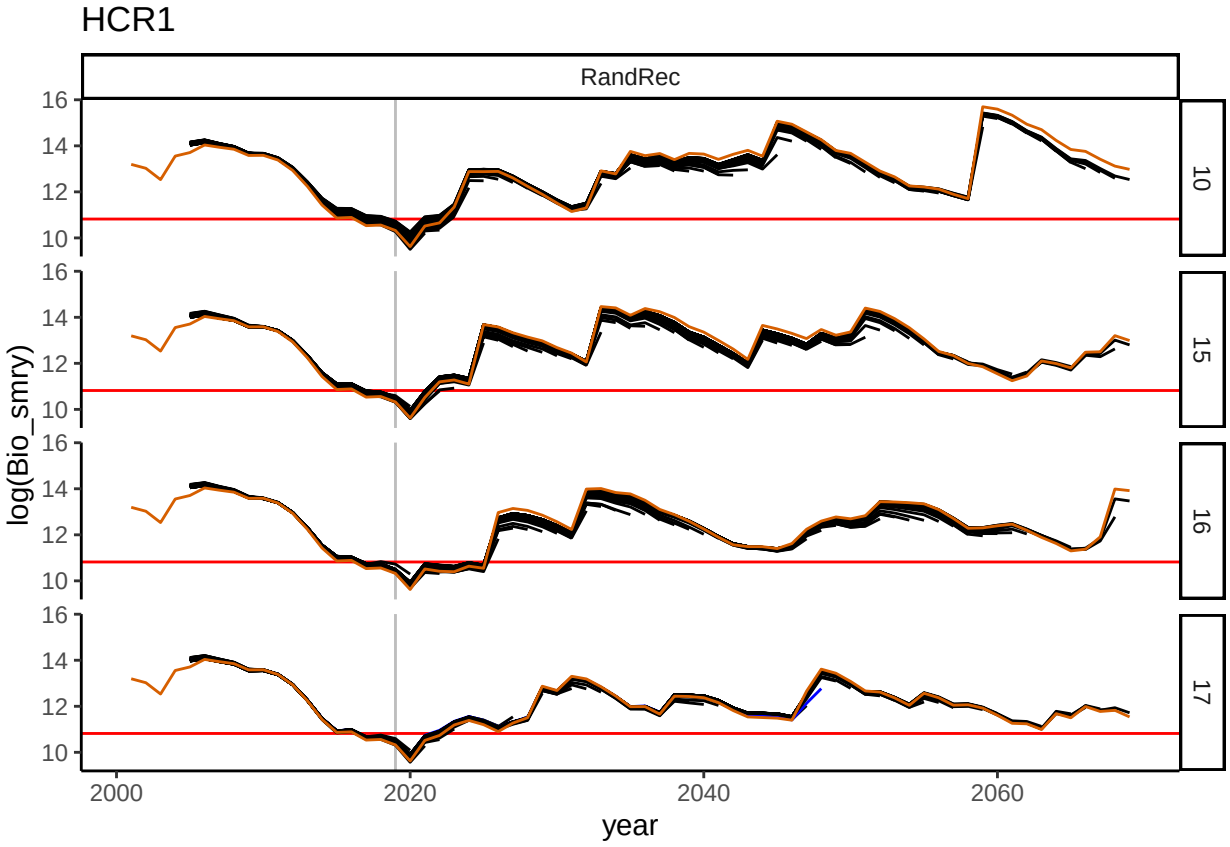
```
## Warning: Removed 3 row(s) containing missing values (geom_path).
```

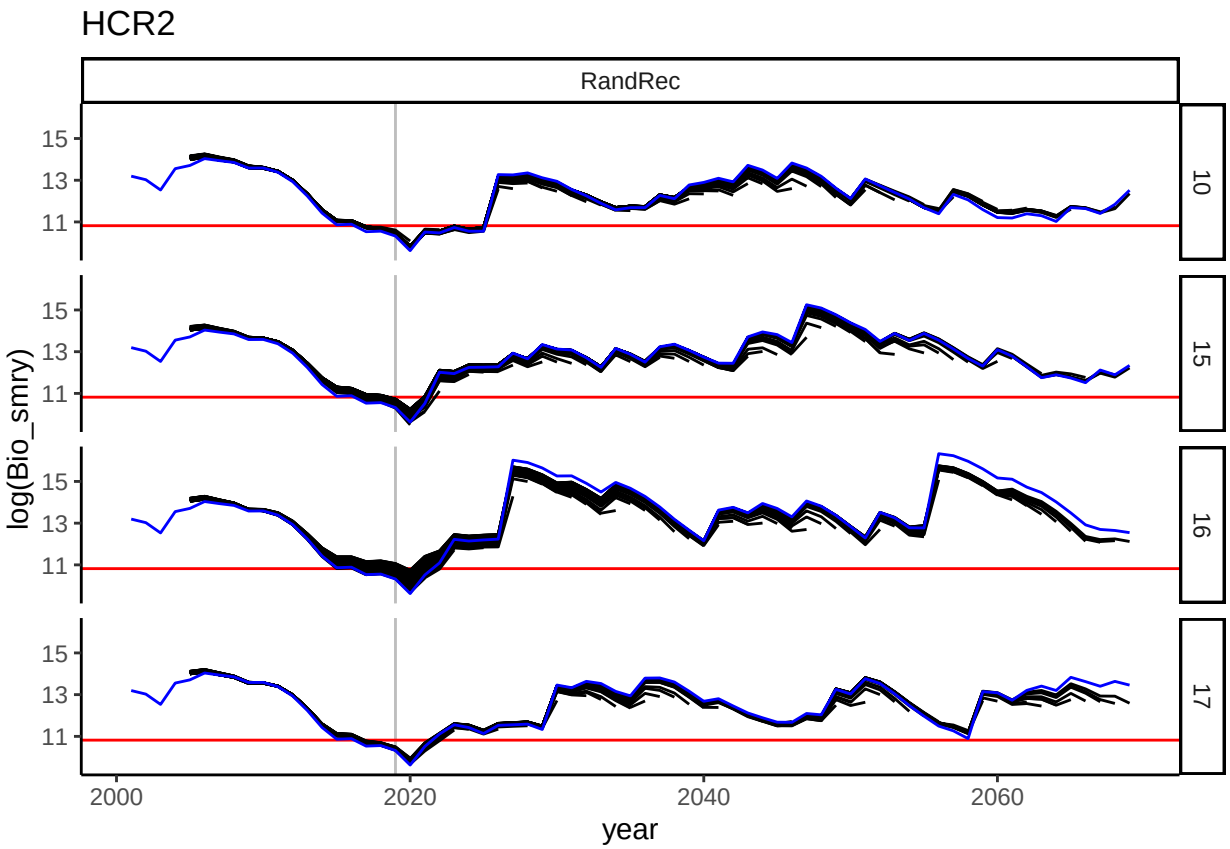


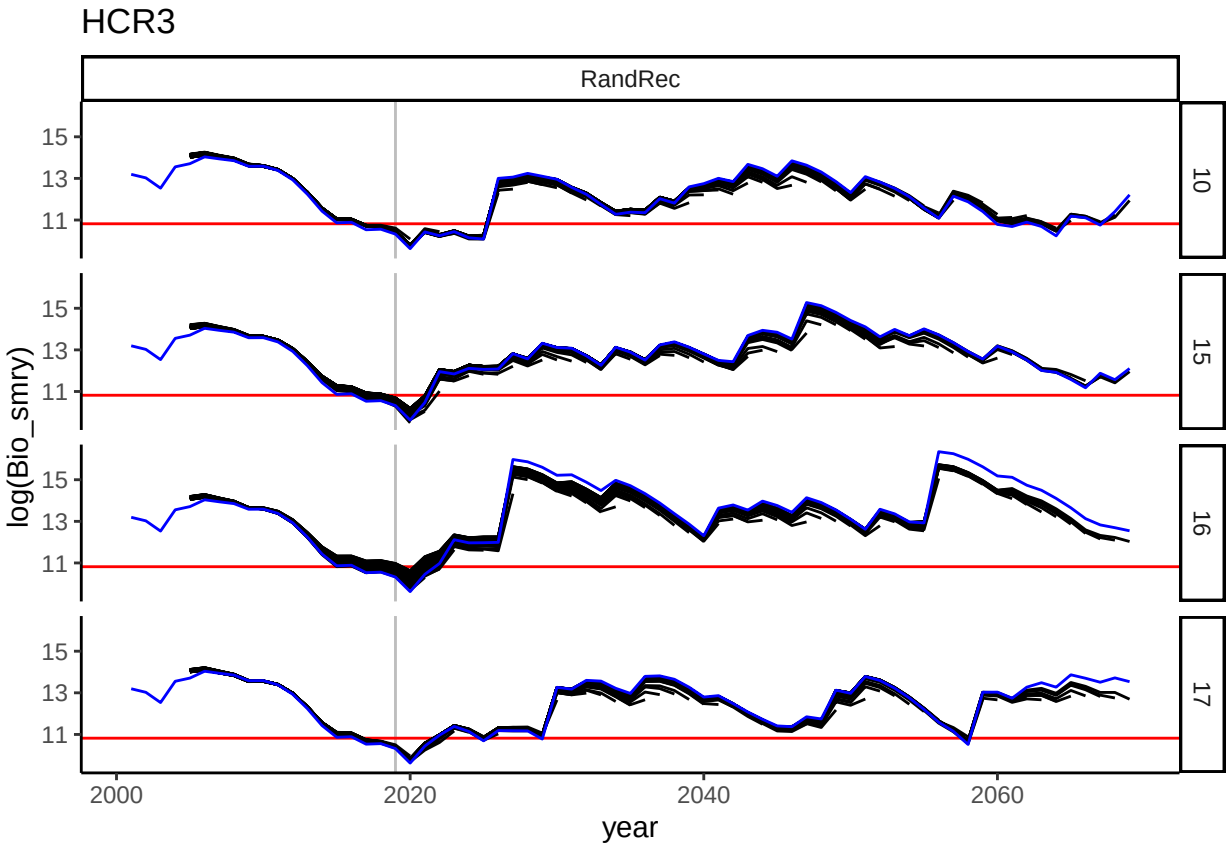
Plot some timeseries over estimation models for each run

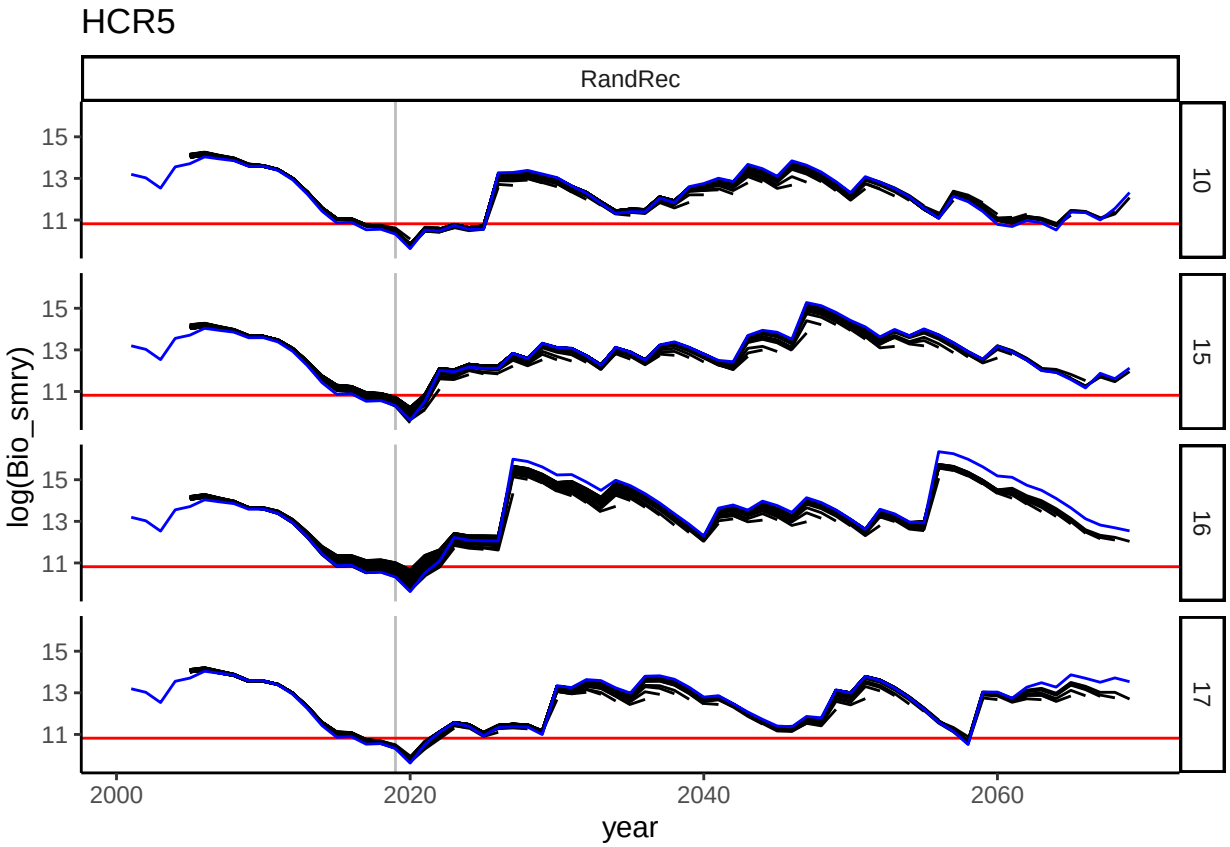
```
hcrs <- unique(termTS$HCR)
exIters <- sample(termTS$iteration, size = 4)

cnvrgTS <- smryOutputList$tsSmry %>% mutate(HCR = sub(pattern = ".*Rec","", scenario),
                                    recScen = sub(pattern = "HCR.*","", scenario)) %>%
      mutate(recScen = sub(pattern = ".*EM_","", recScen)) %>%
      left_join(y = convrgCheck, by = c("iteration", "model_run", "scenario", "HCR", "recScen")) %>%
      mutate(plotGroup = case_when(model_run == omName ~ "OM",
                                    max_grad > 0.01 ~ "non-convrg",
                                    max_grad < 0.01 ~ "convrg"))

for(hcr in 2:length(scenarios)){
  print(cnvrgTS %>% filter(HCR == hcrs[hcr], iteration %in% exIters, Seas == 1) %>%
      ggplot(aes(x = year, y = log(Bio_smry))) +
```

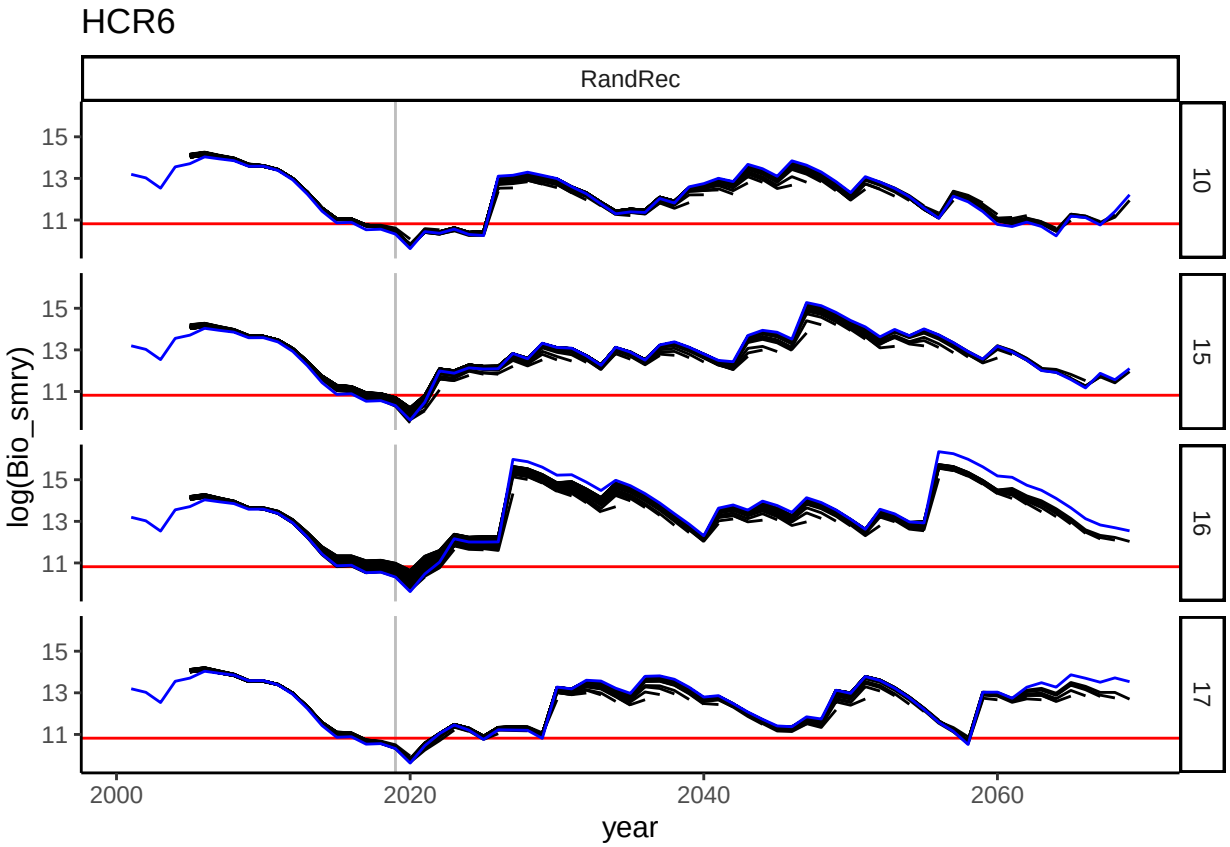
```
      ggplot2::geom_vline(xintercept = 2019, color = "gray") +
      ggplot2::geom_hline(yintercept = log(50000), color = "red") +
      ggplot2::geom_line(aes(linetype = model_run, color = plotGroup))+
      ggplot2::scale_color_manual(values = c("black", "blue", "#D65F00")) +
      ggplot2::scale_linetype_manual(values = rep("solid", 51)) +
      ggplot2::guides(linetype = "none") +
      ggplot2::facet_grid(rows = vars(iteration), cols = vars(recScen)) +
      ggplot2::theme_classic() + theme(legend.position="none") +
      labs(title = hcrs[hcr]))
}
```

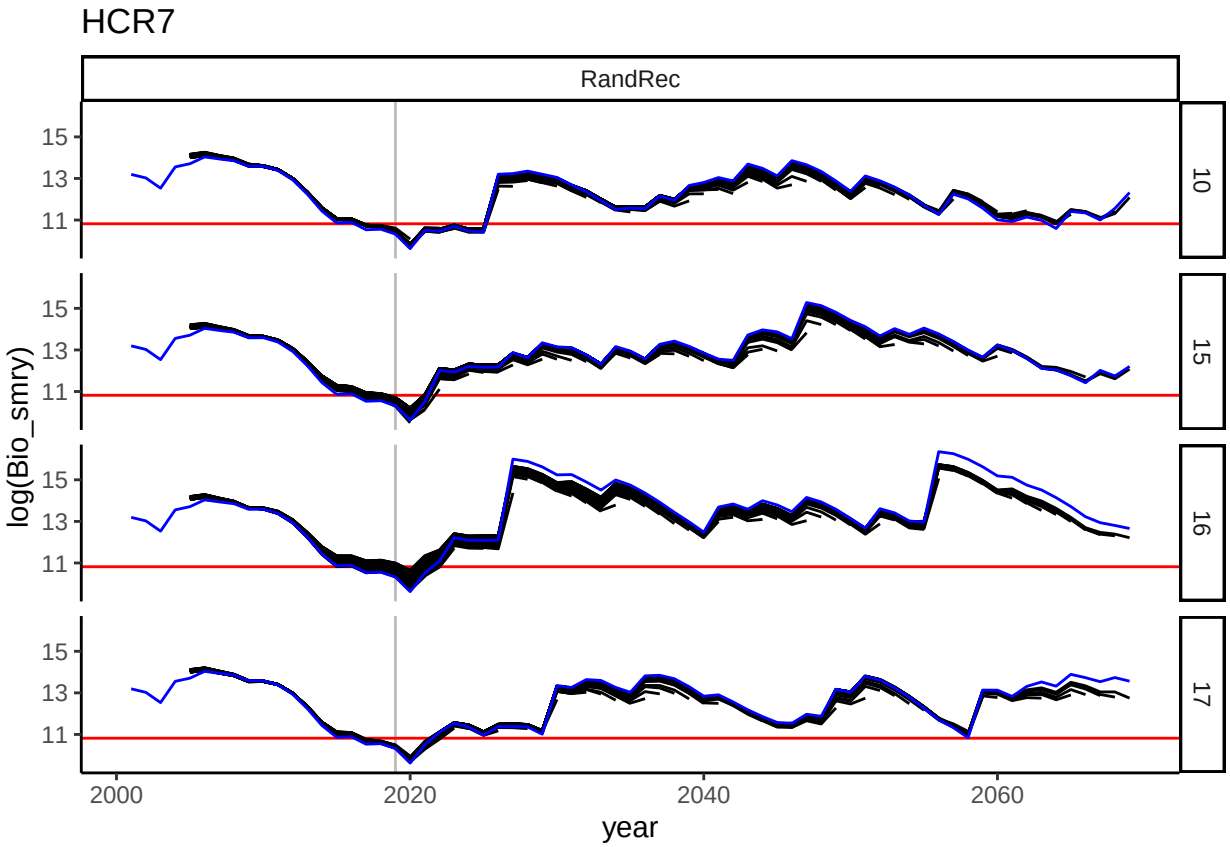


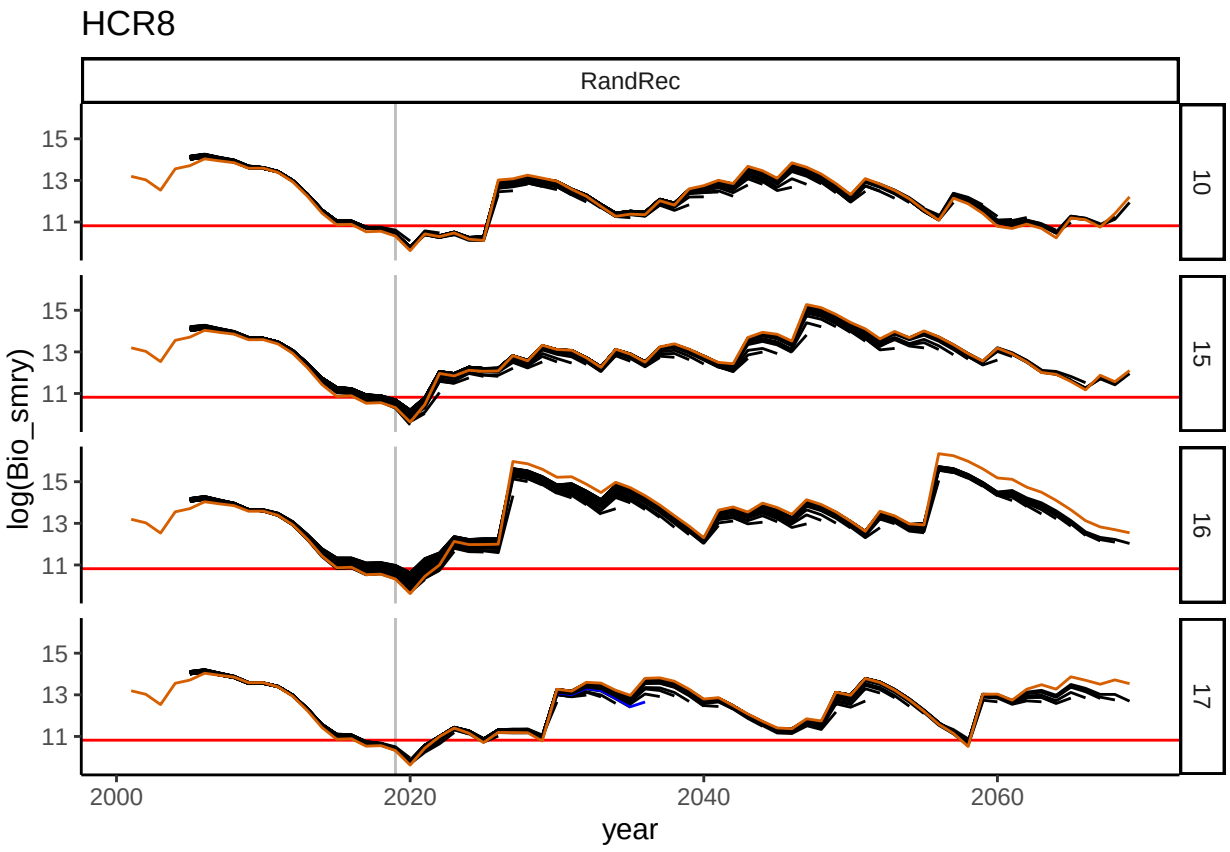

HCR2
RandRec
10
15
16
17
log(Bio_smry)
15
13
11
2000
2020
2040
2060
year


HCR3
RandRec
10
15
16
17
log(Bio_smry)
15
13
11
2000
2020
2040
2060
year


HCR5
RandRec
log(Bio_smry)
15
13
11
10
15
16
17
2000
2020
2040
2060
year


HCR6
RandRec
10
15
16
17
log(Bio_smry)
15
13
11
2000
2020
2040
2060
year


HCR7
RandRec
log(Bio_smry)
15
13
11
10
15
16
17
2000
2020
2040
2060
year

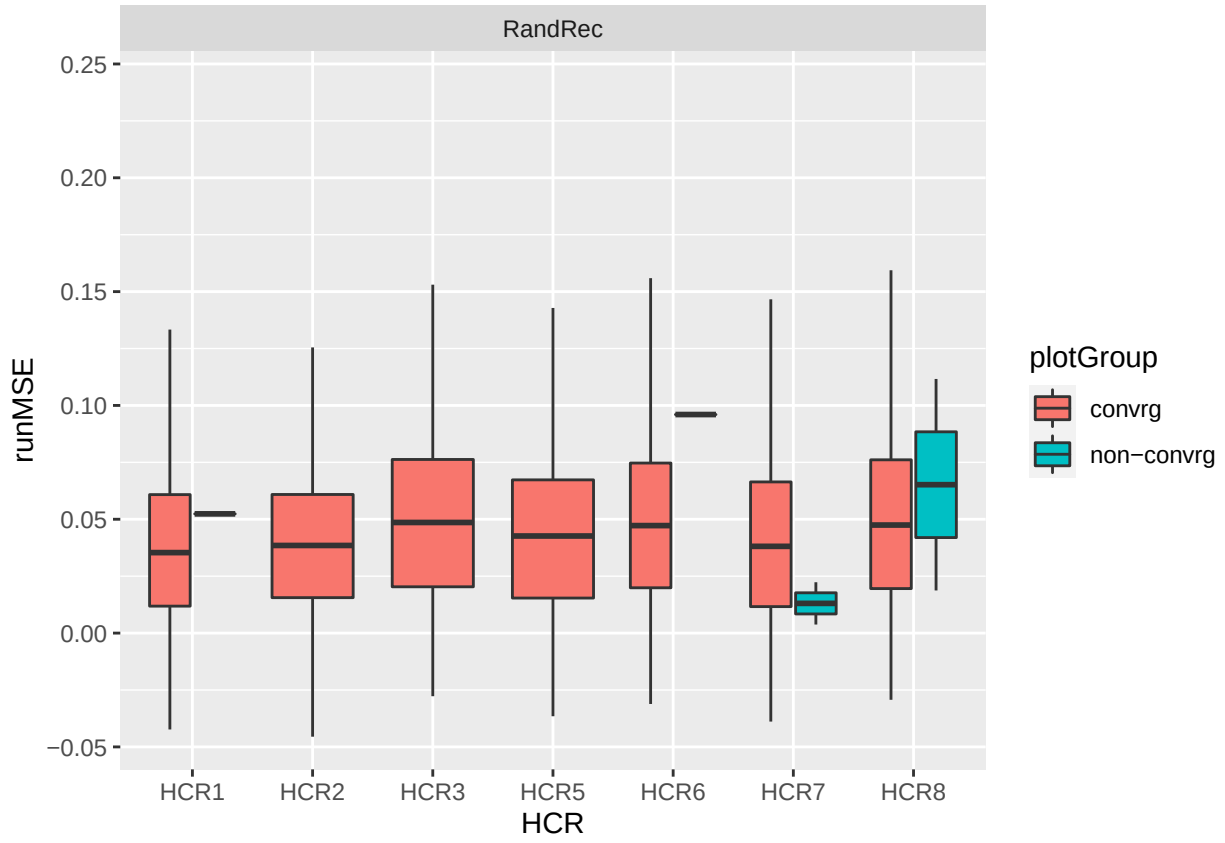



```
#termTS %>% filter(model_run == omName)

errCompare <- cnvrgTS %>% filter(Seas == 1, model_run != omName) %>%
                select(Bio_smry, year, model_run, iteration, scenario, HCR, recScen, emYear, plotGroup)
                inner_join(y = subset(termTS, model_run == omName),
                           by = c("year", "iteration", "scenario", "HCR", "recScen")) %>%
                #filter(iteration == 1) %>%
                  rename(age1plusOM = Bio_smry.y,
                         age1plusEM = Bio_smry.x) %>%
                  mutate(errSmryBio = (age1plusEM - age1plusOM)/age1plusOM) %>%
                select(age1plusEM, age1plusOM, errSmryBio, year, model_run.x,
                       iteration, scenario, HCR, recScen, plotGroup) %>%
                group_by(model_run.x, iteration, scenario, HCR, recScen, plotGroup) %>%
                summarize(runMSE = mean(errSmryBio)) #%>%
```

```
## `summarise()` has grouped output by 'model_run.x', 'iteration', 'scenario',
## 'HCR', 'recScen'. You can override using the `.groups` argument.
```

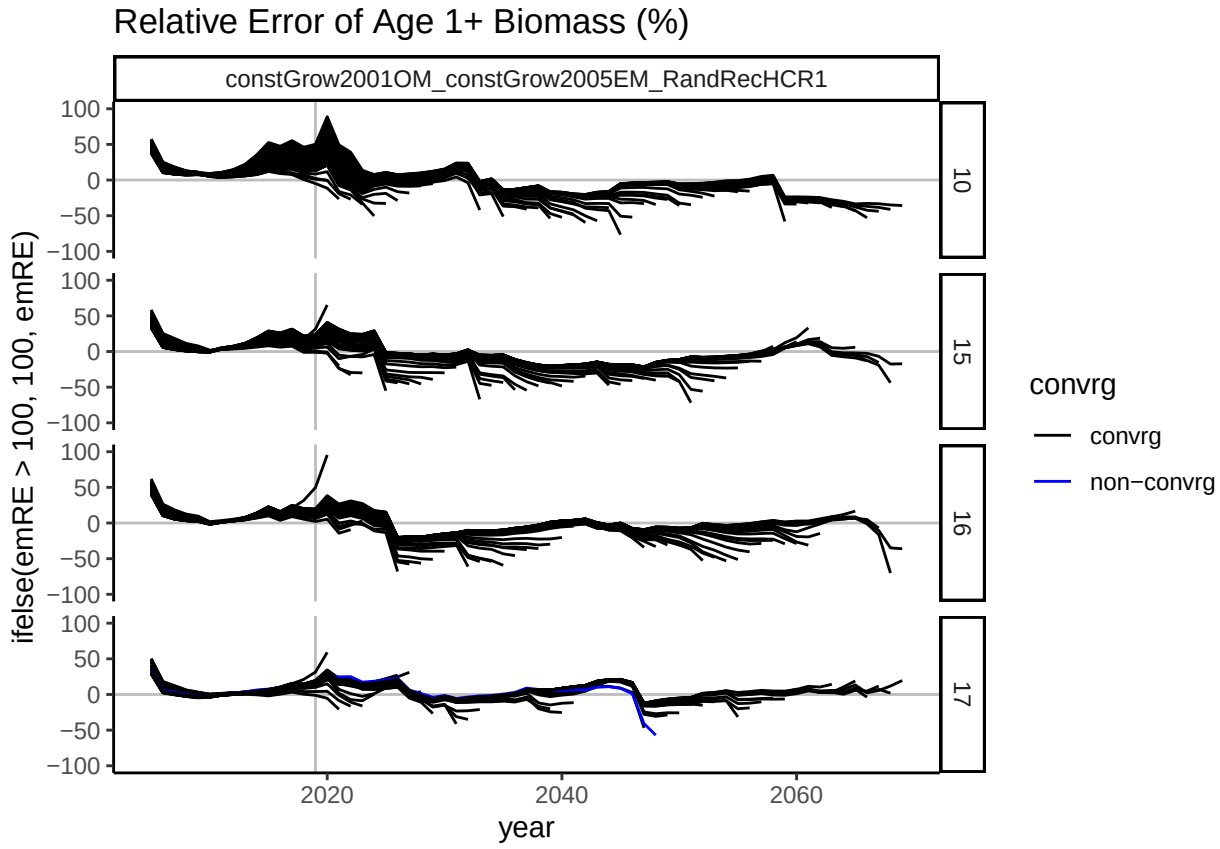
```
                # group_by(iteration, scenario, HCR, recScen, plotGroup) %>%
                # summarize(runMSE = mean(runMSE))  %>%
                # group_by(scenario, HCR, recScen, plotGroup) %>%
                # summarize(runMSE = mean(runMSE))

errCompare %>% #filter(HCR != "HCR3") %>%
  ggplot(aes(x = HCR, y = runMSE, fill = plotGroup)) +
```

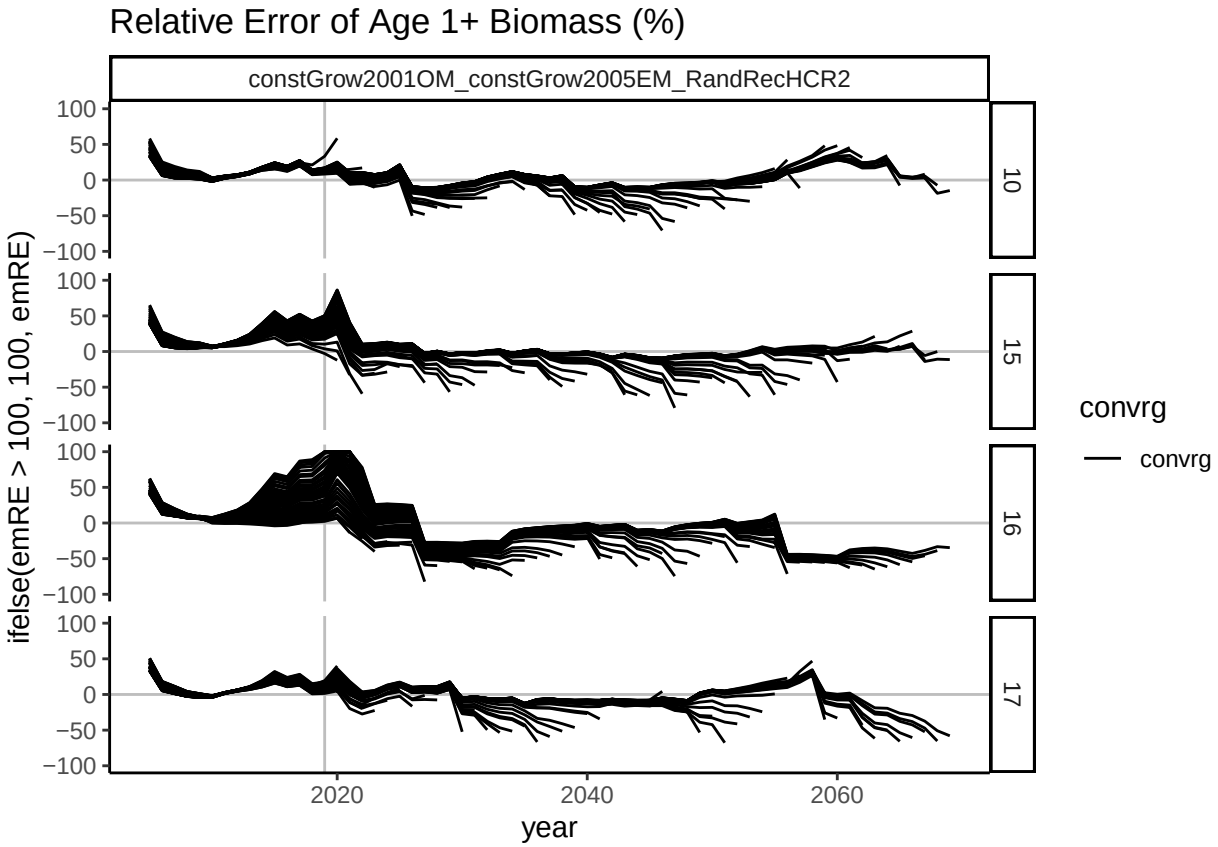
```
  geom_boxplot(outlier.shape = NA) +
  facet_wrap(~recScen) #+
```

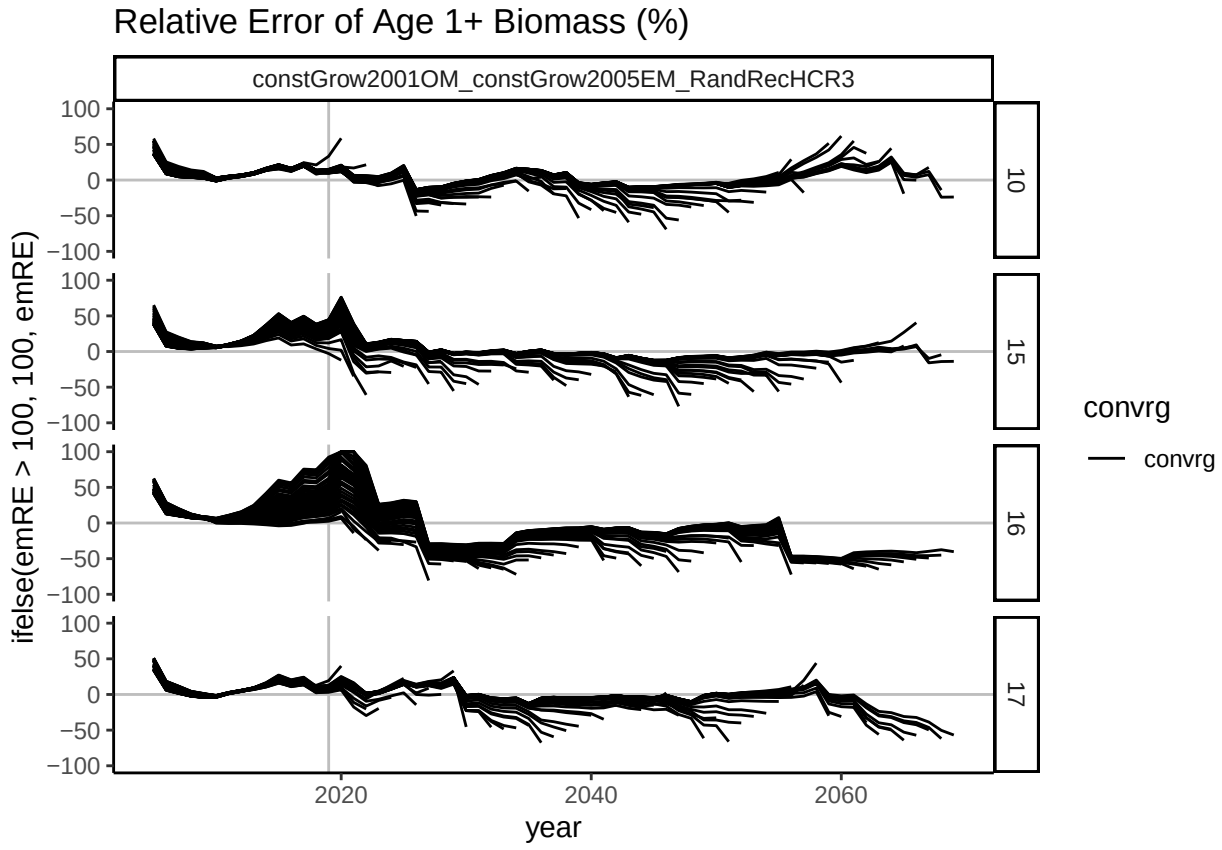


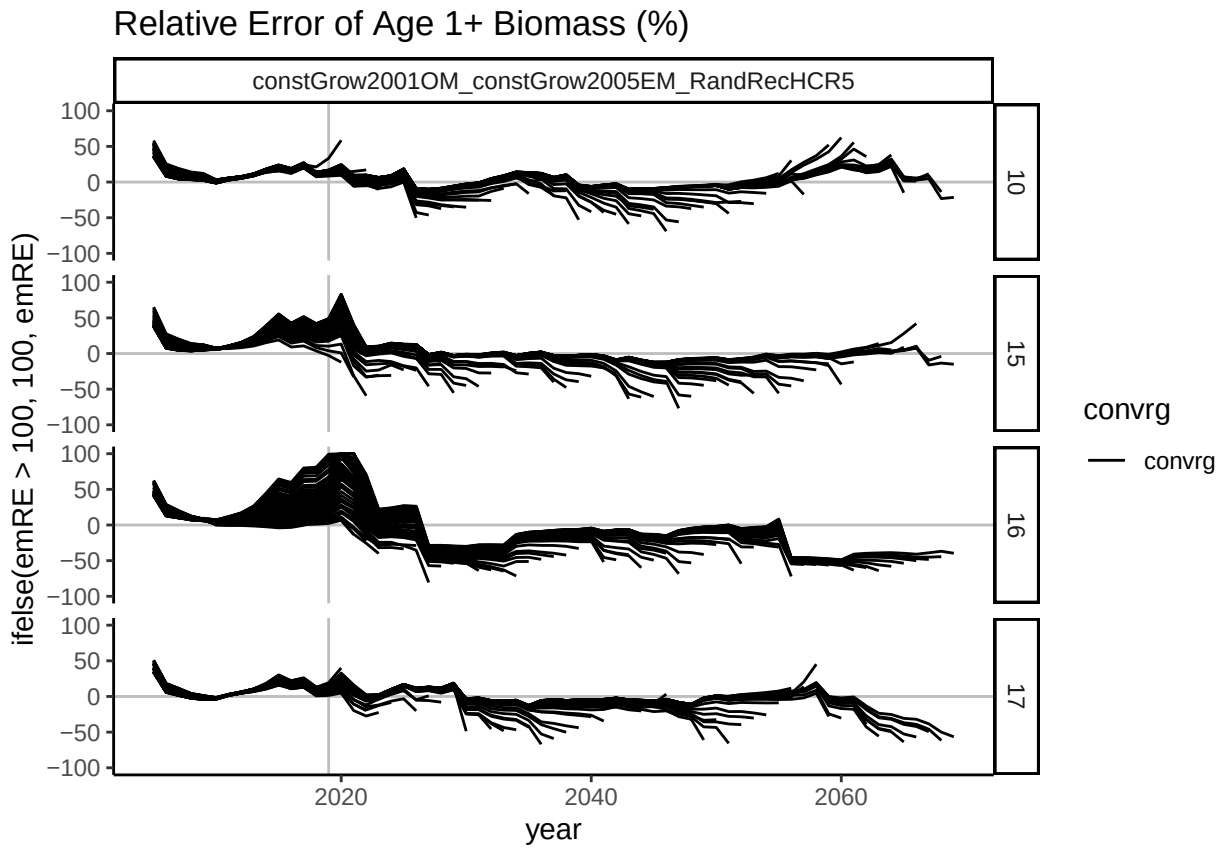

```
#  coord_cartesian(ylim =  c(0, 30))

age1PlusBio <- smryOutputList$tsSmry %>% filter(Seas == 1) %>%
                  select(year, Bio_smry, model_run, iteration, scenario) %>%
                  mutate(plotGroup = case_when(grepl("_OM", model_run, fixed = TRUE) ~ "OM",
                                               TRUE ~ "EM"))

age1PlusRE <- age1PlusBio %>% filter(plotGroup != "OM")
age1PlusRE <- age1PlusRE %>% pivot_wider(names_from = "plotGroup", values_from = "Bio_smry") %>%
                  left_join(y = convrgCheck,
                            by = c("model_run", "iteration", "scenario")) %>%
                  full_join(y = subset(age1PlusBio, subset = plotGroup == "OM"),
                            by = c("iteration", "scenario", "year")) %>%
                  mutate(convrg = case_when(max_grad > 0.01 ~ "non-convrg",
                                            max_grad < 0.01 ~ "convrg",
                                            TRUE ~ "OM"),
                         emRE = (EM - Bio_smry)/Bio_smry * 100)

# Plot relative errors of biomass over time
for(hcr in 2:length(scenarios)){
  print(age1PlusRE %>% filter(HCR == hcrs[hcr], iteration %in% exIters) %>%
    ggplot(aes(x = year, y = ifelse(emRE > 100, 100, emRE))) + #y = emRE)) +
```

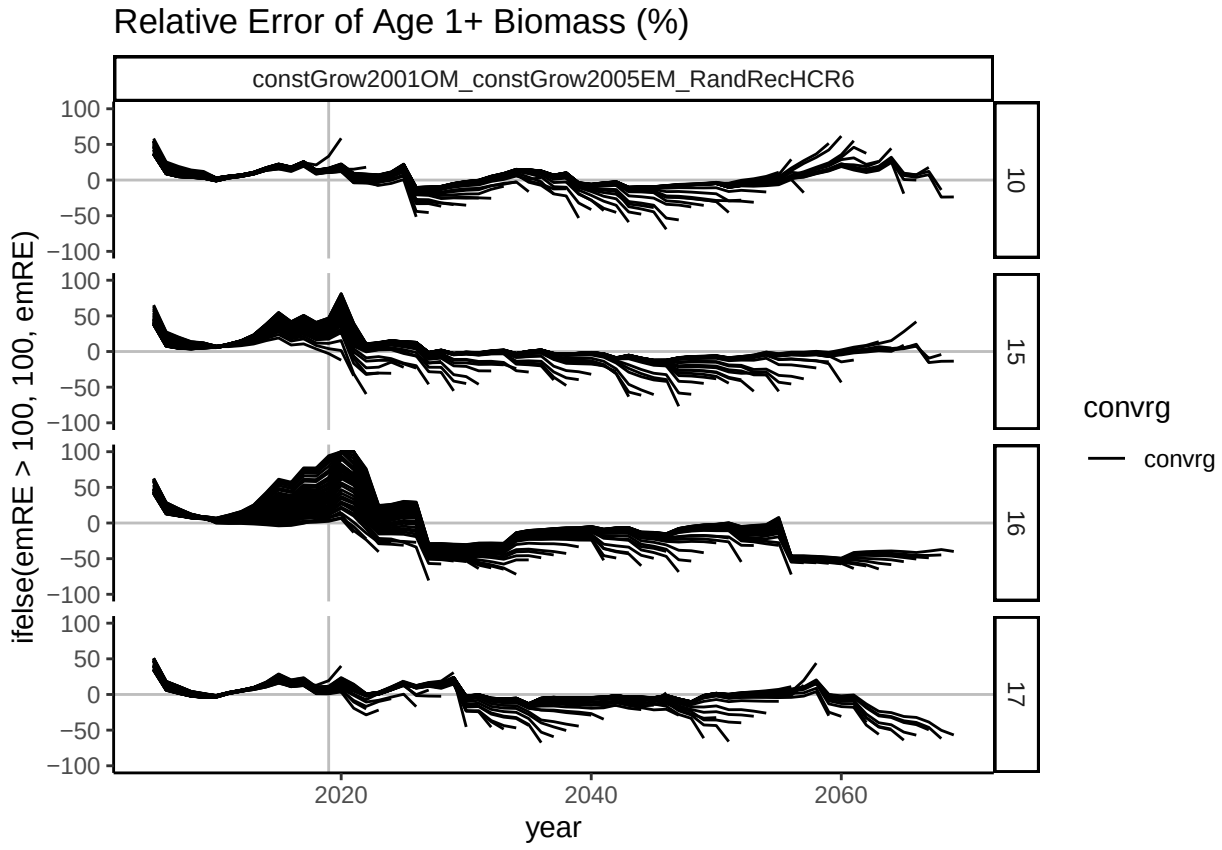
```
    geom_vline(xintercept = 2019, color = "gray") +
    geom_hline(yintercept = 0, color = "gray") +
    geom_line(aes(linetype = as.character(model_run.x), color = convrg))+
    scale_color_manual(values = c("black", "blue", "#D65F00")) +
    scale_linetype_manual(values = rep("solid", 51)) +
    guides(linetype = "none") +
    facet_grid(rows = vars(iteration), cols = vars(scenario)) +
    theme_classic() + labs(title = "Relative Error of Age 1+ Biomass (%)") +
    ylim(-100, 100))
}
```

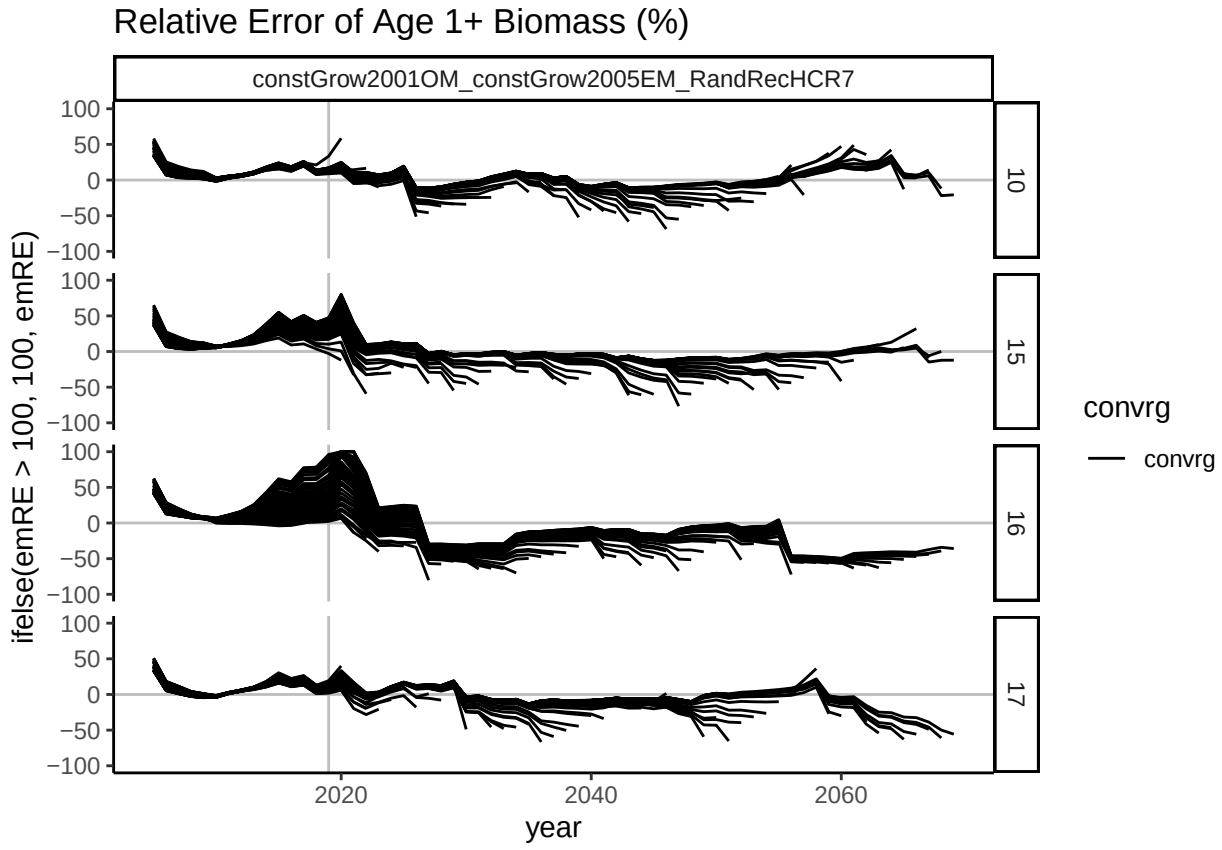


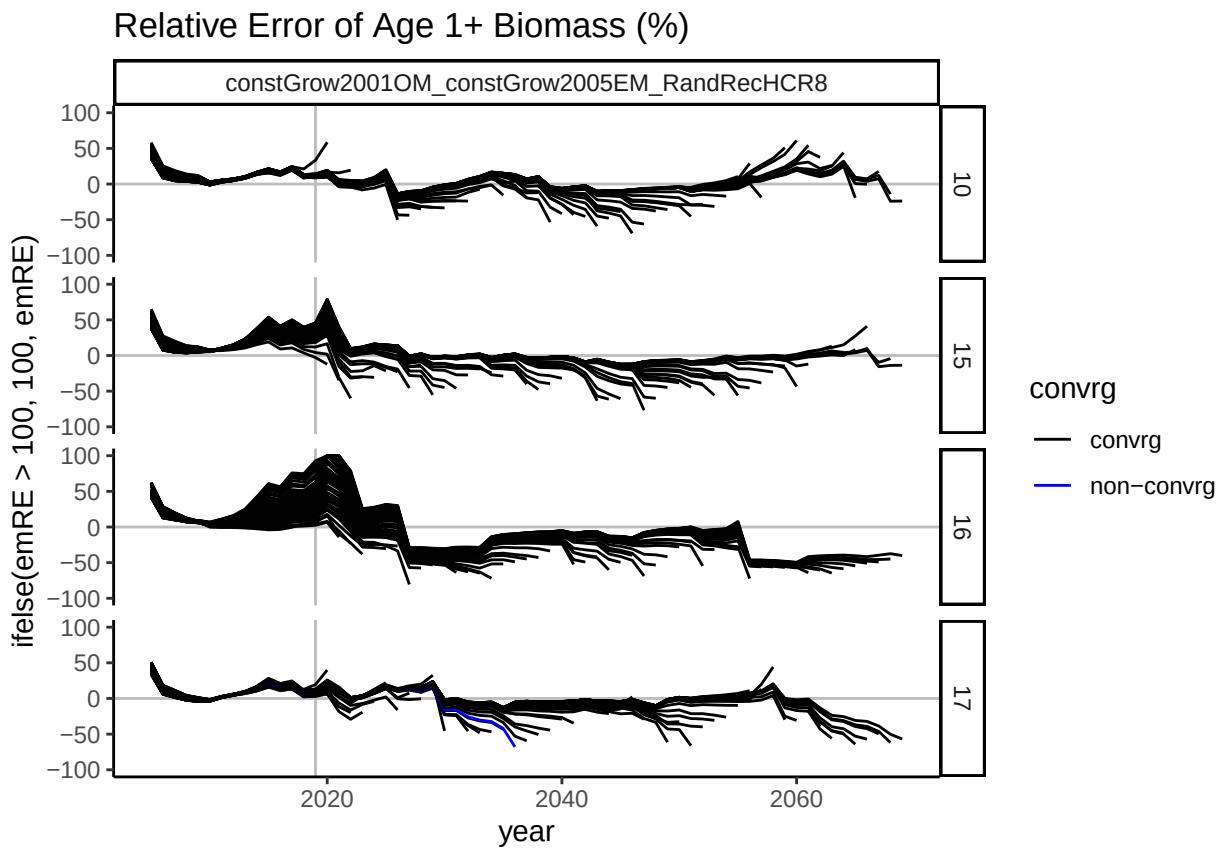

Relative Error of Age 1+ Biomass (%)
constGrow2001OM_constGrow2005EM_RandRecHCR2
ifelse(emRE > 100, 100, emRE)
100
50
0
−50
−100
10
15
16
17
2020
2040
2060
year
convrg
convrg


Relative Error of Age 1+ Biomass (%)
constGrow2001OM_constGrow2005EM_RandRecHCR3
ifelse(emRE > 100, 100, emRE)
100
50
0
−50
−100
10
15
16
17
2020
2040
2060
year
convrg
convrg


Relative Error of Age 1+ Biomass (%)
constGrow2001OM_constGrow2005EM_RandRecHCR5
ifelse(emRE > 100, 100, emRE)
100
50
0
−50
−100
10
15
16
17
2020
2040
2060
year
convrg
convrg

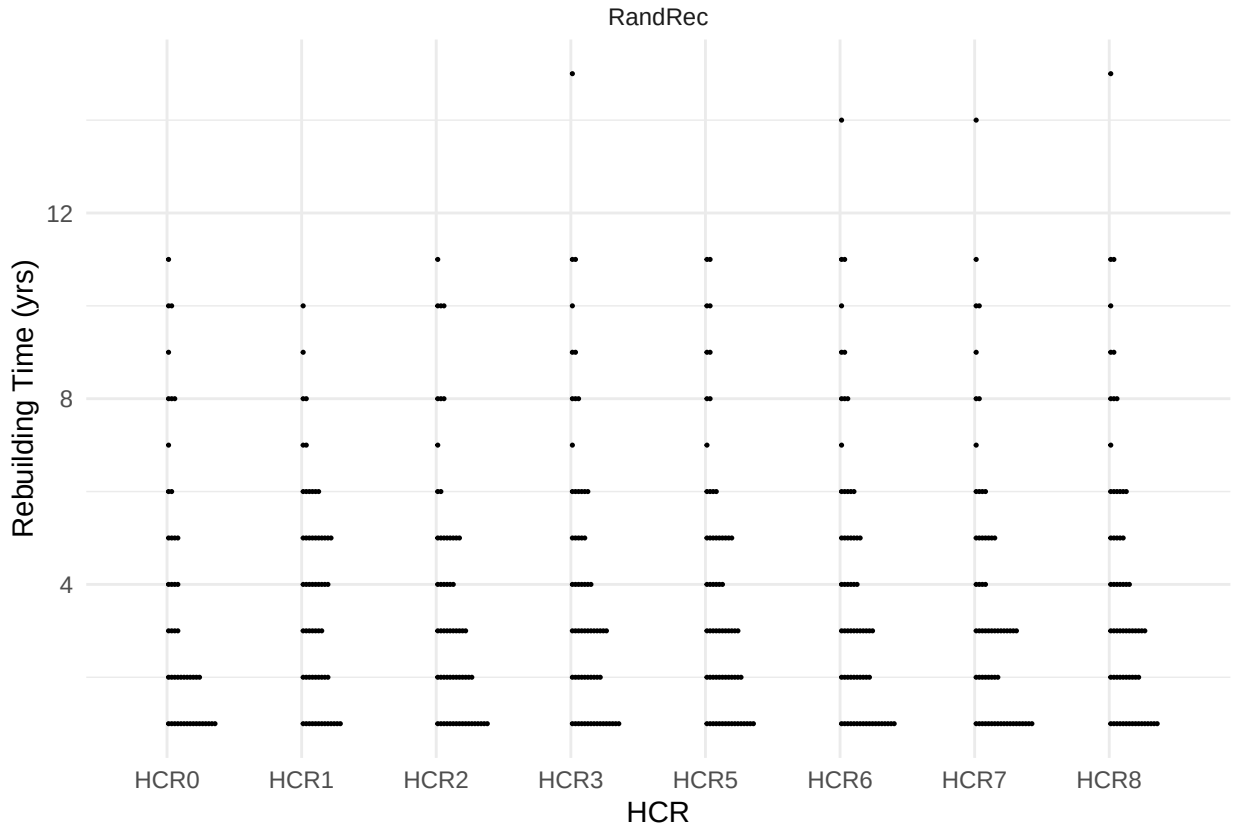

Relative Error of Age 1+ Biomass (%)
constGrow2001OM_constGrow2005EM_RandRecHCR6
ifelse(emRE > 100, 100, emRE)
100
50
0
−50
−100
10
15
16
17
2020
2040
2060
year
convrg
convrg


Relative Error of Age 1+ Biomass (%)
constGrow2001OM_constGrow2005EM_RandRecHCR7
ifelse(emRE > 100, 100, emRE)
100
50
0
−50
−100
10
15
16
17
2020
2040
2060
year
convrg
convrg



Investigate rebuilding lengths

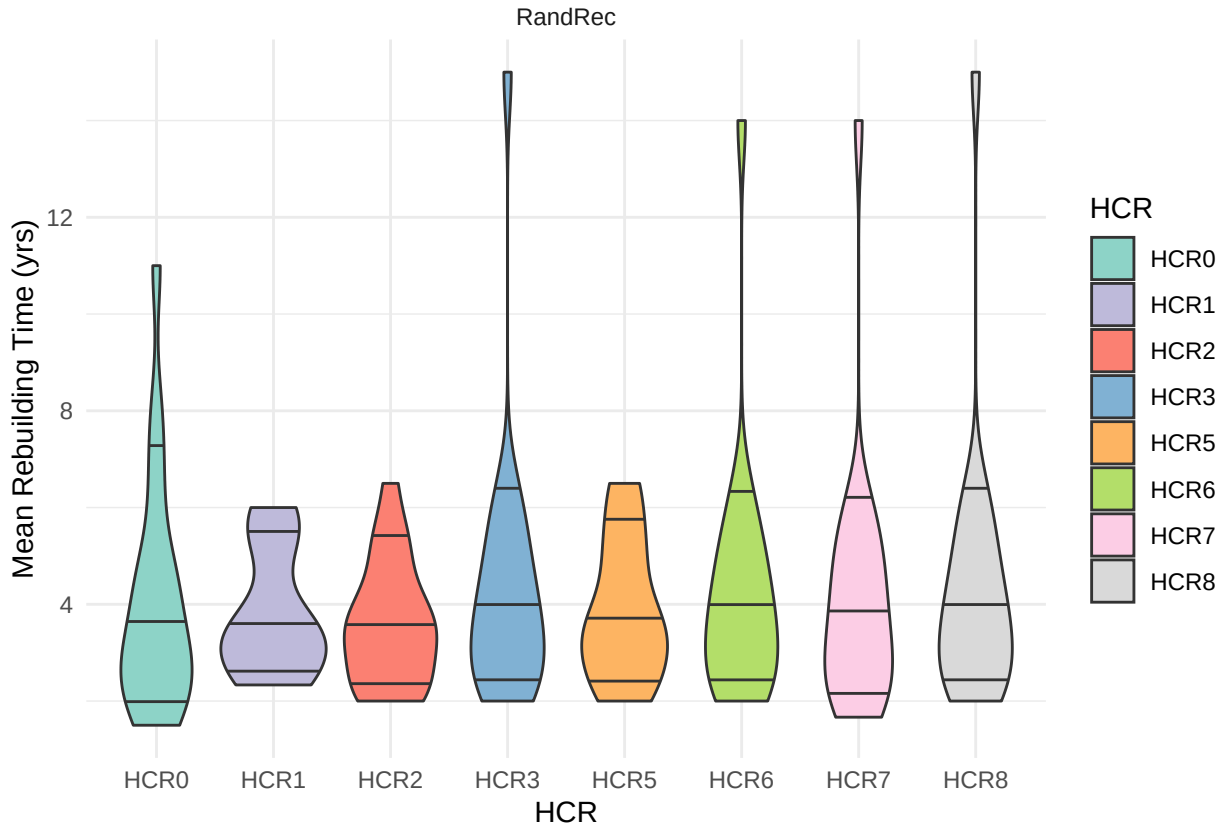
```
rebuildTime <- performanceList$tsSmry %>% select(Bio_smry , year, model_run, iteration, scenario, closu
  filter(closure) %>% arrange(scenario, iteration, year) #%>% filter(year == 2020)
#all(rebuildTime$closure)
rebuildTime$isRebuilt <- NA
for(i in 1:(nrow(rebuildTime)-1)){
  rebuildTime$isRebuilt[i] <- if(rebuildTime$closureLength[i] != rebuildTime$closureLength[i+1]-1){
                                TRUE
                              } else { FALSE }
}
rebuildTime <- rebuildTime %>% filter(isRebuilt, year != 2069) %>% # remove last year b/c can't guarant
                   mutate(HCR = sub(pattern = ".*Rec","", scenario),
                          recScen = sub(pattern = "HCR.*","", scenario)) %>%
                   mutate(recScen = sub(pattern = ".*EM_","", recScen))

rebuildTime %>%
  ggplot(aes(x = HCR, y = closureLength)) +
  #geom_violin(aes(fill = HCR), draw_quantiles = c(0.1, 0.5, 0.9)) +
  geom_dotplot(binaxis = "y", binwidth = 1/15) +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal) +
  labs(y = "Rebuilding Time (yrs)", x = "HCR")
```



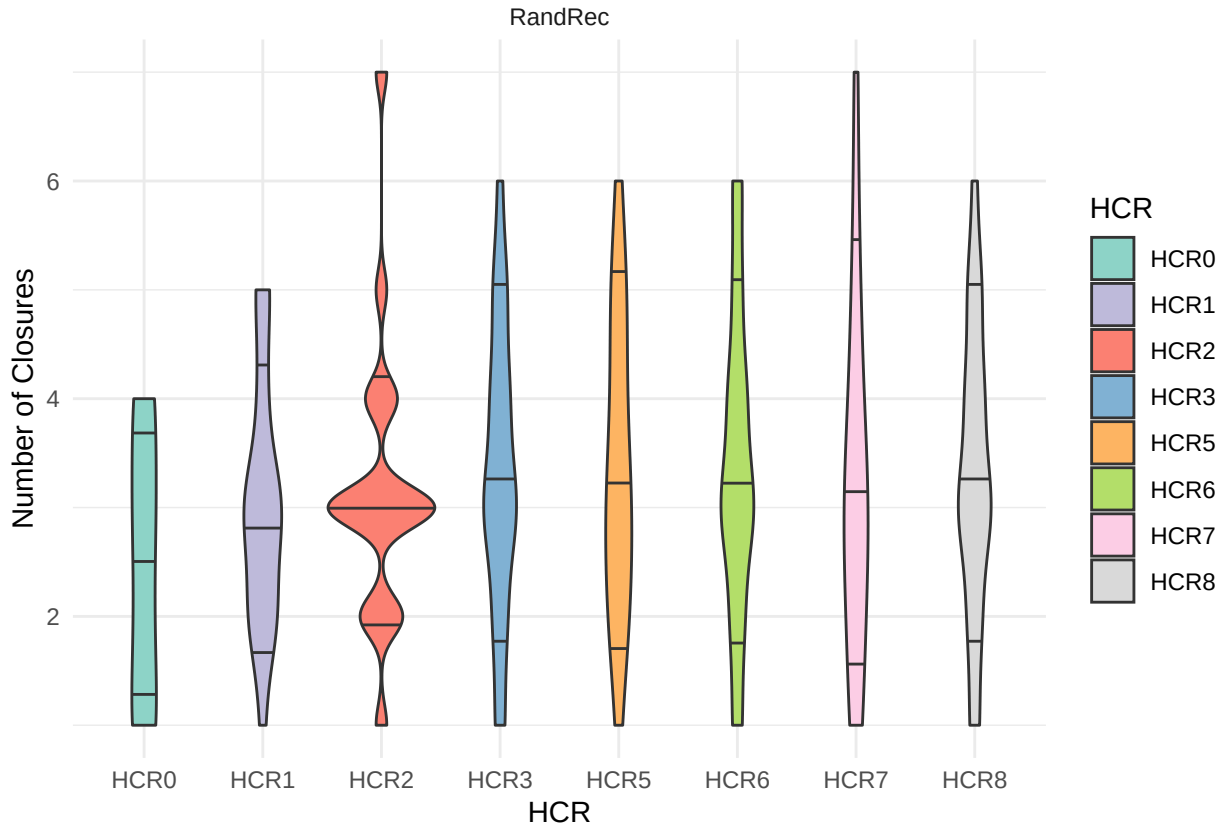

```
rebuildTime <- rebuildTime %>% group_by(model_run, iteration, scenario, HCR, recScen) %>%
                  summarize(nClosures = n(),
                            meanRebuildTime = mean(closureLength))
```

```
## `summarise()` has grouped output by 'model_run', 'iteration', 'scenario',
## 'HCR'. You can override using the `.groups` argument.
```

```
rebuildTime %>% filter(recScen == "RandRec", HCR != "HCR4") %>%
  ggplot(aes(x = HCR, y = meanRebuildTime)) +
  geom_violin(aes(fill = HCR), draw_quantiles = c(0.1, 0.5, 0.9)) +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal) +
  labs(y = "Mean Rebuilding Time (yrs)", x = "HCR")
```




```
rebuildTime %>%
  ggplot(aes(x = HCR, y = nClosures)) +
  geom_violin(aes(fill = HCR), draw_quantiles = c(0.1, 0.5, 0.9)) +
  facet_wrap(~recScen) +
  theme_minimal() +
  scale_fill_manual(values = hcrPal) +
  labs(y = "Number of Closures", x = "HCR")
```


RandRec
Number of Closures
6
4
2
HCR0
HCR1
HCR2
HCR3
HCR5
HCR6
HCR7
HCR8
HCR
HCR
HCR0
HCR1
HCR2
HCR3
HCR5
HCR6
HCR7
HCR8