Delaware River PIT tag data analysis

Ben Letcher

2022-08-04

Note: the results shown here are preliminary and have not been officially reviewed by USGS, NYDEC or PA Fish and Boat

This notebook uses targets to manage running code and updating R objects. Targets sets up dependencies among specified objects and only re-runs code as necessary (when an upstream component gets updated). This can save run times for projects with models that take a while to run, like capture-mark-recapture models.

Data preparation and model running happens using targets and exploration of the data and model runs is below in this Markdown document.

'tar_make()' runs all the R scripts and functions specified in '_targets.R'. Only updated code or sections that are downstream from updated data are re-run.
'tar_read()' reads 'target' data into the global environment.

To set up a targets project, use use_targets()

This section (tar_make()) reruns the model and has some other helpful functions.

```
#  tar_watch(seconds = 10, outdated = FALSE, targets_only = TRUE)


# comment this out when knitting - get Latex error that it can't find the check mark the tar_make() use
#tar_make()

# tar_prune() # cleans unused data files
#tar_invalidate(everything())
#tar_invalidate(ends_with("ttt"))


#str(d)
```

Load raw data for exploration
Without a postscript ('_main' or '_trib'), the data are for all fish (main and trib).

```
dRaw0 <- tar_read(dRaw0) #all data - including untagged
dRaw <- tar_read(dRaw) #all data for CMR models
d <- tar_read(target_d)
eh <- tar_read(target_eh)
```

Visualize the network - does not work with pdf output

```
#tar_visnetwork()
```

Which rivers (Water) riverN corresponds to

```
table(d$Water, d$riverN)
#>
#>                           1    2    3    4    5    6
#>   Balls Creek            41    0    0    0    0    0
#>   Cold Spring Creek       0   95    0    0    0    0
#>   Roods Creek             0    0  159    0    0    0
#>   Sands Creek             0    0    0  139    0    0
#>   Shehawken Creek         0    0    0    0   91    0
#>   West Br Delaware River  0    0    0    0    0 5634
```

Raw data summary tables

```
kable(data.frame(ftable(d$date)))
```

Var1	Freq
2018-05-07	110
2018-05-08	34
2018-05-09	48
2018-06-11	129
2018-06-12	99
2018-06-13	88
2018-07-16	212
2018-07-17	176
2018-07-18	142
2018-08-21	11
2018-09-17	21
2018-09-20	89
2018-10-22	85
2018-10-23	131
2018-10-24	129
2019-04-08	240
2019-04-10	129
2019-05-06	170
2019-05-07	91
2019-06-10	169
2019-06-11	129
2019-07-15	212
2019-07-16	312
2019-07-17	25
2019-08-12	131
2019-08-13	139
2019-08-14	186
2019-08-15	49
2019-09-16	108
2019-09-17	55

Var1	Freq
2019-09-18	293
2019-10-21	262
2019-10-22	31
2019-10-23	74
2020-07-16	146
2020-07-20	249
2020-07-21	29
2020-08-10	89
2020-08-11	41
2020-08-17	145
2020-08-20	110
2020-09-10	187
2020-09-14	252
2020-09-15	47
2020-10-13	368
2020-10-14	55
2020-10-15	132

```
#kable(data.frame(ftable(d$Water, d$riverN)))

#kable(data.frame(ftable(d$Water, d$riverN, d$date)))
kable(data.frame(ftable(d$species)))
```

Var1	Freq
brook trout	13
brown trout	5534
rainbow trout	611

```
### Number of unique tags
length(unique(d$tag))
#> [1] 4610
```

Group observations by month.
Luckily, sampling periods do not span months, so we can use month as a grouping variable for sampling occasion

```
kable(data.frame(ftable(d$dateYM)))
```

Var1	Freq
2018-05	192
2018-06	316
2018-07	530
2018-08	11
2018-09	110
2018-10	345
2019-04	369
2019-05	261

Var1	Freq
2019-06	298
2019-07	549
2019-08	505
2019-09	456
2019-10	367
2020-07	424
2020-08	385
2020-09	486
2020-10	555

Tag information

Grouped by Water (sampling area)

```
tagN <- d %>%
  group_by(tag, Water) %>%
  summarize(n = n()) %>%
  filter(tag != "") %>%
  arrange(desc(n))
#> `summarise()` has grouped output by 'tag'. You can override using the `.groups`
#> argument.

### Number of times individual fish were observed
table(tagN$n)
#>
#>    1    2    3    4    5    6    7    8    9
#> 3641  611  223   90   30   12    4    1    1

### Number of times individual fish were observed by river
(table(tagN$Water, tagN$n))
#>
#>                            1    2    3    4    5    6    7    8    9
#>   Balls Creek             30    4    1    0    0    0    0    0    0
#>   Cold Spring Creek       39    9    4    2    1    1    1    0    0
#>   Roods Creek             78   21    6    4    1    0    0    0    0
#>   Sands Creek             97   16    2    1    0    0    0    0    0
#>   Shehawken Creek         45   10    6    2    0    0    0    0    0
#>   West Br Delaware River 3352  551  204   81   28   11    3    1    1
```

Grouped by state

States
River size1 size2 size3
Main 1 2 3
Trib 4 5 6

```
tagN_s <- d %>%
  group_by(tag, state) %>%
  summarize(n = n()) %>%
  filter(tag != "") %>%
```

```
  arrange(desc(n))
#> `summarise()` has grouped output by 'tag'. You can override using the `.groups`
#> argument.

### Number of times individual fish were observed
table(tagN_s$n)
#>
#>    1    2    3    4    5    6    7    9
#> 3949  640  170   57   23    9    2    1

### Number of times individual fish were observed by river
table(tagN_s$state, tagN_s$n)
#>
#>        1    2    3    4    5    6    7    9
#>   1  963   69    4    0    0    0    0    0
#>   2 1660  274   58   17    7    0    0    0
#>   3 1025  232   87   35   15    8    2    1
#>   4  264   59   15    5    1    1    0    0
#>   5   29    6    6    0    0    0    0    0
#>   6    8    0    0    0    0    0    0    0
```

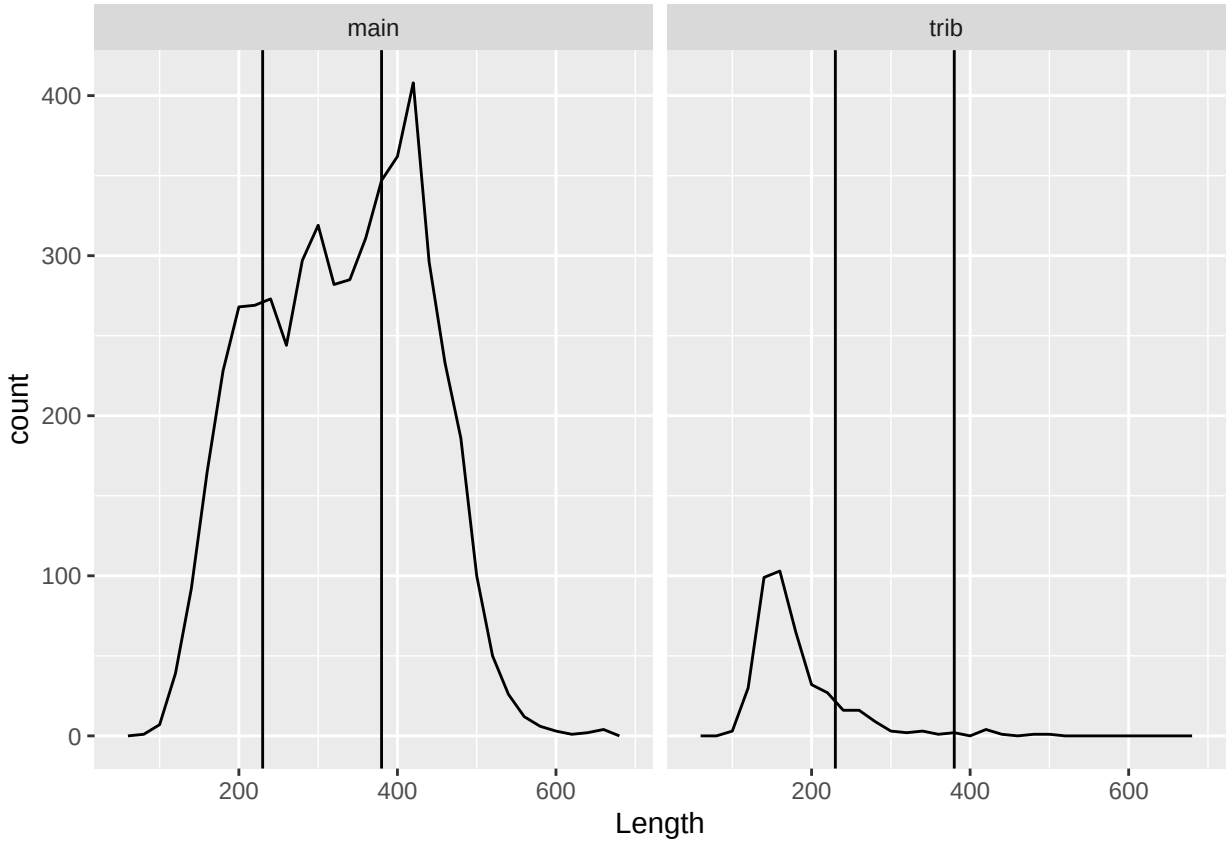
Grouped by main/trib This is what is used for the primary analysis

```
tagN_mt <- d %>%
  group_by(tag, mainTrib) %>%
  summarize(n = n()) %>%
  filter(tag != "") %>%
  arrange(desc(n))
#> `summarise()` has grouped output by 'tag'. You can override using the `.groups`
#> argument.

### Number of times individual fish were observed
table(tagN_mt$n)
#>
#>    1    2    3    4    5    6    7    8    9
#> 3641  611  223   90   30   12    4    1    1

### Number of times individual fish were observed by river
(table(tagN_mt$mainTrib, tagN_mt$n))
#>
#>           1    2    3    4    5    6    7    8    9
#>   main 3352  551  204   81   28   11    3    1    1
#>   trib  289   60   19    9    2    1    1    0    0
```

Basic summary plots of raw tagging data

```
ggplot(d %>% filter(species == "brown trout"), aes(Length)) +
  geom_freqpoly() +
  geom_vline(xintercept = c(tar_read(target_sizeCutoff1), tar_read(target_sizeCutoff2))) +
```

```
  facet_grid(~mainTrib)
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
# dTame <- d %>%
#             select(Latitude, Longitude, tag, dateTime, species, Length, Weight) %>%
#             filter(tag != "", tag != "ad")
#
# write.csv(dTame, './dataOut/dTame.csv', row.names = FALSE)
```

Encounter histories

This is the data structure for the capture-recapture models. Each column is a sampling 'occasion' (here = month) and each row is an individual, where a '1' indicates capture and a '0' indicates not captured.

```
str(eh$eh)
#>  num [1:3673, 1:17] 1 1 1 1 1 1 1 1 1 1 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : NULL
#>   ..$ : chr [1:17] "date_2018-05" "date_2018-06" "date_2018-07" "date_2018-08" ...
kable(head(eh$eh,8))
```

date__2018-05	2018-06	2018-07	2018-08	2018-09	2018-10	2019-04	2019-05	2019-06	2019-07	2019-08	2019-09	2019-10	2019-07	2020-08	2020-09	2020-10	2020-
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

```
table(paste(eh$first, eh$last, sep="_"))
#>
#>  1_17 10_17 11_17 12_17 13_17 14_17 15_17 16_17  2_17  3_17  4_17  5_17  6_17
#>   173   360   295   264   257   297   246   291   264   373     8    62   231
#>  7_17  8_17  9_17
#>   233   130   189
```

There are very few transitions between main and trib and we can characterize size transitions better with a growth model - run phiT_pT_psiT models separately for main and trib

```
s <- eh$stateMatrix
s[s == 0] <- NA

s2 <- data.frame(s) %>%
  unite("all", sep = "", na.rm = TRUE, remove = TRUE) %>%
  #distinct() %>%
  arrange()

table(s2)
#> all
#>         1        11       111      1112       112      1122        12       122
#>       774        58         3         1         6         3        34        14
#>      1222       132         2        22       222      2222     22222      2223
#>         5         1      1097       185        46        17         7         4
#>     22233       223      2233     22333    223333   2233333  22333333        23
#>         2        25        10         6         2         1         1        53
#>       233      2333     23333    233333       253         3        32        33
#>        33         8         1         2         1       687         2       172
#>       333      3333     33333    333333   3333333 333333333         4        42
#>        71        32        12         7         2         1       182         1
#>        44       444      4444     44444    444444     44445   4444555      4445
#>        47        12         3         1         1         1         1         1
#>      4455        45       455      4555         5      5455        55       555
#>         2         2         2         2        17         1         1         2
#>       556         6        63
#>         1         6         1
```

Summary info for years and occasions

```
years <- colnames(eh$eh) %>%
  substr(6,9) %>%
  as.numeric()

months <- colnames(eh$eh) %>%
  substr(11,12) %>%
  as.numeric()

occs <- colnames(eh$eh)
```

Models

'phi' = apparent survival (probability of staying in the area = p(survival) + p(not moving out of area)). 'p' = probability of capture given that the fish is alive. 'psi' = probability of transitioning from one state to another. Here, states are size bins.

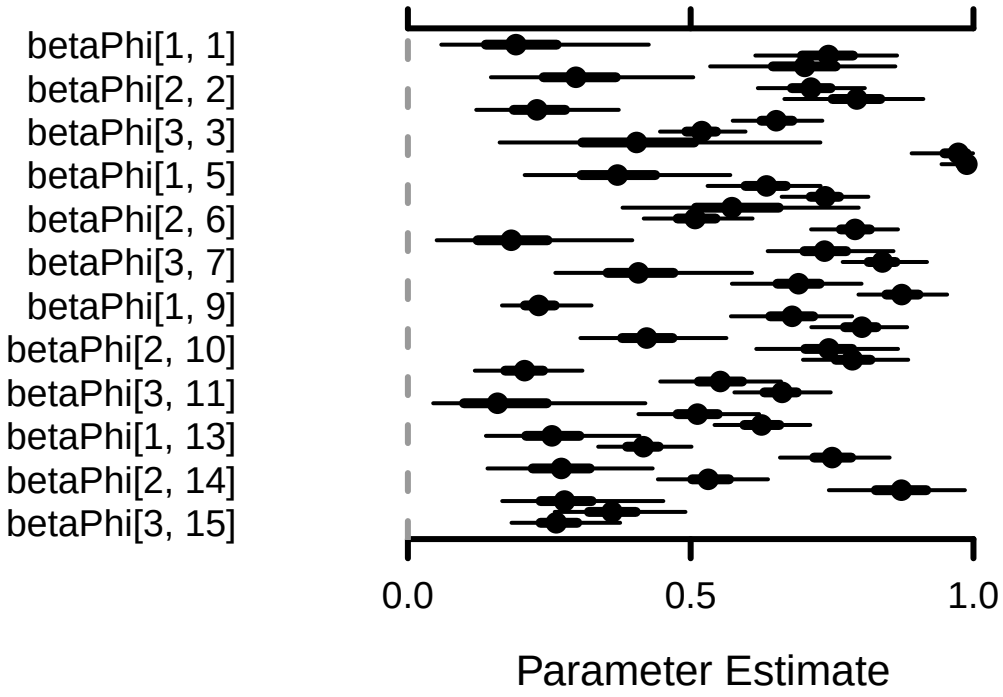
phiT_pT_psiT_main

Load data 'main' for analysis

```
#d_tt <- tar_read(target_d_trib)
eh_main <- tar_read(target_eh_main)

str(eh_main$eh)
#>  num [1:3388, 1:16] 1 1 1 1 1 1 1 1 1 1 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : NULL
#>   ..$ : chr [1:16] "date_2018-05" "date_2018-06" "date_2018-07" "date_2018-09" ...
#kable(eh_main$eh[1:8,1:10])

table(paste(eh_main$first, eh_main$last, sep="_"))
#>
#>  1_16 10_16 11_16 12_16 13_16 14_16 15_16  2_16  3_16  4_16  5_16  6_16  7_16
#>   173   242   244   238   297   246   291   264   313    51   216   233   130
#>  8_16  9_16
#>   189   261
```

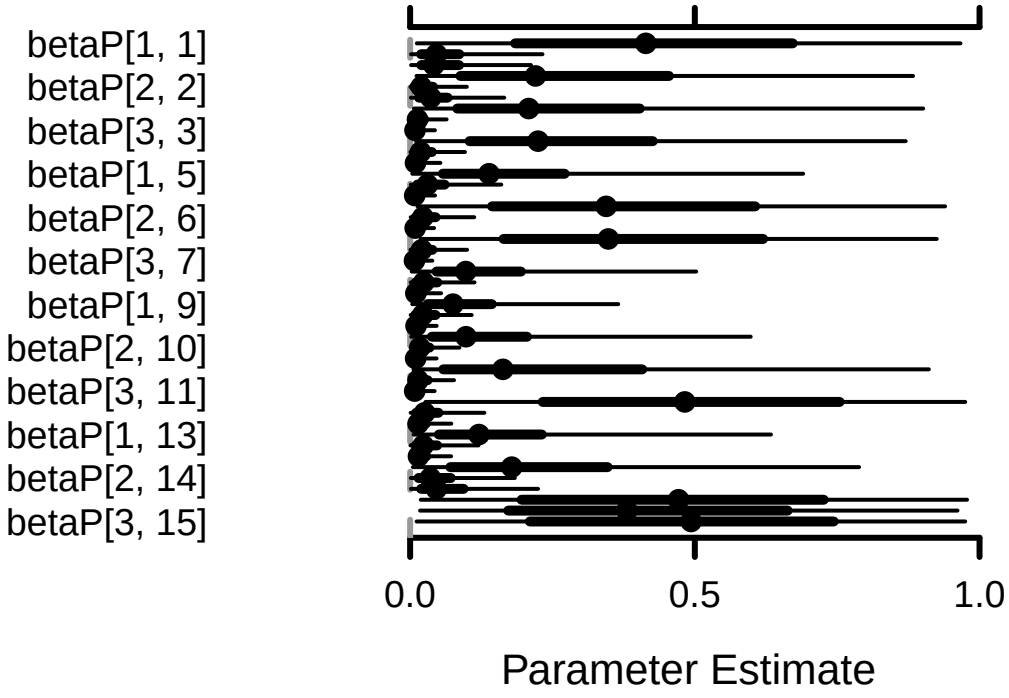
```
### Read the model run into global memory
if (tar_exist_objects(c("tt_modelOut_main"))) {
  mod_tt_main <- tar_read(tt_modelOut_main)

  MCMCplot(object = mod_tt_main$mcmc, params = "betaPhi")
  MCMCplot(object = mod_tt_main$mcmc, params = "betaP")

   priors <- rnorm(tar_read(tt_runData_main)$nIter * tar_read(tt_runData_main)$nChains, 0, 1/sqrt(.1))
   MCMCtrace(object = mod_tt_main$mcmc,
             ISB = FALSE,
             exact = TRUE,
             params = c("betaPhi[1, 1]", "betaPhi[2, 1]", "betaPhi[3, 1]",
                        "betaPhi[1, 2]", "betaPhi[2, 2]", "betaPhi[3, 2]"
                        #"betaPhi[1, 3]", "betaPhi[2, 3]", "betaPhi[3, 3]"
```

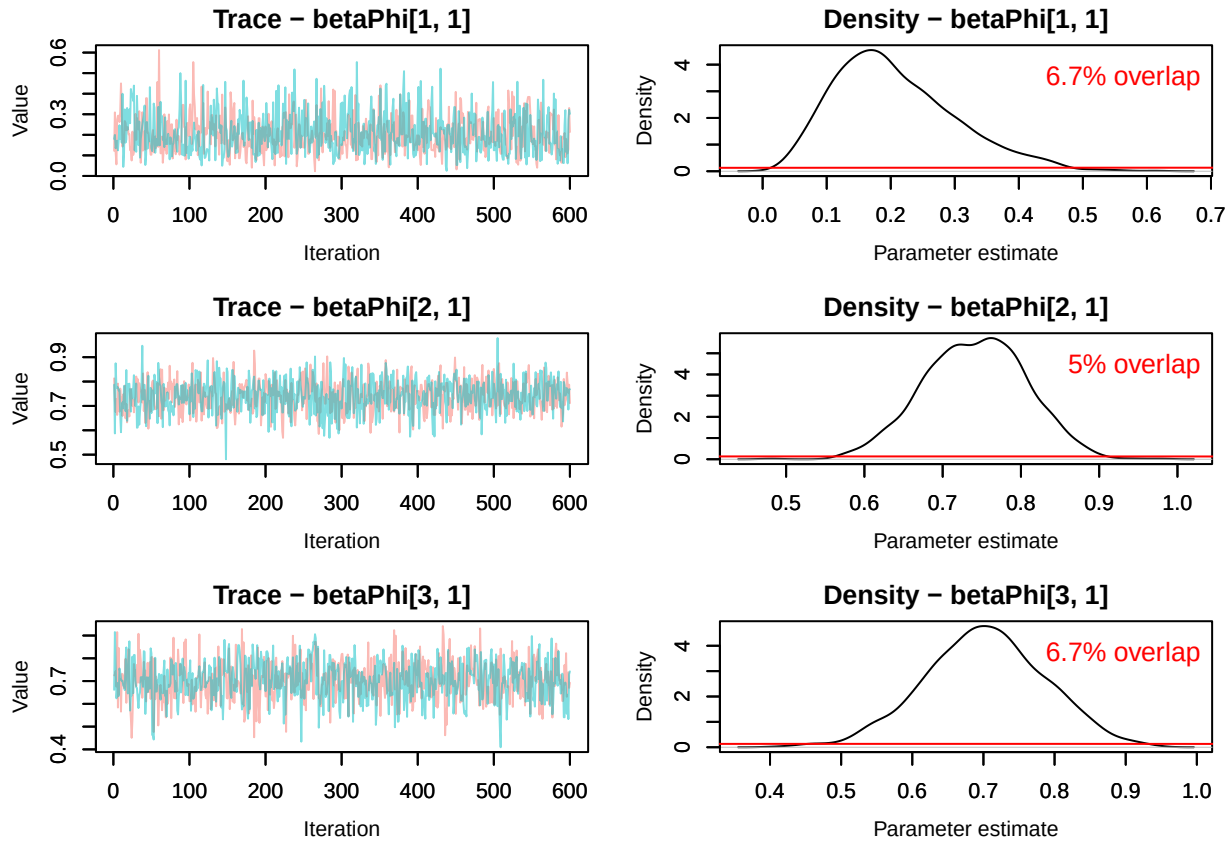
```
                        ),
              pdf = FALSE,
              priors = priors
              )

      MCMCtrace(object = mod_tt_main$mcmc,
              ISB = FALSE,
              exact = TRUE,
              params = c("betaP[1, 1]", "betaP[2, 1]", "betaP[3, 1]",
                         "betaP[1, 2]", "betaP[2, 2]", "betaP[3, 2]"
                         #"betaP[1, 3]", "betaP[2, 3]", "betaP[3, 3]"
                         ),
              pdf = FALSE,
              priors = priors
              )

    modSummary_tt_main <- MCMCsummary(object = mod_tt_main$mcmc, round = 3) %>%
      rename(lo = '2.5%', med = '50%', hi = '97.5%')
}
```



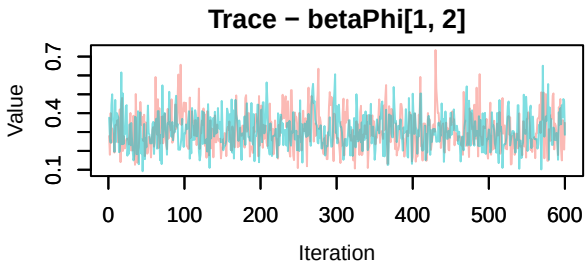

```
#> Warning in MCMCtrace(object = mod_tt_main$mcmc, ISB = FALSE, exact = TRUE, :
#> Only one prior specified for > 1 parameter. Using a single prior for all
#> parameters.
#> Warning in MCMCtrace(object = mod_tt_main$mcmc, ISB = FALSE, exact = TRUE, :
#> Number of samples in prior is greater than number of total or specified
```

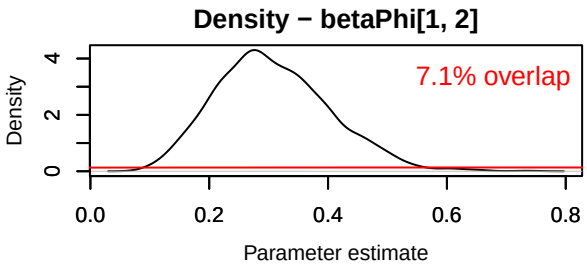
```
#> iterations (for all chains) for specified parameter. Only last 1200 iterations
#> will be used.
```

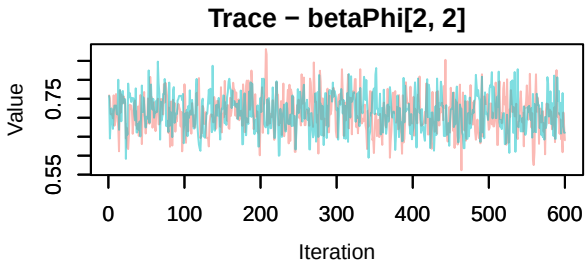


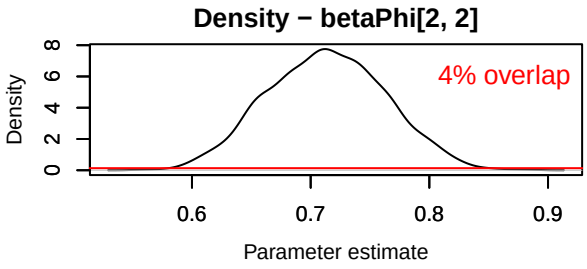


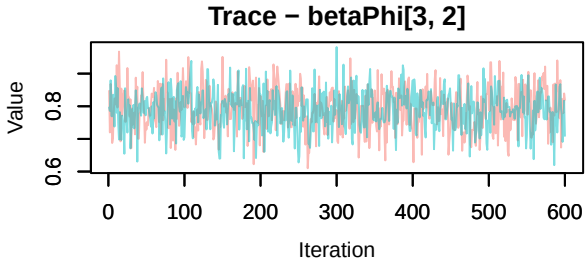

```
#> Warning in MCMCtrace(object = mod_tt_main$mcmc, ISB = FALSE, exact = TRUE, :
#> Only one prior specified for > 1 parameter. Using a single prior for all
#> parameters.
#> Warning in MCMCtrace(object = mod_tt_main$mcmc, ISB = FALSE, exact = TRUE, :
#> Number of samples in prior is greater than number of total or specified
#> iterations (for all chains) for specified parameter. Only last 1200 iterations
#> will be used.
```

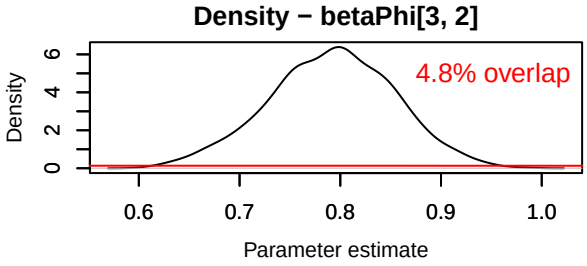

Trace − betaPhi[1, 2]
Value
0.1
0.4
0.7
0
100
200
300
400
500
600
Iteration

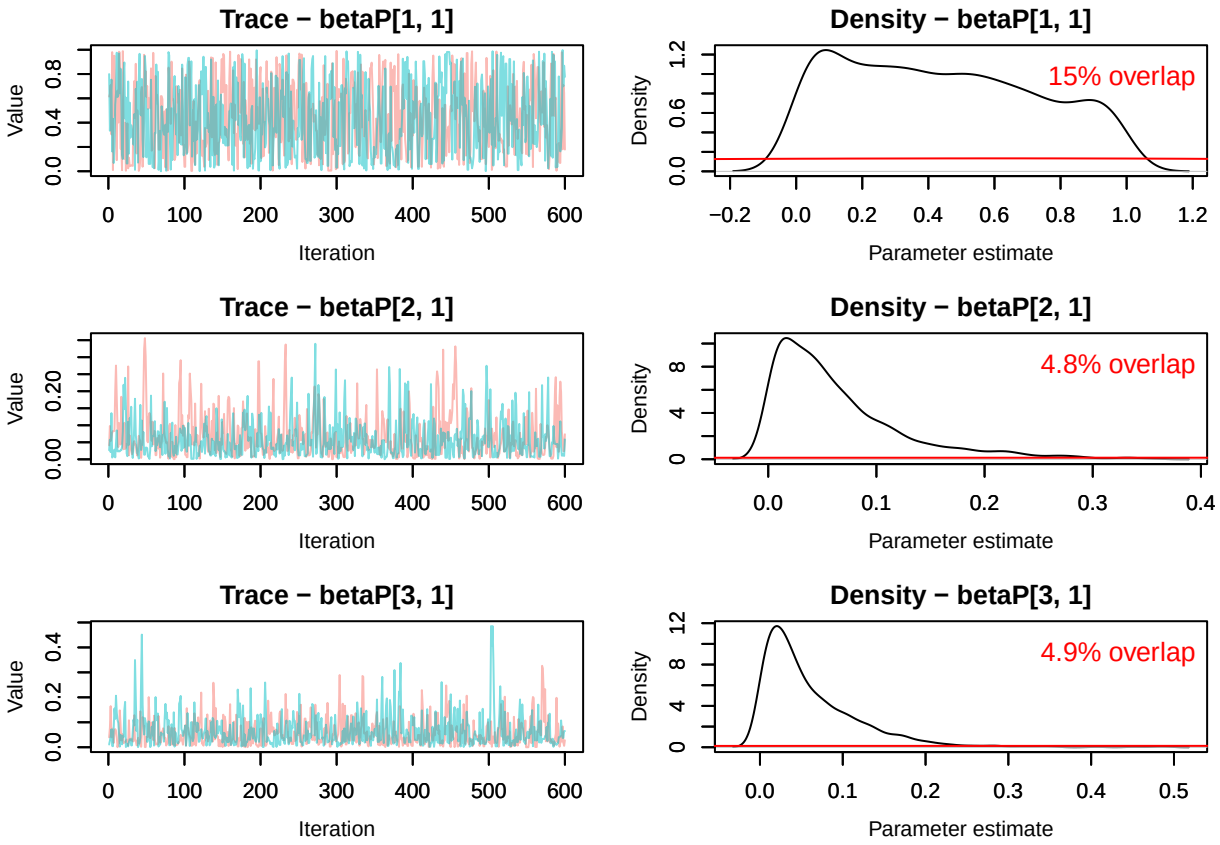

Density − betaPhi[1, 2]
Density
0
2
4
7.1% overlap
0.0
0.2
0.4
0.6
0.8
Parameter estimate

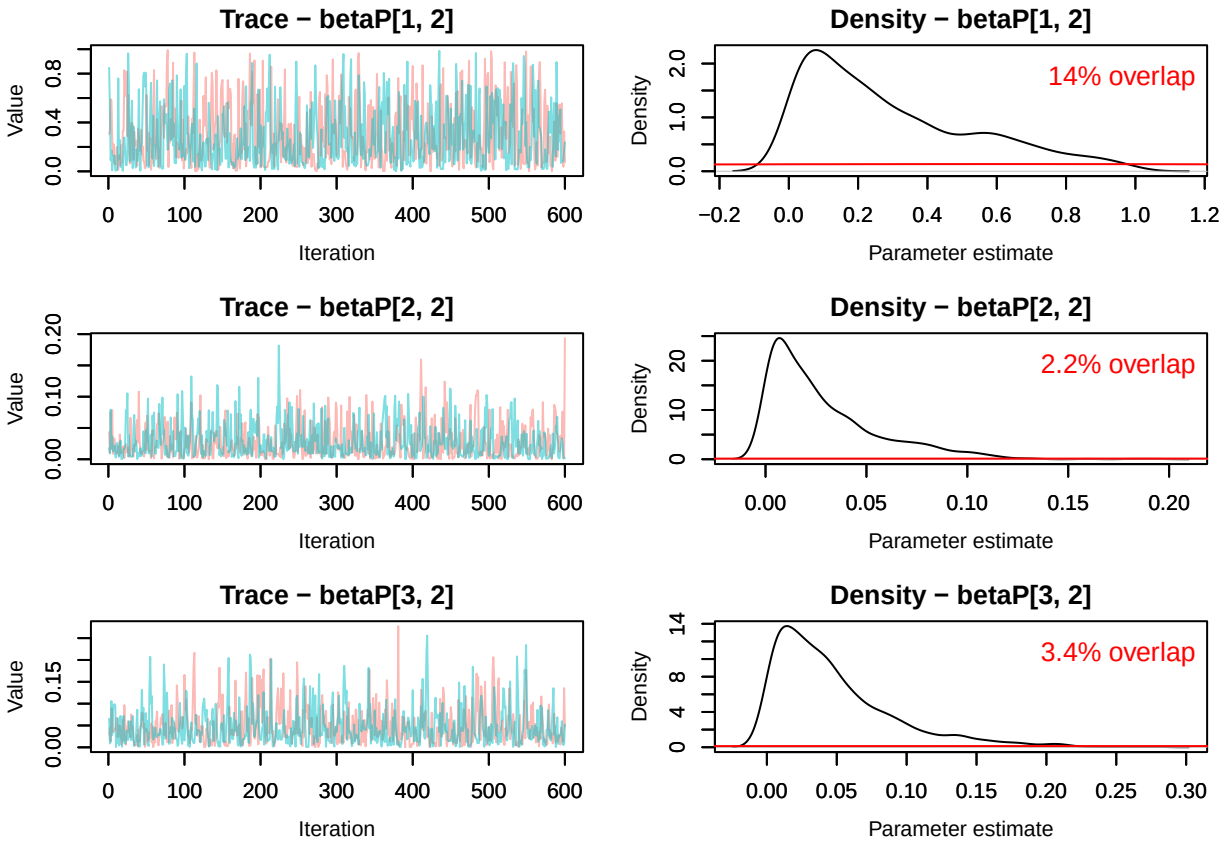

Trace − betaPhi[2, 2]
Value
0.55
0.75
0
100
200
300
400
500
600
Iteration


Density − betaPhi[2, 2]
Density
0
2
4
6
8
4% overlap
0.6
0.7
0.8
0.9
Parameter estimate


Trace − betaPhi[3, 2]
Value
0.6
0.8
0
100
200
300
400
500
600
Iteration


Density − betaPhi[3, 2]
Density
0
2
4
6
4.8% overlap
0.6
0.7
0.8
0.9
1.0
Parameter estimate


Trace − betaP[1, 1]
Value
Iteration
Density − betaP[1, 1]
15% overlap
Density
Parameter estimate
Trace − betaP[2, 1]
Value
Iteration
Density − betaP[2, 1]
4.8% overlap
Density
Parameter estimate
Trace − betaP[3, 1]
Value
Iteration
Density − betaP[3, 1]
4.9% overlap
Density
Parameter estimate

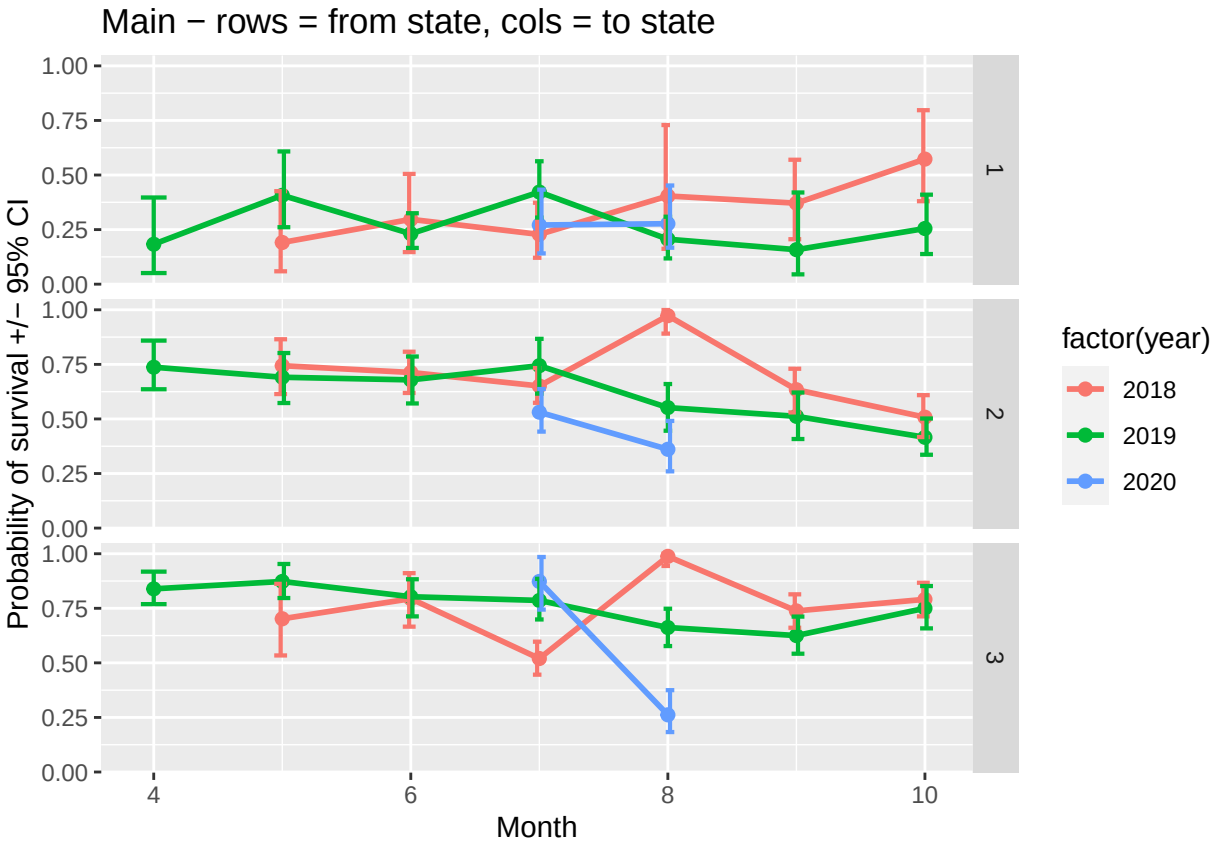


Main, phi, p and psi

```
nS <- mod_tt_main$myConstants$nStates
nT <- mod_tt_main$myConstants$T

phi_tt_main <- modSummary_tt_main %>%
  filter(substr(row.names(modSummary_tt_main), 1, 7) == "betaPhi") %>%
  add_column(data.frame(state = 1:nS, dateYM = rep(1:(nT - 1), each = nS))) %>%
  mutate(year = years[dateYM],
         month = months[dateYM],
         river = "main")

p_tt_main <- modSummary_tt_main %>%
  filter(substr(row.names(modSummary_tt_main), 1, 6) == "betaP[") %>%
  add_column(data.frame(state = 1:nS, dateYM = rep(1:(nT - 1), each = nS))) %>%
  mutate(year = years[dateYM],
         month = months[dateYM],
         river = "main")

psi_tt_main <- modSummary_tt_main %>%
  filter(substr(row.names(modSummary_tt_main), 1, 3) == "psi") %>%
  add_column(data.frame(stateFrom = 1:nS, stateTo = rep(1:nS, each = nS), dateYM = rep(1:(nT - 1), ea
  mutate(year = years[dateYM],
         month = months[dateYM],
```

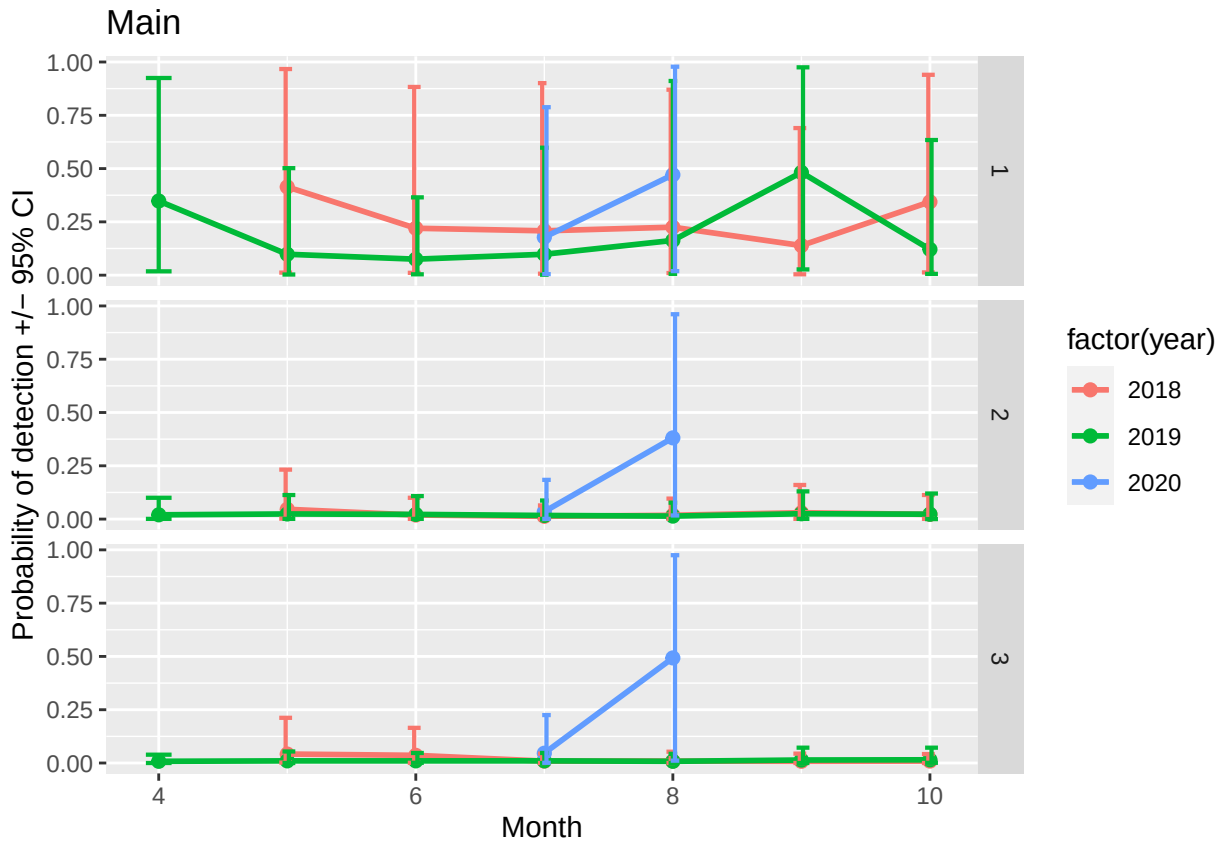
```
          river = "main")
```

```
#phi
ggplot(phi_tt_main, aes(month, med, color = factor(year))) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  geom_errorbar(aes(ymin = lo, ymax = hi),
                width = 0.2,
                position = position_dodge(0.05),
                size = 0.75) +
  ylab("Probability of survival +/- 95% CI") +
  xlab("Month") +
  ggtitle("Main - rows = from state, cols = to state") +
  facet_grid(rows = vars(state))
```

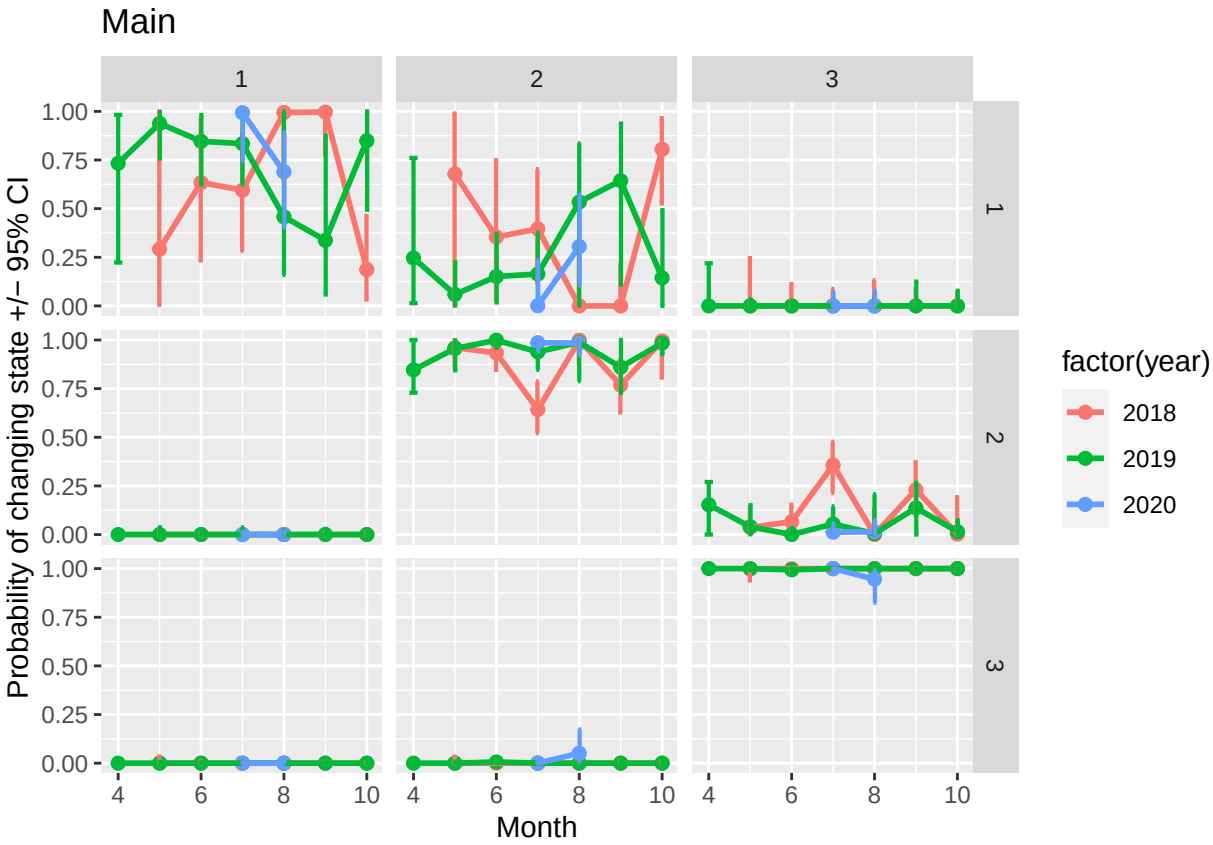



```
# p
ggplot(p_tt_main, aes(month, med, color = factor(year))) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  geom_errorbar(aes(ymin = lo, ymax = hi),
                width = 0.2,
                position = position_dodge(0.05),
                size = 0.75) +
  ylab("Probability of detection +/- 95% CI") +
```

```
  xlab("Month") +
  ggtitle("Main") +
  facet_grid(rows = vars(state))
```




```
#psi
ggplot(psi_tt_main, aes(month, med, color = factor(year))) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  geom_errorbar(aes(ymin = lo, ymax = hi),
                width = 0.2,
                position = position_dodge(0.03),
                size = 0.75) +
  ylab("Probability of changing state +/- 95% CI") +
  xlab("Month") +
  ggtitle("Main") +
  facet_grid(rows = vars(stateFrom),
             cols = vars(stateTo))
```



phiT_pT_psiT_trib

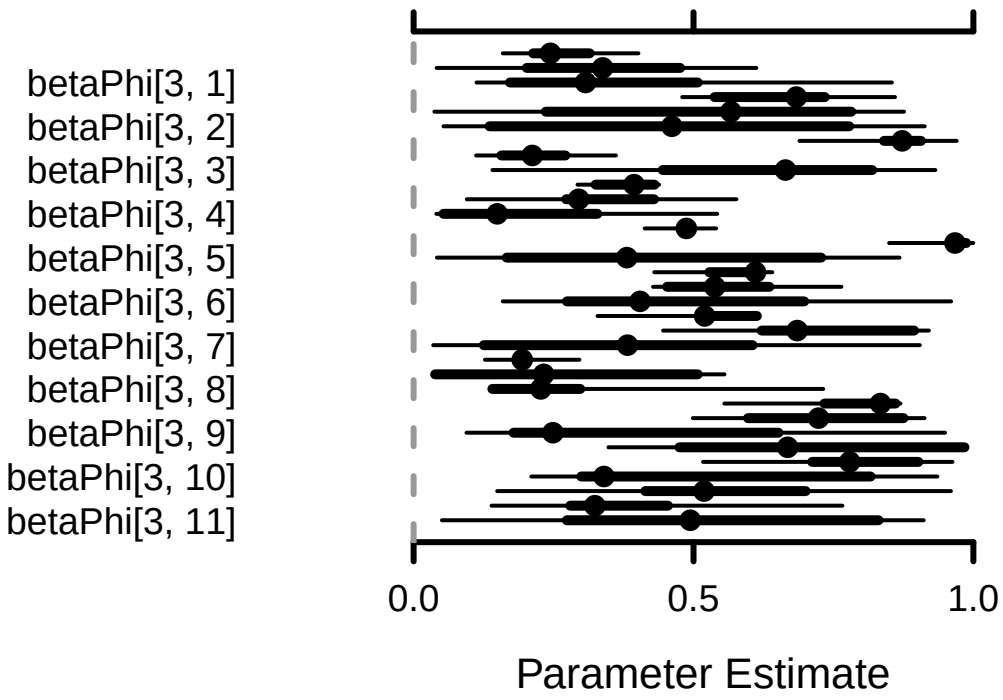
Load data 'trib' for analysis

```
#d_tt <- tar_read(target_d_trib)
eh_trib <- tar_read(target_eh_trib)

str(eh_trib$eh)
#>  num [1:288, 1:12] 1 1 1 1 1 1 1 1 1 1 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : NULL
#>   ..$ : chr [1:12] "date_2018-07" "date_2018-08" "date_2018-09" "date_2018-10" ...
#kable(eh_trib$eh[1:8,1:10])

table(paste(eh_trib$first, eh_trib$last, sep="_"))
#>
#> 1_12 2_12 3_12 4_12 5_12 6_12 7_12 8_12
#>   60    8   12   15  100   53   20   20
```

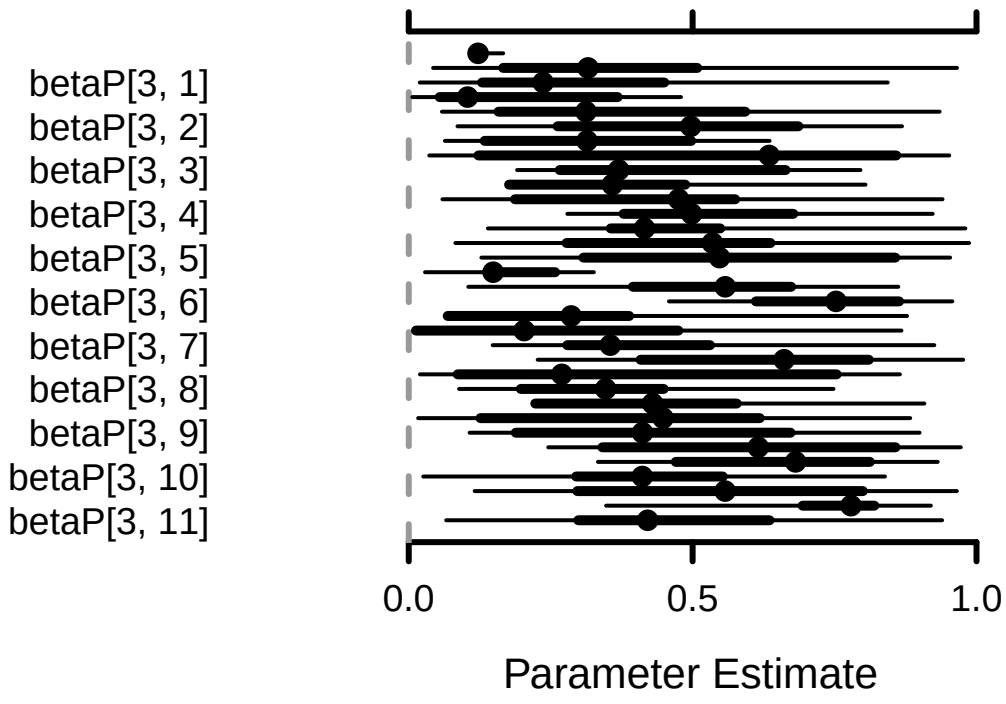
Plot trib model estimates

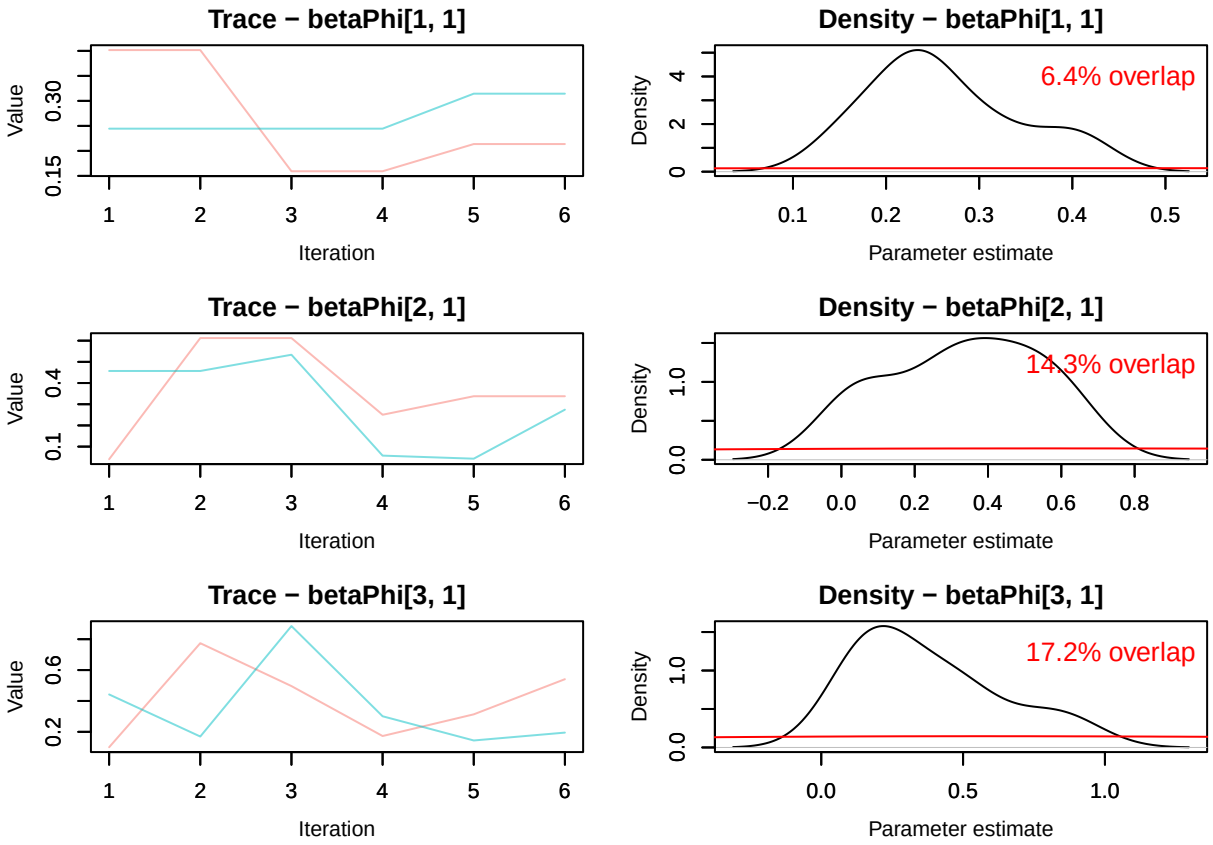
```
### Read the model run into global memory
if (tar_exist_objects(c("tt_modelOut_trib"))) {
  mod_tt_trib <- tar_read(tt_modelOut_trib)
```

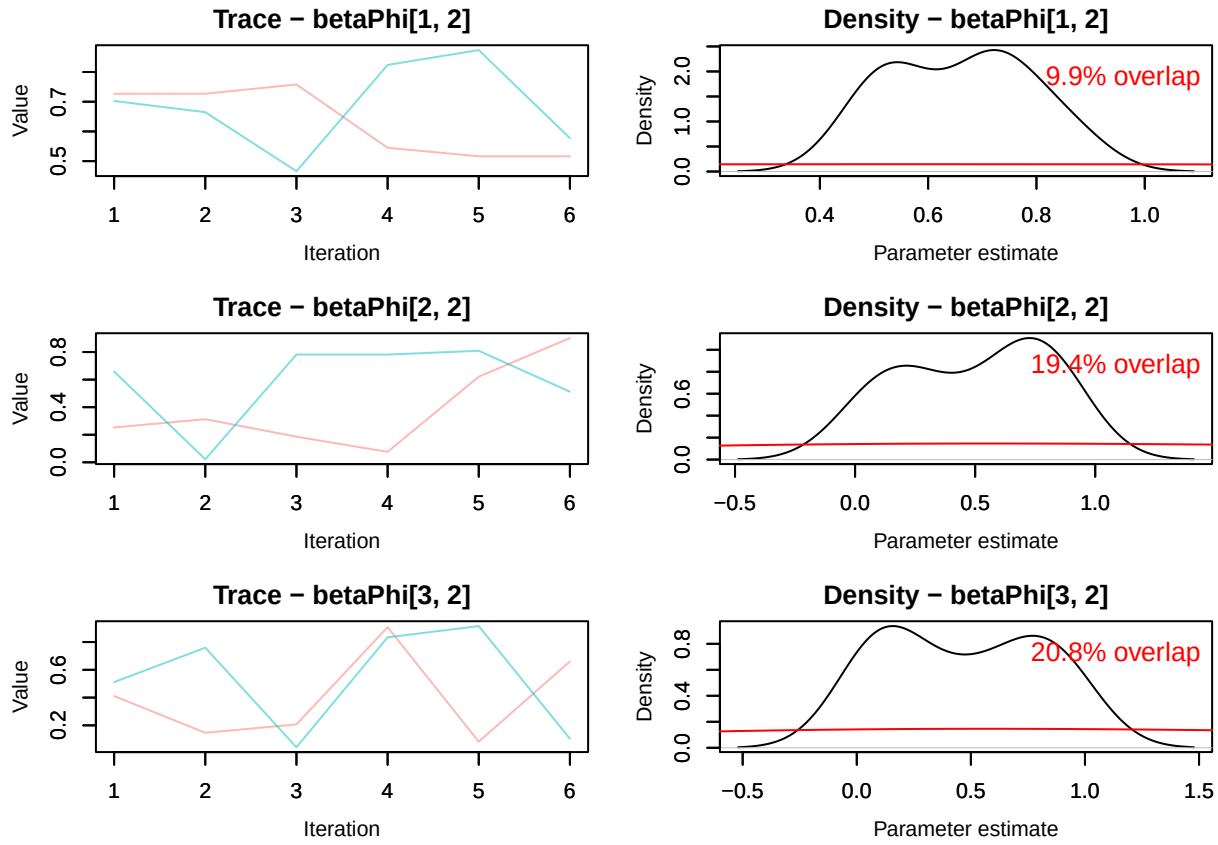
```
  MCMCplot(object = mod_tt_trib$mcmc, params = "betaPhi")
  MCMCplot(object = mod_tt_trib$mcmc, params = "betaP")

   MCMCtrace(object = mod_tt_trib$mcmc,
             ISB = FALSE,
             exact = TRUE,
             params = c("betaPhi[1, 1]", "betaPhi[2, 1]", "betaPhi[3, 1]",
                        "betaPhi[1, 2]", "betaPhi[2, 2]", "betaPhi[3, 2]",
                        "betaPhi[1, 3]", "betaPhi[2, 3]", "betaPhi[3, 3]"),
             pdf = FALSE,
             priors = priors
             )

    MCMCtrace(object = mod_tt_trib$mcmc,
           ISB = FALSE,
           exact = TRUE,
           params = c("betaP[1, 1]", "betaP[2, 1]", "betaP[3, 1]",
                      "betaP[1, 2]", "betaP[2, 2]", "betaP[3, 2]",
                      "betaP[1, 3]", "betaP[2, 3]", "betaP[3, 3]"),
           pdf = FALSE,
           priors = priors
           )

    modSummary_tt_trib <- MCMCsummary(object = mod_tt_trib$mcmc, round = 3) %>%
      rename(lo = '2.5%', med = '50%', hi = '97.5%')
}
```



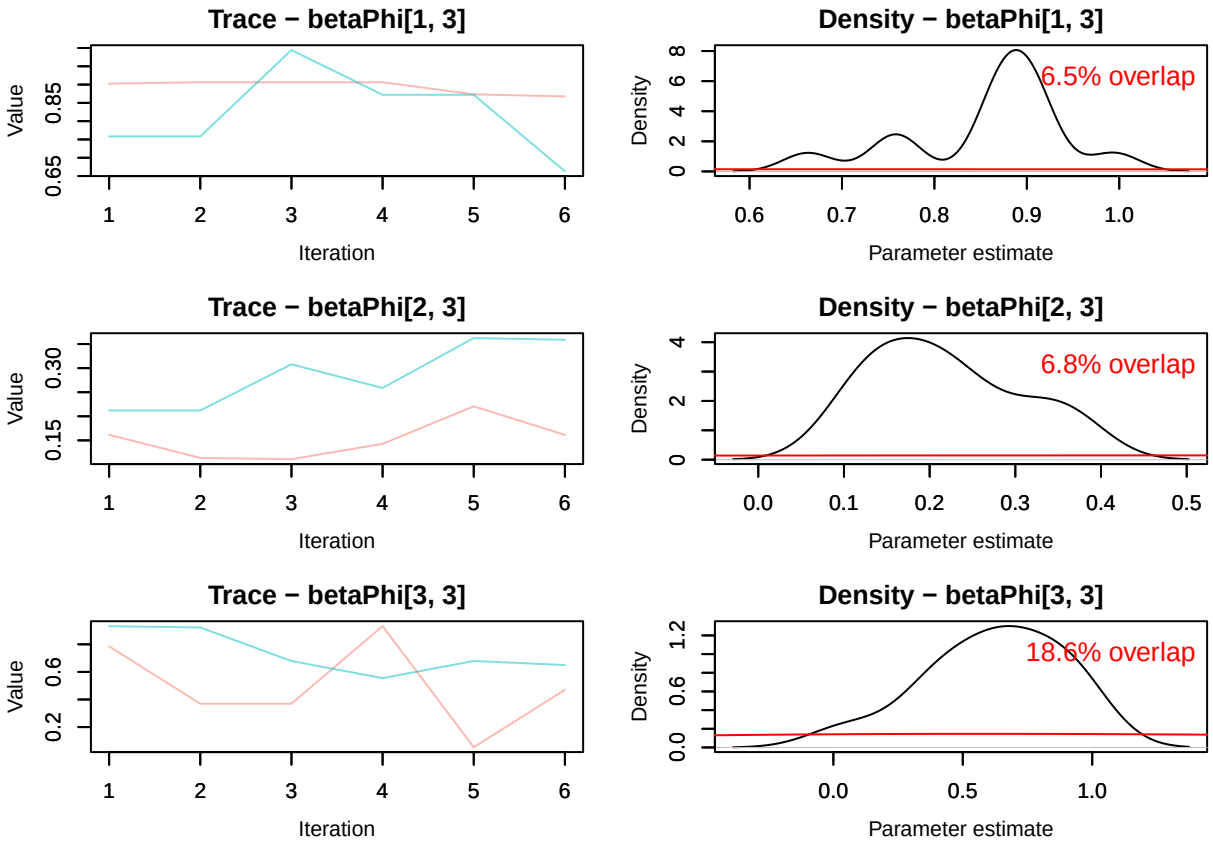

```
#> Warning in MCMCtrace(object = mod_tt_trib$mcmc, ISB = FALSE, exact = TRUE, :
#> Only one prior specified for > 1 parameter. Using a single prior for all
#> parameters.
#> Warning in MCMCtrace(object = mod_tt_trib$mcmc, ISB = FALSE, exact = TRUE, :
#> Number of samples in prior is greater than number of total or specified
#> iterations (for all chains) for specified parameter. Only last 12 iterations
#> will be used.
```

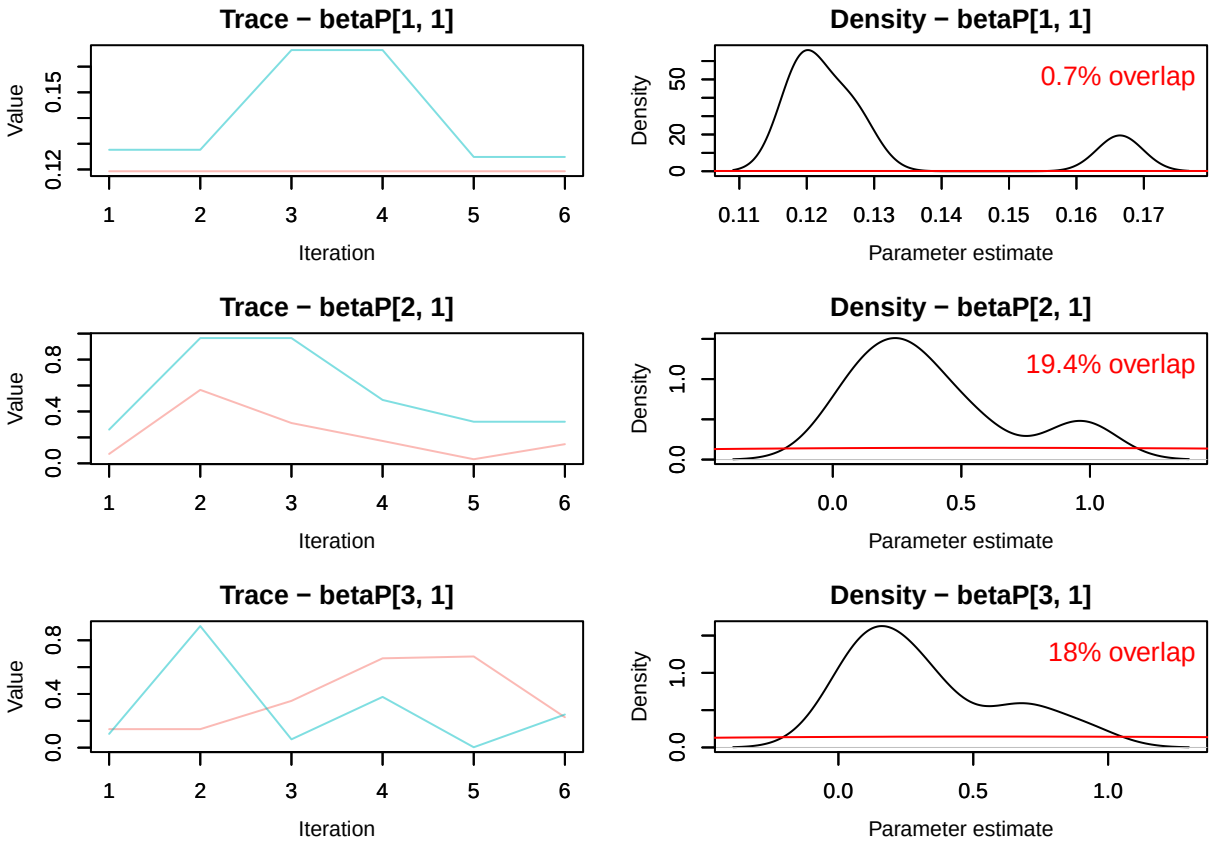

betaP[3, 1]
betaP[3, 2]
betaP[3, 3]
betaP[3, 4]
betaP[3, 5]
betaP[3, 6]
betaP[3, 7]
betaP[3, 8]
betaP[3, 9]
betaP[3, 10]
betaP[3, 11]
0.0
0.5
1.0
Parameter Estimate

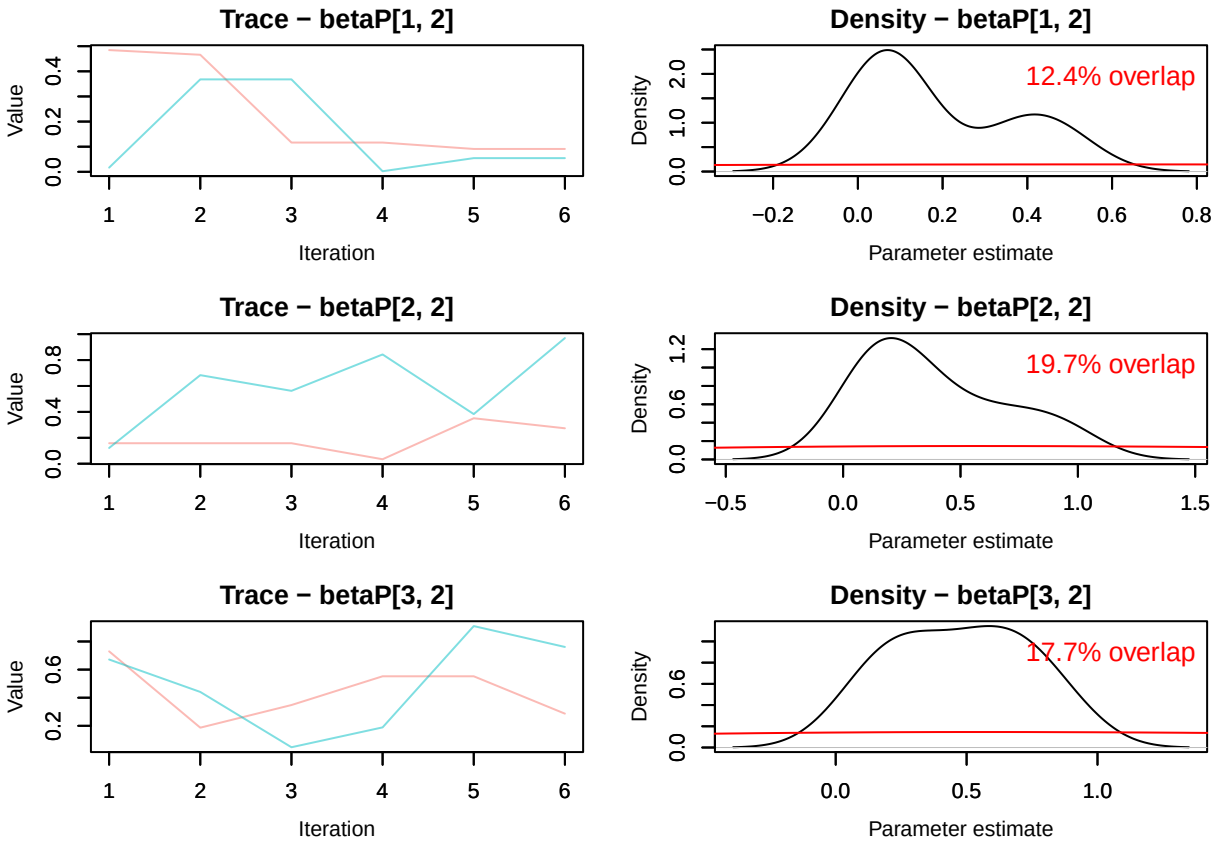

Trace − betaPhi[1, 1]
Value
0.15
0.30
1
2
3
4
5
6
Iteration
Density − betaPhi[1, 1]
Density
0
2
4
6.4% overlap
0.1
0.2
0.3
0.4
0.5
Parameter estimate
Trace − betaPhi[2, 1]
Value
0.1
0.4
Iteration
Density − betaPhi[2, 1]
Density
0.0
1.0
14.3% overlap
−0.2
0.0
0.2
0.4
0.6
0.8
Parameter estimate
Trace − betaPhi[3, 1]
Value
0.2
0.6
Iteration
Density − betaPhi[3, 1]
Density
0.0
1.0
17.2% overlap
0.0
0.5
1.0
Parameter estimate

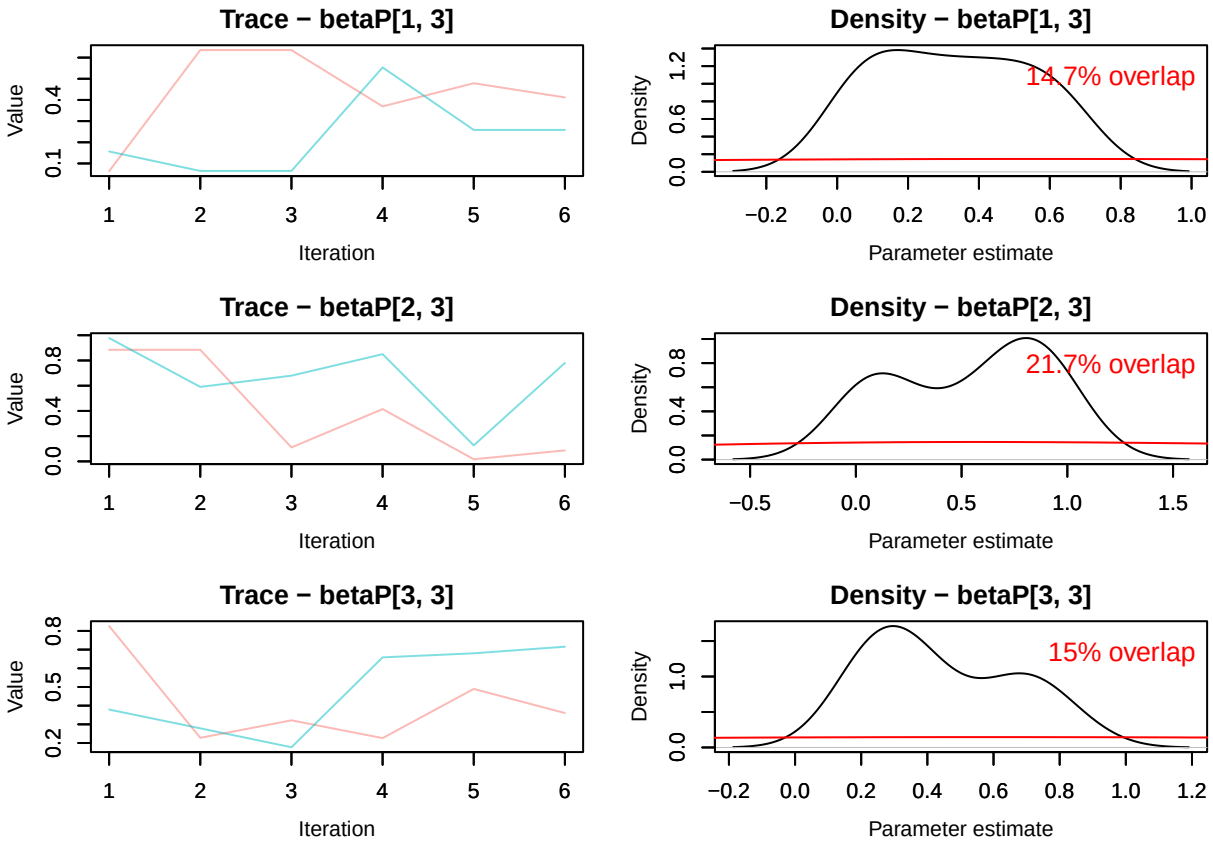



```
#> Warning in MCMCtrace(object = mod_tt_trib$mcmc, ISB = FALSE, exact = TRUE, :
#> Only one prior specified for > 1 parameter. Using a single prior for all
#> parameters.
#> Warning in MCMCtrace(object = mod_tt_trib$mcmc, ISB = FALSE, exact = TRUE, :
#> Number of samples in prior is greater than number of total or specified
#> iterations (for all chains) for specified parameter. Only last 12 iterations
#> will be used.
```


Trace − betaPhi[1, 3]
Value
0.65
0.85
1
2
3
4
5
6
Iteration
Density − betaPhi[1, 3]
Density
0
2
4
6
8
6.5% overlap
0.6
0.7
0.8
0.9
1.0
Parameter estimate
Trace − betaPhi[2, 3]
Value
0.15
0.30
Iteration
Density − betaPhi[2, 3]
Density
0
2
4
6.8% overlap
0.0
0.1
0.2
0.3
0.4
0.5
Parameter estimate
Trace − betaPhi[3, 3]
Value
0.2
0.6
Iteration
Density − betaPhi[3, 3]
Density
0.0
0.6
1.2
18.6% overlap
0.0
0.5
1.0
Parameter estimate

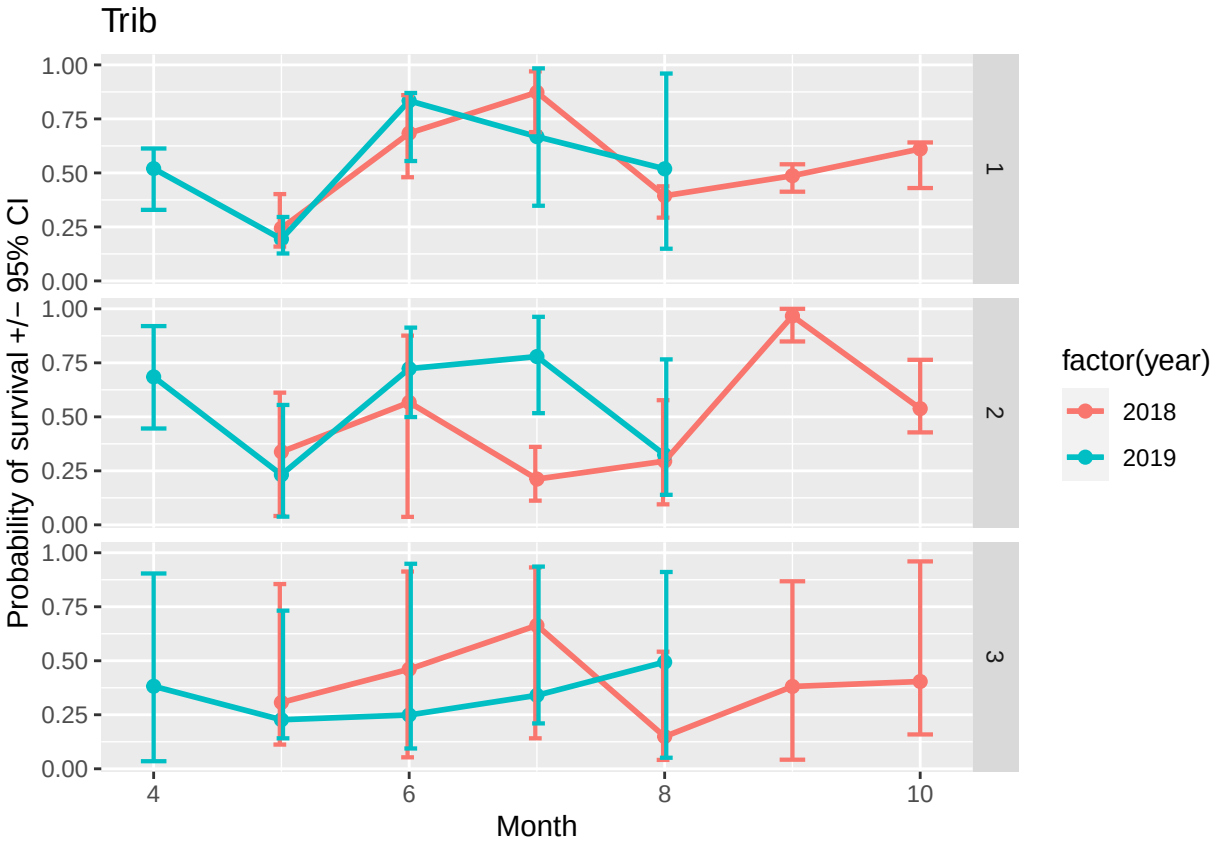

Trace − betaP[1, 1]
Value
0.12
0.15
1
2
3
4
5
6
Iteration
Density − betaP[1, 1]
Density
0
20
50
0.7% overlap
0.11
0.12
0.13
0.14
0.15
0.16
0.17
Parameter estimate
Trace − betaP[2, 1]
Value
0.0
0.4
0.8
Iteration
Density − betaP[2, 1]
Density
0.0
1.0
19.4% overlap
0.0
0.5
1.0
Parameter estimate
Trace − betaP[3, 1]
Value
Iteration
Density − betaP[3, 1]
Density
18% overlap
Parameter estimate


Trace − betaP[1, 2]
Value
Iteration
Density − betaP[1, 2]
Density
Parameter estimate
12.4% overlap
Trace − betaP[2, 2]
Value
Iteration
Density − betaP[2, 2]
Density
Parameter estimate
19.7% overlap
Trace − betaP[3, 2]
Value
Iteration
Density − betaP[3, 2]
Density
Parameter estimate
17.7% overlap

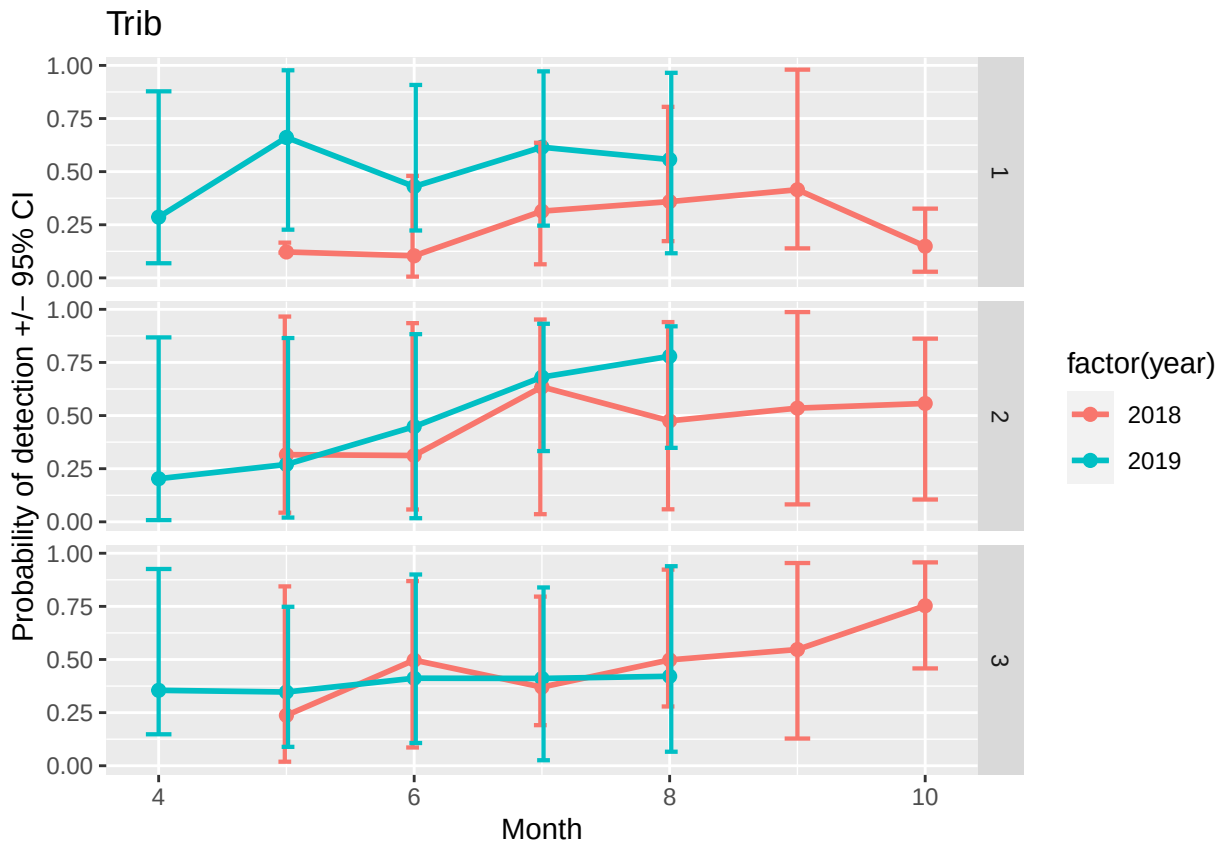



```
nS <- mod_tt_trib$myConstants$nStates
nT <- mod_tt_trib$myConstants$T

phi_tt_trib <- modSummary_tt_trib %>%
  filter(substr(row.names(modSummary_tt_trib), 1, 7) == "betaPhi") %>%
  add_column(data.frame(state = 1:nS, dateYM = rep(1:(nT - 1), each = nS))) %>%
  mutate(year = years[dateYM],
         month = months[dateYM],
         river = "trib")

p_tt_trib <- modSummary_tt_trib %>%
  filter(substr(row.names(modSummary_tt_trib), 1, 6) == "betaP[") %>%
  add_column(data.frame(state = 1:nS, dateYM = rep(1:(nT - 1), each = nS))) %>%
  mutate(year = years[dateYM],
         month = months[dateYM],
         river = "trib")

psi_tt_trib <- modSummary_tt_trib %>%
  filter(substr(row.names(modSummary_tt_trib), 1, 3) == "psi") %>%
  add_column(data.frame(stateFrom = 1:nS, stateTo = rep(1:nS, each = nS), dateYM = rep(1:(nT - 1), ea
  mutate(year = years[dateYM],
         month = months[dateYM],
         river = "trib")
```

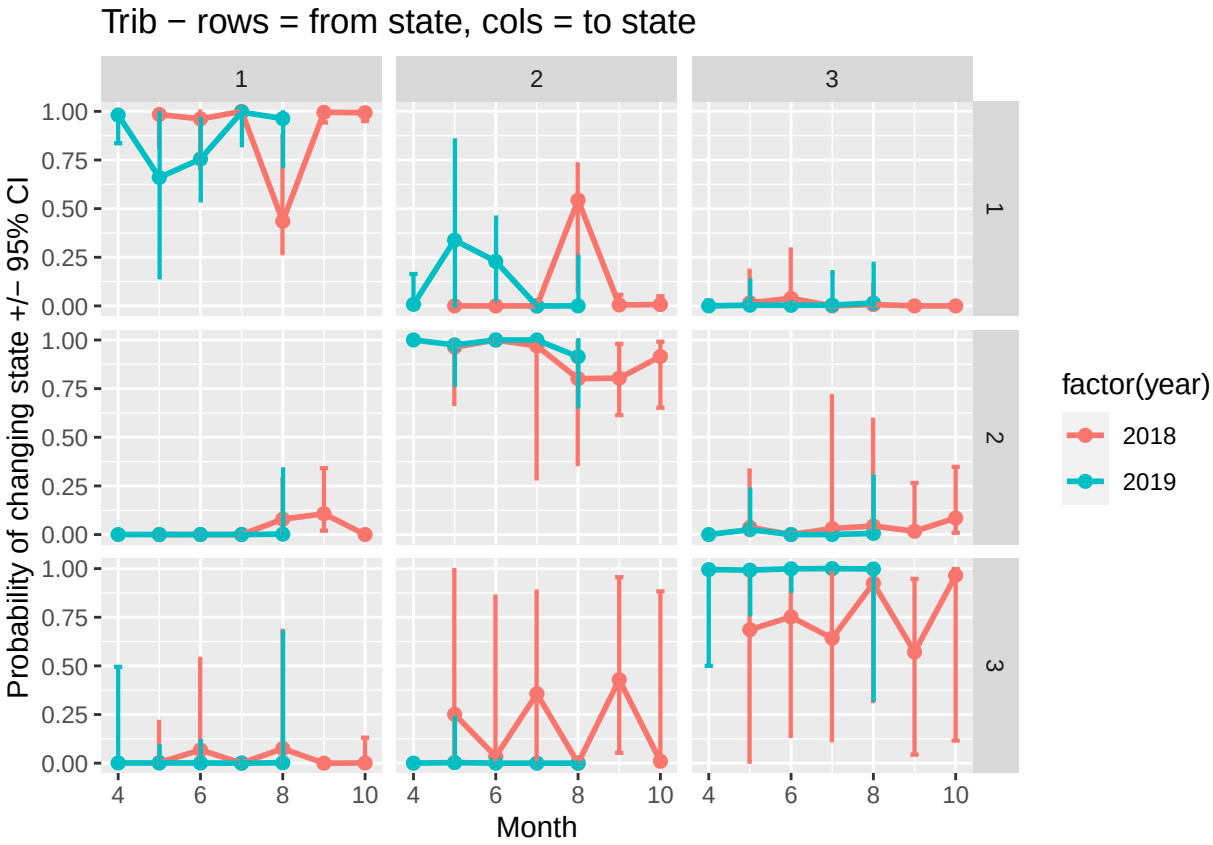
```
#phi
ggplot(phi_tt_trib, aes(month, med, color = factor(year))) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  geom_errorbar(aes(ymin = lo, ymax = hi),
                width = 0.2,
                position = position_dodge(0.05),
                size = 0.75) +
  ylab("Probability of survival +/- 95% CI") +
  xlab("Month") +
  ggtitle("Trib") +
  facet_grid(rows = vars(state))
```




```
# p
ggplot(p_tt_trib, aes(month, med, color = factor(year))) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  geom_errorbar(aes(ymin = lo, ymax = hi),
                width = 0.2,
                position = position_dodge(0.05),
                size = 0.75) +
  ylab("Probability of detection +/- 95% CI") +
  xlab("Month") +
  ggtitle("Trib") +
  facet_grid(rows = vars(state))
```



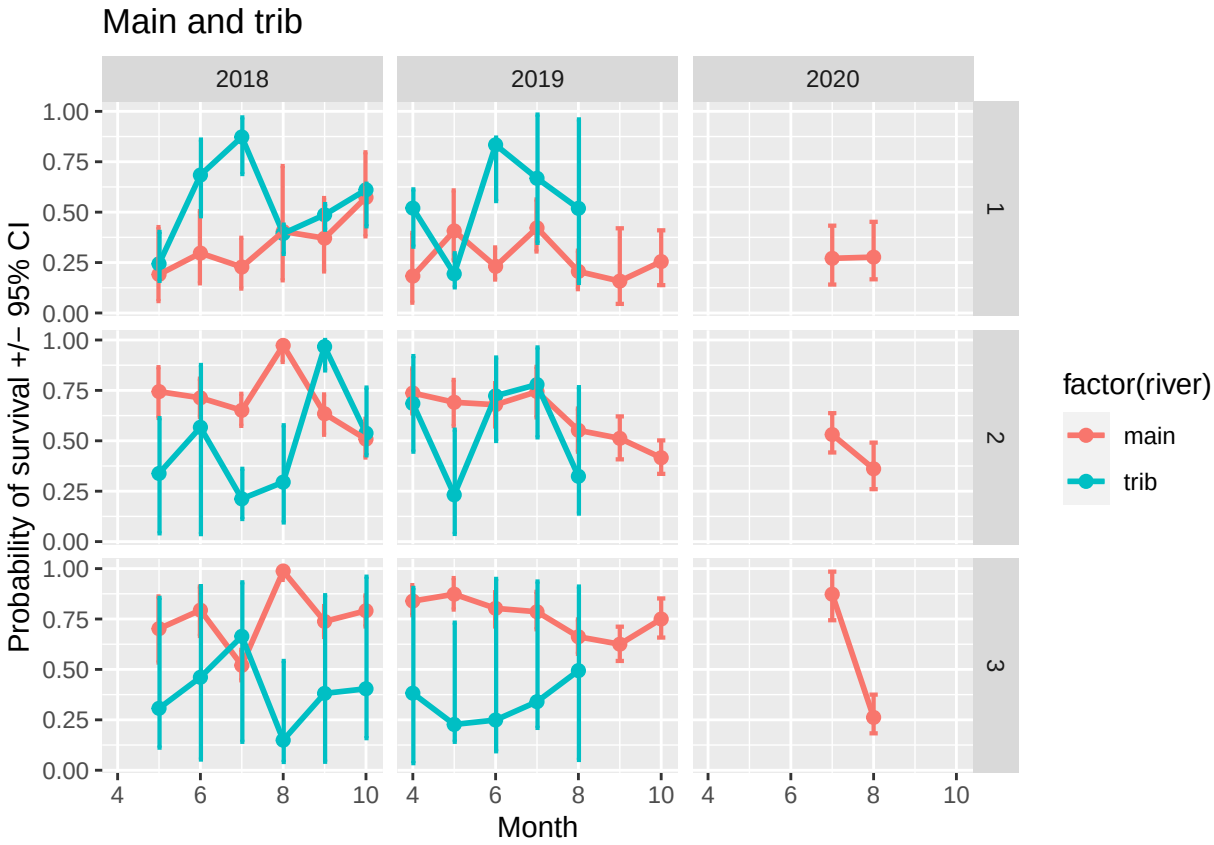

```
#psi
ggplot(psi_tt_trib, aes(month, med, color = factor(year))) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  geom_errorbar(aes(ymin = lo, ymax = hi),
                width = 0.2,
                position = position_dodge(0.03),
                size = 0.75) +
  ylab("Probability of changing state +/- 95% CI") +
  xlab("Month") +
  ggtitle("Trib - rows = from state, cols = to state") +
  facet_grid(rows = vars(stateFrom),
             cols = vars(stateTo))
```



Combine main and trib estimates

```
phi_tt_mainTrib <- add_row(phi_tt_main, phi_tt_trib)
psi_tt_mainTrib <- add_row(psi_tt_main, psi_tt_trib)
```

```
ggplot(phi_tt_mainTrib, aes(month, med, color = factor(river))) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  geom_errorbar(aes(ymin = lo, ymax = hi),
                width = 0.2,
                position = position_dodge(0.05),
                size = 0.75) +
  ylab("Probability of survival +/- 95% CI") +
  xlab("Month") +
  ggtitle("Main and trib") +
  facet_grid(rows = vars(state),
             cols = vars(year))
```



Calculate means for broad comparisons

```
  # Overall means across occasions
# Main
  (
    phi_tt_main_means <- phi_tt_main %>%
    group_by(state) %>%
    summarise(meanPhi = mean(mean),
              sdPhi = sd(mean),
              meanLo = mean(lo),
              meanMed = mean(med),
              meanHi = mean(hi)) %>%
    mutate(river = "main") %>%
    ungroup()
  )
#> # A tibble: 3 x 7
#>   state meanPhi sdPhi meanLo meanMed meanHi river
#>   <int>   <dbl> <dbl>  <dbl>   <dbl>  <dbl> <chr>
#> 1     1   0.306 0.113  0.164   0.298  0.488 main
#> 2     2   0.630 0.152  0.529   0.630  0.731 main
#> 3     3   0.734 0.170  0.643   0.734  0.824 main

  (
    psi_tt_main_means <- psi_tt_main %>%
```

```
    group_by(stateFrom, stateTo) %>%
    summarise(meanPhi = mean(mean),
              sdPhi = sd(mean),
              meanLo = mean(lo),
              meanMed = mean(med),
              meanHi = mean(hi)) %>%
    mutate(river = "main") %>%
    ungroup()
    )
#> `summarise()` has grouped output by 'stateFrom'. You can override using the
#> `.groups` argument.
#> # A tibble: 9 x 8
#>   stateFrom stateTo meanPhi    sdPhi   meanLo meanMed meanHi river
#>       <int>   <int>   <dbl>    <dbl>    <dbl>   <dbl>  <dbl> <chr>
#> 1         1       1 0.692   0.238    0.404     0.692  0.933  main
#> 2         1       2 0.299   0.236    0.0639    0.299  0.582  main
#> 3         1       3 0.00973 0.00660  0         0      0.0953 main
#> 4         2       1 0.0022  0.00142  0         0      0.0201 main
#> 5         2       2 0.914   0.0970   0.815     0.923  0.977  main
#> 6         2       3 0.0834  0.0973   0.0229    0.0747 0.183  main
#> 7         3       1 0.00133 0.000900 0         0      0.0107 main
#> 8         3       2 0.00573 0.0154   0.000533  0.0038 0.0238 main
#> 9         3       3 0.993   0.0157   0.971     0.996  0.999  main

# Trib
  (
    phi_tt_trib_means <- phi_tt_trib %>%
    group_by(state) %>%
    summarise(meanPhi = mean(mean),
              sdPhi = sd(mean),
              meanLo = mean(lo),
              meanMed = mean(med),
              meanHi = mean(hi)) %>%
    mutate(river = "trib") %>%
    ungroup()
    )
#> # A tibble: 3 x 7
#>   state meanPhi sdPhi meanLo meanMed meanHi river
#>   <int>   <dbl> <dbl>  <dbl>   <dbl>  <dbl> <chr>
#> 1     1   0.543 0.204 0.361    0.548  0.689 trib
#> 2     2   0.525 0.242 0.291    0.515  0.755 trib
#> 3     3   0.433 0.110 0.0981   0.369  0.864 trib

  (psi_tt_trib_means <- psi_tt_trib %>%
    group_by(stateFrom, stateTo) %>%
    summarise(meanPhi = mean(mean),
              sdPhi = sd(mean),
              meanLo = mean(lo),
              meanMed = mean(med),
              meanHi = mean(hi)) %>%
    mutate(river = "trib") %>%
    ungroup()
    )
```

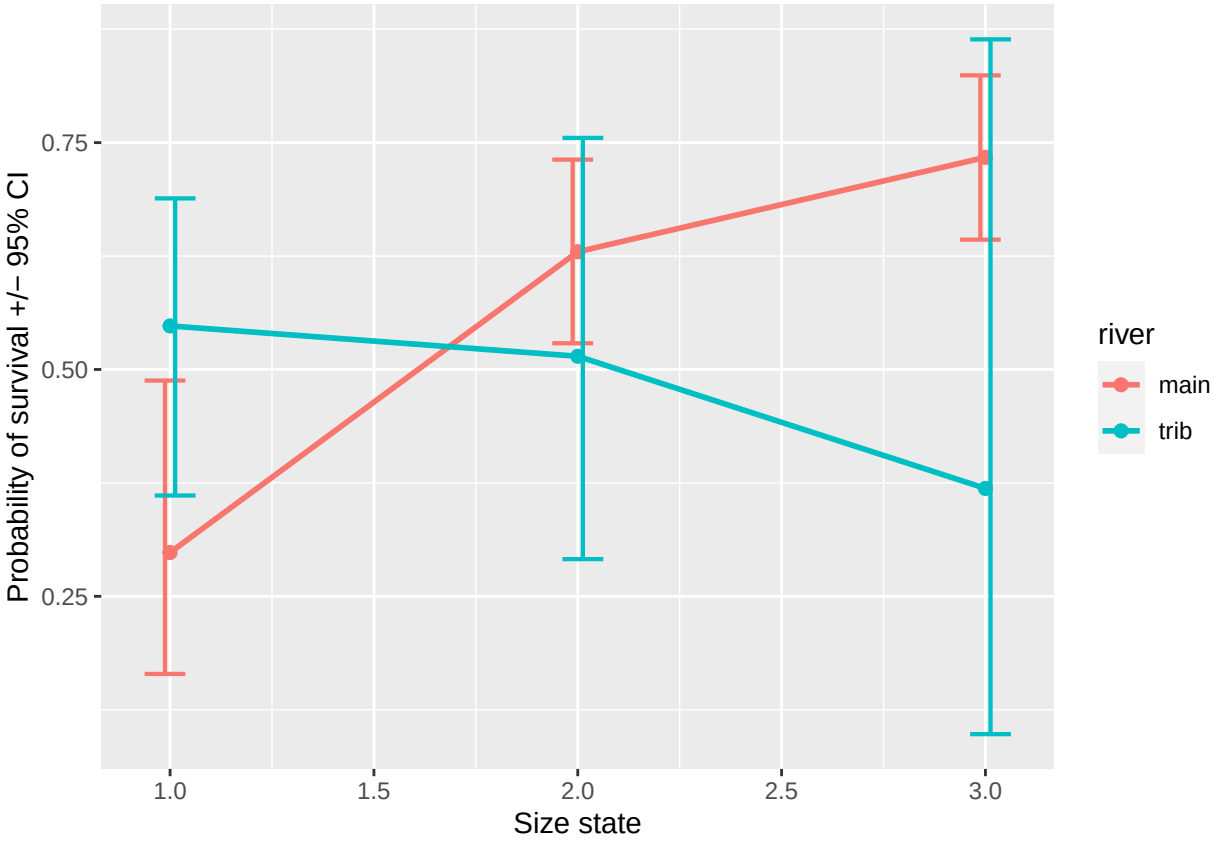
```
#> `summarise()` has grouped output by 'stateFrom'. You can override using the
#> `.groups` argument.
#> # A tibble: 9 x 8
#>   stateFrom stateTo meanPhi  sdPhi   meanLo meanMed meanHi river
#>       <int>   <int>   <dbl>  <dbl>    <dbl>   <dbl>  <dbl> <chr>
#> 1         1       1  0.870  0.168  0.706     0.884  0.984  trib
#> 2         1       2  0.107  0.170  0.0103    0.103  0.232  trib
#> 3         1       3  0.0235 0.0263 0.000273  0.008  0.105  trib
#> 4         2       1  0.0269 0.0488 0.00191   0.0171 0.0879 trib
#> 5         2       2  0.908  0.0860 0.728     0.940  0.996  trib
#> 6         2       3  0.0651 0.0613 0.00173   0.0223 0.252  trib
#> 7         3       1  0.0577 0.0688 0.000182  0.0138 0.266  trib
#> 8         3       2  0.147  0.183  0.00664   0.0985 0.437  trib
#> 9         3       3  0.795  0.172  0.384     0.866  0.988  trib
```

Combine mean main and trib estimates

```
phi_tt_mainTrib_means <- add_row(phi_tt_main_means, phi_tt_trib_means)
psi_tt_mainTrib_means <- add_row(psi_tt_main_means, psi_tt_trib_means)
```

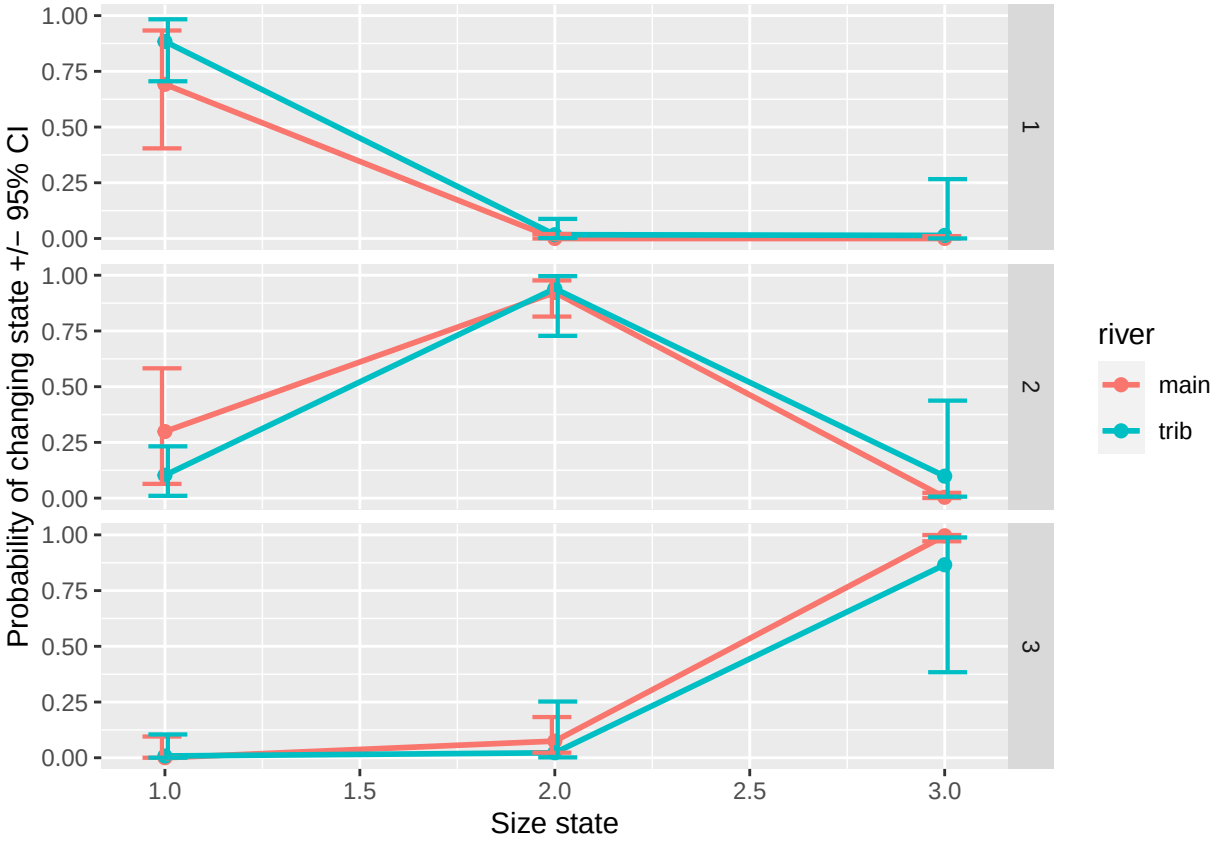
Probability of survival for each size state

```
ggplot(phi_tt_mainTrib_means, aes(state, meanMed, color = river)) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  geom_errorbar(aes(ymin = meanLo, ymax = meanHi),
                width = 0.2,
                position = position_dodge(0.05),
                size = 0.75) +
  ylab("Probability of survival +/- 95% CI") +
  xlab("Size state")
```



Probability of changing state. 'From' size states are on the x-axis, 'to' size states are in the rows of the facets

```
ggplot(psi_tt_mainTrib_means, aes(stateFrom, meanMed, color = river)) +
  geom_point(size = 2) +
  geom_line(size = 1) +
  geom_errorbar(aes(ymin = meanLo, ymax = meanHi),
                width = 0.2,
                position = position_dodge(0.03),
                size = 0.75) +
  ylab("Probability of changing state +/- 95% CI") +
  xlab("Size state") +
  facet_grid(rows = vars(stateTo))
```



Matrix model

```
# mMain <- matrix(
#   c()
# )
```

phiT_pT_psiT_mainTrib

Main and trib modeled together. Not using this as of 7-27-22.

```
### Read the model run into global memory
# if (tar_exist_objects(c("ttt_modelOut"))) {
#   mod_ttt <- tar_read(ttt_modelOut)
#
#   #MCMCplot(object = mod$mcmc)
#
#   modSummary_ttt <- MCMCsummary(object = mod_ttt$mcmc, round = 3)
# }
#kable(modSummary %>%
#        add_column(data.frame(year = rep(years[1:15], 2), dateYM = rep(occs[1:15], 2)) )


#  d %>%
#  summarize(unique(data.frame(dateYM, occ)))
```

```
# nS <- tar_read(ttt_nStates)
# nT <- tar_read(ttt_myConstants)$T
#
# modSummaryPhi_ttt <- modSummary_ttt %>%
#   filter(substr(row.names(modSummary), 1, 10) == "betaPhiOut") %>%
#   add_column(data.frame(state = 1:nS, dateYM = rep(1:(nT - 1), each = nS))) %>%
#   mutate(mainTrib = ifelse(state < 4, "Main", "Trib"),
#          size = paste0("Size", (state - 1) %% 3 + 1))
#
# ggplot(modSummaryPhi_ttt, aes(dateYM, mean)) +
#   geom_point() +
#   geom_line() +
#   facet_grid(mainTrib ~ size)
#
#
# # modSummaryYears <- modSummary %>%
# #   filter(substr(row.names(modSummary), 1, 3) == "betaPhiout") %>%
# #   add_column(data.frame(year = years[1:15], dateYM = occs[1:15], occ = 1:15)) %>%
# #   group_by(year) %>%
# #   mutate(maxSampPerYear = occ == max(occ))
# #
# # kable(
# #   modSummaryYears %>%
# #   group_by(year) %>%
# #   filter(!maxSampPerYear) %>%
# #   summarize(phiProd = prod(mean),
# #             dateRange = range(dateYM)) %>%
# #     as.data.frame()
# # )
#   MCMCplot(object = mod$mcmc, params = "betaPhiRiverOut")
#
#   priors <- rnorm(tar_read(ttt_runData)$nIter * tar_read(ttt_runData)$nChains, 0, 1/sqrt(.1))
#   MCMCtrace(object = mod$mcmc,
#             #ISB = FALSE,
#             #exact = TRUE,
#             params = c("betaPhiRiverOut"),
#             pdf = FALSE,
#             priors = priors
#             )
#
#     MCMCtrace(object = mod$mcmc,
#             #ISB = FALSE,
#             #exact = TRUE,
#             params = c("betaPhiOut"),
#             pdf = FALSE,
#             priors = priors
#             )
```

```
#create data frame for summarizing p results

# modSummaryYearsP <- modSummary %>%
#   filter(substr(row.names(modSummary), 1, 2) == "p[") %>%
#   add_column(data.frame(year = years[1:15], dateYM = occs[1:15], occ = 1:15)) %>%
```

```
#   group_by(year) %>%
#   mutate(maxSampPerYear = occ == max(occ))
#
# kable(
#   modSummaryYearsP %>%
#   group_by(year) %>%
#   summarize(pMean = mean(mean),
#             dateRange = range(dateYM))
#   )
```

```
# # modSummaryYearsP <- modSummary %>%
# #   filter(substr(row.names(modSummary), 1, 2) == "p[") %>%
# #   add_column(data.frame(year = years[1:15], dateYM = occs[1:15], occ = 1:15)) %>%
# #   group_by(year) %>%
# #   mutate(maxSampPerYear = occ == max(occ))
#
# modSummaryPsi_ttt <- modSummary_ttt %>%
#   filter(substr(row.names(modSummary_ttt), 1, 3) == "psi") %>%
#   add_column(data.frame(state = 1:nS, state2 = rep(1:nS, each = nS), dateYM = rep(1:(nT - 1), each =
#   mutate(mainTrib = ifelse(state < 4, "Main", "Trib"),
#          size = paste0("Size", (state - 1) %% 3 + 1))
#
# ggplot(modSummaryPsi_ttt, aes(dateYM, mean, color = factor(state2))) +
#   geom_point() +
#   facet_grid(mainTrib ~ size)
#
```