mvgam case study 2: multivariate models

In this example we will examine multivariate forecasting models using `mvgam`. Here we will access monthly search volume data from `Google Trends`, focusing on relative importances of search terms related to tick paralysis in Queensland, Australia

```r
library(mvgam)
library(tidyr)
if (!require(gtrendsR)) {
    install.packages("gtrendsR")
}

terms = c("tick bite", "tick paralysis",
    "dog tick", "paralysis tick dog")
trends <- gtrendsR::gtrends(terms, geo = "AU-QLD",
    time = "all", onlyInterest = T)
```

Google Trends modified their algorithm in 2012, so we filter the series to only include observations after this point in time

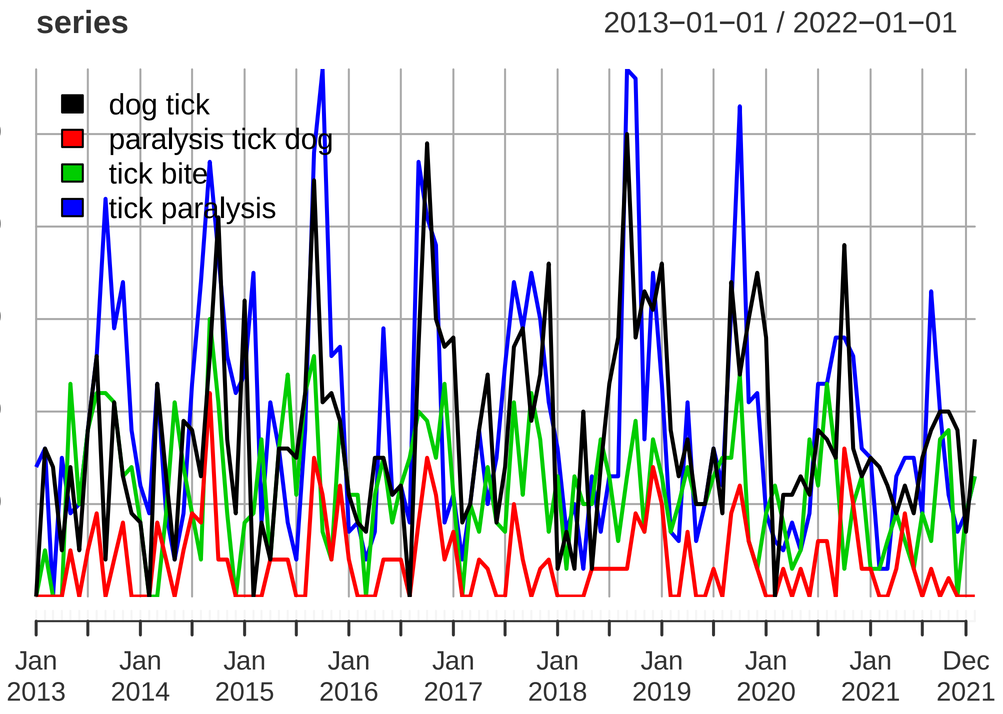
```r
gtest <- trends$interest_over_time %>% tidyr::spread(keyword,
    hits) %>% dplyr::select(-geo, -time,
    -gprop, -category) %>% dplyr::mutate(date = lubridate::ymd(date)) %>%
    dplyr::mutate(year = lubridate::year(date)) %>%
    dplyr::filter(year > 2012) %>% dplyr::select(-year)
```

Convert to an `xts` object and then to the required `mvgam` format

```r
series <- xts::xts(x = gtest[, -1], order.by = gtest$date)
trends_data <- series_to_mvgam(series, freq = 12,
    train_prop = 0.9)
```

Plot the series to see how their seasonal shapes are quite similar over time

```r
plot(series, legend.loc = "topleft")
```



Now we will fit an `mvgam` model with shared seasonality and random intercepts per series. Another advantage of using `JAGS` for sampling is that we can implement truncated likelihood functions to recognise known bounds for particular parameters. As we know that Google trends relative search volumes cannot go above `100`, we specify this as the upper bound for each series (note that these truncated likelihoods are much slower to estimate in `JAGS` so only use them if required). Here we assume the latent dynamic trends can be represented using three latent factors that each follow an `AR3` process, and we assume a Negative Binomial distrubution for the response

```
trends_mod <- mvjagam(data_train = trends_data$data_train,
    data_test = trends_data$data_test, formula = y ~
        s(series, bs = "re") + s(season,
            k = 12, m = 2, bs = "cc"), knots = list(season = c(0.5,
        12.5)), use_lv = T, trend_model = "AR3",
    n_lv = 3, family = "nb", n.burnin = 1000,
    n.iter = 1000, thin = 1, upper_bounds = rep(100,
        length(terms)), auto_update = F)
```

```
## Fitting a multivariate GAM with latent dynamic factors for the trends...
```

```
## module glm loaded
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 392
##    Unobserved stochastic nodes: 842
##    Total graph size: 13245
##
## Initializing model
```

Given that these series could potentially be following a hierarchical seasonality, we will also trial a slghtly more complex model with an extra smoothing term per series that allows its seasonal curve to deviate from the global seasonal smooth

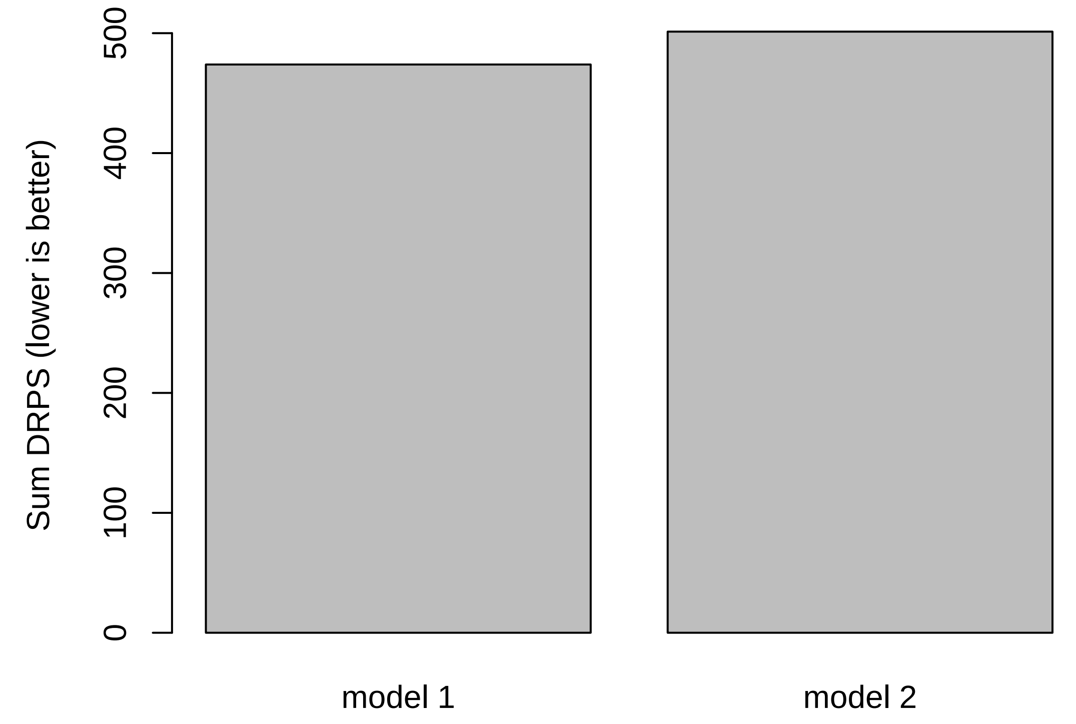
```r
trends_mod2 <- mvjagam(data_train = trends_data$data_train,
    data_test = trends_data$data_test, formula = y ~
        s(series, bs = "re") + s(season,
            k = 12, m = 2, bs = "cc") + s(season,
            by = series, k = 5, m = 1), knots = list(season = c(0.5,
        12.5)), use_lv = T, trend_model = "AR3",
    n_lv = 3, family = "nb", n.burnin = 1000,
    n.iter = 1000, thin = 1, upper_bounds = rep(100,
        length(terms)), auto_update = F)
```

```
## Fitting a multivariate GAM with latent dynamic factors for the trends...
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 392
##    Unobserved stochastic nodes: 850
##    Total graph size: 20325
##
## Initializing model
```

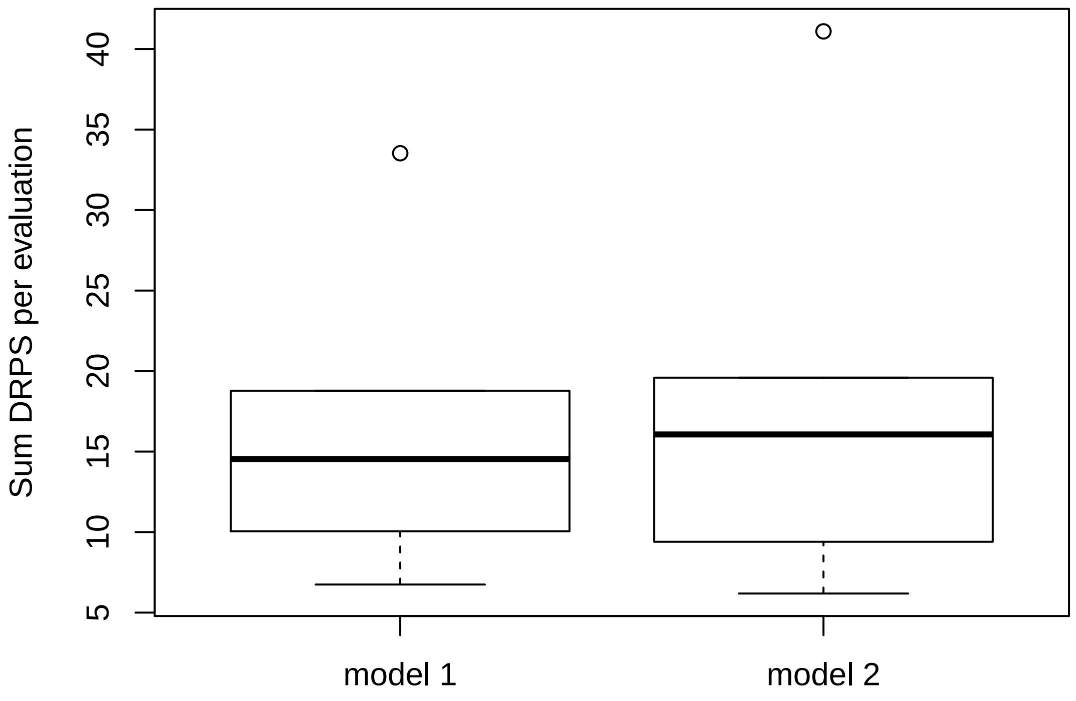
Compare the models using rolling forecast DRPS evaluation. Here we focus on near-term forecasts (`fc_horizon = 3`) when comparing model performances

```r
compare_mvgams(trends_mod, trends_mod2, fc_horizon = 3,
    n_cores = 3, n_evaluations = 10)
```

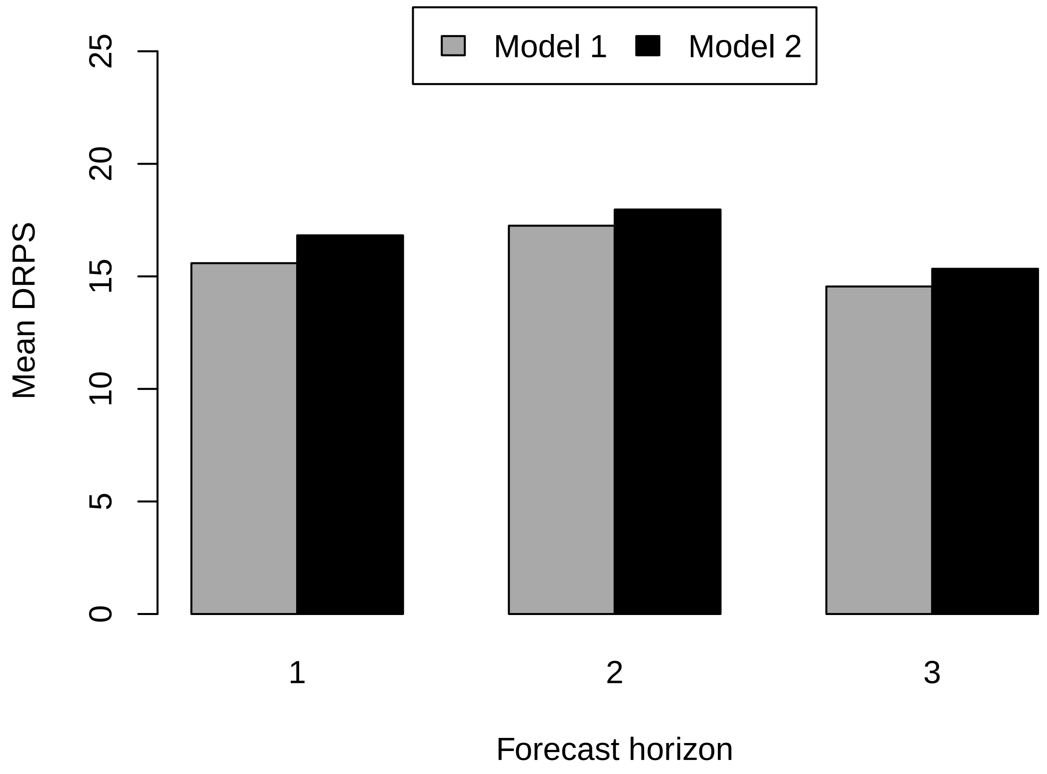
```
## DRPS summaries per model (lower is better)
##             Min.   1st Qu.   Median     Mean  3rd Qu.     Max.
## Model 1 6.743116 10.048139 13.28437 15.79580 18.78093 33.52938
## Model 2 6.184484  9.400143 15.41745 16.70978 19.59062 41.09871
##
## 90% interval coverages per model (closer to 0.9 is better)
## Model 1 0.8833333
## Model 2 0.825
```



Sum DRPS (lower is better)
0
100
200
300
400
500
model 1
model 2

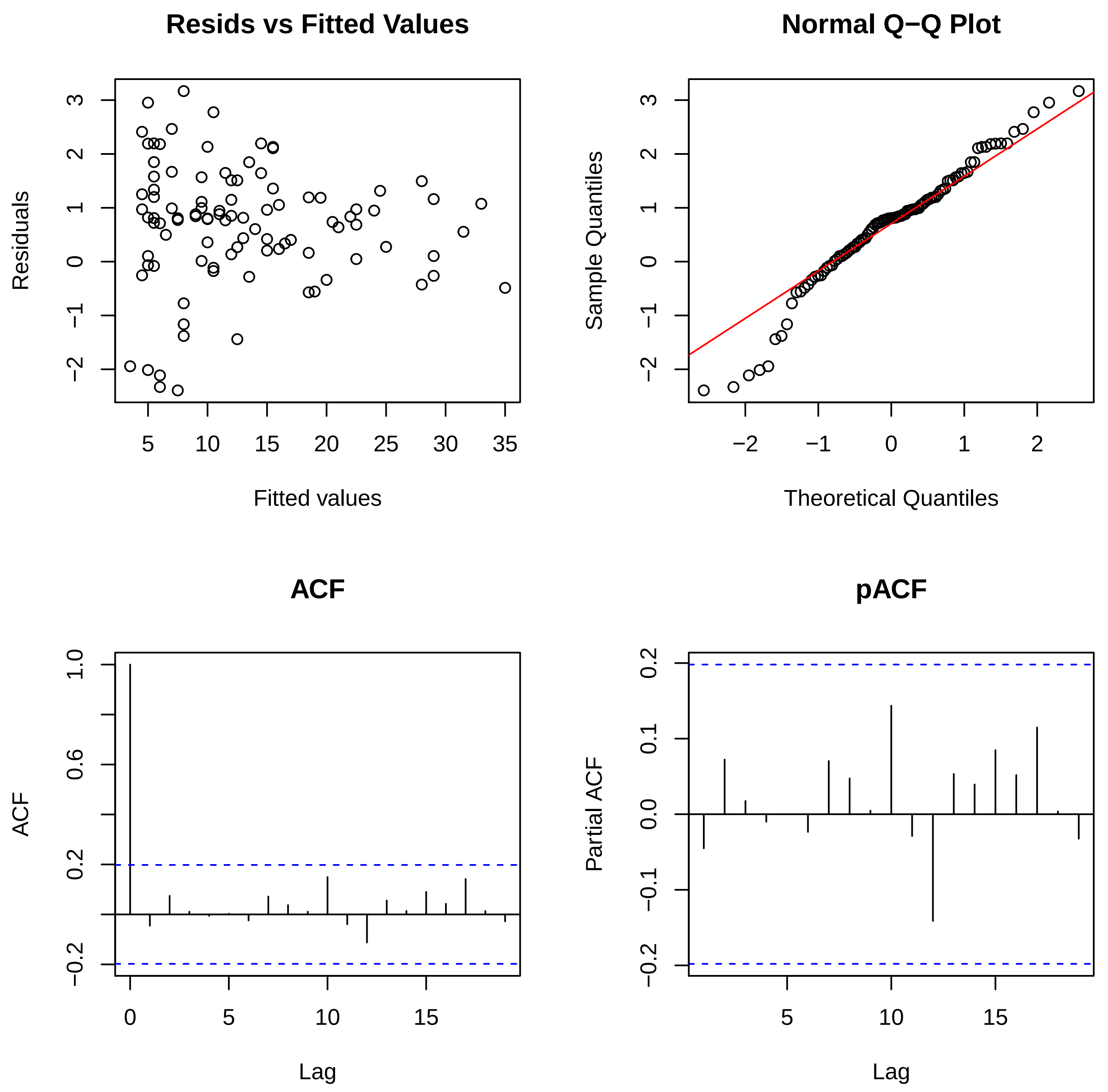


Sum DRPS per evaluation
5
10
15
20
25
30
35
40
model 1
model 2

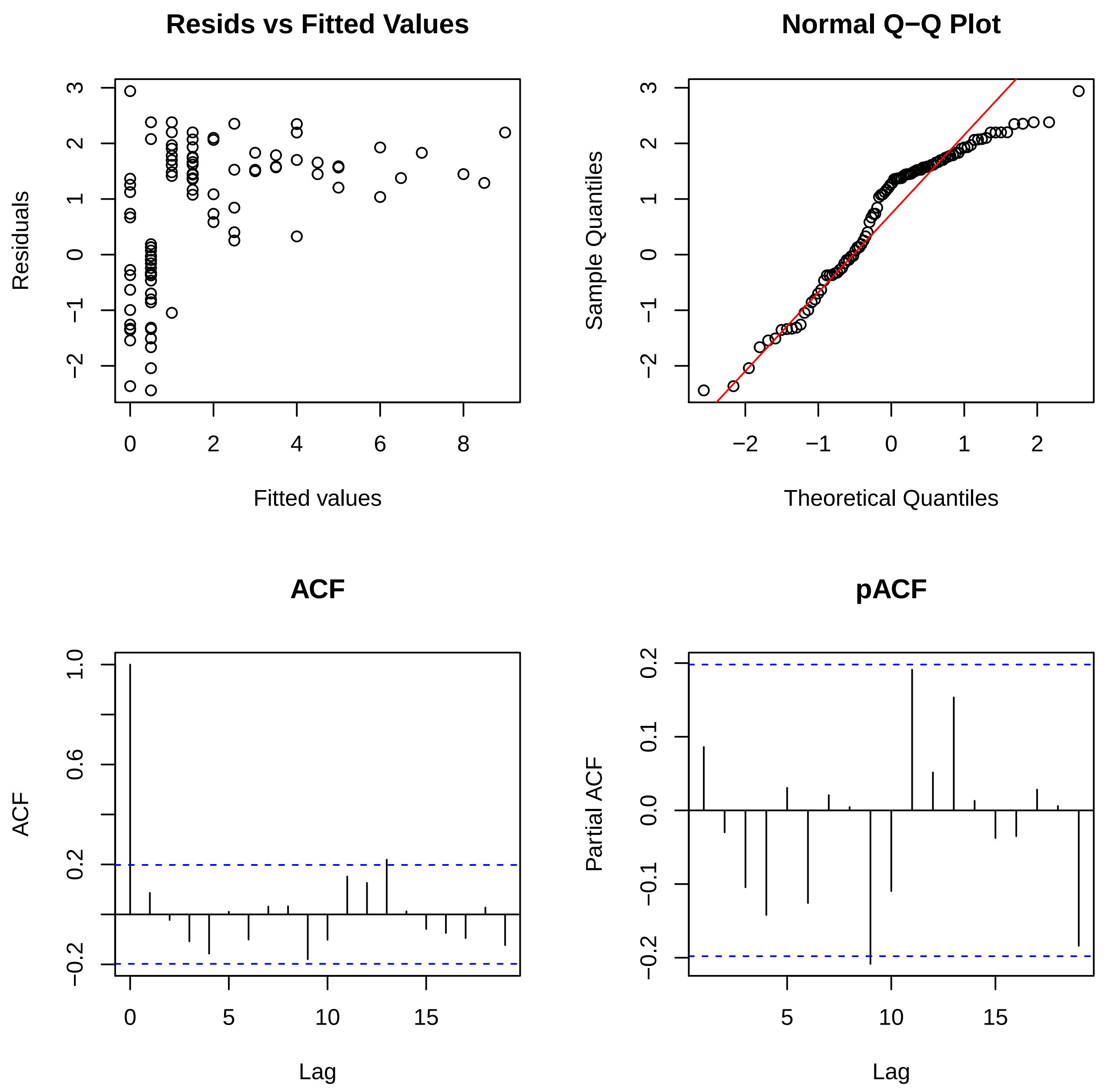


Model 1 (with shared seasonality) is slightly preferred, suggesting there is not sufficient evidence that the variation in the seasonal curves for these series is useful for forecasting. Look at Dunn-Smyth residuals for some series from this preferred model

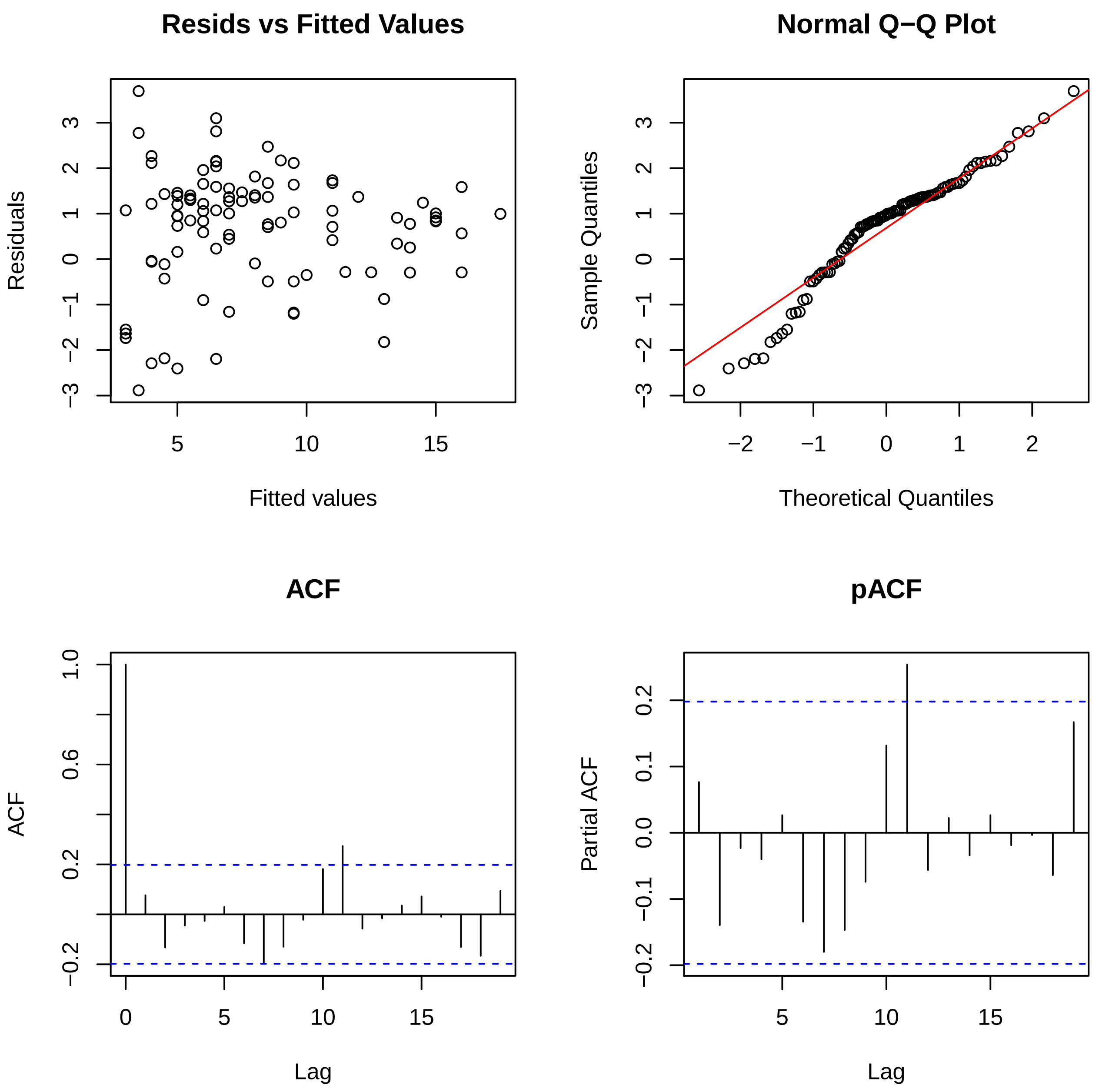
```
plot_mvgam_resids(trends_mod, 1)
```



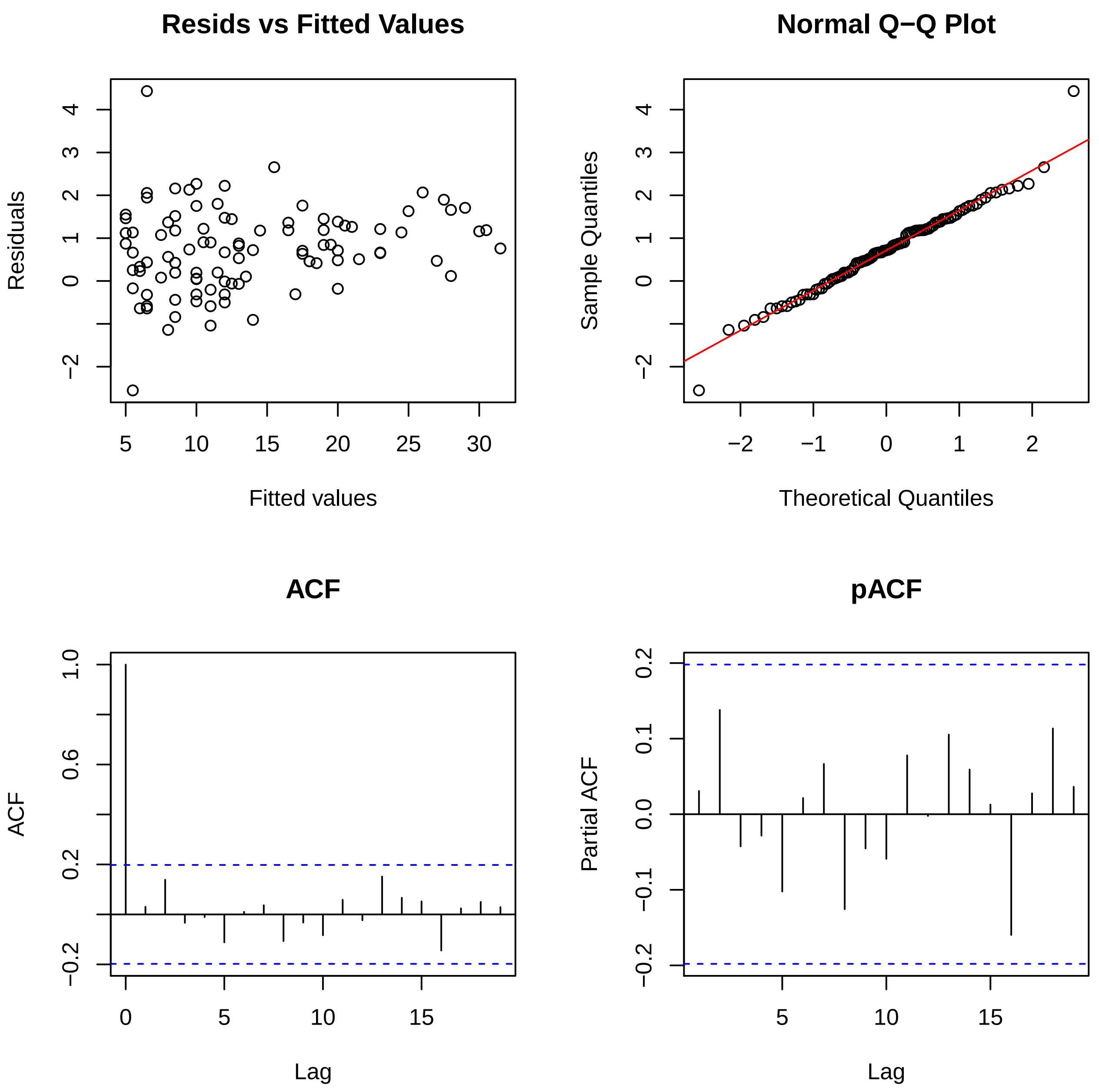
```
plot_mvgam_resids(trends_mod, 2)
```



```
plot_mvgam_resids(trends_mod, 3)
```

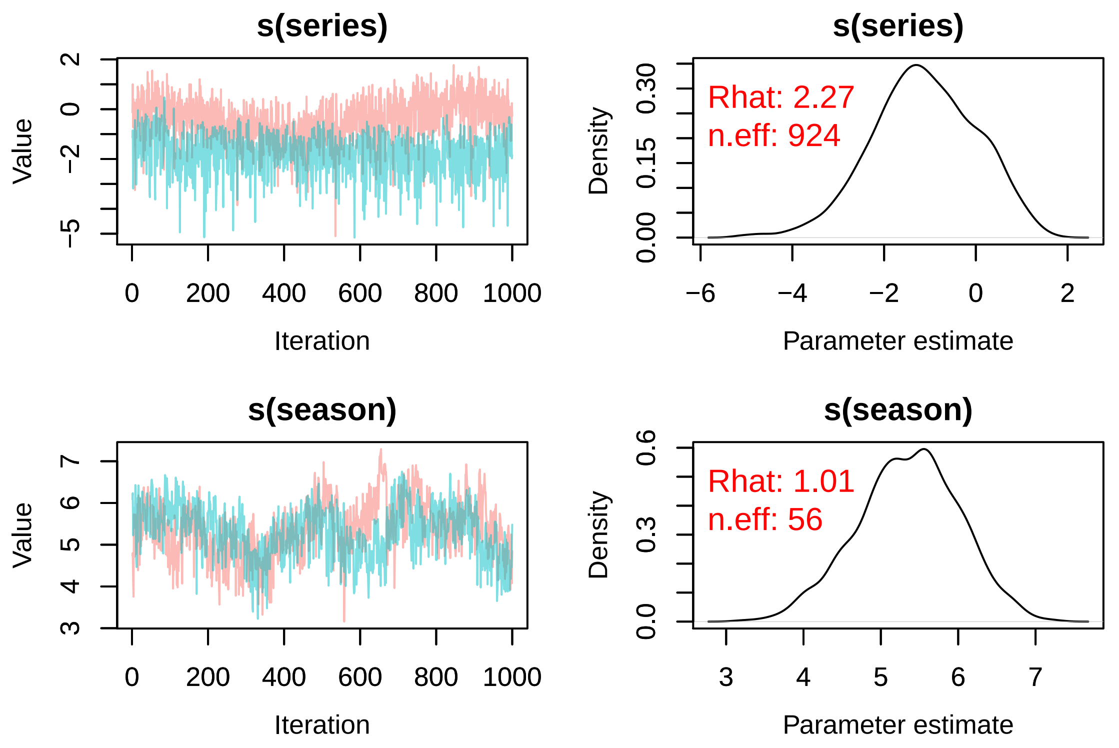


```r
plot_mvgam_resids(trends_mod, 4)
```

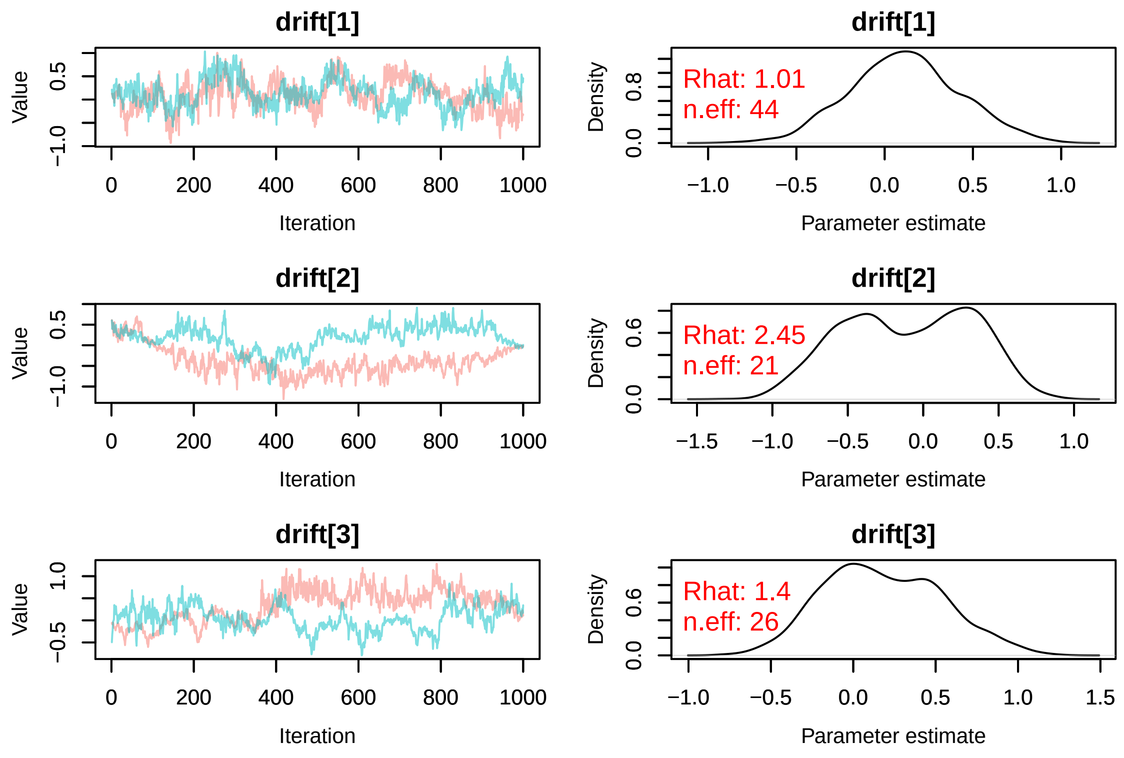


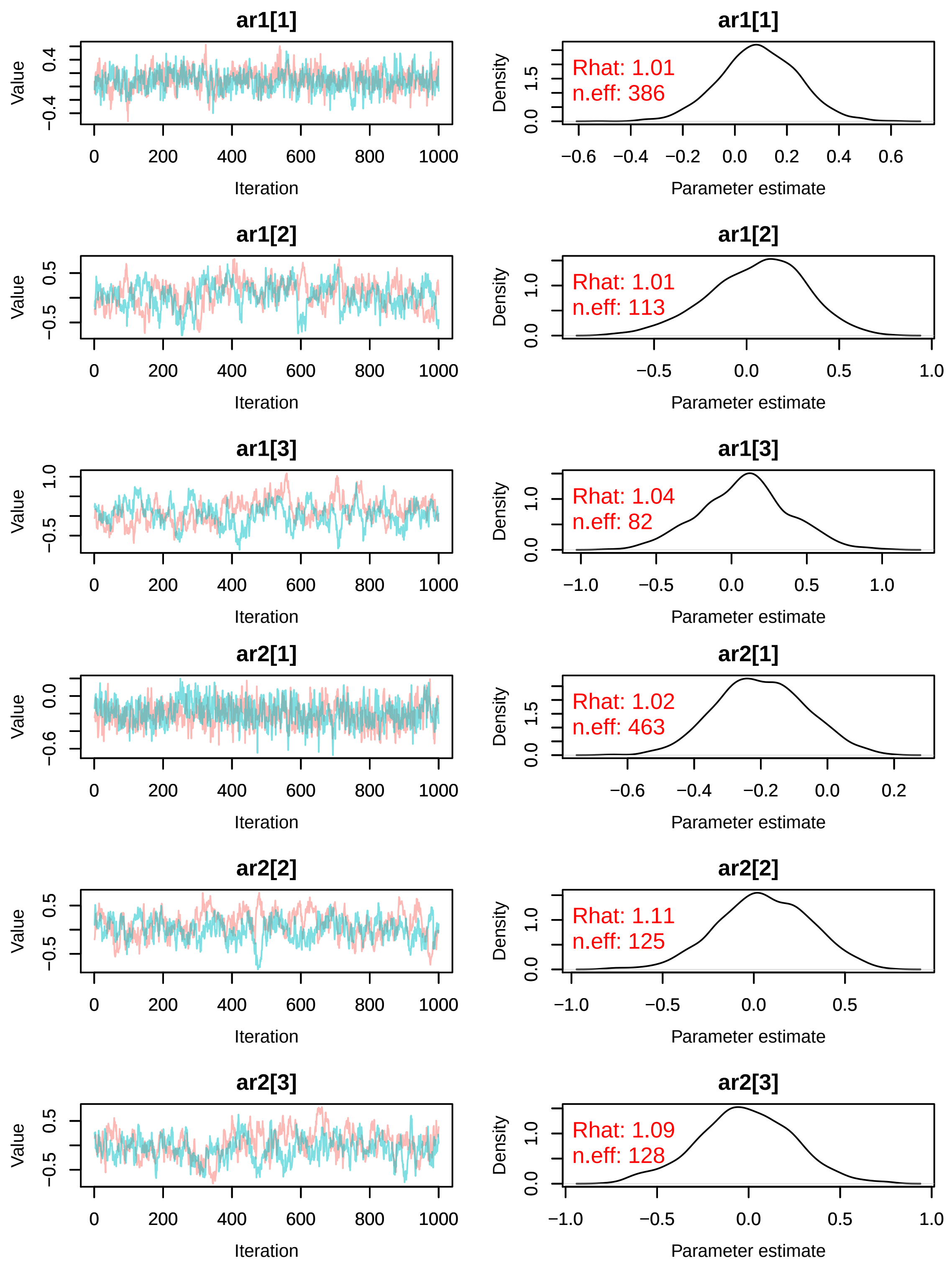
Look at traceplots for the smoothing parameters (`rho`) and latent trend parameters

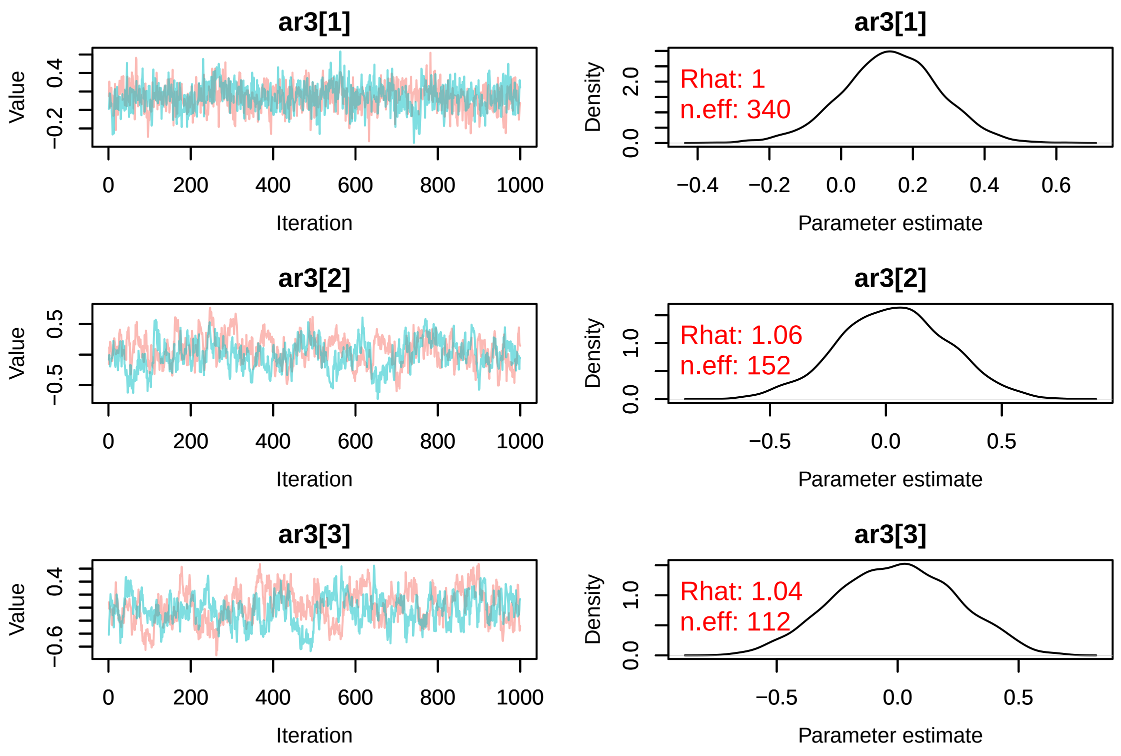
```
plot_mvgam_trace(object = trends_mod, param = "rho")
```



```
plot_mvgam_trace(object = trends_mod, param = "trend")
```

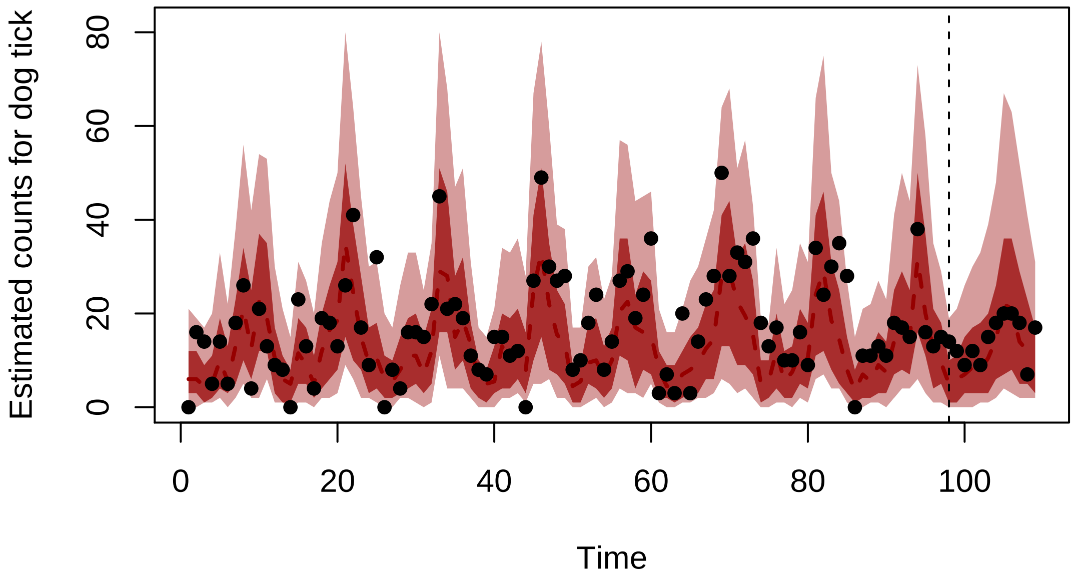



ar1[1]
Value
Iteration
Density
Parameter estimate
Rhat: 1.01
n.eff: 386
ar1[2]
Rhat: 1.01
n.eff: 113
ar1[3]
Rhat: 1.04
n.eff: 82
ar2[1]
Rhat: 1.02
n.eff: 463
ar2[2]
Rhat: 1.11
n.eff: 125
ar2[3]
Rhat: 1.09
n.eff: 128

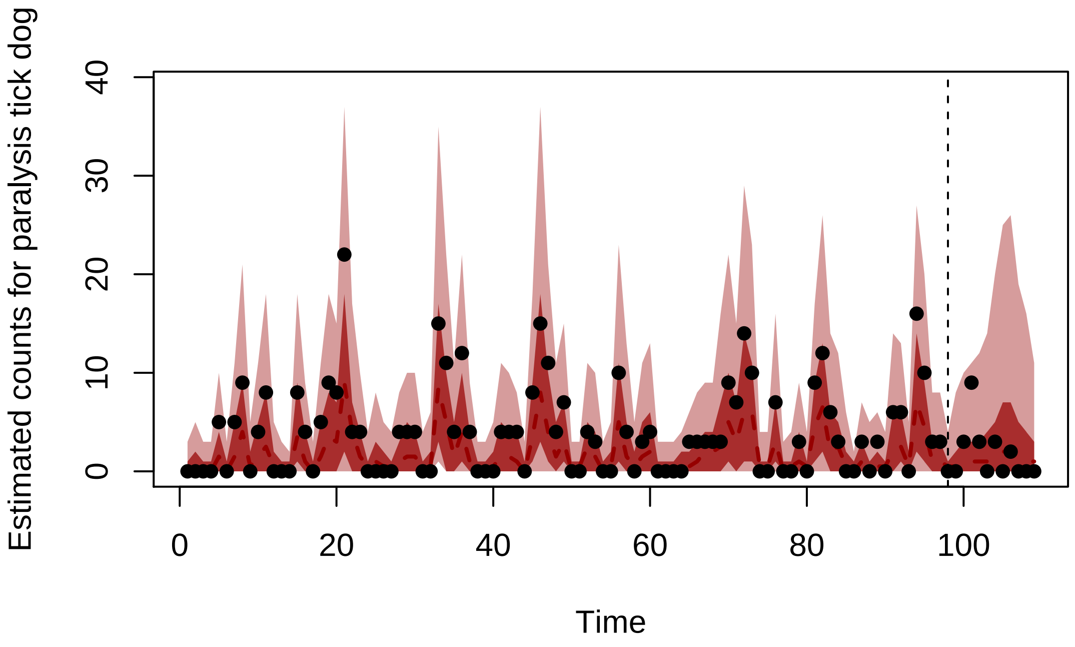


Plot posterior predictive distributions for the training and testing periods

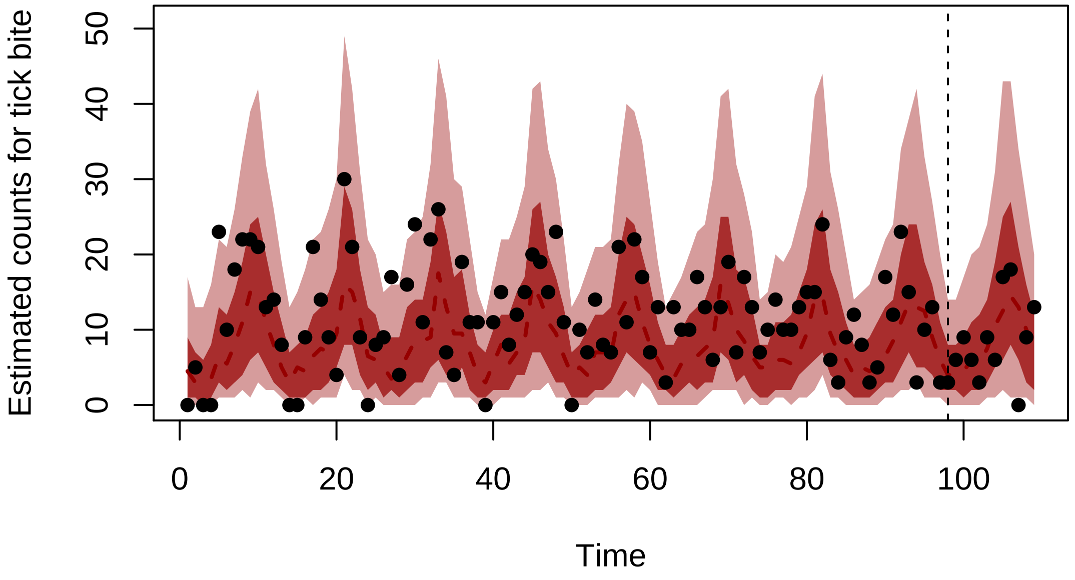
```
plot_mvgam_fc(object = trends_mod, series = 1,
    data_test = trends_data$data_test)
```



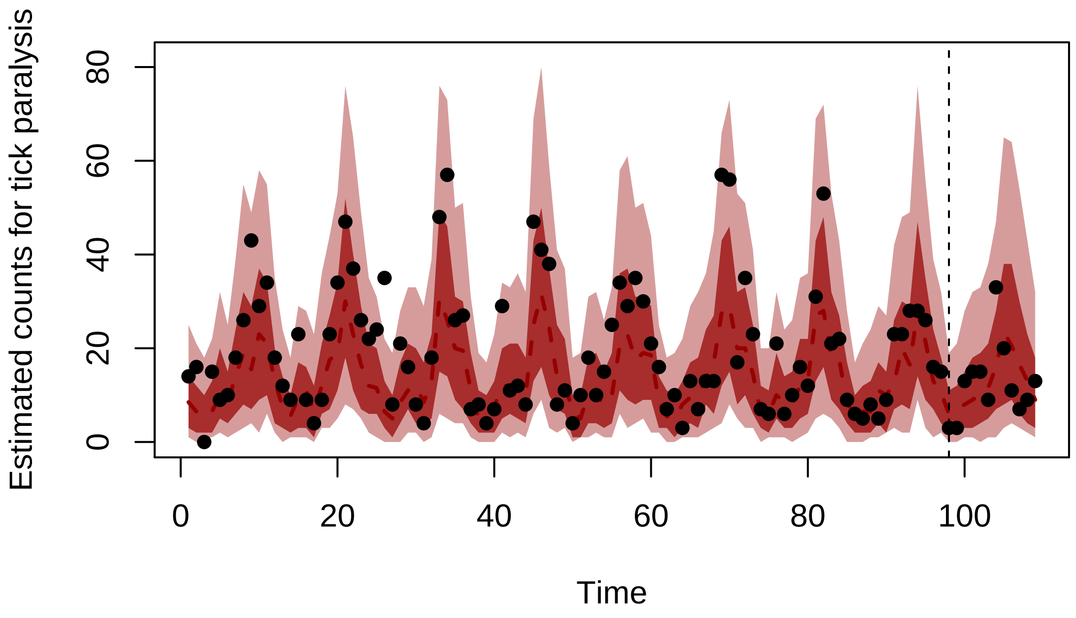
```
plot_mvgam_fc(object = trends_mod, series = 2,
    data_test = trends_data$data_test)
```



```r
plot_mvgam_fc(object = trends_mod, series = 3,
    data_test = trends_data$data_test)
```

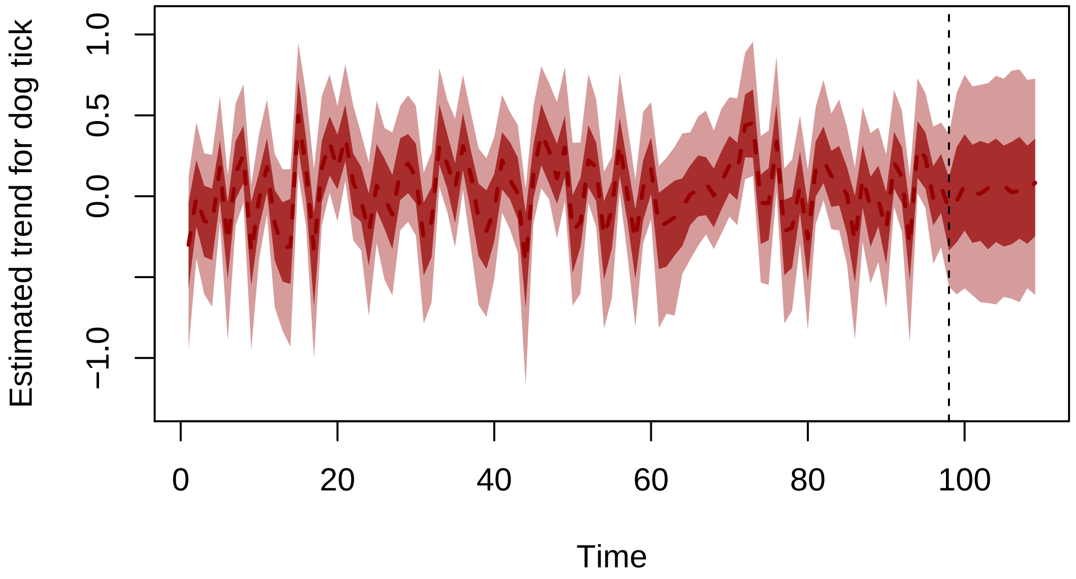


```r
plot_mvgam_fc(object = trends_mod, series = 4,
    data_test = trends_data$data_test)
```

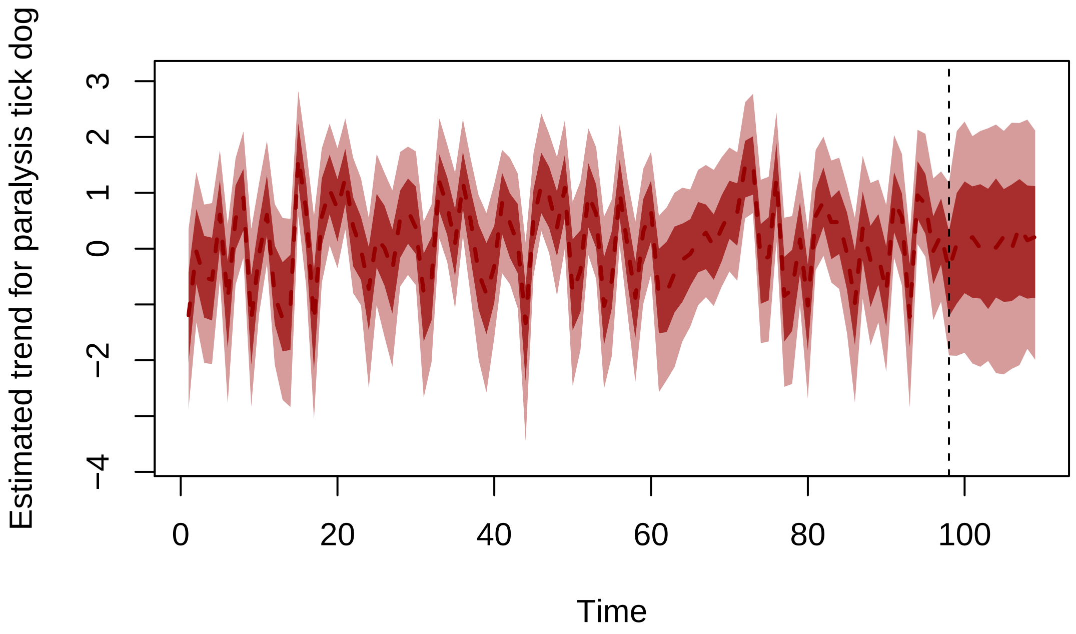


Plot posterior distributions for the latent trend estimates, again for the training and testing periods

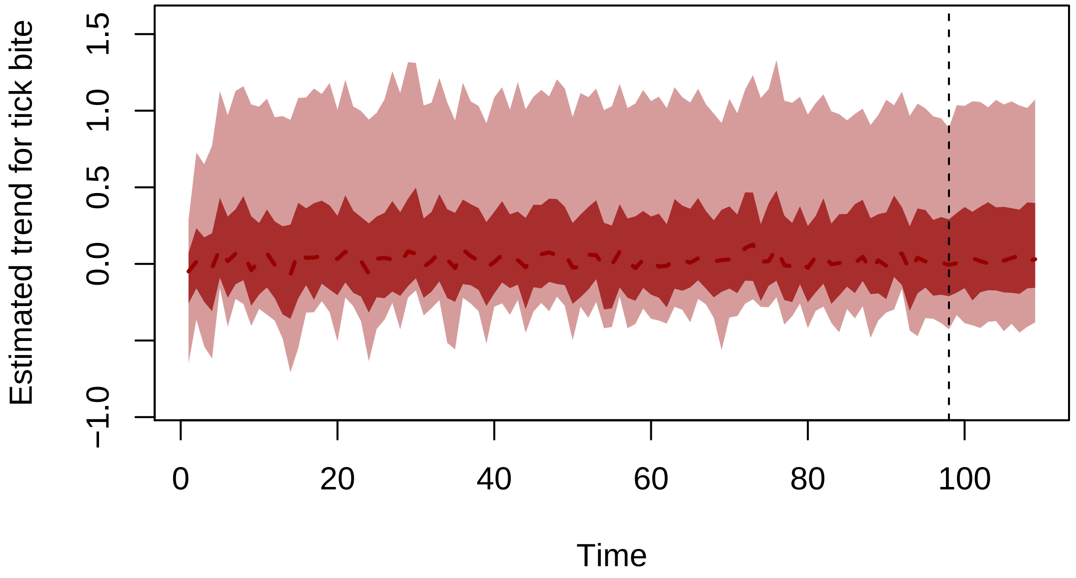
```
plot_mvgam_trend(object = trends_mod, series = 1,
    data_test = trends_data$data_test)
```



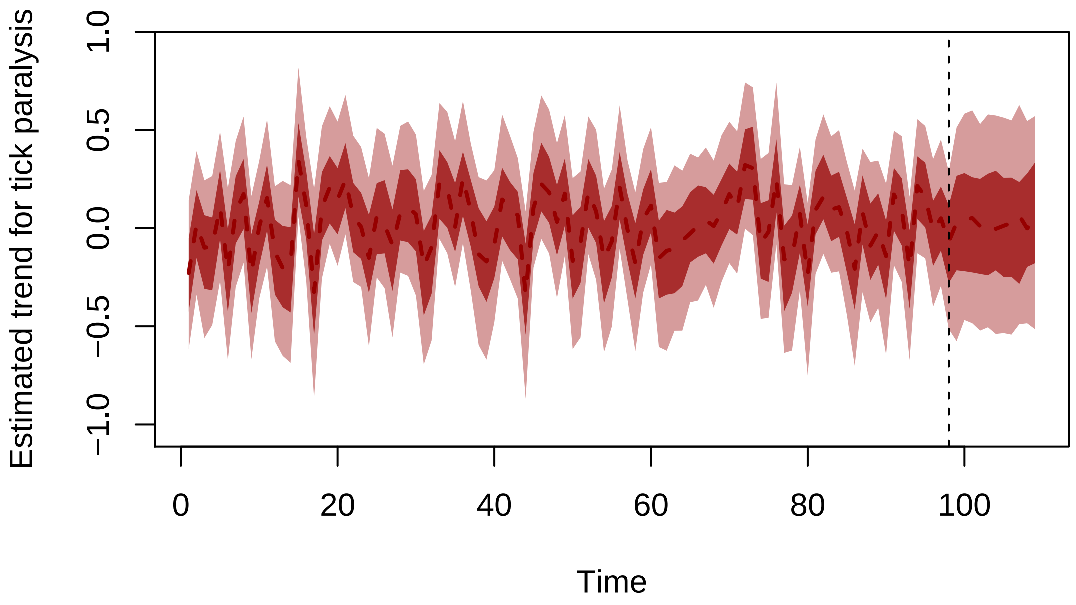
```
plot_mvgam_trend(object = trends_mod, series = 2,
    data_test = trends_data$data_test)
```



```
plot_mvgam_trend(object = trends_mod, series = 3,
    data_test = trends_data$data_test)
```

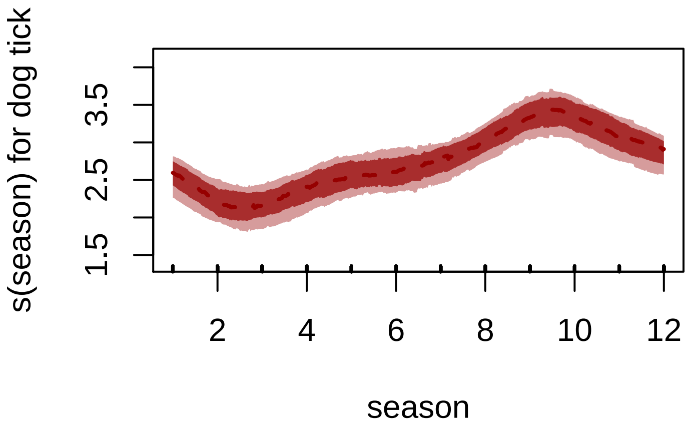


```
plot_mvgam_trend(object = trends_mod, series = 4,
    data_test = trends_data$data_test)
```



Given that we fit a model with a shared seasonal pattern, the seasonal smooths for each series will be identical. Plot it here

```
plot_mvgam_smooth(object = trends_mod, series = 1,
    smooth = "season")
```



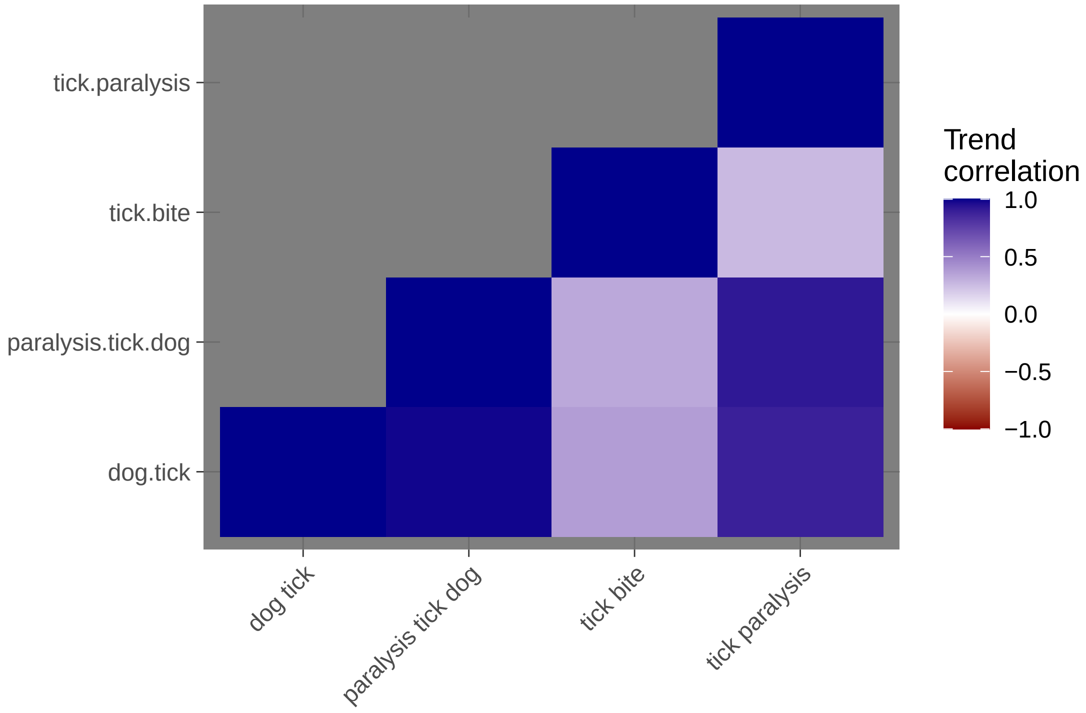
Plot posterior median estimates of latent trend correlations

```
correlations <- lv_correlations(object = trends_mod)
```

```
library(ggplot2)
```

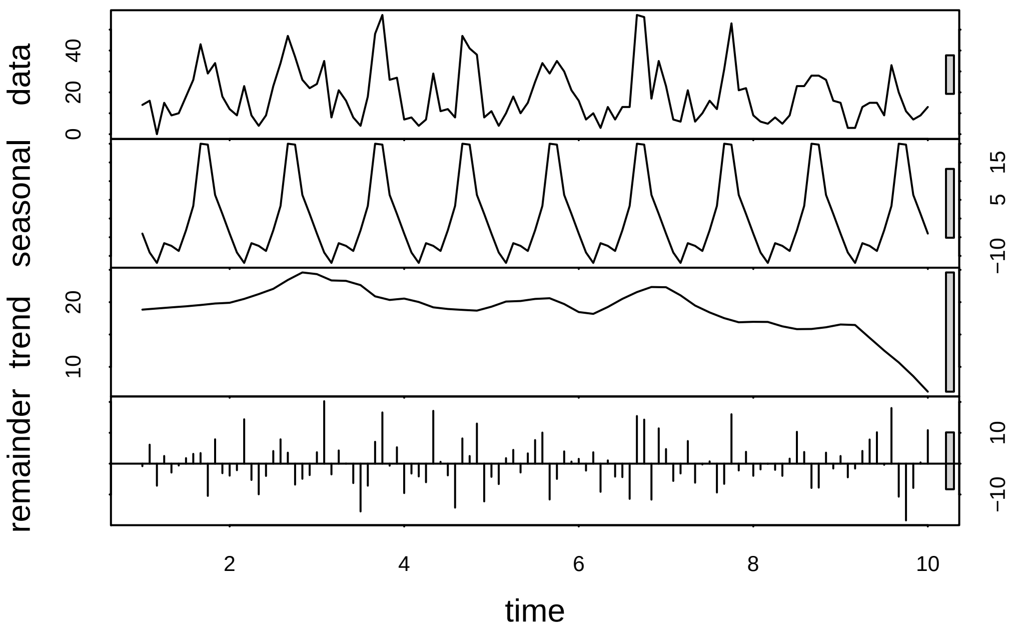
```
## Warning: package 'ggplot2' was built under R version 3.6.2
```

```
mean_correlations <- correlations$mean_correlations
mean_correlations[upper.tri(mean_correlations)] <- NA
mean_correlations <- data.frame(mean_correlations)
ggplot(mean_correlations %>% tibble::rownames_to_column("series1") %>%
    tidyr::pivot_longer(-c(series1), names_to = "series2",
        values_to = "Correlation"), aes(x = series1,
    y = series2)) + geom_tile(aes(fill = Correlation)) +
    scale_fill_gradient2(low = "darkred",
        mid = "white", high = "darkblue",
        midpoint = 0, breaks = seq(-1, 1,
            length.out = 5), limits = c(-1,
            1), name = "Trend\ncorrelation") +
    labs(x = "", y = "") + theme_dark() +
    theme(axis.text.x = element_text(angle = 45,
        hjust = 1))
```

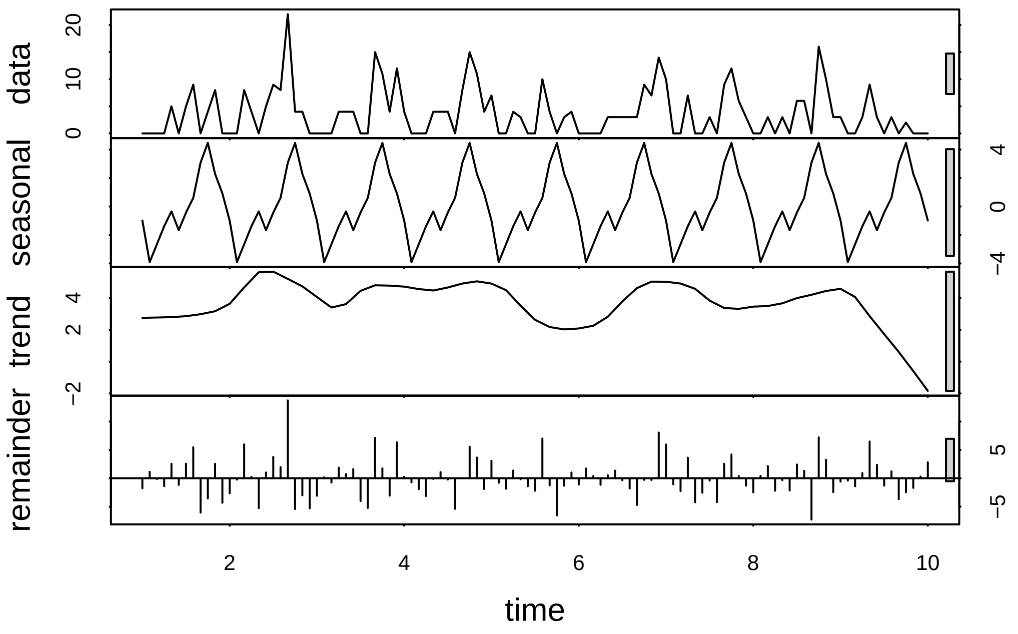


There is certainly some evidence of positive trend correlations for a few of these search terms, which is not surprising. Plot some STL decompositions of these series to see if these trends are noticeable in the data

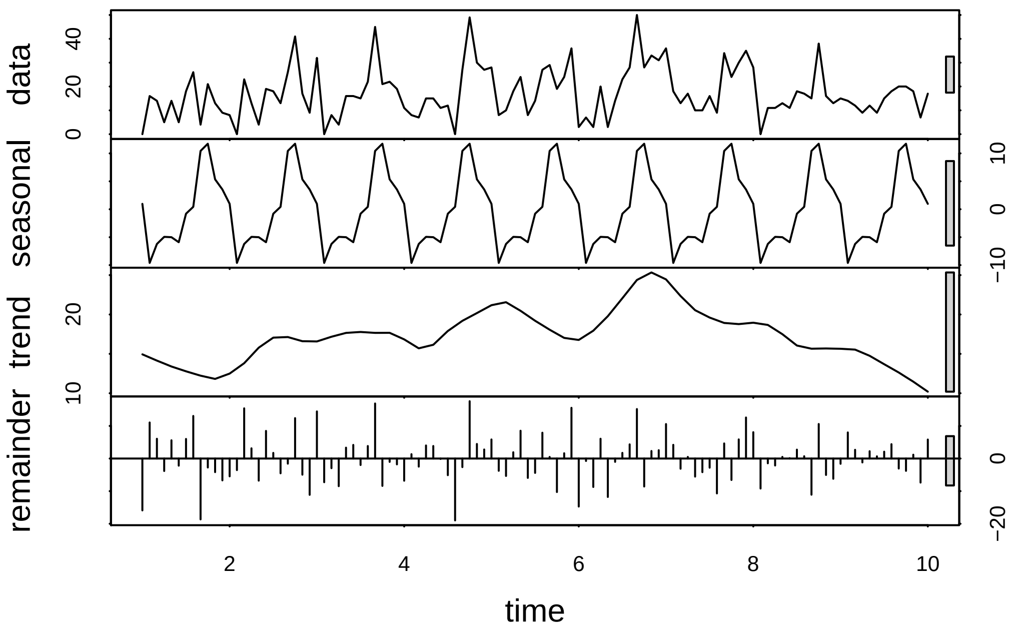
```
plot(stl(ts(as.vector(series$`tick paralysis`),
    frequency = 12), "periodic"))
```



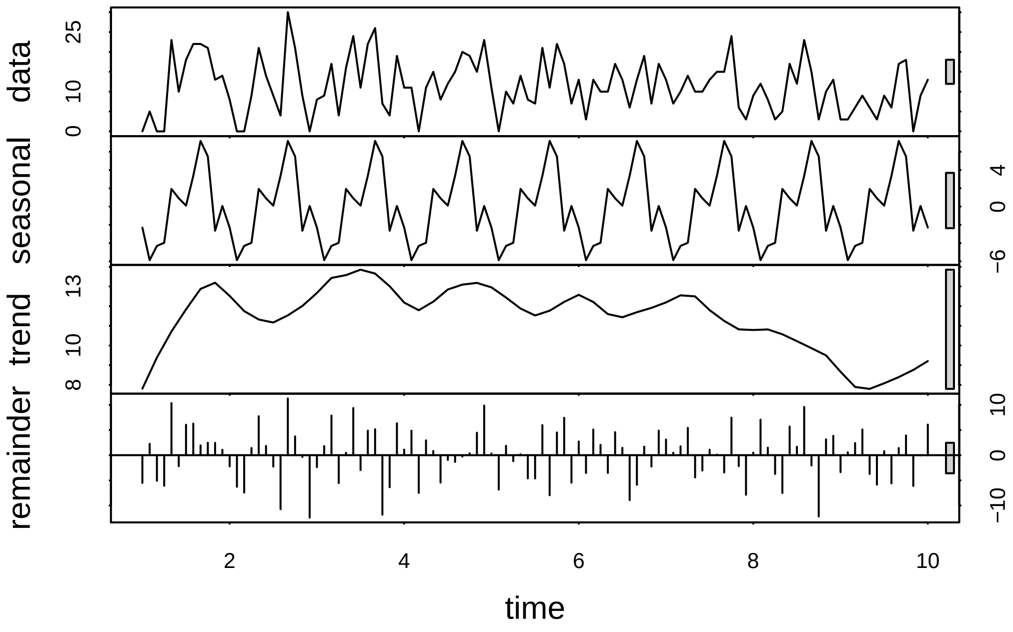
```
plot(stl(ts(as.vector(series$`paralysis tick dog`),
    frequency = 12), "periodic"))
```



```
plot(stl(ts(as.vector(series$`dog tick`),
    frequency = 12), "periodic"))
```



```r
plot(stl(ts(as.vector(series$`tick bite`),
    frequency = 12), "periodic"))
```



We have seen in other case studies how useful the recursive particle filter is for online assimilation of new observations, meaning that our forecasts can adapt without needing to re-estimate the model. For multivariate series, the particles' weights are governed by a composite multivariate likelihood, which should give us a good forecast overall but could of course lead to improvements for one series at the expense of another. Let's investigate this behaviour here. Initiate 10000 particles by assimilating the next observation in `data_test`

```r
pfilter_mvgam_init(object = trends_mod, data_assim = trends_data$data_test,
    n_particles = 10000, n_cores = 3)
```

```
## Saving particles to pfilter/particles.rda
##  ESS = 6942.695
```

Assimilate the next two observations per series

```
pfilter_mvgam_online(data_assim = trends_data$data_test[1:(4 *
    3), ], file_path = "pfilter", n_cores = 3,
    kernel_lambda = 1)
```

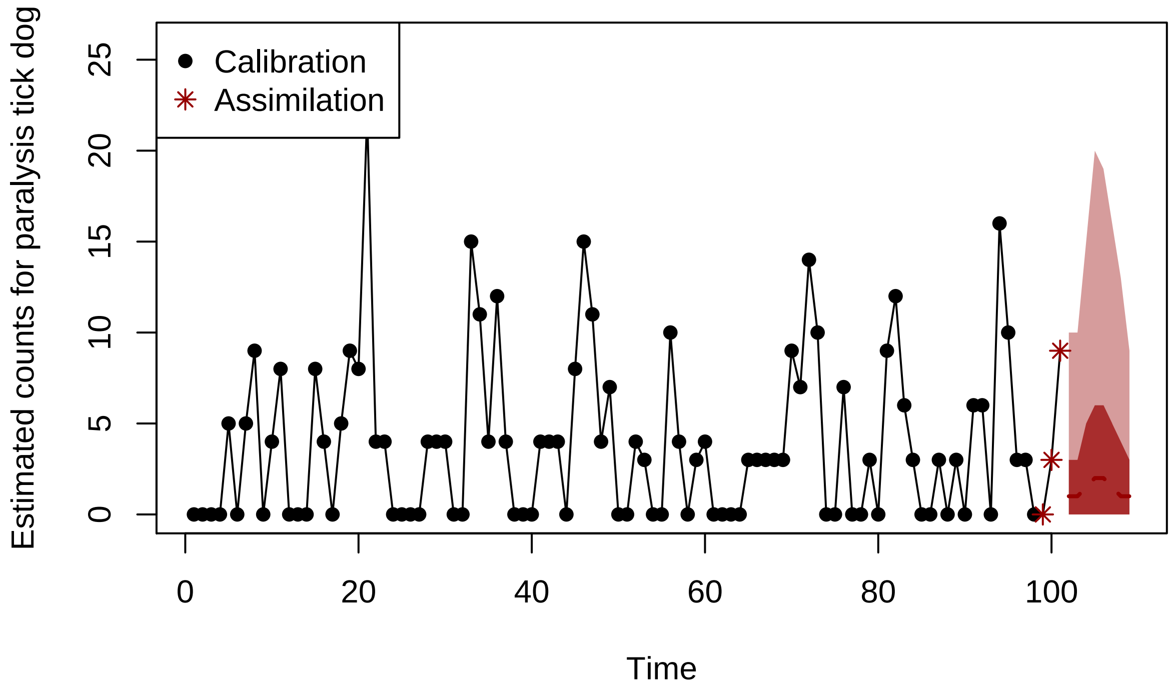
```
## Particles have already assimilated one or more observations. Skipping these
##
## Assimilating the next 2 observations
##
## Effective sample size is 7258.369 ...
##
## Smoothing particles ...
##
## Effective sample size is 1759.459 ...
##
## Smoothing particles ...
##
## Last assimilation time was 5 2021
##
## Saving particles to pfilter/particles.rda
##  ESS = 1759.459
```

Forecast from particles using the covariate information in remaining `data_test` observations

```
fc <- pfilter_mvgam_fc(file_path = "pfilter",
    n_cores = 3, data_test = trends_data$data_test,
    return_forecasts = T)
```

Inspect forecasts for `paralysis tick dog` and overlay true values in the test horizon to see how the forecast changes over time as new observations are assimilated

```
fc$fc_plots$`paralysis tick dog`()
```



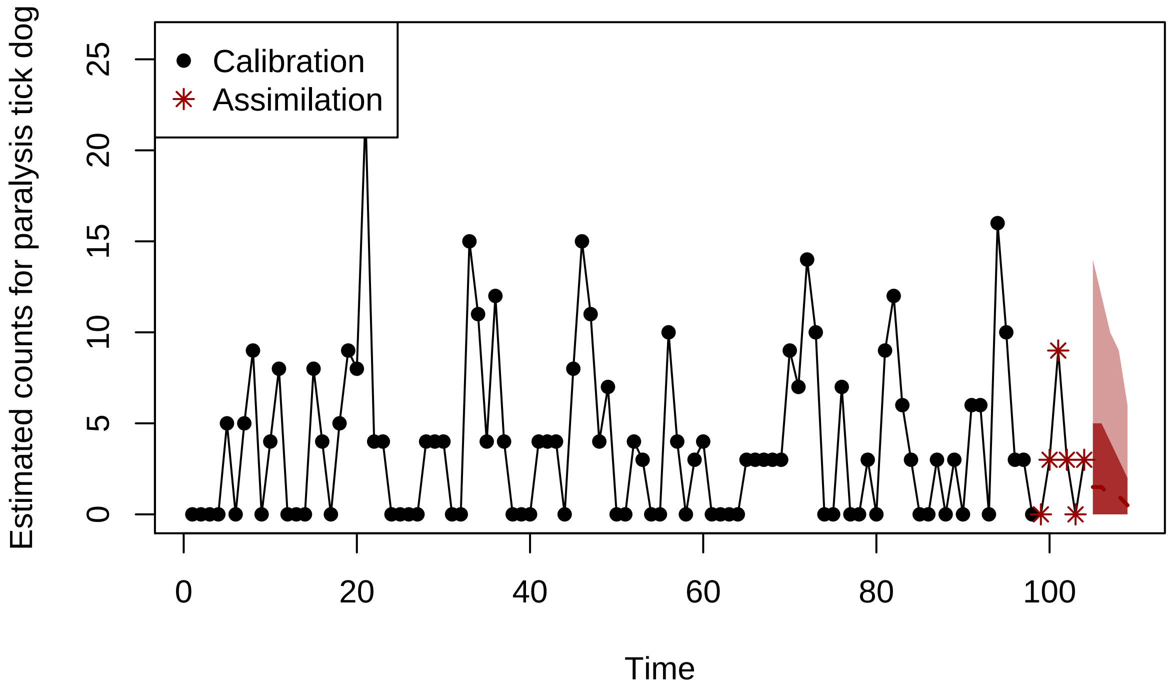
Assimilate three more observations per series

```r
pfilter_mvgam_online(data_assim = trends_data$data_test[1:(4 *
    6), ], file_path = "pfilter", n_cores = 3,
    kernel_lambda = 1)
```

```
## Particles have already assimilated one or more observations. Skipping these
##
## Assimilating the next 3 observations
##
## Effective sample size is 6192.256 ...
##
## Smoothing particles ...
##
## Effective sample size is 6599.847 ...
##
## Smoothing particles ...
##
## Effective sample size is 6746.933 ...
##
## Smoothing particles ...
##
## Last assimilation time was 8 2021
##
## Saving particles to pfilter/particles.rda
##  ESS = 6746.933
```

Forecast the remaining observations

```r
fc <- pfilter_mvgam_fc(file_path = "pfilter",
    n_cores = 3, data_test = trends_data$data_test,
    return_forecasts = T)
fc$fc_plots$`paralysis tick dog`()
```



Remove the particles

```r
unlink("pfilter", recursive = T)
```