Working with 

A University of Queensland Advanced Workshop

Session 6: Generalized Linear and Generalized Additive Models

Bill Venables, CSIRO/Data61, Dutton Park

Rhetta Chappell, Griffith University

2–5 February, 2021

Contents

1 An example from MASS: low birth weight

From Venables and Ripley (2002, Chap. 7), taken from an original in ?.

low indicator of birth weight less than 2.5 kg.

age mother's age in years.

lwt mother's weight in pounds at last menstrual period.

race mother's race ('1' = white, '2' = black, '3' = other).

smoke smoking status during pregnancy.

ptl number of previous premature labours.

ht history of hypertension.

ui presence of uterine irritability.

ftv number of physician visits during the first trimester.

bwt birth weight in grams.

The original MASS code:

```
attach(birthwt)
race <- factor(race, labels = c("white", "black", "other"))
table(ptl)

ptl
  0   1   2   3
159  24   5   1

ptd <- factor(ptl > 0)
table(ftv)

ftv
  0   1   2   3   4   6
100  47  30   7   4   1

ftv <- factor(ftv)
levels(ftv)[-(1:2)] <- "2+"
table(ftv)  # as a check

ftv
  0   1  2+
100  47  42

bwt <- data.frame(low = factor(low), age, lwt, race,
   smoke = (smoke > 0), ptd, ht = (ht > 0), ui = (ui > 0), ftv)
detach(); rm(race, ptd, ftv)
```

My preference now:

```
BirthWt <- birthwt %>%
  within({
    low <-   case_when(low == 0 ~ "Normal",
                       low == 1 ~ "Low") %>% factor() ## why? why factor??
    race <-  case_when(race == 1 ~ "White",
                       race == 2 ~ "Black",
                       race == 3 ~ "Other")
    smoke <- ifelse(smoke > 0, "Some", "None")
    ptl <-   ifelse(ptl > 0, "1+", "0")
    ht <-    ifelse(ht > 0, "Yes", "No")
    ui <-    ifelse(ui > 0, "Yes", "No")
    ftv <-   ifelse(ftv > 1, "2+", ftv)  ## change numeric -> character
  }) %>%
  select(-bwt)  ## not useful for modelling.
Store(BirthWt)
head(BirthWt, 2)
```

```
      low age lwt  race smoke ptl ht  ui ftv
85 Normal  19 182 Black  None   0 No Yes   0
86 Normal  33 155 Other  None   0 No  No  2+
```

Advice from the `fortunes` package:

> *If I were to be treated by a cure created by stepwise regression, I would prefer voodoo.*
>
> — Dieter Menne *(in a thread about regressions with many variables) R-help (October 2009)*

- Automated screening is more defensible in cases of pure prediction.
- Automated screening is dangerous if used for inference.

Caveat emptor!

To start things off:

```
options(show.signif.stars = FALSE)
ls("package:WWRUtilities", pattern = "^(step|drop)") %>% noquote()
```

```
[1] dropterm  step_AIC  step_BIC  step_down step_GIC
```

1.1 Automated screening of variables

A starting point, main effects only:

```
BW0 <- glm(low ~ ., binomial, BirthWt)
dropterm(BW0)

Single term deletions


Model:
low ~ age + lwt + race + smoke + ptl + ht + ui + ftv
       Df Deviance    AIC    LRT  Pr(Chi)
ftv     2   196.83 214.83 1.3582 0.507077
age     1   196.42 216.42 0.9419 0.331796
<none>      195.48 217.48
ui      1   197.59 217.59 2.1100 0.146342
smoke   1   198.67 218.67 3.1982 0.073717
race    2   201.23 219.23 5.7513 0.056380
lwt     1   200.95 220.95 5.4739 0.019302
ht      1   202.93 222.93 7.4584 0.006314
ptl     1   203.58 223.58 8.1085 0.004406
```

Screen for possible interactions:

```
sBW0 <- step_AIC(BW0, scope = list(
  lower = ~1,
  upper = ~.^2+poly(age, 2)+poly(lwt,2)))
dropterm(sBW0)

Single term deletions


Model:
low ~ age + lwt + smoke + ptl + ht + ui + ftv + age:ftv + smoke:ui
         Df Deviance    AIC     LRT   Pr(Chi)
<none>        183.07 207.07
smoke:ui  1   186.99 208.99  3.9127 0.0479224
ht        1   191.21 213.21  8.1374 0.0043361
lwt       1   191.56 213.56  8.4856 0.0035797
ptl       1   193.59 215.59 10.5146 0.0011843
age:ftv   2   199.00 219.00 15.9295 0.0003475
```

1.2 An extended model with smooth terms

We consider some flexibility in the *age* term and its interaction with *ftv*.

```
suppressPackageStartupMessages(library(mgcv))
BW1 <- bam(low ~ smoke*ui + ht + s(lwt) + ptl + s(age) +
           poly(age, 2)*ftv, family = binomial, data = BirthWt,
           control = gam.control(trace = TRUE))

Setup complete. Calling fit

Deviance = 108.743823386773 Iterations - 1

Deviance = 185.8543718777252 Iterations - 2

Deviance = 181.9291105967722 Iterations - 3

Deviance = 180.2280499723356 Iterations - 4

Deviance = 179.8826614654222 Iterations - 5

Deviance = 179.8322860374117 Iterations - 6

Deviance = 179.8277550010866 Iterations - 7

Deviance = 179.8275490285344 Iterations - 8

Deviance = 179.8275409001919 Iterations - 9
```

```
Fit complete. Finishing gam object.
          user.self sys.self elapsed
initial       0.392    0.000   0.392
gam.setup     0.008    0.000   0.008
pre-fit       0.000    0.000   0.000
fit           0.276    0.001   0.279
finalise      0.282    0.000   0.282
```

```
BW1 <- update(BW1, control = gam.control(trace = FALSE))  ## turn off tracing
```

```
anova(BW1)
Family: binomial
Link function: logit
Formula:
low ~ smoke * ui + ht + s(lwt) + ptl + s(age) + poly(age, 2) *
    ftv
Parametric Terms:
                 df Chi.sq p-value
smoke             1  3.776 0.05201
ui                1  7.561 0.00596
ht                1  7.102 0.00770
ptl               1  8.754 0.00309
poly(age, 2)      1  0.092 0.76137
ftv               2  3.377 0.18475
smoke:ui          1  3.832 0.05029
poly(age, 2):ftv  4 13.268 0.01004
Approximate significance of smooth terms:
         edf Ref.df Chi.sq p-value
s(lwt) 1.000  1.000  7.007 0.00812
s(age) 1.556  1.990  2.239 0.34807
```

1.3 Looking at the terms

We will first do a visualisation "by hand" and follow up with a packaged tool: *visreg::visreg*

1.4 A helper function: the most frequent value

```
mostFreq <- function(x, ...) {
  UseMethod("mostFreq")
}

mostFreq.logical <- function(x, ...) {
  tx <- as.vector(table(x))
  tx[2] > tx[1]
}

mostFreq.character <- function(x, ...) {
  tx <- table(x)
  names(tx)[which.max(tx)]
}
```

```r
mostFreq.factor <- function(x, ...)
    mostFreq.character(as.character(x))

mostFreq.numeric <- stats::median.default  ## check argument names
                                           ## covers class 'integer' as well
# Store(list = ls(pattern = "^mostFreq"), lib = .Robjects)
```

1.5 The main two-way interaction

Predict the probability of low birth weight with varying *age* and *ftv*, and other variables at or near their modal value.

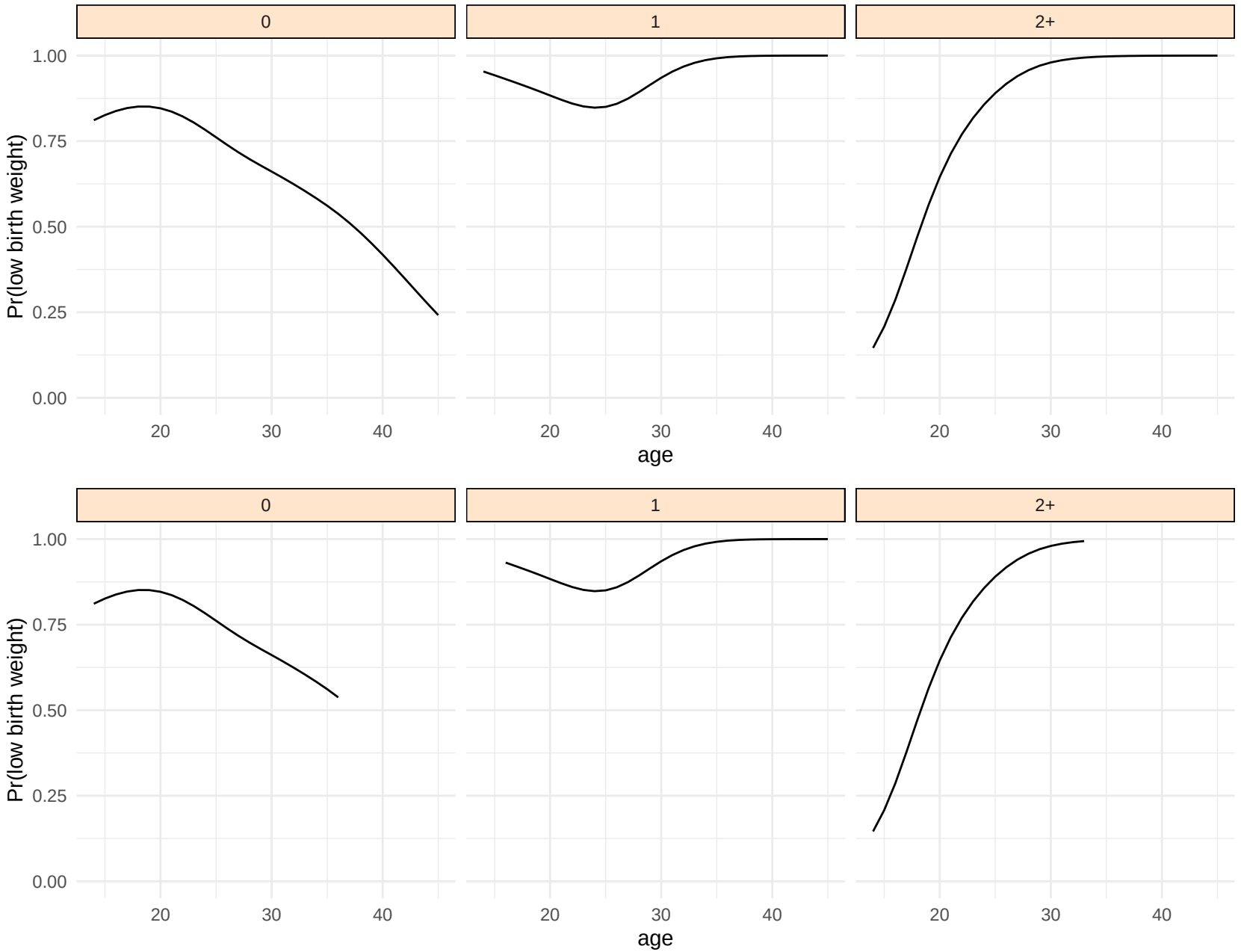
```
all.vars(formula(BW1))
```

```
[1] "low"   "smoke" "ui"    "ht"    "lwt"   "ptl"   "age"   "ftv"
```

```
pBirthWt <- with(BirthWt,
    expand.grid(smoke = mostFreq(smoke),
                ui = mostFreq(ui),
                ht = mostFreq(ht),
                lwt = mostFreq(lwt),
                ptl = mostFreq(ptl),
                age = min(age):max(age),    ## first branch
                ftv = sort(unique(ftv))))   ## second branch
pBirthWt$pBW1 <- predict(BW1, pBirthWt, type = "response")
p0 <- ggplot(pBirthWt) + aes(x = age, y = pBW1) +
  geom_line() + ylim(0,1) + ylab("Pr(low birth weight)") +
  facet_grid(. ~ ftv)
```

To bring the predictions closer to the actual *data*, confine the predictions to *age* ranges that apply within the levels of *ftv*

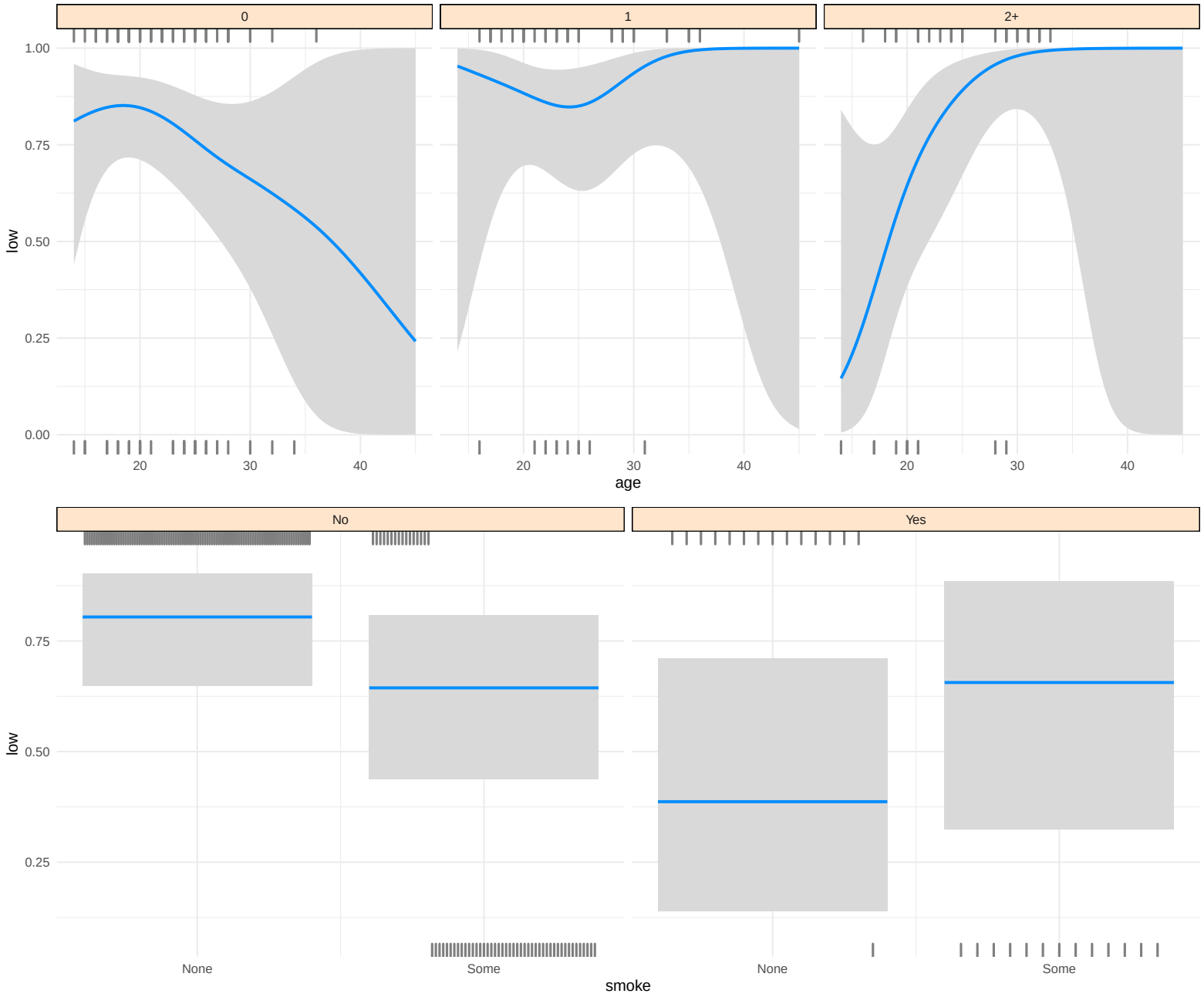
```
pBirthWt <- within(pBirthWt, {
    rngs <- do.call(cbind, with(BirthWt, tapply(age, ftv, range)))
    pBW1a <- pBW1
    is.na(pBW1a[age < rngs[1, ftv] | age > rngs[2, ftv]]) <- TRUE
    rm(rngs)
})
p1 <- ggplot(na.omit(pBirthWt)) + aes(x = age, y = pBW1a) +
    geom_line() + ylim(0,1) + # theme_bw() +
    ylab("Pr(low birth weight)") + facet_grid(. ~ ftv)

# gridExtra::grid.arrange(p0, p1, nrow=2)
p0/p1
```


0
1
2+
1.00
0.75
0.50
0.25
0.00
Pr(low birth weight)
20
30
40
age

1.6 The *visreg* alternative

This is a more recent package, which offers some advantages and offers some traps.

```
library(visreg)
p0 <- visreg(BW1, xvar = "age", by = "ftv", scale = "resp", ylim = 0:1, plot = FALS
p1 <- visreg(BW1, xvar = "smoke", by = "ui", scale = "resp", ylim = 0:1, plot = FA
plot(p0, gg = TRUE)/plot(p1, gg = TRUE)
```


0
1
2+
1.00
0.75
0.50
0.25
0.00
low
20
30
40
age
No
Yes
0.75
0.50
0.25
low
None
Some
None
Some
smoke

2 The “churn” example

This example has already been introduced.

```
fname <- system.file("extdata", "churnData.csv.gz", package = "WWRCourse")
churnData <- read_csv(gzfile(fname),      ## neater read (for eventual notebook)
                      col_types = cols(.default = col_double(),
                                       state = col_character(),
                                       area_code = col_character(),
                                       international_plan = col_character(),
                                       voice_mail_plan = col_character(),
                                       churn = col_character(),
                                       sample = col_character())) %>%
  within({
    area_code <- sub("^area_code_", "", area_code) %>% factor()
    churn <- (churn == "no") + 0       ## replace by a binary response
  }) %>%
  untibble()
names(churnData) <- sub("^(total|number)_", "", names(churnData)) ## neater
```

Split into training and test samples, as directed, and remove variables known to be irrelevant or redundant (in a linear models sense).

```
names(churnData) %>% noquote()
```

```
 [1] state                 account_length        area_code
 [4] international_plan    voice_mail_plan       vmail_messages
 [7] day_minutes           day_calls             day_charge
[10] eve_minutes           eve_calls             eve_charge
[13] night_minutes         night_calls           night_charge
[16] intl_minutes          intl_calls            intl_charge
[19] customer_service_calls churn                sample
```

```
churn_train <- churnData %>%
  filter(sample == "train") %>%
  select(-sample, -ends_with("charge"))
churn_test  <- churnData %>%
  filter(sample == "test" ) %>%
  select(-sample, -ends_with("charge"))
```

Two modelling construction paradigms: AIC and BIC

```
churnAIC <- glm(churn ~ ., binomial, churn_train) %>%
  step_AIC(scope = list(lower = ~state))
churnBIC <- glm(churn ~ ., binomial, churn_train) %>%
  step_BIC(scope = list(lower = ~state))
```

Look at what terms remain:

```
dropterm(churnAIC) %>% booktabs(digits = c(0,0,2,2,2,6))
```

	Df	Deviance	AIC	LRT	Pr(Chi)
state	50	2160.27	2182.27	87.41	0.000838
<none>		2072.86	2194.86		
day_calls	1	2074.89	2194.89	2.03	0.154048
vmail_messages	1	2076.96	2196.96	4.10	0.043006
night_minutes	1	2084.59	2204.59	11.73	0.000615
intl_calls	1	2085.70	2205.70	12.84	0.000339
voice_mail_plan	1	2086.62	2206.62	13.76	0.000208
intl_minutes	1	2088.94	2208.94	16.07	0.000061
eve_minutes	1	2117.24	2237.24	44.38	0.000000
day_minutes	1	2228.71	2348.71	155.84	0.000000
customer_service_calls	1	2253.55	2373.55	180.69	0.000000
international_plan	1	2274.72	2394.72	201.86	0.000000

```
dropterm(churnBIC) %>% booktabs(digits = c(0,0,2,2,2,6))
```

	Df	Deviance	BIC	LRT	Pr(Chi)
state	50	2165.55	2238.55	86.52	0.001033
<none>		2079.03	2557.61		
night_minutes	1	2090.62	2561.10	11.60	0.000660
intl_calls	1	2092.03	2562.50	13.00	0.000311
intl_minutes	1	2095.26	2565.74	16.24	0.000056
eve_minutes	1	2122.48	2592.95	43.45	0.000000
voice_mail_plan	1	2127.48	2597.95	48.45	0.000000
day_minutes	1	2234.55	2705.03	155.53	0.000000
customer_service_calls	1	2258.99	2729.46	179.96	0.000000
international_plan	1	2281.14	2751.62	202.12	0.000000

See how well each perform on the test data:

```
(confAIC <- table(actual = churn_test$churn,
                  predicted = (predict(churnAIC, churn_test) > 0) + 0))
```

```
      predicted
actual    0    1
     0   55  169
     1   45 1398
```

```
(confBIC <- table(actual = churn_test$churn,
                  predicted = (predict(churnBIC, churn_test) > 0) + 0))
```

```
      predicted
actual    0    1
     0   56  168
     1   44 1399
```

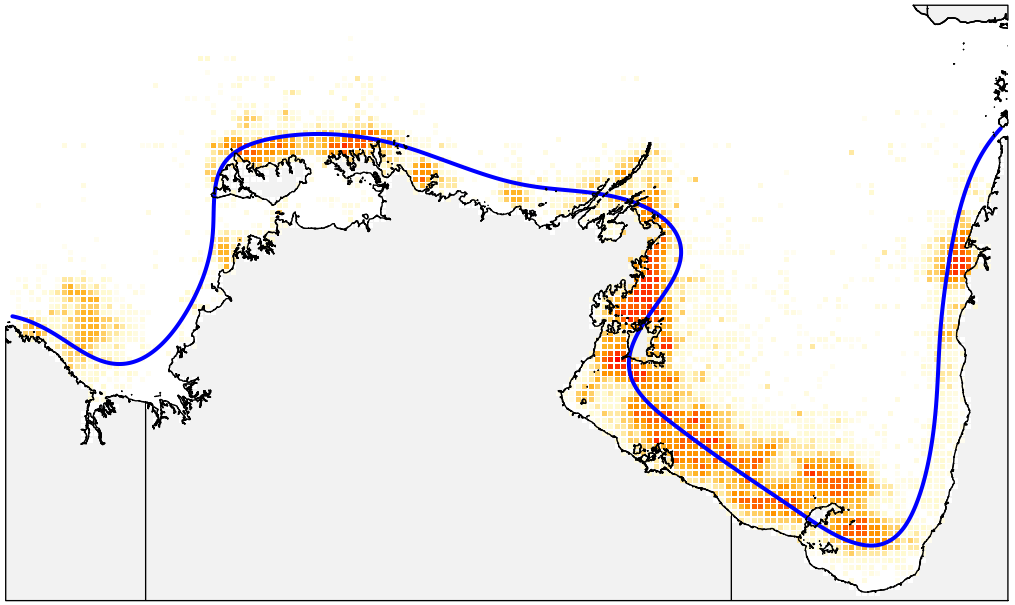
The (crude) error rates are:

```
c(AIC = (1 - sum(diag(confAIC))/sum(confAIC))*100,
  BIC = (1 - sum(diag(confBIC))/sum(confBIC))*100) %>% round(2)
```

```
  AIC   BIC
12.84 12.72
```

3 Tiger prawn species split

The Northern Prawn Fishery: Tiger prawn effort and “the blue line”



Background

- Two species of Tiger prawns are caught together.
- Both species require separate Stock Assessment.
- The assessment model requires catches of Tiger prawns to be split (by weight)
- Problem: Build a model for partitioning catches into the two component species.
- Data: independent surveys (12 in all) where catches have been split into the two species, *Penaeus semisulcatus* (Grooved) and *P. esculentus* (Brown).
- Both species have annual offshore migration patterns.

Variables available:

- Response: *Psem*, *Pesc*, (*Total* = *Psem* + *Pesc*) in gms;

  Predictors:

- *Longitude*, *Latitude* – of trawl shot;
- *Coast*, *Sea* – alternative spatial coordinates;
- *Depth* – of trawl shot;
- *Mud* – the % mud in the substrate;
- *DayOfYear* – to allow for annual migration periodicity;
- *ElapsedDays* – days since 1970-01-01, for long term trend
- *Survey* – used for a random effect extension.

Strategy:

- Build a simpler GLM using mainly splines, with a term in *DayOfYear* and *Sea* to allow for temporal (annual migration) effects.
- Develop a more sophisticated GAM to take advantage of more recent modelling technology
- Look at a long-term trend term as a perturbation to the model
- Consider GLMMs with random terms for *Survey*, eventually

Model terms, GLM:

- Spline in *Coast* surrogate for large-scale benthic changes,
- Splines in *Sea*, *Depth* and *Mud* – more local spatial effects,
- Periodic term in *DayOfYear* and its interaction with *Sea* – annual migration effects,
- Spline in *ElapsedDays* – testing for long-term stability.

3.1 An initial GLM

The GLM fitting process for such models can be slow to converge under the normal algorithm. Two possible alternatives:

- Use the `glm2` library, which has a modified convergence process, (Marschner, 2011).
- In this case the problem is with the variable weights needed. Only the *relative* weights are used with `quasi` models. It can be an advantage to *scale* the weights so that, on average, they are about unity.

The periodic terms will use Fourier polynomials:

```
Annual <- function (day, k = 4) {  ## day of the year, starting from 0
  theta <- Arg(complex(argument = 2*base::pi*day/364.25))
  X <- matrix(0, length(theta), 2 * k)
  nam <- as.vector(outer(c("c", "s"), 1:k, paste, sep = ""))
  dimnames(X) <- list(names(day), nam)
  m <- 0
  for (j in 1:k) {
    X[, (m <- m + 1)] <- cos(j * theta)
    X[, (m <- m + 1)] <- sin(j * theta)
  }
  X
}
Store(Annual,Tigers, lib = .Robjects)
```

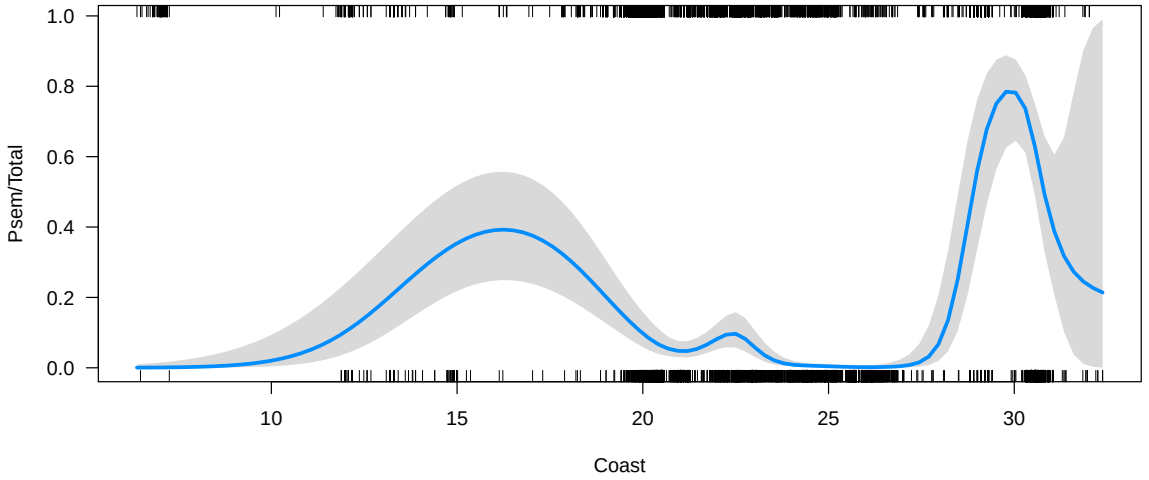
```
TModelGLM <- glm(Psem/Total ~ ns(Coast, 10) +
                    ns(Sea, 5) +
                    ns(Depth, 5) +
                    ns(Mud, 4) +
                    Annual(DayOfYear, 4)*Sea,
                  family = quasibinomial,
                  data = Tigers, weights = Total/mean(Total),
                  trace = TRUE)

Deviance = 3722.33 Iterations - 1
Deviance = 3043.545 Iterations - 2
Deviance = 2910.096 Iterations - 3
Deviance = 2896.355 Iterations - 4
Deviance = 2895.734 Iterations - 5
Deviance = 2895.73 Iterations - 6
Deviance = 2895.73 Iterations - 7

TModelGLM$call$trace <- NULL  ## for future updating
Store(TModelGLM, lib = .Robjects)
# (nam <- names(model.frame(TModelGLM)))  ## for term plotting
# nam <- nam[2:7]                         ## terms to plot
```

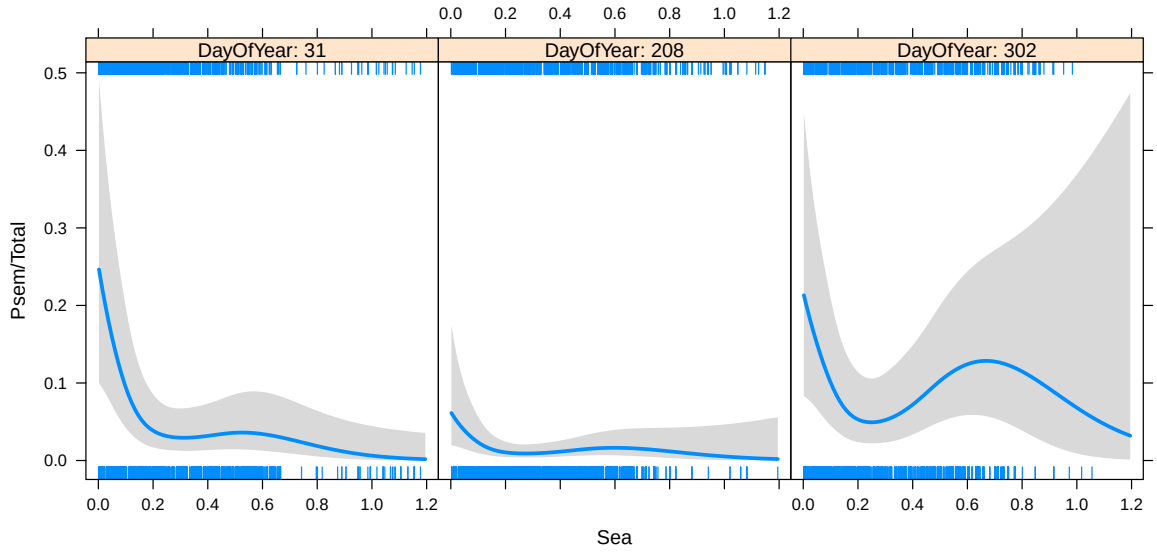
Look at the shape of the main effect terms, to see implications:

```
# layout(matrix(1:6, 2, 3, byrow=TRUE))   ## 2 x 3 array of plots
# termplot(TModelGLM, terms = nam, se = TRUE, rug=TRUE,
#          ylim = c(-5, 5))
# layout(1)
visreg(TModelGLM, "Coast", scale = "resp") ## issues warning...
```

```
Conditions used in construction of plot
Sea: 0.2241294
Depth: 24.70833
Mud: 38.58479
DayOfYear: 208
Psem: 258.6195
```

```
visreg(TModelGLM, "Sea", by = "DayOfYear", scale = "resp")
```


Psem/Total
1.0
0.8
0.6
0.4
0.2
0.0
10
15
20
25
30
Coast


DayOfYear: 31
DayOfYear: 208
DayOfYear: 302
Psem/Total
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Sea

3.2 A long-term trend?

The stability of species ratios over time is important. We can check for this by including a spline term in *ElapsedDays*:

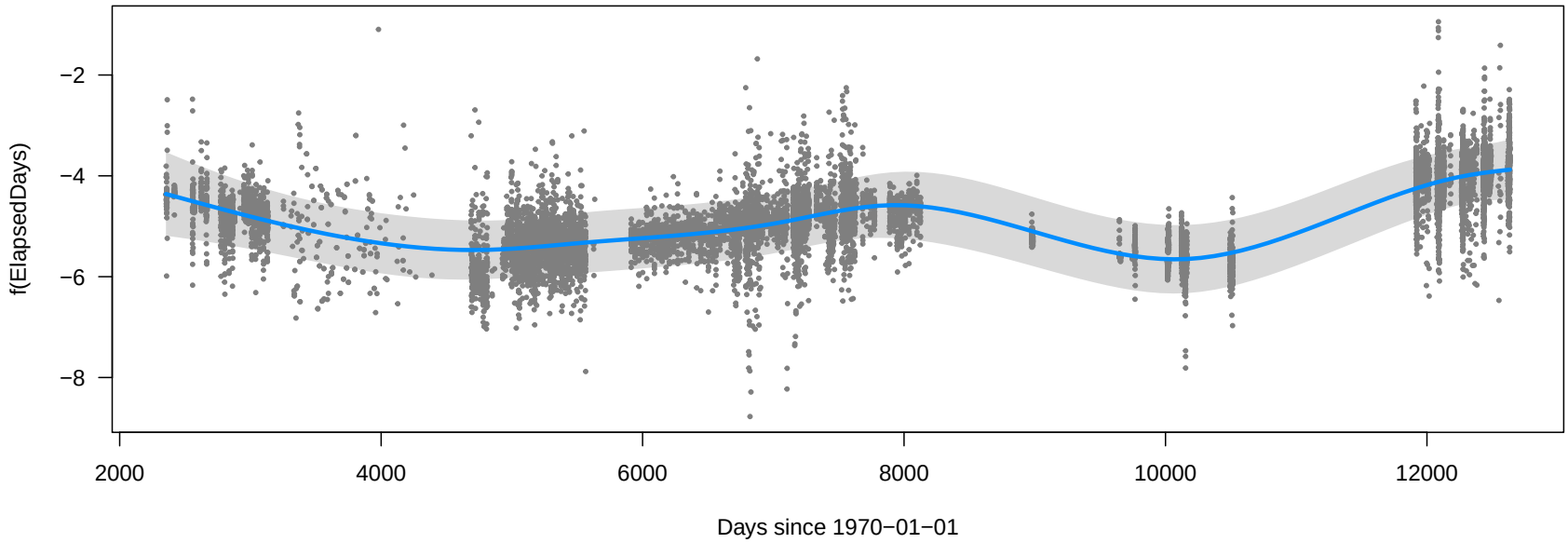
```
TM2 <- update(TModelGLM, . ~ . + ns(ElapsedDays, 7))
Store(TM2, lib = .Robjects)
```

```
anova(TModelGLM, TM2, test = "F")
```

```
Analysis of Deviance Table

Model 1: Psem/Total ~ ns(Coast, 10) + ns(Sea, 5) + ns(Depth, 5) + ns(Mud,
    4) + Annual(DayOfYear, 4) * Sea
Model 2: Psem/Total ~ ns(Coast, 10) + ns(Sea, 5) + ns(Depth, 5) + ns(Mud,
    4) + Annual(DayOfYear, 4) + Sea + ns(ElapsedDays, 7) + Annual(DayOfYear,
    4):Sea
  Resid. Df Resid. Dev Df Deviance      F    Pr(>F)
1     12528     2895.7
2     12521     2667.4  7   228.28 25.613 < 2.2e-16
```

Significant, but is it important?

```
visreg(TM2, "ElapsedDays", xlab = "Days since 1970-01-01")

Conditions used in construction of plot
Coast: 23.96775
Sea: 0.2241294
Depth: 24.70833
Mud: 38.58479
DayOfYear: 208
Psem: 258.6195
```



3.3 A working GAM with new technology

The mgcv package represents a major advance in smooth model fitting technology in several respects, including

- Smoothed terms in multiple predictors can now be handled,
- A wide variety of basis functions is available, including e.g. thin plate splines, cyclic spline bases, &c,
- A powerful *visualisation* tool in projections of predictor variable space is available in addition to tools for inspection of individual terms

The price is:

- The package is still under development and new versions are fairly common (though becoming less so)
- The implementation is to some extent non-standard R.

The working model:

```
suppressPackageStartupMessages(library(mgcv))
TModelGAM <- bam(Psem/Total ~ s(Longitude, Latitude) +  ## bam(...) oscillates!
                 te(DayOfYear, Sea,   k = c(5, 5), bs = c("cc", "cs")) +
                 te(DayOfYear, Depth, k = c(5, 5), bs = c("cc", "cs")) +
                 te(Sea,       Depth, k = c(5, 5), bs = c("cs", "cs")) + ## NB!
                 s(Mud, k = 5),
                 family = quasibinomial, data = Tigers,
                 knots = list(DayOfYear = seq(1, 365.25, length = 5)),
                 control = gam.control(trace = TRUE),    ## to check progress
                 weights = Total/mean(Total))              ## takes forever!

Setup complete. Calling fit

Deviance = 3544.05160018479 Iterations - 1

Deviance = 2980.17536523122 Iterations - 2

Deviance = 2449.93360729799 Iterations - 3

Deviance = 2348.71063304965 Iterations - 4

Deviance = 2329.8783285786 Iterations - 5

Deviance = 2323.61041290277 Iterations - 6

Deviance = 2322.61169505222 Iterations - 7
```

```
Deviance = 2323.73010787116 Iterations - 8

Deviance = 2324.12178460156 Iterations - 9

Deviance = 2324.0884565695 Iterations - 10

Deviance = 2324.09852686282 Iterations - 11

Deviance = 2324.09595894672 Iterations - 12

Deviance = 2324.0966149238 Iterations - 13

Deviance = 2324.09644747008 Iterations - 14

Fit complete. Finishing gam object.
          user.self sys.self elapsed
initial       0.483    0.001   0.483
gam.setup     0.157    0.000   0.157
pre-fit       0.000    0.000   0.000
fit           9.056    0.019   9.077
finalise      0.822    0.005   0.825

TModelGAM_NS <- update(TModelGAM, . ~ . + ns(ElapsedDays, 10),
                       mustart = fitted(TModelGAM))  ## non-stationary

Setup complete. Calling fit

Deviance = 2324.09644747008 Iterations - 1
```

```
Deviance = 2158.61286477359 Iterations - 2

Deviance = 2141.28817009492 Iterations - 3

Deviance = 2140.57007108738 Iterations - 4

Deviance = 2140.74444309258 Iterations - 5

Deviance = 2140.81461411052 Iterations - 6

Deviance = 2140.81445366671 Iterations - 7

Fit complete. Finishing gam object.
          user.self sys.self elapsed
initial       0.488    0.000   0.488
gam.setup     0.153    0.000   0.153
pre-fit       0.000    0.000   0.000
fit           4.904    0.011   4.916
finalise      0.819    0.001   0.819

Store(TModelGAM, TModelGAM_NS, lib = .Robjects)
```

Some notes:

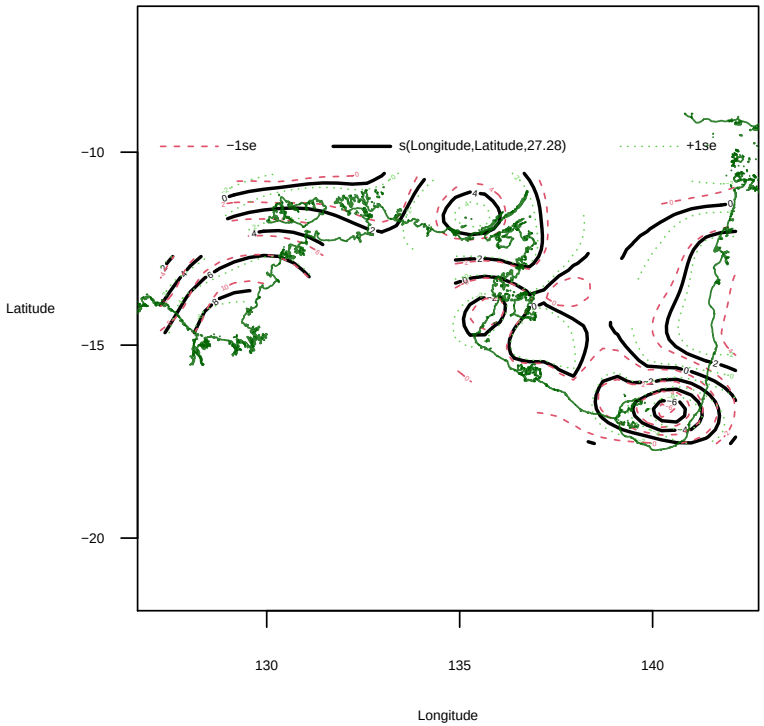
- An *isotropic* spatial term in *Longitude* and *Latitude* to account for purely spatial effects;
- Tensor spline (non-isotropic) terms in each pair of *DayOfYear*, *Depth* and *Sea* to account for temporal and environmental effects;
- A smooth term in *Mud*, also for environmental effects;
- The terms in *DayOfYear* are periodic with period 365 days (guaranteed by the data) in that coordinate *bs = "cc"*;
- Other terms use a smooth spline basis, *bs = "cs"*. Other choices of bases are available.

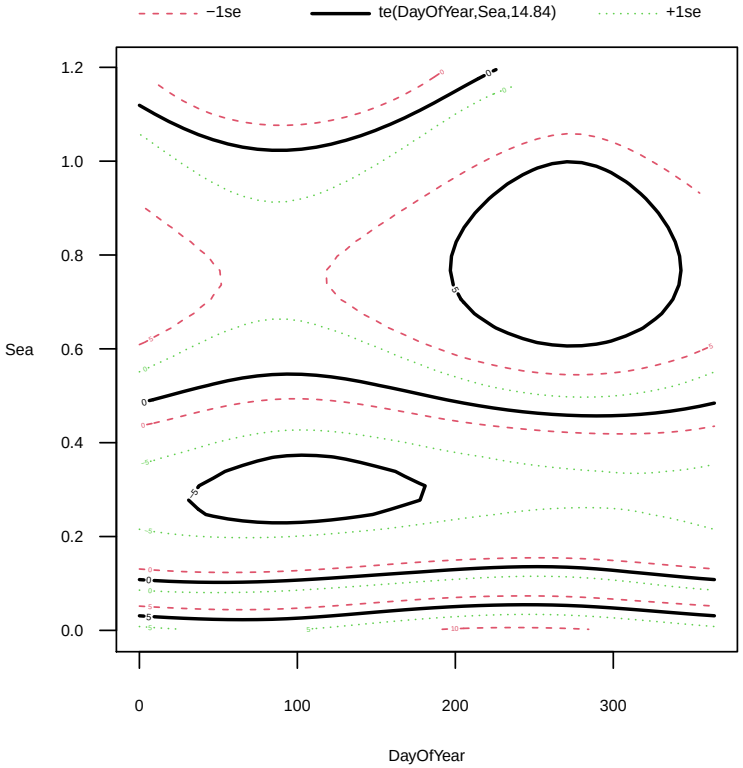
Some views of the fit:

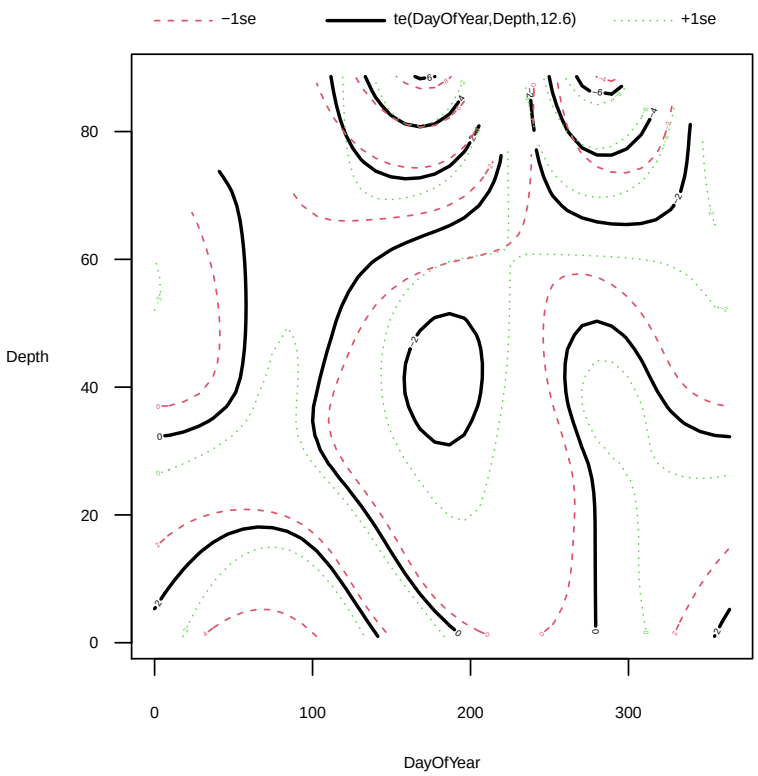
```
layout(rbind(1:2))
plot(TModelGAM_NS, select = 1, asp = 1)
lines(Oz, col = alpha("dark green", 0.8))
title(main = "Lon x Lat, isotropic")
for(j in 2:6)
    plot(TModelGAM_NS, select = j)
```

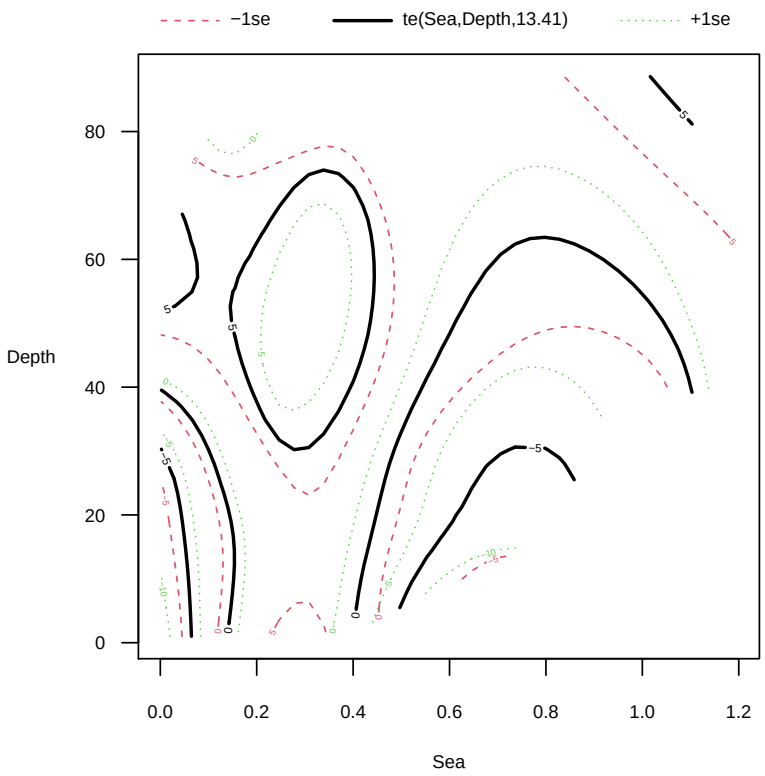
```
layout(rbind(1:2))
vis.gam(TModelGAM_NS, view = c("Longitude","Latitude"),
        r = 1000, theta = 30, phi = 15)
title(main = "Lon x Lat, isotropic")
vis.gam(TModelGAM_NS, view = c("DayOfYear","Sea"),
        r = 1000, theta = 30, phi = 15)
title(main = "Day of year x Sea")
vis.gam(TModelGAM_NS, view = c("DayOfYear","Depth"),
        r = 1000, theta = 30, phi = 15)
title(main = "Day of year x Depth")
vis.gam(TModelGAM_NS, view = c("Depth","Mud"),
        r = 1000, theta = 30, phi = 15)
title(main = "Depth x Mud")
```

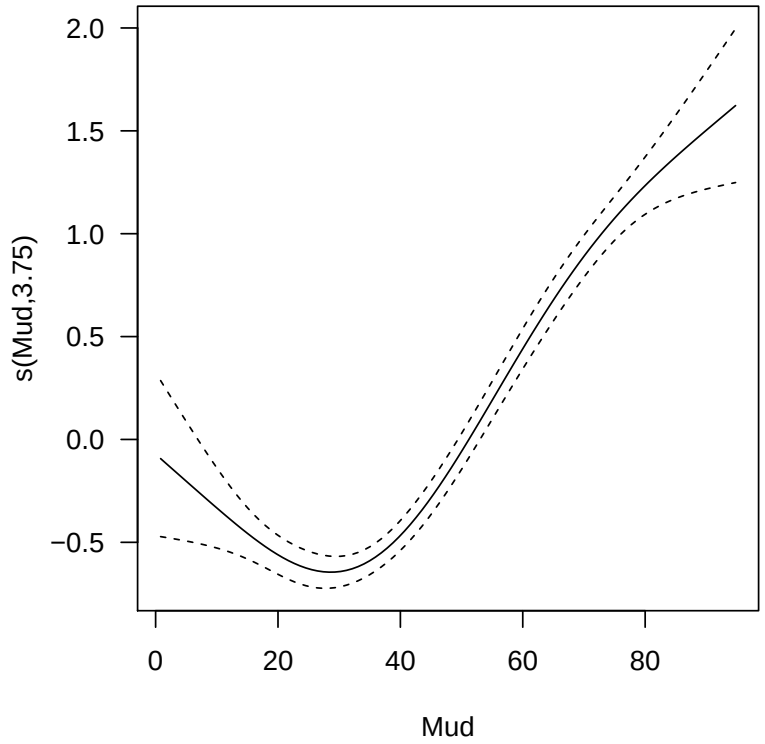
Lon x Lat, isotropic



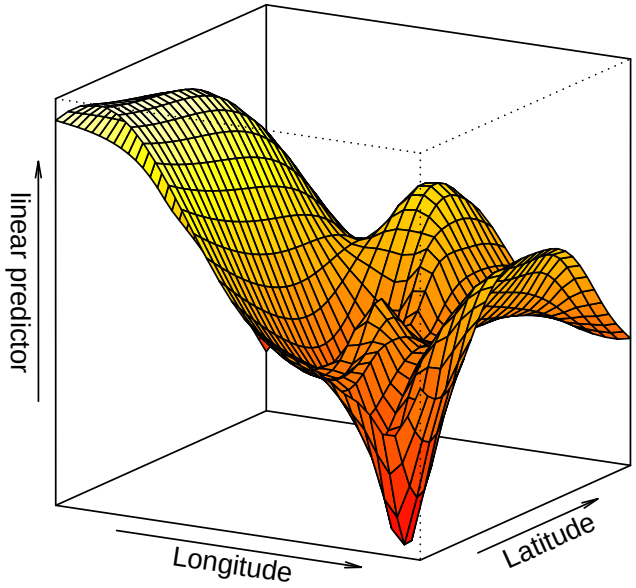



−1se
te(DayOfYear,Depth,12.6)
+1se
Depth
DayOfYear
0
20
40
60
80
0
100
200
300

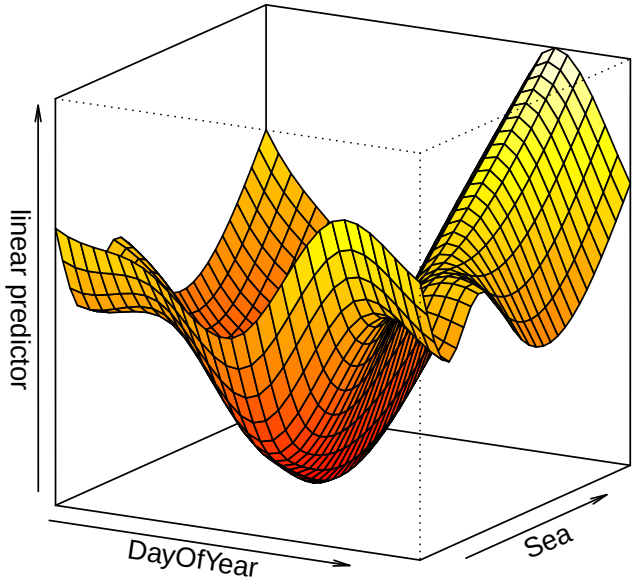

−1se
te(Sea,Depth,13.41)
+1se
Depth
Sea
0
20
40
60
80
0.0
0.2
0.4
0.6
0.8
1.0
1.2


2.0
1.5
1.0
0.5
0.0
−0.5
s(Mud,3.75)
0
20
40
60
80
Mud

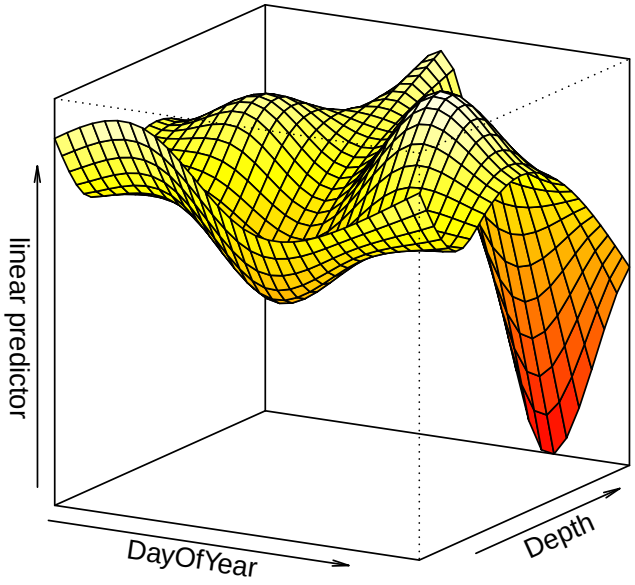
Lon x Lat, isotropic



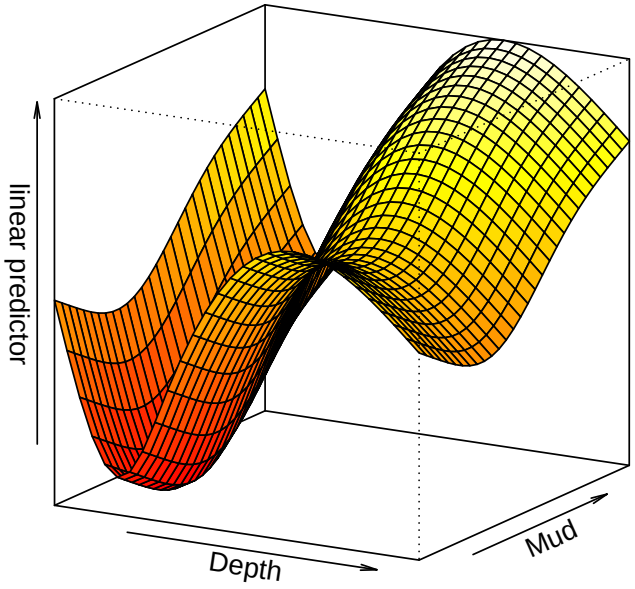
Day of year x Sea



Day of year x Depth

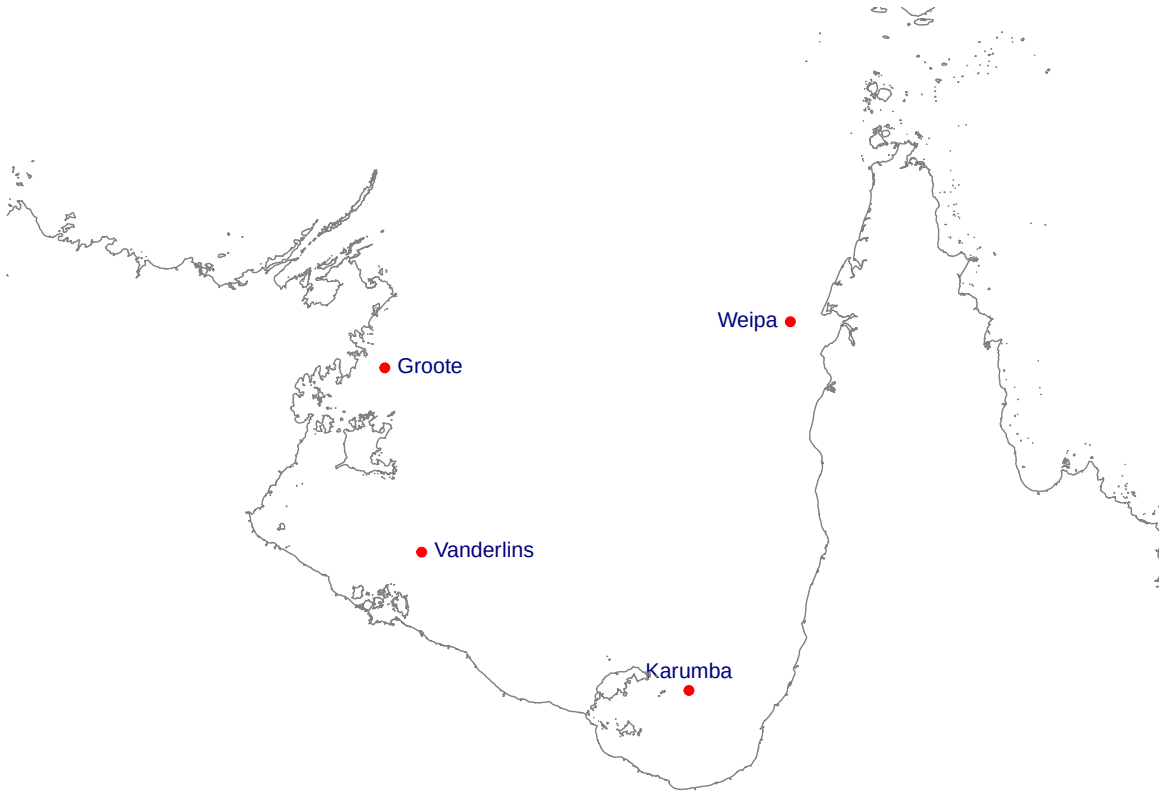


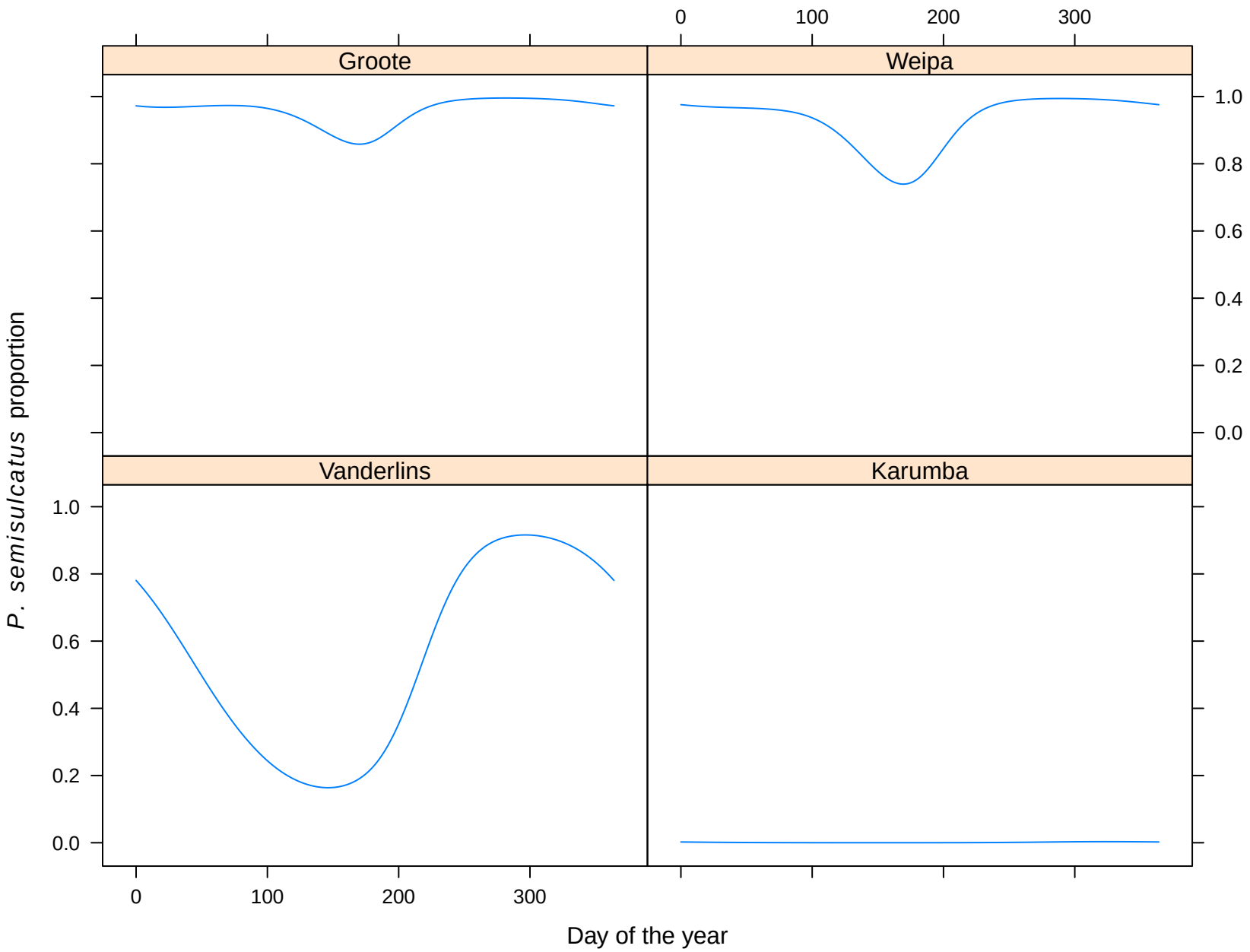
Depth x Mud



3.4 The spatio-temporal effect

Finally, we look at daily predictions for one year at 4 locations within the Gulf of Carpentaria:




Groote
Weipa
Vanderlins
Karumba
P. semisulcatus proportion
Day of the year
0
100
200
300
0.0
0.2
0.4
0.6
0.8
1.0

Note that the migration effect is strongest in the Vanderlins islands region, where the Tiger prawn catch is high.

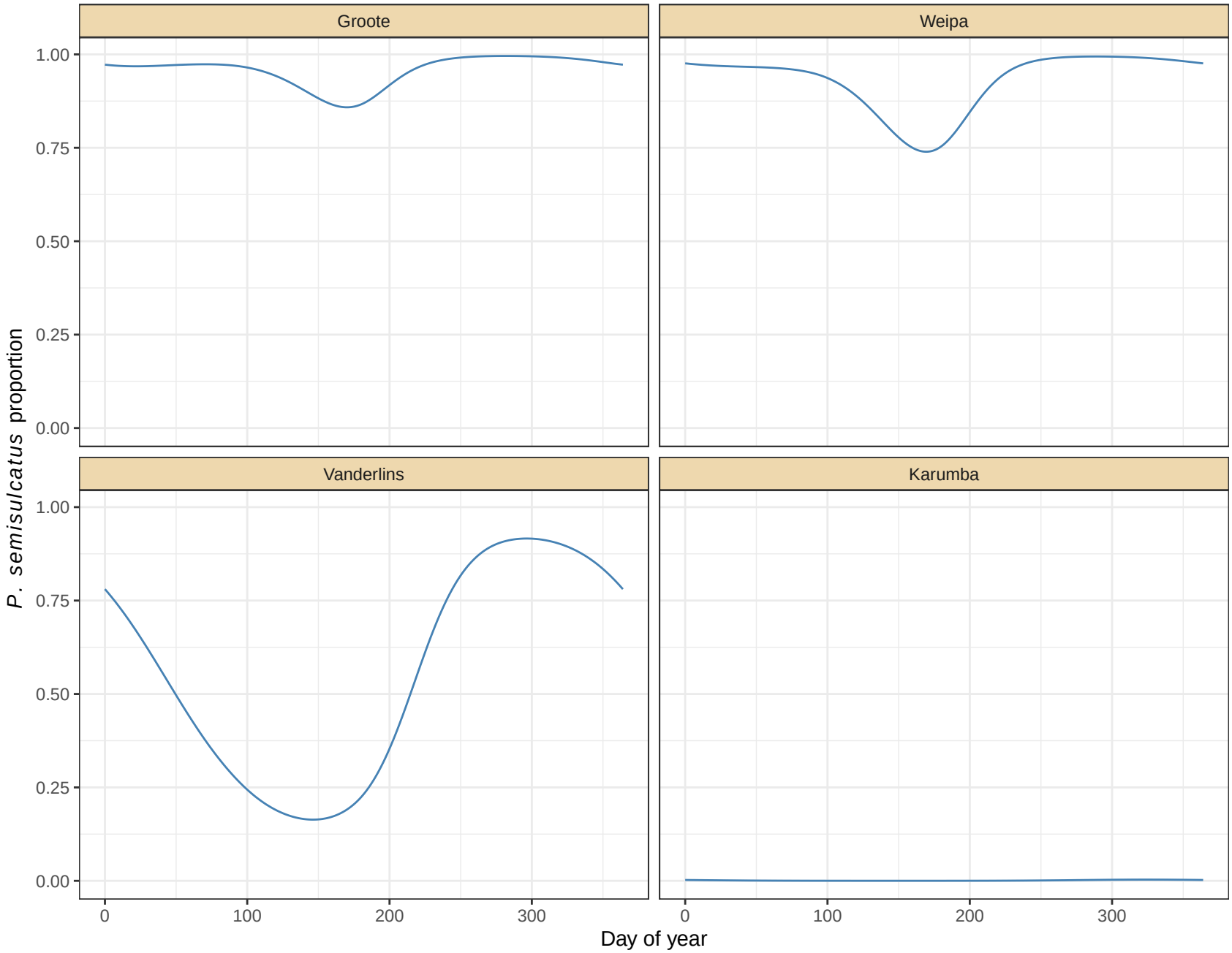
The prediction code is not shown, but the results are stored in a data frame called *FSemiData4*. The graphic is generated by:

```
Attach()
require(lattice)
trellis.par.set("grid.pars", list(fontfamily = "sans"))
xyplot(Fsemi ~ DayOfYear|Place, FSemiData4, type = "l",
       ylab=expression(italic(P.)~~italic(semisulcatus)~~plain(proportion)),
       xlab = "Day of the year", aspect = 0.7)
```

Note the device for mixed fonts in the annotations.

A *ggplot* version could be

```
ggplot(FSemiData4) + aes(x = DayOfYear, y = Fsemi) +
  geom_line(colour = "steelblue") +
  labs(y = expression(italic(P.)~~italic(semisulcatus)~~plain(proportion)),
       x = "Day of year") + facet_wrap(~Place, as.table = FALSE) +
  theme_bw() + theme(strip.background = element_rect(fill = "#EED8AE"))  # "#FFE5C
```


Groote
Weipa
Vanderlins
Karumba
P. semisulcatus proportion
1.00
0.75
0.50
0.25
0.00
0
100
200
300
Day of year

References

Marschner, I. C. (2011, December). glm2: Fitting generalized linear models with convergence problems. *The* **R** *Journal 3*(2), 12–15.

Venables, W. N. and B. D. Ripley (2002). *Modern Applied Statistics with* **S** (Fourth ed.). New York: Springer. ISBN 0-387-95457-0.

Session information

Date: 2021-01-29

- R version 4.0.3 (2020-10-10), `x86_64-pc-linux-gnu`
- Running under: `Ubuntu 20.04.1 LTS`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0`
- LAPACK: `/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: dplyr 1.0.3, english 1.2-5, forcats 0.5.1, ggplot2 3.3.3, ggthemes 4.2.4, gridExtra 2.3, knitr 1.31, lattice 0.20-41, mgcv 1.8-33, nlme 3.1-151, patchwork 1.1.1, purrr 0.3.4, readr 1.4.0, scales 1.1.1, stringr 1.4.0, tibble 3.0.5, tidyr 1.1.2, tidyverse 1.3.0, visreg 2.7.0, WWRCourse 0.2.3, WWRData 0.1.0, WWRGraphics 0.1.2, WWRUtilities 0.1.2, xtable 1.8-4
- Loaded via a namespace (and not attached): assertthat 0.2.1, backports 1.2.1, broom 0.7.3, cellranger 1.1.0, cli 2.2.0, colorspace 2.0-0, compiler 4.0.3, crayon 1.3.4, DBI 1.1.1, dbplyr 2.0.0, digest 0.6.27, ellipsis 0.3.1, evaluate 0.14, fansi 0.4.2, farver 2.0.3, fractional 0.1.3, fs 1.5.0, generics 0.1.0, glue 1.4.2, grid 4.0.3, gtable 0.3.0, haven 2.3.1, highr 0.8, hms 1.0.0, httr 1.4.2, iterators 1.0.13, jsonlite 1.7.2, labeling 0.4.2, lazyData 1.1.0, lifecycle 0.2.0, lubridate 1.7.9.2, magrittr 2.0.1, MASS 7.3-53, Matrix 1.3-2, modelr 0.1.8, munsell 0.5.0, parallel 4.0.3, PBSmapping 2.73.0, pillar 1.4.7, pkgconfig 2.0.3, R6 2.5.0, randomForest 4.6-14, Rcpp 1.0.6, readxl 1.3.1, reprex 1.0.0, rlang 0.4.10, rpart 4.1-15, rstudioapi 0.13, rvest 0.3.6, SOAR 0.99-11, splines 4.0.3, stringi 1.5.3, tidyselect 1.1.0, tools 4.0.3, vctrs 0.3.6, withr 2.4.1, xfun 0.20, xml2 1.3.2