Learn Generalised Additive (Mixed) Models

Dr Stefano Coretta

University of Edinburgh

2023/05/18

Outline

- Part I: Introduction to GAMMs
  - What are GAMMs?
  - GAMMs basics
  - Comparing groups and between groups (interactions)
  - Random effects
- Break
- Part II: Hands-on

PART I

Generalised additive models

- **G**enrealised **A**dditive **M**odel**s** (GAMs)
- $y = f(x)$
  - $f(x)$ = some function of x (or *smooth function*)

Smooth terms

LMs have only **parametric terms**

- `f0 ~ vowel + voicing + duration`
- Parametric terms fit linear effects.

GAMs add (non-parametric) **smooth terms** (or simply smooths, also smoothers):

- `f0 ~ vowel + voicing + s(duration)`
- `f(x)`: *some function of* x.
- Smooth terms fit non-linear effects.

```
library(mgcv)
gam(y ~ s(x), data)
```

The model: y as *some* function of x

Pupil size

- **Pupillometry data** from English young and older adults (McLaughlin et al 2022, https://doi.org/10.3758/s13423-021-01991-0). In Arbitrary Units (AU).
- **Word recognition task** (verbal stimulus + verbal response).
- Words with **sparse and dense neighbourhood** density.
- **Hypotheses**:
  - Recognizing words with more competitors (dense neighbourhood) should come at a greater cognitive cost (greater pupil size) relative to recognizing words with fewer competitors (sparse neighbourhood).
  - The cognitive demands associated with increased neighbourhood density (greater pupil size) should be greater for older adults compared with young adults.

Pupil size

- The original study used Growth Curve Analysis (GCA).
- We will apply GAMs instead.

- CAVEAT: We are analysing the whole time course, rather than just a subset as done in the original study.

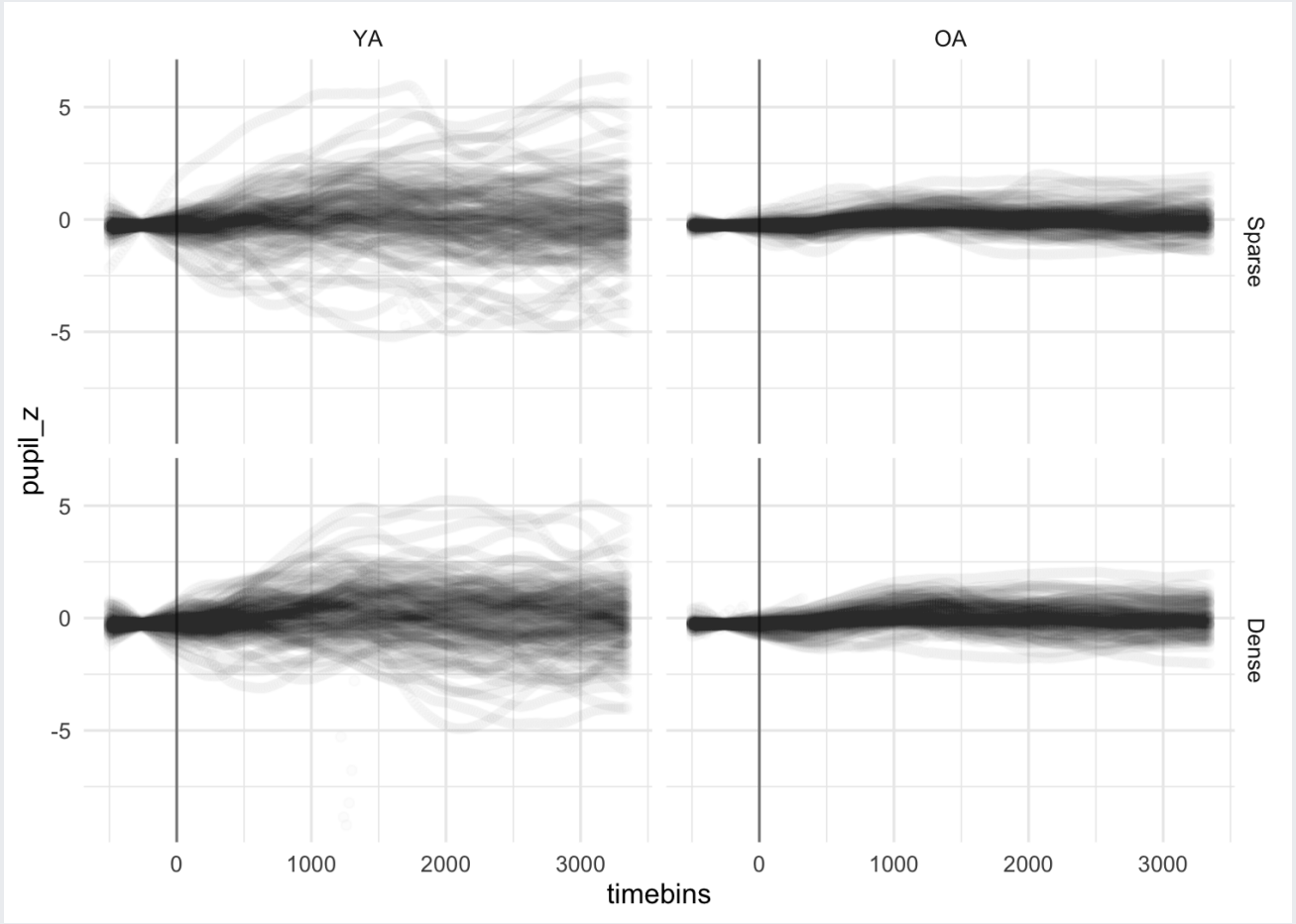
Pupil size

```
pdq_20 <- readRDS("data/pdq_20.rds") %>%
  mutate(
    Condition = factor(Condition, levels = c("Sparse", "Dense")),
    Age = factor(Age, levels = c("YA", "OA")),
    pupil_z = (pupil.binned - mean(pupil.binned)) / sd(pupil.binned)
  )

pdq_20
```

```
## # A tibble: 88,008 × 8
##    subject trial Condition Age   timebins Soundfile        pupil.binned  pupil_z
##      <dbl> <dbl> <fct>     <fct>    <dbl> <chr>                   <dbl>    <dbl>
##  1       1     1 Sparse    YA        -500 NAMword_675_Mul…       84.5    0.00347
##  2       1     1 Sparse    YA        -480 NAMword_675_Mul…       75.8   -0.0239
##  3       1     1 Sparse    YA        -460 NAMword_675_Mul…       65.4   -0.0570
##  4       1     1 Sparse    YA        -440 NAMword_675_Mul…       54.3   -0.0922
##  5       1     1 Sparse    YA        -420 NAMword_675_Mul…       35.7   -0.151
##  6       1     1 Sparse    YA        -400 NAMword_675_Mul…       20.2   -0.200
##  7       1     1 Sparse    YA        -380 NAMword_675_Mul…        8.72  -0.237
##  8       1     1 Sparse    YA        -360 NAMword_675_Mul…        0.680 -0.262
##  9       1     1 Sparse    YA        -340 NAMword_675_Mul…      -11.4   -0.300
## 10       1     1 Sparse    YA        -320 NAMword_675_Mul…      -23.3   -0.338
## # ℹ 87,998 more rows
```

Pupil size



A simple GAM

```
library(mgcv)

pdq_gam <- gam(
  # Outcome
  pupil_z ~
    # Smooth over timebins
    s(timebins),
  data = pdq_20
)
```

A simple GAM

```
summary(pdq_gam)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## pupil_z ~ s(timebins)
##
## Parametric coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.271e-14  3.316e-03       0        1
##
## Approximate significance of smooth terms:
##               edf Ref.df     F p-value
## s(timebins) 7.891  8.679 334.7  <2e-16
##
## R-sq.(adj) =  0.0321   Deviance explained = 3.21%
## GCV = 0.96804  Scale est. = 0.96794   n = 88008
```

- The parametric term's coefficient is an estimate of the mean height.
- The smooth's EDFs (estimated degrees of freedom) indicate whether it is a straight line (EDF = 1) or not.

A simple GAM

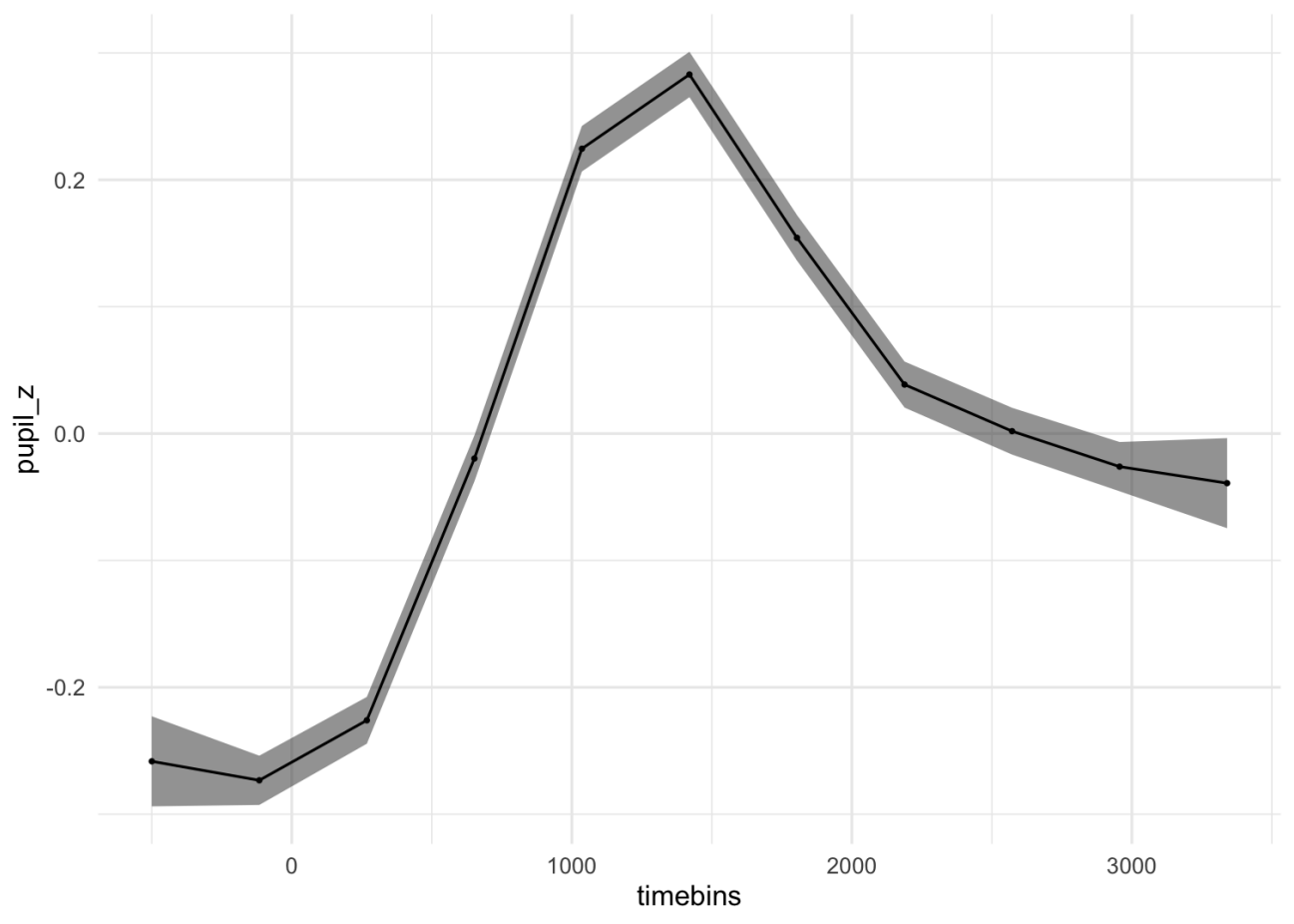
```
library(tidygam)

predict_gam(pdq_gam)
```

```
## # A tibble: 11 × 5
##    timebins   pupil_z         se  lower_ci  upper_ci
##       <dbl> <dbl[1d]> <dbl[1d]> <dbl[1d]> <dbl[1d]>
##  1     -500  -0.258     0.0181    -0.294   -0.223
##  2     -116  -0.273     0.00990   -0.293   -0.254
##  3      268  -0.226     0.00941   -0.244   -0.208
##  4      652  -0.0196    0.00925   -0.0378  -0.00152
##  5     1036   0.224     0.00916    0.207    0.242
##  6     1420   0.283     0.00912    0.265    0.301
##  7     1804   0.154     0.00916    0.136    0.172
##  8     2188   0.0386    0.00925    0.0205   0.0568
##  9     2572   0.00188   0.00941   -0.0166   0.0203
## 10     2956  -0.0261    0.00990   -0.0455  -0.00666
## 11     3340  -0.0391    0.0181    -0.0746  -0.00359
```

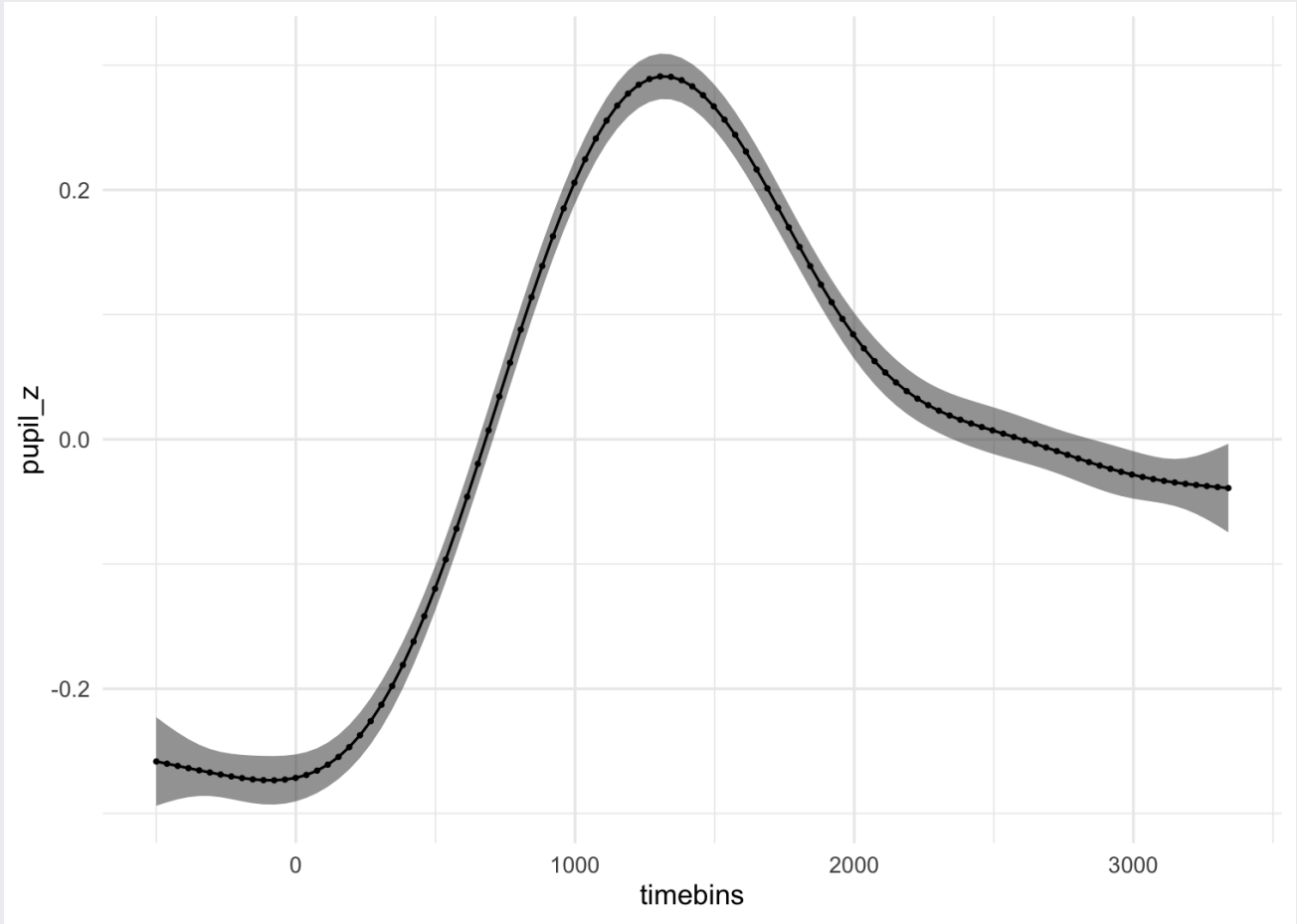
A simple GAM

```
predict_gam(pdq_gam) %>% plot(series = "timebins")
```



A simple GAM

```
predict_gam(pdq_gam, length_out = 100) %>% plot(series = "timebins")
```



Number of knots k

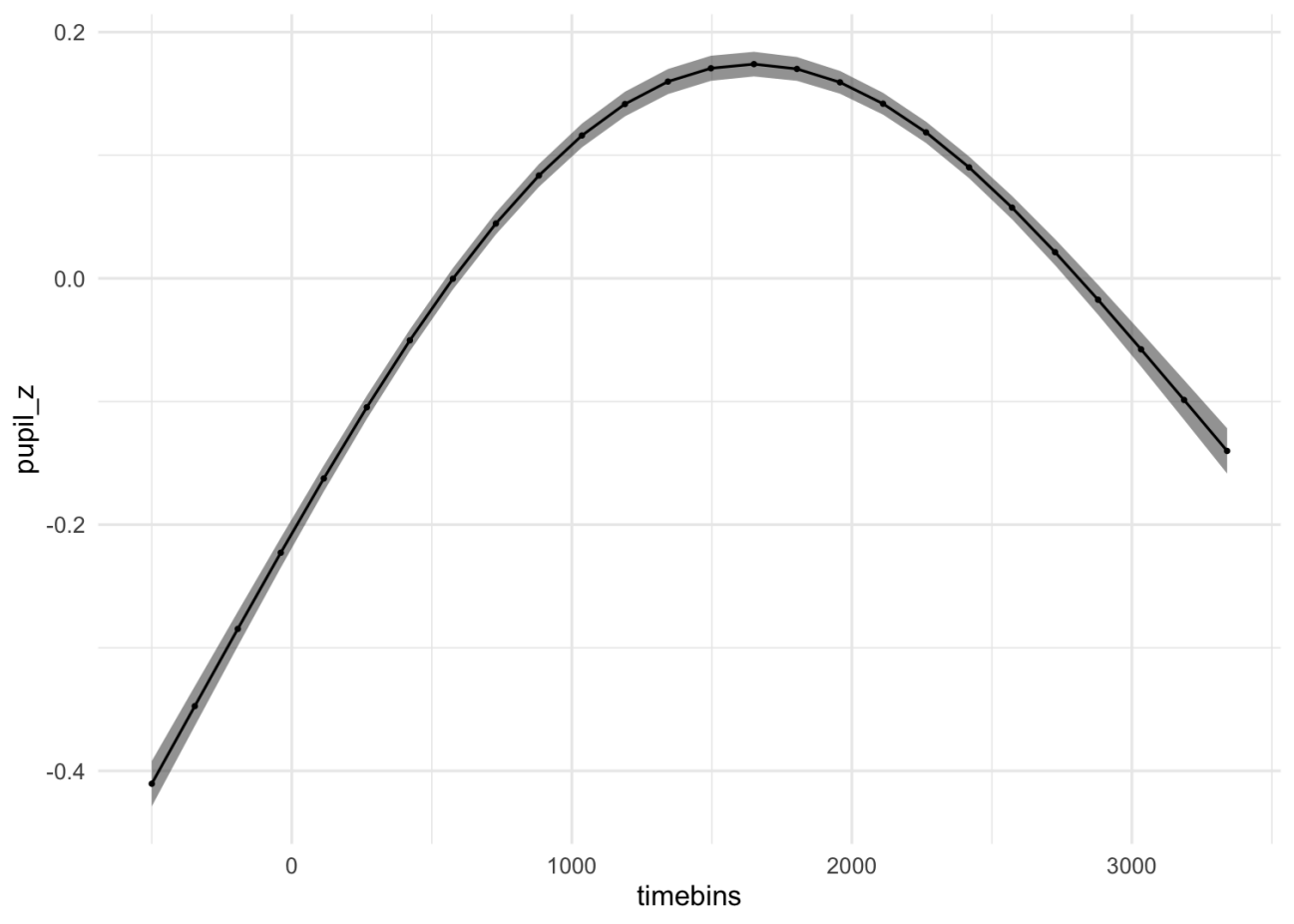
- The "wiggliness" of the resulting spline is partially constrained by the number of *knots* (`k`).
- The more knots, the more wiggly the spline can be. Or the more knots the less smooth the spline can be.

- You can set the number of knots `k` with the argument `k` in `s()`.
- `k` cannot be larger than the number of "sampling points" in the variable to smooth over.

```
# Use bam(): Big gAM
pdq_gam_2 <- bam(
  pupil_z ~
    s(timebins, k = 3),
  data = pdq_20
)
```

Number of knots k

```
predict_gam(pdq_gam_2, length_out = 25) %>% plot(series = "timebins")
```

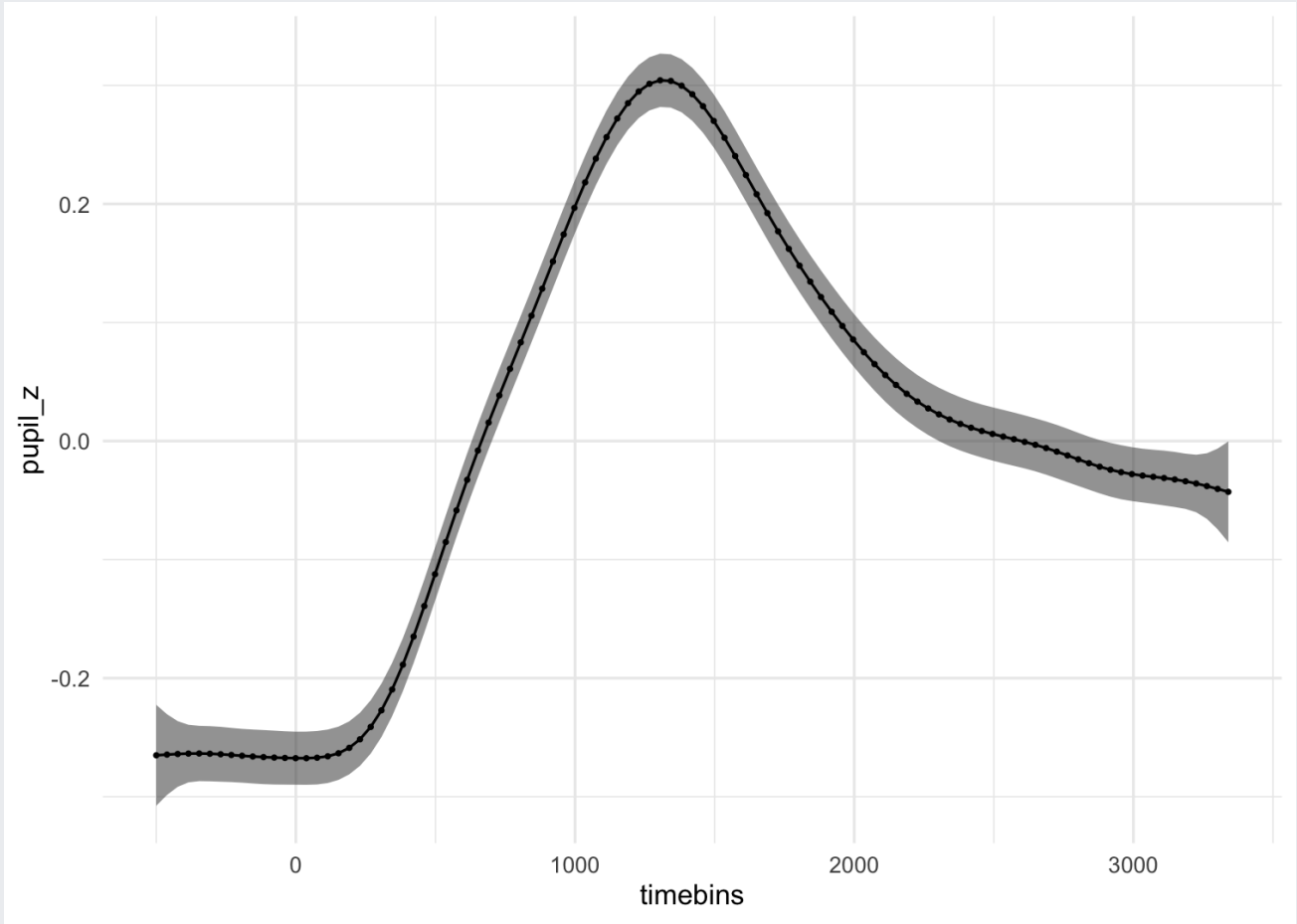


Number of knots k

```
pdq_gam_2 <- bam(
  pupil_z ~
    s(timebins, k = 20),
  data = pdq_20
)
```

Number of knots k

```
predict_gam(pdq_gam_2, length_out = 100) %>% plot(series = "timebins")
```



Comparing groups

- Comparing levels from a variable (like age: young vs old) can be achieved with the **by-variable method**,
  - i.e. by specifying the variable as the value of the `by` argument in `s()`.

```
pdq_gam_3a <- bam(
  pupil_z ~
    s(timebins, by = Age, k = 20),
  data = pdq_20
)
```

Comparing groups

```
summary(pdq_gam_3a)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## pupil_z ~ s(timebins, by = Age, k = 20)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.001158   0.003315   0.349    0.727
##
## Approximate significance of smooth terms:
##                      edf Ref.df      F p-value
## s(timebins):AgeYA 11.32  13.71 144.31  <2e-16
## s(timebins):AgeOA 10.14  12.40  82.94  <2e-16
##
## R-sq.(adj) =  0.033   Deviance explained = 3.33%
## fREML = 1.2344e+05  Scale est. = 0.96698   n = 88008
```

But what about comparing YA and OA?

Comparing groups

To compare levels to a reference level:

- Change factor to an **ordered factor**.
- Change factor contrasts to **treatment contrasts** (`contr.treatment`).
  - The default in ordered factors is `contr.poly`, this won't work.
- Include the factor as a **parametric term**.
- Include a **reference smooth** and a **difference smooth** with the `by`-variable.

Comparing groups

```
pdq_20 <- pdq_20 %>%
  mutate(
    # Make the variables into an ordered factor
    Condition_o = as.ordered(Condition),
    Age_o = as.ordered(Age)
  )

# Change the contrasts to treatment
contrasts(pdq_20$Condition_o) <- "contr.treatment"
contrasts(pdq_20$Age_o) <- "contr.treatment"
```

Comparing groups

Let's start with `Age_o`.

```
pdq_gam_3 <- bam(
  pupil_z ~
    # Parametric term
    Age_o +
    # Reference smooth (Age_0 == "YA")
    s(timebins, k = 20) +
    # Difference smooth ("OA" - "YA")
    s(timebins, by = Age_o, k = 20),
  data = pdq_20
)
```

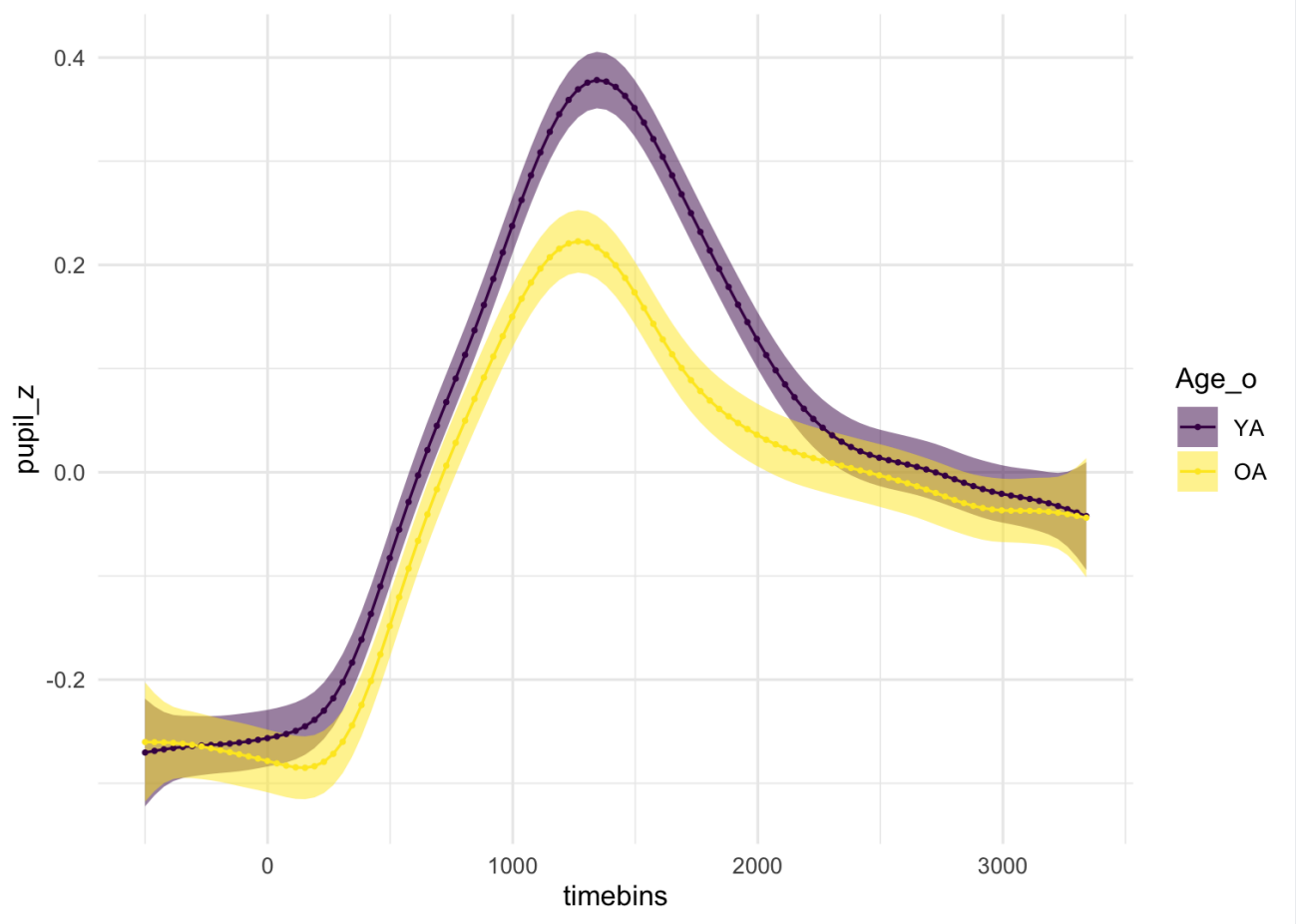
Comparing groups

```
## Parametric coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.029495   0.004512   6.537  6.3e-11
## Age_oOA     -0.061473   0.006648  -9.247  < 2e-16
##
## Approximate significance of smooth terms:
##                          edf Ref.df       F p-value
## s(timebins)           11.985 14.367 138.640  <2e-16
## s(timebins):Age_oOA  7.397  9.139   8.446  <2e-16
```

- `Intercept`: mean height when Age == "YA".
- `Age_oOA`: height difference between "OA" and "YA".
- `s(timebins)`: EDF > 1 = trajectory is not a straight line.
- `s(timebins):Age_oOA`: EDF > 1 = trajectory of "OA" is different from "YA".

Comparing groups

```
predict_gam(pdq_gam_3, length_out = 100) %>% plot(series = "timebins", comparison = "Age_o")
```



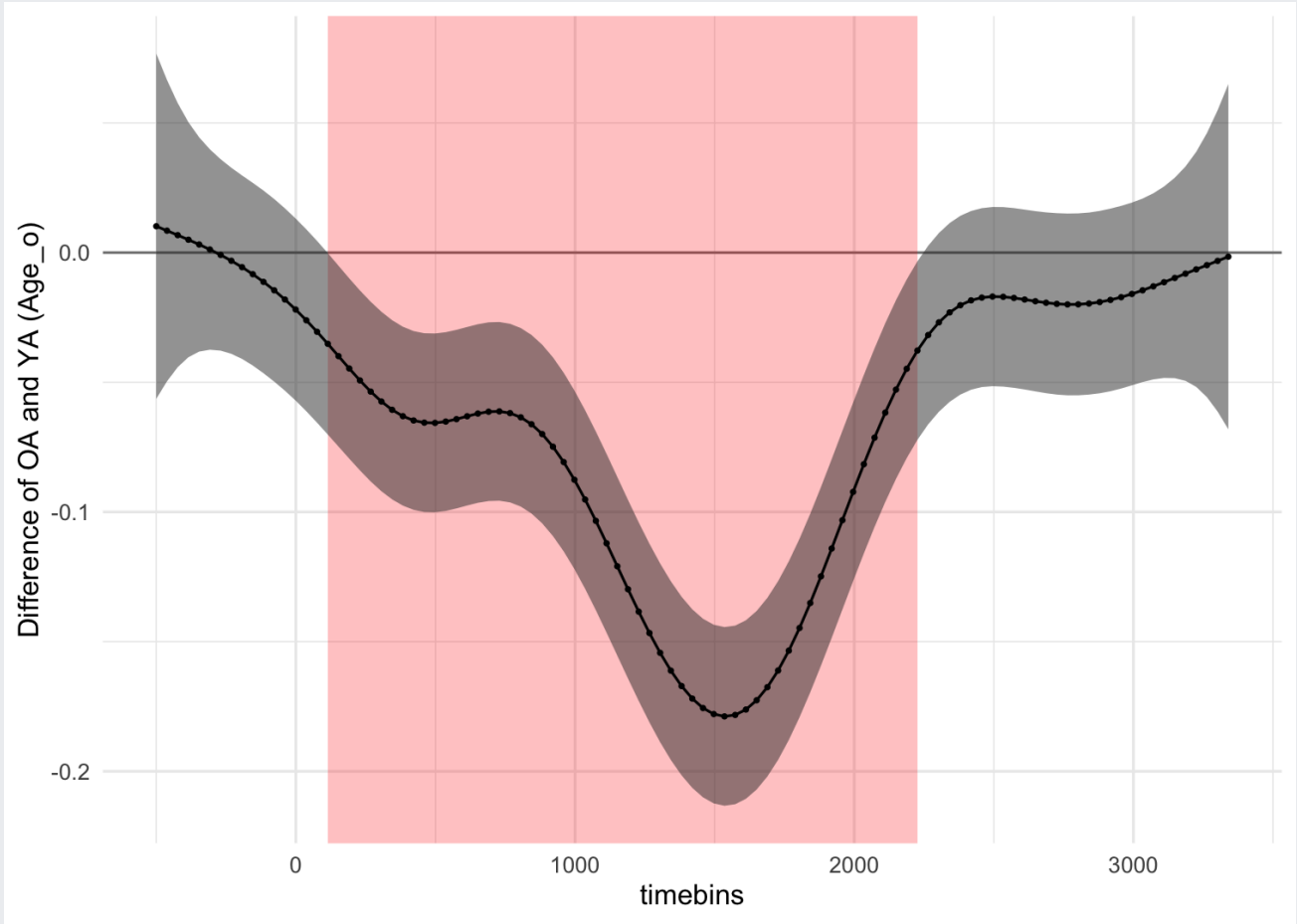
Comparing groups

```
pdq_gam_3_diff <- get_difference(
  pdq_gam_3, series = "timebins", length_out = 100,
  compare = list(Age_o = c("OA", "YA"))
)
pdq_gam_3_diff
```

```
## # A tibble: 101 × 6
##    Age_o timebins      diff     se lower_ci upper_ci
##    <chr>    <dbl>     <dbl>  <dbl>    <dbl>    <dbl>
##  1 OA-YA     -500   0.0102   0.0340  -0.0564   0.0767
##  2 OA-YA     -462.  0.00845  0.0296  -0.0496   0.0666
##  3 OA-YA     -423.  0.00673  0.0260  -0.0442   0.0576
##  4 OA-YA     -385.  0.00496  0.0231  -0.0403   0.0503
##  5 OA-YA     -346.  0.00313  0.0211  -0.0382   0.0444
##  6 OA-YA     -308   0.00119  0.0197  -0.0374   0.0398
##  7 OA-YA     -270. -0.000898 0.0188  -0.0378   0.0360
##  8 OA-YA     -231. -0.00316  0.0183  -0.0390   0.0327
##  9 OA-YA     -193. -0.00562  0.0180  -0.0410   0.0297
## 10 OA-YA     -154. -0.00833  0.0180  -0.0435   0.0269
## # ℹ 91 more rows
```

Comparing groups

```
pdq_gam_3_diff %>% plot()
```



Random effects

Only **fixed effects** so far...

- Parametric terms.
- Smooth terms.

Generalised **A**dditive **M**ixed **M**odels (GAMMs).

Two ways of including random effects:

- Use the `"re"` basis function (`bs` argument in `s()`) for random intercept and slopes.
- Include a **random smooth** term with the **factor smooth interaction** as basis (`bs = "fs"`).

Random effects

Factor smooth interaction:

- Specified with `bs = "fs"` in `s()`.
- A smooth is fitted at each level of a factor.
- NOTE: it has *interaction* in the name but has nothing to do with interactions.

The random effect variable *needs to be a factor*.

Random effects

Let's change `subject` to a factor (no need to make it an ordered factor).

```
pdq_20 <- pdq_20 %>%
  mutate(
    subject = as.factor(subject)
  )
pdq_20
```

```
## # A tibble: 88,008 × 10
##    subject trial Condition Age   timebins Soundfile        pupil.binned  pupil_z
##    <fct>   <dbl> <fct>     <fct>    <dbl> <chr>                   <dbl>    <dbl>
##  1 1           1 Sparse    YA        -500 NAMword_675_Mul…       84.5    0.00347
##  2 1           1 Sparse    YA        -480 NAMword_675_Mul…       75.8   -0.0239
##  3 1           1 Sparse    YA        -460 NAMword_675_Mul…       65.4   -0.0570
##  4 1           1 Sparse    YA        -440 NAMword_675_Mul…       54.3   -0.0922
##  5 1           1 Sparse    YA        -420 NAMword_675_Mul…       35.7   -0.151
##  6 1           1 Sparse    YA        -400 NAMword_675_Mul…       20.2   -0.200
##  7 1           1 Sparse    YA        -380 NAMword_675_Mul…        8.72  -0.237
##  8 1           1 Sparse    YA        -360 NAMword_675_Mul…        0.680 -0.262
##  9 1           1 Sparse    YA        -340 NAMword_675_Mul…      -11.4   -0.300
## 10 1           1 Sparse    YA        -320 NAMword_675_Mul…      -23.3   -0.338
## # ℹ 87,998 more rows
## # ℹ 2 more variables: Condition_o <ord>, Age_o <ord>
```

Random effects

```
pdq_gam_4 <- bam(
  pupil_z ~
    # Paramteric term
    Age_o +
    # Reference smooth
    s(timebins, k = 20) +
    # Difference smooth
    s(timebins, by = Age_o, k = 20) +
    # Factor smooth interaction by subject
    s(timebins, subject, bs = "fs", m = 1),
  data = pdq_20
)
```

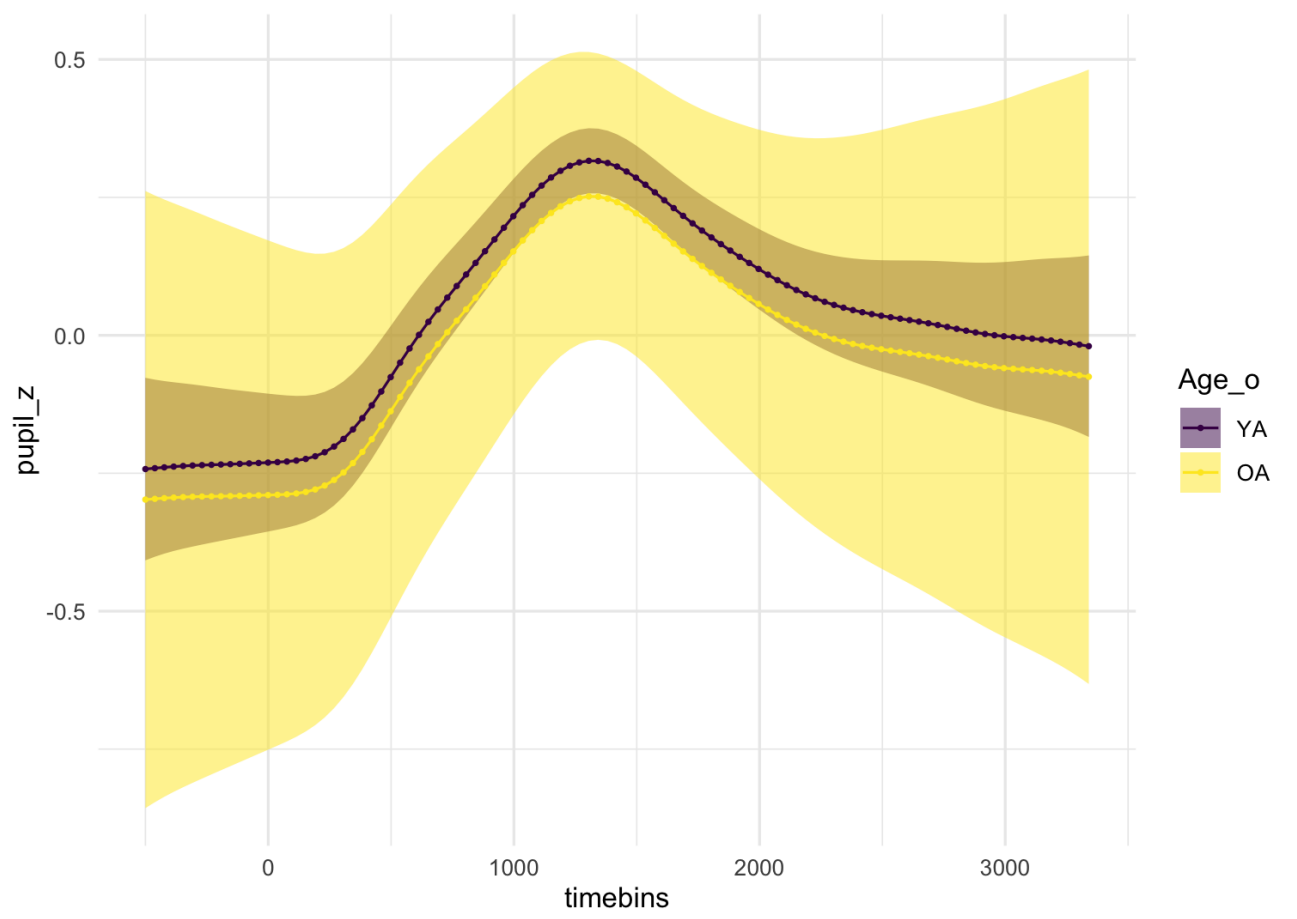
Random effects

```
summary(pdq_gam_4)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## pupil_z ~ Age_o + s(timebins, k = 20) + s(timebins, by = Age_o,
##     k = 20) + s(timebins, subject, bs = "fs", m = 1)
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.02904    0.06406   0.453    0.650
## Age_oOA     -0.06098    0.09061  -0.673    0.501
##
## Approximate significance of smooth terms:
##                           edf  Ref.df      F p-value
## s(timebins)           11.168  13.456  8.981  <2e-16
## s(timebins):Age_oOA    1.124   1.144  0.014   0.973
## s(timebins,subject) 144.184 178.000 33.754  <2e-16
##
## R-sq.(adj) =  0.0952   Deviance explained = 9.68%
## fREML = 1.2073e+05  Scale est. = 0.9048    n = 88008
```

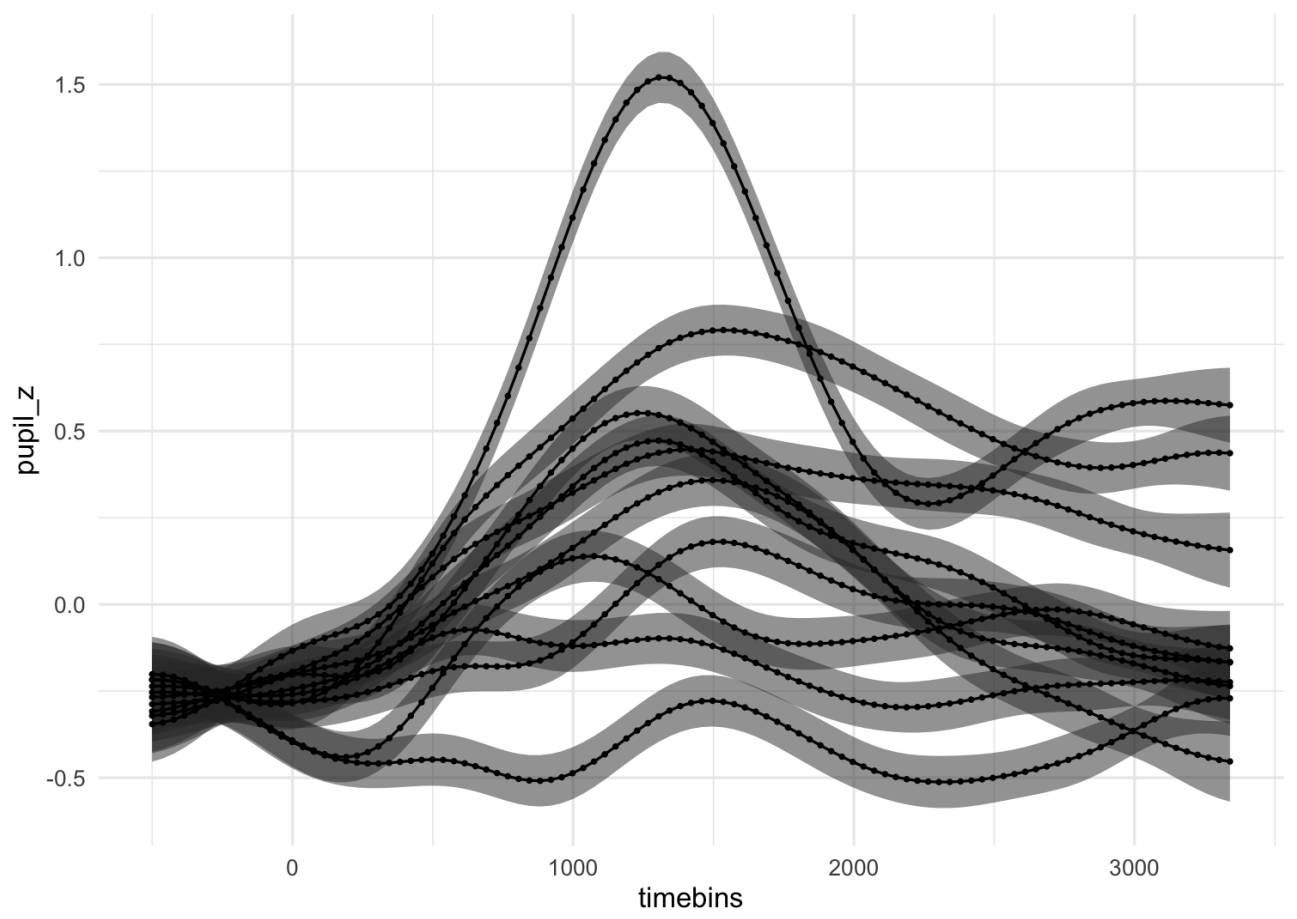
Random effects

```
predict_gam(pdq_gam_4, length_out = 100, exclude_terms = "s(timebins,subject)") %>%
  plot(series = "timebins", comparison = "Age_o")
```



Random effects

```
predict_gam(pdq_gam_4, length_out = 100, values = c(Age_o = "YA")) %>% # filter only YA subjects
  filter(subject %in% c(1:10)) %>% plot(series = "timebins")
```



Comparing across groups (interactions)

Technically, GAMs **don't allow interactions**.

- They are ADDITIVE (interactions require multiplication).

We can get interaction-like comparisons by creating **factor interactions** and using them as `by`-variables.

Comparing across groups (interactions)

- Let's create a factor interaction between `Age` and `Condition`.
- We also need to make it into an ordered factor with treatment contrasts.
- Note that the model will include this factor interaction (no need for `Age` and `Condition`).

```
pdq_20 <- pdq_20 %>%
  mutate(
    Age_Cond = as.ordered(interaction(Age, Condition))
  )

contrasts(pdq_20$Age_Cond) <- "contr.treatment"
```

Comparing across groups (interactions)

```
pdq_gam_5 <- bam(
  pupil_z ~
    # Paramteric term
    Age_Cond +
    # Reference smooth
    s(timebins, k = 20) +
    # Difference smooth
    s(timebins, by = Age_Cond, k = 20) +
    # Factor smooth interaction by subject
    s(timebins, subject, bs = "fs", m = 1),
  data = pdq_20
)
```

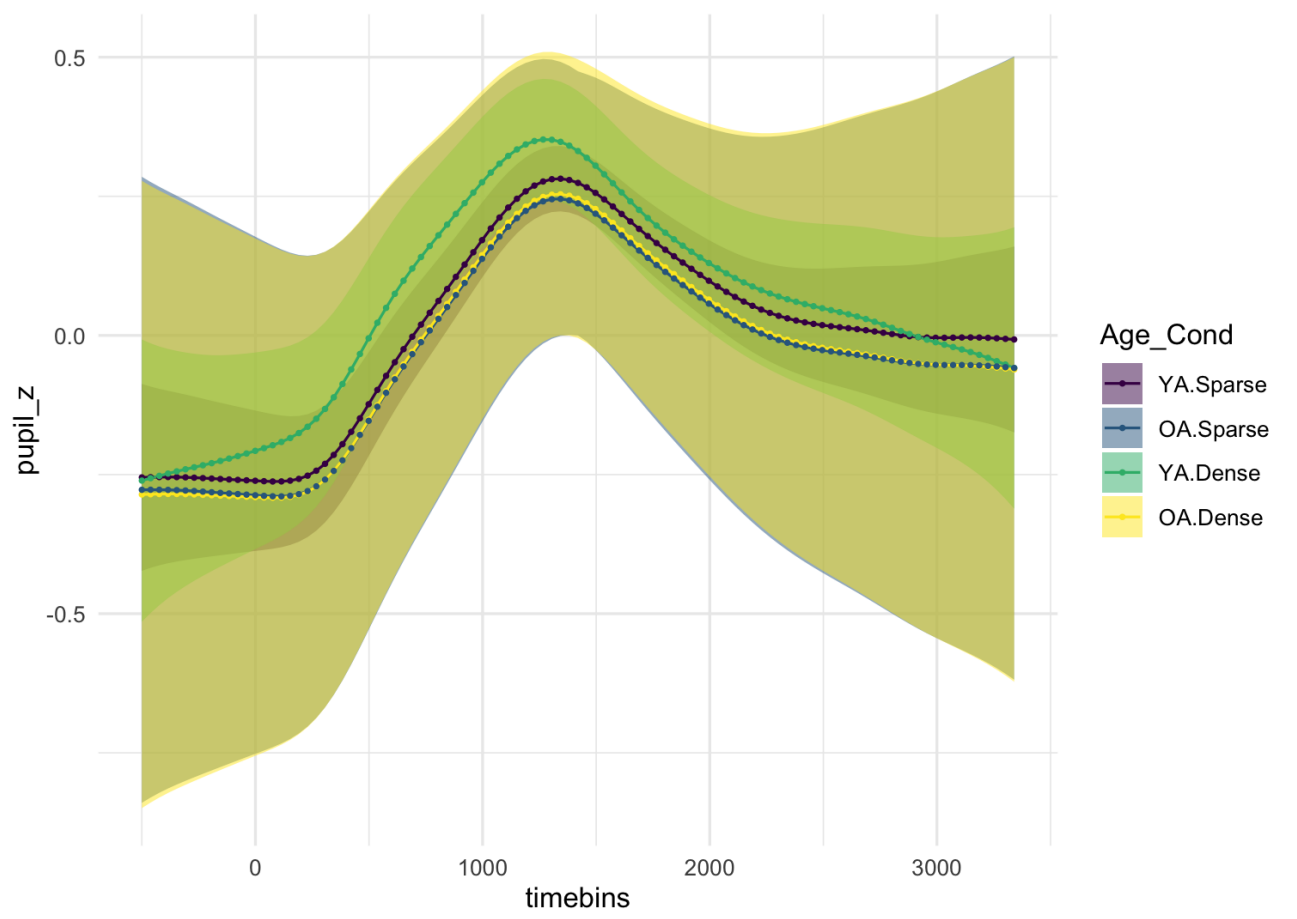
Comparing across groups (interactions)

```
summary(pdq_gam_5)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## pupil_z ~ Age_Cond + s(timebins, k = 20) + s(timebins, by = Age_Cond,
##     k = 20) + s(timebins, subject, bs = "fs", m = 1)
##
## Parametric coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         0.003526   0.064261   0.055    0.956
## Age_CondOA.Sparse -0.036832   0.090907  -0.405    0.685
## Age_CondYA.Dense    0.047145   0.008803   5.356 8.54e-08
## Age_CondOA.Dense  -0.034121   0.090892  -0.375    0.707
##
## Approximate significance of smooth terms:
##                                 edf  Ref.df      F  p-value
## s(timebins)                  11.184  13.466  8.625  < 2e-16
## s(timebins):Age_CondOA.Sparse   1.000   1.000  0.017    0.896
## s(timebins):Age_CondYA.Dense    4.828   6.013  4.894 5.43e-05
## s(timebins):Age_CondOA.Dense    1.445   1.757  0.245    0.748
## s(timebins,subject)           144.284 178.000 33.809  < 2e-16
##
```

Comparing across groups (interactions)

```
predict_gam(pdq_gam_5, length_out = 100, exclude_terms = "s(timebins,subject)") %>%
  plot(series = "timebins", comparison = "Age_Cond")
```

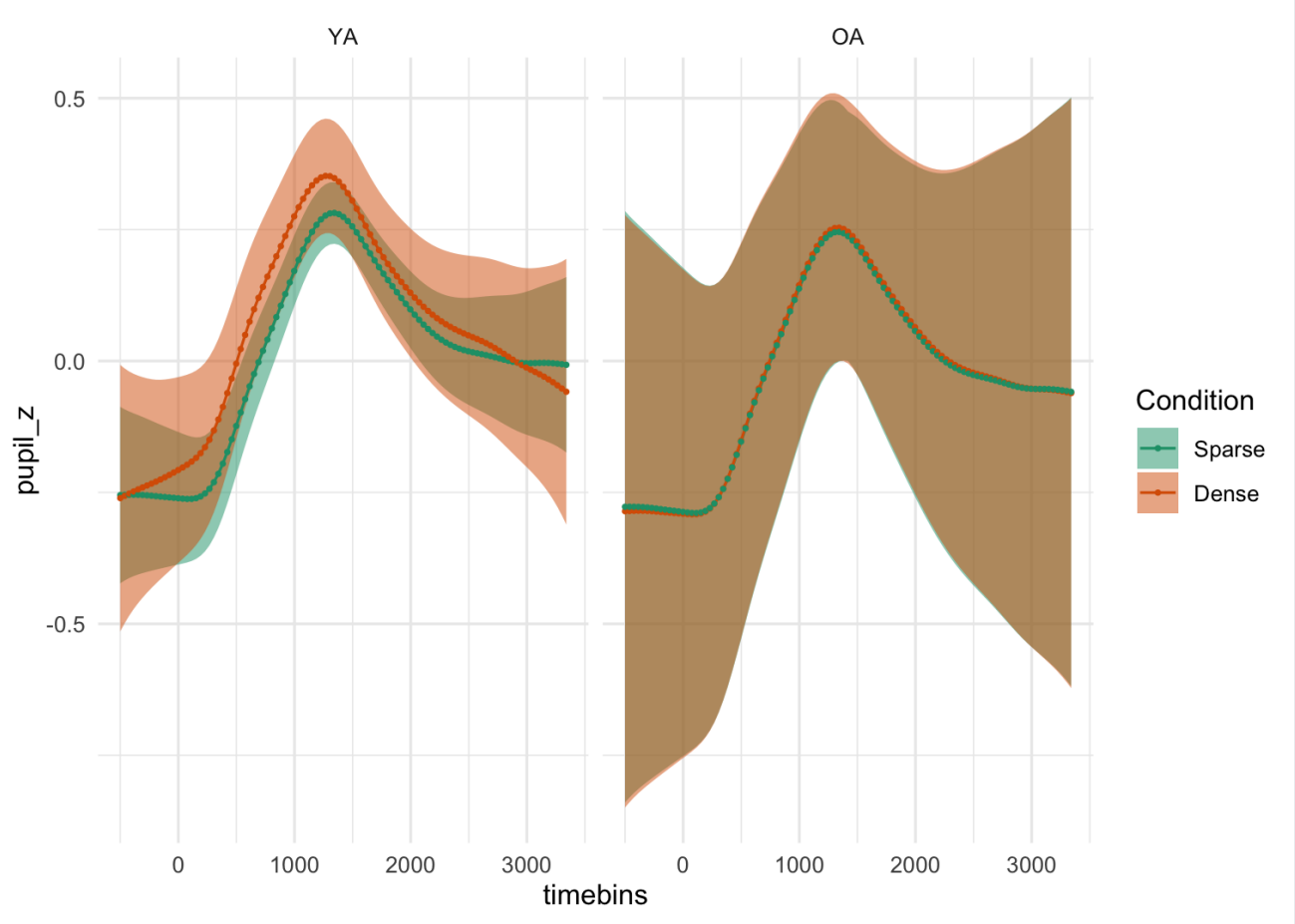


Comparing across groups (interactions)

```
pdq_gam_5_pred_2 <- predict_gam(
  pdq_gam_5, length_out = 100, exclude_terms = "s(timebins,subject)",
  separate = list(Age_Cond = c("Age", "Condition"))
) %>%
  # The separate arguments returns variables with default alphabetical order.
  # Let's reorder the levels in Condition and Age.
  mutate(
    Condition = factor(Condition, levels = c("Sparse", "Dense")),
    Age = factor(Age, levels = c("YA", "OA")),
  )
```

Comparing across groups (interactions)

```
pdq_gam_5_pred_2 %>% plot(series = "timebins", comparison = "Condition") + facet_grid(~ Age)
```

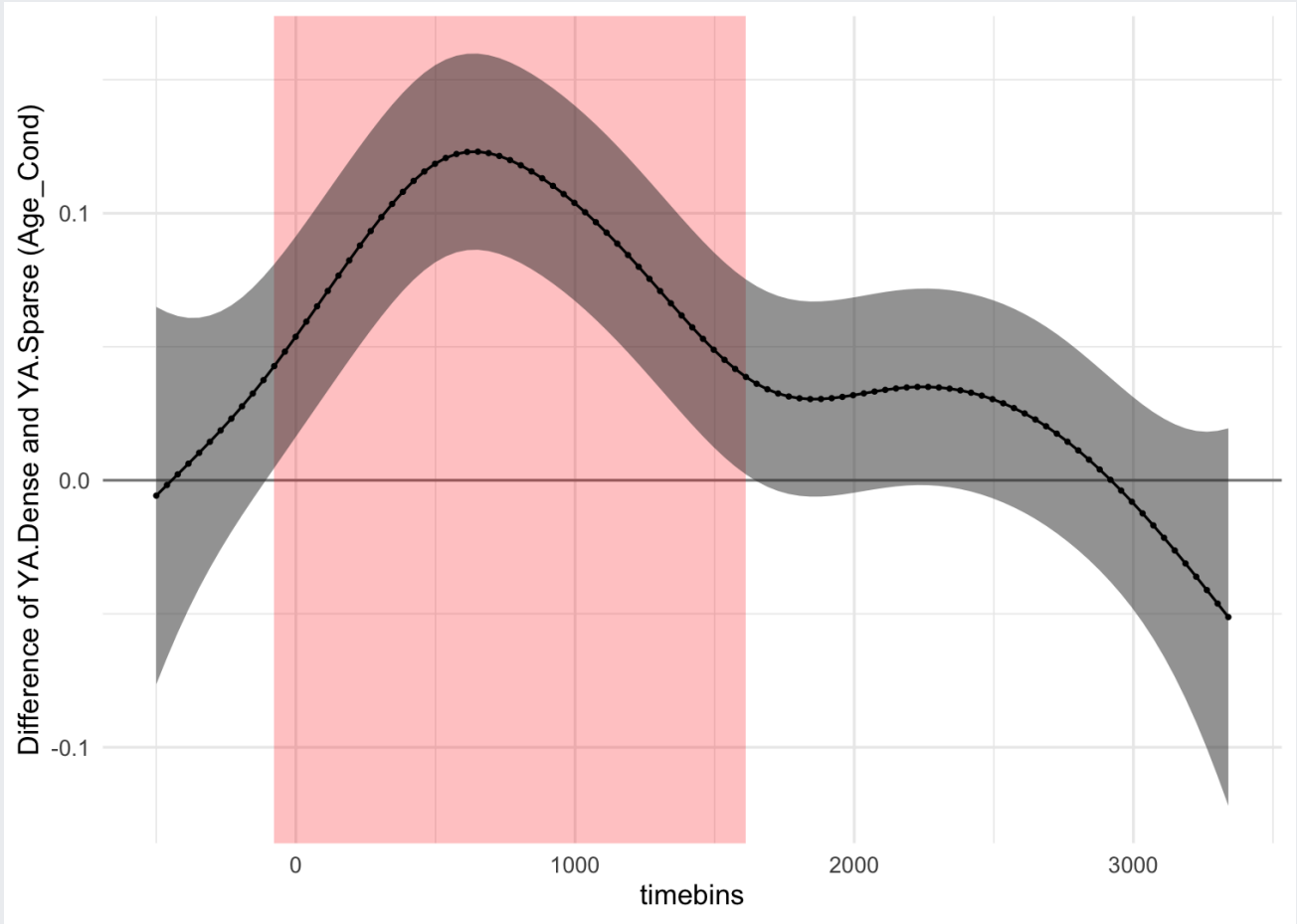


Comparing across groups (interactions)

```
pdq_gam_5_diff <- get_difference(
  pdq_gam_5, series = "timebins", length_out = 100, exclude_terms = "s(timebins,subject)",
  compare = list(Age_Cond = c("YA.Dense", "YA.Sparse"))
)
```

Comparing across groups (interactions)

```
pdq_gam_5_diff %>% plot()
```



PART II

Hands-on