INTRODUCTION

MGCV Skill training Session

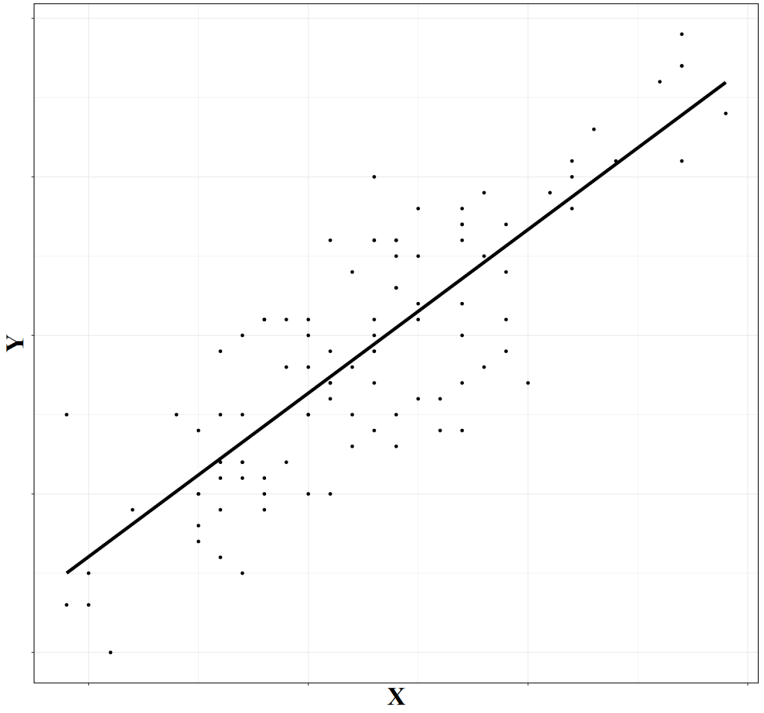
ASGER SVENNING
PHD STUDENT

A MINIMAL STATISTICS INTRO

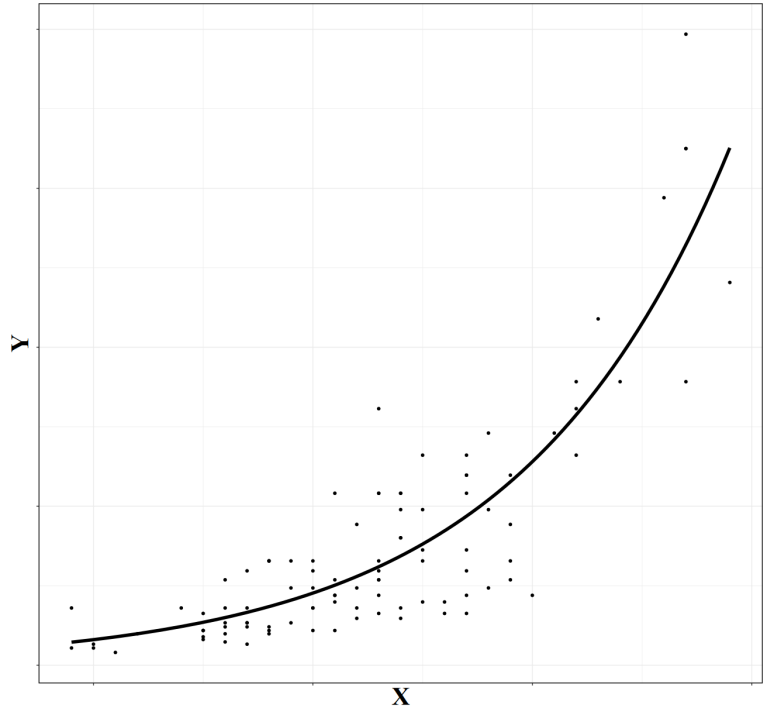
- What is a LM/GLM/GAM?
- Is it working??
- Ohh no…

LINEAR REGRESSION

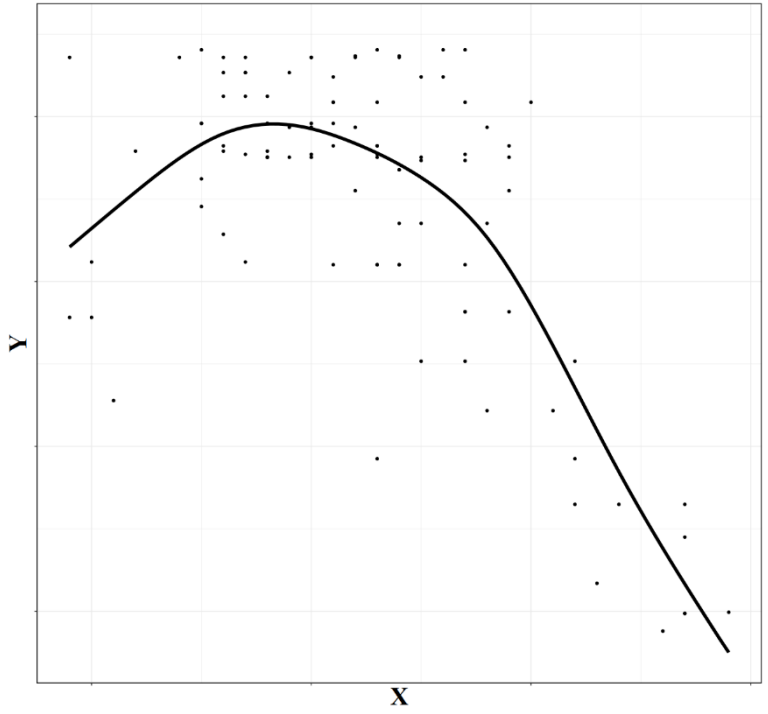
ordinary **L**inear **M**odel (LM)



Generalized **L**inear **M**odel (GLM)



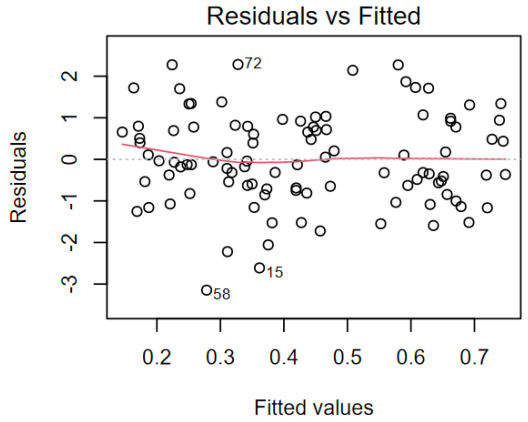
Generalized **A**dditive **M**odel (GAM)

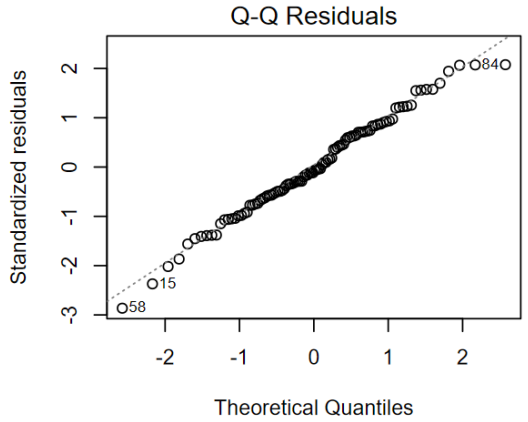


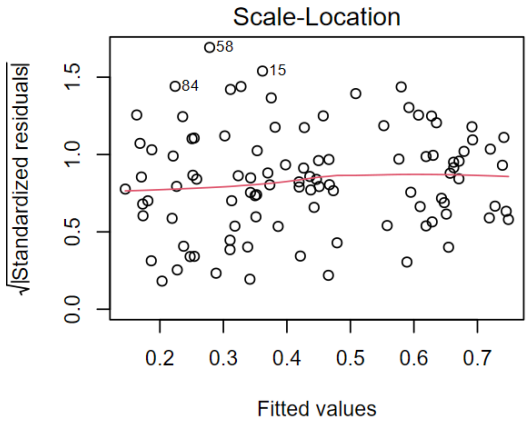
IS IT WORKING??

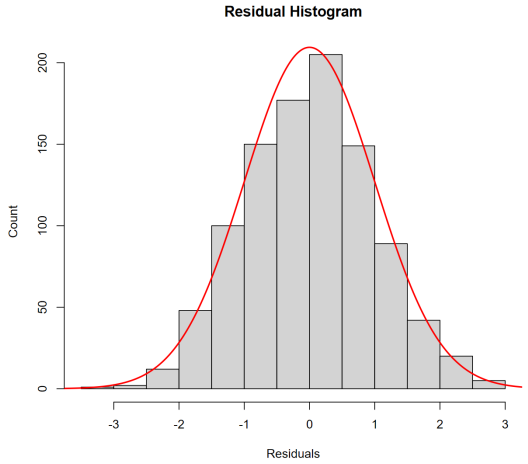
Many approaches, but…

```
lm(y ~ x, data = data) %>%
  plot
```

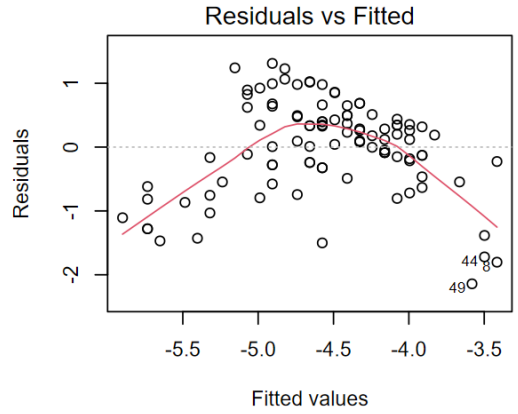


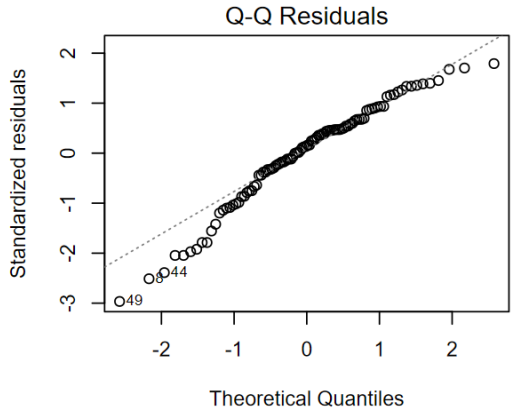


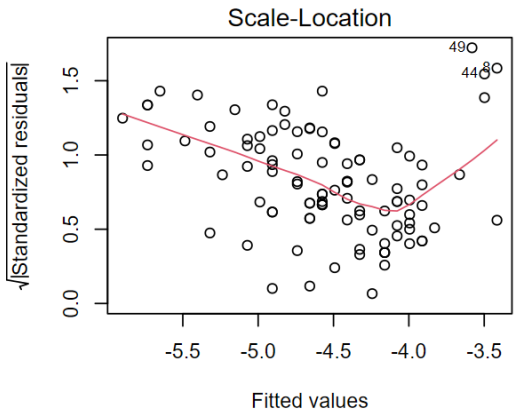


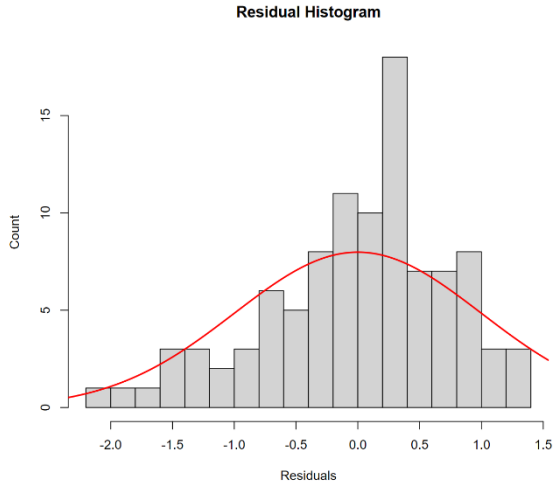


IS IT WORKING??

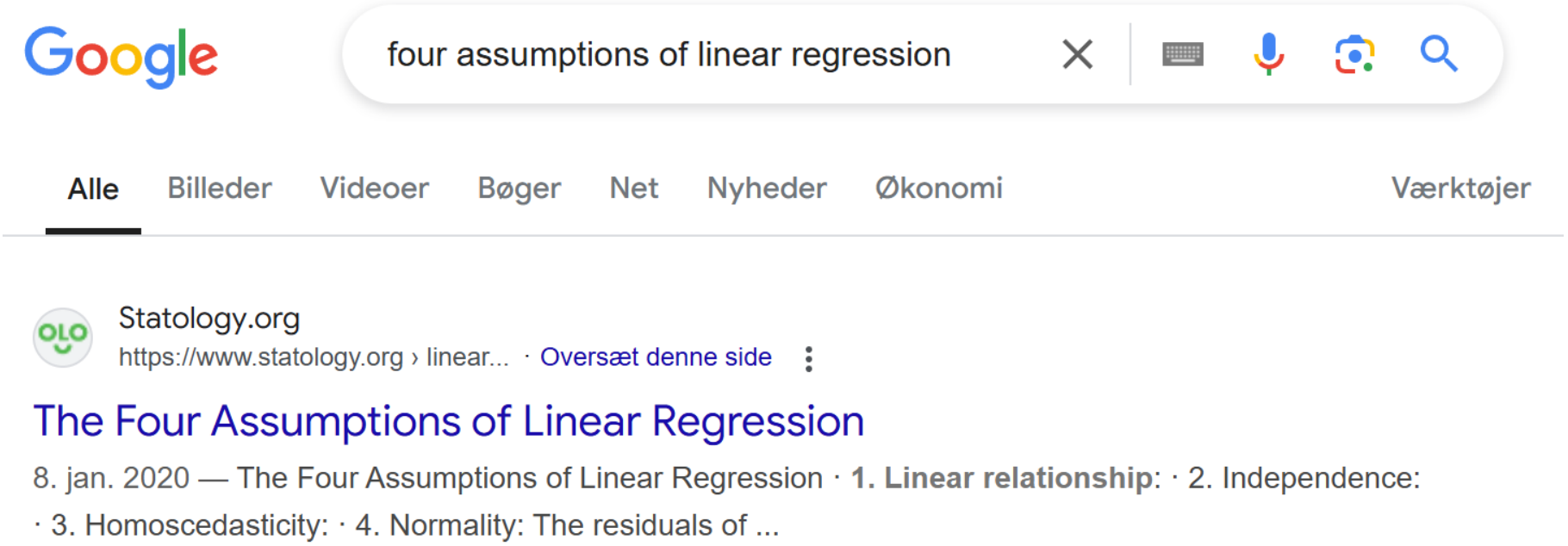








LET'S ASK GOOGLE



LET'S ASK GOOGLE

The Four Assumptions of "Linear" Regression

BY ZACH BOBBITT JANUARY 8, 2020

Linear regression is a useful statistical method we can use to understand the relationship between two variables, x and y. However, before we conduct linear regression, we must first make sure that four assumptions are met:

1. Linear relationship: There exists a linear relationship between the independent variable, x, and the dependent variable, y.

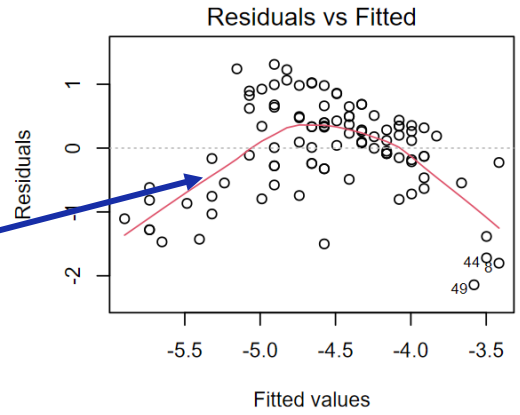
2. Independence: The residuals are independent. In particular, there is no correlation between consecutive residuals in time series data.

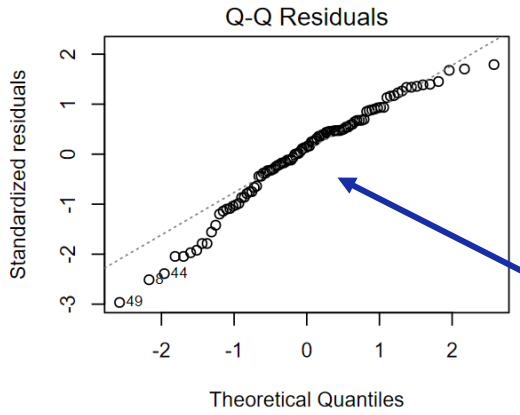
3. Homoscedasticity: The residuals have constant variance at every level of x.

4. Normality: The residuals of the model are normally distributed.

IS IT WORKING??

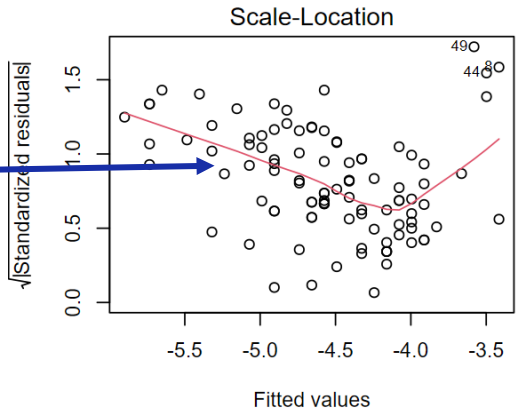
Not linear!

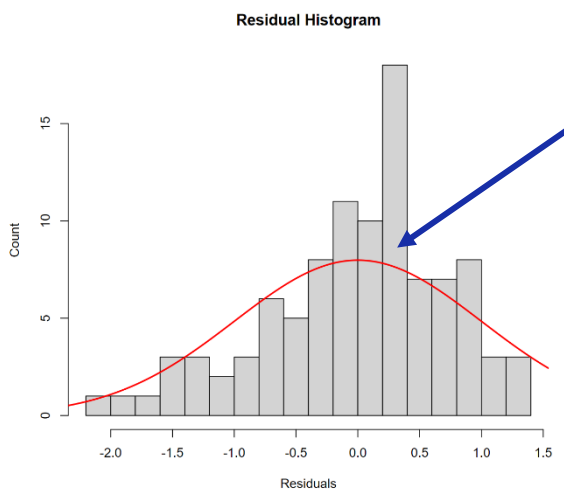




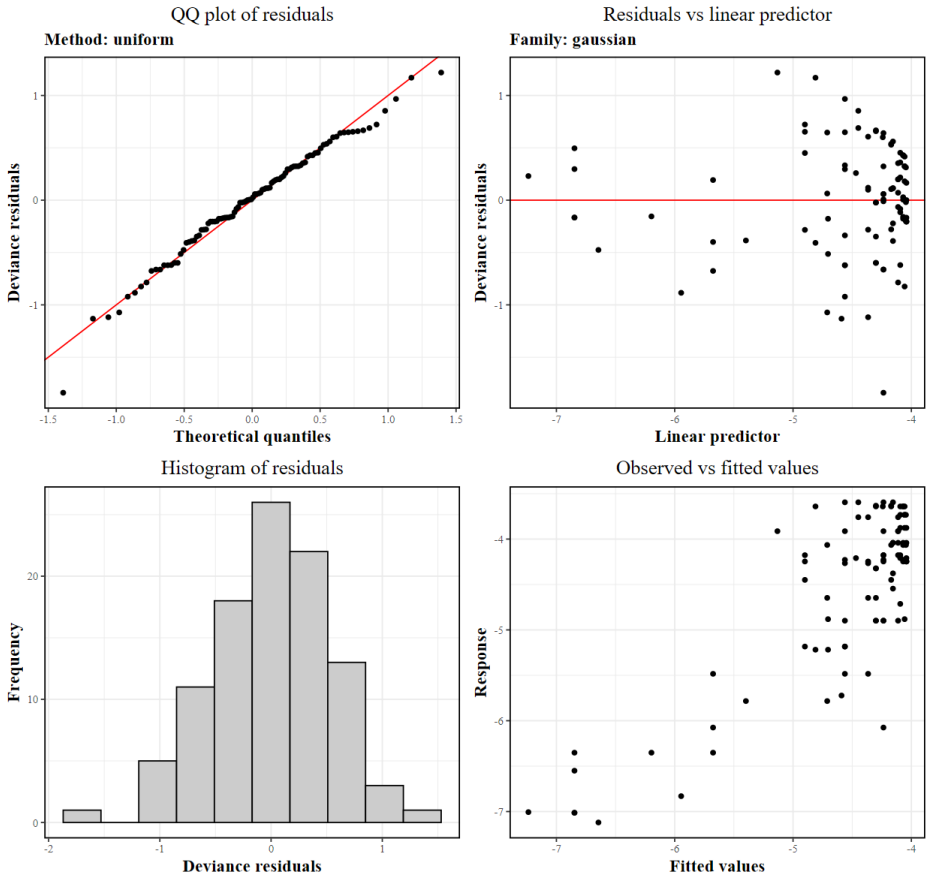
Residuals NOT normal

"Unequal variance" (heteroscedastic)



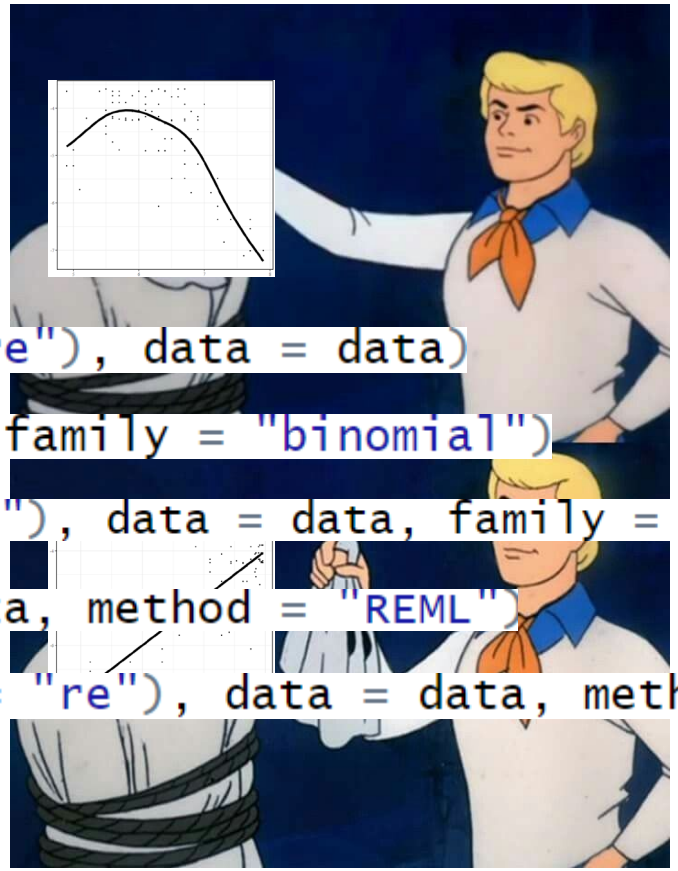


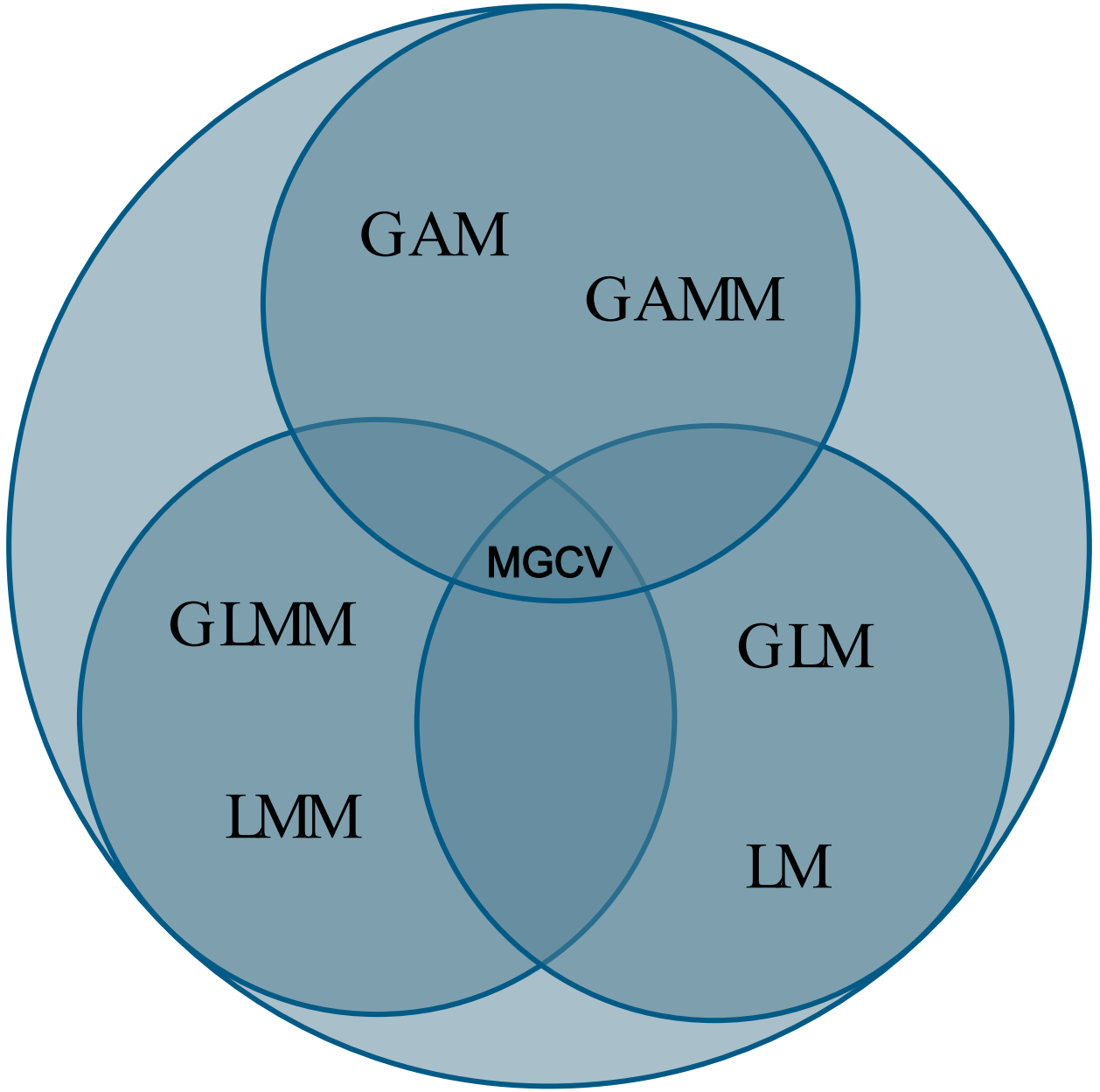
IS IT WORKING?? NOW WITH MGCV



WHAT IS A GAM AND MGCV?!

- LM `gam(y ~ x, data = data)`
- LMM `gam(y ~ x + s(z, bs = "re"), data = data)`
- GLM `gam(y ~ x, data = data, family = "binomial")`
- GLMM `gam(y ~ x + s(z, bs = "re"), data = data, family = "binomial")`
- GAM `gam(y ~ s(x), data = data, method = "REML")`
- GAMM `gam(y ~ s(x) + s(z, bs = "re"), data = data, method = "REML", family = "binomial")`

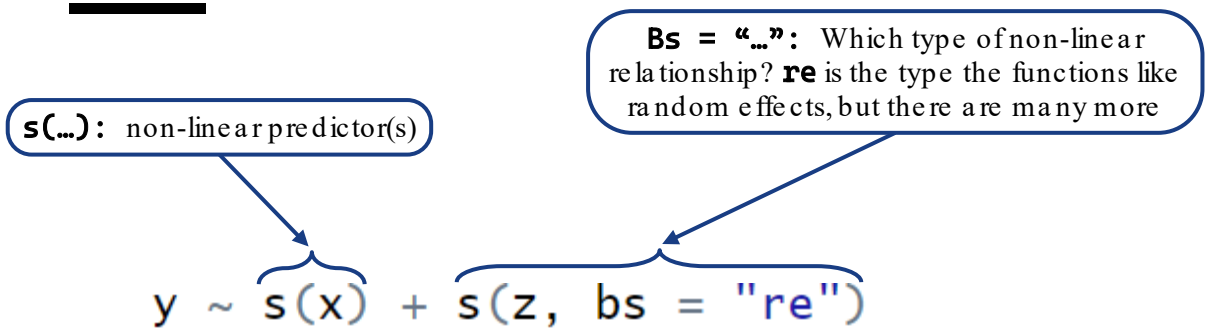



GAM
GAMM
MGCV
GLMM
GLM
LMM
LM

DISSECTING A MGCV GAM

```
gam(y ~ s(x) + s(z, bs = "re"), data = data, method = "REML", family = "binomial")
```

DISSECTING A MGCV GAM




```
gam(Response ~ Predictors + Random effects, data = data, method = "REML", family = "binomial")
```

DISSECTING A MGCV GAM

```
y ~ s(x) + s(z, bs = "re")
```

```
gam(Response ~ Predictors + Random effects, Modelling data, method = "REML", family = "binomial")
```

```
, data = data
```

DISSECTING A MGCV GAM

`method = "REML"`: Technical detail, don't worry about it.
If using any `s(...)` terms, use "REML". Otherwise simply omit `method`.

```
gam(Response ~ Predictors + Random effects, Modelling data, {Estimation method}, family = "binomial")
```

```
y ~ s(x) + s(z, bs = "re")
```

```
, method = "REML"
```

```
, data = data
```

DISSECTING AN MGCV GAM

```
y ~ s(x) + s(z, bs = "re")                    , method = "REML"

gam(Response ~ Predictors + Random effects | Modelling data | Estimation method | Response family (~type))

                    , data = data                      , family = "binomial"
```

family = "…": Relationship between response and predictor, as well as residual distribution. Like standard **glm**, but much more powerful.

DISSECTING A MGCV GAM

s(…): non-linear predictor(s)

Bs = "…": Which type of non-linear relationship? **re** is the type the functions like random effects, but there are many more

method = "REML": Technical detail, don't worry about it. If using any **s(…)** terms, use "REML" otherwise simply omit `method`.

```
y ~ s(x) + s(z, bs = "re")
, method = "REML"
```

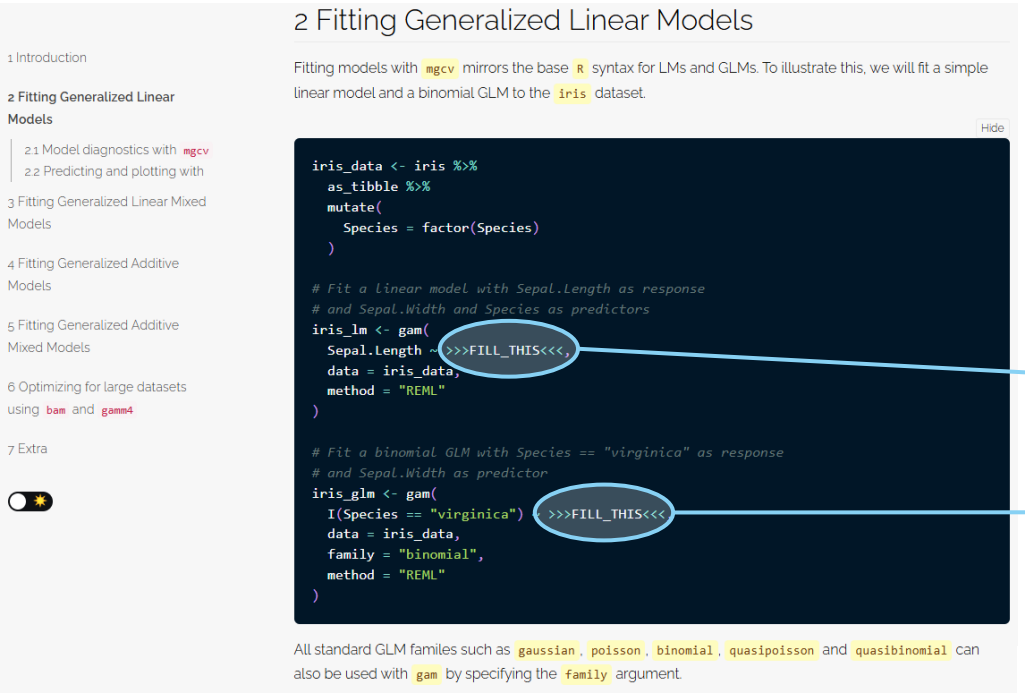
```
gam(Response ~ Predictors + Random effects, Modelling data, {Estimation method}, Response family (~type))
```

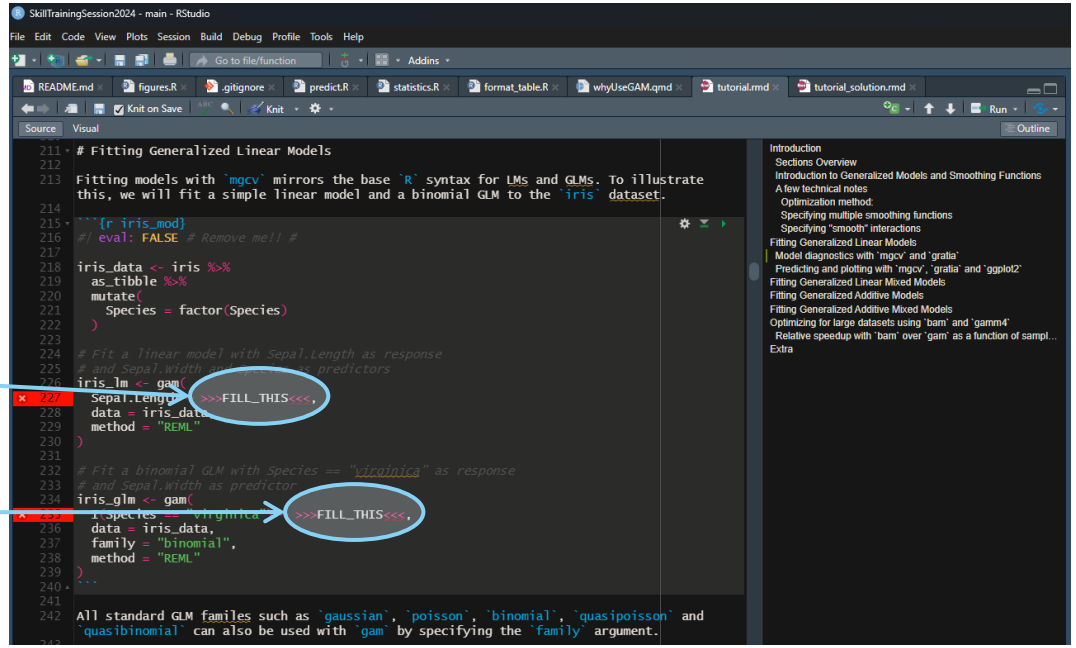
```
, data = data
, family = "binomial"
```

family = "…": Relationship between response and predictor, as well as residual distribution. Like standard **glm**, but much more powerful.

HOW DO I COMPLETE THE TUTORIAL?

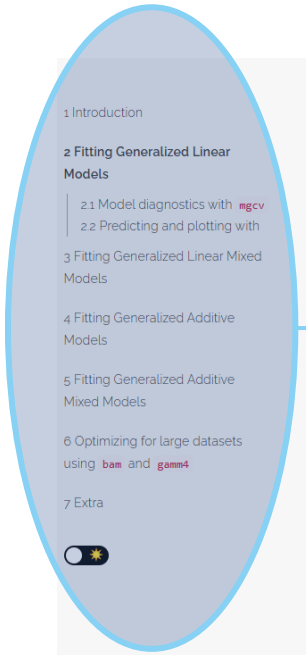
Code

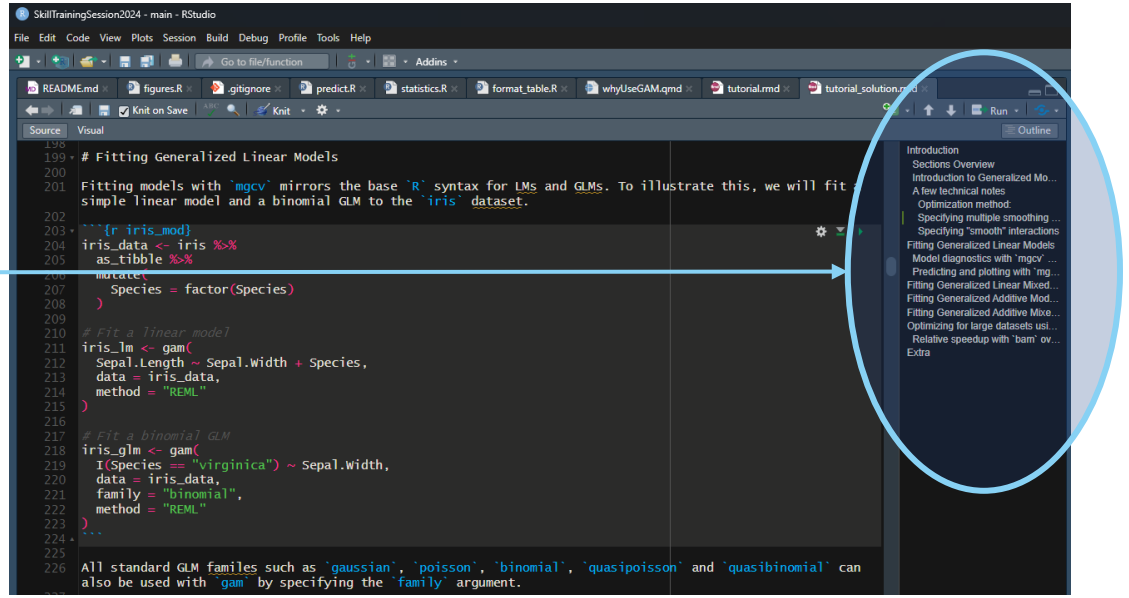




HOW DO I COMPLETE THE TUTORIAL?

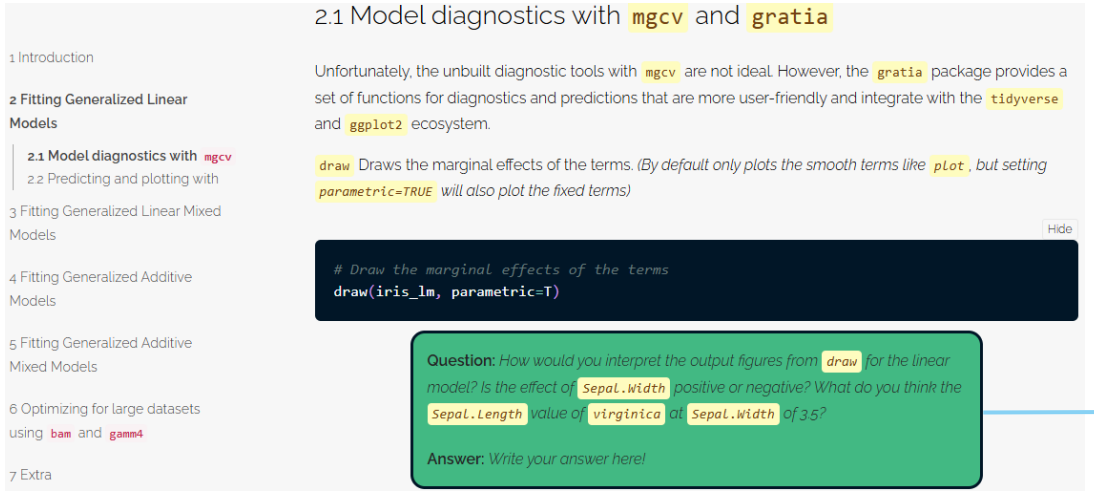
Navigation

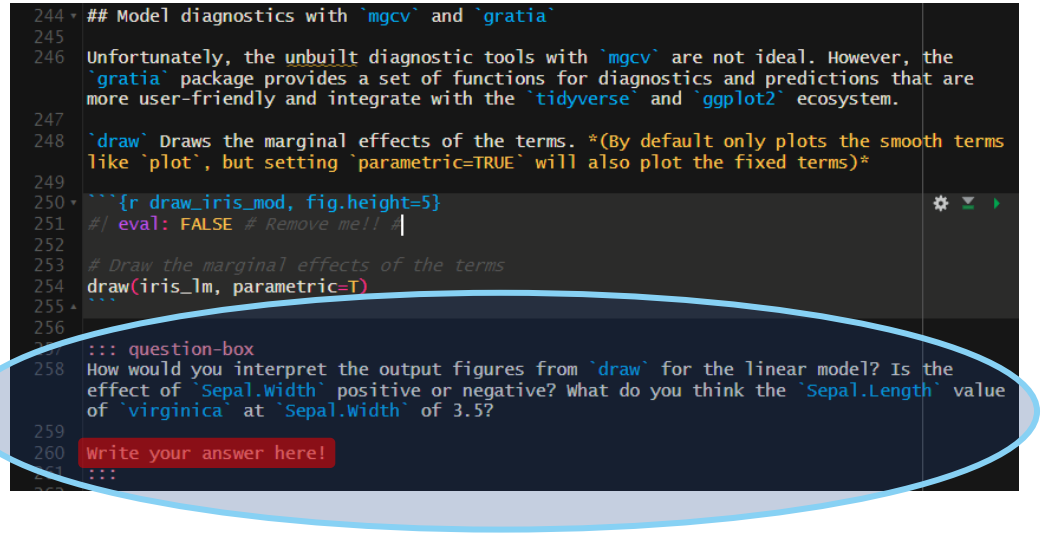




HOW DO I COMPLETE THE TUTORIAL?

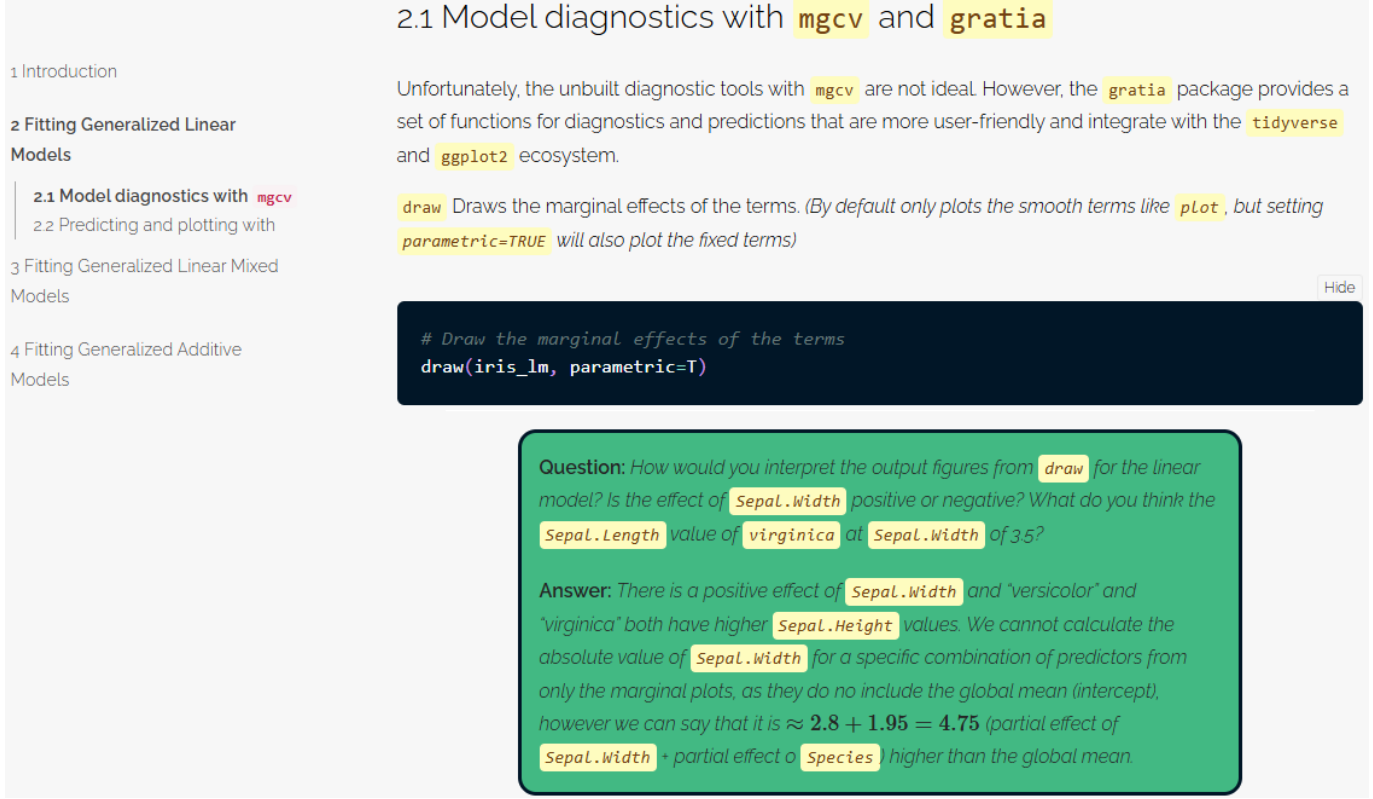
Questions





HOW DO I COMPLETE THE TUTORIAL?

Questions



1 Introduction

2 Fitting Generalized Linear Models

2.1 Model diagnostics with `mgcv`

2.2 Predicting and plotting with

3 Fitting Generalized Linear Mixed Models

4 Fitting Generalized Additive Models

2.1 Model diagnostics with `mgcv` and `gratia`

Unfortunately, the unbuilt diagnostic tools with `mgcv` are not ideal. However, the `gratia` package provides a set of functions for diagnostics and predictions that are more user-friendly and integrate with the `tidyverse` and `ggplot2` ecosystem.

`draw` Draws the marginal effects of the terms. *(By default only plots the smooth terms like `plot`, but setting `parametric=TRUE` will also plot the fixed terms)*

Hide

```
# Draw the marginal effects of the terms
draw(iris_lm, parametric=T)
```

Question: *How would you interpret the output figures from `draw` for the linear model? Is the effect of `Sepal.Width` positive or negative? What do you think the `Sepal.Length` value of `virginica` at `Sepal.Width` of 3.5?*

Answer: *There is a positive effect of `Sepal.Width` and "versicolor" and "virginica" both have higher `Sepal.Height` values. We cannot calculate the absolute value of `Sepal.Width` for a specific combination of predictors from only the marginal plots, as they do no include the global mean (intercept), however we can say that it is* $\approx 2.8 + 1.95 = 4.75$ *(partial effect of `Sepal.Width` + partial effect o `Species`) higher than the global mean.*

EXERCISE

- 30 minutes
- Pair up
- Discuss questions, and solve the problems together
- (Later → Explanations of advanced use-cases)

AARHUS
UNIVERSITY

ADVANCED USAGE

MGCV Skill training Session

ASGER SVENNING
PHD STUDENT

FAMILIES IN MGCV

scat (scaled t): For heavy tailed data

ziP (zero-inflated poisson): For count data with excessive zeros

nb (negative binomial): For count data, when poisson is not appropriate

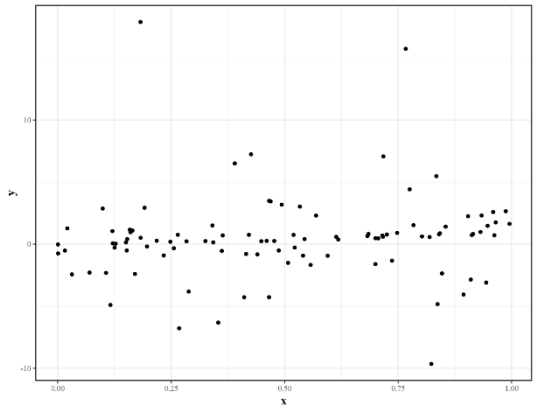
betar (beta regression): For proportion data between, not including, 0 and 1

gaulss (gaussian location-scale): For linear regression, where the variance is non-constant

ziplss (zero-inflated poisson location-scale): For count data, where zeros are modelled separately

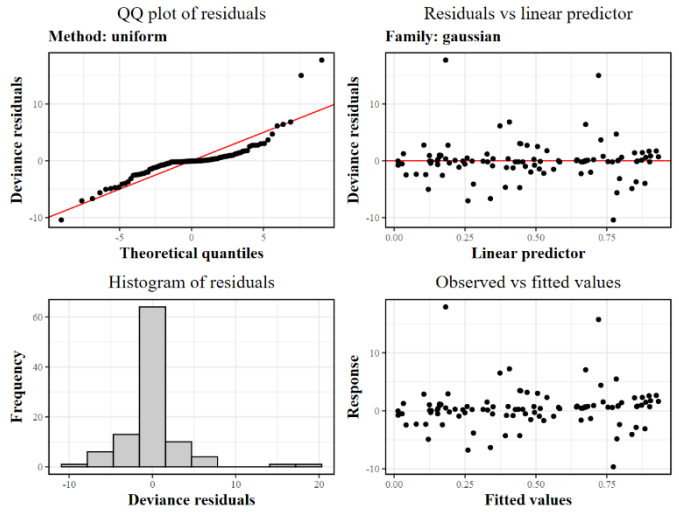
SCALED T

```r
data <- tibble(
  x = runif(100),
  y = rnorm(100, x) %>%
    add(sign(.) * . ^2)
)
```



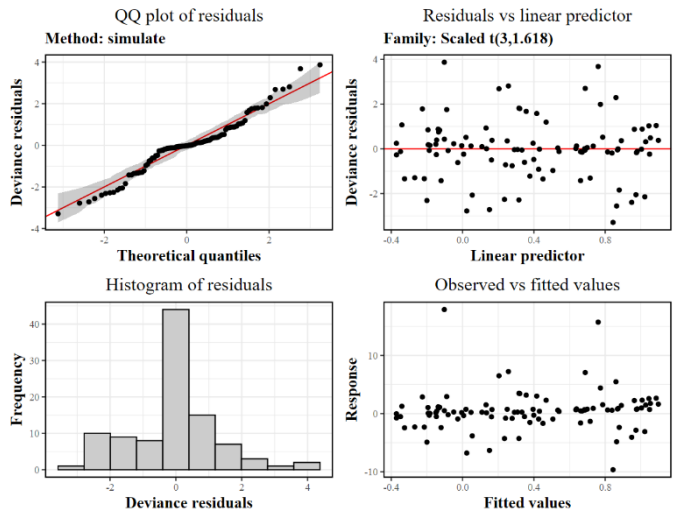

```r
gam(y ~ x, data = data) %>%
  appraise()
```

ordinary LM



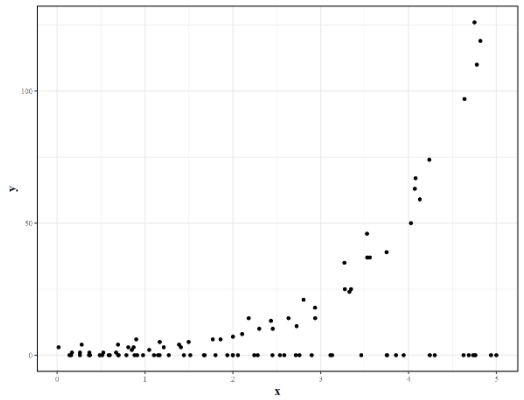

```r
gam(y ~ x, data = data, family = "scat") %>%
  appraise()
```

family = “scat”



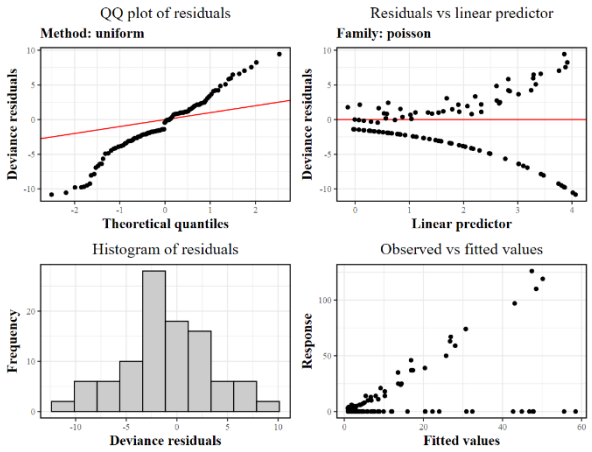
ZERO-INFLATED POISSON

```
data <- tibble(
  x = runif(100) * 5,
  y = rpois(x, exp(x))
    * (runif(100) > 0.5)
)
```



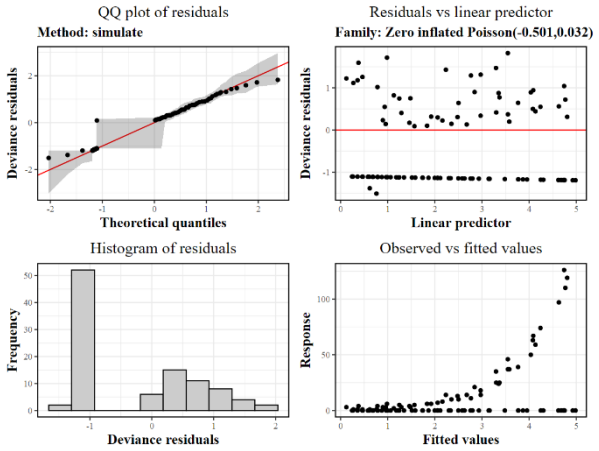

```
gam(y ~ x, data = data, family = "poisson") %>%
  appraise()
```

family = “poisson”



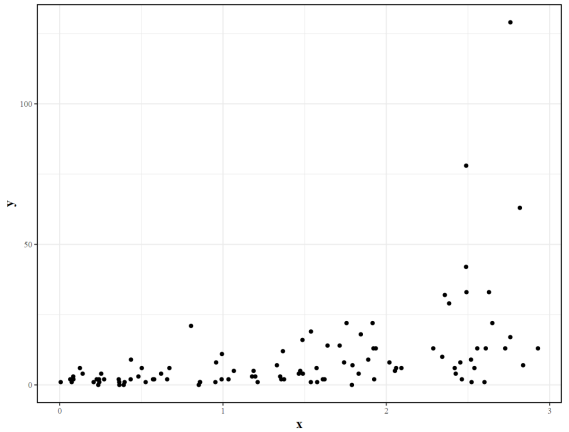

```
gam(y ~ x, data = data, family = "ziP") %>%
  appraise()
```

family = “ziP”



NEGATIVE BINOMIAL

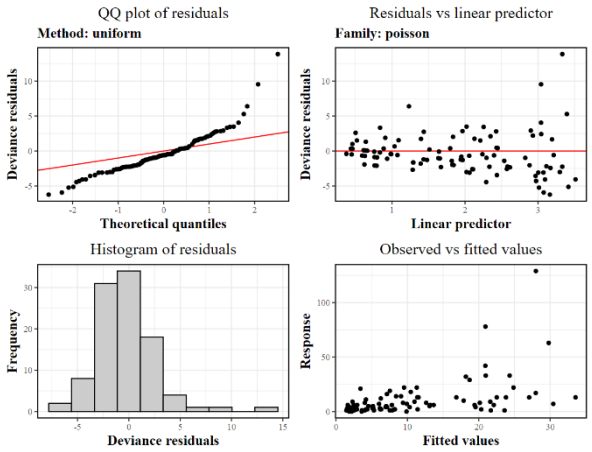
```r
data <- tibble(
  x = runif(100) * 3,
  y = rnorm(100, x) %>%
    exp %>%
    round
)
```



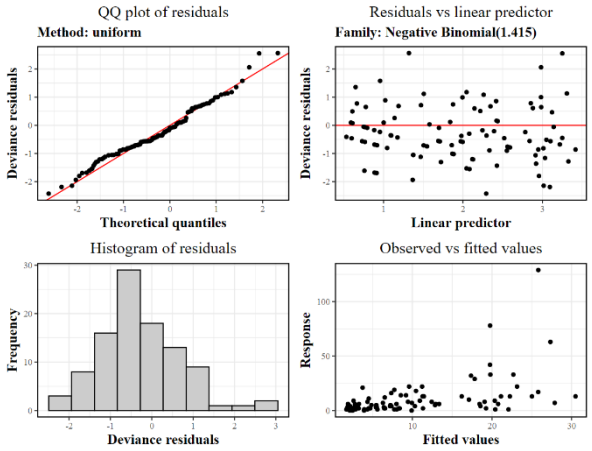

```r
gam(y ~ x, data = data, family = "poisson") %>%
  appraise()
```

```r
gam(y ~ x, data = data, family = "nb") %>%
  appraise()
```

family = “poisson”

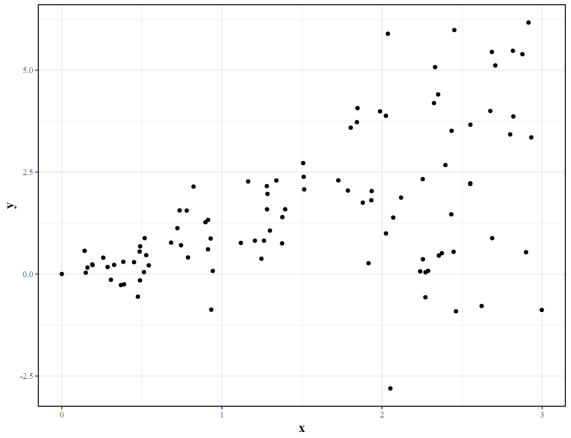


family = “nb”



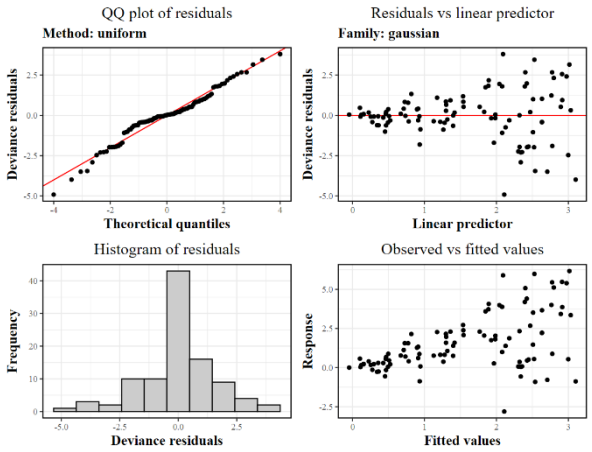
GAUSSIAN LOCATION-SCALE

```
data <- tibble(
  x = runif(100) * 3,
  y = rnorm(100, x, x)
)
```



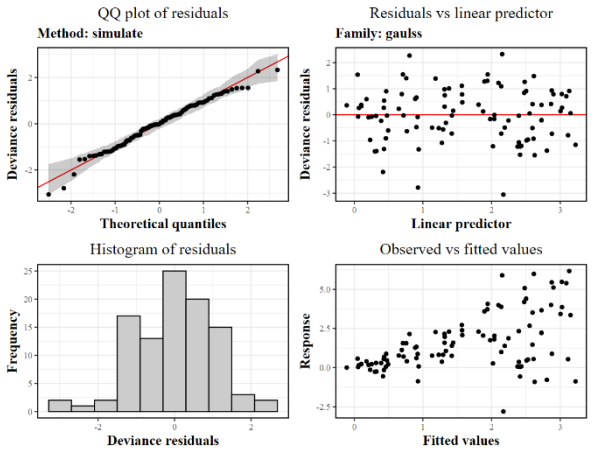

```
gam(y ~ x, data = data) %>%
  appraise()
```

Ordinary LM



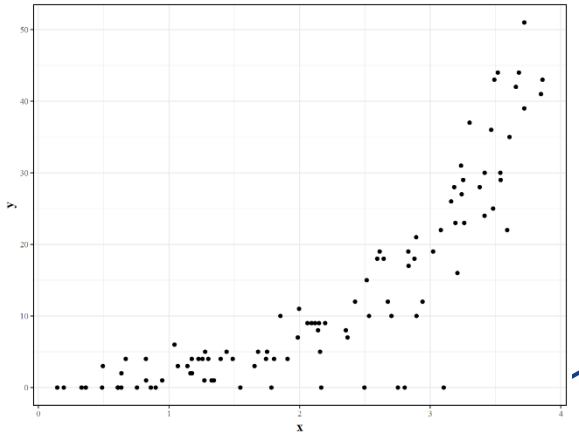

```
gam(list(y ~ x, ~ x), data = data, family = "gaulss") %>%
  appraise()
```

family = “gaulss”



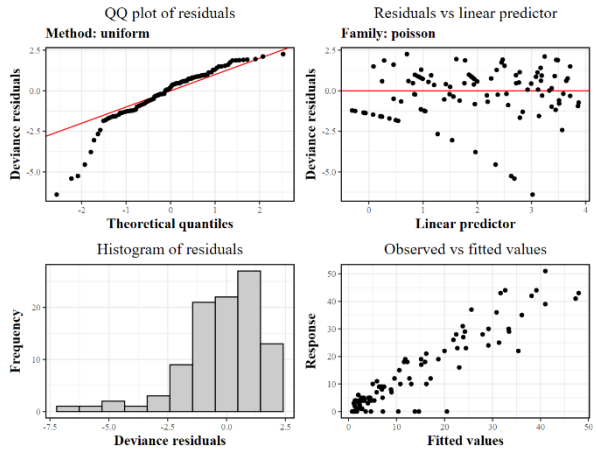
ZERO-INFLATED POISSON LOCATION-SCALE

```r
data <- tibble(
  x = runif(100) * 4,
  y = rpois(x, exp(x))
    * (rnorm(100, x, 2) > 0)
)
```



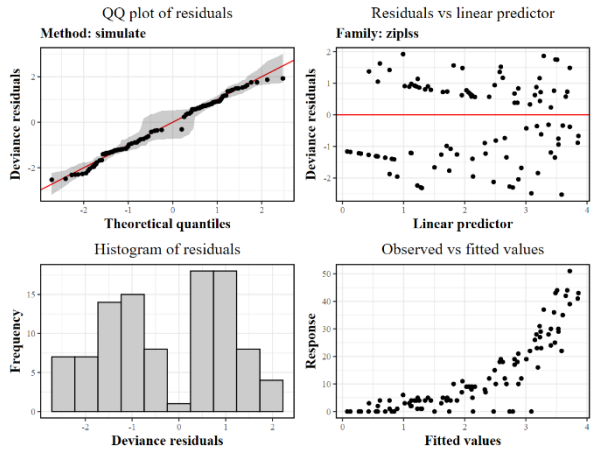

```r
gam(y ~ x, data = data, family = "poisson") %>%
  appraise()
```

family = “poisson”




```r
gam(list(y ~ x, ~ x), data = data, family = "ziplss") %>%
  appraise()
```

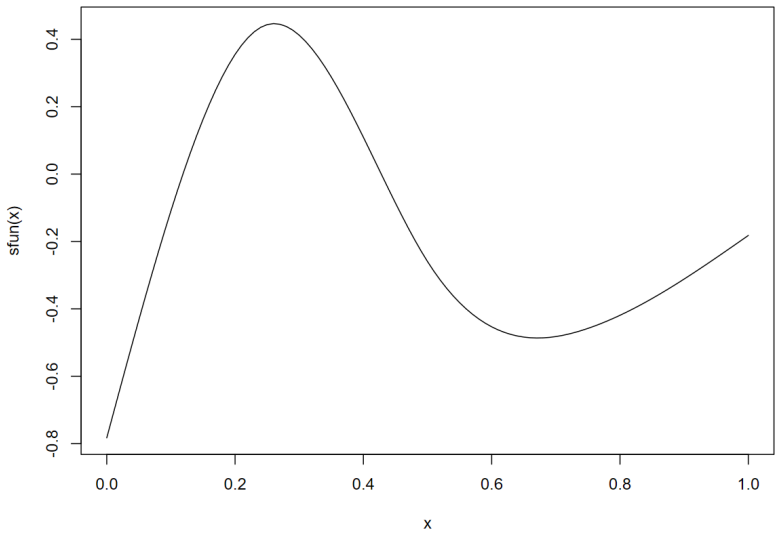
family = “ziplss”



SMOOTH TERMS

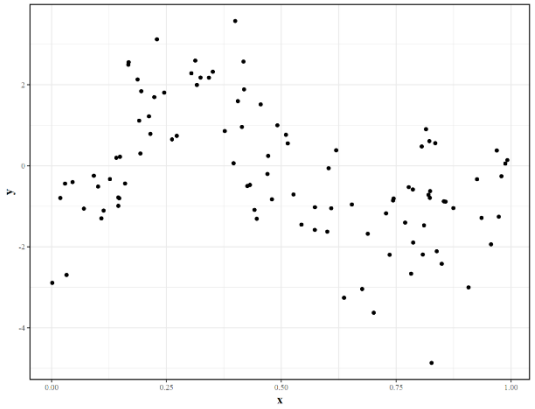
- tp (thin-plate splines): default smooth
- cr (cubic spline): old-school smooth
- ts/cs: shrinkage versions of tp/cr; smooth can have no effect
- cc (cyclic cubic): matches the "ends", useful for e.g. time-of-day/year
- re (random effect): similar to a normal factor term, but uses fewer degrees of freedom
- fs (random factor smooth): for the case where there are multiple similar, but not equal, non-linear relationships for different factor levels (groups)

```r
get_sfun <- function(
    zlim = c(-0.1, 1.1),
    slim = c(-1, 1),
    ctrl.pts=10,
    plot=F
) {
  z <- seq(zlim[1], zlim[2], length.out = ctrl.pts)
  c <- runif(ctrl.pts, min = slim[1], max = slim[2]) - z

  sfun <- splinefun(z, c, method = "natural")
  if (plot) plot_smooth(sfun, z, c, zlim)

  return(invisible(sfun))
}

sfun <- get_sfun(c(0, 1), ctrl.pts=5)

curve(sfun(x), 0, 1)
```



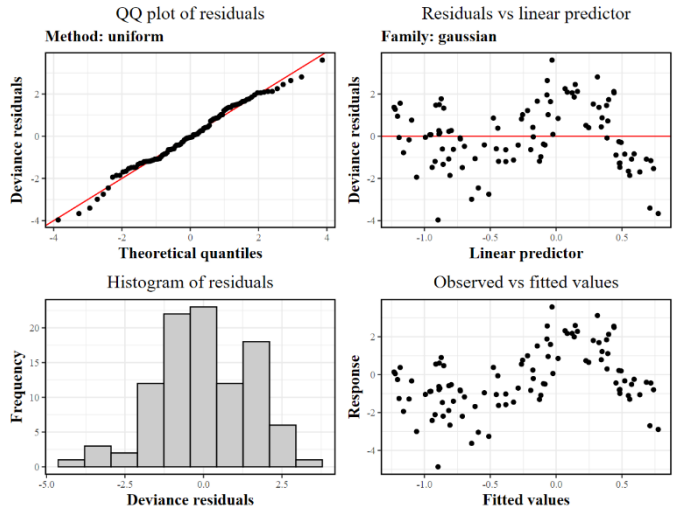
THIN-PLATE SPLINES

```
data <- tibble(
  x = runif(100),
  y = 3 * sfun(x) + rnorm(100, 0.25)
)
```



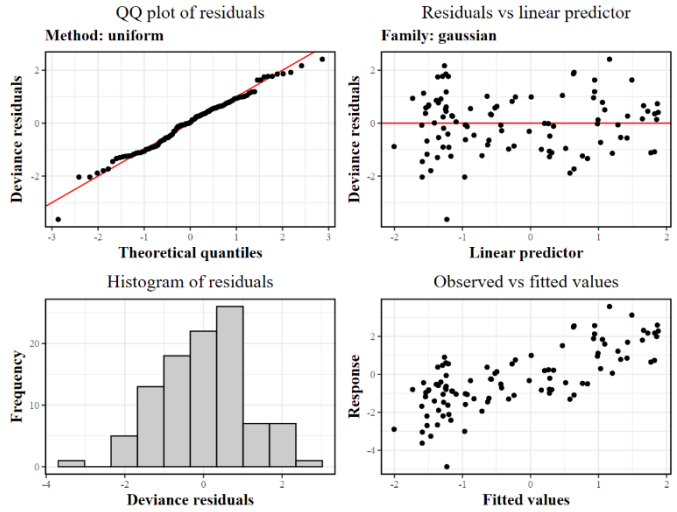
```
gam(y ~ x, data = data) %>%
  appraise()
```

ordinary LM



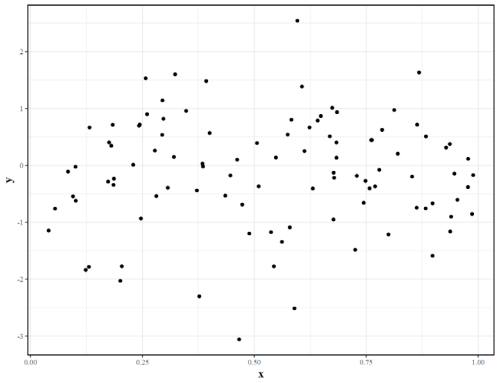

```
gam(y ~ s(x, bs = "tp"), data = data, method = "REML") %>%
  appraise()
```

family = “scat”



SHRINKAGE SPLINES

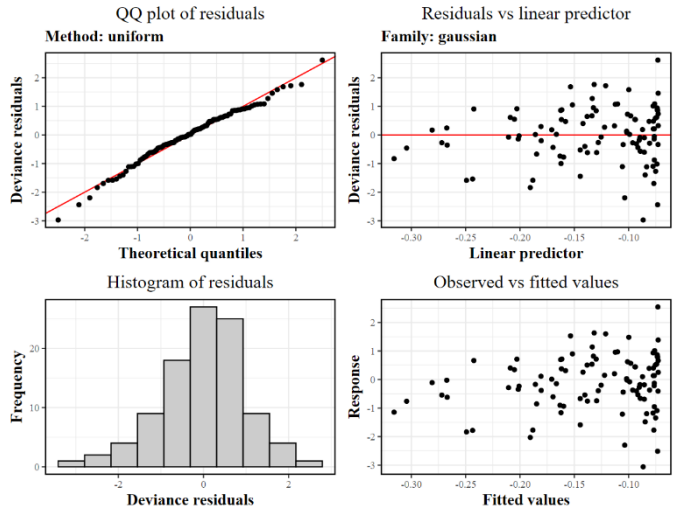
```
data <- tibble(
  x = runif(100),
  y = rnorm(100)
)
```



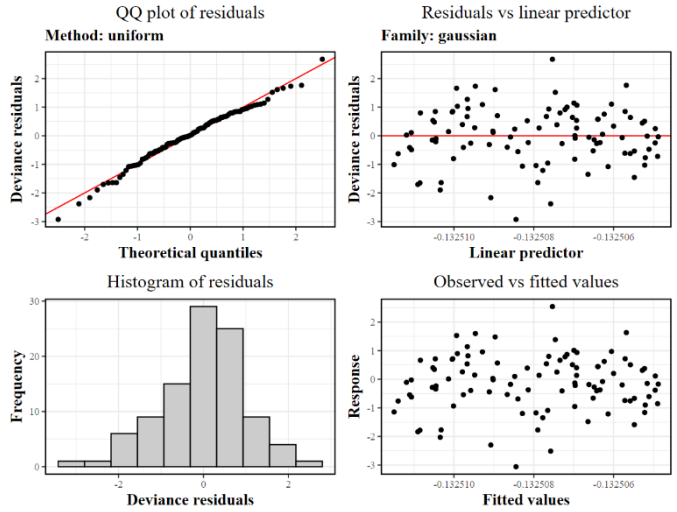

```
gam(y ~ s(x, bs = "tp"), data = data, method = "REML") %>%
  appraise
```

```
gam(y ~ s(x, bs = "ts"), data = data, method = "REML") %>%
  appraise
```

bs = “tp”

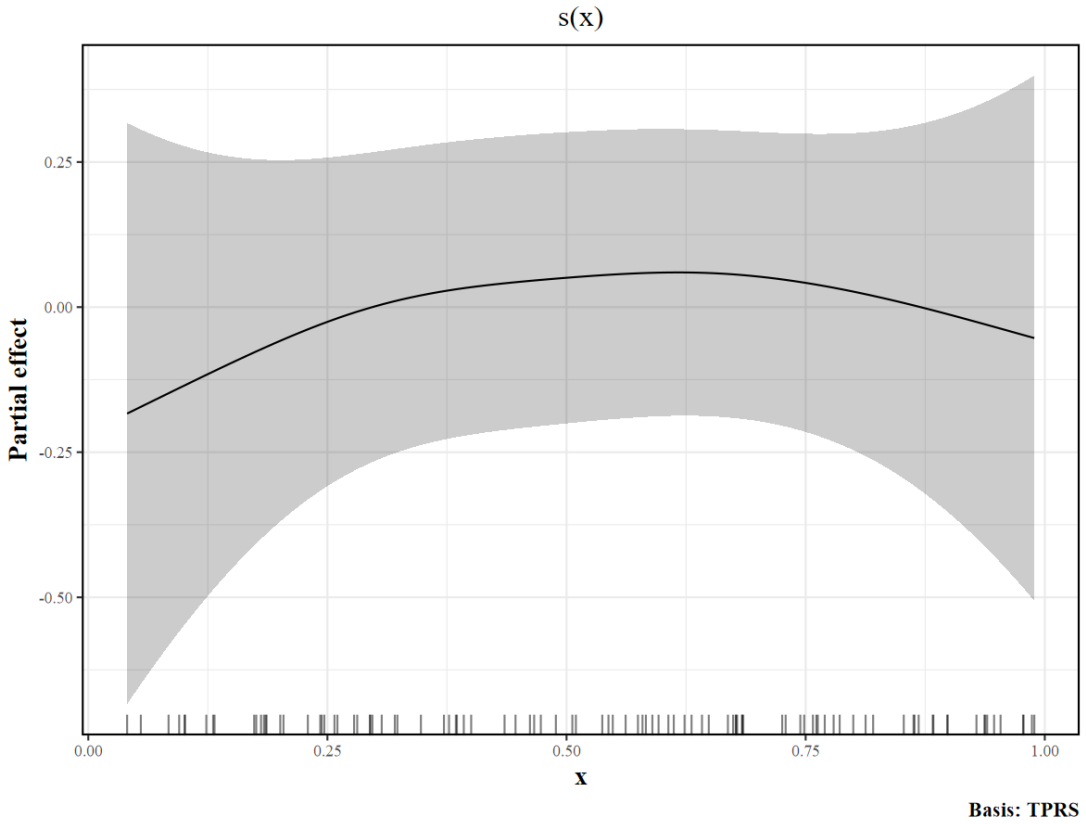


bs = “ts”



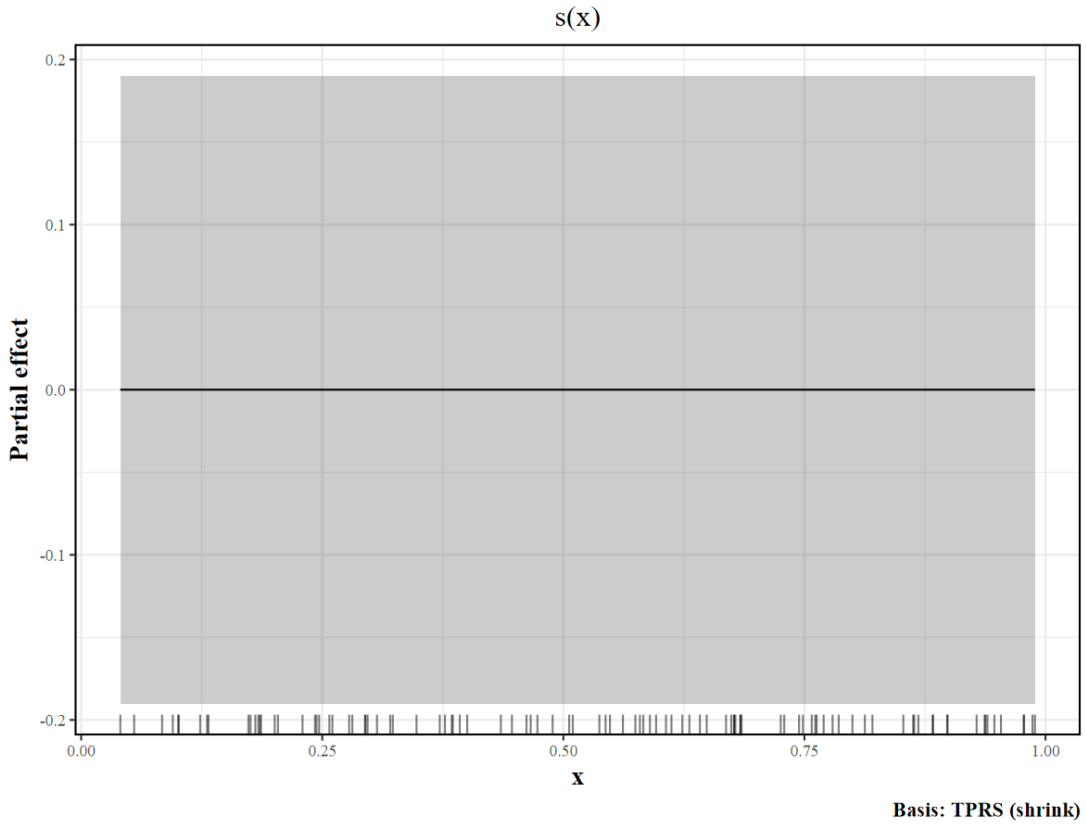
STANDARD

`bs = "tp"`

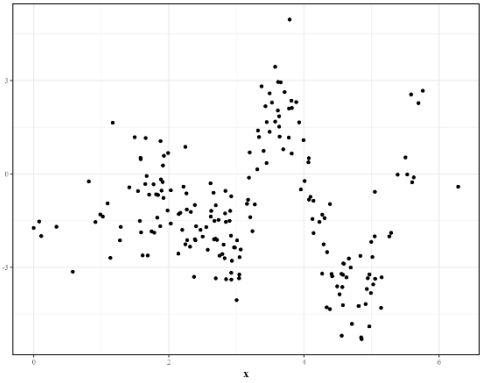


SHRINKAGE

`bs = "ts"`



CYCLIC SPLINES

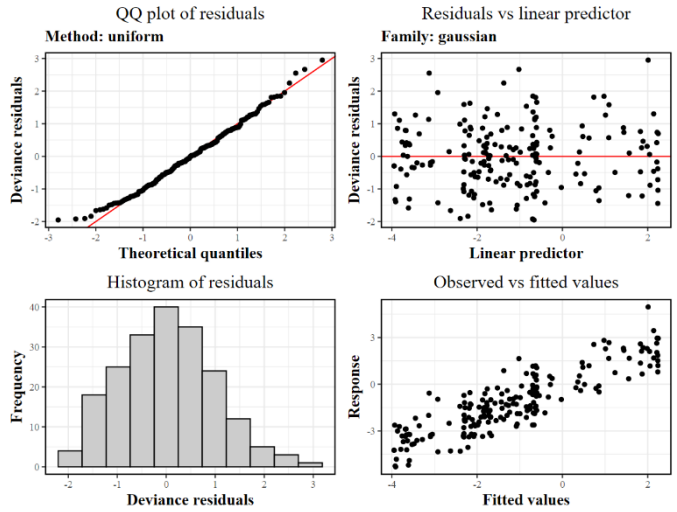


```r
data <- tibble(
  x = c(0, (runif(198) + runif(198)) * pi, 2 * pi),
  y = sfun((sin(x) + 1) / 2) * 5 + rnorm(200, 0.15)
)
```

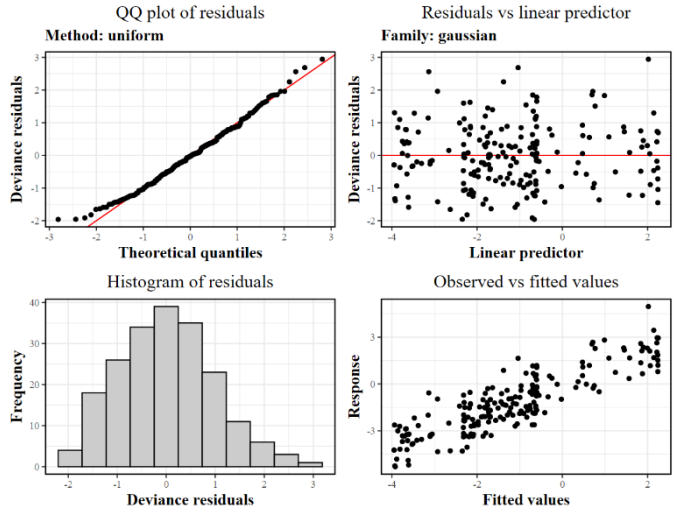
```r
gam(y ~ s(x, bs = "cr"), data = data, method = "REML") %>%
  appraise
```

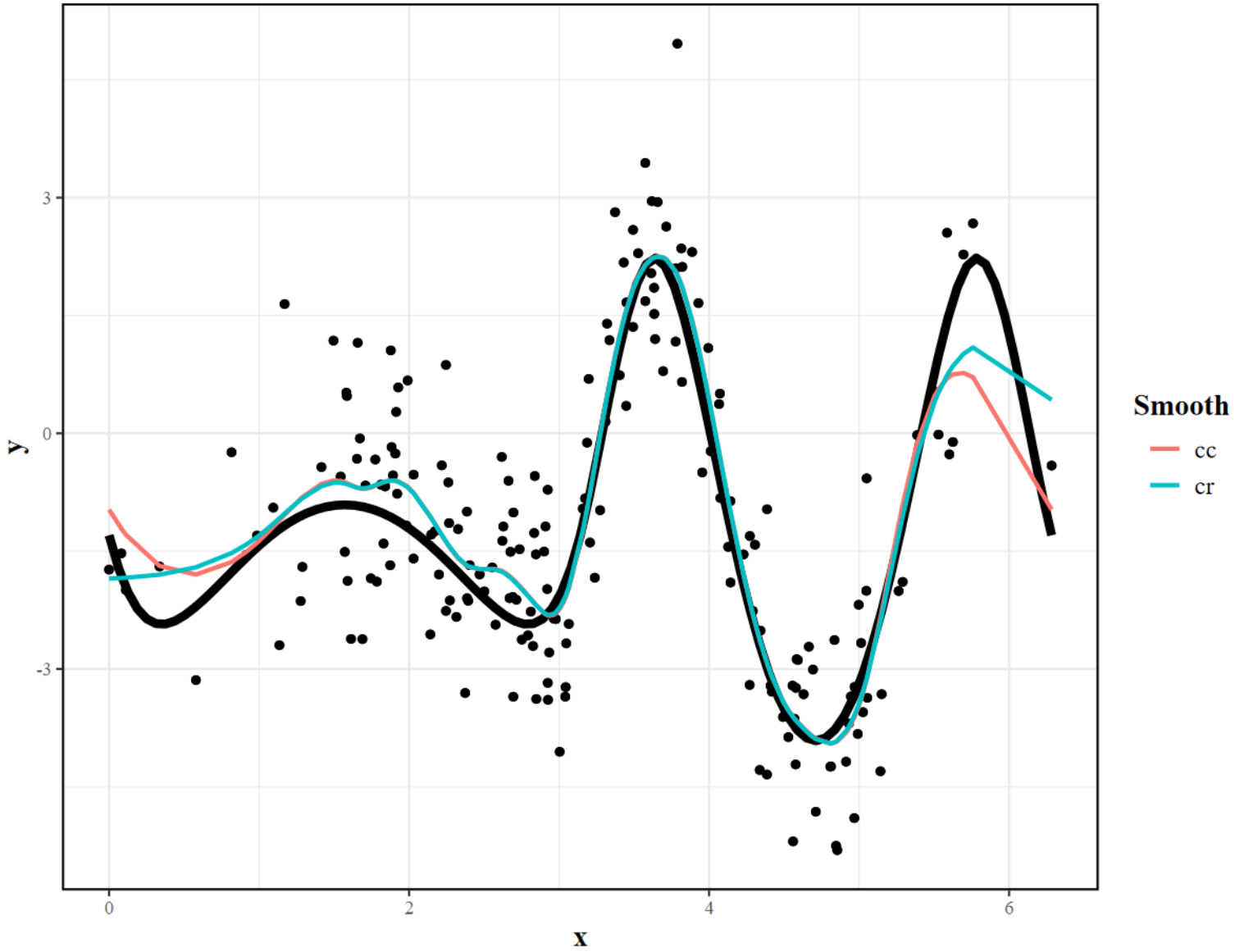
```r
gam(y ~ s(x, bs = "cc"), data = data, method = "REML") %>%
  appraise
```

bs = “cr”

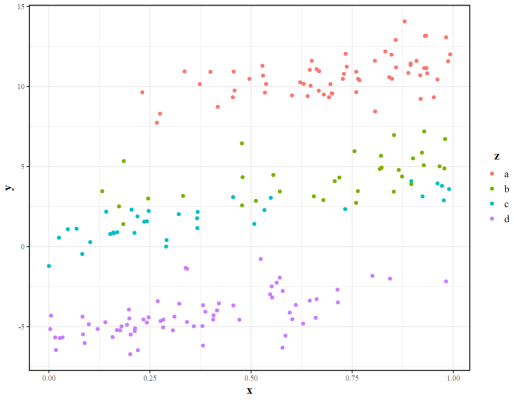


bs = “cc”




3
0
-3
y
0
2
4
6
x
Smooth
cc
cr

RANDOM EFFECTS

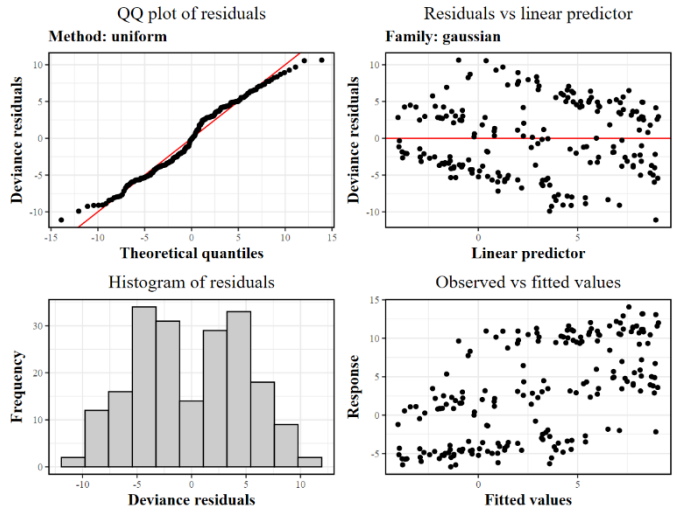



```
data <- tibble(
  x = runif(200),
  w = abs(((matrix(rep(1:4, 200), 200, byrow = T) - 1/2) / 4) - x),
  z = apply(w, 1, function(x) sample(letters[1:4], 1, prob = x)) %>%
    factor
) %>%
  group_by(z) %>%
  mutate(
    r = rnorm(1) * 5
  ) %>%
  ungroup %>%
  mutate(
    y = 3 * x + rnorm(100, 0.1) + r
  )
```

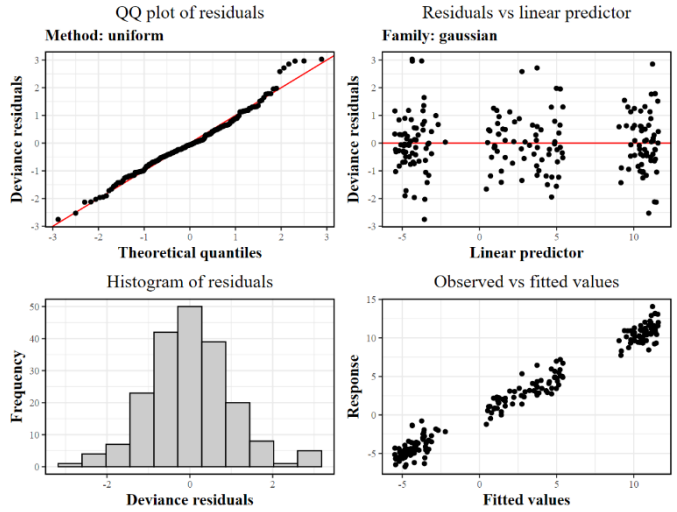
```
gam(y ~ x, data = data, method = "REML") %>%
  appraise()
```

```
gam(y ~ x + s(z, bs = "re"), data = data, method = "REML") %>%
  appraise()
```

bs = “cr”

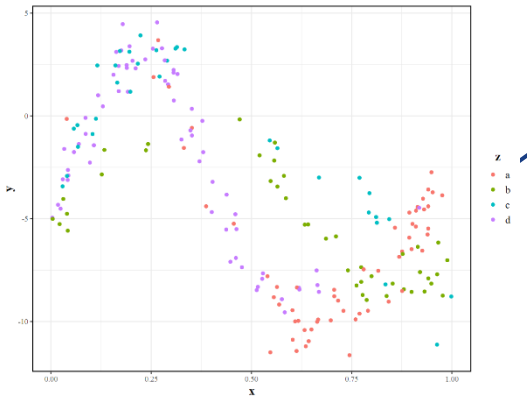


bs = “cc”



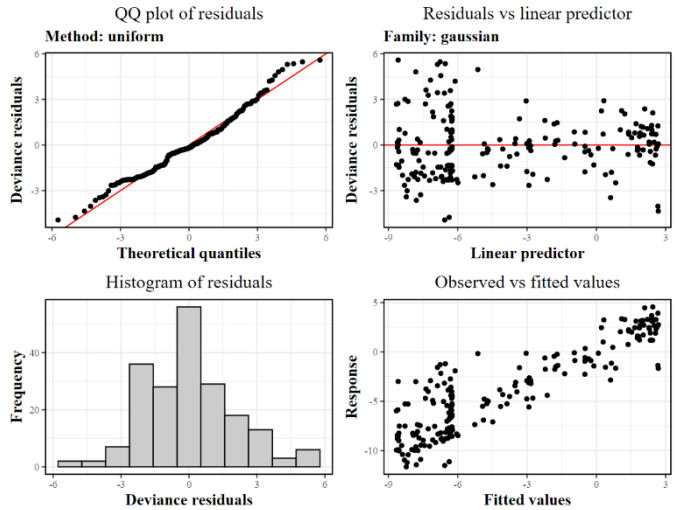
RANDOM FACTOR SMOOTH

```r
data <- tibble(
  x = runif(200),
  w = abs(((matrix(rep(1:4, 200), 200, byrow = T) - 1/2) / 4) - x),
  z = apply(w, 1, function(x) sample(letters[1:4], 1, prob = x)) %>%
    factor
) %>%
  group_by(z) %>%
  mutate(
    r = get_sfun(c(0, 1), ctrl.pts=5)(x) * 5
  ) %>%
  ungroup %>%
  mutate(
    y = 3 * sfun(x) + rnorm(100, 0.1) + r
  )
```



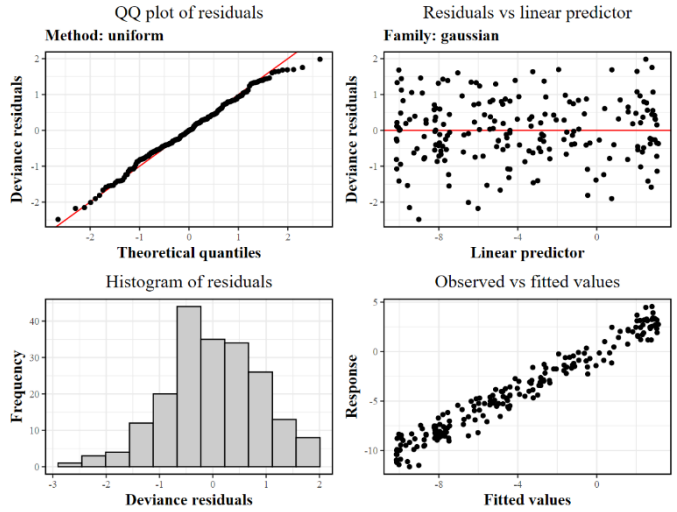

```r
gam(y ~ s(x), data = data, method = "REML") %>%
  appraise()
```

bs = “tp”



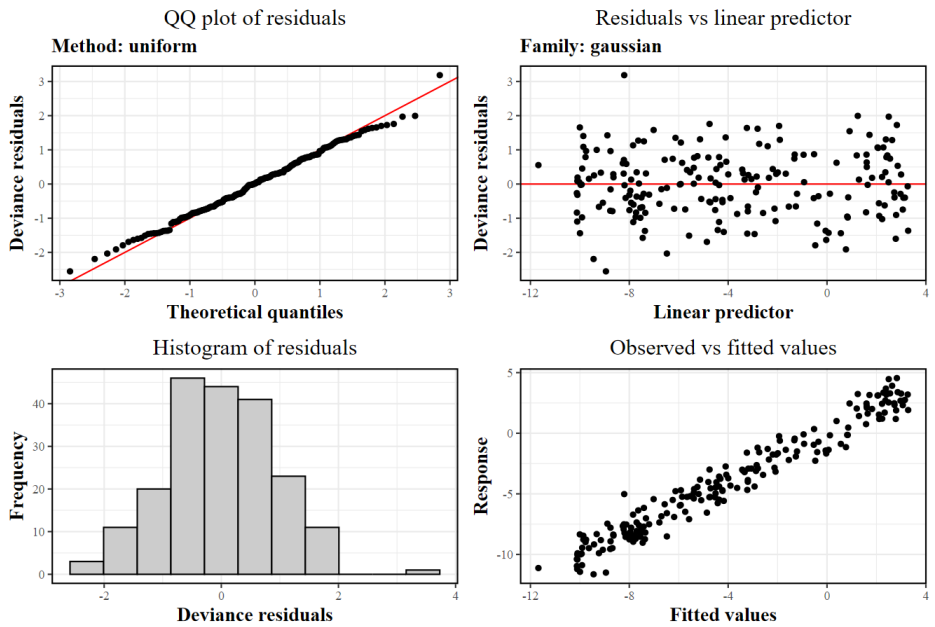

```r
gam(y ~ s(x, z, bs = "fs"), data = data, method = "REML") %>%
  appraise()
```

Interaction + bs = “fs”



MULTIPLE INDEPENDENT SMOOTHS

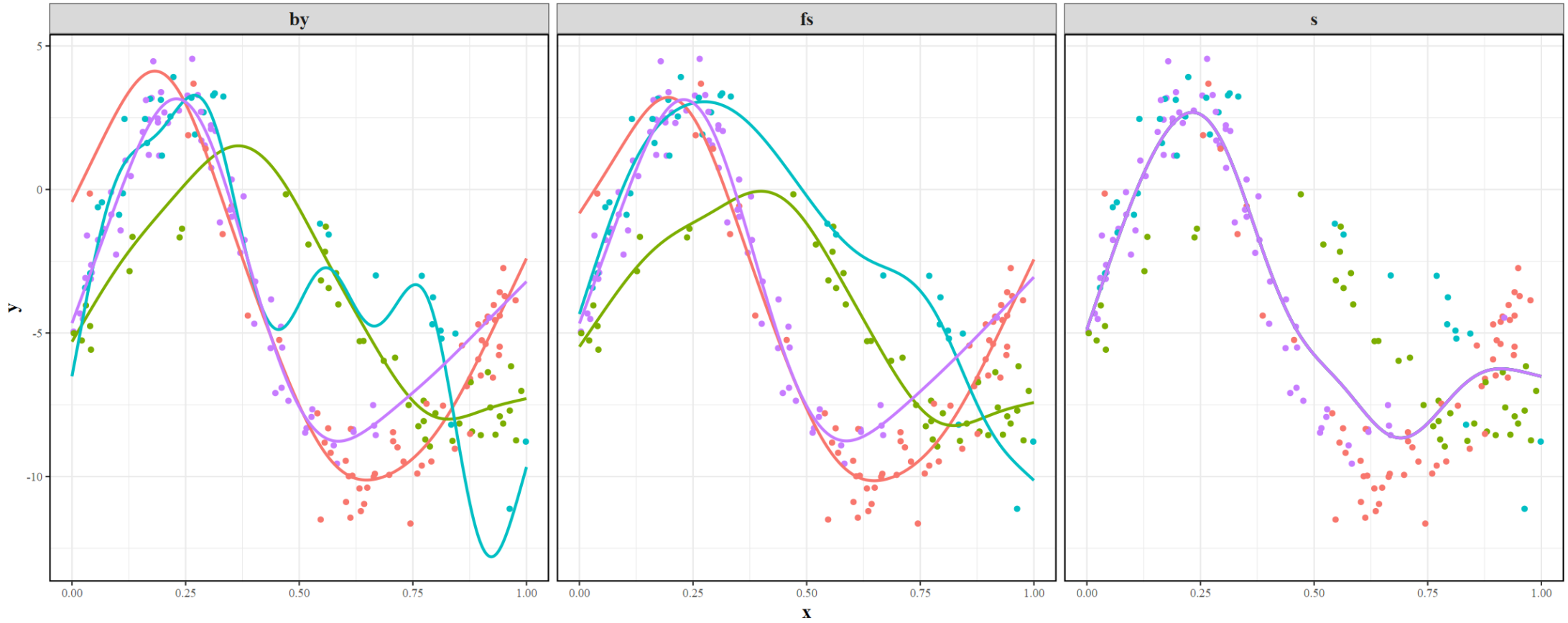
```
gam(y ~ s(x, by = z), data = data, method = "REML") %>%
  appraise()
```



INDEPENDENT SMOOTHS

RANDOM FACTOR SMOOTHS

DEFAULT SMOOTH



INTERACTIONS

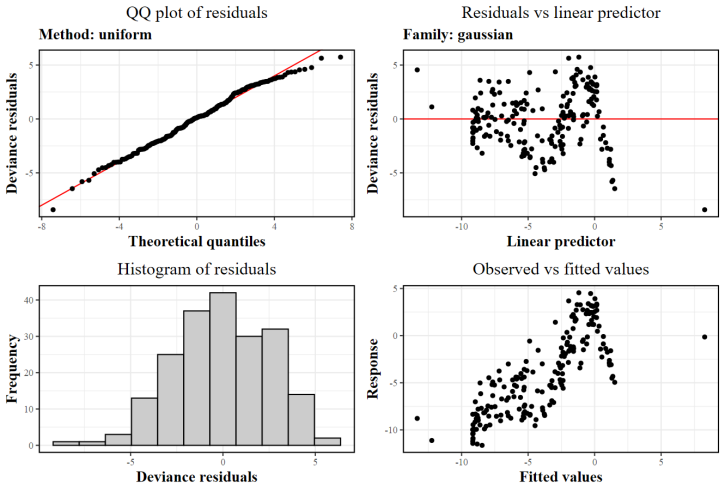
- `s(x, y, bs = "…")` | Simple, but covariates must use the same smooth type and preferably be on the same(ish) scale.
- `te(x, y, bs = c("…", "…"))` | More powerful, but results are aggregated.
- `s(x, bs = "…") + s(y, bs = "…") + ti(x, y, bs = "…")` | Almost as performant as the prior approaches, but much easier to interpret formula and outputs!

MORE DIAGNOSTICS AND MARGINAL PLOT

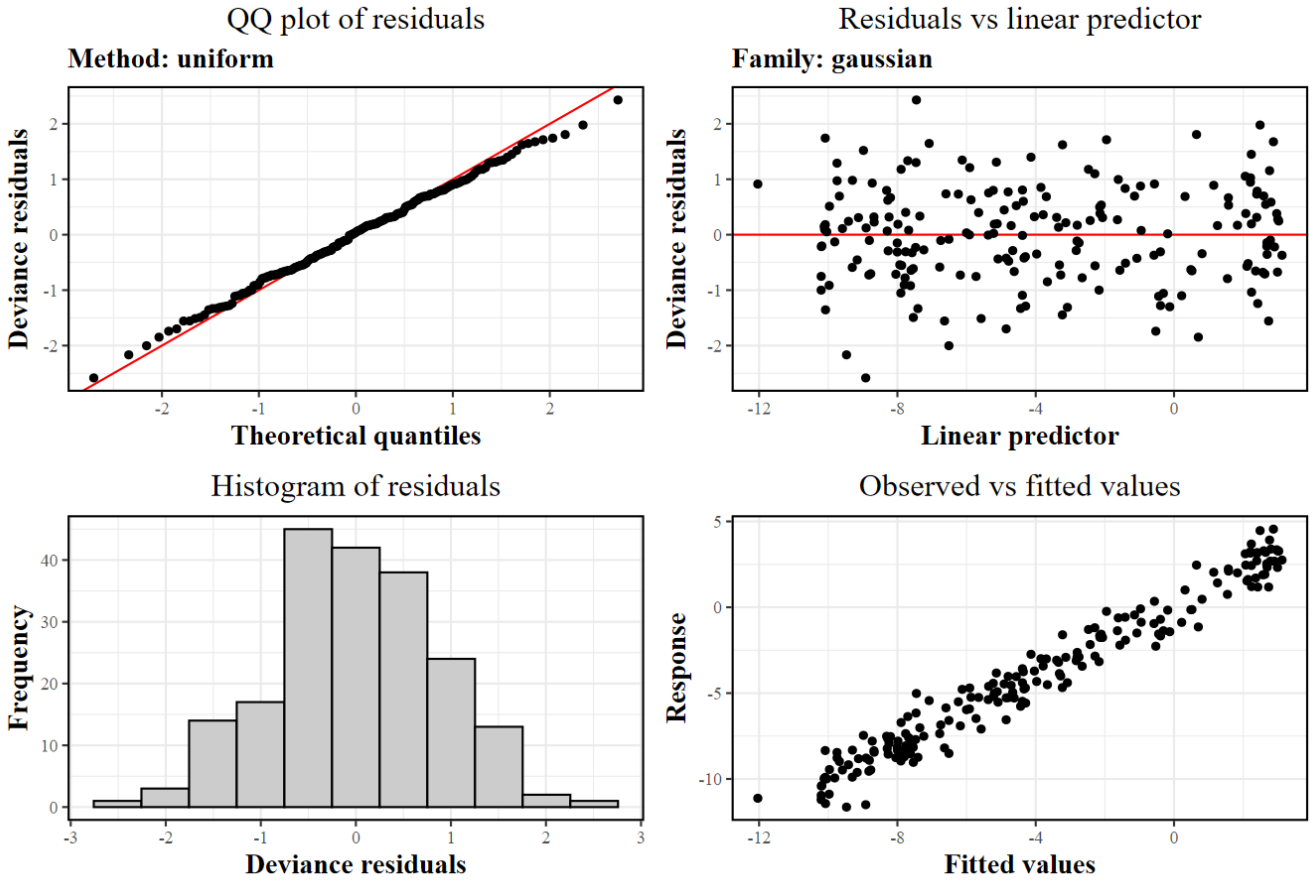
S(..., k = X) | "k" is often a very important parameter. It specifies the maximum degrees of freedom a specific smooth term may use.

The only downside to increasing "k" is that it is slower!

```
gam(y ~ s(x, by = z, k = 3), data = data, method = "REML") %>%
  appraise
```

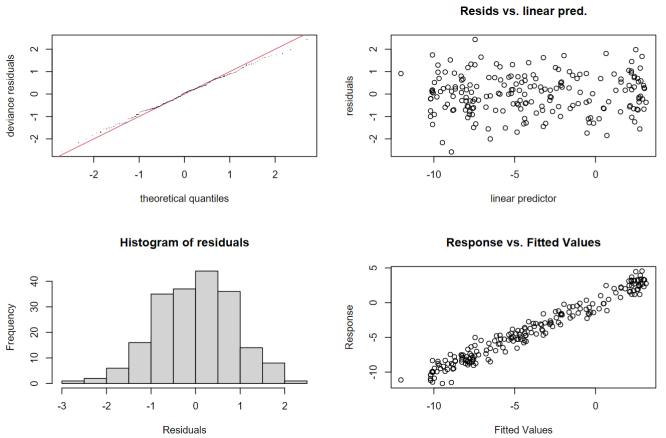


```
gam(y ~ s(x, by = z, k = 25), data = data, method = "REML") %>%
  appraise
```



GAM.CHECK

```
> gam(y ~ s(x, by = z, k = 25), data = data, method = "REML") %>%
+   gam.check

Method: REML   Optimizer: outer newton
full convergence after 5 iterations.
Gradient range [-2.515882e-06,5.118079e-07]
(score 330.3092 & scale 0.9271843).
Hessian positive definite, eigenvalue range [2.101391,98.16812]. ].
Model rank =  97 / 97

Basis dimension (k) checking results. Low p-value (k-index<1) may
indicate that k is too low, especially if edf is close to k'.

           k'   edf k-index p-value
s(x):za 24.00  7.53    0.97    0.30
s(x):zb 24.00  6.03    0.97    0.35
s(x):zc 24.00 12.58    0.97    0.30
s(x):zd 24.00  8.24    0.97    0.38
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```



AARHUS
UNIVERSITY