Introducing mgcViz

Matteo Fasiolo (University of Bristol, UK)

Joint work with:

Simon N. Wood (University of Bristol, UK)
Yannig Goude (EDF R&D)
Raphaël Nedellec (Talend, formerly EDF R&D)

matteo.fasiolo@bristol.ac.uk

Material available at:

https://github.com/mfasiolo/workshop_EDF19

This morning's plan

First session: basic mgcViz framework and tools

Second session: advanced mgcViz tools

Basic mgcViz: talk structure

These slides cover:

1. mgcViz: why do we need it and how does it work?
   - Limitations of mgcv plotting methods
   - mgcViz solution and smooth effects plots
2. Diagnostics and model selection tools
   - QQ-plots with qq.gamViz
   - Layered checks
3. Intro to hands-on session

mgcViz: why and how

Let's start with `mgcv`'s standard plotting methods:

```
fit <- gam(y ~ s(x1) + s(x2, x3))
plot(fit)
```

this calls `plot.gam`:

```
plot(x,residuals=FALSE,rug=NULL,se=TRUE,pages=0,
     select=NULL,scale=-1,n=100,n2=40,n3=3,pers=FALSE,
     theta=30,phi=30,jit=FALSE,xlab=NULL,ylab=NULL,
     main=NULL,ylim=NULL,xlim=NULL,too.far=.1,
     all.terms=FALSE,shade=FALSE,shade.col="gray80",
     shift=0,trans=I,seWithMean=FALSE,
     unconditional=FALSE,by.resids=FALSE,scheme=0,...)
```

Quite a lot of arguments!

mgcViz: why and how

Let's group the main arguments:

1. `se,residuals,rug` add optional graphical layers?
2. `se=TRUE` $\rightarrow$ `unconditional,seWithMean,shade,shade.col`
3. 1D effect $\rightarrow$ `n` or 2D $\rightarrow$ `n2,too.far,pers,jit,phi,theta`
4. `xlab,ylab,main,ylim,xlim` labs and limits
5. `select,all.terms` which effects to plot?
6. `page,n3` control layout of multiple plots
7. `...` extra graphical parameters

Problems:

- order in which layers are rendered is fixed
- cannot change all graphical settings via "`...`"
- difficult to add new layers

mgcViz: why and how

mgvViz's solution is:

1. break GAM model into individual effects

```
fit <- getViz( gam(y ~ s(x1) + s(x2, x3)) )
e <- sm(fit, 1)
```

2. plot effect using effect-specific methods

```
pl <- plot(e)  # calls plot.mgcv.smooth.1D()
```

3. implement layers separately from plots

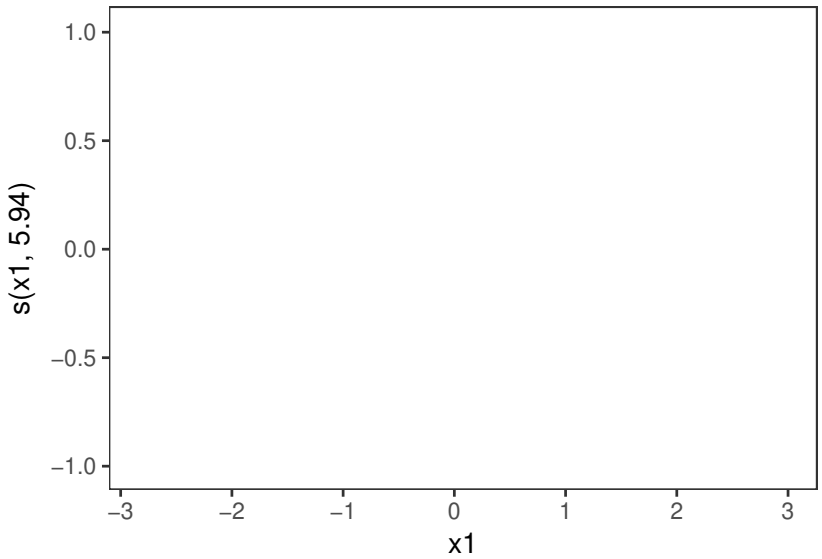
```
pl <- pl + l_fitLine() + l_ciLine()
```

4. modify applied using ggplot2 and rendering done at the end

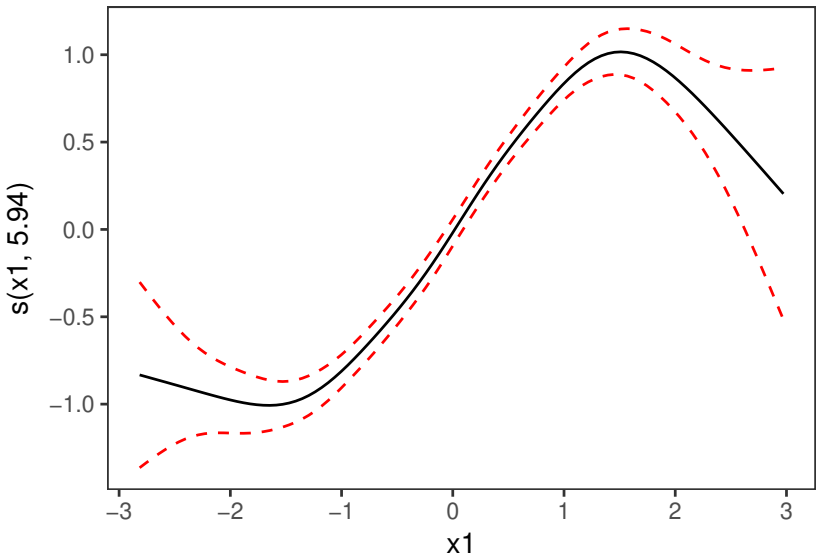
```
pl <- pl + xlim(1, 10) + xlab("myLabel")
pl # calls print.plotSmooth()
```

mgcViz: why and how

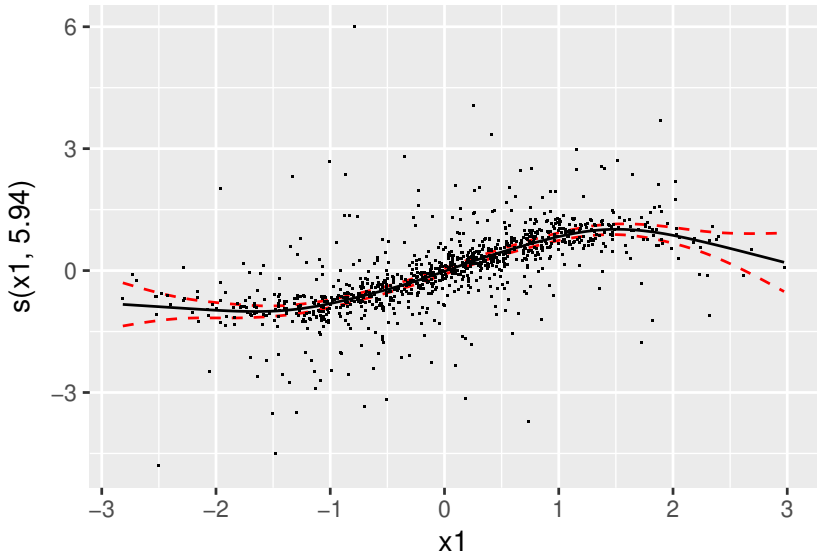
o <– plot(sm(b, 1))



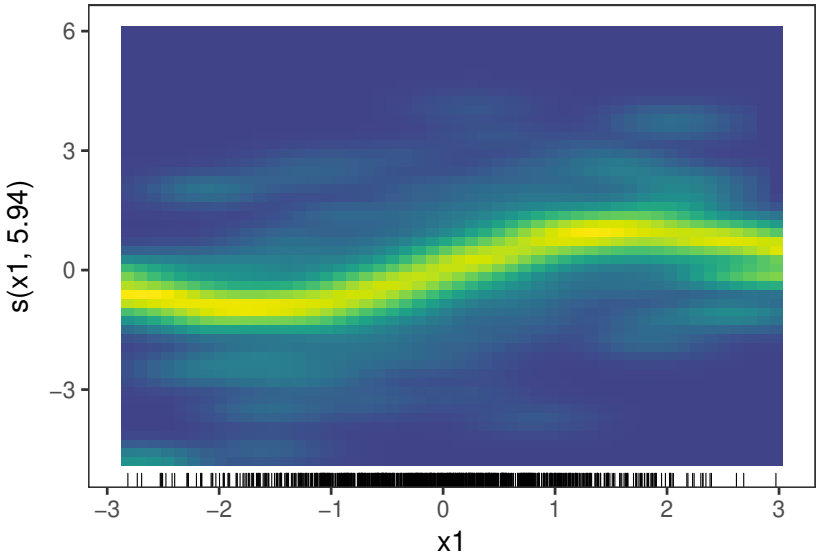
o + l_fitLine() + l_ciLine(col=2)



o + ... + l_points() + theme_get()

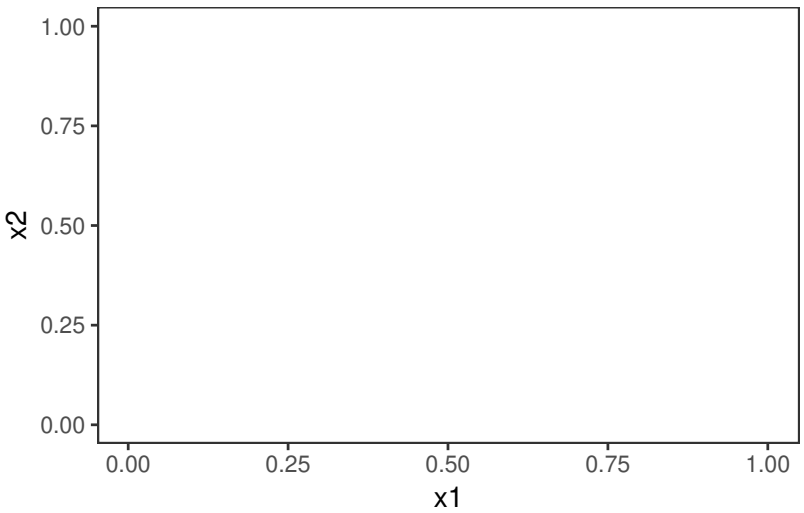


o + l_dens(type = "cond") + l_rug()

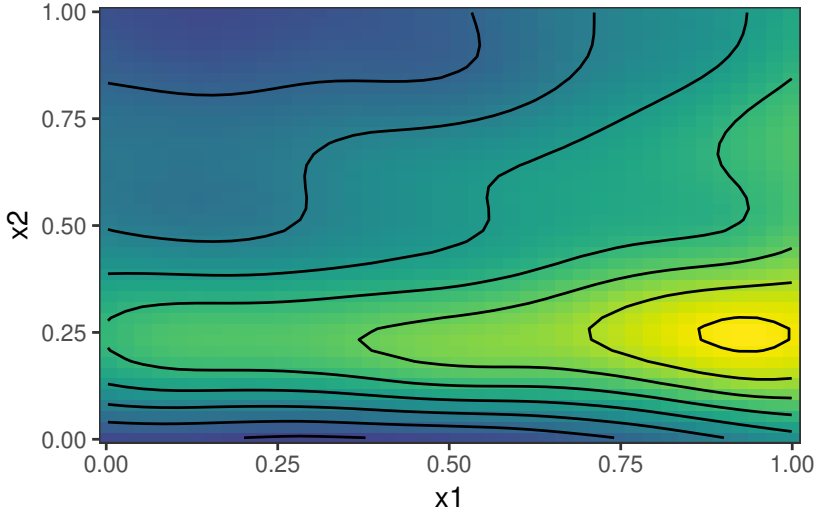


mgcViz: why and how

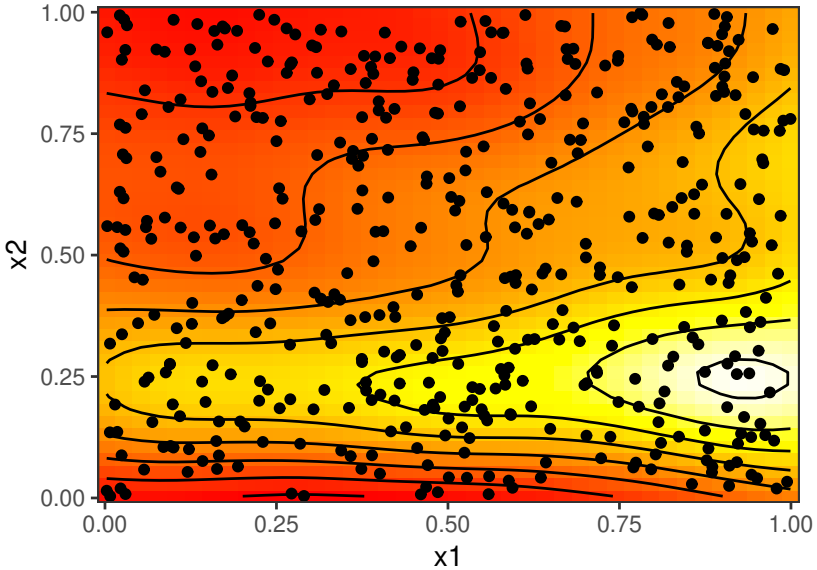
o <– plot(sm(b, 2))



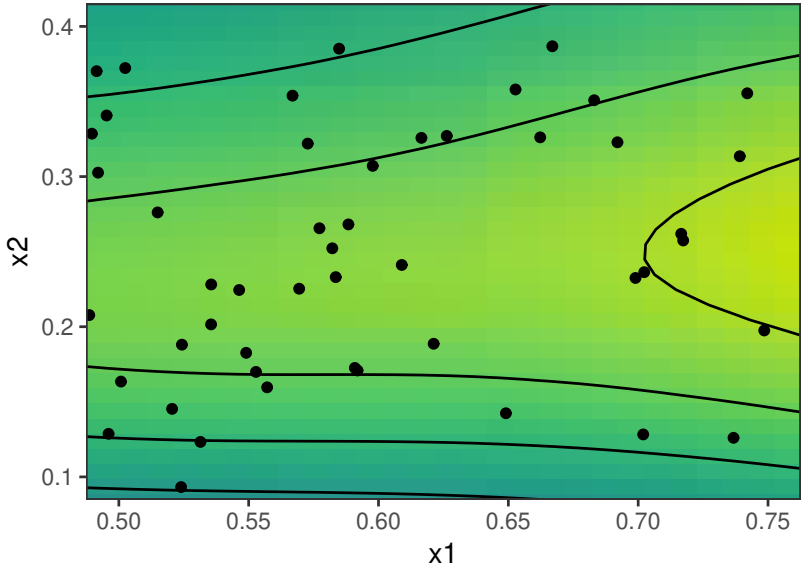
o + l_fitRaster() + l_fitContour()



o + l_points() +
scale_fill_gradientn(colours = heat.colors(50))



o + coord_cartesian(xlim =
c(0.5, 0.75), ylim = c(0.1, 0.4))



mgcViz: why and how

Basic workflow:

```
fit <- gam(y ~ s(x1) + s(x2, x3))
fit <- getViz(fit) # convert to "gamViz" object
```

or in one step:

```
fit <- gamV(y ~ s(x1) + s(x2, x3),      # see also bamV()
            aGam = list("method"="REML")# to gam()
            aViz = list("nsim" = 50))   # to getViz()
```

Then

```
pl <- plot(sm(fit, 1), n = 100)
pl + l_fitLine() + l_ciLine()
```

mgcViz: why and how

To list all available layers:

```
listLayers( plot(sm(fit, 1)) )  # 1D effect
[1] "l_ciLine"  "l_ciPoly"  "l_dens2D"  "l_fitDens"  ...

listLayers( plot(sm(fit, 2)) )  # 2D effect
[1] "l_den" "l_fitContour" "l_fitRaster"  "l_points" ...
```

To plot all effects do:

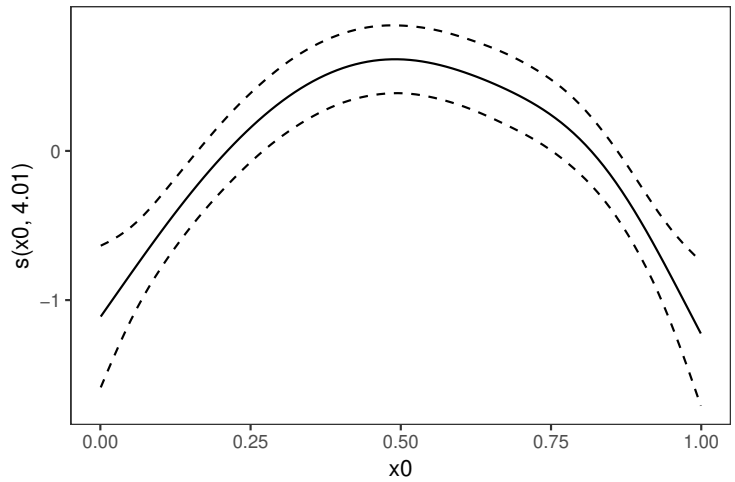
```
plot(fit) # calls plot.gamViz()
```

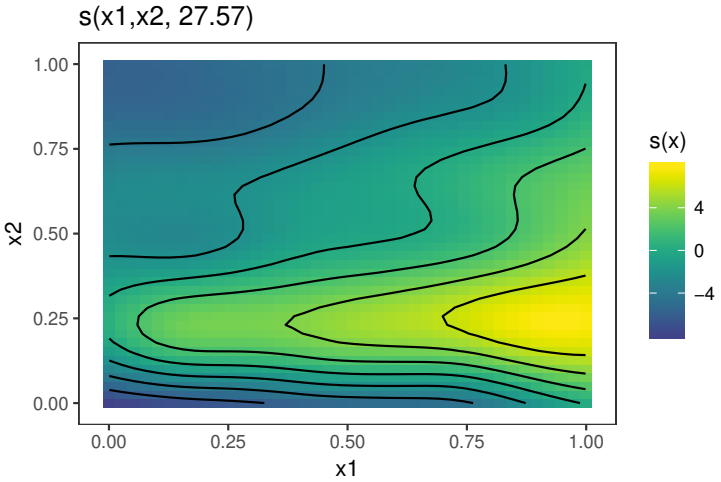
this loops over all effects, and adds default layers.
Layout controlled by `print.plotGam`:

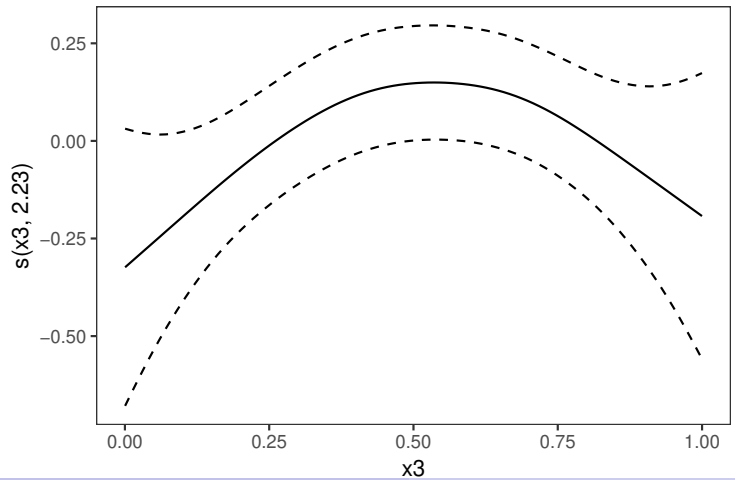
```
pl <- plot(fit)
print(pl, pages = 1)
```

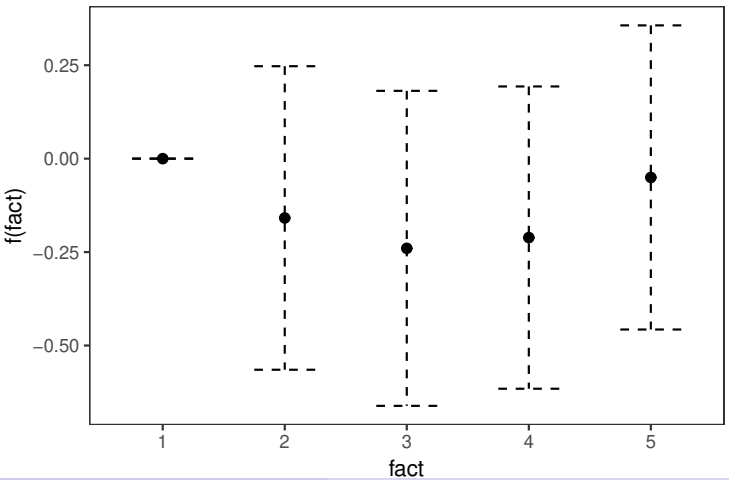
mgcViz: why and how

```
b <- bamV(y ~ fact + s(x0) + s(x1, x2) + s(x3), data=dat)
print(plot(b, allTerms = TRUE), pages = 1)
```



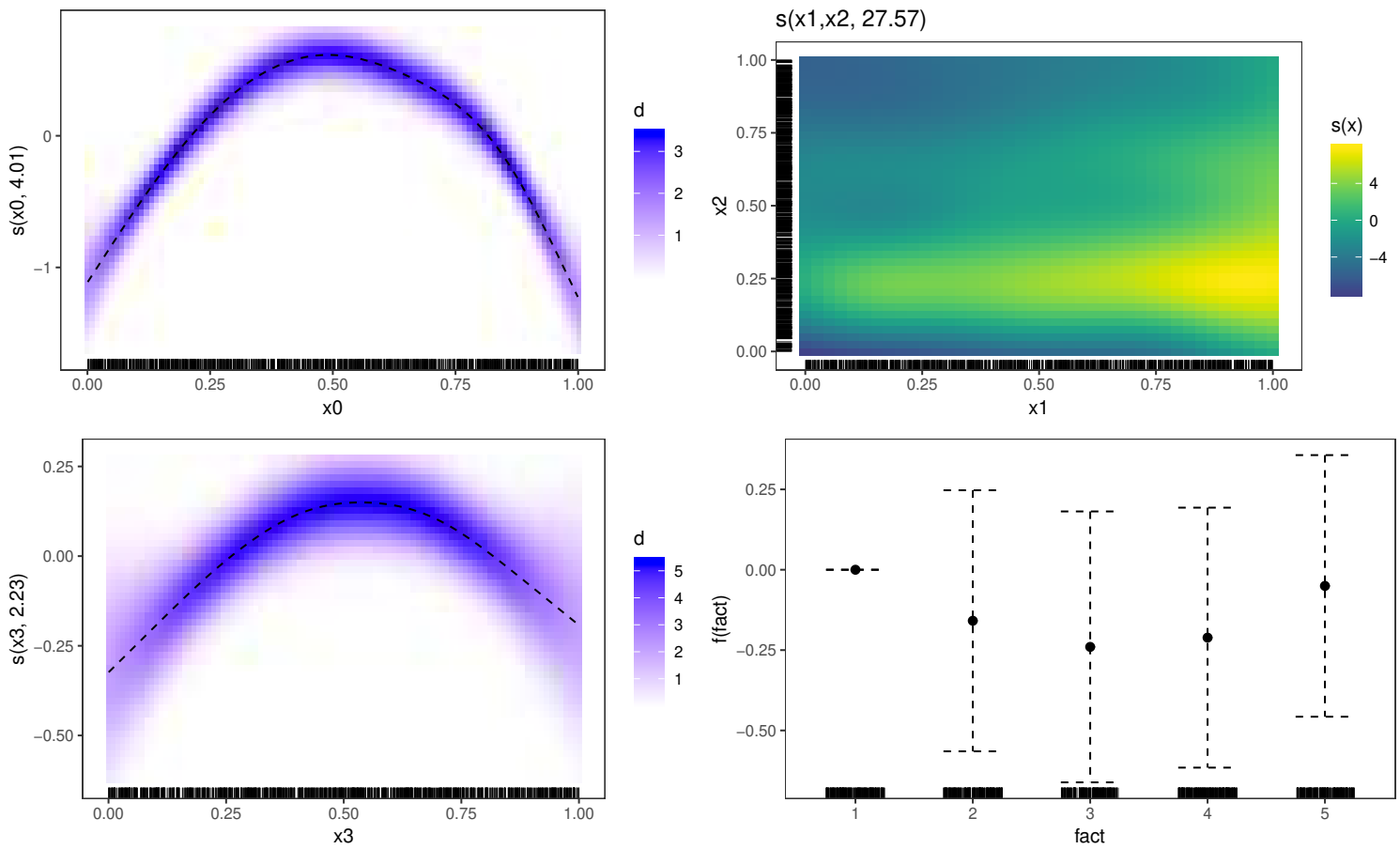






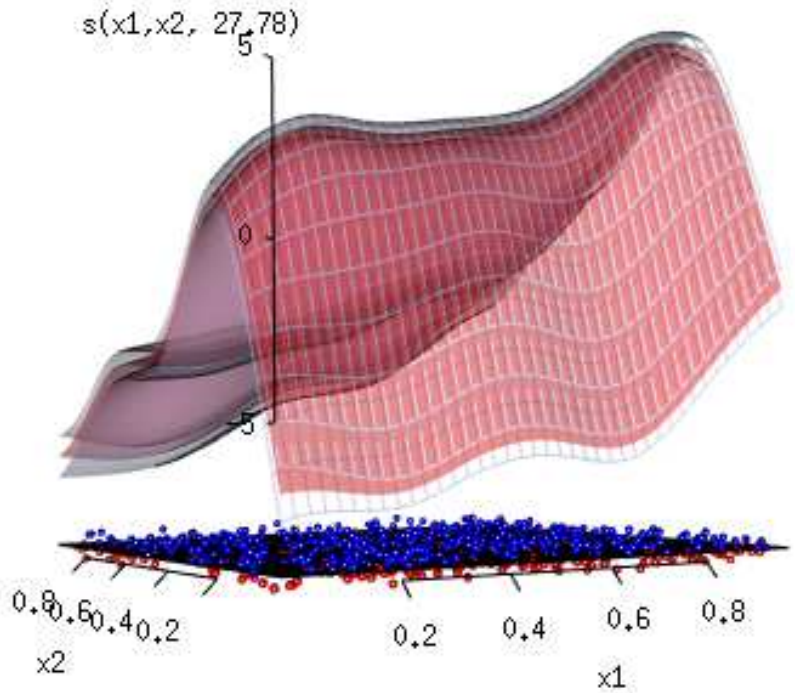
mgcViz: why and how

```
print(plot(b, allTerms = TRUE) + l_fitDens() +
          l_fitLine(linetype = 2)+ l_fitRaster() +
          l_fitPoints() + l_ciBar() + l_rug(), pages=1)
```



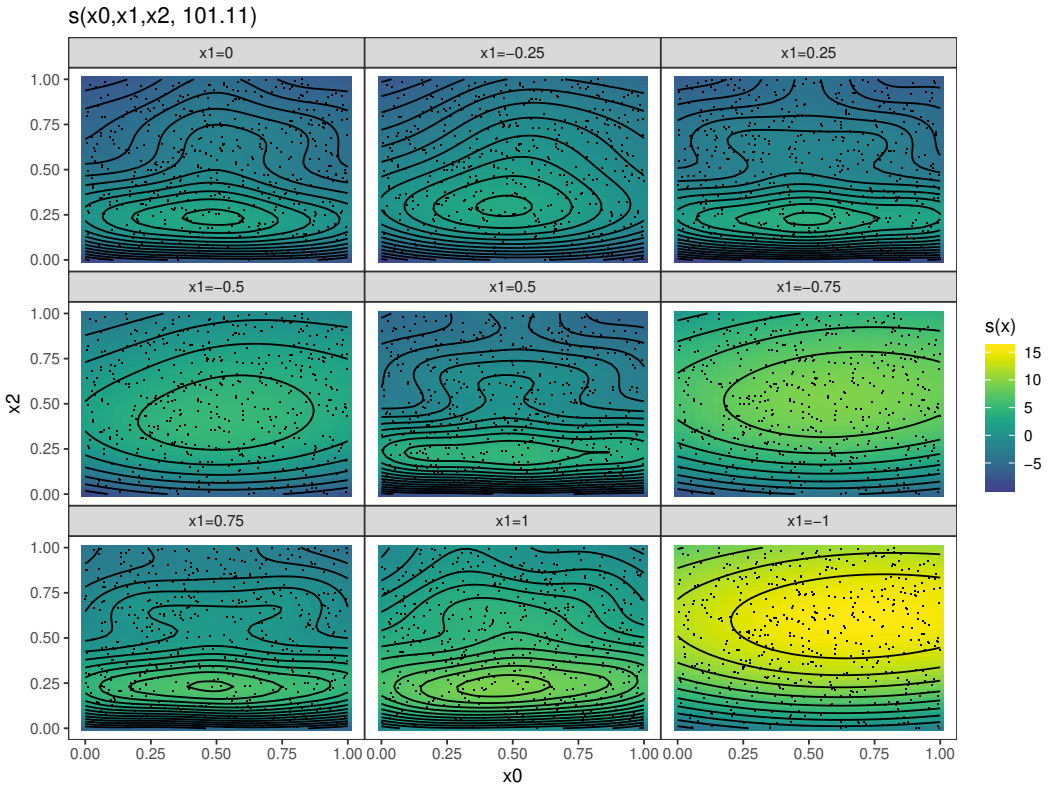
mgcViz: why and how

```
library(mgcViz) # see ?plotRGL.mgcv.smooth.2D
dat <- gamSim(1, n=1000, dist="normal", scale=2)
b <- gamV(y~s(x0)+s(x1, x2)+s(x3), data = dat)
plotRGL(sm(b, 2), residuals = TRUE) # <- not layered!
```



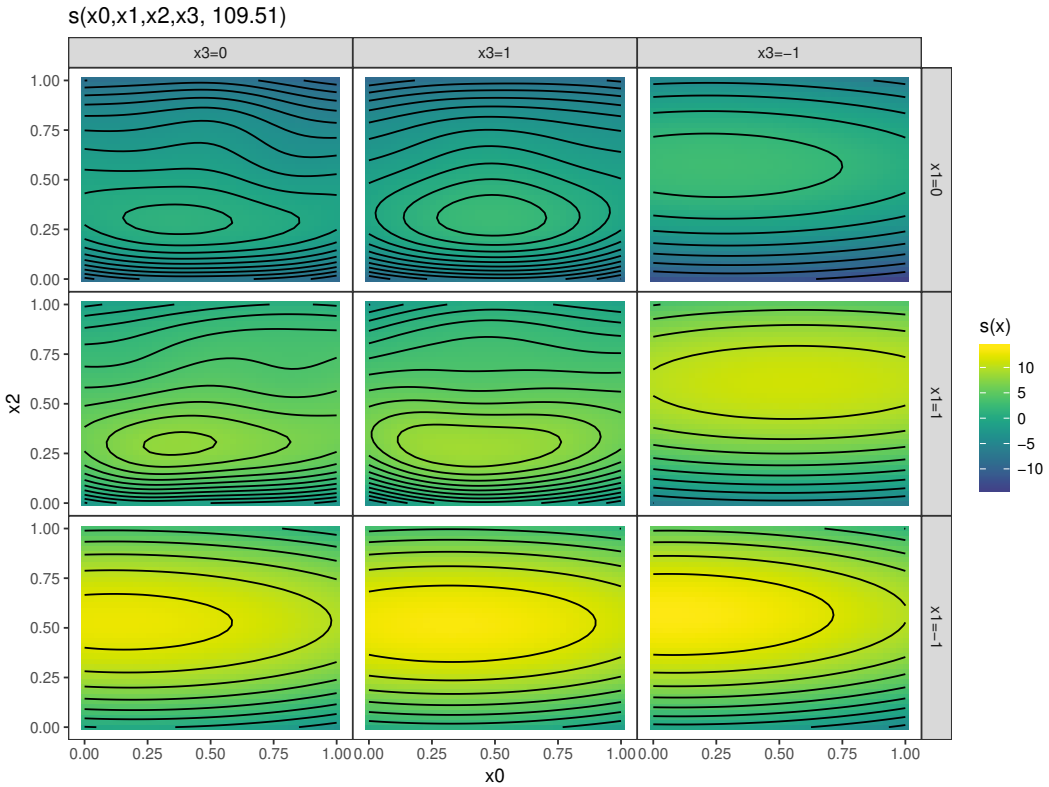
mgcViz: why and how

```
dat <- gamSim(1, n=5000, dist="normal", scale=2)
b <- bamV(y ~ s(x0, x1, x2), data = dat)
pl <- plotSlice(x = sm(b, 1),
                fix = list("x1" = seq(-1, 1, len = 9)))
pl + l_fitRaster() + l_fitContour() + l_points()
```



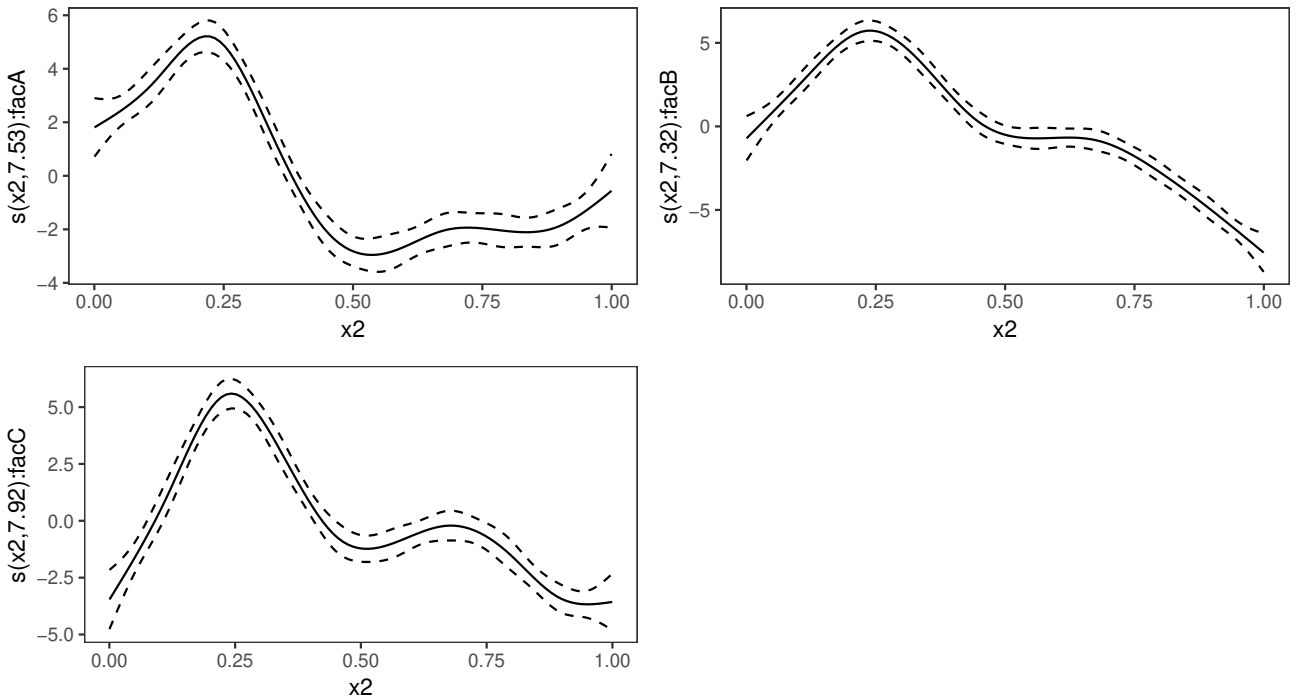
mgcViz: why and how

```
b <- bamV(y ~ s(x0, x1, x2, x3), data = dat)
pl <- plotSlice(x = sm(b, 1),
                fix = list("x1" = seq(-1, 1, len = 3),
                           "x3" = seq(-1, 1, len = 3)))
pl
```



mgcViz: why and how

```
dat <- gamSim(1,n=2000,dist="normal",scale=2)
dat$fac <- factor(sample(c("A","B","C"),nrow(dat),r=TRUE))
dat$y <- with(dat, dat$y + 30*(fac=="A")*(x2-0.5)^2 -
                           30*(fac=="B")*(x2-0.5)^3)
b <- gamV(y ~ s(x2, bs="cr", by = fac), data=dat)
print(plot(b), pages = 1)
```

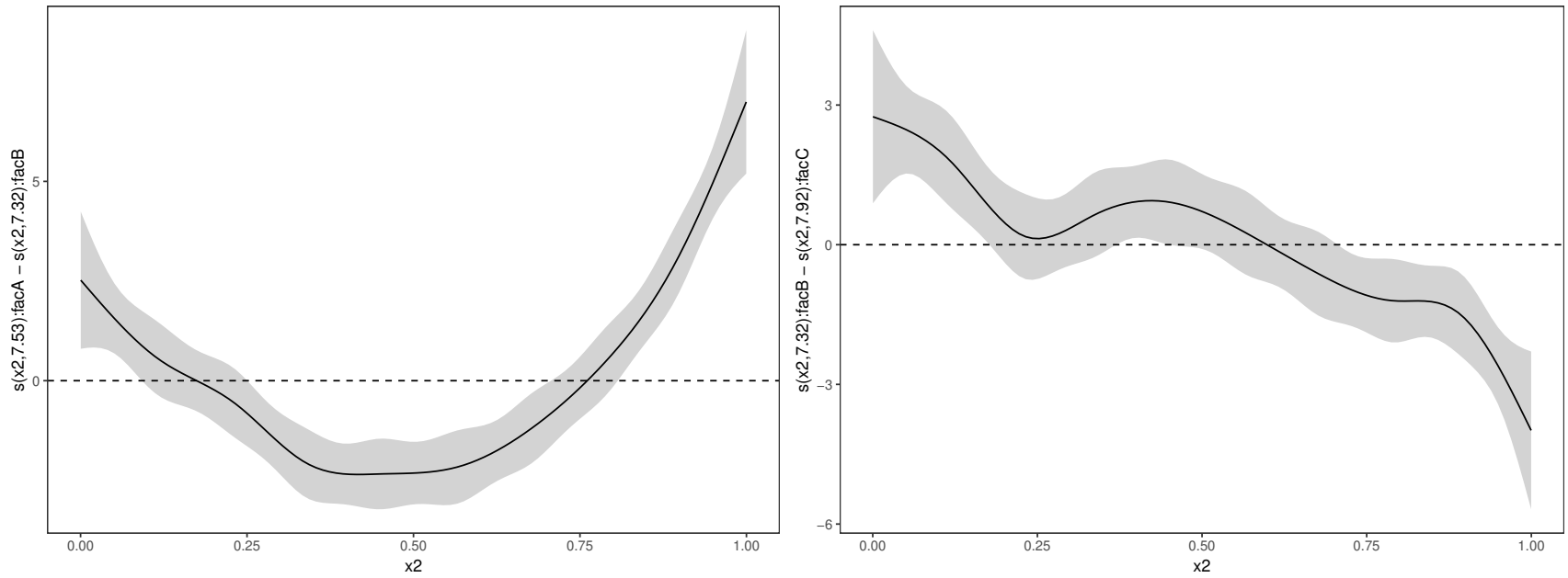


mgcViz: why and how

```
d1 <- plotDiff(s1=sm(b, 1), s2=sm(b, 2)) + l_ciPoly() +
      l_fitLine() + geom_hline(yintercept=0, linetype=2)

d2 <- plotDiff(s1=sm(b, 2), s2=sm(b, 3)) + l_ciPoly() +
      l_fitLine() + geom_hline(yintercept=0, linetype=2)

gridPrint(d1, d2, ncol = 2)
```



mgcViz: why and how

Smooth effect plots in `mgcViz`, summary:

- `o<-gam(...)` then `o<-getViz(o)`, or faster `o<-gamV(...)`
- `plot(sm(o,1))`, `plotDiff()`, ... output class `plotSmooth`
- can add `mgcViz` (start with `l_`) or `ggplot2` layers
- layers available depends on effect type
- `listLayers()` shows available `mgcViz` layers
- `plot(o)` plots all effects (as `plot.gam`)

Special smooth effects plots:

- `plotRGL` interactive 2D (not layered)
- `plotSlice` above 2D
- `plotDiff` for 1D, 2D and smooths on sphere

Basic mgcViz: talk structure

This set of slides covers:

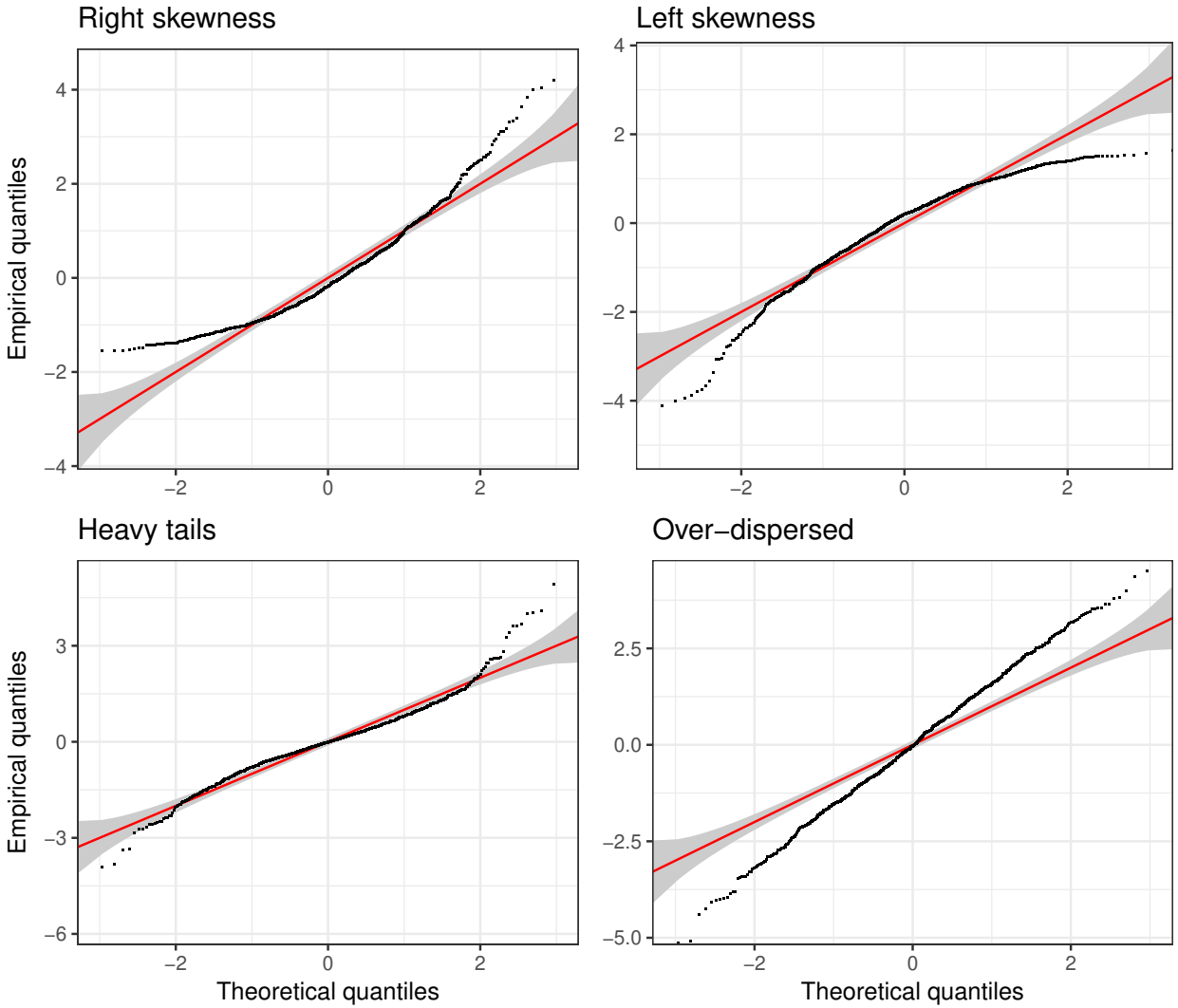
1. mgcViz: why do we need it and how does it work?
   - Limitations of mgcv plotting methods
   - mgcViz solution and smooth effects plots

2. Diagnostics and model selection tools
   - QQ-plots with qq.gamViz
   - Layered checks

3. Intro to hands-on session

Diagnostics and model selection tools

QQ-plots



Diagnostics and model selection tools

QQ-plots produced by `qq.gamViz`.

Usage:

```
b <- gamV(y ~ s(x))
qq(b, type = "deviance", CI = "normal") # see ?qq.gamViz
```

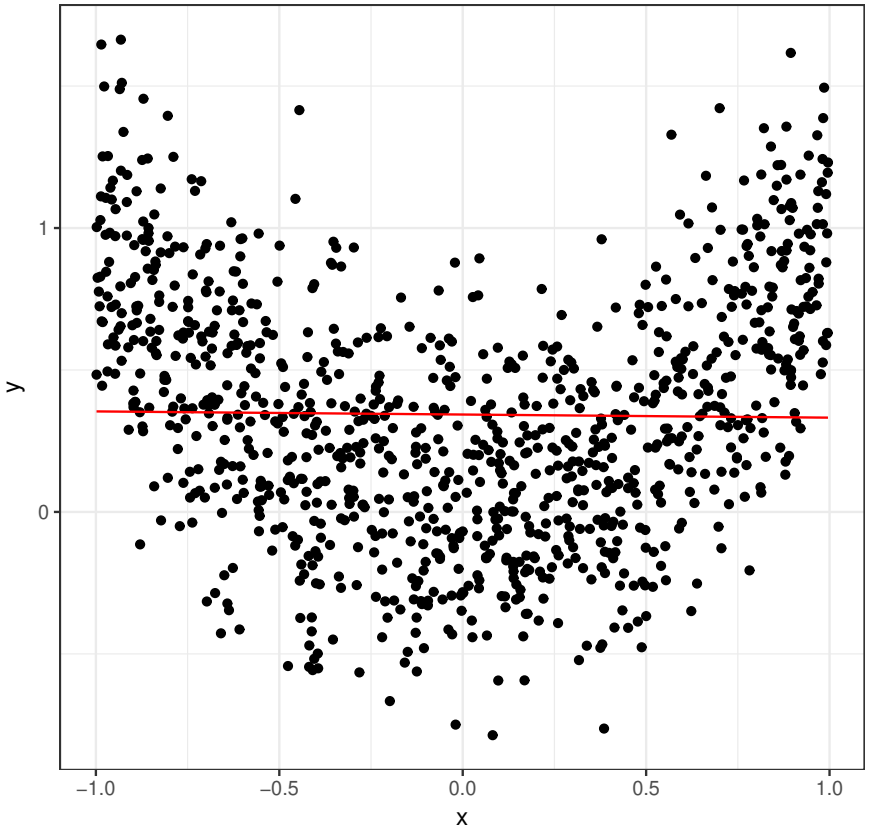
Notice:

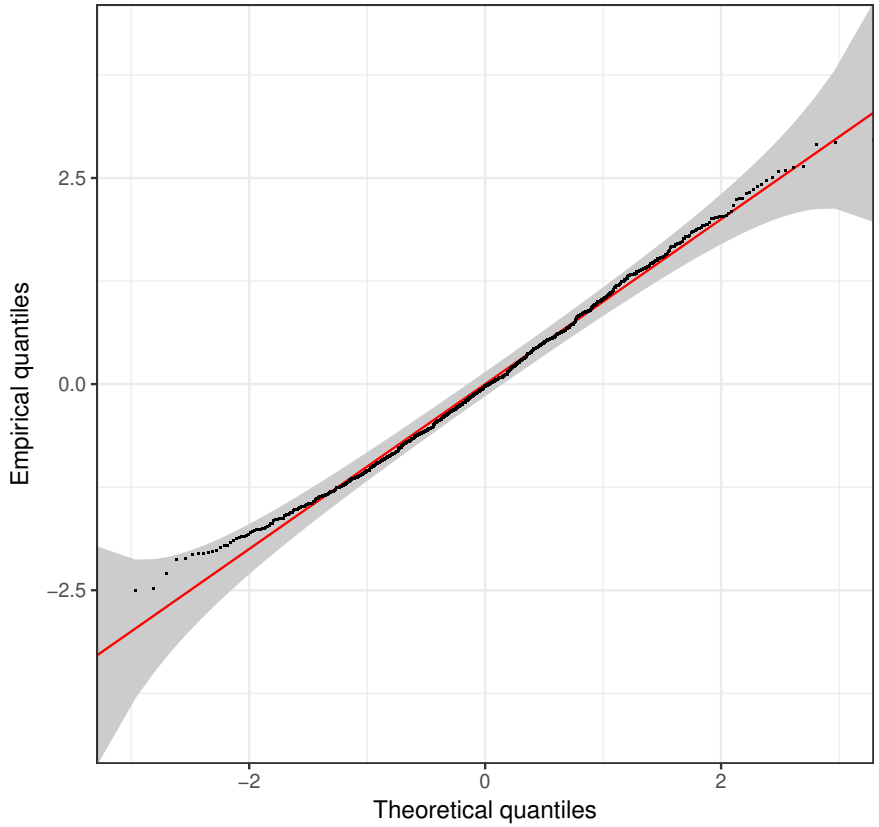
1. currently not layered
2. scales to large data sets (if `discrete = TRUE`)
3. interactivity by `shine(qq(fit))`

Diagnostics and model selection tools

QQ-plots useful for choosing $\text{Dist}_m(y|\mathbf{x})$ (e.g. Poisson vs Neg. Binom.)

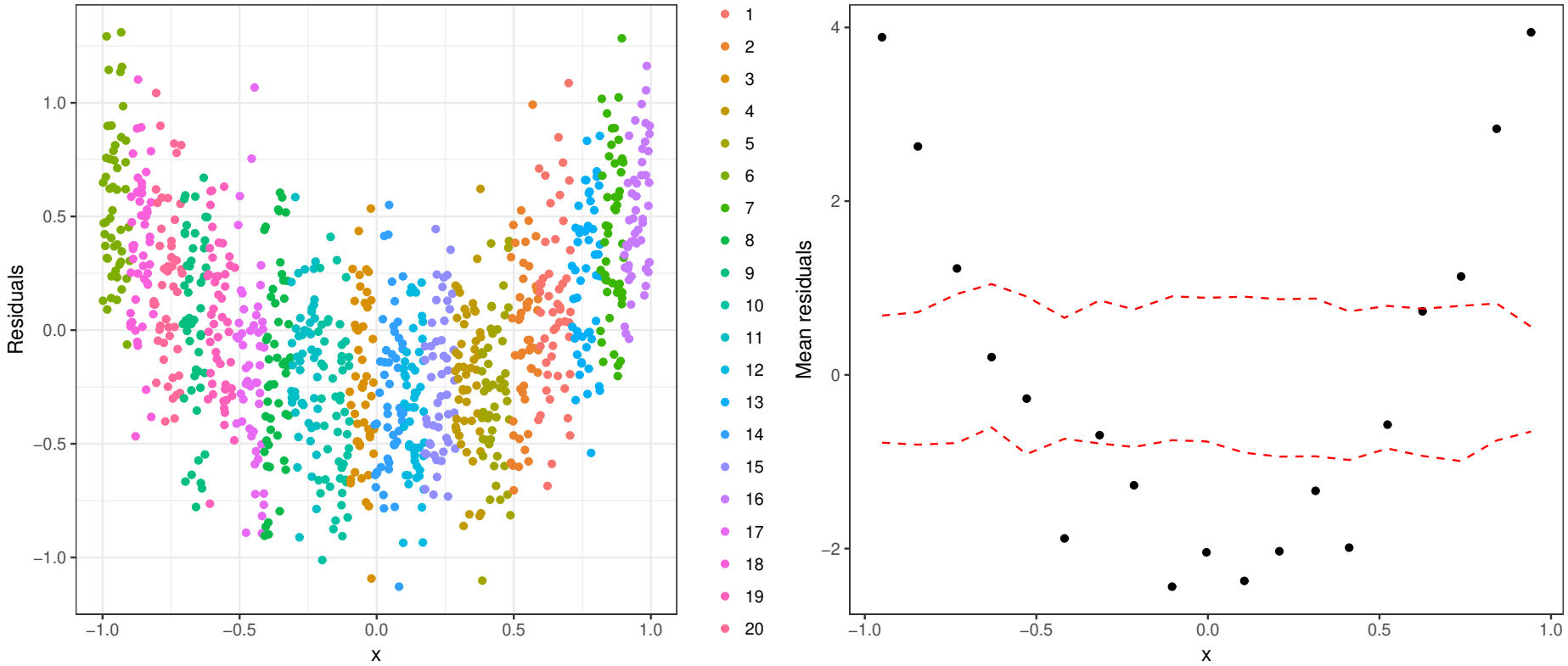
Less useful for finding omitted variables and non-linearities.





Diagnostics and model selection tools

Conditional residuals checks are more helpful here.

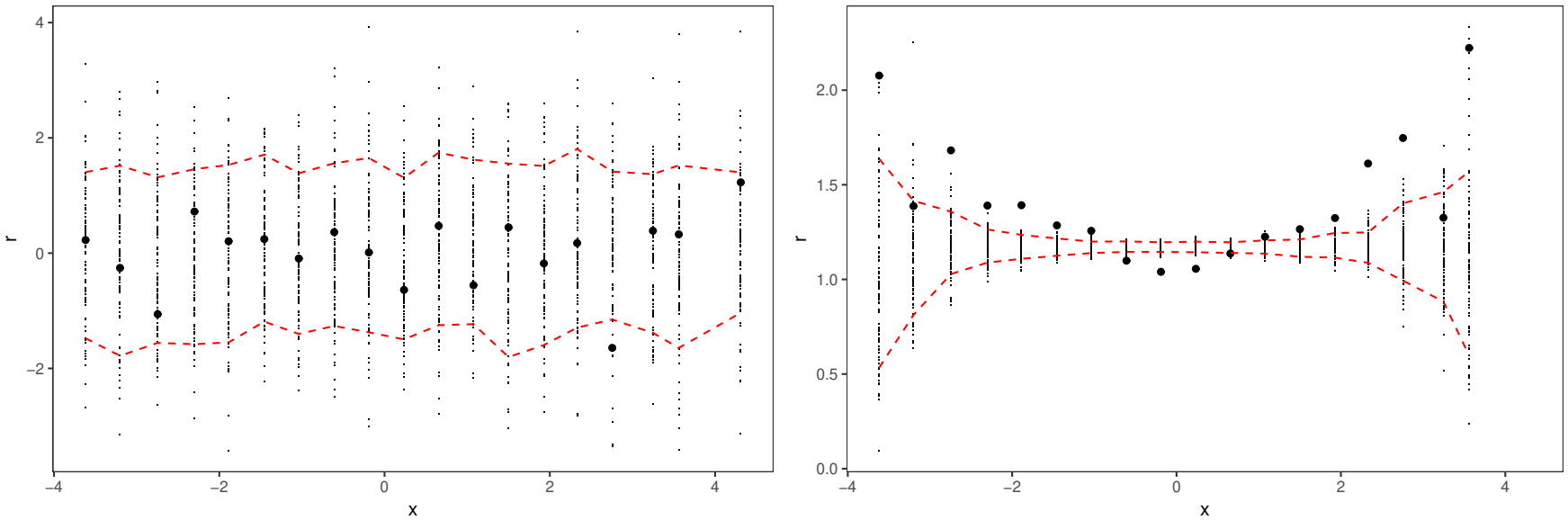


Diagnostics and model selection tools

`check1D` useful for 1D checks. Usage:

```
b <- gam(y ~ s(x) + s(z))
b <- getViz(b, nsim = 100)

ck <- check(b, "x")
gridPrint(ck + l_gridCheck1D(),
          ck + l_gridCheck1D(gridFun = sd), ncol = 2)
```



Diagnostics and model selection tools

Especially useful for GAMLSS models.

Example: `mgcFam::shash` family of Jones and Pewsey (2009):

```
library(mgcFam)
fit <- gam(list(y ~ s(x1),       # Location (mean)
                ~ x5 + s(x2), # Scale (variance)
                ~ s(x0) + x7, # Asymmetry (skewness)
                ~ s(x2, x4) ) # Tails (kurtosis)
           family = shash )
```

Should we put `x3` in model for skewness? `x5` in kurtosis formula?

```
library(e1071)
check1D(fit, "x3") + l_gridCheck1D(gridFun = skewness)

check1D(fit, "x5") + l_gridCheck1D(gridFun = kurtosis)
```

Diagnostics and model selection tools

`l_densCheck` compares empirical and theoret. residuals distribution.

Generates heat-map of:

$$\delta(x, r) = \texttt{dFun}\{\hat{p}(r|x) - \phi(r|x)\},$$

where

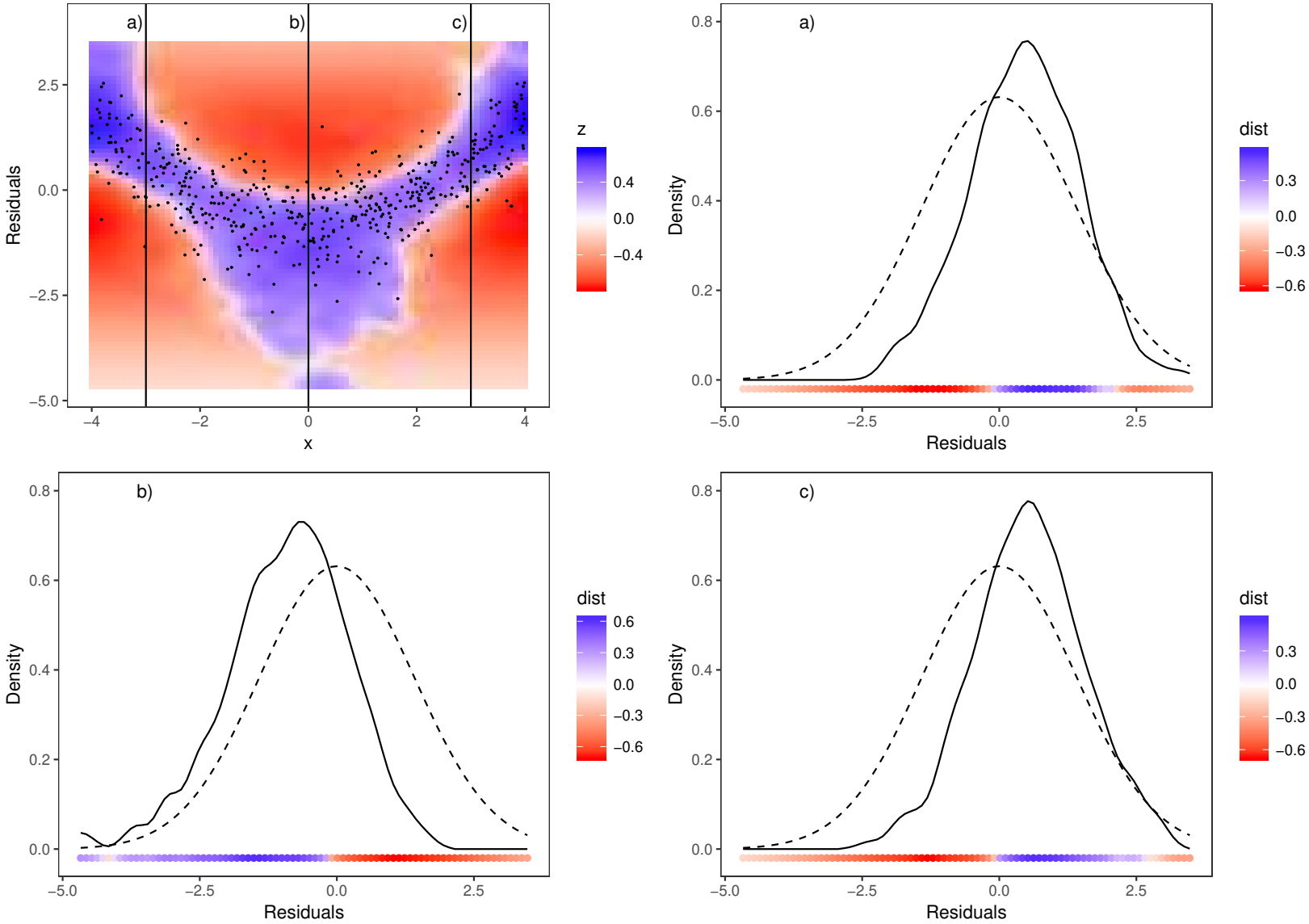
- $\hat{p}(r|x)$ k.d.e. of $p(r|x)$;
- $\phi(r|x)$ is theoretical distribution (generally $N(0, 1)$);
- `dFun` is a distance function. By default:
  `dFun <- function(emp, th) abs(sqrt(em)-sqrt(th))^(1/3);`

Usage

```
check1D(fit, "x3") + l_densCheck()
```

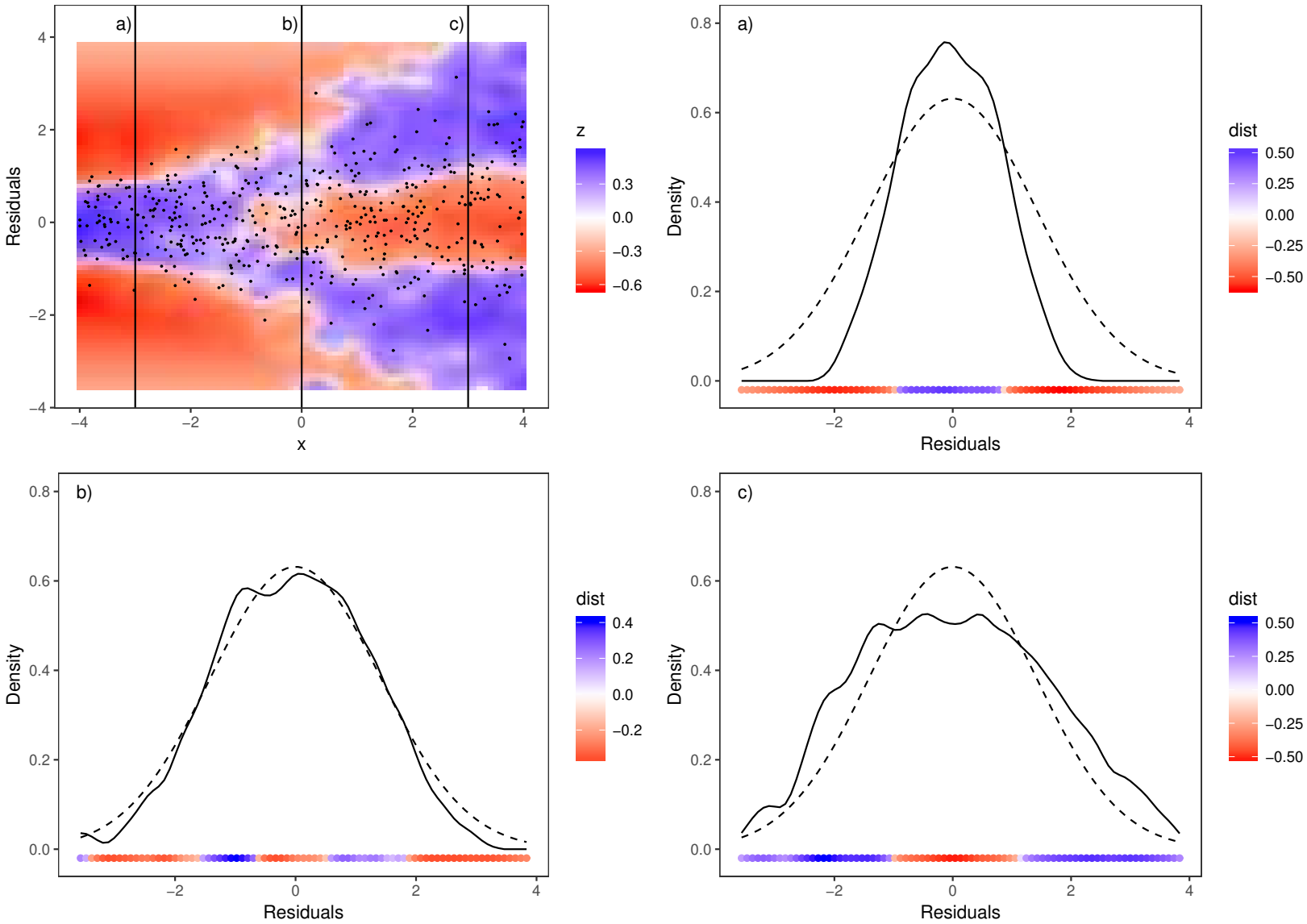
Diagnostics and model selection tools

Mean changes with x



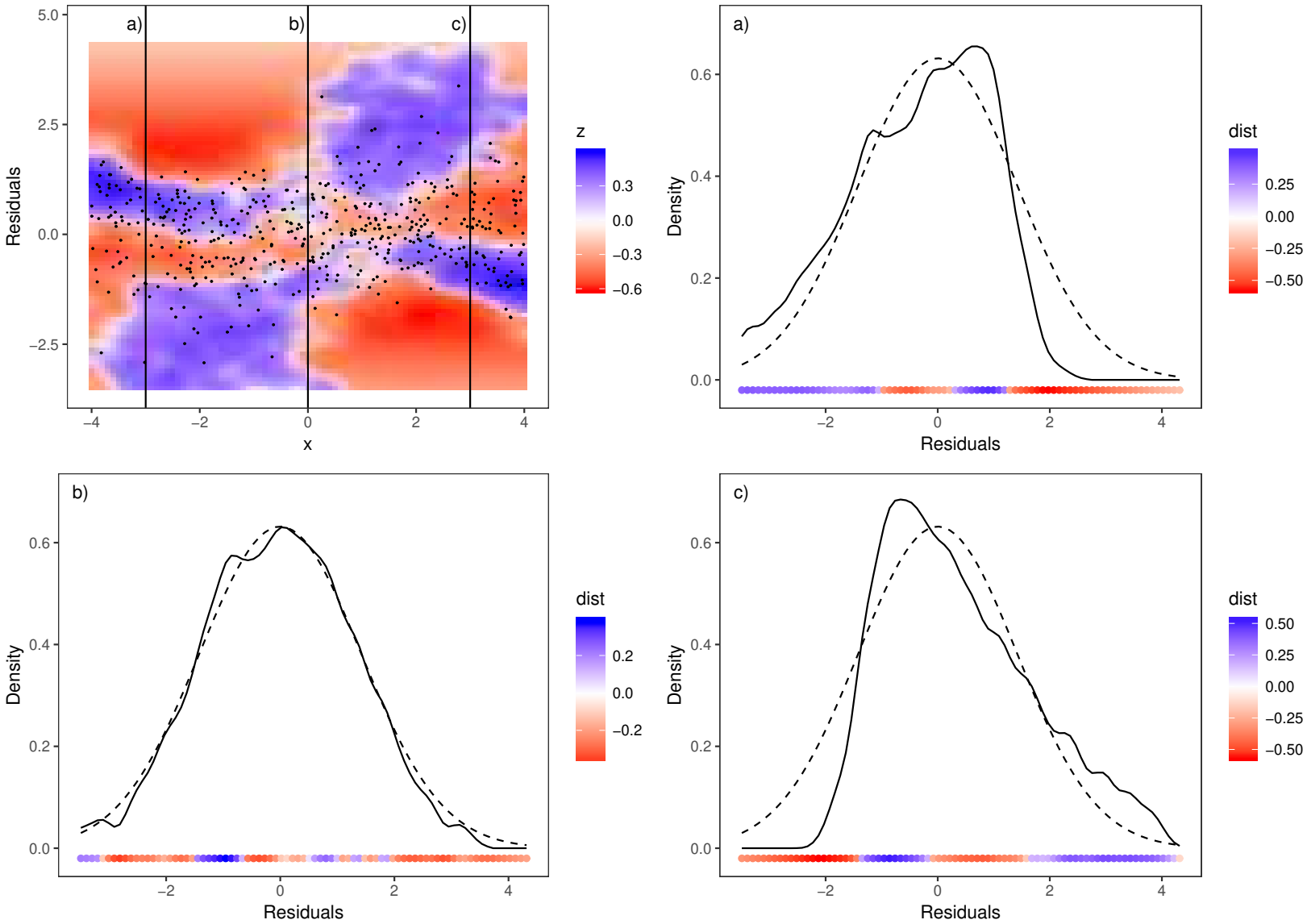
Diagnostics and model selection tools

Variance changes with x



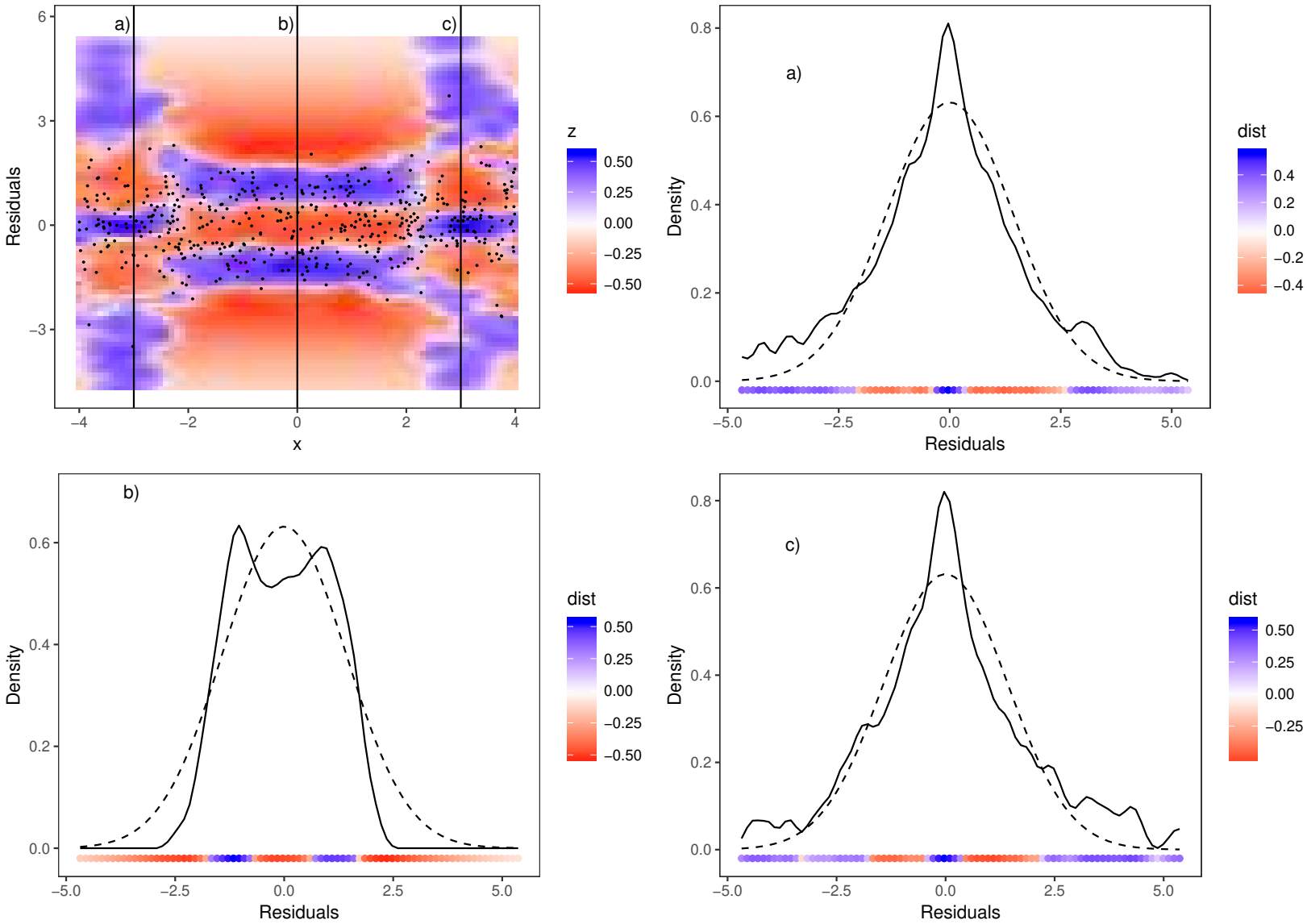
Diagnostics and model selection tools

Skewness (asymmetry) changes with x



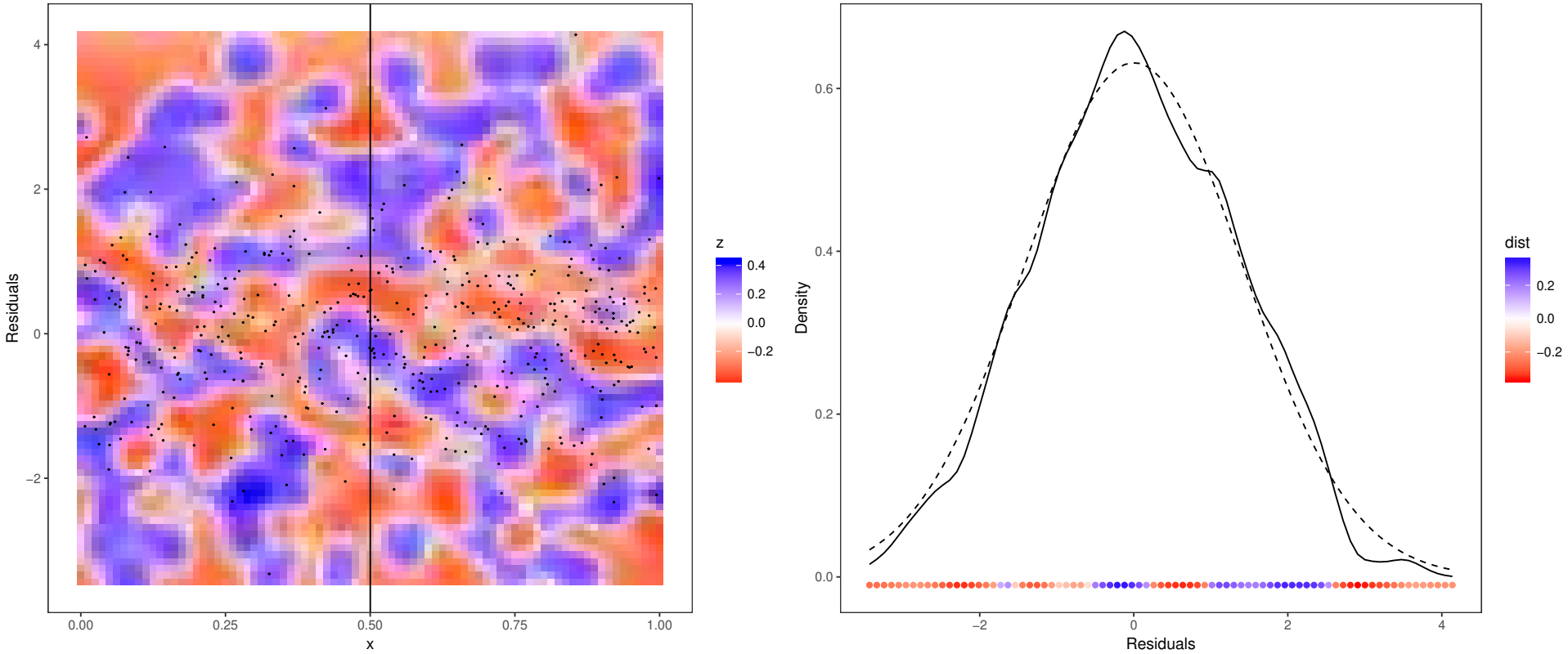
Diagnostics and model selection tools

Kurtosis (tail-weight) changes with x



Diagnostics and model selection tools

Nothing seems to change with `x`

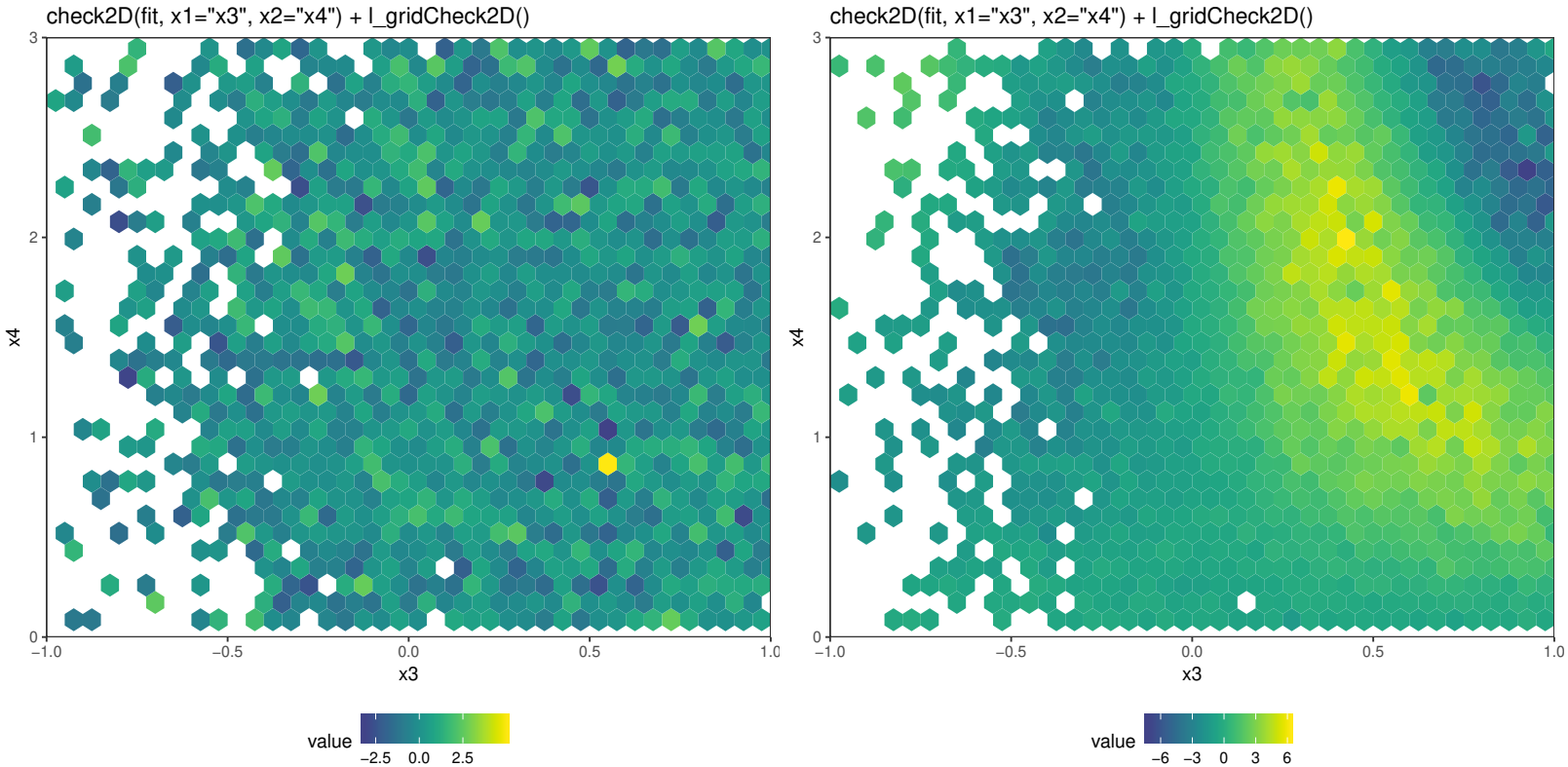


Diagnostics and model selection tools

In 2D we can use `check2D`. Usage:

```
b <- gamV(y ~ s(x1) + s(x2))

check2D(fit, "x1", "x2") + l_gridCheck2D()
check2D(fit, "x1", "x2") + l_gridCheck2D(gridFun = sd)
```



Diagnostics and model selection tools

Finally the `check(fit)` function produces:

```
Method: GCV   Optimizer: magic
Smooth param selection converged after 8 iterations.
The RMS GCV score gradient at convergence was 1.09e-05 .
The Hessian was positive definite.
Model rank =  37 / 37

...

##              k'  edf k-index p-value
## s(wM)      9.00 8.60    0.91  <2e-16 ***
## s(wM_s95) 9.00 8.13    1.02    0.76
```

Diagnostics and model selection tools

To summarize model checking in `mgcViz`:

- get `gamViz` object by doing

```
fit <- gam(y ~ s(x)) #Option 1
fit <- getViz(fit, nsim = 100)

fit <- gamV(y~s(x), aViz = list(nsim=100)) #Option 2
```

- QQ-plots using `qq(fit)` (no layers)
- Checks along 1D or 2D with `check1D` or `check2D` + `l_` ...
- Check convergence and number of basis with `check(fit)`

Intro to hands on session

Further `mgcViz` material on

```
https://mfasiolo.github.io/mgcViz/
```

In particular, see article "An introduction to mgcViz: visual tools for GAMs".

Also see our paper "Scalable visualisation methods for modern Generalized Additive Models" for an example on UK load forecasting.

References I

Fasiolo, M., R. Nedellec, Y. Goude, and S. N. Wood (2018). Scalable visualisation methods for modern generalized additive models. *arXiv preprint arXiv:1809.10632*.

Hastie, T. and R. Tibshirani (1990). *Generalized Additive Models*, Volume 43. CRC Press.

Jones, M. and A. Pewsey (2009). Sinh-arcsinh distributions. *Biometrika 96*(4), 761–780.

Rigby, R. A. and D. M. Stasinopoulos (2005). Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society: Series C (Applied Statistics) 54*(3), 507–554.

Wickham, H., W. Chang, et al. (2008). ggplot2: An implementation of the grammar of graphics. *R package version 0.7, URL: http://CRAN. R-project. org/package= ggplot2*.

Wood, S. (2006). *Generalized additive models: an introduction with R*. CRC press.