Appendix

R packages

The following is a non-exhaustive list of R packages which contain GAM functionality. Each is linked to the CRAN page for the package. Note also that several build upon the mgcv package used for this document.

brms Allows for Bayesian GAMs via the Stan modeling language (very new implementation).

CausalGAM This package implements various estimators for average treatment effects.

COZIGAM Constrained and Unconstrained Zero-Inflated Generalized Additive Models.

CoxBoost This package provides routines for fitting Cox models. See also cph in rms package for nonlinear approaches in the survival context.

gam Functions for fitting and working with generalized additive models.

GAMBoost: This package provides routines for fitting generalized linear and and generalized additive models by likelihood based boosting.

gamboostLSS: Boosting models for fitting generalized additive models for location, shape and scale (gamLSS models).

GAMens: This package implements the GAMbag, GAMrsm and GAMens ensemble classifiers for binary classification.

gamlss: Generalized additive models for location, shape, and scale.

gamm4: Fit generalized additive mixed models via a version of mgcv's gamm function.

gammSlice: Bayesian fitting and inference for generalized additive mixed models.

GMMBoost: Likelihood-based Boosting for Generalized mixed models.

gss: A comprehensive package for structural multivariate function estimation using smoothing splines.

mboost: Model-Based Boosting.

mgcv: Routines for GAMs and other generalized ridge regression with multiple smoothing parameter selection by GCV, REML or UBRE/AIC. Also GAMMs.

VGAM: Vector generalized linear and additive models, and associated models.

A comparison to mixed models

We noted previously that there were ties between generalized additive and mixed models. Aside from the identical matrix representation noted in the technical section, one of the key ideas is that the penalty parameter for the smooth coefficients reflects the ratio of the residual variance to the variance components for the random effects (see Fahrmeier et al., 2013, p. 483). Conversely, we can recover the variance components by dividing the scale by the penalty parameter.

To demonstrate this, we can set things up by running what will amount to equivalent models in both mgcv and lme4 using the sleepstudy data set that comes from the latter. I'll run a model with random intercepts and slopes, and for this comparison the two random effects will not be correlated. We will use the standard smoothing approach in mgcv, just with the basis specification for random effects - `bs='re'` . In addition, we'll use restricted maximum likelihood as is the typical default in mixed models.

See `?sleepstudy` for detail

```
library(lme4)
mixed_model = lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject),
                   data=sleepstudy)
ga_model = gam(Reaction ~  Days + s(Subject, bs='re') + s(Days, Subject, bs='re'),
               data=sleepstudy, method = 'REML')
```

We can see they agree on the fixed effects, but our output for the GAM is in the usual, albeit, uninterpretable form. So, we'll have to translate the smooth terms from the GAM to variance components as in the mixed model.

```
summary(mixed_model)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: Reaction ~ Days + (1 | Subject) + (0 + Days | Subject)
   Data: sleepstudy

REML criterion at convergence: 1743.7

Scaled residuals:
    Min      1Q  Median      3Q     Max
-3.9626 -0.4625  0.0204  0.4653  5.1860

Random effects:
 Groups    Name        Variance Std.Dev.
 Subject   (Intercept) 627.57   25.051
 Subject.1 Days         35.86    5.988
 Residual              653.58   25.565
Number of obs: 180, groups:  Subject, 18

Fixed effects:
            Estimate Std. Error t value
(Intercept)  251.405      6.885  36.513
Days          10.467      1.560   6.712

Correlation of Fixed Effects:
     (Intr)
Days -0.184
```

```
summary(ga_model)
```

```
Family: gaussian
Link function: identity

Formula:
Reaction ~ Days + s(Subject, bs = "re") + s(Days, Subject, bs = "re")

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  251.405      6.885  36.513  < 2e-16 ***
Days          10.467      1.560   6.712 3.67e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
                 edf Ref.df      F  p-value
s(Subject)     12.94     17  89.29 4.56e-07 ***
s(Days,Subject) 14.41     17 104.56 1.82e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.794   Deviance explained = 82.7%
-REML = 871.83  Scale est. = 653.58    n = 180
```

Conceptually, we can demonstrate the relationship with the following code that divides the scale by the penalty parameters, one for each of the smooth terms. However, there has been some rescaling behind the scenes regarding the Days effect, so we have to rescale it to get what we need.

```
rescaled_results = c(ga_model$reml.scale/ga_model$sp[1],
                     ga_model$reml.scale/(ga_model$sp[2]/ga_model$smooth[[2]]$S.scale),
                     NA)
lmer_vcov = VarCorr(mixed_model) %>% data.frame()
gam_vcov = data.frame(var=rescaled_results, gam.vcomp(ga_model))
```

grp	var1	vcov	sdcor
Subject	(Intercept)	627.57	25.05
Subject.1	Days	35.86	5.99
Residual	NA	653.58	25.57

	var	std.dev	lower	upper
s(Subject)	627.57	25.05	16.09	39.02
s(Days,Subject)	35.86	5.99	4.03	8.91
	NA	25.57	22.79	28.68

Think about it this way. Essentially what is happening behind the scenes is that effect interactions with the grouping variable are added to the model matrix (e.g. `~ ... + Days:Subject - 1`)[38]. The coefficients pertaining to the interaction terms are then penalized in the typical GAM estimation process. A smaller

estimated penalty parameter suggests more variability in the random effects. A larger penalty means more shrinkage of the random intercepts and slopes toward the population level (fixed) effects.

Going further, we can think of smooth terms as adding random effects to the linear component[39]. A large enough penalty and the result is simply the linear part of the model. In this example here, that would be akin to relatively little random effect variance.

Time and Space

One of the things to know about GAMs is just how flexible they are. Along with all that we have mentioned, they can also be applied to situations where one is interested in temporal trends or the effects of spatial aspects of the data. The penalized regression approach used by GAMs can easily extend such situations, and the mgcv package in particular has a lot of options here.

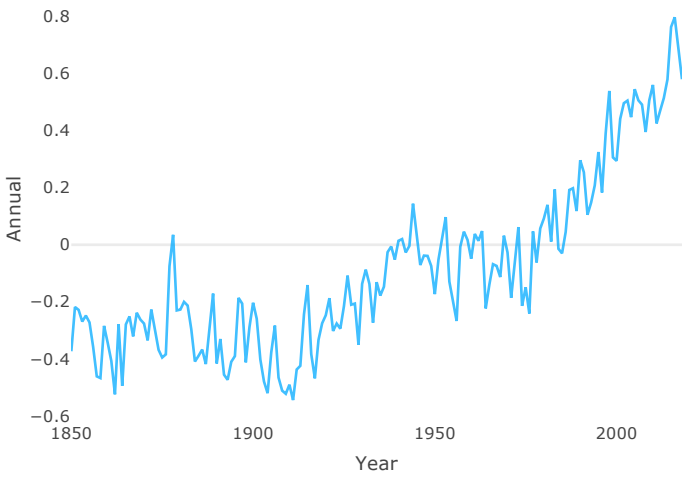
Time

A natural setting for GAMs is where there are observations over time. Perhaps we want to examine the trend over time. The SLiM would posit a linear trend, but we often would doubt that is the case. How would we do this with a GAM? We can incorporate a covariate representing the time component and add it as a smooth term. There will be some additional issues though as we will see.

Here I use the data and example at Gavin Simpon's nifty blog, though with my own edits, updated data, and different model. The data regards global temperature anomalies, which for some people assume does not exist, but for our purposes is actually in front of our face.

Simpson (2018) has offered a
this topic as well.

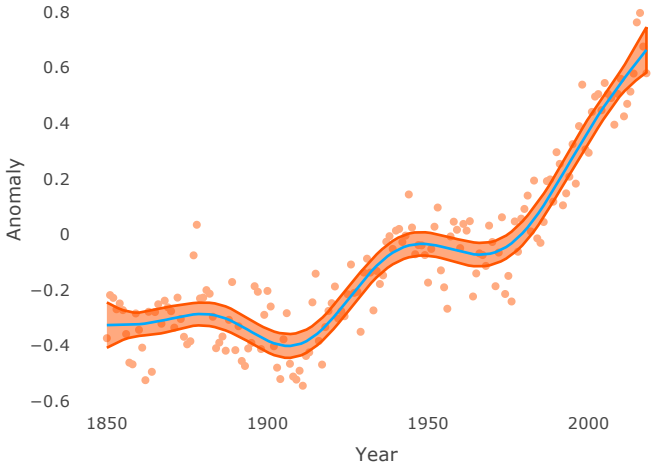
```
## Global temperatures
## gtemp = read.table("https://crudata.uea.ac.uk/cru/data/temperature/HadCRUT4-gl.dat",
load('data/global_temperatures.RData')

## Drop the even rows
gtemp = gtemp %>% drop_na()
```



Fitting a straight line to this would be disastrous, so let's do a GAM.

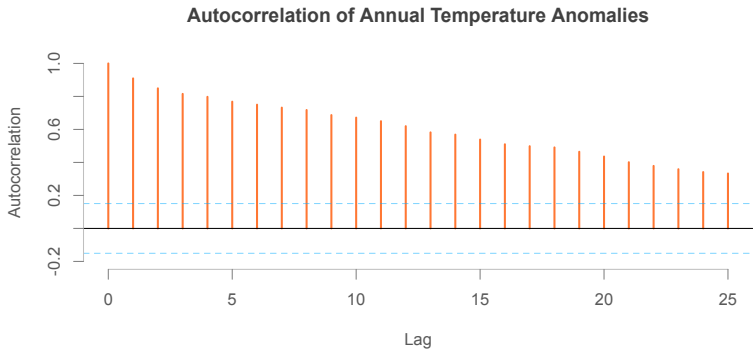
```
hot_gam = gam(Annual ~ s(Year), data=gtemp)
summary(hot_gam)
```

```
Family: gaussian
Link function: identity

Formula:
Annual ~ s(Year)

Parametric coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.090710   0.007653  -11.85   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
          edf Ref.df     F p-value
s(Year) 7.899  8.683 158.4  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.891   Deviance explained = 89.6%
GCV = 0.010449  Scale est. = 0.0098984  n = 169
```



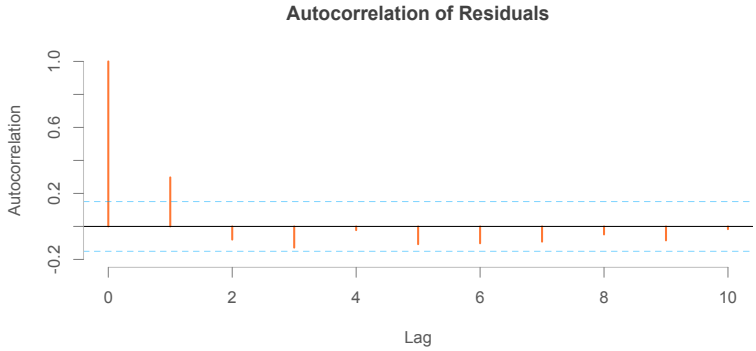
We can see that the trend is generally increasing, and has been more or less since the beginning of the 20th century. We have a remaining issue though. In general, a time series is autocorrelated, i.e. correlated with itself over time. We can see this in the following plot.

```
acf(gtemp$Annual)
```



What the plot shows is the correlation of the values with themselves at different lags, or time spacings. Lag 0 is it's correlation with itself, so the value is 1.0. It's correlation with itself at the previous time point, i.e. lag = 1, is 0.91, it's correlation with itself at two time points ago is slightly less, 0.85, and the decreasing trend continues slowly. The dotted lines indicate a 95% confidence interval around zero, meaning that the autocorrelation is still significant 25 years apart.

With our model, the issue remains in that there is still autocorrelation among the residuals, at least at lag 1.



The practical implications of autocorrelated residuals is that this positive correlation would result in variance estimates that are too low. However, we can take this into account with a slight tweaking of our model to incorporate such autocorrelation. For our purposes, we'll switch to the gamm function. It adds additional functionality for generalized additive *mixed* models, though we can just use it to incorporate autocorrelation of the residuals. In running this, two sets of output are provided, one in our familiar gam model object, and the other as a lme object from the nlme package.

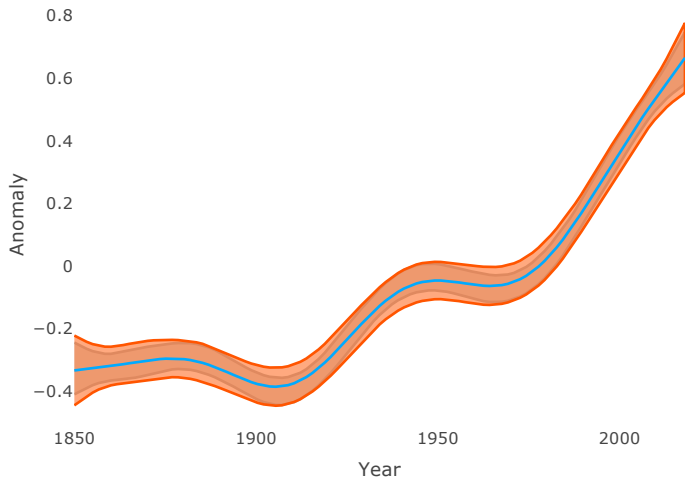
```
hot_gam_ar = gamm(Annual ~ s(Year), data=gtemp, correlation=corAR1(form = ~Year))
# summary(hot_gam)
summary(hot_gam_ar$gam)
```

```
Family: gaussian
Link function: identity

Formula:
Annual ~ s(Year)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.09114    0.01154  -7.897 4.18e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
         edf Ref.df     F p-value
s(Year) 6.79   6.79 89.31  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.889
  Scale est. = 0.010424  n = 169
```

```
summary(hot_gam_ar$lme)
```

```
Linear mixed-effects model fit by maximum likelihood
 Data: strip.offset(mf)
        AIC       BIC   logLik
  -282.6788 -267.0293 146.3394

Random effects:
 Formula: ~Xr - 1 | g
 Structure: pdIdnot
             Xr1      Xr2      Xr3      Xr4      Xr5      Xr6      Xr7      Xr8  Residual
StdDev: 1.227362 1.227362 1.227362 1.227362 1.227362 1.227362 1.227362 1.227362 0.1020979

Correlation Structure: AR(1)
 Formula: ~Year | g
 Parameter estimate(s):
      Phi
0.3672207
Fixed effects: y ~ X - 1
                  Value Std.Error  DF   t-value p-value
X(Intercept) -0.0911422 0.0115764 167 -7.873104  0.0000
Xs(Year)Fx1   0.4181527 0.1699447 167  2.460522  0.0149
 Correlation:
            X(Int)
Xs(Year)Fx1 0

Standardized Within-Group Residuals:
        Min          Q1         Med          Q3         Max
-2.19174564 -0.70736150  0.03003736  0.71267301  3.23213330

Number of Observations: 169
Number of Groups: 1
```

In the gam output, we see some slight differences from the original model, but not much (and we wouldn't expect it). From the lme output we can see the estimated autocorrelation value denoted as `Phi` [40]. Let's see what it does for our fit.



We can in fact see that we were a bit optimistic in the previous fit (non-colored band). Our new fit is wider at every point[41]. Thus, in using a GAM for time-series data, we have the same issues we'd have with standard regression settings, and we can deal with them in much the same way to get a better sense of the

uncertainty in our estimates.

Space

Consider a data set with latitude and longitude coordinates along with other covariates used to model some target variable. A spatial regression analysis uses an approach to account for spatial covariance among the observation points. A common technique used is a special case of Gaussian process which, as we noted, certain types of GAMs can be seen as such also. In addition, some types of spatial models can be seen similar to random effects models, much like GAMs. Such connections mean that we can add spatial models to the sorts of models covered by GAMs too.

When dealing with space, we may have spatial locations of a continuous sort, such as with latitude and longitude, or in a discrete sense, such as regions. In what follows we'll examine both cases.

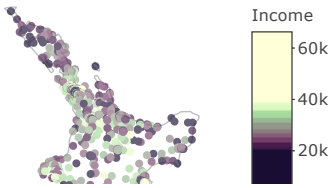
Continuous Spatial Setting

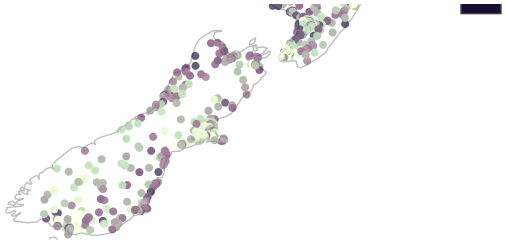
This example is inspired by tl you can find here.

Our example will use census data from New Zealand and focus on median income. It uses the nzcensus package[42] which includes median income, latitude, longitude and several dozen other variables. The latitude and longitude are actually centroids of the area unit, so this technically could also be used as a discrete example based on the unit.

Let's take an initial peek.

```
library(nzcensus)
nz_census <- AreaUnits2013 %>%
  filter(WGS84Longitude > 0 & !is.na(MedianIncome2013)) %>%
  rename(lon = WGS84Longitude,
         lat = WGS84Latitude,
         Income = MedianIncome2013) %>%
  drop_na()


map_data('nz') %>%
  filter(grepl(region, pattern='North|South')) %>%
  group_by(group) %>%
  plot_geo(x = ~nz_census$lon, y = ~nz_census$lat) %>%
  add_markers(color=~Income,
              colors=pal,
              # size=I(5),  # plotlys special bug
              opacity=.66,
              text=~ labs,
              marker=list(name=NULL),
              hoverinfo='text',
              data=as_tibble(nz_census)) %>%
  config(displayModeBar=F) %>%
  layout(title = '',
         geo = g1) %>%
  theme_plotly()
```





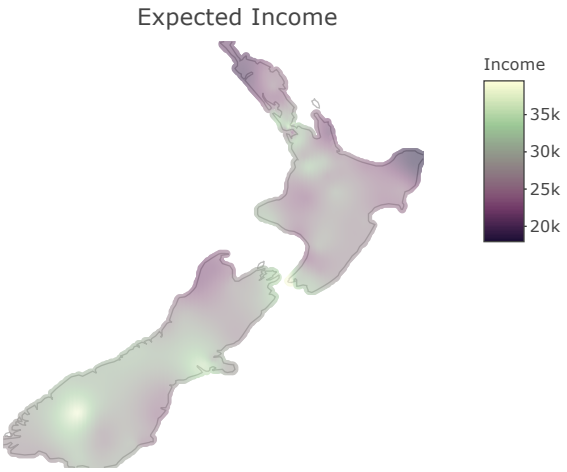
So we can go ahead and run a model predicting median income solely by geography. We'll use a Gaussian process basis, and allowing latitude and longitude to interact (bumping up the default wiggliness possible to allow for a little more nuance). What the GAM will allow us to do is smooth our predictions beyond the points we have in the data to get a more complete picture of income distribution across the whole area.

The `m=` argument allows on of covariance functions.

```
nz_gam = gam(Income ~ s(lon, lat, bs='gp', k=100, m=2), data=nz_census)
summary(nz_gam)
```

```
Family: gaussian
Link function: identity

Formula:
Income ~ s(lon, lat, bs = "gp", k = 100, m = 2)

Parametric coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  29497.8      148.1   199.2   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
             edf Ref.df     F p-value
s(lon,lat) 76.38   90.1 7.445  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =   0.27   Deviance explained = 30.1%
GCV = 4.0878e+07  Scale est. = 3.9105e+07  n = 1784
```

```
nz_map_data %>%
  mutate(Size=.3) %>%
  plot_geo(x=~lon, y=~lat, mode='markers', marker=list(size=~Size)) %>%
  layout(title = 'Expected Income',
         geo = g1) %>%
  config(displayModeBar=FALSE) %>%
  add_markers(x=~lon,
              y=~lat,
              color =~Income,
              opacity=.5,
              # size=~Size,
              data=heatvals,
              colors=scico::scico(100, palette = 'tokyo'), inherit = F
              )# %>%
```



This visualization was create
wondering, it was a notable c
make it, but in the end it the c

Using the Gaussian process smooth produces a result that is akin to a traditional spatial modeling technique called kriging. There are many other covariates to play with, as well as other bases that would be applicable, so you should feel free to play around with models that include those.

Alternatively, as we did with the time series, we could instead deal with spatial autocorrelation by specifying a model for the residual structure. First, we can simply test for spatial autocorrelation in the income variable via the well-worn Moran's I statistic. Given some weight matrix that specifies the neighborhood structure, such that larger values mean points are closer to one another, we can derive an estimate. The following demonstrates this via the ape package.

Using `scaled=T` results in a
goes from -1 to 1.

```
inv_dist = with(nz_census, 1/dist(cbind(lat, lon), diag = T, upper = T))
inv_dist = as.matrix(inv_dist)
## ape::Moran.I(nz_census$Income, weight = inv_dist, scaled=T)
```

observed	expected	sd	p.value
0.14	0	0	0

While statistically significant, there actually isn't too much going on, though it may be enough to warrant dealing with in some fashion. As with the time series, we'll have to use the functionality with gamm, where the underlying nlme package provides functions for spatial correlation structures. The following shows how this might be done.

If you run this be prepared to

```
gamm_spat = gamm(Income ~ s(x) + s(y) + z,
                 data=nz_census,
                 correlation = corSpatial(form = ~ lon + lat, type='gaussian'))
plot(gamm_spat)
```

So whether you choose to deal with the spatial autocorrelation explicitly by using something like coordinates as covariates in the model itself, or via the residual correlation structure, or perhaps both, is up to you.

For a nice discussion of this, Exchange, and note the top t "Why does including latitude account for spatial autocorre

Discrete

What about the discrete case, where the spatial *random effect* is based on geographical regions? This involves a penalty that is based on the adjacency matrix of the regions, where if there are g regions, the adjacency matrix is a $g \times g$ indicator matrix where there is some non-zero value when region i is connected to region j, and 0 otherwise. In addition, an approach similar to that for a random effect is used to incorporate observations belonging to specific regions. These are sometimes referred to as geoadditive models.

You'll be shocked to know that mgcv has a smooth construct for this situation as well, `bs='mrf'`, where `mrf` stands for Markov random field, which is an undirected graph.

This example comes from Ga itself is based on an article a

The following will model the percentage of adults with only a high school education. Unfortunately, when dealing with spatial data, getting it into a format amenable to modeling will often take some work. Specifically, mgcv will need a neighborhood list to tell it how the different areas are linked[43]. Furthermore, the data we want to use will need to be linked to the data used for mapping.

The first step is to read a shapefile that has some county level information. You could get this from census data as well.

Data found on GitHub.

```
# contiguous states c(1,4:6, 8:13, 16:42, 44:51, 53:56)
library(sp)
shp <- rgdal::readOGR('data/us_county_hs_only')
```

```
OGR data source with driver: ESRI Shapefile
Source: "/Users/micl/Documents/Stats/Repositories/Docs/generalized-additive-models/data/us_co
with 3233 features
It has 11 fields
Integer64 fields read as strings:  ALAND AWATER
```

```
## select michigan, and convert % to proportion
mich_df <- shp[shp$STATEFP %in% c(26), ] %>%   # add other FIPS codes as desired
  as.data.frame() %>%
  droplevels() %>%
  mutate(hsd = hs_pct / 100,
         county = stringr::str_replace(tolower(NAME), pattern='\\.', ''),
         county = factor(county))
```

The following creates a neighborhood list[44]. We also need names to match the values in the data, as well as the plotting data to be used later. I just made them lower case and remove punctuation. If you use more than one state, you will have to deal with duplicated names in some fashion.

```
nb <- spdep::poly2nb(shp[shp$STATEFP %in% c(26), ], row.names = mich_df$county)
names(nb) <- attr(nb, "region.id")
```

With neighborhood in place, we can now finally run the model. Note that the ID used for the smooth, in this case `county`, needs to be a factor variable. If not, you will get an uninformative error message that doesn't tell you that's the reason. There are a couple speed options used, but which would only be required if you're doing many states. Again, for this demonstration we'll not include any other covariates in the model, but normally you would include any relevant ones.

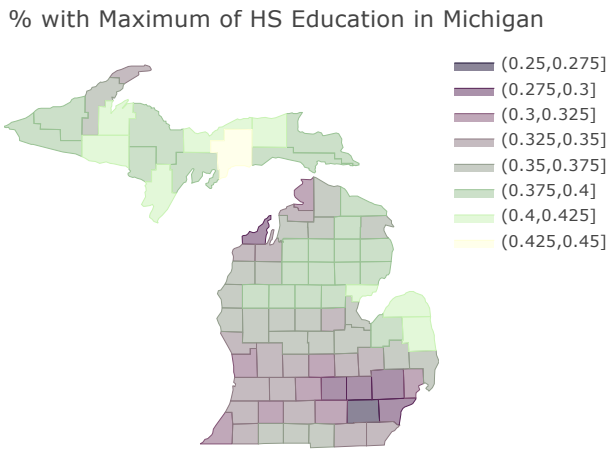
```
ctrl <- gam.control(nthreads = 6) # use 6 parallel threads, reduce if fewer physical CPU core

gam_mrf <- gam(hsd ~ s(county, bs = 'mrf', xt = list(nb = nb)), # define MRF smooth
               data = mich_df,
               method = 'REML',
               family = betar,  # fit a beta regression
               control = ctrl)
summary(gam_mrf)
```

```
Family: Beta regression(103.966)
Link function: logit

Formula:
hsd ~ s(county, bs = "mrf", xt = list(nb = nb))

Parametric coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.57837    0.02237  -25.85   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
            edf Ref.df Chi.sq p-value
s(county) 29.75  45.58  65.33  0.0297 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.492   Deviance explained = 67.4%
-REML = -113.47  Scale est. = 1         n = 83
```

```
mich_df = mich_df %>%
  mutate(fit = predict(gam_mrf, type='response'))
```

Now we can plot it. Plotly works with maps package objects that have been converted via ggplot2's map_data function. So, we create some plot-specific data, and then add our fitted values to it. We then add our own coloring based on the fitted values, and a custom clean theme.

```
plotdat = map_data("county", 'michigan') %>%
  left_join(mich_df, by = c('subregion' = 'county')) %>%
  mutate(fillcol = cut(fit, breaks=seq(.25, .45, by = .025)))

p = plotdat %>%
  group_by(subregion) %>%
  plot_ly(x = ~long, y = ~lat,
          color = ~fillcol,
          colors = scico::scico(100, palette = 'tokyo'),
          text = ~subregion, hoverinfo = 'text') %>%
  add_polygons(line = list(width = 0.4)) %>%
  layout(title = "% with Maximum of HS Education in Michigan") %>%
  theme_blank()
```



Be prepared, as this potentially will be a notable undertaking to sort out for your given situation, depending on the map objects and structure you're dealing with.

A Discrete Alternative

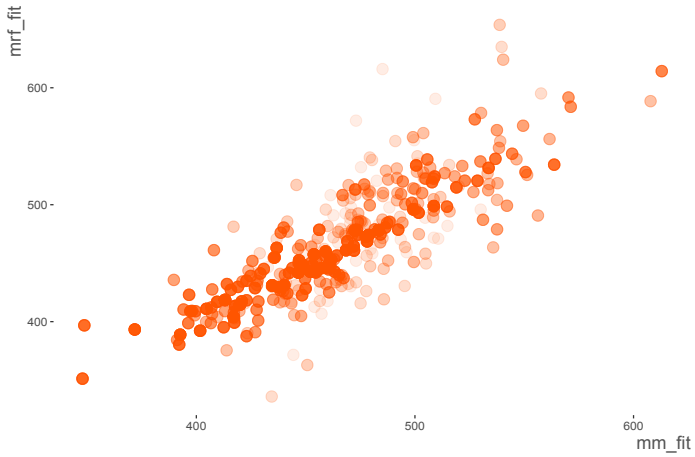
One of the things that has puzzled me is just how often people deal with geography while ignoring what would almost always be inherent correlation in discrete geographical or other units. In the social sciences for example, one will see a standard random effects approach, i.e. a mixed model, applied in the vast majority of situations where the data comes from multiple regions. This will allow for region specific effects, which is very useful, but it won't take advantage of the fact that the regions may be highly correlated with one another with regard to the target variable of interest.

We've already been using **gamm**, but haven't been using the typical random effects approach with it. We could do so here, but we can also just stick to the usual **gam** function, as it has a basis option for random effects. One thing that distinguishes the mixed model setting is that observations will be clustered within the geographical units. So for our example, we'll use the Munich rent data available from the gamlss family of packages, which contains objects for the Munich rent data and boundaries files of the corresponding districts from the 1999 survey. The rent99 data contains information about rent, year of construction, weather it has central heating, etc. Important for our purposes is the district identifier. The following shows the data structure.

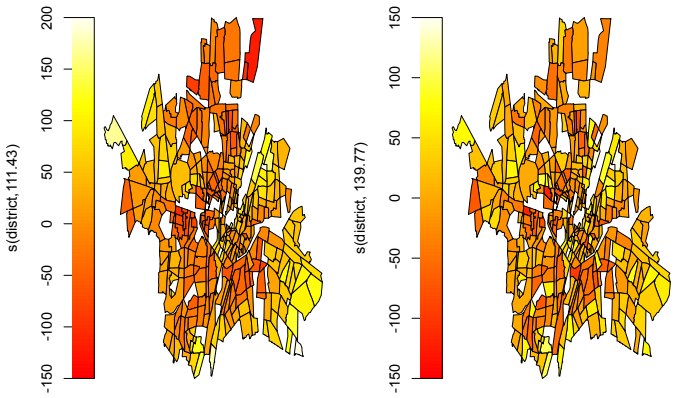
rent	rentsqm	area	yearc	location	bath	kitchen	cheating	district
110.0	4.23	26	1918	2	0	0	0	916
243.3	8.69	28	1918	2	0	0	1	813
261.6	8.72	30	1918	1	0	0	1	611
106.4	3.55	30	1918	2	0	0	0	2025
133.4	4.45	30	1918	2	0	0	1	561
339.0	11.30	30	1918	2	0	0	1	541
215.2	6.94	31	1918	1	0	0	0	822
323.2	10.43	31	1918	1	0	1	1	1713
216.3	6.76	32	1918	1	0	0	0	1812
245.3	7.43	33	1918	2	0	0	0	152
285.4	8.39	34	1918	2	0	0	0	943
238.3	6.81	35	1918	1	0	0	1	1711
374.5	10.70	35	1918	2	0	0	1	231
137.9	3.83	36	1918	2	0	0	1	411
188.4	4.96	38	1918	1	0	0	0	1711

Here again we'll use a Markov random field smooth, and for comparison a mixed model with a random effect for district. The plots show that, while close, they don't exactly come to the same conclusions for the district fitted values.

```
library(gamlss.data)

# prep data
rent99 = rent99 %>%
  mutate(district=factor(district))

rent99.polys[!names(rent99.polys) %in% levels(rent99$district)] = NULL

# run mrf and re models
gam_rent_mrf = gam(rent ~ s(district, bs = 'mrf', xt = list(polys=rent99.polys)),
                   data = rent99,
                   method = 'REML')
gam_rent_re = gam(rent ~ s(district, bs = 're'),
                  data = rent99,
                  method = 'REML')
```



Next we show the plot of the estimated random effects of both models on the map of districts[45]. Gaps appear because there isn't data for every district available, as some are districts without houses like parks, industrial areas, etc.



As we might have expected, there appears to be more color coordination with the MRF result (left), since neighbors are more likely to be similar. Meanwhile, the mixed model approach, while showing similar patterning, does nothing inherently to correlate one district with the ones it's next to, but may allow for more regularization.

Either of these models is appropriate, but they ask different questions. The MRF approach may produce better results since it takes into account the potential similarity among neighbors, but also may not be necessary if there isn't much similarity among geographical units. One should also think about whether the other covariates in the model may account spatial autocorrelation or not, or unspecified unit effects, and proceed accordingly. In addition, there are recent approaches that would allow for a mix of both the unstructured and spatial random effects. See Riebler et al. (2016)[46].

References

Breiman, Leo. 2001. "Statistical Modeling: The Two Cultures (with Comments and a Rejoinder by the Author)." *Statistical Science* 16 (3): 199–231. https://doi.org/10.1214/ss/1009213726.

Bybee, Rodger W., and Barry McCrae. 2009. *Pisa Science 2006: Implications for Science Teachers and Teaching*. NSTA Press.

Fahrmeir, Ludwig, Thomas Kneib, Stefan Lang, and Brian Marx. 2013. *Regression: Models, Methods and Applications*. Springer Science & Business Media.

Fox, John. 2000a. *Nonparametric Simple Regression: Smoothing Scatterplots*. SAGE.

———. 2000b. *Multiple and Generalized Nonparametric Regression*. SAGE.

Friedman, Jerome, Trevor Hastie, Robert Tibshirani, and others. 2000. "Additive Logistic Regression: A Statistical View of Boosting (with Discussion and a Rejoinder by the Authors)." *The Annals of Statistics* 28 (2). Institute of Mathematical Statistics: 337–407.

Hardin, James W., and Joseph M. Hilbe. 2012. *Generalized Linear Models and Extensions, Third Edition*. Stata Press.

Hastie, T.J., and R.J. Tibshirani. 1990. *Generalized Additive Models*. CRC Press.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. 2nd ed. 2009. Corr. 3rd printing 5th Printing. Springer.

Rasmussen, Carl Edward, and Christopher K. I Williams. 2006. *Gaussian Processes for Machine Learning*. Cambridge, Mass.: MIT Press.

Riebler, Andrea, Sigrunn H Sørbye, Daniel Simpson, and Håvard Rue. 2016. "An Intuitive Bayesian Spatial Model for Disease Mapping That Accounts for Scaling." *Statistical Methods in Medical Research* 25 (4). SAGE Publications Sage UK: London, England: 1145–65.

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). https://doi.org/10.1111/j.1467-9876.2005.00510.x.

Ruppert, David, Matt P. Wand, and Raymond J. Carroll. 2003. *Semiparametric Regression*. Cambridge University Press.

Shalizi, Cosma. 2016. *Advanced Data Analysis from an Elementary Point of View*.

Simpson, Gavin L. 2018. "Modelling Palaeoecological Time Series Using Generalised Additive Models." *Frontiers in Ecology and Evolution* 6. https://doi.org/10.3389/fevo.2018.00149.

Venables, William N., and Brian D. Ripley. 2002. *Modern Applied Statistics with S*. Birkhäuser.

Wasserman, Larry. 2006. *All of Nonparametric Statistics*. Springer.

Wood, Simon N. 2008. "Fast Stable Direct Fitting and Smoothness Selection for Generalized Additive Models." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 70 (3). Wiley Online Library: 495–518.

Wood, S. N. 2006. *Generalized Additive Models: An Introduction with R*. Vol. 66. CRC Press.

———. 2017. *Generalized Additive Models: An Introduction with R*. 2nd ed. CRC Press.

References

Simpson, Gavin L. 2018. "Modelling Palaeoecological Time Series Using Generalised Additive Models." *Frontiers in Ecology and Evolution* 6. https://doi.org/10.3389/fevo.2018.00149.

Riebler, Andrea, Sigrunn H Sørbye, Daniel Simpson, and Håvard Rue. 2016. "An Intuitive Bayesian Spatial Model for Disease Mapping That Accounts for Scaling." *Statistical Methods in Medical Research* 25 (4). SAGE Publications Sage UK: London, England: 1145–65.

38. You can verify this by running `model.matrix(ga_model)`.↩

39. Much like with Gaussian process regression, where it's perhaps a bit more explicit.↩

40. All the same variables in the lme output start with X. This is more to avoid confusion in the functions behind the scenes.↩

41. I don't show it to keep the plot clean, but the fitted values are essentially the same.↩

42. Because of course there is an R package just for New Zealand census data.↩

43. There are actually three different types of objects one could supply here, but unfortunately the one thing mgcv doesn't do is work with any spatial objects one would already have from the popular R packages used for spatial modeling and visualization. The fact that this list *is* actually a special class object is of no importance here, it is read simply as a standard list object.↩

44. If you look at the markdown document for this on GitHub you'll see the code for how to create this using an object from the maps packages rather than needing a shapefile.↩

45. Unfortunately, I have neither the data nor the desire to try and make this a pretty plot. It just uses the basic mgcv plot and an attempted trick (which may not be entirely accurate) to superimpose the mixed model results onto the MRF data.↩

46. Stan developer Mitzi Morris provided an example of the model in Stan to the Ann Arbor R Users Group, but it wasn't yet available at the time I last updated this document. See the A2RUG site for April 2018 in case it is posted there.↩