Chapter 9

DJM

6 March 2018

Chapter 9

- Here we introduce the concept of GAMs (**G** eneralized **A** dditive **M** odels)
- The basic idea is to imagine that the response is the sum of some functions of the predictors:

$$\mathbb{E}\left[Y_i \mid X_i = x_i\right] = \alpha + f_1(x_{i1}) + \cdots + f_p(x_{ip}).$$

- Note that OLS **is** a GAM (take $f_j(x_{ij}) = \beta_j x_{ij}$):

$$\mathbb{E}\left[Y_i \mid X_i = x_i\right] = \alpha + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}.$$

- The algorithm for fitting these things is called "backfitting":
  1. Center Y and X.
  2. Hold f_k for all $k \neq j$ fixed, and regress f_j on the partial residuals using your favorite smoother.
  3. Repeat for $1 \leq j \leq p$.
  4. Repeat steps 2 and 3 until the estimated functions "stop moving" (iterate)
  5. Return the results.

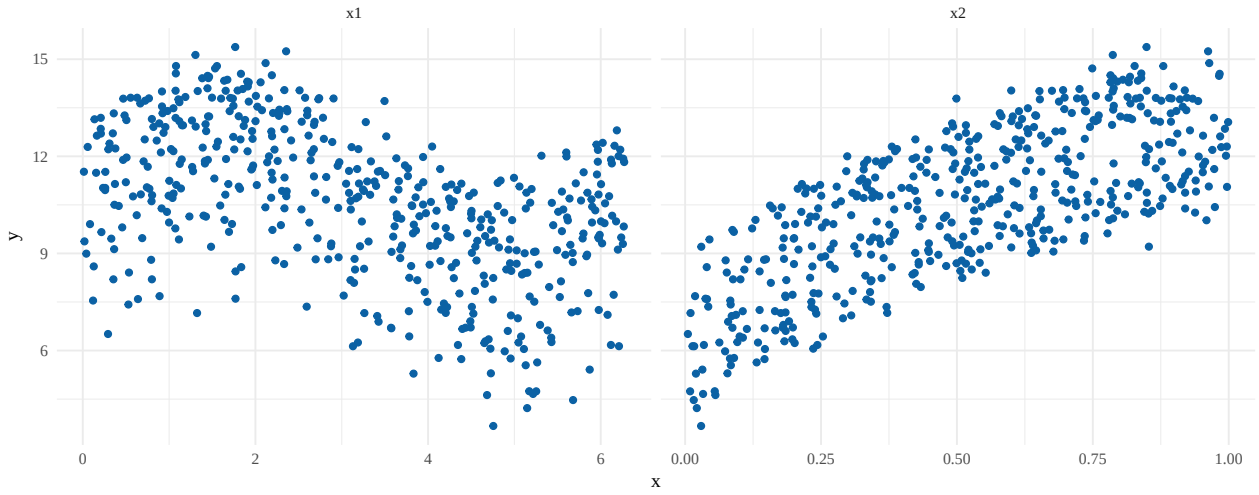
Results

- We will code it next time.
- There are two `R` packages that do this for us. I find `mgcv` easier.
- Let's look at a small example.

```
library(mgcv)
set.seed(03-06-2018)
n = 500
x1 = runif(n, 0, 2*pi)
x2 = runif(n)
y = 5 + 2*sin(x1) + 8*sqrt(x2)+rnorm(n,sd=.5)
df = data.frame(y=y,x1=x1,x2=x2)
```

Some plots

```
gather(df, predictor, x, -y) %>%
  ggplot(aes(x=x,y=y)) + geom_point(col=blue) +
  facet_wrap(~predictor,scales = 'free_x')
```



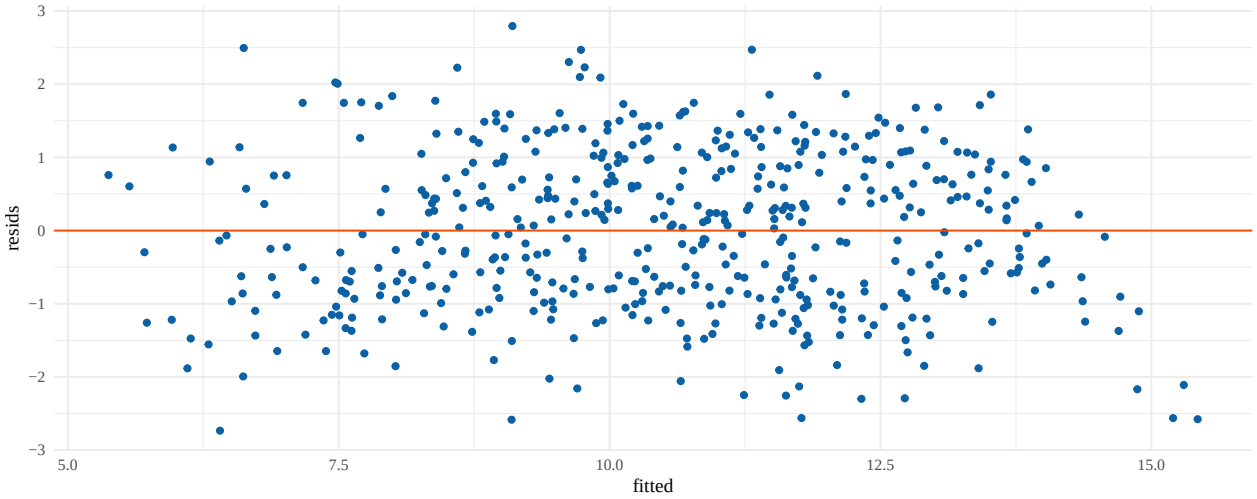
Small example

This just fits the linear model.

```
ex = gam(y~x1+x2, data=df)
summary(ex)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## y ~ x1 + x2
##
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.24940    0.13228   69.92   <2e-16
## x1          -0.63754    0.02659  -23.98   <2e-16
## x2           6.35533    0.17615   36.08   <2e-16
##
##
## R-sq.(adj) =  0.788   Deviance explained = 78.9%
## GCV = 1.1703  Scale est. = 1.1632    n = 500
```

```
ggplot(data.frame(fitted=fitted(ex),resids=residuals(ex)), aes(fitted,resids))+
  geom_point(color=blue) + geom_hline(yintercept = 0, color=red)
```

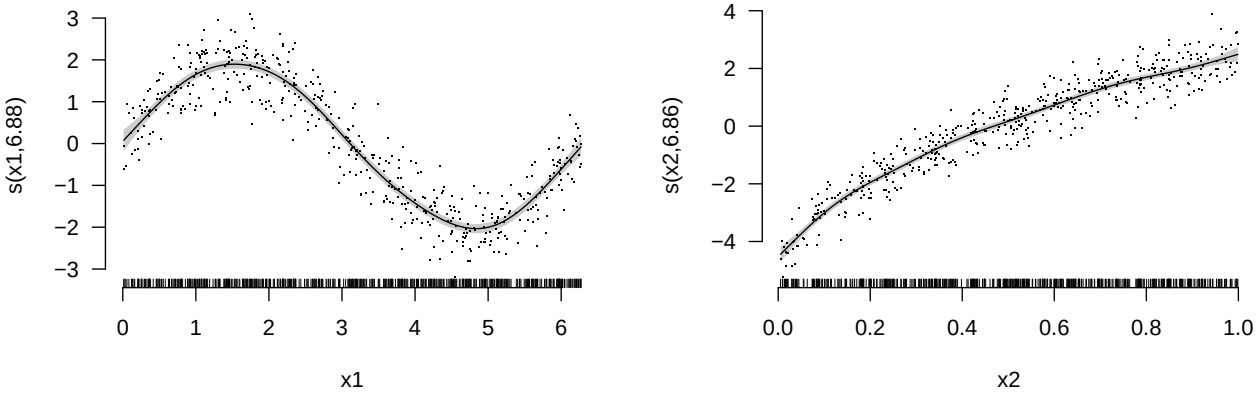


Smoothing

```
ex.smooth = gam(y~s(x1)+s(x2), data=df) # Smooths each coordinate independently
coefficients(ex.smooth) # still produces something
```

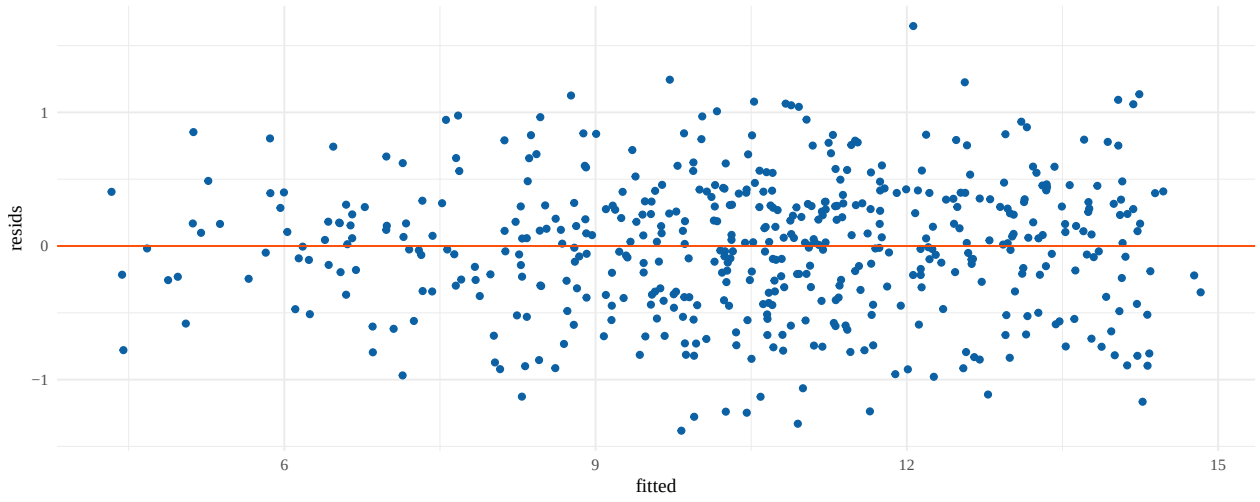
```
## (Intercept)     s(x1).1     s(x1).2     s(x1).3     s(x1).4     s(x1).5
##  10.5325141  -4.4644279  -0.3998284  -0.5041757   0.4559100  -0.2671849
##     s(x1).6     s(x1).7     s(x1).8     s(x1).9     s(x2).1     s(x2).2
##   0.2009503  -0.1013919  -0.5854346   3.4931675  -0.8903538   1.4849200
##     s(x2).3     s(x2).4     s(x2).5     s(x2).6     s(x2).7     s(x2).8
##   0.4656511  -0.8739181  -0.4677584  -0.9463167  -0.3104969   3.0361407
##     s(x2).9
##   2.8282590
```

```
plot(ex.smooth, pages = 1, scale=0, shade=TRUE, resid=TRUE, se=2, bty='n', las=1)
```



Residuals vs. fitted

```
smoothdf = data.frame(fitted=fitted(ex.smooth),resids=residuals(ex.smooth),
                      mdl='original')
ggplot(smoothdf, aes(fitted,resids))+
  geom_point(color=blue) + geom_hline(yintercept = 0, color=red)
```

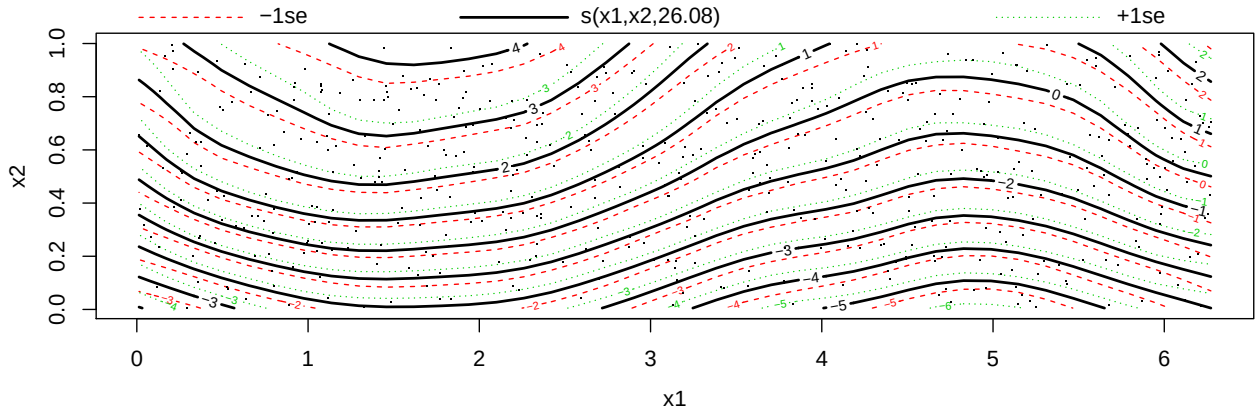


Another version

```
ex.toosmooth = gam(y~s(x1,x2), data=df) # smooths together (like npreg)
coefficients(ex.toosmooth) # still produces something
```

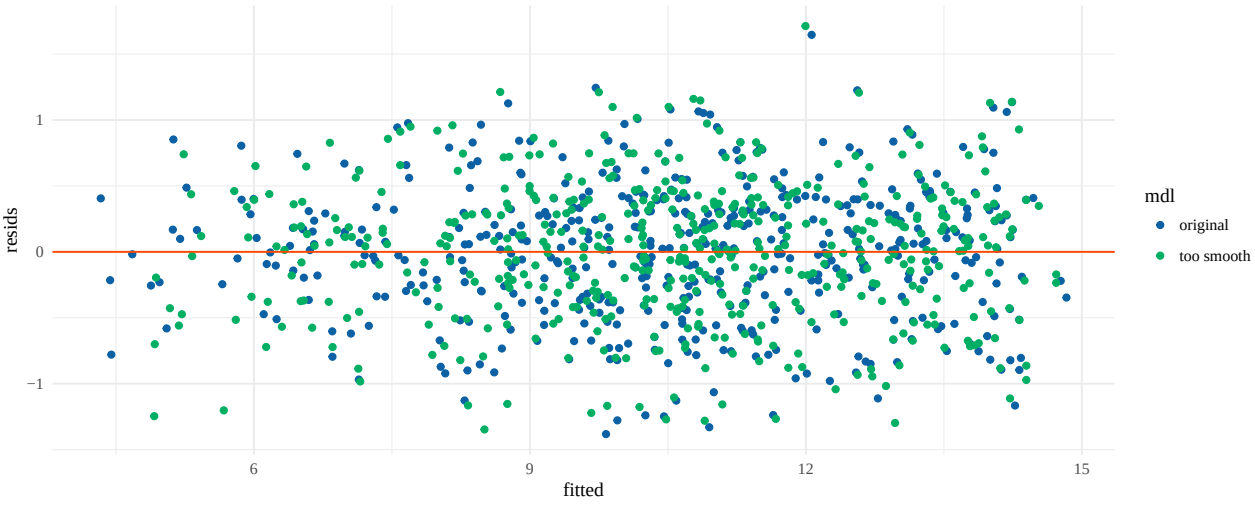
```
## (Intercept)  s(x1,x2).1  s(x1,x2).2  s(x1,x2).3  s(x1,x2).4  s(x1,x2).5
## 10.53251414 -1.96277970 -0.37782969 -0.10255843  0.29911906  0.04273830
##  s(x1,x2).6  s(x1,x2).7  s(x1,x2).8  s(x1,x2).9 s(x1,x2).10 s(x1,x2).11
## -0.16508674  0.03993086 -0.11299700  0.02532941  0.52141463 -0.15128974
## s(x1,x2).12 s(x1,x2).13 s(x1,x2).14 s(x1,x2).15 s(x1,x2).16 s(x1,x2).17
##  0.21147841 -0.19611471 -0.68032210  0.38504029 -0.25113776 -0.06274758
## s(x1,x2).18 s(x1,x2).19 s(x1,x2).20 s(x1,x2).21 s(x1,x2).22 s(x1,x2).23
## -0.08753100  0.30829041  0.25439911  0.17545400  0.18993961 -0.23722085
## s(x1,x2).24 s(x1,x2).25 s(x1,x2).26 s(x1,x2).27 s(x1,x2).28 s(x1,x2).29
## -0.01992779  0.12443188  0.07059786  1.43465881  1.30046867  1.79243618
```

```
plot(ex.toosmooth, pages = 1, scale=0, shade=TRUE, resid=TRUE, se=2, bty='n', las=1)
```



Residuals vs. fitted (not too different)

```
rbind(smoothdf,
      data.frame(fitted=fitted(ex.toosmooth),resids=residuals(ex.toosmooth),
                 mdl='too smooth')) %>%
  ggplot(aes(fitted,resids,color=mdl)) + geom_point() +
  scale_color_manual(values=c(blue,green)) +
  geom_hline(yintercept = 0, color=red)
```



Redoing the example in the text

```
housing <- read.csv("http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/data/calif_penn_2011.csv") # load the
housing <- na.omit(housing) # removes any row with an NA
calif <- filter(housing, STATEFP==6) # gets the california data
```

Linear model

```
calif.lm <- lm(log(Median_house_value) ~ Median_household_income
  + Mean_household_income + POPULATION + Total_units + Vacant_units + Owners
  + Median_rooms + Mean_household_size_owners + Mean_household_size_renters
  + LATITUDE + LONGITUDE, data = calif) # why this model and not another?
print(summary(calif.lm), signif.stars=FALSE, digits = 3) # less annoying output
```

```
##
## Call:
## lm(formula = log(Median_house_value) ~ Median_household_income +
##     Mean_household_income + POPULATION + Total_units + Vacant_units +
##     Owners + Median_rooms + Mean_household_size_owners + Mean_household_size_renters +
##     LATITUDE + LONGITUDE, data = calif)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.855 -0.153  0.034  0.189  1.214
```

```
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  -5.74e+00   5.28e-01  -10.86  < 2e-16
## Median_household_income       1.34e-06   4.63e-07    2.90   0.0038
## Mean_household_income         1.07e-05   3.88e-07   27.71  < 2e-16
## POPULATION                   -4.15e-05   5.03e-06   -8.27  < 2e-16
## Total_units                   8.37e-05   1.55e-05    5.41  6.4e-08
## Vacant_units                  8.37e-07   2.37e-05    0.04   0.9719
## Owners                       -3.98e-03   3.21e-04  -12.41  < 2e-16
## Median_rooms                 -1.62e-02   8.37e-03   -1.94   0.0525
## Mean_household_size_owners    5.60e-02   7.16e-03    7.83  5.8e-15
## Mean_household_size_renters -7.47e-02   6.38e-03  -11.71  < 2e-16
## LATITUDE                     -2.14e-01   5.66e-03  -37.76  < 2e-16
## LONGITUDE                    -2.15e-01   5.94e-03  -36.15  < 2e-16
##
## Residual standard error: 0.317 on 7469 degrees of freedom
## Multiple R-squared:  0.639,  Adjusted R-squared:  0.638
## F-statistic: 1.2e+03 on 11 and 7469 DF,  p-value: <2e-16
```

Some model evaluation

Note: Some differences from text to demonstrate `tidyverse`

```
round(sqrt(mean(residuals(calif.lm)^2)),3) # how big are our errors (on log scale)
```

```
## [1] 0.317
```

```
round(exp(sqrt(mean(residuals(calif.lm)^2)))-1,3) # on the actual scale
```

```
## [1] 0.373
```

```
preds.lm.all = predict(calif.lm, se.fit=TRUE, interval = 'prediction')
# The `preds.lm.all$fit` object contains lwr and upr limits, but won't for gam
# to match, we recalculate
preds.lm = data.frame(
  obs.value = calif$Median_house_value,
  fit = preds.lm.all$fit[,1],
  fit.se = preds.lm.all$se.fit)
sigma.lm = summary(calif.lm)$sigma
preds.lm = preds.lm %>% mutate(
  lwr = fit - 2*sqrt(fit.se^2 + sigma.lm^2), # remember this formula???
  upr = fit + 2*sqrt(fit.se^2 + sigma.lm^2), # remember this formula???
  captured = (log(obs.value) <= upr) & (log(obs.value) >= lwr)
  )
mean(preds.lm$captured)# percentage of actual observations inside the CI
```

```
## [1] 0.9632402
```

```
round(median(preds.lm.all$se.fit),3) # median size of the pred SEs (log scale)
```

```
## [1] 0.011
```

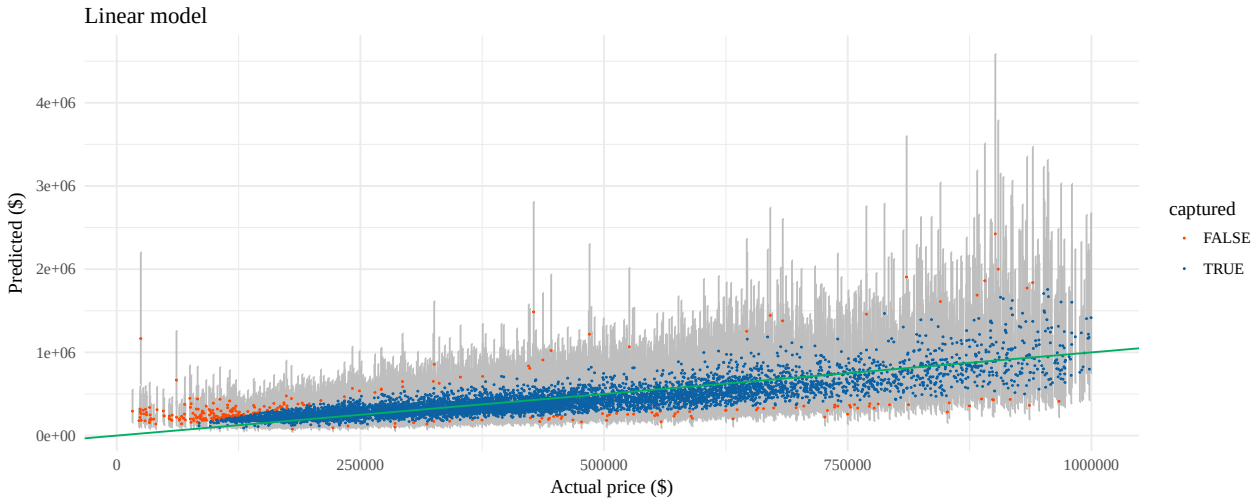
```
round(exp(median(preds.lm.all$se.fit))-1,3) # percent in $
```

```
## [1] 0.011
```

Plot our predictions

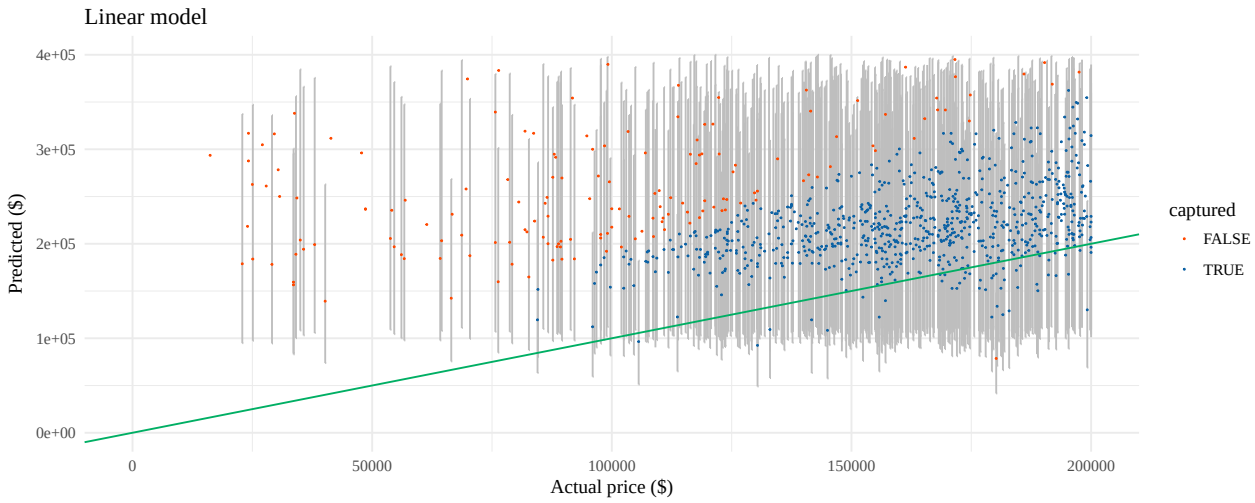
This part is modified to use `ggplot`. See the text for an alternative.

```
plm <- preds.lm %>%
  ggplot(aes(x=obs.value,y=exp(fit),color=captured)) +
  geom_errorbar(aes(ymin=exp(lwr), ymax=exp(upr)), width=.1,color='grey') +
  geom_point(size=.1) + geom_abline(slope=1, intercept = 0, color=green) +
  scale_color_manual(values=c(red,blue)) +
  xlab('Actual price ($)') + ylab('Predicted ($)') + ggtitle('Linear model')
plm
```



Zoom in on the bad part

```
plm + xlim(0,2e5) + ylim(0,4e5)
```



The GAM

```
calif.gam <- gam(log(Median_house_value)
  ~ s(Median_household_income) + s(Mean_household_income) + s(POPULATION)
  + s(Total_units) + s(Vacant_units) + s(Owners) + s(Median_rooms)
  + s(Mean_household_size_owners) + s(Mean_household_size_renters)
  + s(LATITUDE) + s(LONGITUDE), data=calif) # just put 's( )' around everything
round(sqrt(mean(residuals(calif.gam)^2)),3) # how big are our errors (on log scale),
```

```
## [1] 0.266
```

```
  # doing better (in sample) than before
round(exp(sqrt(mean(residuals(calif.gam)^2)))-1,3) # on the actual scale
```

```
## [1] 0.304
```

```
preds.gam.all = predict(calif.gam, se.fit=TRUE) # no interval here
preds.gam = data.frame(
  obs.value = calif$Median_house_value,
  fit = preds.gam.all$fit,
  fit.se = preds.gam.all$se.fit)
sigma.gam = sqrt(calif.gam$sig2) # annoyingly in a different place
preds.gam = preds.gam %>% mutate(
  lwr = fit - 2*sqrt(fit.se^2 + sigma.gam^2),
  upr = fit + 2*sqrt(fit.se^2 + sigma.gam^2),
  captured = (log(obs.value) <= upr) & (log(obs.value) >= lwr)
  )
mean(preds.gam$captured)# percentage of actual observations inside the CI
```

```
## [1] 0.9612351
```

```
round(median(preds.gam.all$se.fit),3) # median size of the pred SEs (log scale)
```

```
## [1] 0.02
```

```
round(exp(median(preds.gam.all$se.fit))-1,3) # percent in $, not as precise as before,
```

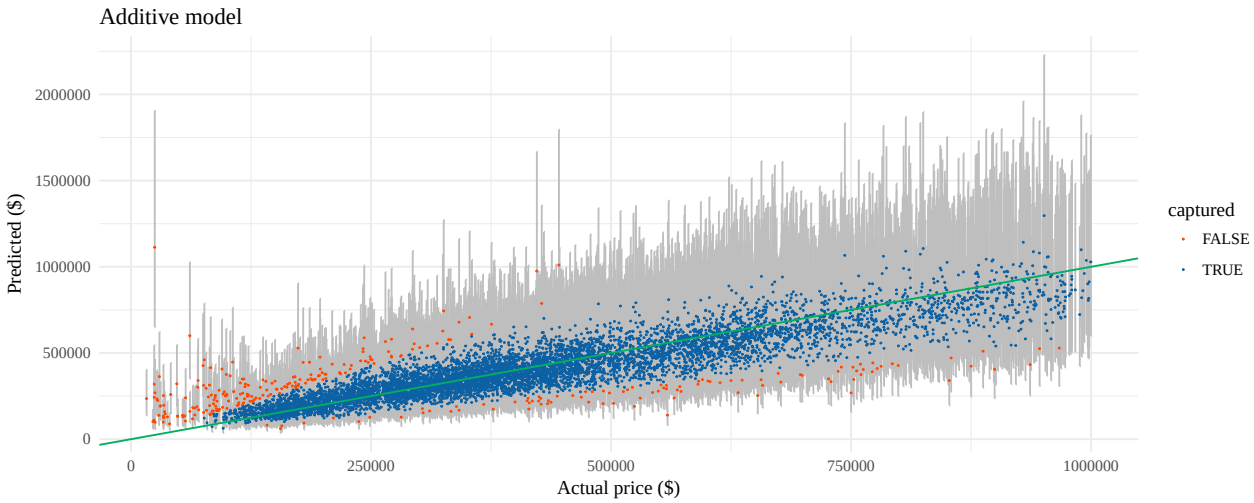
```
## [1] 0.021
```

```
  # recognizing uncertainty
```

Evaluating

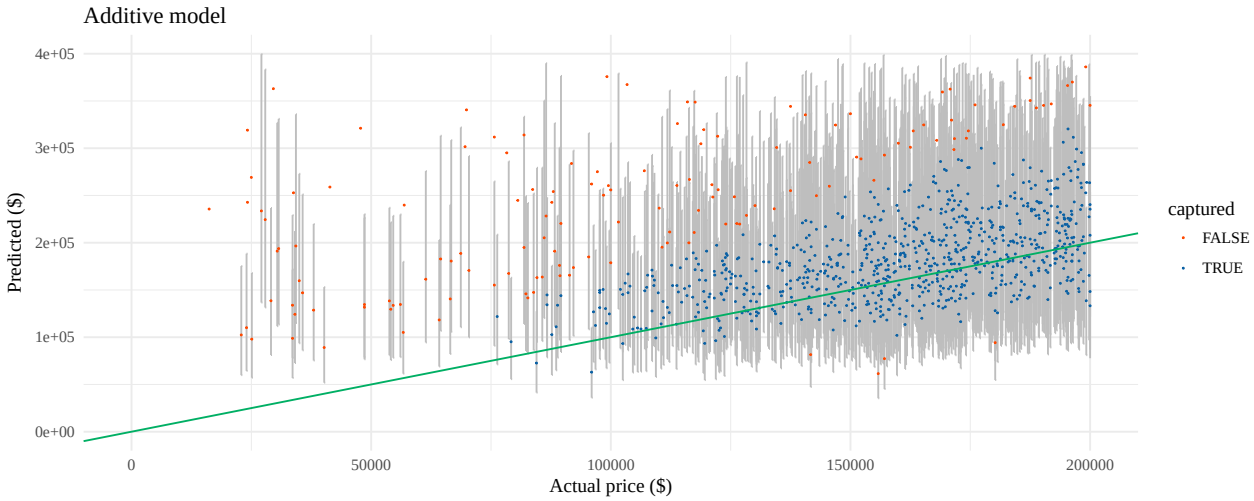
Our plot again, but for the gam

```
pgam <- preds.gam %>%
  ggplot(aes(x=obs.value,y=exp(fit),color=captured)) +
  geom_errorbar(aes(ymin=exp(lwr), ymax=exp(upr)), width=.1,color='grey') +
  geom_point(size=.1) + geom_abline(slope=1, intercept = 0, color=green) +
  scale_color_manual(values=c(red,blue)) +
  xlab('Actual price ($)') + ylab('Predicted ($)') + ggtitle('Additive model')
pgam
```



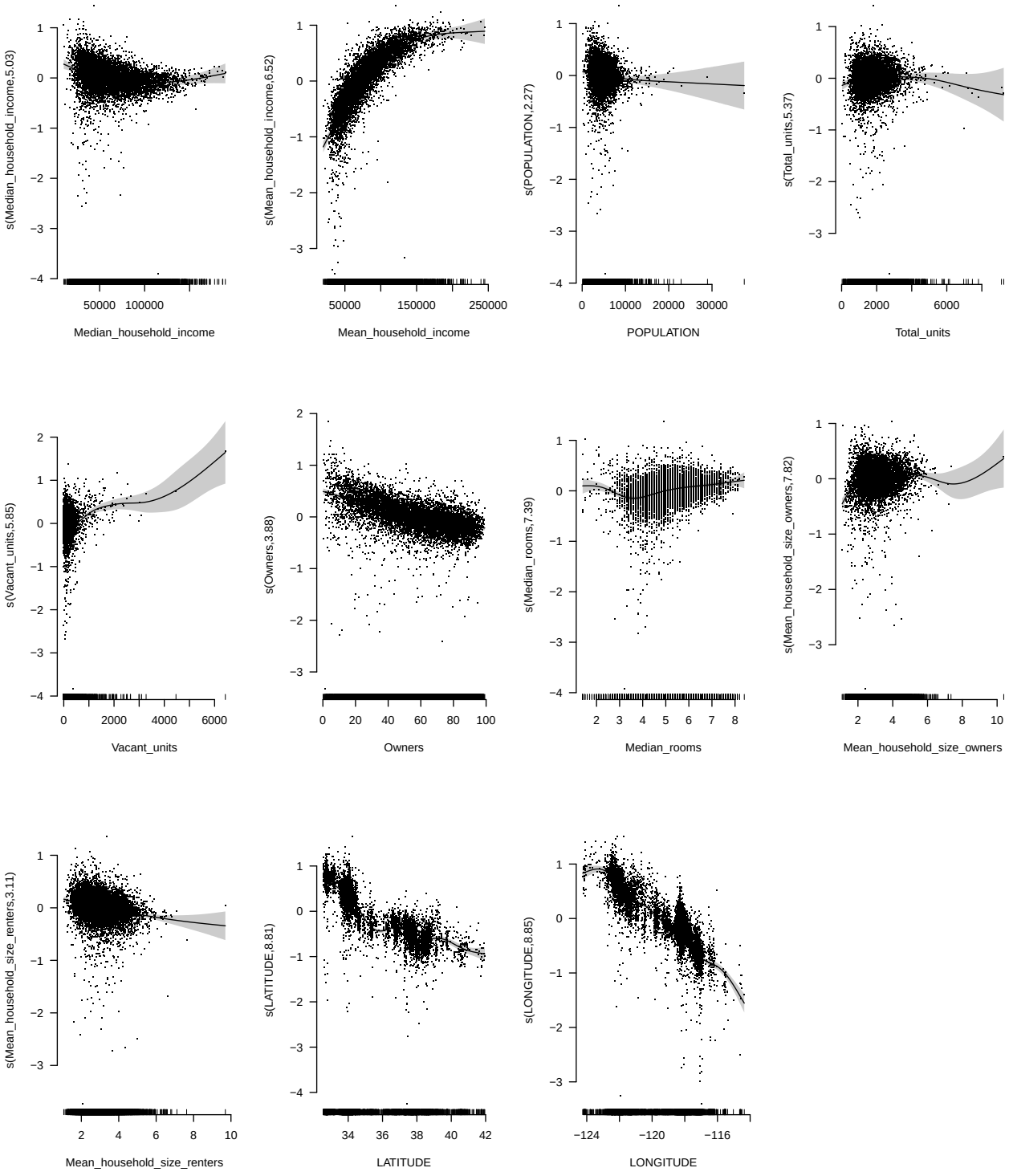
Zooming...

```
pgam + xlim(0,2e5) + ylim(0,4e5)
```



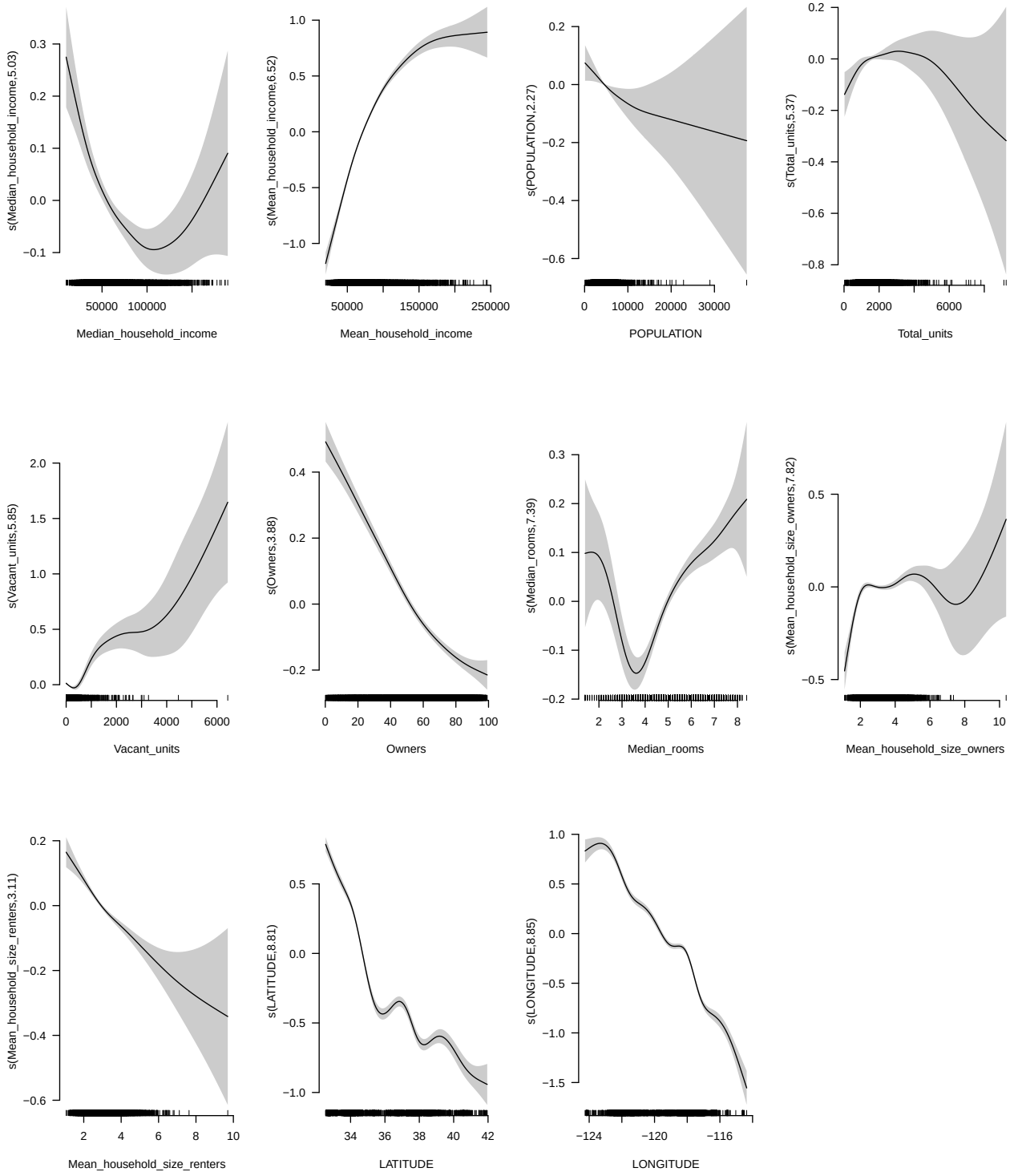
Partial response functions

```
plot(calif.gam, pages = 1, scale=0, shade=TRUE, resid=TRUE, se=2, bty='n', las=1)
```



Partial response functions, take 2

```
plot(calif.gam, pages = 1, scale=0, shade=TRUE, se=2, bty='n', las=1)
```



Do it again, a bit differently

```
calif.gam2 <- gam(log(Median_house_value)
  ~ s(Median_household_income) + s(Mean_household_income) + s(POPULATION)
  + s(Total_units) + s(Vacant_units) + s(Owners) + s(Median_rooms)
  + s(Mean_household_size_owners) + s(Mean_household_size_renters)
  + s(LONGITUDE,LATITUDE), data=calif)
  # does fully nonparametric regression on Long and Lat

round(exp(sqrt(mean(residuals(calif.gam2)^2)))-1,3) # on the actual scale
```

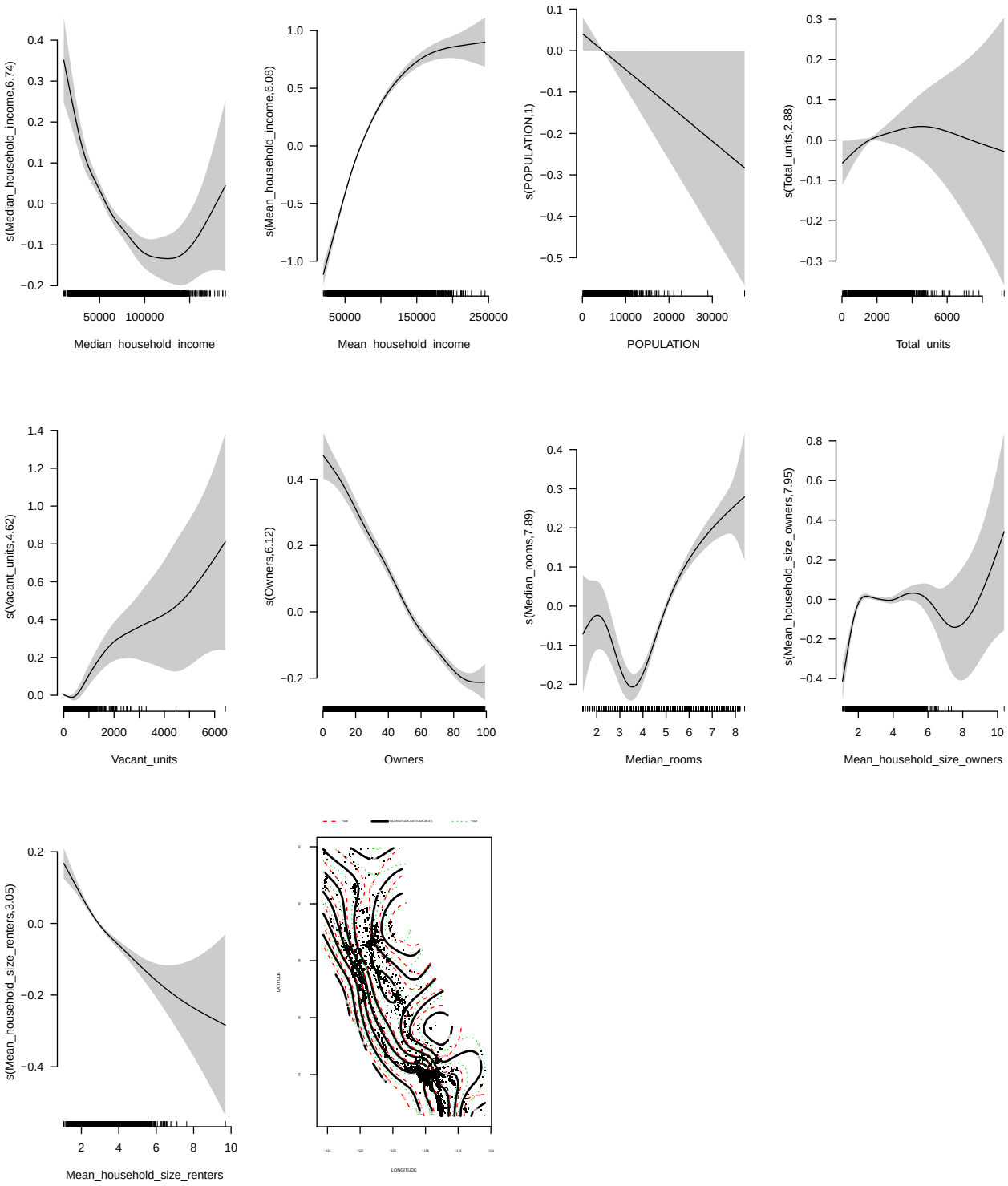
```
## [1] 0.286
```

```
preds.gam.all2 = predict(calif.gam, se.fit=TRUE) # no interval here
preds.gam2 = data.frame(
  obs.value = calif$Median_house_value,
  fit = preds.gam.all2$fit,
  fit.se = preds.gam.all2$se.fit)
sigma.gam2 = sqrt(calif.gam2$sig2) # annoyingly in a different place
preds.gam2 = preds.gam2 %>% mutate(
  lwr = fit - 2*sqrt(fit.se^2 + sigma.gam2^2),
  upr = fit + 2*sqrt(fit.se^2 + sigma.gam2^2),
  captured = (log(obs.value) <= upr) & (log(obs.value) >= lwr)
  )
mean(preds.gam2$captured)# percentage of actual observations inside the CI
```

```
## [1] 0.9573586
```

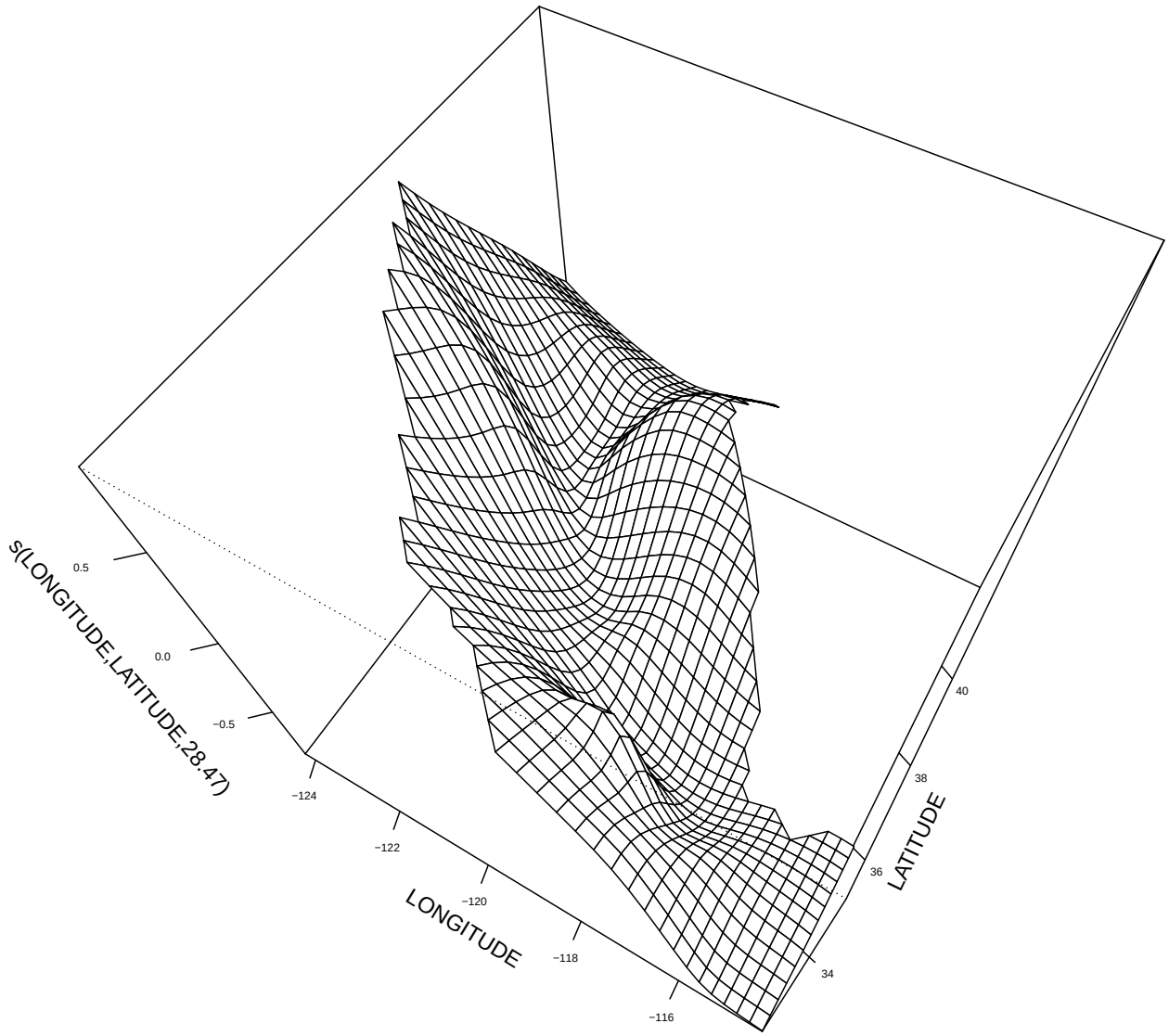
Another partial response plot

```
plot(calif.gam2, pages = 1, scale=0, shade=TRUE, se=2, bty='n', las=1)
```



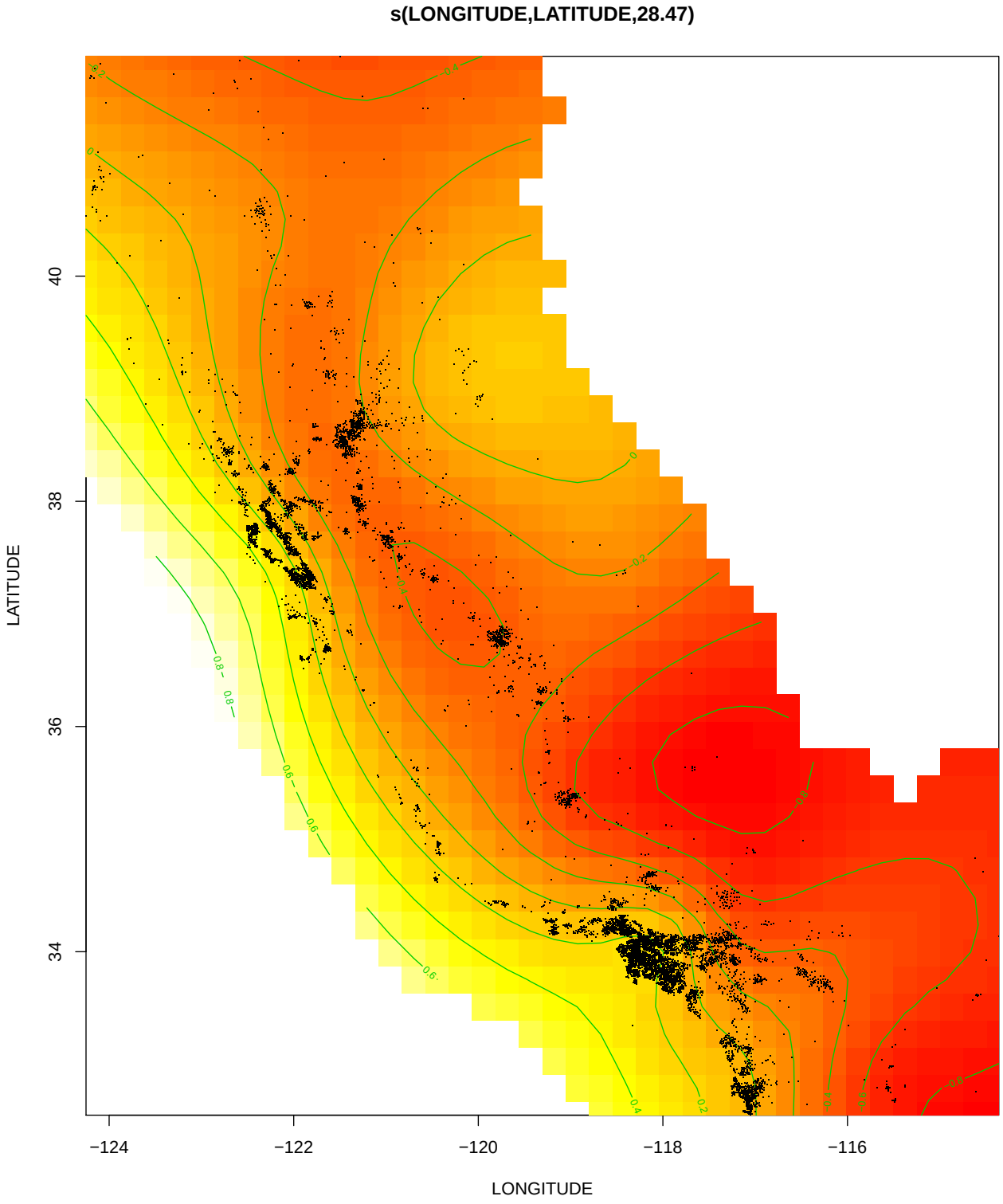
Zoom in on the interaction of Lat. and Long.

```
plot(calif.gam2,select=10,phi=60,pers=TRUE,ticktype="detailed",cex.axis=0.5)
```



A different version

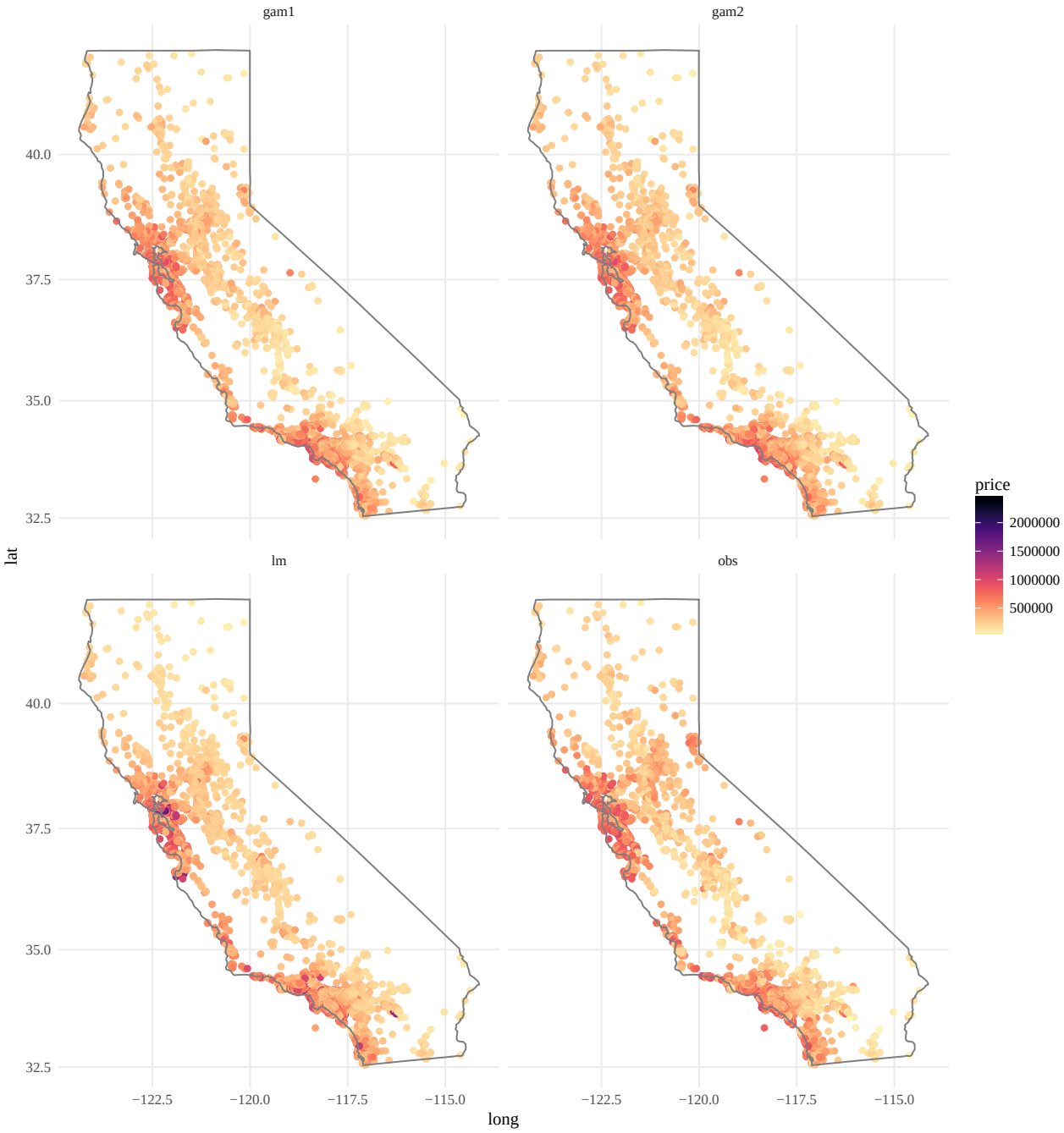
```
plot(calif.gam2,select=10,scheme=2,se=FALSE)
```



Drawing maps with colored points

This is **much** different from the text (easier)

```r
library(viridis) # good color scales
dfpreds = data.frame(obs = calif$Median_house_value,
                     lm = exp(preds.lm$fit),
                     gam1 = exp(preds.gam$fit),
                     gam2 = exp(preds.gam2$fit),
                     long = calif$LONGITUDE,
                     lat = calif$LATITUDE)
dflong = dfpreds %>% gather('model','price',-c(long,lat))
ggplot(dflong, aes(x=long,y=lat,color=price)) + geom_point() +
  facet_wrap(~model,nrow = 2) + coord_map() +
  borders('state','california') +
  scale_color_viridis(option = 'magma', direction=-1)
```

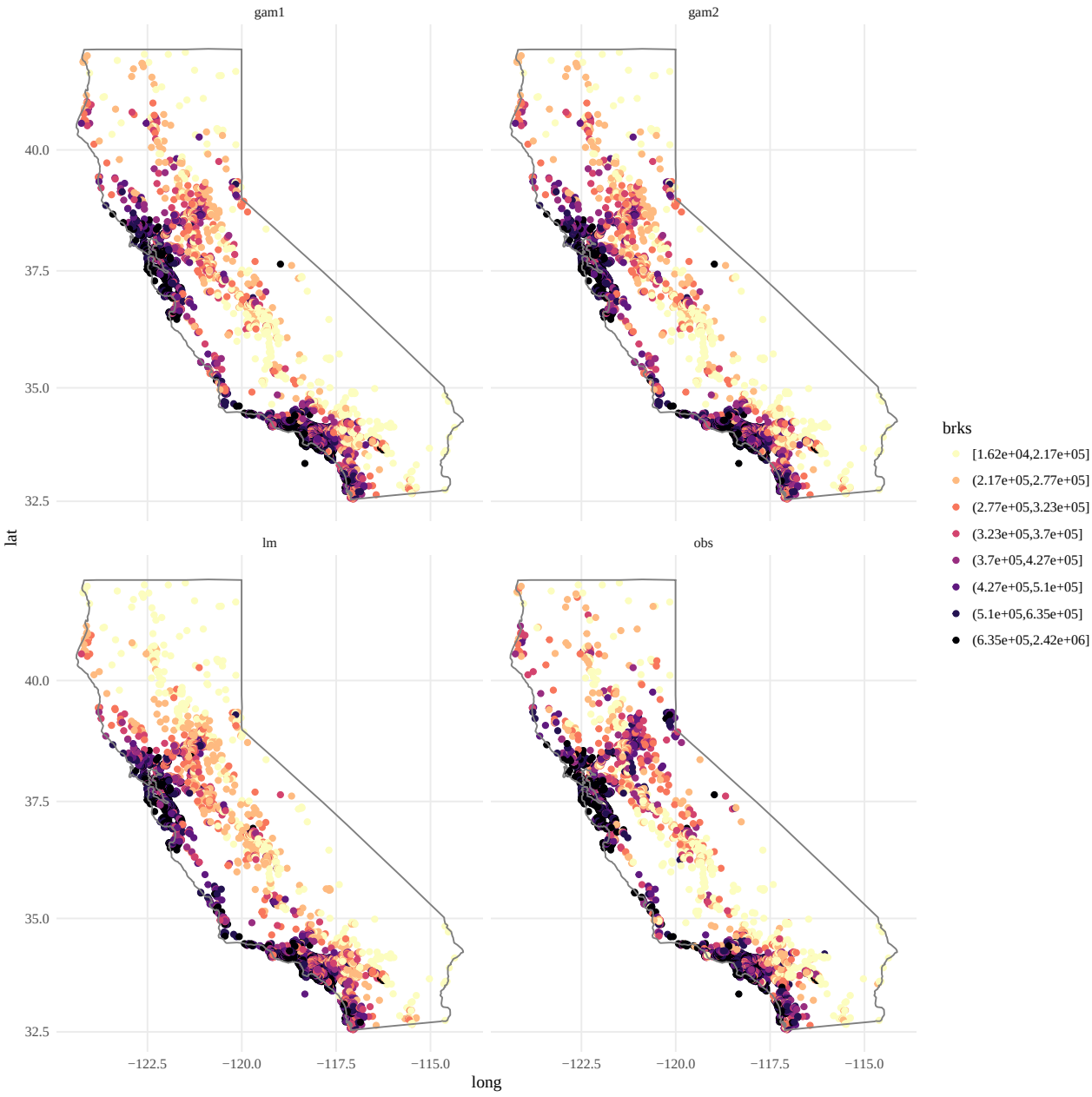


These look terrible. The scaling isn't very good.

Some modifications

```
nclasses = 8
dflong = dflong %>% mutate(brks = cut_number(price, nclasses))
ggplot(dflong, aes(x=long,y=lat,color=brks)) + geom_point() +
  facet_wrap(~model,nrow = 2) + coord_map() +
  borders('state','california') +
```

```
  scale_color_viridis(discrete=TRUE, option = 'magma', direction=-1)
```

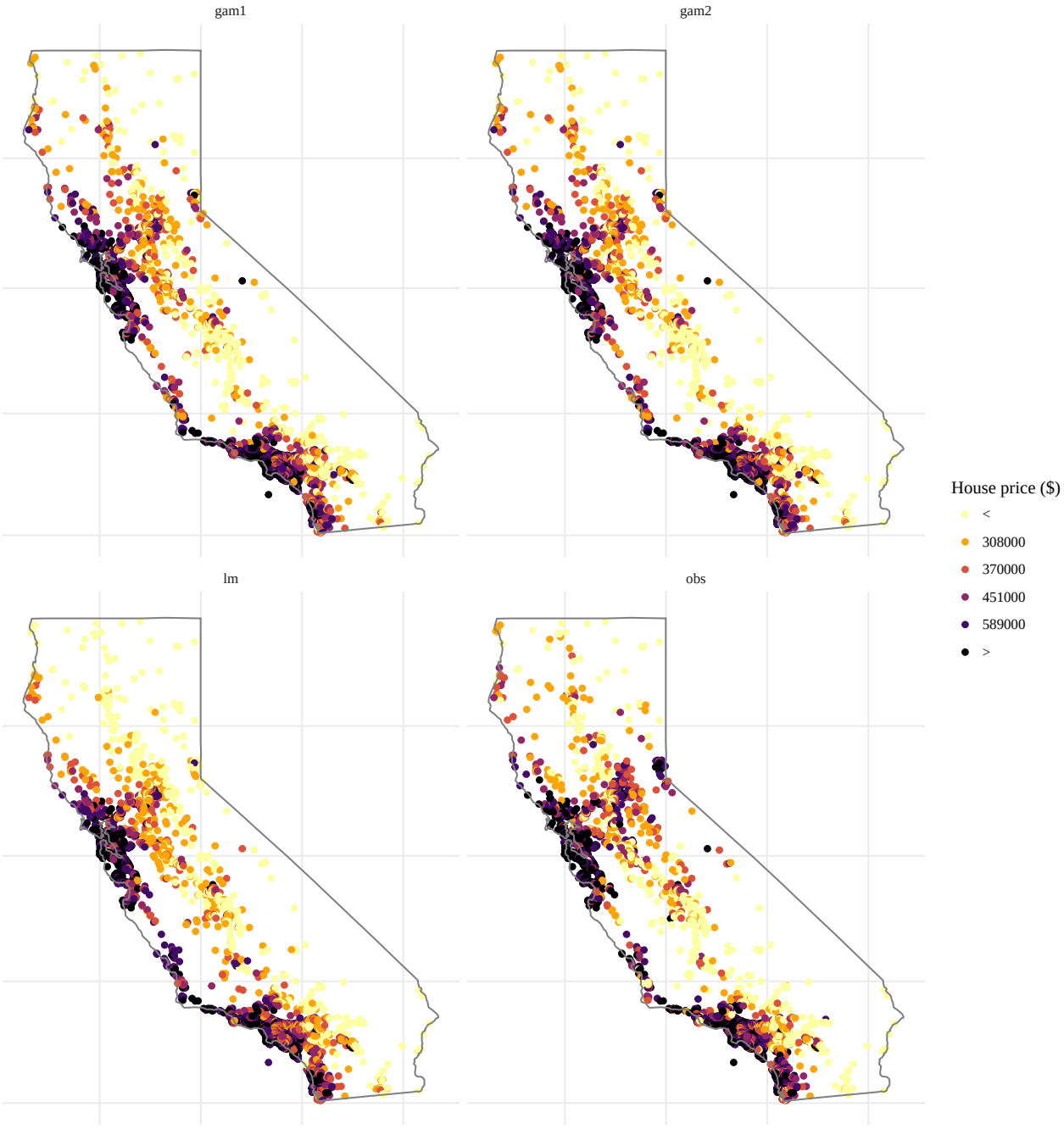


Much better, but the legend is ugly. And probably too many classes.

Final version

```
nclasses = 6
dflong = dflong %>% mutate(brks = cut_number(price, nclasses))
# A trick I found in ?cut
labs = levels(dflong$brks)
the.nums = cbind(lower = as.numeric( sub("\\((.+),.*", "\\1", labs) ),
```

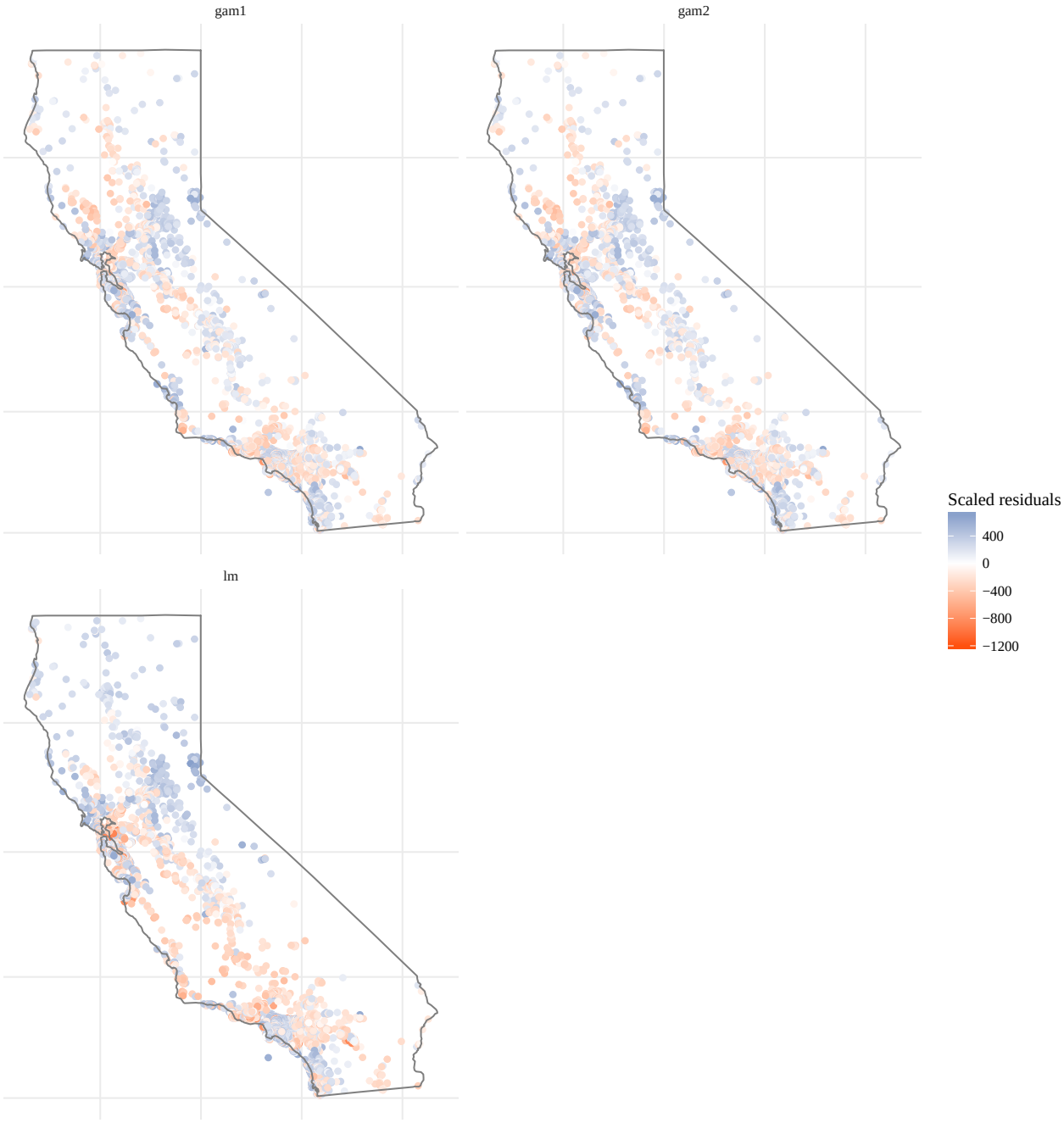
```
      upper = as.numeric( sub("[^,]*,([^]]*)\\]", "\\1", labs) ))
# There's a warning here for the lowest level, we ignore it
new.labs = c('<', the.nums[-c(1,nclasses), 2], '>')
levels(dflong$brks) = new.labs
ggplot(dflong, aes(x=long,y=lat,color=brks)) + geom_point() +
  facet_wrap(~model,nrow = 2) + coord_map() +
  borders('state','california') +
  scale_color_viridis(discrete=TRUE, option = 'inferno', direction=-1,
                      guide = guide_legend(title='House price ($)')) +
  theme( # kill off annoying annotation
    axis.line = element_blank(),
    axis.text.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks = element_blank(),
    axis.title.x = element_blank(),
    axis.title.y = element_blank())
```



Maps of errors

```
dferrs = dfpreds %>% mutate(lm = obs-lm,
                            gam1 = obs-gam1,
                            gam2 = obs-gam2)
dflong2 = dferrs %>% gather('model','resids',-c(long,lat,obs))
ggplot(dflong2, aes(x=long, y=lat,
                    color=sign(resids)*(abs(resids))^(1/2))) +
  # Makes it easier to see the color. Not meaningful
  geom_point() +
```

```
  facet_wrap(~model,nrow = 2) + coord_map() +
  borders('state','california') +
  scale_color_gradient2(midpoint=0, mid='white', low=red, high=blue,
                        guide=guide_colorbar(title='Scaled residuals')) +
  theme( # kill off annoying annotation
    axis.line = element_blank(),
    axis.text.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks = element_blank(),
    axis.title.x = element_blank(),
    axis.title.y = element_blank())
```


gam1
gam2
lm
Scaled residuals
400
0
−400
−800
−1200