# StatR 502 Homework 6

*Rebecca Hadi*

*Due Thursday, Feb 15, 2018, 6:30 pm*

Submission guidelines: please submit a knitted PDF or Word document, and optionally your `.Rmd` file. As always, ask in the discussion forum if you're having trouble!

## 1. Smoothing one predictor with Shiny

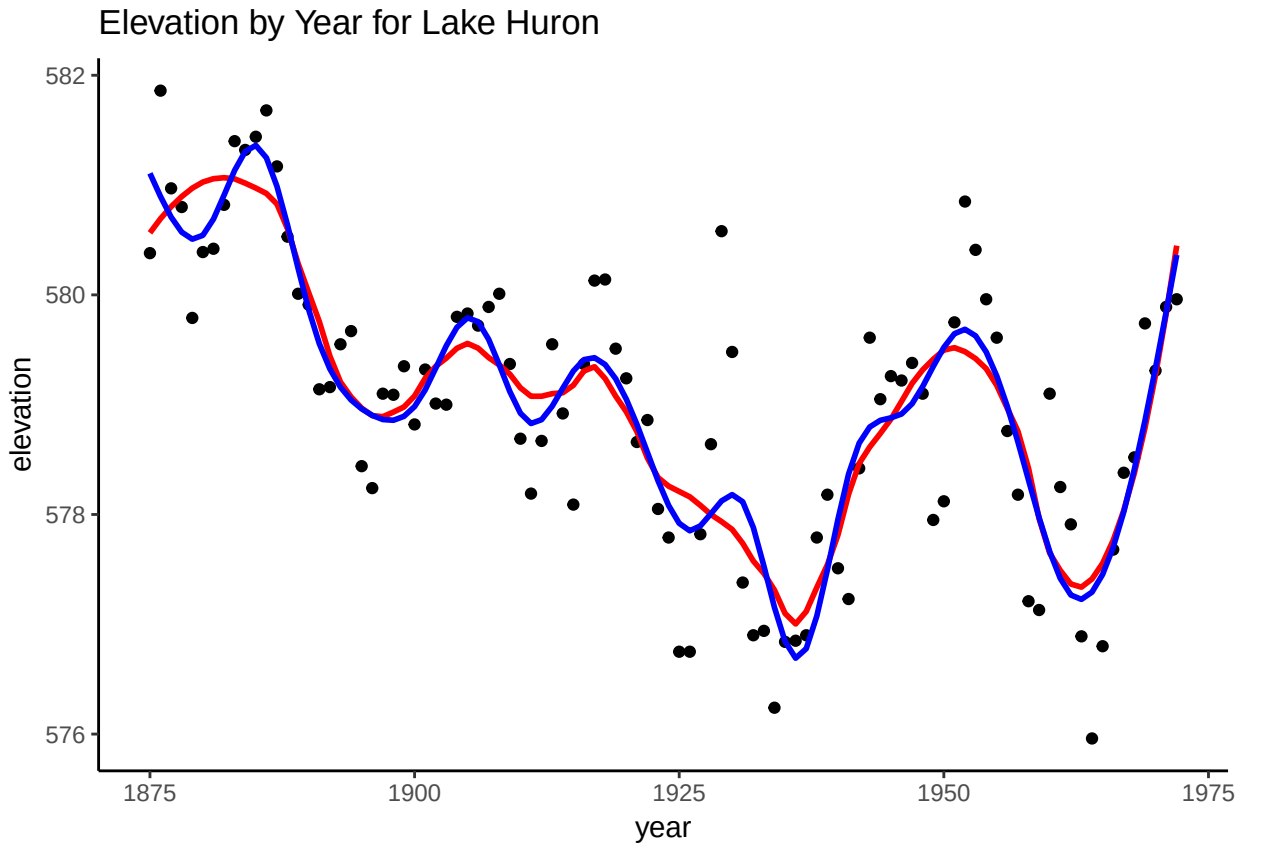
**(a)** Using the `LakeHuron` data in the `datasets` package (it should be loaded by default), construct a data frame with two columns, one of lake elevation and one for year. You will want to coerce the the `LakeHuron` data to class `numeric`, the built-in data is a special `ts` time-series class.

Produce a single plot that shows the raw data along with two smoothed lines: one fit by Loess, the other fit with cubic regression splines (`s(..., bs = "cr")` in the `mgcv` package). Make sure both smooths are "tuned" to capture the important-seeming features of the data.

(*Note:* it is possible to do this by fitting the models directly and using the `predict()` method, or by using `geom_smooth` to do the fitting. Either method is fine.)

```
library(datasets)
library(zoo)
library(dplyr)
library(ggplot2)
library(mgcv)

#Import data
lakehuron <- as.data.frame(datasets::LakeHuron)

#extract columns
lakehuron$year <- as.yearmon(time(lakehuron$x)) %>%
                  as.numeric(format(., "%Y"))

lakehuron$x <- as.numeric(lakehuron$x)

colnames(lakehuron) <- c("elevation","year")

#Create base plot
lake.plot <-  ggplot(data = lakehuron, aes(x = year, y = elevation)) +
              geom_point()


#Loess
lakehuron.lo <- loess(elevation ~ year, data = lakehuron, span = .25)
lakehuron$pred.loess <- predict(lakehuron.lo)


#Spline (w 20 knots)
lake.spl <- gam(elevation ~ s(year, bs = "cr", k = 20), data = lakehuron)
lakehuron$pred.spl <- predict(lake.spl)
```

```r
#Combine both lines into one chart
lake.plot %+%
        geom_line(aes(y = pred.loess), size = 1, col = "red", data = lakehuron) %+%  #Loess
        geom_line(aes(y = pred.spl), size = 1, col = "blue", data = lakehuron) + #Spline
        ggtitle("Elevation by Year for Lake Huron") +
        theme_classic()
```




**(b)** In a new R script, create an interactive version of your plot from (a) with Shiny. Your Shiny app should have two sliders contained inside a single sidebarPanel: the first slider controls the `span` parameter of the Loess fit, and the second controls the `k` parameter of the smoother `s()`. Set the `min` and `max` arguments of the sliders to values you think are appropriate; the results should be able to get anywhere between a model that resembles an almost linear fit, to a model that almost resembles the data itself.

**(c)** Save your R script from (b) anonymously to Gist at https://gist.github.com/. It is important that the filename in Gist is "App.R".

**(d)** Test your anonymous Gist Shiny app with the command `runGist("gist_number")`, where `"gist_number"` is the unique identifier of your Gist code. You can see what your end result should resemble by running this command:

```r
shiny::runGist("e0bd1c8a922fe75450c298762594c638")
```

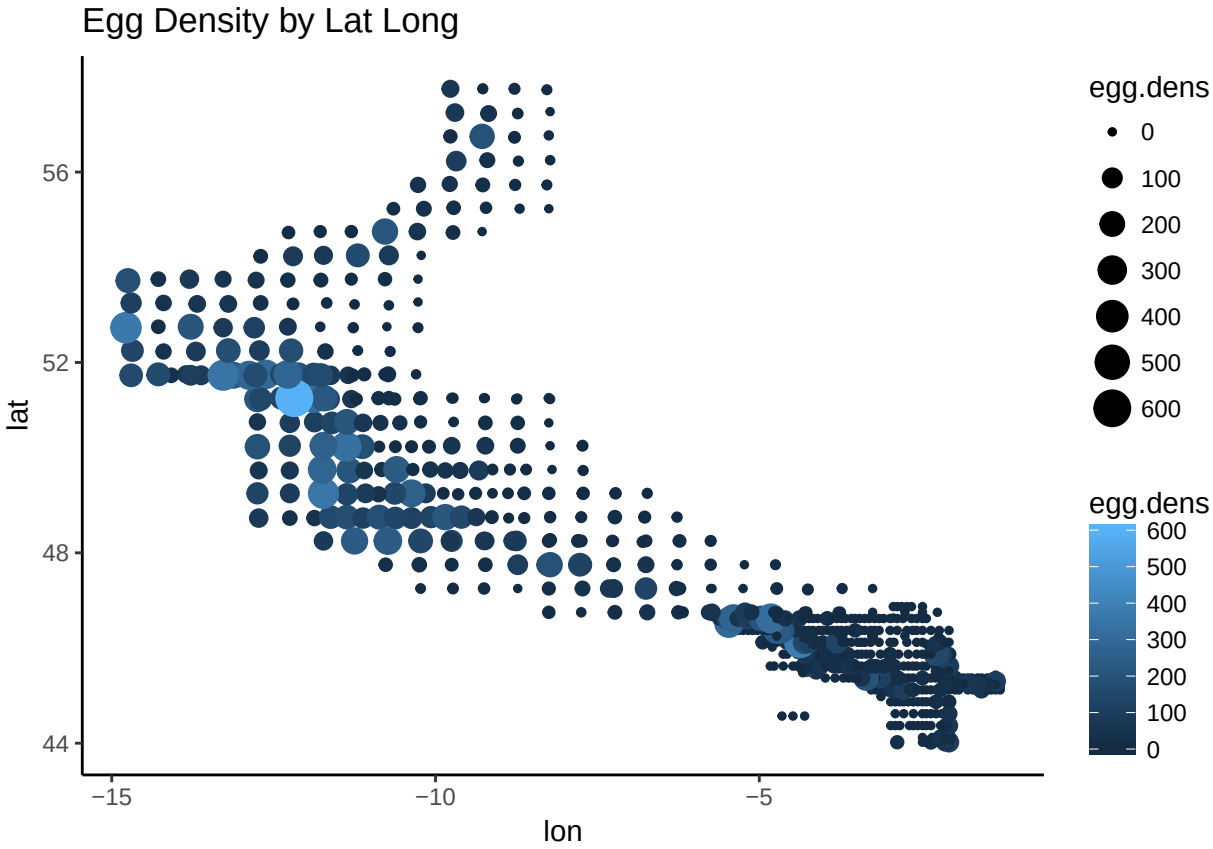
## 2. A real GAM

Use the `mack` data in the `gamair` package. The response variable, `egg.count`, is the count of mackerel eggs in net that has been pulled up to the sea surface from an adequate depth. Many samples were taken, and the latitude/longitude are recorded for each observation, along with quite a few other variables. See

`?gamair::mack` for more detail. These data would be of interest for stock assessment of mackerel.

**(a)** Plot the egg density, `egg.dens`, using longitude/latitude as x/y, and a different aesthetic for the density. Bonus points for using `ggmap` to make a nice map of the data! (See below for optional addition.)

```
#load package
library(gamair)

#load data
data(mack)

#Create plot
ggplot(data = mack, aes(x = lon, y = lat, col = egg.dens)) +
       geom_point(data = mack, aes(size = egg.dens)) +
       theme_classic() +
       ggtitle("Egg Density by Lat Long")
```



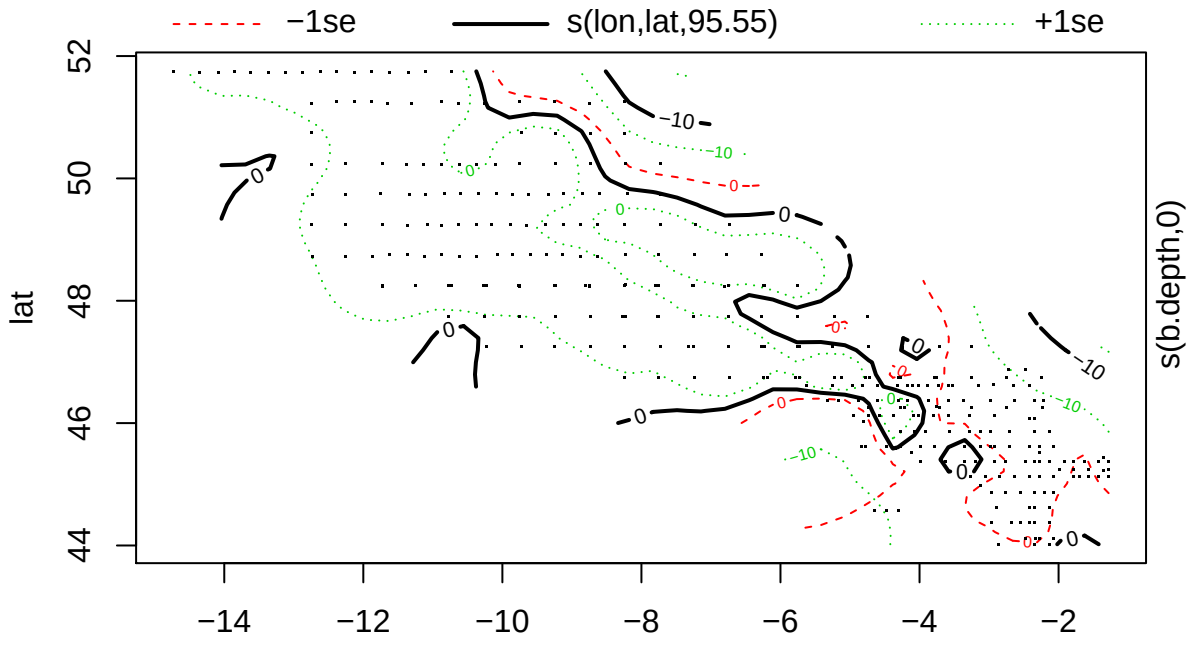

**(b)** Despite using `egg.dens` for the plot, for our model we will `egg.count` as the response. Since it's a count, we'll use a Poisson-family GAM. Much like the doctor complaint data where we used an offset term because we expected a one-to-one relationship between the number of complaints and the number of hours worked, here we would expect a one-to-one relationship between the number of eggs and the size (cross-sectional area) of net used. Thus, add `+ offset(log(net.area))` to your model formula.

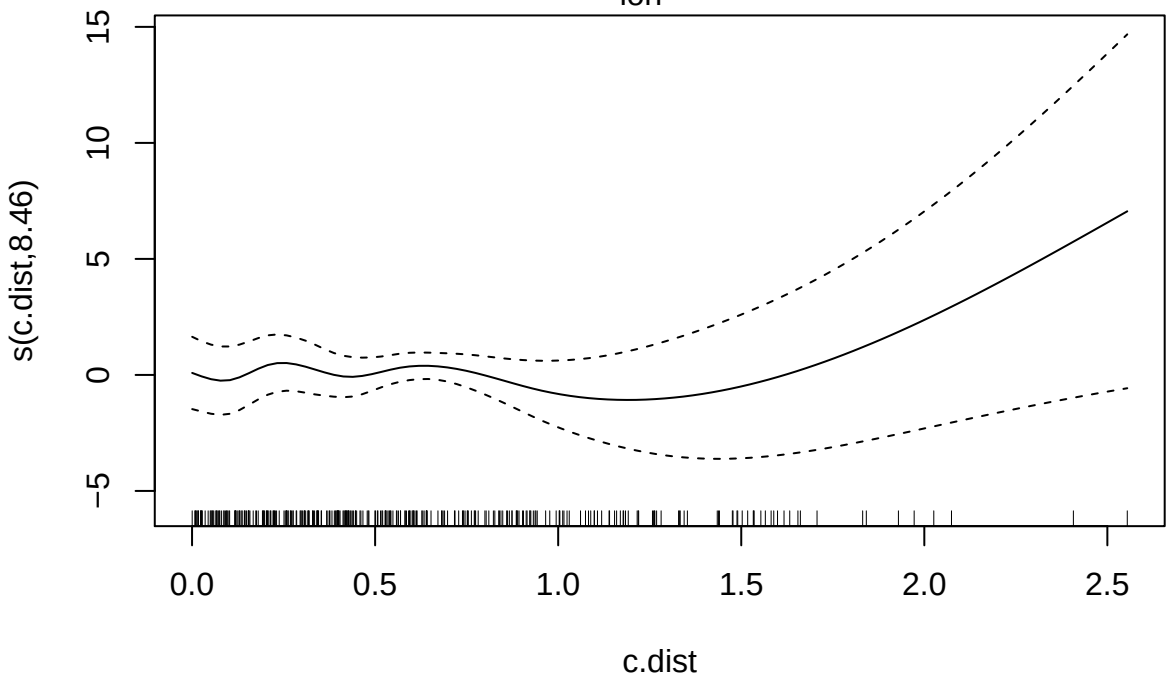
- Use a two-dimensional smooth of longitude and latitude (`s(lon, lat)`) and one-dimensional smooths of depth, distance to the continental shelf, salinity, surface temperature, temperature at 20 m depth, and time.
- *For the 1-d smooths* the default `k` value is probably fine; for the 2-d smooth you should experiment with several different `k` values, assessing the resulting plot. However, don't go *too* high with `k` - I tried

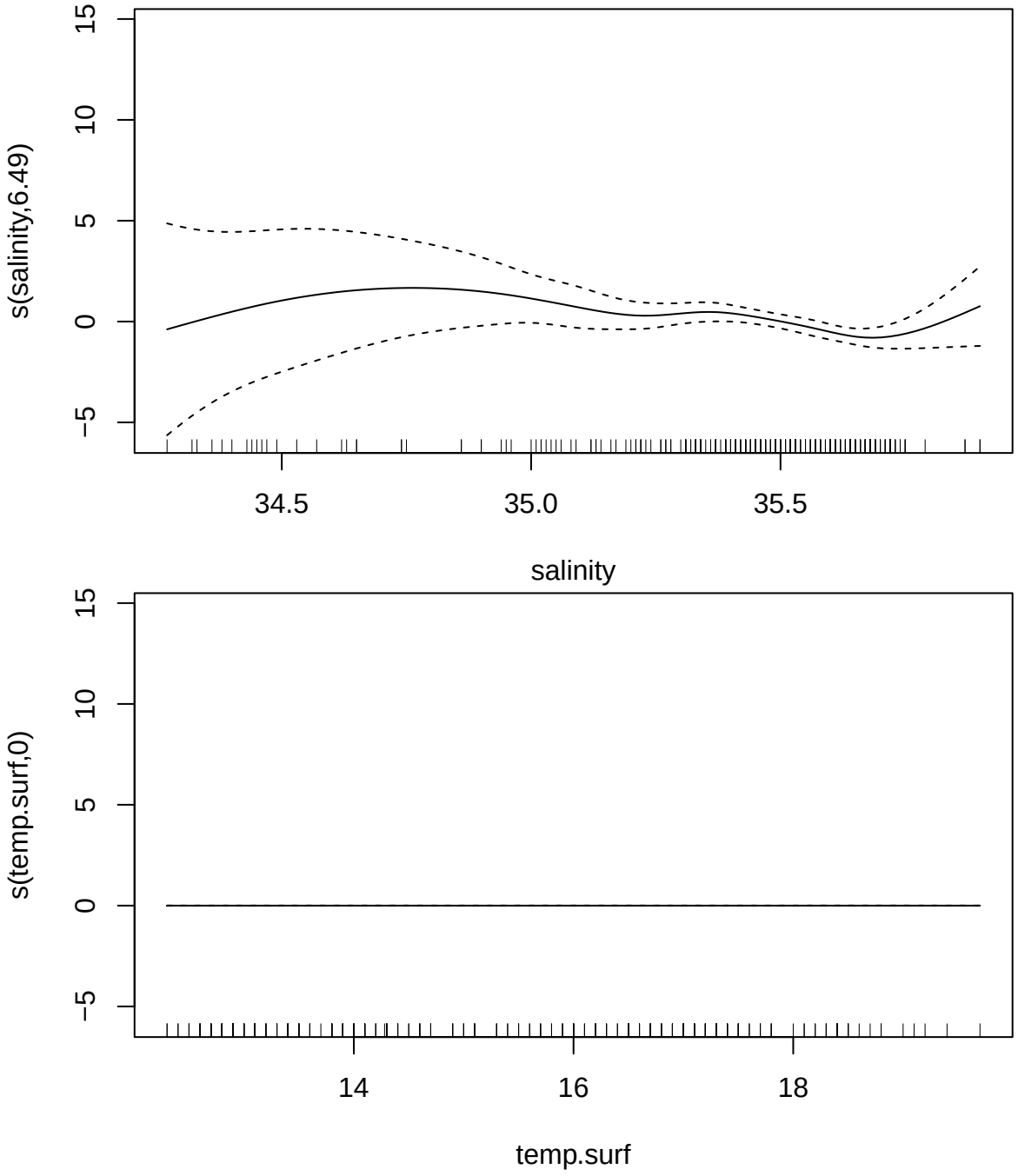
`k = 500` and it kept my computer busy for about 10 minutes (it did eventually produce pretty plot, though probably over-fit).

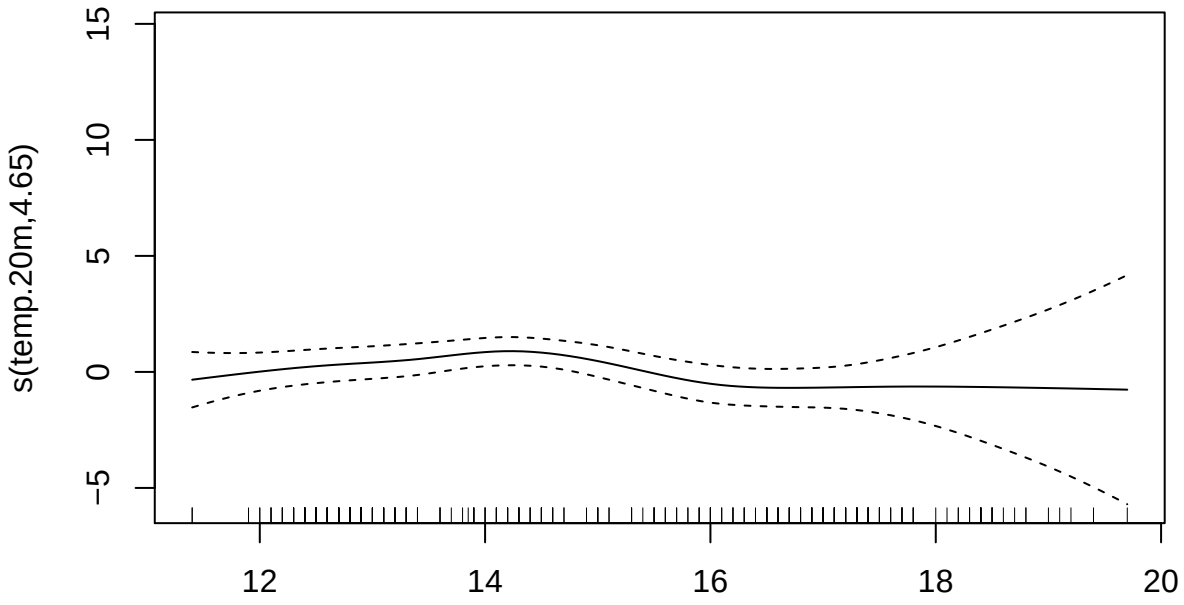
- *For your basis functions,* `mgcv` offers "shrinkage" options as well. (You'll learn more about shrinkage – pulling a regression coefficient towards 0 – when you learn about LASSO next quarter). Instead of the vanilla cubic spline `bs = "cr"` argument, use a cubic spline with shrinkage basis by specifying `bs = "cs"`. Do this for all the smooths *except:* (1) the cubic spline basis with shrinkage does not work for multivariate smooths. Luckily, the thin plate spline basis with shrinkage does, so for the lat/long smooth use `bs = "ts"`. (2) For time of day, it would be really cool to use a *cyclic* smooth. This will make it so that the endpoints of the data (i.e., 11:59 PM and 12:00 AM) have the same smoothness constraints as any other adjacent data points. To use a cyclic cubic basis, specify `bs = "cc"`. In all cases, the choice of basis `bs` is an argument to the smooth function `s()`.
- *Overdispersion:* The `gam()` function has a different way of allowing for overdispersion (quasipoisson) than the `glm` function. Set `scale = -1` to tell `gam` to estimate an overdispersion parameter, otherwise it will act under the assumption that the residual variance is exactly equal to the fitted value.
- *Finally!* Fit your model as described *at length* above, and show the model summary and the plots of the smooths produced by `plot(your_model)`. (If you'd like, you can add residuals to these plots by specifying `residuals = TRUE`; see `?plot.gam` for more options).

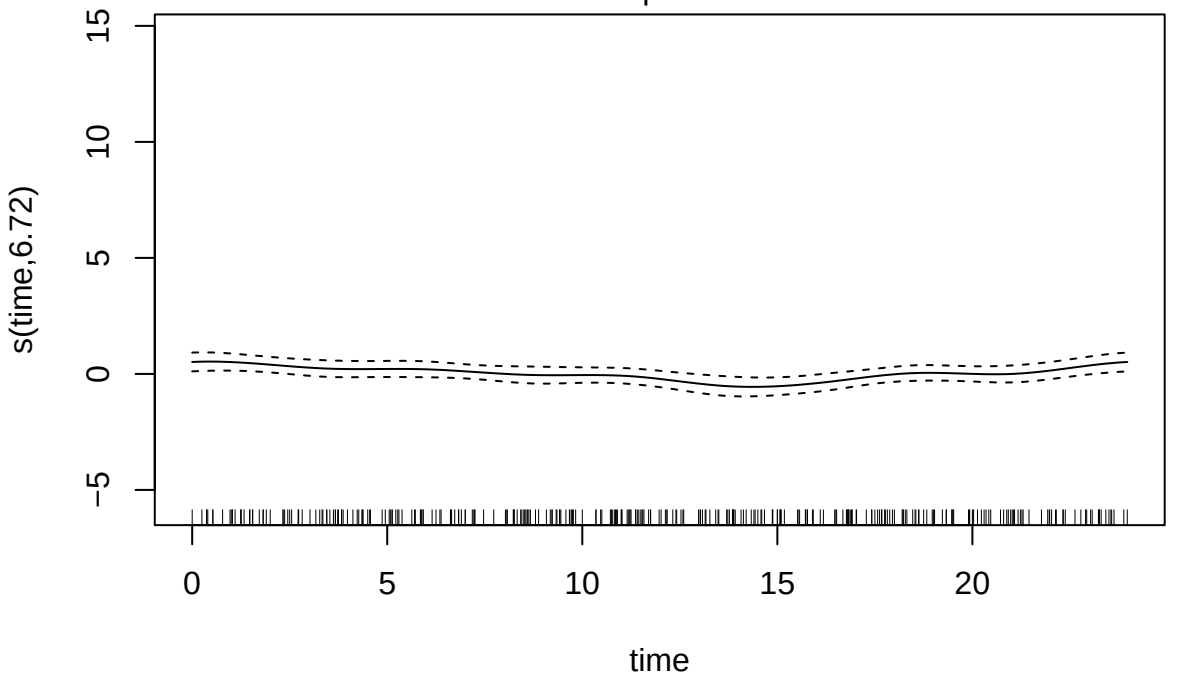
```
library(mgcv)

mod1.egg <- gam(egg.count ~ s(lon, lat, k = 110, bs = "ts") +
                            s(b.depth, bs = "cs") +
                            s(c.dist, bs = "cs") +
                            s(salinity, bs = "cs") +
                            s(temp.surf, bs = "cs") +
                            s(temp.20m, bs = "cs") +
                            s(time, bs = "cc") +
                            offset(log(net.area))
                            ,family = poisson
                            ,scale = -1
                            ,data = mack)

#Plot output and summary
plot.gam(mod1.egg)
```


−1se
s(lon,lat,95.55)
+1se
lat
lon
s(b.depth,0)


s(c.dist,8.46)
c.dist


s(salinity,6.49)
15
10
5
0
−5
34.5
35.0
35.5
salinity
s(temp.surf,0)
15
10
5
0
−5
14
16
18
temp.surf




```
summary(mod1.egg)
```

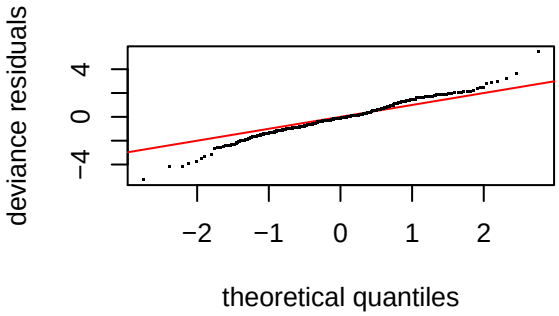
```
##
## Family: poisson
## Link function: log
##
## Formula:
## egg.count ~ s(lon, lat, k = 110, bs = "ts") + s(b.depth, bs = "cs") +
##     s(c.dist, bs = "cs") + s(salinity, bs = "cs") + s(temp.surf,
##     bs = "cs") + s(temp.20m, bs = "cs") + s(time, bs = "cc") +
##     offset(log(net.area))
##
```

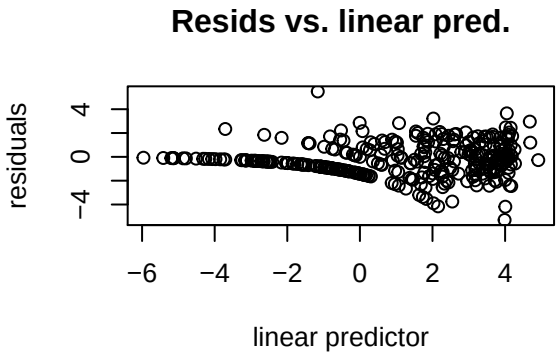
```
## Parametric coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.6169     0.1887   13.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                     edf Ref.df     F  p-value
## s(lon,lat)   9.555e+01    109 6.134  < 2e-16 ***
## s(b.depth)   3.237e-04      9 0.000  0.32802
## s(c.dist)    8.457e+00      9 3.645 1.88e-05 ***
## s(salinity)  6.487e+00      9 2.962 5.33e-05 ***
## s(temp.surf) 3.144e-06      9 0.000  0.60932
## s(temp.20m)  4.654e+00      9 1.758  0.00146 **
## s(time)      6.721e+00      8 1.954  0.01189 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.897   Deviance explained = 94.6%
## GCV = 4.7989  Scale est. = 3.3847    n = 330
```

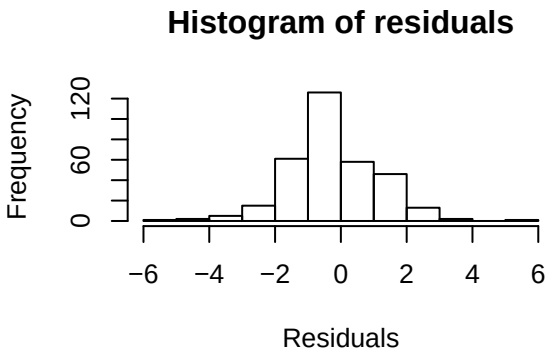
**(c)** Several terms should be clearly not helping your model. You can run an `anova` on your model or simply assess the graphic output to drop the un-helpful covariates. Fit a simpler model, and look again at the plots of the smooths.

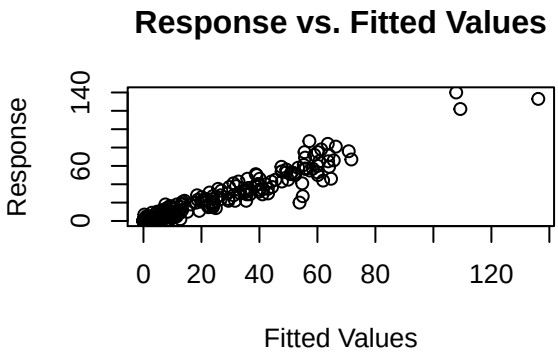
Based on the smooth term p-values from the gam.check output, I can see that temp.surf and b.depth are not statistically significant and can be removed from the model.

```
mod2.egg <- gam(egg.count ~ s(lon, lat, k = 110, bs = "ts") +
                            s(c.dist, bs = "cs") +
                            s(salinity, bs = "cs") +
                            s(temp.20m, bs = "cs") +
                            s(time, bs = "cc") +
                            offset(log(net.area))
                            ,family = poisson
                            ,scale = -1
                            ,data = mack)

#Check mod output
gam.check(mod2.egg)
```



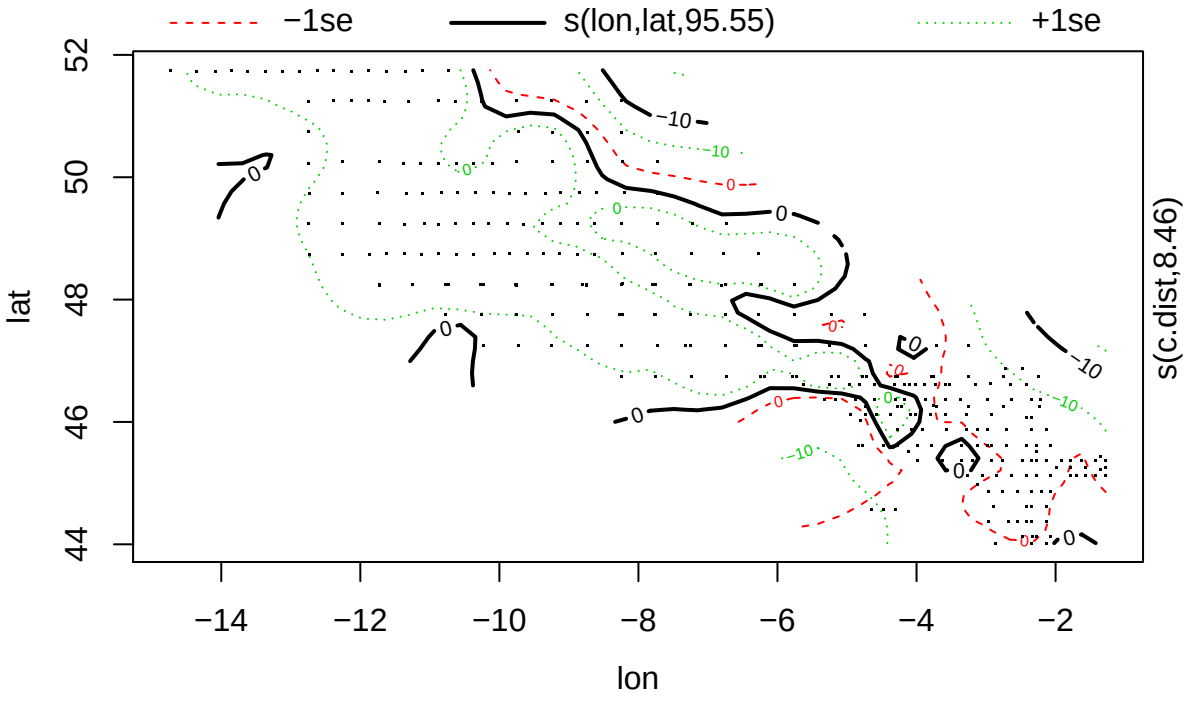

```
##
## Method: GCV   Optimizer: outer newton
## full convergence after 8 iterations.
## Gradient range [-1.049934e-07,1.593773e-06]
## (score 4.798887 & scale 3.384668).
## Hessian positive definite, eigenvalue range [0.01073337,0.03961003].
## Model rank =  145 / 145
##
## Basis dimension (k) checking 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(lon,lat)  109.00  95.55    1.22    1.00
## s(c.dist)     9.00   8.46    0.94    0.26
## s(salinity)   9.00   6.49    0.93    0.24
## s(temp.20m)   9.00   4.65    1.00    0.59
## s(time)       8.00   6.72    0.94    0.23
```

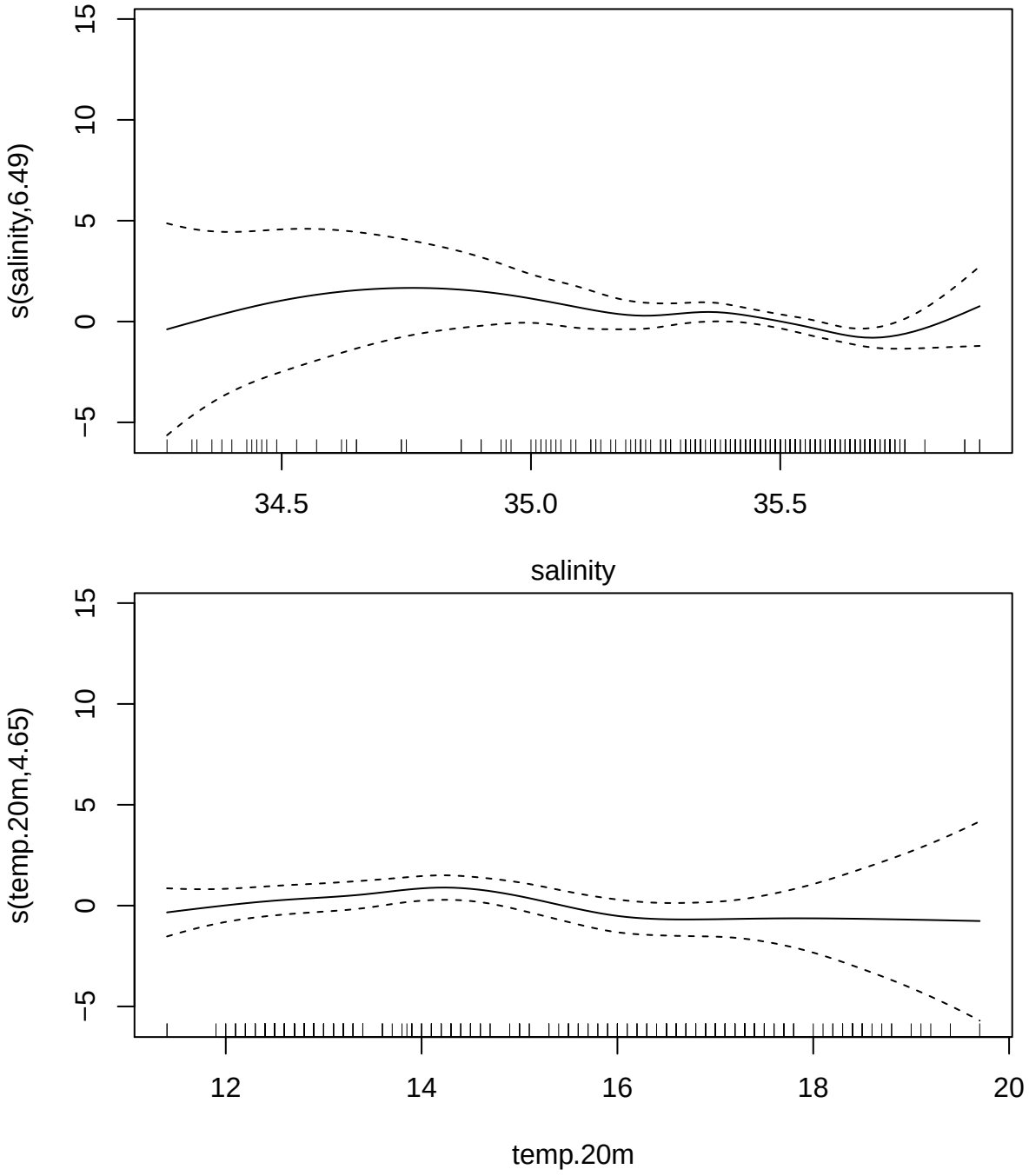
```
summary(mod2.egg)
```

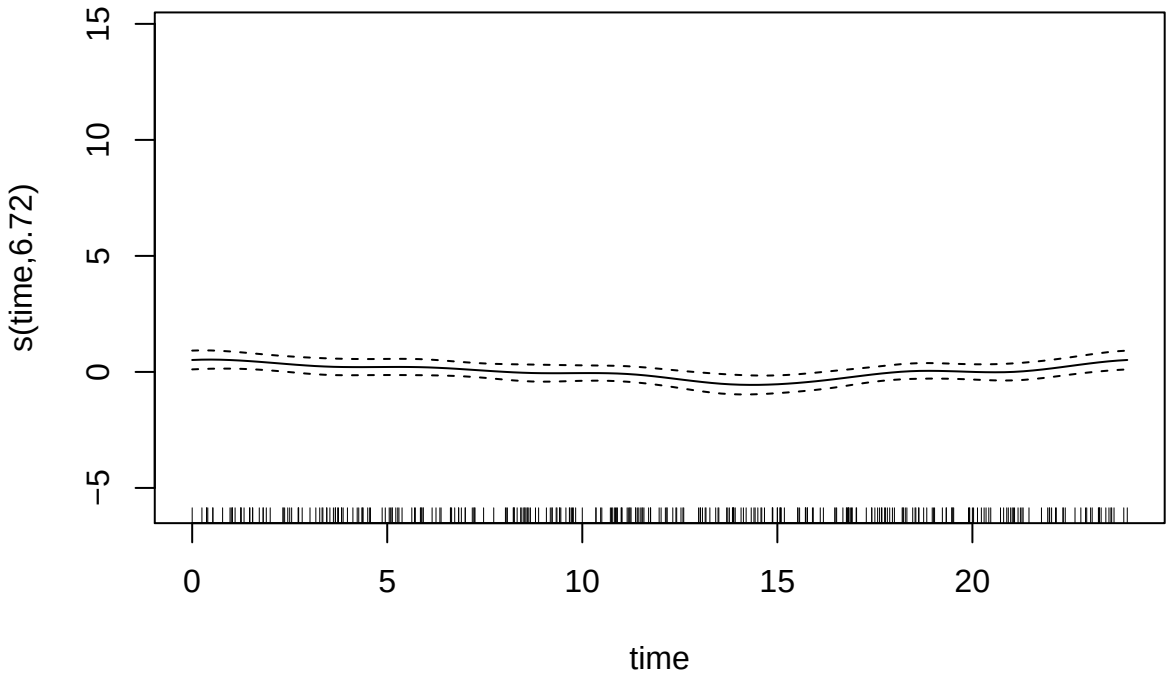
```
##
## Family: poisson
## Link function: log
##
## Formula:
## egg.count ~ s(lon, lat, k = 110, bs = "ts") + s(c.dist, bs = "cs") +
##     s(salinity, bs = "cs") + s(temp.20m, bs = "cs") + s(time,
##     bs = "cc") + offset(log(net.area))
##
## Parametric coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.6169     0.1887   13.87   <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)  95.545    109 6.136  < 2e-16 ***
## s(c.dist)    8.457      9 3.645 1.88e-05 ***
## s(salinity)  6.487      9 2.962 5.33e-05 ***
## s(temp.20m)  4.654      9 1.758  0.00146 **
## s(time)      6.721      8 1.954  0.01188 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.897   Deviance explained = 94.6%
## GCV = 4.7989  Scale est. = 3.3847    n = 330
```

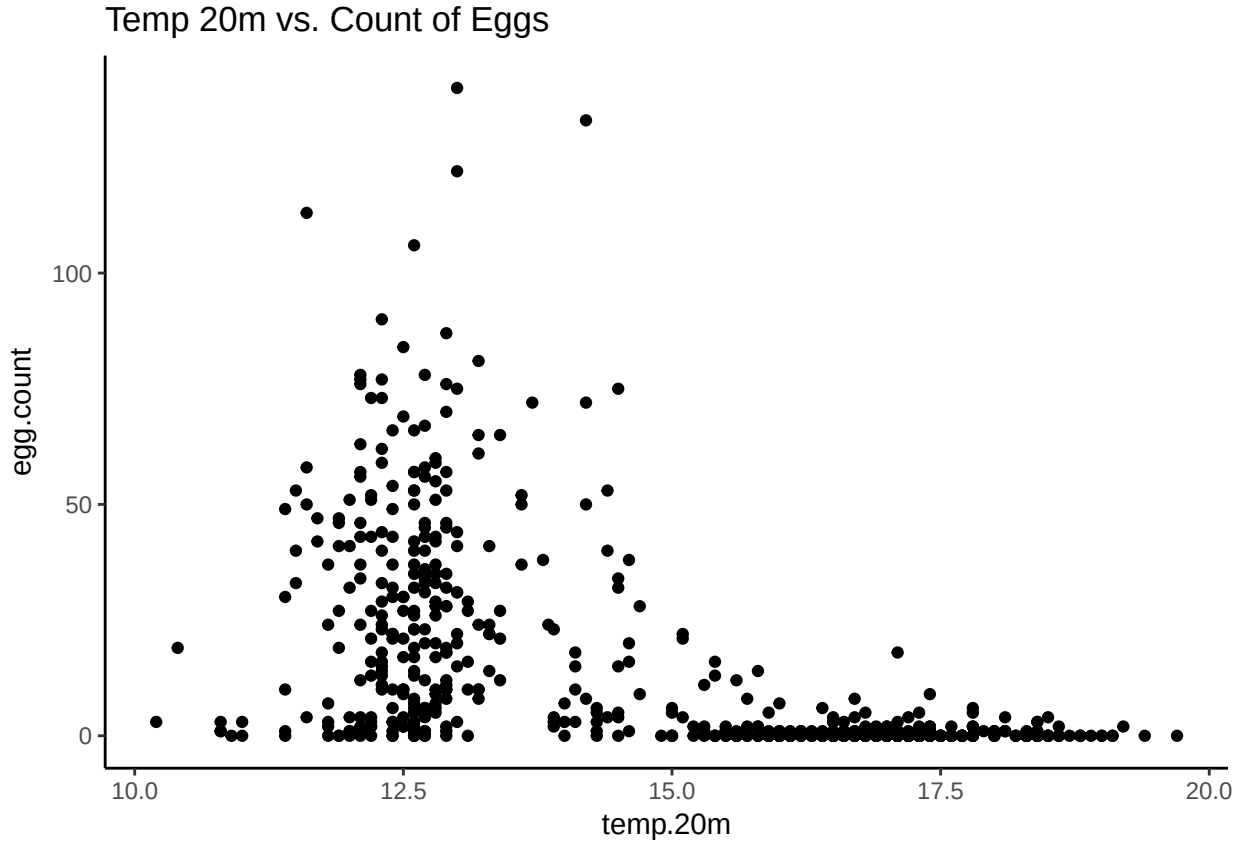
```
#Plot gam
plot.gam(mod2.egg)
```




s(salinity,6.49)
15
10
5
0
−5
34.5
35.0
35.5
salinity
s(temp.20m,4.65)
15
10
5
0
−5
12
14
16
18
20
temp.20m




```
 #Plot relationship of temp.20m to Egg Count
ggplot(data = mack, aes(x = temp.20m, y = egg.count)) +
      geom_point() +
      theme_classic() +
      ggtitle("Temp 20m vs. Count of Eggs")
```



**(d)** In one or two sentences, how would you describe the effect of temperature 20 m below the surface on the count of mackerel eggs?

Examining the spline plot, the effect of temp.20m on count of eggs appears to hover around zero (meaning not a large effect), with a slight increase when the temperate is around 14 degrees. The range of error increases dramatically above 18, which likely indicates there is not substantial data in that range and the spline fit may not be as reliable.