Fall of Warness Power: Spatial

Lindesay Scott-Hayward

23 August 2016

# Real Data: Falls of Warness

### Fit Initial Model

init\_glm<-glm(response ~ as.factor(TideState) + as.factor(WindStrength) + as.factor(SeaState) + as.factor(SimpPrecipitation) + as.factor(CloudCover) + offset(log(areatime)), data=dat, family=quasipoisson)

### Run SALSA1D

factorlist<-c('TideState', 'WindStrength','SeaState', 'SimpPrecipitation', 'CloudCover')  
varlist<-c('Depth', 'MonthInt')  
  
salsa1dlist<-list(fitnessMeasure='QAIC', minKnots\_1d = c(1,1), maxKnots\_1d=c(5,5), startKnots\_1d = c(1,1), degree=c(2,2), maxIterations=100, gaps=c(0,0))  
  
salsa1dout<-runSALSA1D\_withremoval(init\_glm, salsa1dlist, varlist, factorlist, varlist\_cyclicSplines = c('MonthInt'), splineParams = NULL, datain=dat, suppress.printout = TRUE)

## [1] "TideState will be fitted as a factor variable; there are non-zero counts for all levels"  
## [1] "SeaState will be fitted as a factor variable; there are non-zero counts for all levels"  
## [1] "SimpPrecipitation will be fitted as a factor variable; there are non-zero counts for all levels"  
## [1] "CloudCover will be fitted as a factor variable; there are non-zero counts for all levels"

salsa1dout$bestModel$splineParams<-salsa1dout$splineParams

salsa1dout$bestModel<-make.gamMRSea(salsa1dout$bestModel, panelid = dat$newbid, splineParams = salsa1dout$splineParams)  
#splineParams<-bestModel1D$splineParams

summary(salsa1dout$bestModel)

##   
## Call:  
## glm(formula = response ~ as.factor(TideState) + as.factor(WindStrength) +   
## as.factor(SeaState) + as.factor(SimpPrecipitation) + as.factor(CloudCover) +   
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree,   
## Boundary.knots = splineParams[[2]]$bd) + smooth.construct(s(MonthInt,   
## bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1],   
## splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[,   
## -1] + offset(log(areatime)), family = quasipoisson(link = log),   
## data = dat)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -8.960 -0.785 -0.346 -0.159 44.392   
##   
## Coefficients:  
## Estimate  
## (Intercept) 1.37990  
## as.factor(TideState)2 -0.16453  
## as.factor(TideState)3 -0.09665  
## as.factor(WindStrength)1 0.19963  
## as.factor(WindStrength)2 -0.05039  
## as.factor(WindStrength)3 -0.21807  
## as.factor(WindStrength)4 -3.16687  
## as.factor(SeaState)1 -0.42704  
## as.factor(SeaState)2 -0.31036  
## as.factor(SeaState)3 -0.62544  
## as.factor(SeaState)4 -0.11413  
## as.factor(SimpPrecipitation)HEAVY 0.42994  
## as.factor(SimpPrecipitation)LIGHT 0.94273  
## as.factor(SimpPrecipitation)NONE 0.50600  
## as.factor(SimpPrecipitation)SHOWERS 0.83424  
## as.factor(CloudCover)2 -0.23454  
## as.factor(CloudCover)4 0.42382  
## as.factor(CloudCover)6 0.49219  
## as.factor(CloudCover)7 0.98962  
## as.factor(CloudCover)8 0.59955  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)1 -0.35390  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)2 -3.05997  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)3 0.07213  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)4 -1.93341  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)5 -0.68303  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)6 -5.00408  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)7 -5.21509  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]1 -0.99814  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]2 -1.74345  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]3 -2.49821  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]4 -1.43810  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]5 -0.37284  
## Std. Error  
## (Intercept) 0.34789  
## as.factor(TideState)2 0.05401  
## as.factor(TideState)3 0.07434  
## as.factor(WindStrength)1 0.21485  
## as.factor(WindStrength)2 0.22572  
## as.factor(WindStrength)3 0.24053  
## as.factor(WindStrength)4 4.14450  
## as.factor(SeaState)1 0.19528  
## as.factor(SeaState)2 0.21160  
## as.factor(SeaState)3 0.23691  
## as.factor(SeaState)4 0.55089  
## as.factor(SimpPrecipitation)HEAVY 0.37444  
## as.factor(SimpPrecipitation)LIGHT 0.22701  
## as.factor(SimpPrecipitation)NONE 0.20729  
## as.factor(SimpPrecipitation)SHOWERS 0.20993  
## as.factor(CloudCover)2 0.24990  
## as.factor(CloudCover)4 0.24267  
## as.factor(CloudCover)6 0.23908  
## as.factor(CloudCover)7 0.24354  
## as.factor(CloudCover)8 0.23730  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)1 0.23898  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)2 0.28049  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)3 0.08386  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)4 0.18505  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)5 0.10682  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)6 0.36963  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)7 0.66742  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]1 0.09929  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]2 0.11906  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]3 0.15920  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]4 0.10343  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]5 0.07969  
## Robust S.E.  
## (Intercept) 0.46055  
## as.factor(TideState)2 0.08933  
## as.factor(TideState)3 0.12079  
## as.factor(WindStrength)1 0.17161  
## as.factor(WindStrength)2 0.18296  
## as.factor(WindStrength)3 0.26486  
## as.factor(WindStrength)4 1.07073  
## as.factor(SeaState)1 0.16963  
## as.factor(SeaState)2 0.21514  
## as.factor(SeaState)3 0.33238  
## as.factor(SeaState)4 0.63574  
## as.factor(SimpPrecipitation)HEAVY 0.38317  
## as.factor(SimpPrecipitation)LIGHT 0.40782  
## as.factor(SimpPrecipitation)NONE 0.27864  
## as.factor(SimpPrecipitation)SHOWERS 0.31127  
## as.factor(CloudCover)2 0.33712  
## as.factor(CloudCover)4 0.34050  
## as.factor(CloudCover)6 0.33002  
## as.factor(CloudCover)7 0.34537  
## as.factor(CloudCover)8 0.33203  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)1 0.50592  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)2 0.58261  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)3 0.30252  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)4 0.46066  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)5 0.25013  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)6 0.45431  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)7 0.53441  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]1 0.21238  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]2 0.17637  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]3 0.29817  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]4 0.16515  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]5 0.25639  
## t value  
## (Intercept) 2.996  
## as.factor(TideState)2 -1.842  
## as.factor(TideState)3 -0.800  
## as.factor(WindStrength)1 1.163  
## as.factor(WindStrength)2 -0.275  
## as.factor(WindStrength)3 -0.823  
## as.factor(WindStrength)4 -2.958  
## as.factor(SeaState)1 -2.517  
## as.factor(SeaState)2 -1.443  
## as.factor(SeaState)3 -1.882  
## as.factor(SeaState)4 -0.180  
## as.factor(SimpPrecipitation)HEAVY 1.122  
## as.factor(SimpPrecipitation)LIGHT 2.312  
## as.factor(SimpPrecipitation)NONE 1.816  
## as.factor(SimpPrecipitation)SHOWERS 2.680  
## as.factor(CloudCover)2 -0.696  
## as.factor(CloudCover)4 1.245  
## as.factor(CloudCover)6 1.491  
## as.factor(CloudCover)7 2.865  
## as.factor(CloudCover)8 1.806  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)1 -0.700  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)2 -5.252  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)3 0.238  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)4 -4.197  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)5 -2.731  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)6 -11.015  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)7 -9.759  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]1 -4.700  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]2 -9.885  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]3 -8.379  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]4 -8.708  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]5 -1.454  
## Pr(>|t|)  
## (Intercept) 0.00274  
## as.factor(TideState)2 0.06551  
## as.factor(TideState)3 0.42363  
## as.factor(WindStrength)1 0.24474  
## as.factor(WindStrength)2 0.78301  
## as.factor(WindStrength)3 0.41033  
## as.factor(WindStrength)4 0.00310  
## as.factor(SeaState)1 0.01183  
## as.factor(SeaState)2 0.14915  
## as.factor(SeaState)3 0.05989  
## as.factor(SeaState)4 0.85752  
## as.factor(SimpPrecipitation)HEAVY 0.26184  
## as.factor(SimpPrecipitation)LIGHT 0.02080  
## as.factor(SimpPrecipitation)NONE 0.06939  
## as.factor(SimpPrecipitation)SHOWERS 0.00736  
## as.factor(CloudCover)2 0.48662  
## as.factor(CloudCover)4 0.21325  
## as.factor(CloudCover)6 0.13587  
## as.factor(CloudCover)7 0.00417  
## as.factor(CloudCover)8 0.07097  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)1 0.48424  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)2 1.51e-07  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)3 0.81154  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)4 2.71e-05  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)5 0.00632  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)6 < 2e-16  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)7 < 2e-16  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]1 2.62e-06  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]2 < 2e-16  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]3 < 2e-16  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]4 < 2e-16  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]5 0.14591  
##   
## (Intercept) \*\*   
## as.factor(TideState)2 .   
## as.factor(TideState)3   
## as.factor(WindStrength)1   
## as.factor(WindStrength)2   
## as.factor(WindStrength)3   
## as.factor(WindStrength)4 \*\*   
## as.factor(SeaState)1 \*   
## as.factor(SeaState)2   
## as.factor(SeaState)3 .   
## as.factor(SeaState)4   
## as.factor(SimpPrecipitation)HEAVY   
## as.factor(SimpPrecipitation)LIGHT \*   
## as.factor(SimpPrecipitation)NONE .   
## as.factor(SimpPrecipitation)SHOWERS \*\*   
## as.factor(CloudCover)2   
## as.factor(CloudCover)4   
## as.factor(CloudCover)6   
## as.factor(CloudCover)7 \*\*   
## as.factor(CloudCover)8 .   
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)1   
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)2 \*\*\*  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)3   
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)4 \*\*\*  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)5 \*\*   
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)6 \*\*\*  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd)7 \*\*\*  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]1 \*\*\*  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]2 \*\*\*  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]3 \*\*\*  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]4 \*\*\*  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1]5   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for quasipoisson family taken to be 17.10652)  
##   
## Null deviance: 153472 on 26334 degrees of freedom  
## Residual deviance: 93358 on 26303 degrees of freedom  
## AIC: NA  
##   
## Max Panel Size = 52; Number of panels = 7609  
## Number of Fisher Scoring iterations: 8

anova(salsa1dout$bestModel)

## Analysis of 'Wald statistic' Table  
## Model: quasipoisson, link: log  
## Response: response  
## Marginal Testing  
## Max Panel Size = 52; Number of panels = 7609  
##   
## Df  
## as.factor(TideState) 2  
## as.factor(WindStrength) 4  
## as.factor(SeaState) 4  
## as.factor(SimpPrecipitation) 4  
## as.factor(CloudCover) 5  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd) 7  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1] 5  
## X2  
## as.factor(TideState) 3.48  
## as.factor(WindStrength) 24.30  
## as.factor(SeaState) 12.78  
## as.factor(SimpPrecipitation) 12.90  
## as.factor(CloudCover) 60.92  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd) 733.89  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1] 184.30  
## P(>|Chi|)  
## as.factor(TideState) 0.17580  
## as.factor(WindStrength) 6.941e-05  
## as.factor(SeaState) 0.01241  
## as.factor(SimpPrecipitation) 0.01178  
## as.factor(CloudCover) 7.836e-12  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd) < 2.2e-16  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1] < 2.2e-16  
##   
## as.factor(TideState)   
## as.factor(WindStrength) \*\*\*  
## as.factor(SeaState) \*   
## as.factor(SimpPrecipitation) \*   
## as.factor(CloudCover) \*\*\*  
## bs(Depth, knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree, Boundary.knots = splineParams[[2]]$bd) \*\*\*  
## smooth.construct(s(MonthInt, bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1], splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[, -1] \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Tide state is not significant so this is removed from the model and the process repeated:

init\_glm<-glm(response ~ as.factor(WindStrength) + as.factor(SeaState) + as.factor(SimpPrecipitation) + as.factor(CloudCover) + offset(log(areatime)), data=dat, family=quasipoisson)  
  
factorlist<-c('WindStrength','SeaState', 'SimpPrecipitation', 'CloudCover')  
varlist<-c('Depth', 'MonthInt')  
  
salsa1dlist<-list(fitnessMeasure='QAIC', minKnots\_1d = c(1,1), maxKnots\_1d=c(5,5), startKnots\_1d = c(1,1), degree=c(2,2), maxIterations=100, gaps=c(0,0))  
  
salsa1dout<-runSALSA1D\_withremoval(init\_glm, salsa1dlist, varlist, factorlist, varlist\_cyclicSplines = c('MonthInt'), splineParams = NULL, datain=dat,suppress.printout=TRUE)

## [1] "SeaState will be fitted as a factor variable; there are non-zero counts for all levels"  
## [1] "SimpPrecipitation will be fitted as a factor variable; there are non-zero counts for all levels"  
## [1] "CloudCover will be fitted as a factor variable; there are non-zero counts for all levels"

bestModel1D<-make.gamMRSea(salsa1dout$bestModel, panelid = dat$newbid, splineParams = salsa1dout$splineParams)  
#splineParams<-bestModel1D$splineParams

summary(salsa1dout$bestModel)

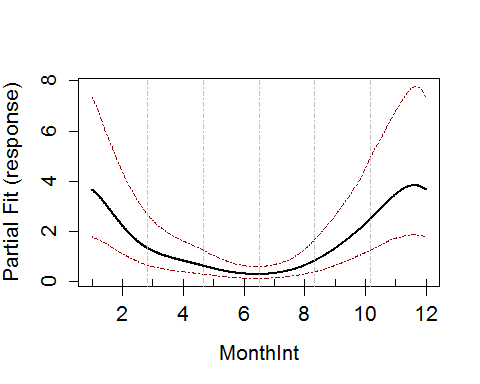
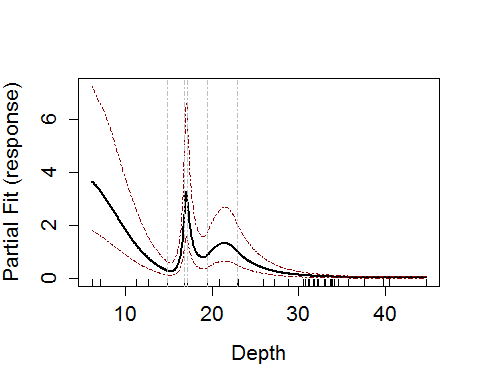
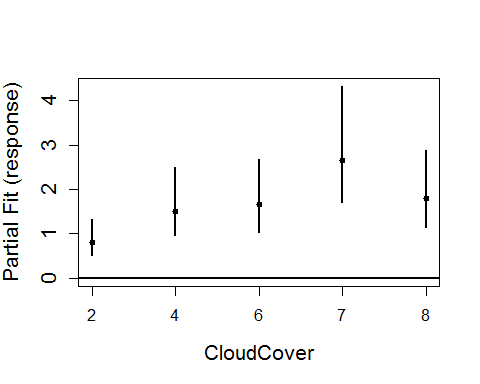
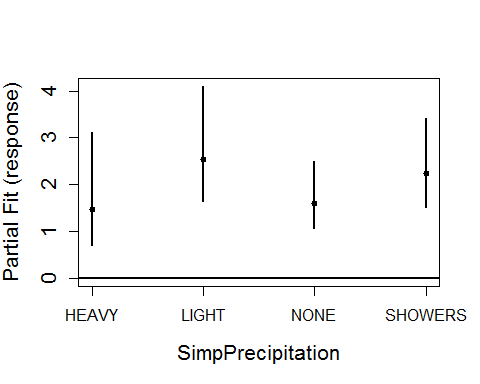
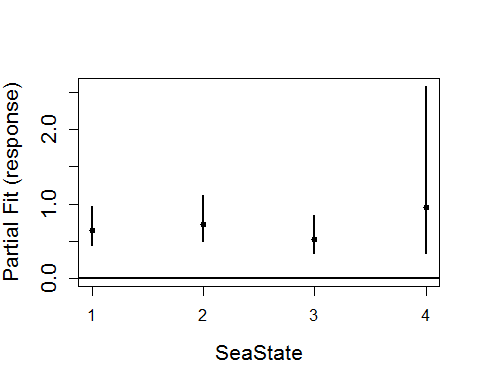
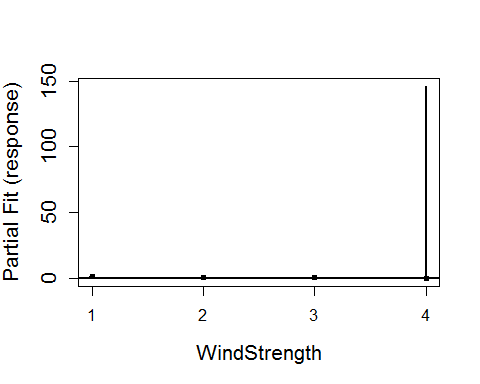
##   
## Call:  
## glm(formula = response ~ as.factor(WindStrength) + as.factor(SeaState) +   
## as.factor(SimpPrecipitation) + as.factor(CloudCover) + bs(Depth,   
## knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree,   
## Boundary.knots = splineParams[[2]]$bd) + smooth.construct(s(MonthInt,   
## bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1],   
## splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[,   
## -1] + offset(log(areatime)), family = quasipoisson(link = log),   
## data = dat)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -9.131 -0.784 -0.348 -0.160 45.307   
##   
## Coefficients:  
## Estimate Std. Error Robust S.E.  
## (Intercept) 1.2979590 0.3472927 0.5253050  
## as.factor(WindStrength)1 0.2385527 0.2137071 0.2325492  
## as.factor(WindStrength)2 -0.0005603 0.2242867 0.2452593  
## as.factor(WindStrength)3 -0.1802945 0.2397137 0.3053152  
## as.factor(WindStrength)4 -3.0614311 4.1496306 1.1034945  
## as.factor(SeaState)1 -0.4298470 0.1944559 0.1616692  
## as.factor(SeaState)2 -0.3208143 0.2107866 0.2071337  
## as.factor(SeaState)3 -0.6373685 0.2365762 0.3230395  
## as.factor(SeaState)4 -0.0490167 0.5507585 0.6501604  
## as.factor(SimpPrecipitation)HEAVY 0.3797335 0.3742401 0.4049794  
## as.factor(SimpPrecipitation)LIGHT 0.9319042 0.2269228 0.3633170  
## as.factor(SimpPrecipitation)NONE 0.4752105 0.2072862 0.2937780  
## as.factor(SimpPrecipitation)SHOWERS 0.8038986 0.2099721 0.2990027  
## as.factor(CloudCover)2 -0.2167560 0.2499365 0.3321969  
## as.factor(CloudCover)4 0.4135122 0.2425850 0.3417430  
## as.factor(CloudCover)6 0.5019924 0.2390359 0.3238095  
## as.factor(CloudCover)7 0.9756647 0.2435947 0.3546166  
## as.factor(CloudCover)8 0.5926870 0.2373257 0.3200180  
## s(Depth)1 -0.3587022 0.2393925 0.3214877  
## s(Depth)2 -3.0533656 0.2810146 0.2851829  
## s(Depth)3 0.0712531 0.0839907 0.1553112  
## s(Depth)4 -1.9312846 0.1852636 0.2633764  
## s(Depth)5 -0.6848871 0.1069783 0.1247200  
## s(Depth)6 -4.9976039 0.3703469 0.3708664  
## s(Depth)7 -5.2179115 0.6688866 0.4681252  
## s(MonthInt)1 -1.0073866 0.0991962 0.1470478  
## s(MonthInt)2 -1.7572224 0.1191836 0.1349274  
## s(MonthInt)3 -2.5044117 0.1587782 0.1466901  
## s(MonthInt)4 -1.4569626 0.1033430 0.1175752  
## s(MonthInt)5 -0.3753892 0.0797580 0.1543173  
## t value Pr(>|t|)   
## (Intercept) 2.471 0.01348 \*   
## as.factor(WindStrength)1 1.026 0.30499   
## as.factor(WindStrength)2 -0.002 0.99818   
## as.factor(WindStrength)3 -0.591 0.55485   
## as.factor(WindStrength)4 -2.774 0.00554 \*\*   
## as.factor(SeaState)1 -2.659 0.00785 \*\*   
## as.factor(SeaState)2 -1.549 0.12144   
## as.factor(SeaState)3 -1.973 0.04850 \*   
## as.factor(SeaState)4 -0.075 0.93990   
## as.factor(SimpPrecipitation)HEAVY 0.938 0.34843   
## as.factor(SimpPrecipitation)LIGHT 2.565 0.01032 \*   
## as.factor(SimpPrecipitation)NONE 1.618 0.10576   
## as.factor(SimpPrecipitation)SHOWERS 2.689 0.00718 \*\*   
## as.factor(CloudCover)2 -0.652 0.51409   
## as.factor(CloudCover)4 1.210 0.22629   
## as.factor(CloudCover)6 1.550 0.12109   
## as.factor(CloudCover)7 2.751 0.00594 \*\*   
## as.factor(CloudCover)8 1.852 0.06403 .   
## s(Depth)1 -1.116 0.26454   
## s(Depth)2 -10.707 < 2e-16 \*\*\*  
## s(Depth)3 0.459 0.64640   
## s(Depth)4 -7.333 2.32e-13 \*\*\*  
## s(Depth)5 -5.491 4.02e-08 \*\*\*  
## s(Depth)6 -13.475 < 2e-16 \*\*\*  
## s(Depth)7 -11.146 < 2e-16 \*\*\*  
## s(MonthInt)1 -6.851 7.51e-12 \*\*\*  
## s(MonthInt)2 -13.023 < 2e-16 \*\*\*  
## s(MonthInt)3 -17.073 < 2e-16 \*\*\*  
## s(MonthInt)4 -12.392 < 2e-16 \*\*\*  
## s(MonthInt)5 -2.433 0.01500 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for quasipoisson family taken to be 17.15019)  
##   
## Null deviance: 153472 on 26334 degrees of freedom  
## Residual deviance: 93518 on 26305 degrees of freedom  
## AIC: NA  
##   
## Max Panel Size = 1; Number of panels = 26335  
## Number of Fisher Scoring iterations: 8

anova(salsa1dout$bestModel)

## Analysis of 'Wald statistic' Table  
## Model: quasipoisson, link: log  
## Response: response  
## Marginal Testing  
## Max Panel Size = 1; Number of panels = 26335  
##   
## Df X2 P(>|Chi|)   
## as.factor(WindStrength) 4 14.11 0.006965 \*\*   
## as.factor(SeaState) 4 11.34 0.022974 \*   
## as.factor(SimpPrecipitation) 4 18.05 0.001209 \*\*   
## as.factor(CloudCover) 5 52.77 3.748e-10 \*\*\*  
## s(Depth) 7 1448.11 < 2.2e-16 \*\*\*  
## s(MonthInt) 5 493.16 < 2.2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

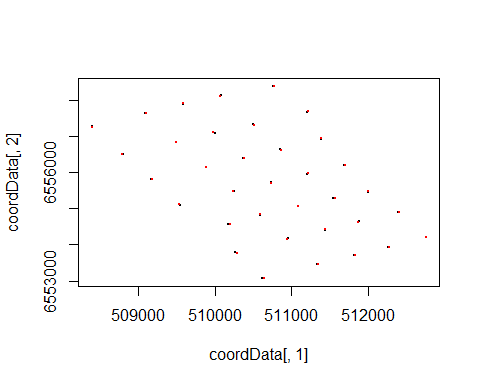
runPartialPlots(bestModel1D, data = dat, factorlist.in = factorlist, varlist.in = varlist, showKnots = T)

## [1] "Making partial plots"



### Run SALSA2D

knotgrid<-getKnotgrid(cbind(dat$x.pos, dat$y.pos))



distMats<-makeDists(cbind(dat$x.pos, dat$y.pos), na.omit(knotgrid))  
# choose sequence of radii  
r\_seq<-getRadiiChoices(8, distMats$dataDist)  
   
salsa2dlist<-list(fitnessMeasure = 'QAIC', knotgrid = knotgrid, knotdim = c(100, 100), startKnots=2, minKnots=2, maxKnots=20, r\_seq=r\_seq, gap=0)

salsa2dOutput<-runSALSA2D(bestModel1D, salsa2dlist, d2k=distMats$dataDist, k2k=distMats$knotDist, splineParams=salsa1dout$splineParams, tol=0, chooserad=F, panels=NULL, suppress.printout=TRUE)

bestModel<-make.gamMRSea(salsa2dOutput$bestModel, panelid = dat$newbid, splineParams = salsa2dOutput$splineParams, varshortnames = varlist, gamMRSea=TRUE)  
  
rm(splineParams, dists)

summary(bestModel)

##   
## Call:  
## gamMRSea(formula = response ~ as.factor(WindStrength) + as.factor(SeaState) +   
## as.factor(SimpPrecipitation) + as.factor(CloudCover) + bs(Depth,   
## knots = splineParams[[2]]$knots, degree = splineParams[[2]]$degree,   
## Boundary.knots = splineParams[[2]]$bd) + smooth.construct(s(MonthInt,   
## bs = "cc", k = (length(splineParams[[3]]$knots)) + 2), knots = list(MonthInt = as.numeric(c(splineParams[[3]]$bd[1],   
## splineParams[[3]]$knots, splineParams[[3]]$bd[2]))), data = data.frame(MonthInt))$X[,   
## -1] + LocalRadialFunction(radiusIndices, dists, radii, aR) +   
## offset(log(areatime)), family = quasipoisson(link = log),   
## data = dat, splineParams = splineParams)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -9.295 -0.787 -0.330 -0.150 44.289   
##   
## Coefficients:  
## Estimate Std. Error Robust S.E.  
## (Intercept) 1.418282 0.350158 0.470460  
## as.factor(WindStrength)1 0.235469 0.214980 0.173592  
## as.factor(WindStrength)2 -0.001657 0.225626 0.188024  
## as.factor(WindStrength)3 -0.179624 0.241150 0.266180  
## as.factor(WindStrength)4 -3.061876 4.174258 1.072595  
## as.factor(SeaState)1 -0.431144 0.195629 0.170696  
## as.factor(SeaState)2 -0.323977 0.212071 0.212588  
## as.factor(SeaState)3 -0.643951 0.238020 0.333627  
## as.factor(SeaState)4 -0.058790 0.554051 0.628296  
## as.factor(SimpPrecipitation)HEAVY 0.381286 0.376458 0.382166  
## as.factor(SimpPrecipitation)LIGHT 0.931807 0.228274 0.405631  
## as.factor(SimpPrecipitation)NONE 0.477007 0.208517 0.282041  
## as.factor(SimpPrecipitation)SHOWERS 0.804586 0.211219 0.306882  
## as.factor(CloudCover)2 -0.216057 0.251418 0.341638  
## as.factor(CloudCover)4 0.412325 0.244018 0.339924  
## as.factor(CloudCover)6 0.501006 0.240449 0.333282  
## as.factor(CloudCover)7 0.972994 0.245034 0.344918  
## as.factor(CloudCover)8 0.593636 0.238730 0.333076  
## s(Depth)1 0.625328 0.346284 0.466458  
## s(Depth)2 -3.522476 0.374977 0.658378  
## s(Depth)3 0.365728 0.109365 0.328411  
## s(Depth)4 -0.805629 0.264939 0.518759  
## s(Depth)5 -0.547321 0.112257 0.244079  
## s(Depth)6 -3.921533 0.428194 0.513367  
## s(Depth)7 -4.941049 0.718398 0.618258  
## s(MonthInt)1 -1.003904 0.099805 0.204614  
## s(MonthInt)2 -1.754632 0.119906 0.176593  
## s(MonthInt)3 -2.501186 0.159740 0.300205  
## s(MonthInt)4 -1.452877 0.103960 0.171045  
## s(MonthInt)5 -0.379891 0.080235 0.247500  
## s(x.pos, y.pos)b1 -1.796023 0.622741 0.635347  
## s(x.pos, y.pos)b2 -0.895110 0.155922 0.298293  
## t value Pr(>|t|)   
## (Intercept) 3.015 0.00258 \*\*   
## as.factor(WindStrength)1 1.356 0.17497   
## as.factor(WindStrength)2 -0.009 0.99297   
## as.factor(WindStrength)3 -0.675 0.49980   
## as.factor(WindStrength)4 -2.855 0.00431 \*\*   
## as.factor(SeaState)1 -2.526 0.01155 \*   
## as.factor(SeaState)2 -1.524 0.12753   
## as.factor(SeaState)3 -1.930 0.05360 .   
## as.factor(SeaState)4 -0.094 0.92545   
## as.factor(SimpPrecipitation)HEAVY 0.998 0.31844   
## as.factor(SimpPrecipitation)LIGHT 2.297 0.02162 \*   
## as.factor(SimpPrecipitation)NONE 1.691 0.09080 .   
## as.factor(SimpPrecipitation)SHOWERS 2.622 0.00875 \*\*   
## as.factor(CloudCover)2 -0.632 0.52712   
## as.factor(CloudCover)4 1.213 0.22514   
## as.factor(CloudCover)6 1.503 0.13279   
## as.factor(CloudCover)7 2.821 0.00479 \*\*   
## as.factor(CloudCover)8 1.782 0.07471 .   
## s(Depth)1 1.341 0.18007   
## s(Depth)2 -5.350 8.86e-08 \*\*\*  
## s(Depth)3 1.114 0.26545   
## s(Depth)4 -1.553 0.12044   
## s(Depth)5 -2.242 0.02494 \*   
## s(Depth)6 -7.639 2.27e-14 \*\*\*  
## s(Depth)7 -7.992 1.38e-15 \*\*\*  
## s(MonthInt)1 -4.906 9.34e-07 \*\*\*  
## s(MonthInt)2 -9.936 < 2e-16 \*\*\*  
## s(MonthInt)3 -8.332 < 2e-16 \*\*\*  
## s(MonthInt)4 -8.494 < 2e-16 \*\*\*  
## s(MonthInt)5 -1.535 0.12482   
## s(x.pos, y.pos)b1 -2.827 0.00470 \*\*   
## s(x.pos, y.pos)b2 -3.001 0.00270 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for quasipoisson family taken to be 17.35436)  
##   
## Null deviance: 153472 on 26334 degrees of freedom  
## Residual deviance: 92863 on 26303 degrees of freedom  
## AIC: NA  
##   
## Max Panel Size = 52; Number of panels = 7609  
## Number of Fisher Scoring iterations: 8

anova(bestModel)

## Analysis of 'Wald statistic' Table  
## Model: quasipoisson, link: log  
## Response: response  
## Marginal Testing  
## Max Panel Size = 52; Number of panels = 7609  
##   
## Df X2 P(>|Chi|)  
## as.factor(WindStrength) 4 23.23 0.0001139  
## as.factor(SeaState) 4 12.49 0.0140591  
## as.factor(SimpPrecipitation) 4 13.31 0.0098543  
## as.factor(CloudCover) 5 58.62 2.348e-11  
## s(Depth) 7 476.21 < 2.2e-16  
## s(MonthInt) 5 190.03 < 2.2e-16  
## LocalRadialFunction(radiusIndices, dists, radii, aR) 2 11.09 0.0039046  
##   
## as.factor(WindStrength) \*\*\*  
## as.factor(SeaState) \*   
## as.factor(SimpPrecipitation) \*\*   
## as.factor(CloudCover) \*\*\*  
## s(Depth) \*\*\*  
## s(MonthInt) \*\*\*  
## LocalRadialFunction(radiusIndices, dists, radii, aR) \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

### Runs Test Check

Data generated under the null hypothesis of the runs test; independence and with no change induced at this stage.

nsim<-500  
d<-as.numeric(summary(bestModel)$dispersion)  
newdat<-generateNoise(nsim, fitted(bestModel), family='poisson', d=d)

500 sets of noisy data are simulated from the model using an overdispersed Poisson distribution where $ = $ 31.

empdistribution<-getRunsCritVals(n.sim = nsim, simData=newdat,   
 model = bestModel, data = dat, plot=TRUE,   
 returnDist = TRUE, dots=FALSE)

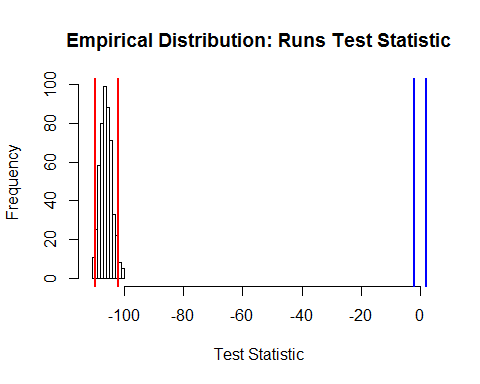


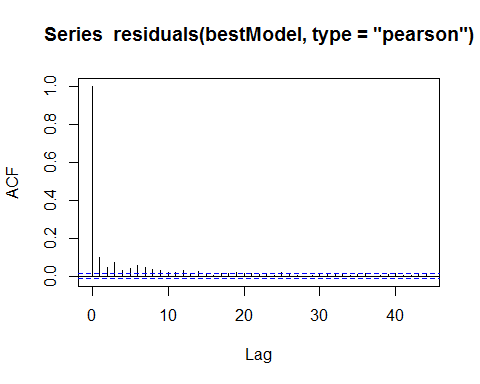
Figure showing the distribution of test statistics from a runs test. The red lines show the lower 2.5% and upper 97.5% critical values of the empirical distribution and the blue lines are from the Normal () distribution.

Evaluate the runs test using the empirical distribution to determine if the data are independent

runs.test(residuals(bestModel, type='pearson'), critvals = empdistribution)

##   
## Runs Test - Two sided; Empirical Distribution  
##   
## data: residuals(bestModel, type = "pearson")  
## Standardized Runs Statistic = -104, p-value = 0.864

acf(residuals(bestModel, type='pearson'))



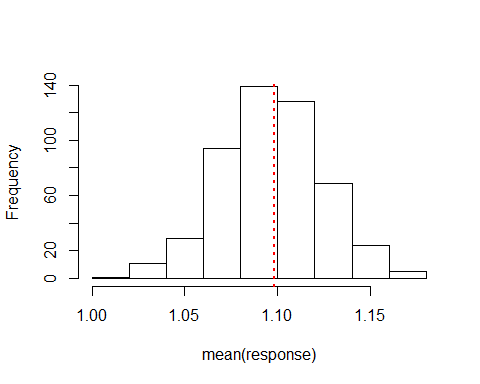
The residuals are considered independent so for the power analysis, the data will be generated as independent.

### Data Generation

Are the generated data consistent with the original data?

#### Mean

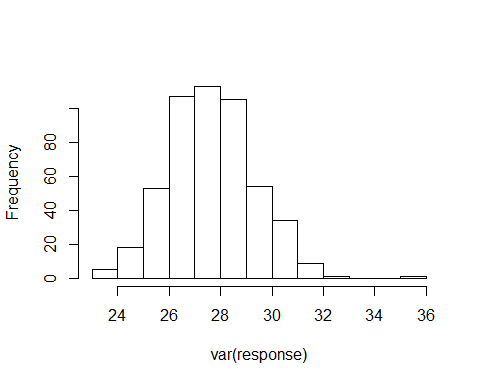
hist(apply(newdat, 2, mean), main='', xlab='mean(response)')  
abline(v=mean(dat$response), col='red', lwd=2, lty=3)



Histogram of the mean of the simulated data, with the red line representing the mean of the original data.

#### Variance

hist(apply(newdat, 2, var), main='', xlab='var(response)')  
abline(v=var(dat$response), col='red', lwd=2, lty=3)



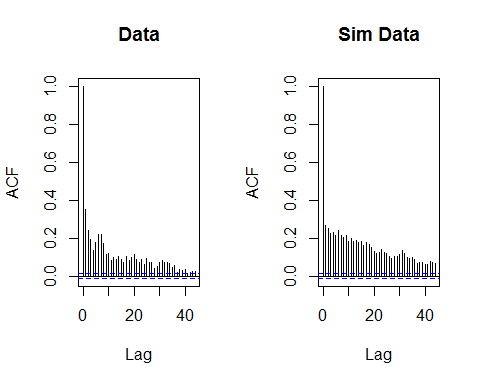
Histogram of the variance of the simulated data, with the red line representing the variance of the original data.

var(dat$response)

## [1] 76.3464

#### Correlation

par(mfrow=c(1,2))  
acf(dat$response, main='Data')  
acf(newdat[,50], main='Sim Data')



#### Check the dispersion parameter

fowsim\_glm<-update(bestModel, newdat[,1] ~ .)  
summary(fowsim\_glm)$dispersion

## [1] 14.72126

#### Data distribution

require(fields)  
par(mfrow=c(2,2))  
quilt.plot(dat$x.pos, dat$y.pos, dat$response, main='Original Data', nrow=18, ncol=14, asp=1)  
quilt.plot(dat$x.pos, dat$y.pos, fitted(init\_glm), main='Fitted Values', nrow=18, ncol=14, asp=1)  
quilt.plot(dat$x.pos, dat$y.pos, newdat[,1], main='Simulated Data A', nrow=18, ncol=14, asp=1)  
quilt.plot(dat$x.pos, dat$y.pos, newdat[,10], main='Simulated Data B', nrow=18, ncol=14, asp=1)

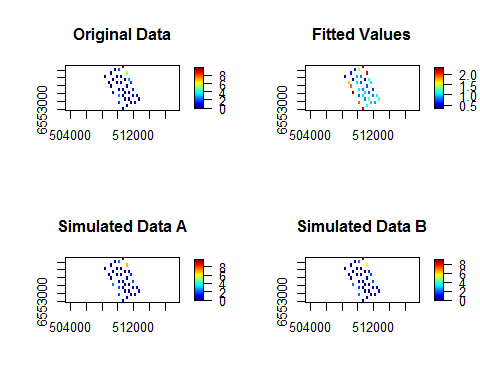


Figure showing the spatial distribution of the original data, the fitted values from the model and two examples of the simulated data.

### Power Analysis

Generate data for power analysis with impact of 50%

nsim=500  
impdata<-genOverallchangeData(log(0.5), bestModel, data = dat, panels = 'newbidNum')  
newdat<-generateNoise(nsim, impdata$truth, family='poisson', d=d)

sum(newdat[impdata$eventphase==0,1])

## [1] 29267

sum(newdat[impdata$eventphase==1,1])

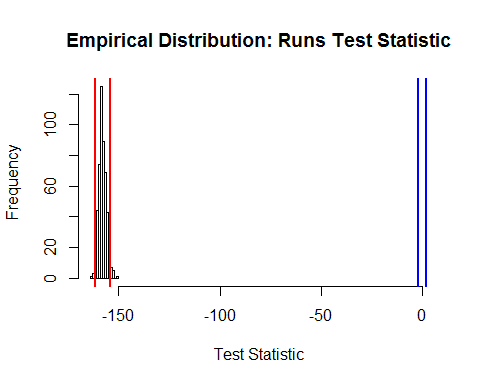
## [1] 13210

Update the initial model to include the eventphase term, indicating overall change.

bestModel$splineParams[[1]]$dist<-rbind(bestModel$splineParams[[1]]$dist, bestModel$splineParams[[1]]$dist)  
fowsim\_glm<-update(bestModel, newdat[,1]~. + eventphase, data=impdata)  
fowsim\_glm$panels<-impdata$panels

Using the simulated data with noise but no correlation, estimate the new values for the empirical distribution for the runs test.

# make sure that the independent data is used to get the null distribution  
empdistpower<-getRunsCritVals(n.sim = nsim, simData=newdat,   
 model = fowsim\_glm, data = impdata, plot=TRUE,   
 returnDist = TRUE, dots=FALSE)



If an offset is present in the model, when setting up the prediction grid the area of each gridcell should be provided so that an accurate estimate of the abundance can be made at the outputs stage.

data("fowshco.grid")  
predictdata<-rbind(data.frame(fowshco.grid, TideState=1, WindStrength=0, SeaState=1, SimpPrecipitation='NONE', CloudCover=7, MonthInt=6, areatime=fowshco.grid$Area, eventphase=0), data.frame(fowshco.grid, TideState=1, WindStrength=0, SeaState=1, SimpPrecipitation='NONE', CloudCover=7, MonthInt=6,areatime=fowshco.grid$Area, eventphase=1))  
  
g2k<-makeDists(cbind(predictdata$x.pos, predictdata$y.pos), knotcoords = na.omit(knotgrid), knotmat = FALSE)$dataDist

nsim=100  
system.time(  
powerout<-powerSimOverallChange(newdat, fowsim\_glm, empdistpower, nsim=nsim, powercoefid=length(coef(fowsim\_glm)), predictionGrid=predictdata, g2k=g2k, splineParams=splineParams, sigdif=TRUE, n.boot=500, impact.loc=c(510700, 6555700))  
)

## user system elapsed   
## 11607.00 21.06 11630.55

# Power Outputs:

To calculate the summary, the values for certain parameters are given under the null hypothesis of 'no change'. If these parameters are not provided, then the summary will not show the comparisons (see below)

### Summary output

summary(powerout, truebeta=log(0.5))

##   
## ++++ Summary of Power Analysis Output ++++  
##   
## Number of power simulations = 100  
## Number of no change simulations = NA   
##   
## Power to select 'change' term:  
##   
## Under Change (true parameter = -0.6931472) = 100%  
## Under no change = Null distribution not specified  
##   
## Coverage for 'change' coefficient:  
##   
## Under change = 95%  
## Under no change = Null distribution not specified  
##   
## Overall Abundance Summary with 95% Confidence Intervals:  
##   
## Abundance LowerCI UpperCI  
## Before 1557.5 992.0 2217  
## After 777.4 487.3 1125  
##   
## Note: These calculations assume the correct area has been given for each prediction grid cell.

In order to assess the summary calculations, we estimate the null parameters to provide the full summary.

null.output<-pval.coverage.null(newdat.ind = newdat, newdat.corr = NULL, model = fowsim\_glm, nsim = 500, powercoefid = length(coef(fowsim\_glm)))

summary(powerout, null.output, truebeta=log(0.5))

##   
## ++++ Summary of Power Analysis Output ++++  
##   
## Number of power simulations = 100  
## Number of no change simulations = 250   
##   
## Power to select 'change' term:  
##   
## Under Change (true parameter = -0.6931472) = 100%  
## Under no change (true parameter = 0) = 3.2%  
##   
## Coverage for 'change' coefficient:  
##   
## Under model = 95%  
## Under no change = 96.4%  
##   
## Overall Abundance Summary with 95% Confidence Intervals:  
##   
## Abundance LowerCI UpperCI  
## Before 1557.5 992.0 2217  
## After 777.4 487.3 1125  
##   
## Note: These calculations assume the correct area has been given for each prediction grid cell.

We expect to see the null power to be 5%, indicating a 5% error rate for inclusion of the eventphase term when the data do not contain any change. We also expect that the coverage for the null coefficient is 95%. This indicates that 5% of the time, the confidence intervals for the eventphase coefficient did not include the truth (0; no change).

### Proportion of significant differences:

plotdata<-plot.sigdiff(powerout, predictdata[predictdata$eventphase==0,c('x.pos', 'y.pos')], tailed='two', error.rate = 0.05, family=FALSE)  
plotdata

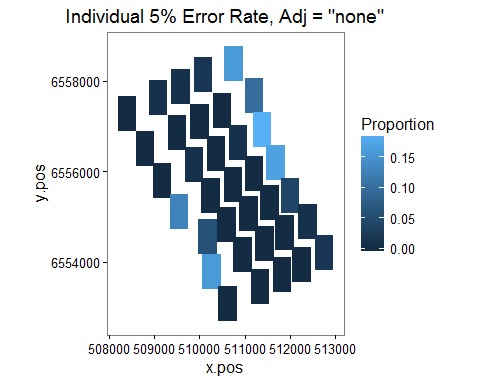


Figure showing the proportion of simulations which showed a significant difference (post event - pre event) in a given grid cell. Significance at the 5% level was determined by comparing the estimated difference in a given cell to the distribution of differences in that cell under the null hypothesis of no-change.

plotdata<-plot.sigdiff(powerout, predictdata[predictdata$eventphase==0,c('x.pos', 'y.pos')], tailed='two', error.rate = 0.05, family=TRUE)  
plotdata

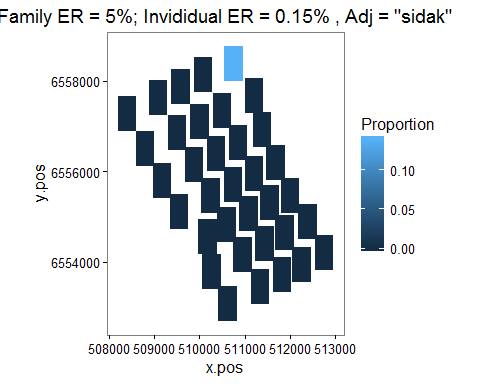


Figure showing the proportion of simulations which showed a significant difference (post event - pre event) in a given grid cell. Significance at the family 5% level was determined by using the Sidak adjustment.

### Distance to event site plot

d2impPlot<-plot.d2imp(powerout)  
d2impPlot

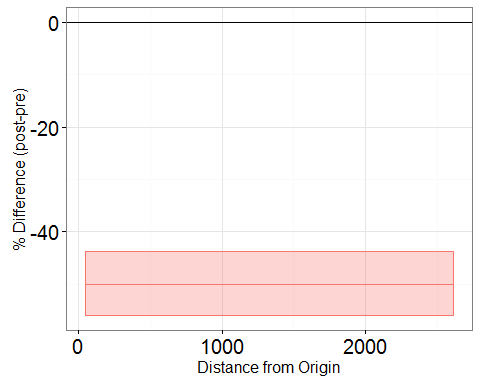


Figure showing the effect of the event on animal numbers with distance from the event site; post event - pre event. The confidence intervals are 95% bootstrap intervals.

d2impPlot<-plot.d2imp(powerout, pct.diff = TRUE)  
  
d2impPlot

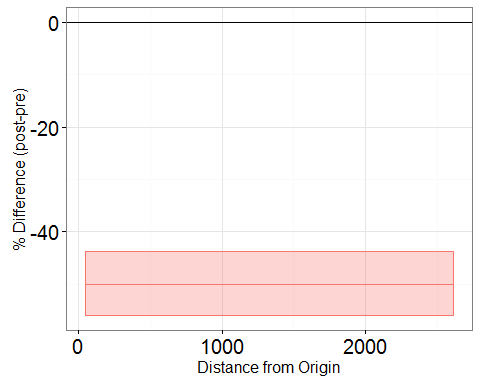


Figure showing the effect of the event on animal numbers with distance from the event site; post event - pre event. The confidence intervals are 95% bootstrap intervals.