Outflow_for_bypass_flows

Laura Twardochleb

2023-06-20

```
knitr::opts_chunk$set(echo = TRUE)
```

1. Global Code and Functions

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0     v purrr   1.0.1
## v tibble  3.1.8     v dplyr   1.1.0
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.3     v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(here)
```

```
## here() starts at C:/Users/ltwardochleb/Documents/Sac_Delta_analyses/Delta_outflow/Delta_outflow
```

```
library(conflicted)
```

```
## Warning: package 'conflicted' was built under R version 4.2.3
```

```
conflict_prefer("select", "dplyr")
```

```
## [conflicted] Will prefer dplyr::select over any other package.
```

```
conflict_prefer("filter", "dplyr")
```

```
## [conflicted] Will prefer dplyr::filter over any other package.
```

```
# Define file path in the repository for figure and table outputs
fp_output <- here("figs_tables")

# Define file path in the repository for dayflow data
fp_outflow <- here("data")
```

2. Import and Prepare Data from Dayflow

```
# Import data
outflow_1996 <- read_csv(file.path(fp_outflow, "dayflow-results-1984-1996.csv"))
```

```
## Rows: 4749 Columns: 29
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): Y, Date
## dbl (27): Year, Month, SAC, YOLO, CSMR, MOKE, MISC, SJR, EAST, TOT, CCC, SWP...
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
outflow_2020 <- read_csv(file.path(fp_outflow, "dayflow-results-1997-2020.csv"))
```

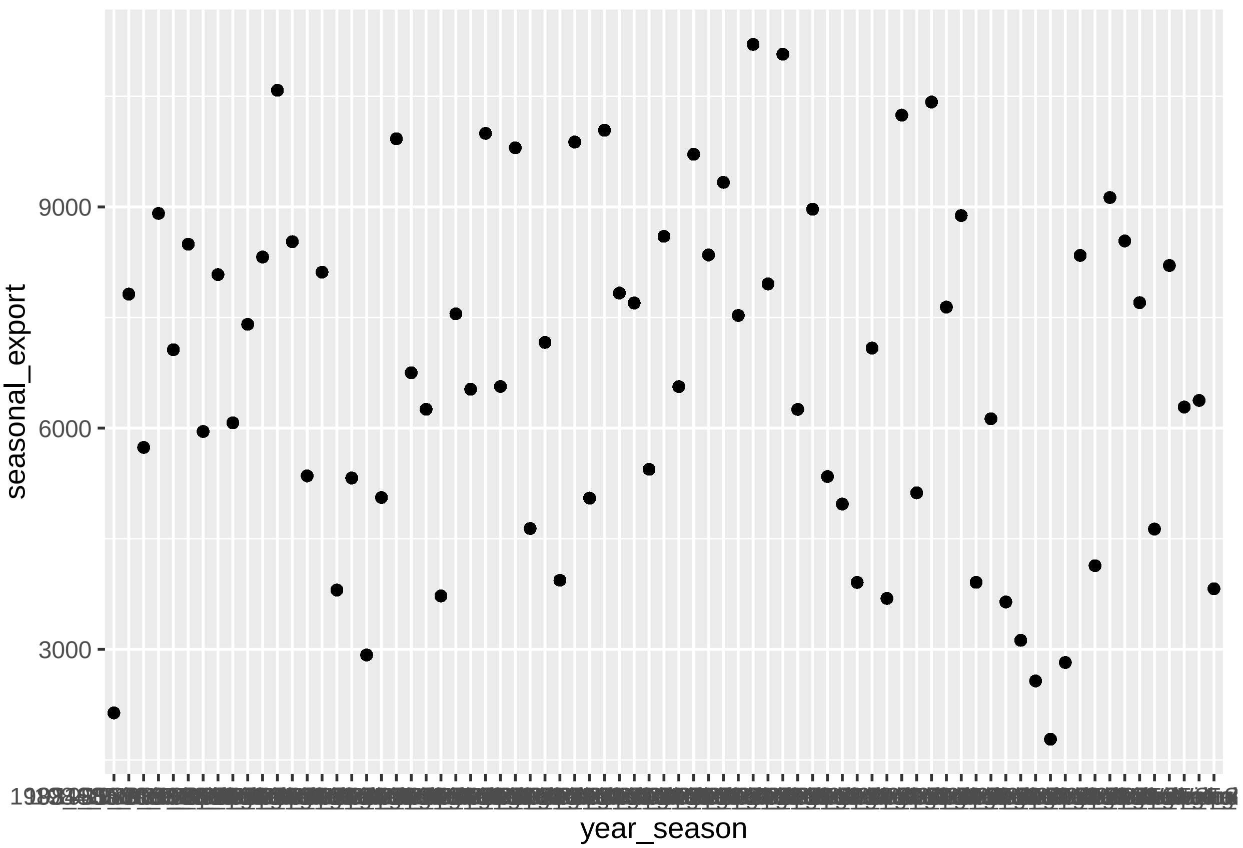
```
## Rows: 8766 Columns: 29
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): Date
## dbl (28): Year, Month, SAC, YOLO, CSMR, MOKE, MISC, SJR, EAST, TOT, CCC, SWP...
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
#Combine and manipulate datasets
outflow<-full_join(outflow_1996, outflow_2020)%>%
  mutate(EXPORTS=ifelse(is.na(EXPORT), EXPORTS, EXPORT))%>% #combine export columns with different name
  #select(-c(DIVER, EFFEC, EFFDIV))%>% #remove unneeded columns
  mutate(Season=case_when(Month%in%1:6 ~ "Winter_Spring", # Create seasonal variables
                          Month%in%7:12 ~ "Summer_Fall",
                          TRUE ~ NA_character_))%>%
  group_by(Year, Season)%>% #create seasonal summaries of delta outflow, exports
  mutate(seasonal_export=mean(EXPORTS))%>%
  mutate(seasonal_outflow=mean(OUT))%>%
  unite('year_season', c(Year, Season), remove=FALSE)%>% #create new variable defining year and season
  mutate(upper_bookend=case_when(seasonal_outflow>=47000 ~ 1,
                                 seasonal_outflow<47000 ~ 0))%>%#identify year_season above 47,000 cfs
  mutate(lower_bookend=case_when(seasonal_outflow>=29200 ~ 1,
                                 seasonal_outflow<29200 ~ 0))#identify year_season above 29,200 cfs
```

```
## Joining with `by = join_by(Year, Month, Date, SAC, YOLO, CSMR, MOKE, MISC, SJR,
## EAST, TOT, CCC, SWP, CVP, NBAQ, GCD, PREC, MISDV, CD, XGEO, WEST, RIO, OUT,
## EXPIN)`
```

3. Explore outflow, exports over time

```
#visually assess stationarity of exports over last 20-30 years
outflow%>%ggplot()+geom_point(aes(x=year_season, y=seasonal_export))
```



```
#assess stationarity of exports using Dickey-Fuller test
library(tseries)
```

```
## Warning: package 'tseries' was built under R version 4.2.3
```

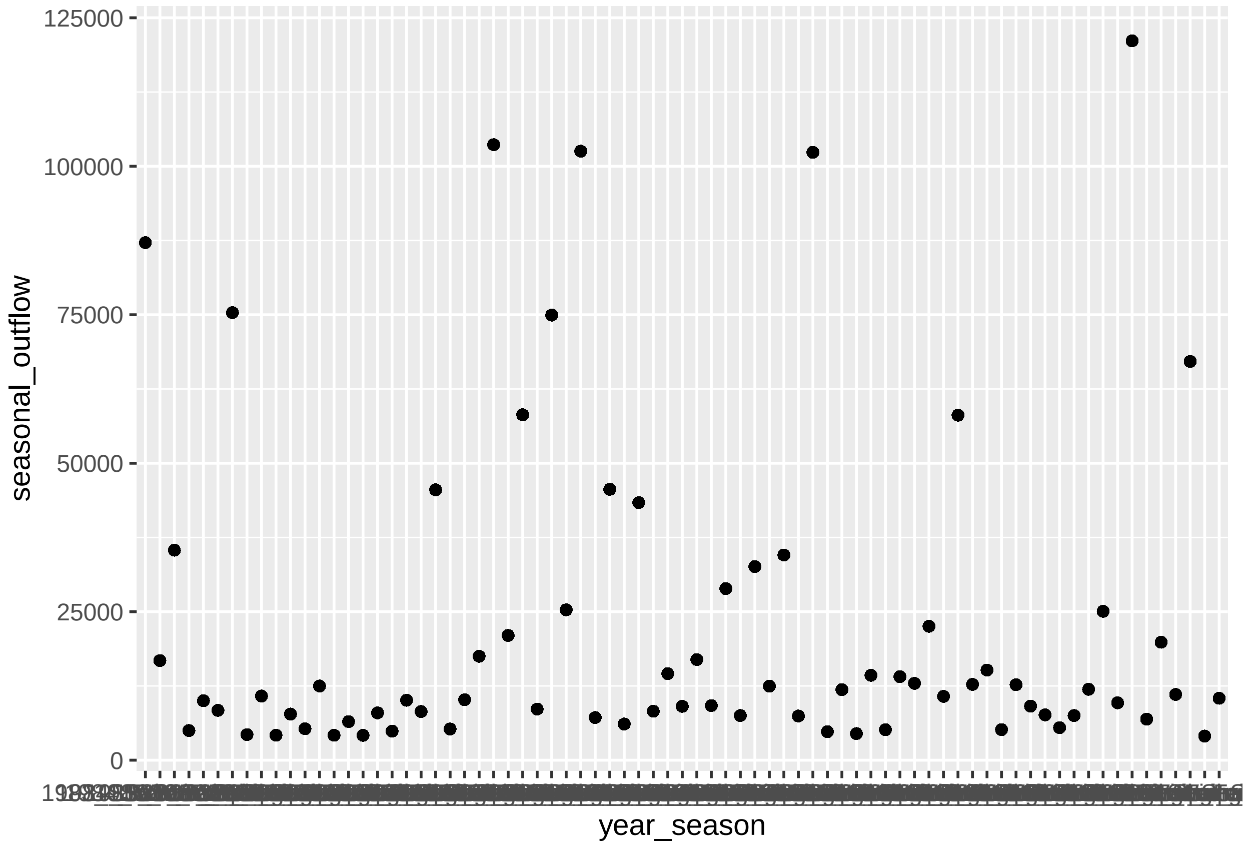
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
adf.test(outflow$EXPORTS) #exports time series is stationary
```

```
## Warning in adf.test(outflow$EXPORTS): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  outflow$EXPORTS
## Dickey-Fuller = -10.419, Lag order = 23, p-value = 0.01
## alternative hypothesis: stationary
```

```
#examine stationarity of delta outflows
outflow%>%ggplot()+geom_point(aes(x=year_season, y=seasonal_outflow))
```

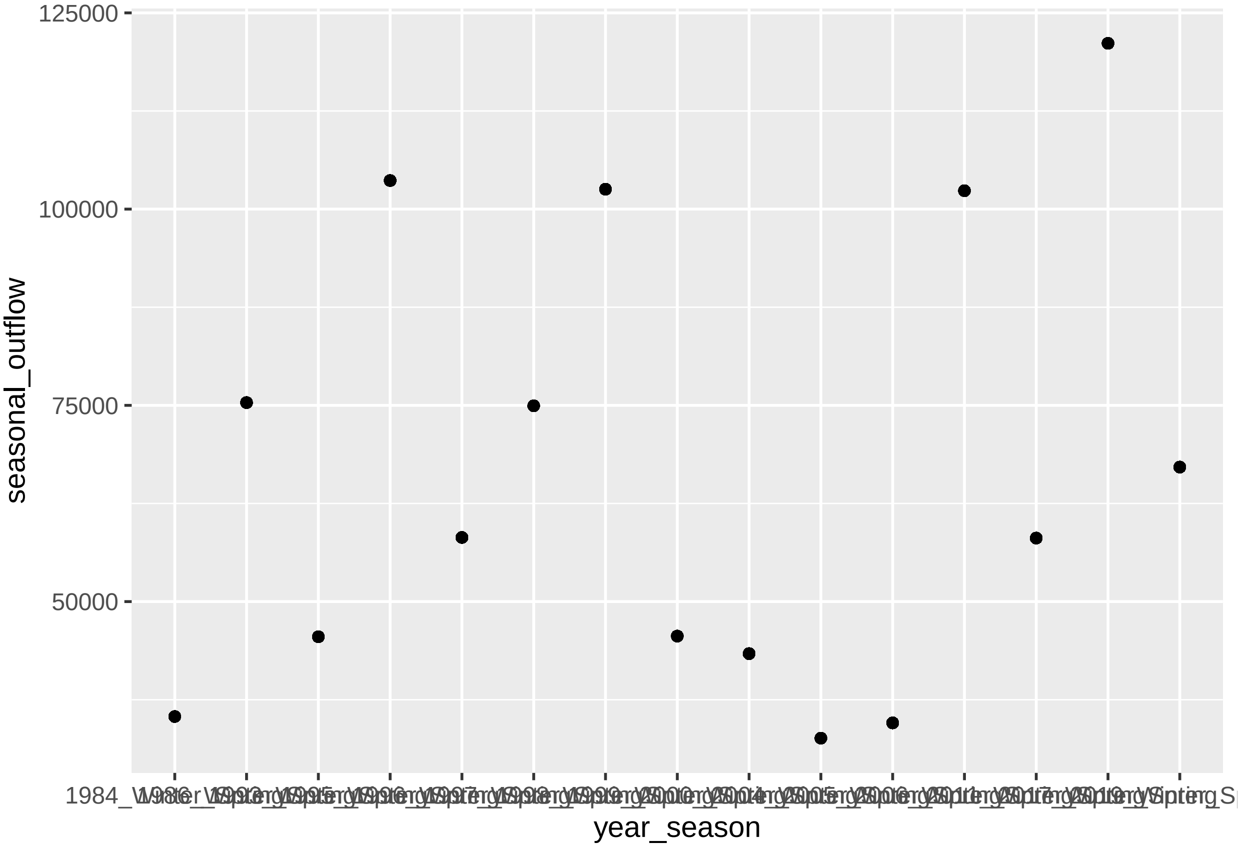


```
adf.test(outflow$OUT) #Delta outflow is also stationary
```

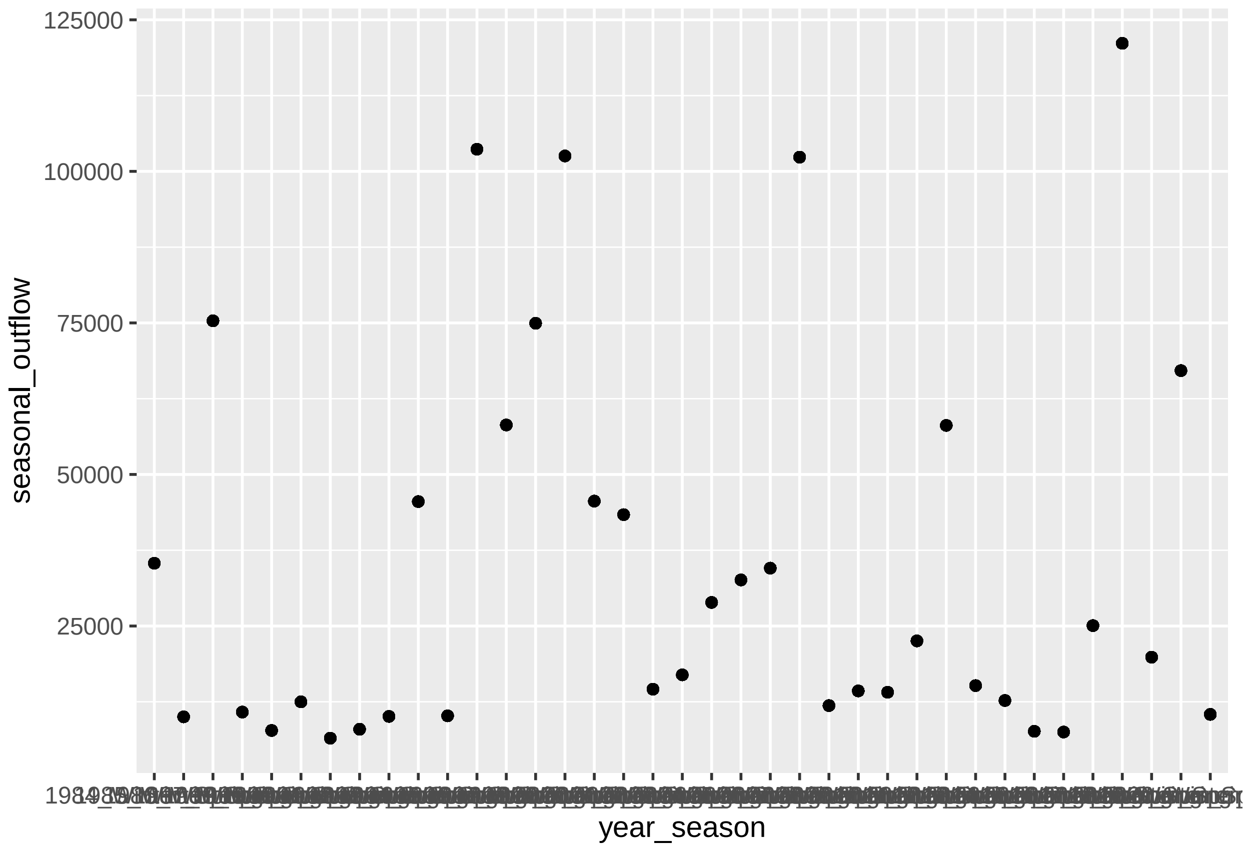
```
## Warning in adf.test(outflow$OUT): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  outflow$OUT
## Dickey-Fuller = -10.497, Lag order = 23, p-value = 0.01
## alternative hypothesis: stationary
```

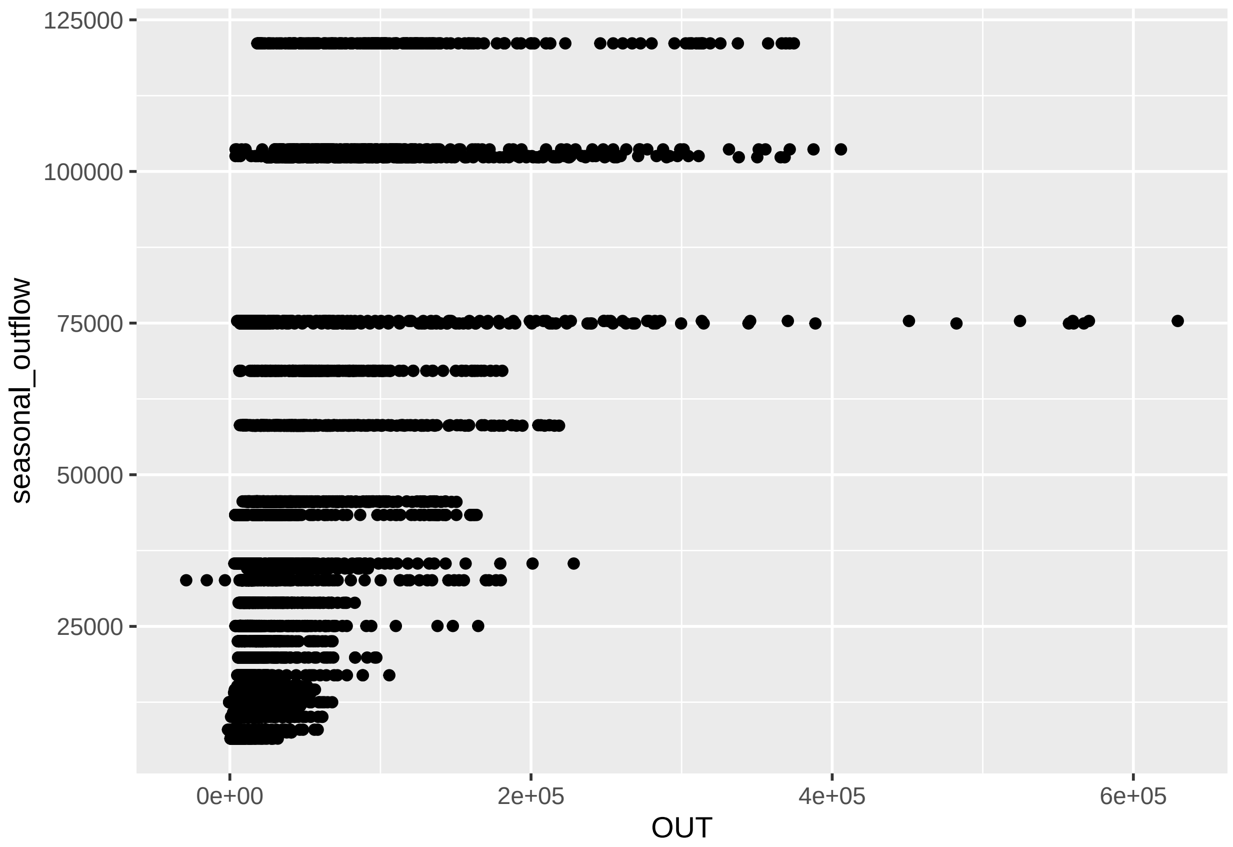
```
#examine data from seasons with outflow thresholds above 29,200 cfs and 47,000 cfs as bookends in winte
outflow%>%filter(Season=="Winter_Spring")%>%filter(seasonal_outflow>=29200&47000)%>%
  ggplot()+geom_point(aes(x=year_season, y=seasonal_outflow))
```



```
#examine all years in winter and spring
outflow%>%filter(Season=="Winter_Spring")%>%ggplot()+geom_point(aes(x=year_season, y=seasonal_outflow))
```



```
outflow%>%filter(Season=="Winter_Spring")%>%ggplot()+geom_point(aes(x=OUT, y=seasonal_outflow))
```



4. Predict seasonal outflow using daily average outflow

```
#logistic regression for lower bookend
spring_outflow<-outflow%>%filter(Season=="Winter_Spring")

lower_bookend<-glm(lower_bookend~OUT, family="binomial", data=spring_outflow)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
lower_bookend
```

```
##
## Call:  glm(formula = lower_bookend ~ OUT, family = "binomial", data = spring_outflow)
##
## Coefficients:
## (Intercept)          OUT
##  -2.246e+00    7.177e-05
##
## Degrees of Freedom: 6706 Total (i.e. Null);  6705 Residual
## Null Deviance:       9056
## Residual Deviance: 5871  AIC: 5875
```

```
summary(lower_bookend)
```

```
##
## Call:
## glm(formula = lower_bookend ~ OUT, family = "binomial", data = spring_outflow)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.3794  -0.6371  -0.5260   0.4637   2.9459
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.246e+00  5.386e-02  -41.70   <2e-16 ***
## OUT          7.177e-05  2.052e-06   34.98   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9056.3  on 6706  degrees of freedom
## Residual deviance: 5871.3  on 6705  degrees of freedom
## AIC: 5875.3
##
## Number of Fisher Scoring iterations: 6
```

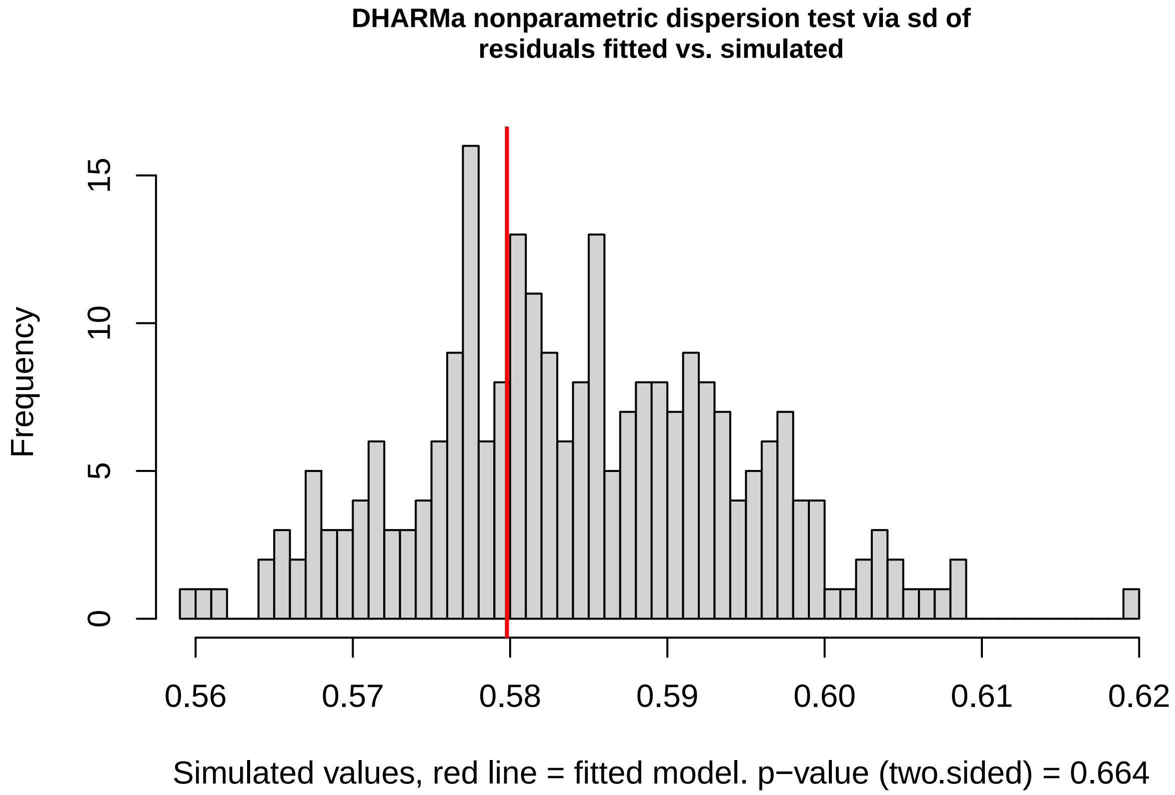
```
#assess model fit

library(DHARMa)
```

```
## Warning: package 'DHARMa' was built under R version 4.2.3
```

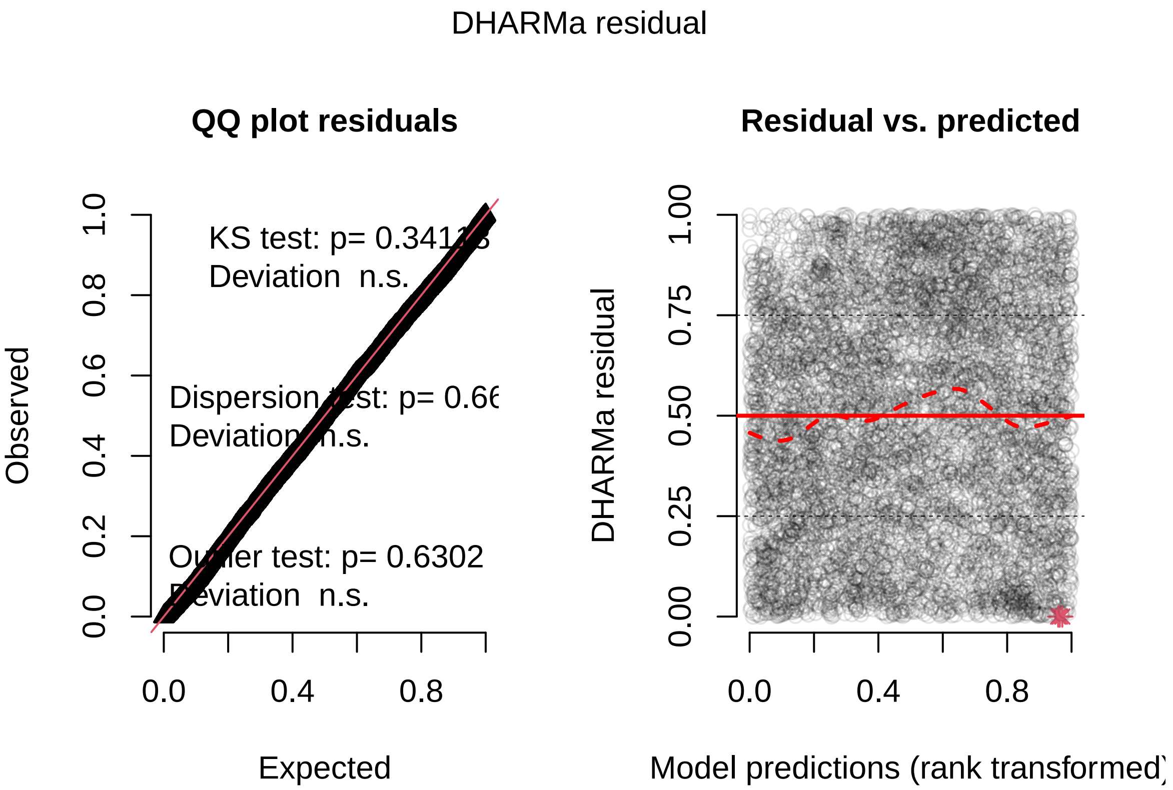
```
## This is DHARMa 0.4.6. For overview type '?DHARMa'. For recent changes, type news(package = 'DHARMa')
```

```
#see: https://cran.r-project.org/web/packages/DHARMa/vignettes/DHARMa.html
testDispersion(lower_bookend)
```

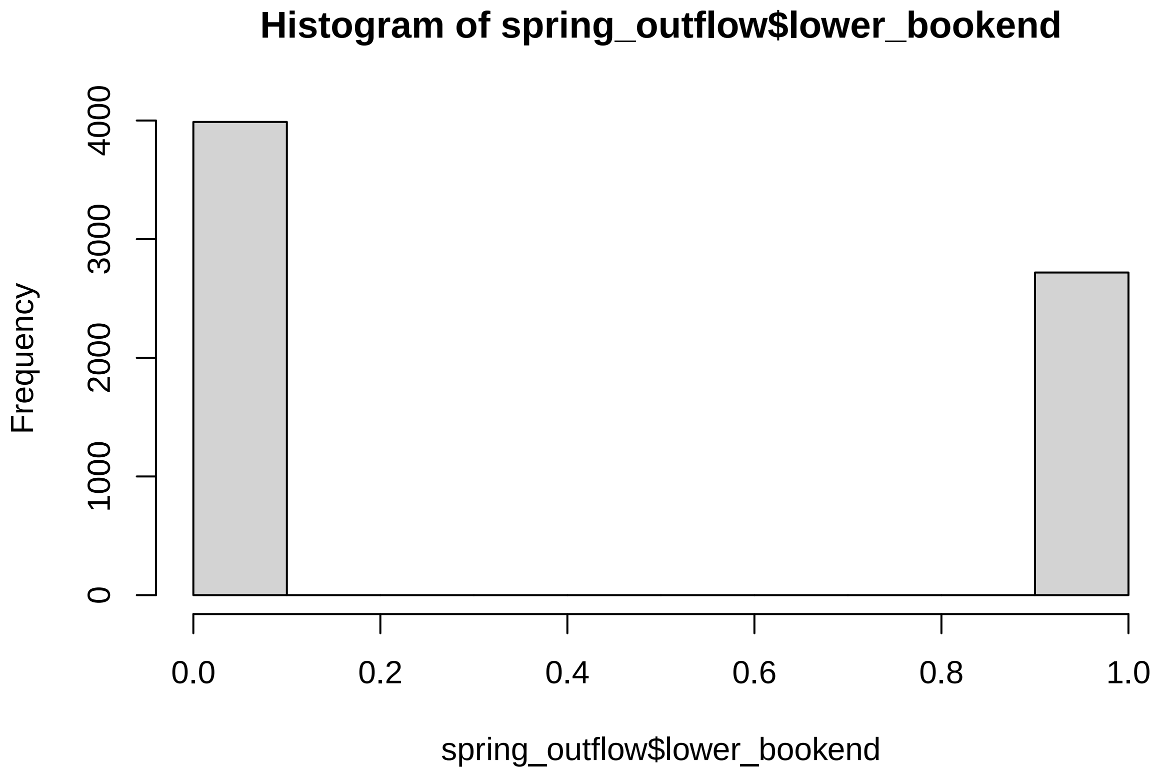


```
##
##  DHARMa nonparametric dispersion test via sd of residuals fitted vs.
##  simulated
##
## data:  simulationOutput
## dispersion = 0.99206, p-value = 0.664
## alternative hypothesis: two.sided
```

```
simres_lower <- simulateResiduals(lower_bookend)
plot(simres_lower) #shows a few potential outliers
```



```
hist(spring_outflow$lower_bookend)
```



```
#what are the outflow values that get you 29,200 cfs of outflow?
LD50=-lower_bookend$coefficients[1]/lower_bookend$coefficients[2]
LD75= (log(.75/(1-.75))-lower_bookend$coefficients[1])/lower_bookend$coefficients[2]
LD99= (log(.99/(1-.99))-lower_bookend$coefficients[1])/lower_bookend$coefficients[2]

LD50
```

```
## (Intercept)
##    31291.91
```

```
LD75
```

```
## (Intercept)
##    46598.31
```

```
LD99
```

```
## (Intercept)
##    95313.35
```

```
#define function to calculate LD values
LDfunc <- function(x, y, z){
  (log(x/(1-x))-y)/z
}
```

```r
#calculate LD values for 1 to 99 for lower bookend
LD<-list(seq(from=0.10, to=0.99, length.out=90))
LDs_lower_bookend<-mapply(LDfunc, x=LD, y=lower_bookend$coefficients[1], z=lower_bookend$coefficients[2

#logistic regression for upper bookend
upper_bookend<-glm(upper_bookend~OUT, family="binomial", data=spring_outflow)
upper_bookend
```

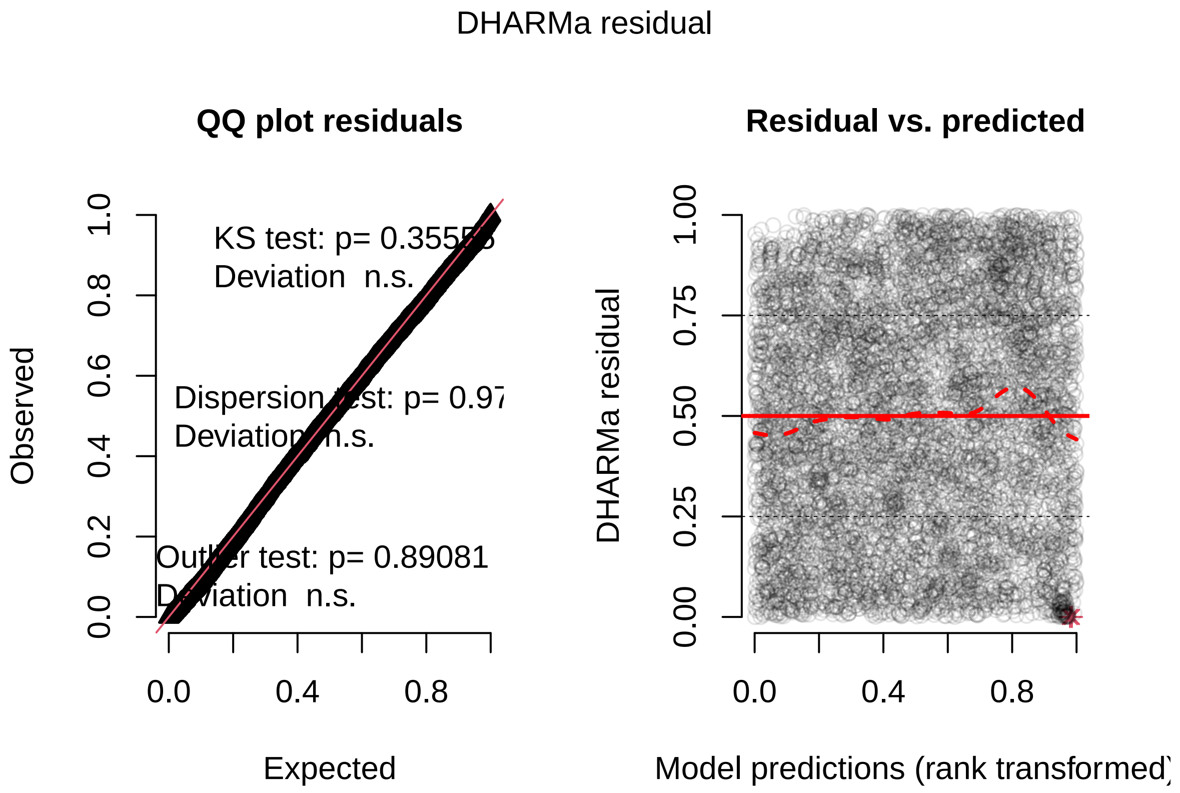
```
##
## Call:  glm(formula = upper_bookend ~ OUT, family = "binomial", data = spring_outflow)
##
## Coefficients:
## (Intercept)          OUT
##  -2.641e+00    4.004e-05
##
## Degrees of Freedom: 6706 Total (i.e. Null);  6705 Residual
## Null Deviance:       7439
## Residual Deviance: 5160  AIC: 5164
```

```r
summary(upper_bookend)
```

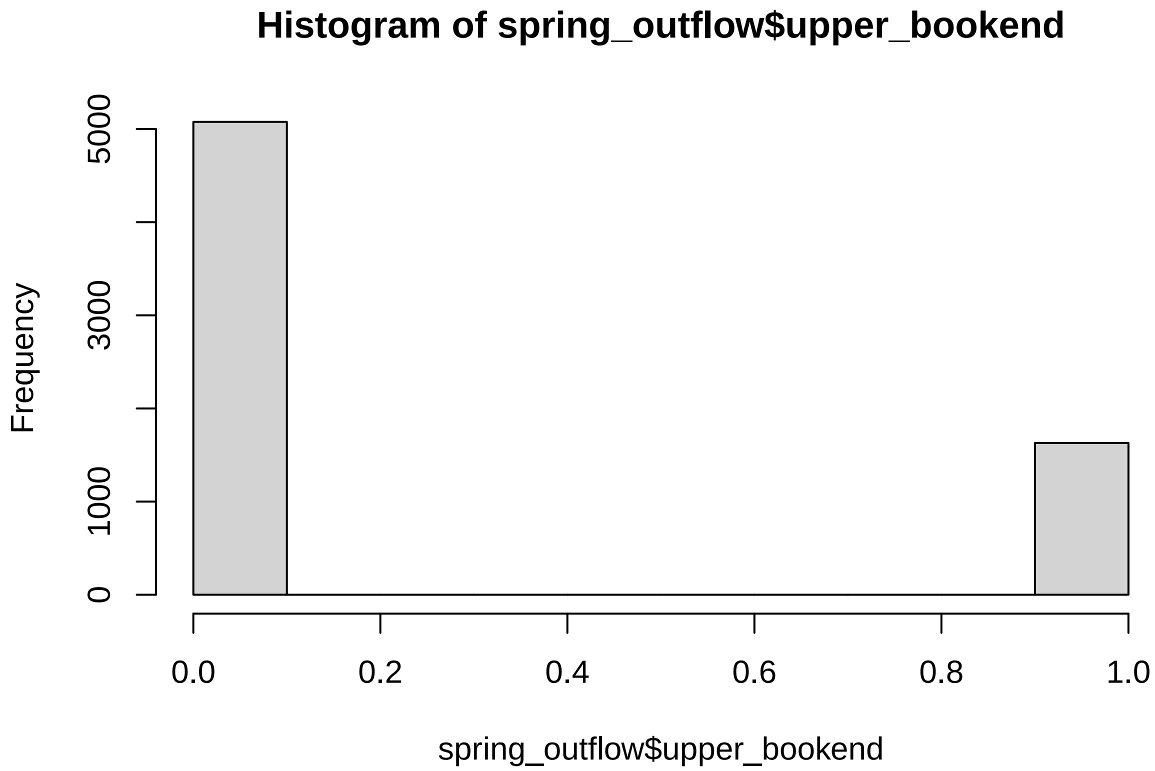
```
##
## Call:
## glm(formula = upper_bookend ~ OUT, family = "binomial", data = spring_outflow)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.6064  -0.5131  -0.4338  -0.3821   2.2675
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.641e+00  5.486e-02  -48.15   <2e-16 ***
## OUT          4.003e-05  1.178e-06   33.99   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 7438.7  on 6706  degrees of freedom
## Residual deviance: 5159.6  on 6705  degrees of freedom
## AIC: 5163.6
##
## Number of Fisher Scoring iterations: 5
```

```r
#assess model fit

simres_upper <- simulateResiduals(upper_bookend)
plot(simres_upper) #potentially one outlier
```



```
hist(spring_outflow$upper_bookend)
```



```
#calculate LD values for 1 to 99 for upper bookend
LD50upper=-upper_bookend$coefficients[1]/upper_bookend$coefficients[2]
LDs_upper_bookend<-mapply(LDfunc, x=LD, y=upper_bookend$coefficients[1], z=upper_bookend$coefficients[2

LD50upper
```

```
## (Intercept)
##    65973.43
```

```
LDs_upper_bookend<-mapply(LDfunc, x=LD, y=upper_bookend$coefficients[1], z=upper_bookend$coefficients[2

#bind the lists together into a dataframe of upper and lower bookend LDs
LD_upper_lower<-as.data.frame(cbind(LDs_lower_bookend, LDs_upper_bookend, unlist(LD)))

colnames(LD_upper_lower)<-c("lower_bookend", "upper_bookend", "percent")
```

5. Prepare figures of outflow and LDs

```
#plot modeled data- lower bookend
ggplot(spring_outflow, aes(x=OUT, y=lower_bookend)) + geom_point() +
stat_smooth(method="glm", color="blue", se=FALSE,
            method.args = list(family=binomial))
```

```
## `geom_smooth()` using formula = 'y ~ x'

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

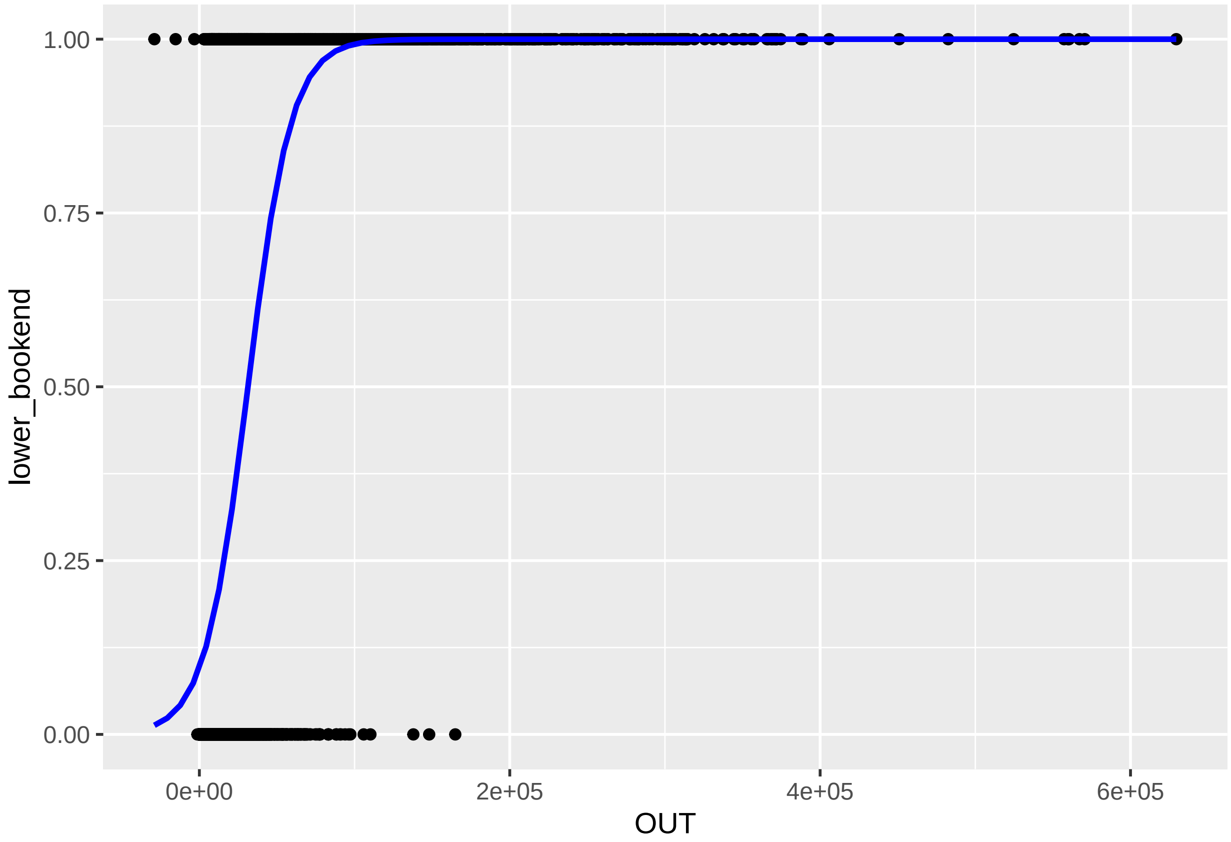
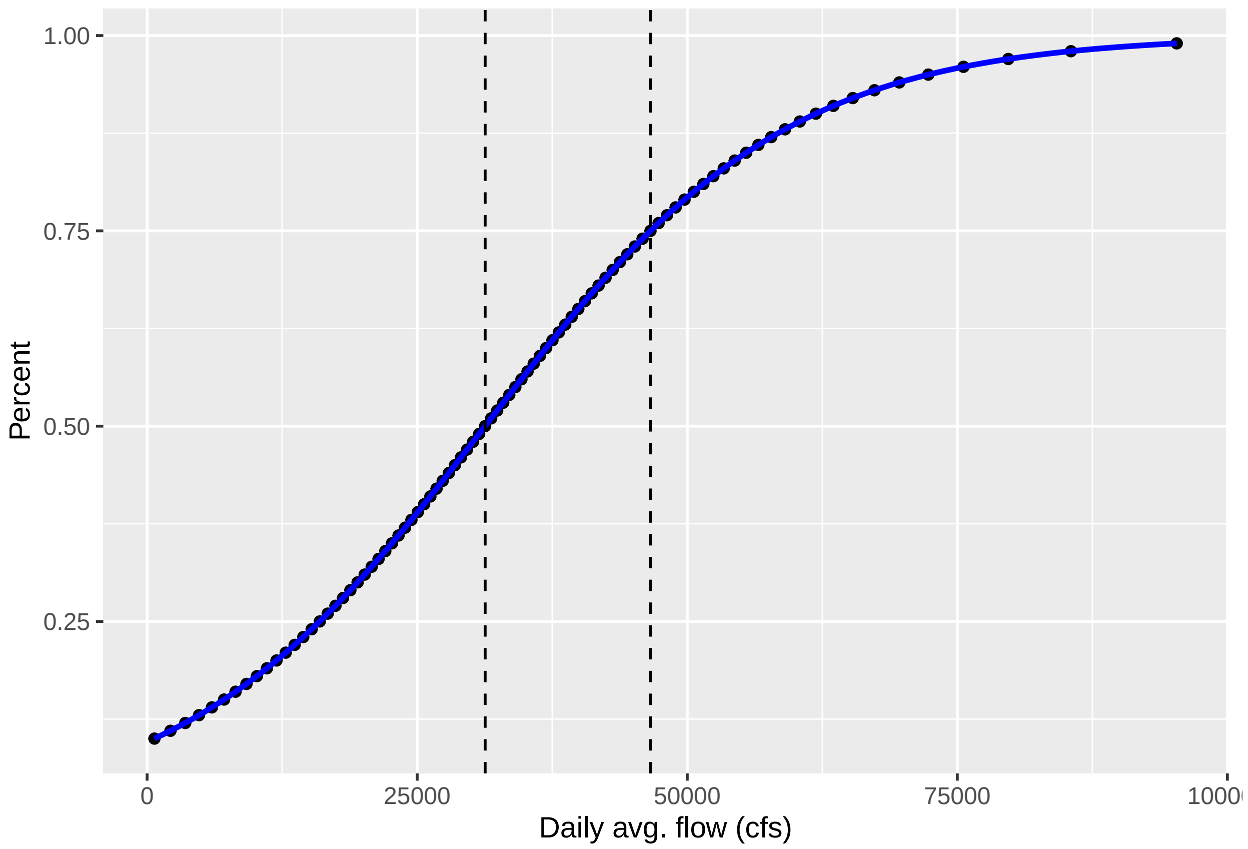


Figure 1: Logistic regression for 29,200 cfs Jan-June outflow

```
#plot LD data- lower bookend
ggplot(LD_upper_lower, aes(x=lower_bookend, y=percent)) + geom_point() +
  stat_smooth(method="glm", color="blue", se=FALSE,
              method.args = list(family=binomial))+
  geom_vline(xintercept=31291.9096,linetype=2)+
  geom_vline(xintercept=46598.3098,linetype=2)+
  ylab("Percent")+xlab("Daily avg. flow (cfs)")
```

```
## `geom_smooth()` using formula = 'y ~ x'

## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```



```r
#plot modeled data- upper bookend
ggplot(spring_outflow, aes(x=OUT, y=upper_bookend)) + geom_point() +
  stat_smooth(method="glm", color="blue", se=FALSE,
              method.args = list(family=binomial))
```

```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```

```r
#plot LD data- upper bookend
ggplot(LD_upper_lower, aes(x=upper_bookend, y=percent)) + geom_point() +
  stat_smooth(method="glm", color="blue", se=FALSE,
              method.args = list(family=binomial))+
  geom_vline(xintercept=65973.43,linetype=2)+
  geom_vline(xintercept=93414.52,linetype=2)+
  ylab("Percent")+xlab("Daily avg. flow (cfs)")
```

```
## ‘geom_smooth()‘ using formula = ’y ~ x’
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial glm!
```

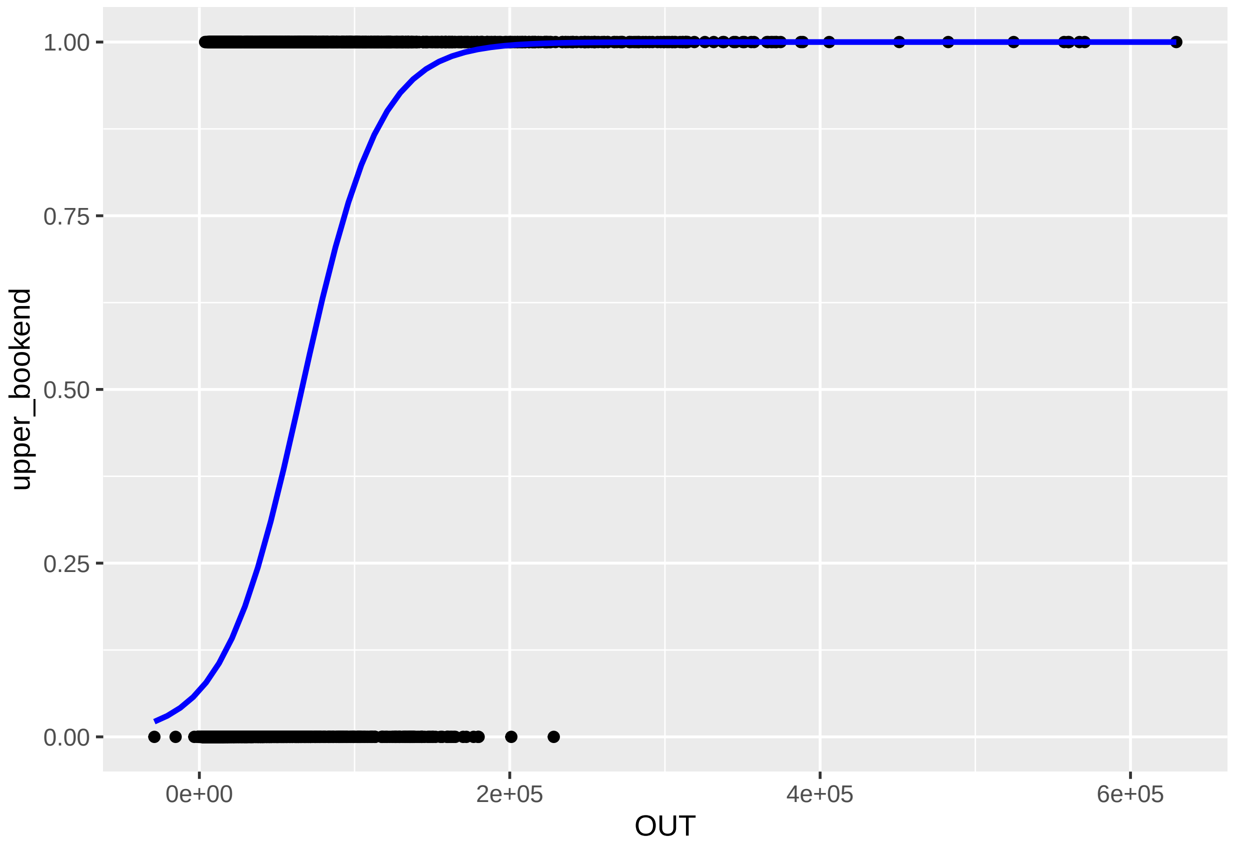
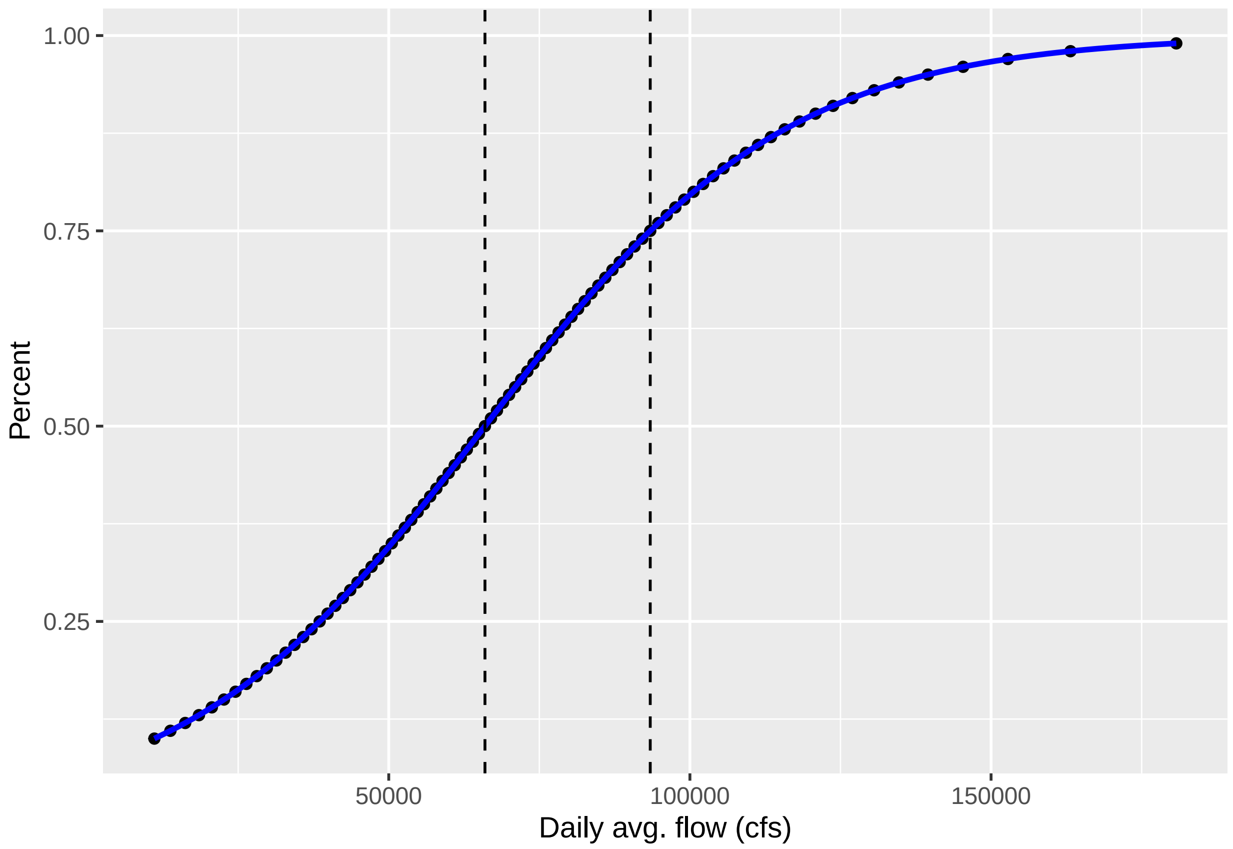


Figure 2: Logistic regression for 47,000 cfs Jan-June outflow


1.00
0.75
0.50
0.25
Percent
50000
100000
150000
Daily avg. flow (cfs)