PH240C Assignment 2

Xiangyu Hu

October 15, 2014

Problem 1.

(a) Write the log likelihood and plot it for a sensible choice of values

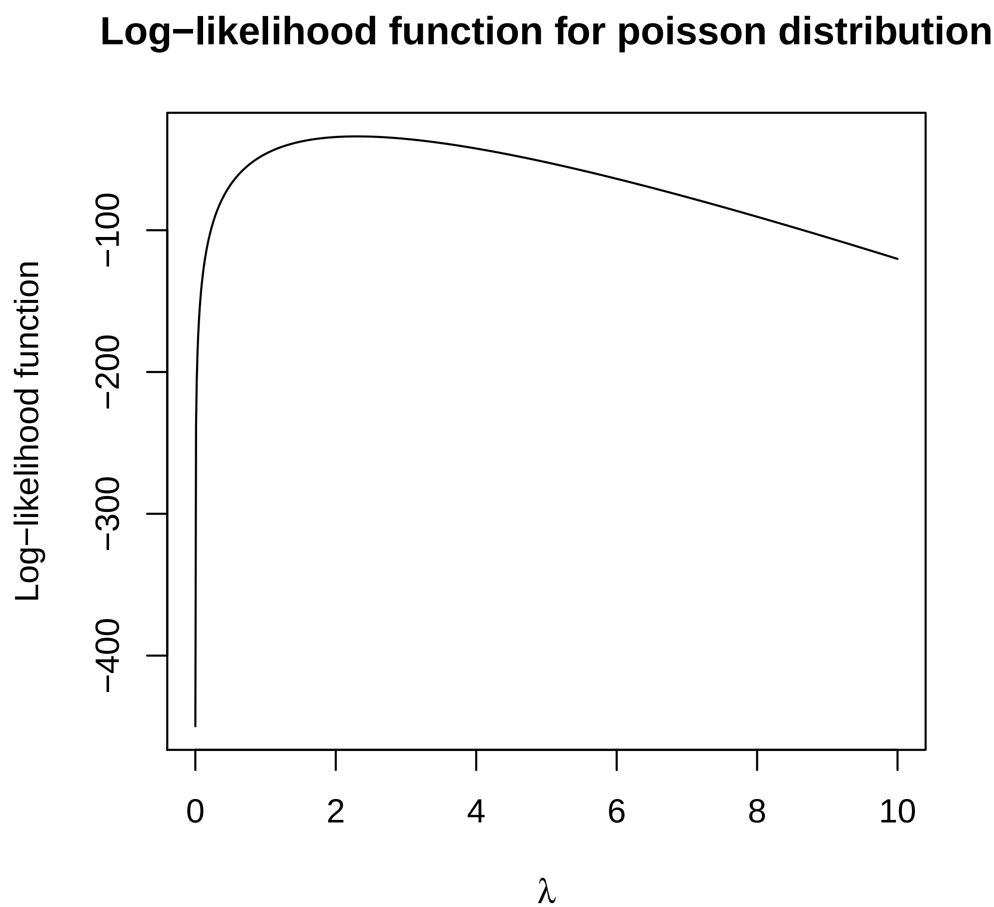
Likelihood function:

$$
\begin{aligned}
L_n(\lambda) &= \prod_{i=1}^{n} \frac{\lambda^{x_i}}{x_i!} e^{-\lambda} \\
&= \frac{\lambda^{\sum_{i=1}^{n} x_i} e^{-n\lambda}}{\prod_{i=1}^{n} x_i!}
\end{aligned}
$$

Log-Likelihood function:

$$
l_n(\lambda) = (\sum_{i=1}^{n} x_i) \log \lambda - n\lambda - \sum_{i=1}^{n} \log x_i!
$$

```r
data1 = c(1,2,6,2,3,3,2,3,2,1,5,2,2,4,2,0,2,1,2,1)

lambda = seq(0.0001,10, length.out = 1000)

loglik = function(lambda,data){
  n = length(data)
  sumX = sum(data)
  sumLogFac = sum(log(factorial(data)))
  loglik = sumX*log(lambda) - n*lambda - sumLogFac
  return(loglik)
}
ll = loglik(lambda, data1)
plot(ll ~ lambda, main="Log-likelihood function for poisson distribution", xlab=expression(lambda),
     ylab = "Log-likelihood function", type ="l")
```



(b)

Find the MLE analytically:
Score function:

$$\frac{dl_n(\lambda)}{d\lambda} = \frac{\sum_{i=1}^{n}}{\lambda} - n$$

Set it equal to 0:

$$\frac{\sum_{i=1}^{n}}{\lambda} - n = 0$$

And, solve for λ :

$$\hat{\lambda} = \frac{\sum_{i=1}^{n} x_i}{n} = \bar{x}$$

Verify if we achieve maximum at the estimate using second derivative:

$\frac{dl_n^2(\lambda)}{d^2\lambda} = -\frac{\sum_{i=1}^{n} x_i}{\lambda^2} < 0$ since both the numerator and the denominator are positive.

Therefore, the MLE of λ is $\bar{x}$

```
# analytically:
mean(data1)

## [1] 2.3

# numerically:
optimize(loglik, interval = c(0,4), data1, maximum =TRUE)
```

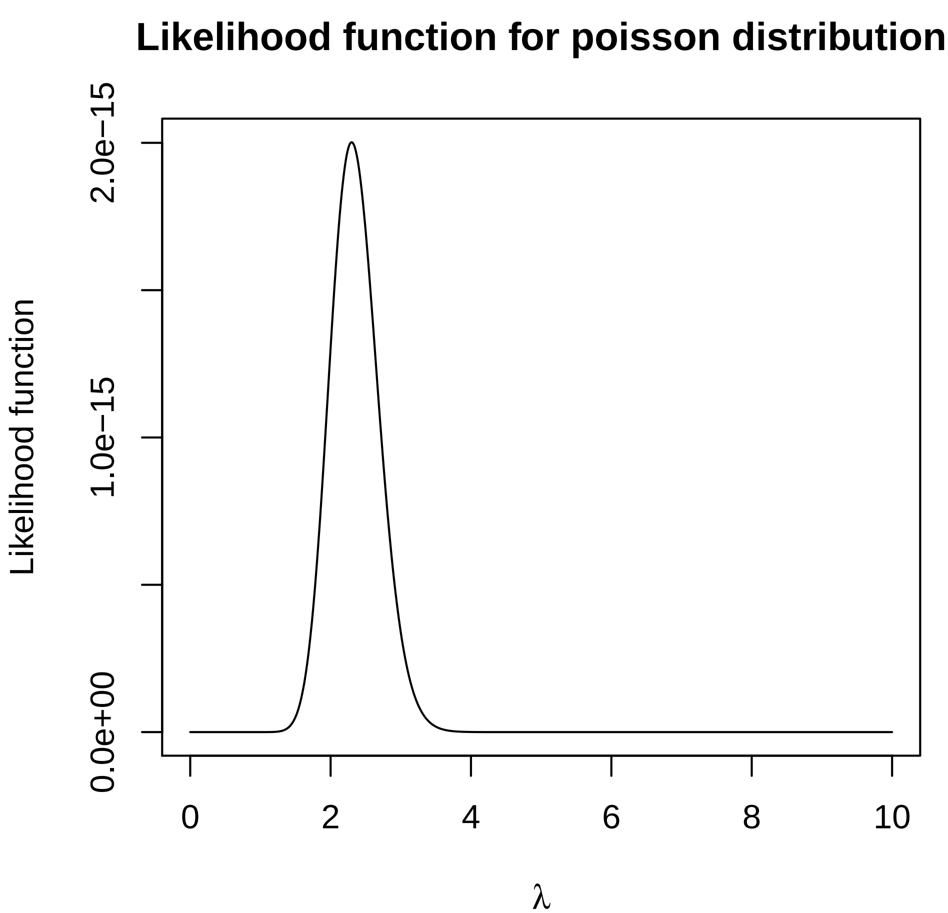
```
## $maximum
## [1] 2.3
##
## $objective
## [1] -33.84
```

From above, we can see that the results of finding MLE of λ both analytically and numerically are the same.

(c)

$\theta = \log(\lambda)$, so $\lambda = e^{\theta}$. If we plug-in the e^{θ} to the likihood function and log-likehood function above, we will then have $L_n(\theta)$, and $l_n(\theta)$ correspondingly. The parameter space for θ: $\theta \in [-\infty, \infty]$

```
lik = function(data, lambda){
  sumX = sum(data)
  n = length(data)
  facProd = prod(factorial(data))
  lik = (lambda^sumX * exp(-n*lambda)) / facProd
  return(lik)
}
l = lik(data1,lambda)
plot(l ~ lambda, main="Likelihood function for poisson distribution", xlab=expression(lambda),
     ylab = "Likelihood function", type ="l")
```

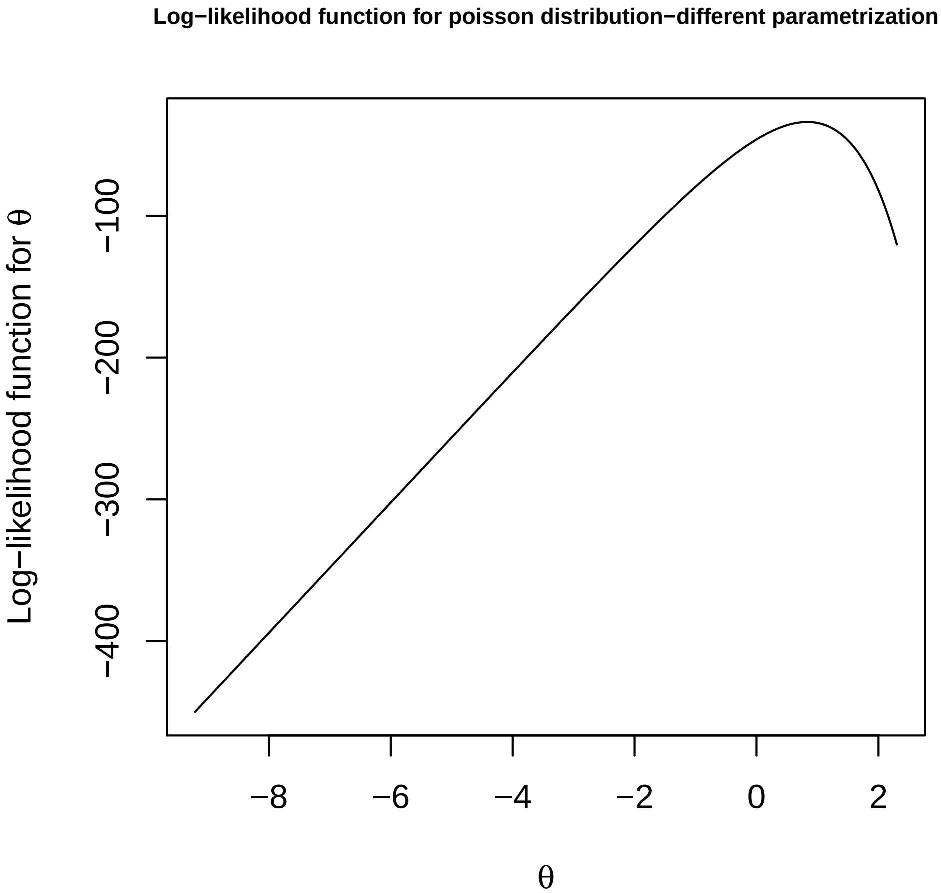


(d)

Log-Likelihood function w.r.t θ:

$$l_n(\theta) = (\sum_{i=1}^{n} x_i)\theta - ne^{\theta} - \sum_{i=1}^{n} \log x_i!$$

```r
# MLE of theta numerically:
loglik_t = function(theta, data){
  sumX = sum(data)
  n = length(data)
  sumLogFac = sum(log(factorial(data)))
  loglik_t = sumX*theta - n*exp(theta) - sumLogFac
  return(loglik_t)
}
theta = log(lambda)
ll_t = loglik_t(theta, data1)
plot(ll_t~theta, xlab=expression(theta),
     ylab = expression(paste("Log-likelihood function for ",theta)), type ="l")
title(main="Log-likelihood function for poisson distribution-different parametrization", cex.main=0.7)
```




```r
optimize(loglik_t, interval = c(-1,2), data1, maximum =TRUE)

## $maximum
## [1] 0.8329
##
## $objective
## [1] -33.84

log(2.3)

## [1] 0.8329
```

From the above, we can see that numerically, $\theta_n = \log(\lambda_n)$

Problem 2.

(a) Write the log likelihood and plot it for a sensible choice of values

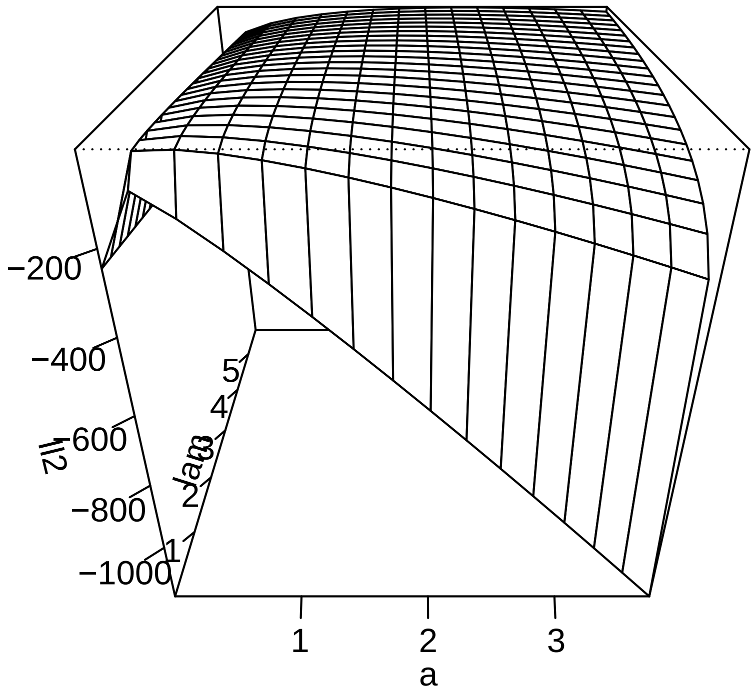
Likelihood function:

$$L_n(\alpha, \lambda) = \prod_{i=1}^{n} \frac{1}{\Gamma(\alpha)} \lambda^{\alpha} y_i^{\alpha-1} e^{-\lambda y_i}$$

Log-likelihood function:

$$l_n(\alpha, \lambda) = -n \log \Gamma(\alpha) + n\alpha \log(\lambda) + (\alpha - 1) \sum_{i=1}^{n} \log y_i - \lambda \sum_{i=1}^{n} y_i$$

```
data2 = c(0.15812, 0.30070, 0.48016, 0.49813, 0.20042, 0.26716, 0.80124,
          0.10914, 0.57169, 0.83686, 1.57027, 0.10458, 0.58490, 1.14454,
          0.61595, 0.28155, 0.13236, 0.36252, 0.08614, 0.27907, 0.46010,
          0.03824, 0.76581, 0.30369, 0.42404, 0.57530, 0.26987, 0.22416,
          0.07673, 1.09659)

loglik2 = function(a,lam, data){
  n = length(data)
  # gammar of alpha
  ga = factorial(a-1)
  sumY = sum(data)
  sumLog = sum(log(data))
  loglik2 = -n*log(ga) + n*a*log(lam) + (a-1)*sumLog - lam*sumY
  return(loglik2)
}

a = seq(0.0001,4,by = 0.25)
lam = seq(0.0001,6,by = 0.25)

pars = c(a=a,lam=lam)
ll2 = outer(a,lam,loglik2,data2)

persp(a,lam,ll2, ticktype = "detail", phi = 30)
```



(b)

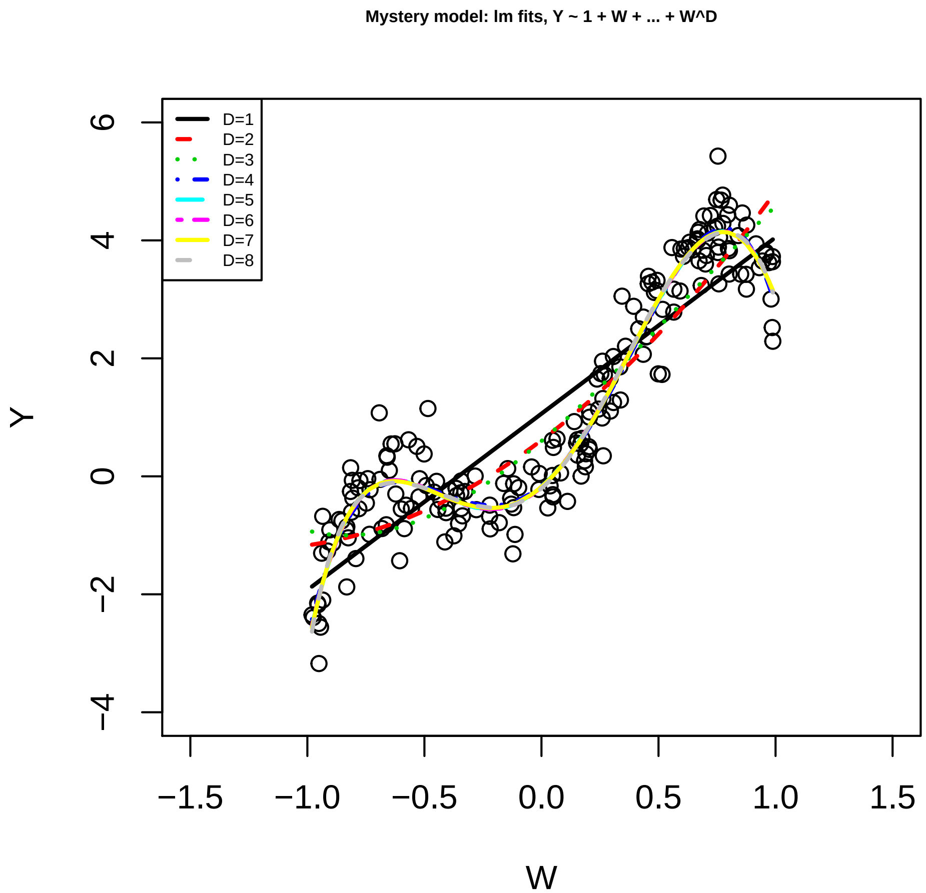
```
negloglik2 = function(pars, data){
  a = pars[1]
  lam = pars[2]
  n = length(data)
  # gammar of alpha
  ga = factorial(a-1)
  sumY = sum(data)
  sumLog = sum(log(data))
  loglik2 = -(-n*log(ga) + n*a*log(lam) + (a-1)*sumLog - lam*sumY)
  return(loglik2)
}

optim(c(0.001,0.001), negloglik2, data=data2)
```

```
## $par
## [1] 1.677 3.694
##
## $value
## [1] 4.235
##
## $counts
## function gradient
##      123       NA
##
## $convergence
## [1] 0
##
```

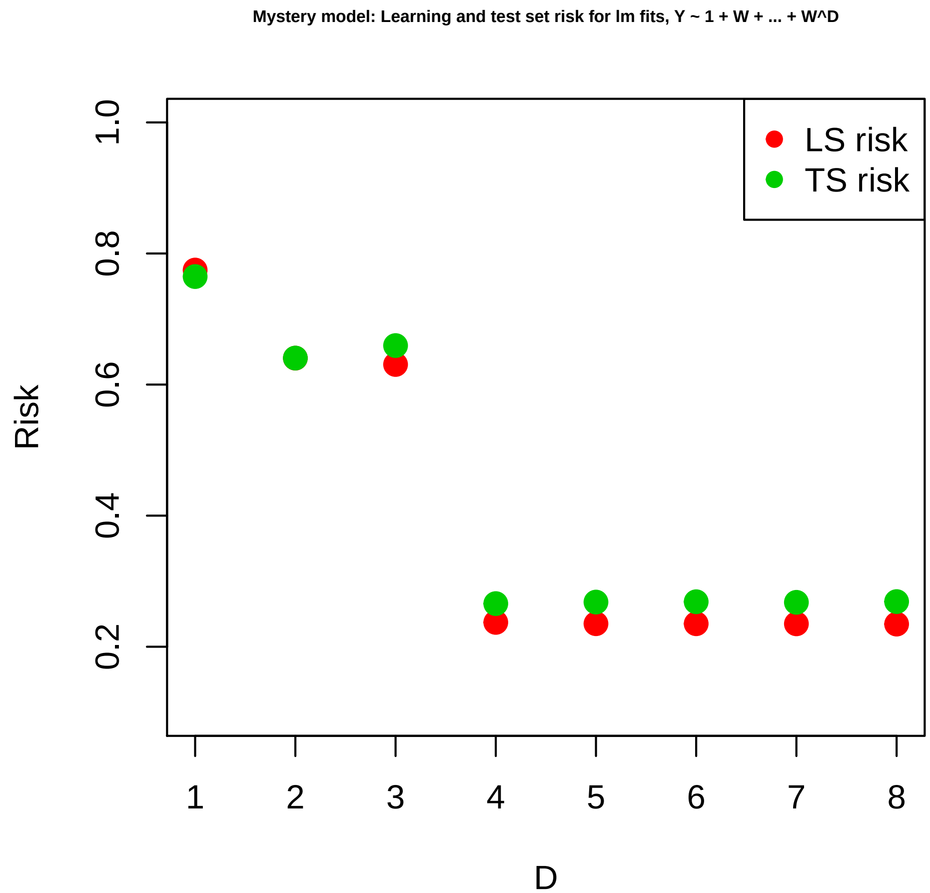
```
## $message
## NULL
```

Problem 3. Answers to part(a) and par(b) are combined below:

```
# reading in the data
ls = read.table("learning.txt")
ts = read.table("test.txt")

###########################
### Polynomial using lm ###
###########################
D = 1:8
# create an empty list to store model for later
fitLm = vector("list", length(D))
# create an empty matrix to store fitted Y for training set
YhatLm.ts = matrix(NA, nrow(ts), length(D))
# create an empty matrix to store fitted Y for learning set
YhatLm.ls = matrix(NA, nrow(ls), length(D))
# create vectors of length(D) to store LS risk and TS risk for each model
LSRiskLm = rep(NA, length(D))
TSRiskLm = rep(NA, length(D))

w.ls = ls$W
w.ts = ts$W
W.ls = NULL
W.ts = rep(1,nrow(ts))
for(d in 1:length(D))
{
  W.ls = cbind(W.ls, w.ls^D[d])
  W.ts = cbind(W.ts, w.ts^D[d] )
  fitLm[[d]] = lm(ls$Y ~ W.ls)
  YhatLm.ts[,d] = W.ts %*% as.matrix(fitLm[[d]]$coef)
  YhatLm.ls[,d] = fitLm[[d]]$fitted
  LSRiskLm[d] = mean(fitLm[[d]]$residuals^2)
  TSRiskLm[d] = mean((YhatLm.ts[,d] - ts$Y)^2, na.rm=TRUE)
}

# lm fits
plot(ls$W, ls$Y, xlim=c(-1.5, 1.5), ylim=c(-4, 6),
     xlab="W", ylab="Y", main="Mystery model: lm fits, Y ~ 1 + W + ... + W^D",
     cex.main = 0.5)
matplot(ls$W, YhatLm.ls, type="l", lty=1:length(D), lwd=2,
        col=1:length(D), add=TRUE)
legend("topleft", paste("D=",D,sep=""), lty=1:length(D),
       lwd=2, col=1:length(D), cex = 0.5)
```

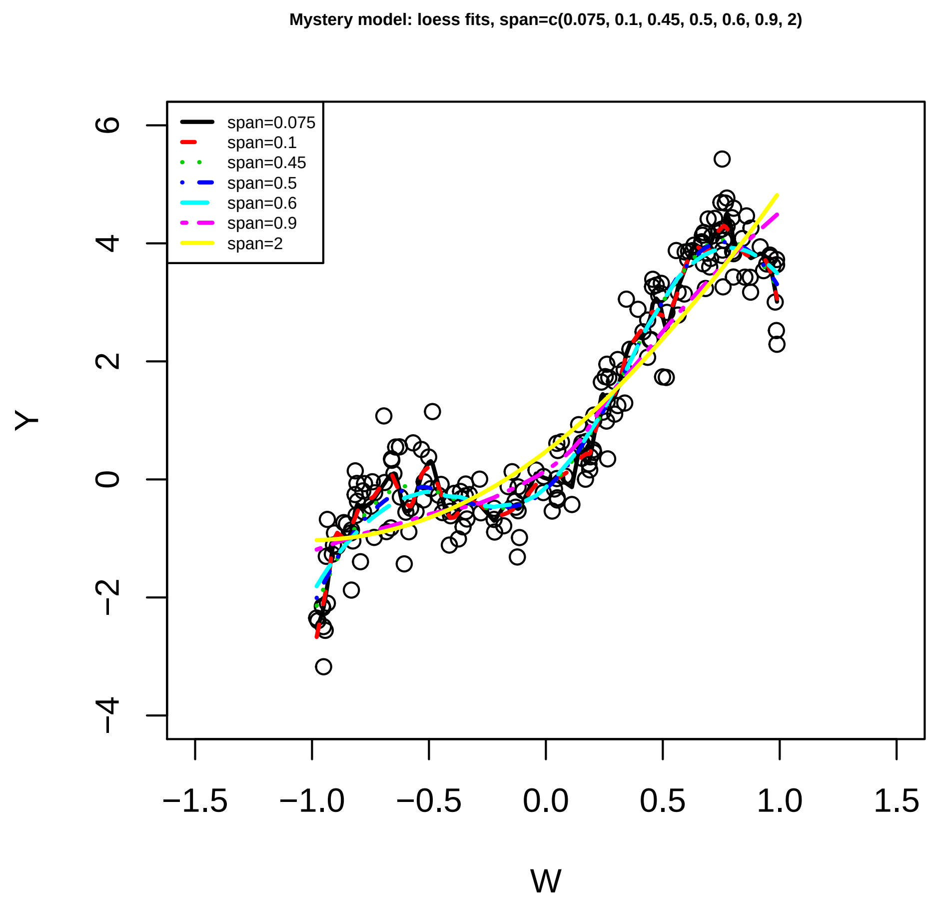



```
## Comment: From the plot, we can see that the ploynomial model with D=4,5,6,7,8 are better fitted

# LS and TS risk for lm fits
matplot(D, cbind(LSRiskLm,TSRiskLm), ylim=c(0.1, 1),
        xlab="D", ylab="Risk", type="p", pch=19, col=2:3, cex=1.5,
        main="Mystery model: Learning and test set risk for lm fits, Y ~ 1 + W + ... + W^D",
        cex.main = 0.5)
legend("topright", c("LS risk", "TS risk"), pch=19, col=2:3, cex=1)
```

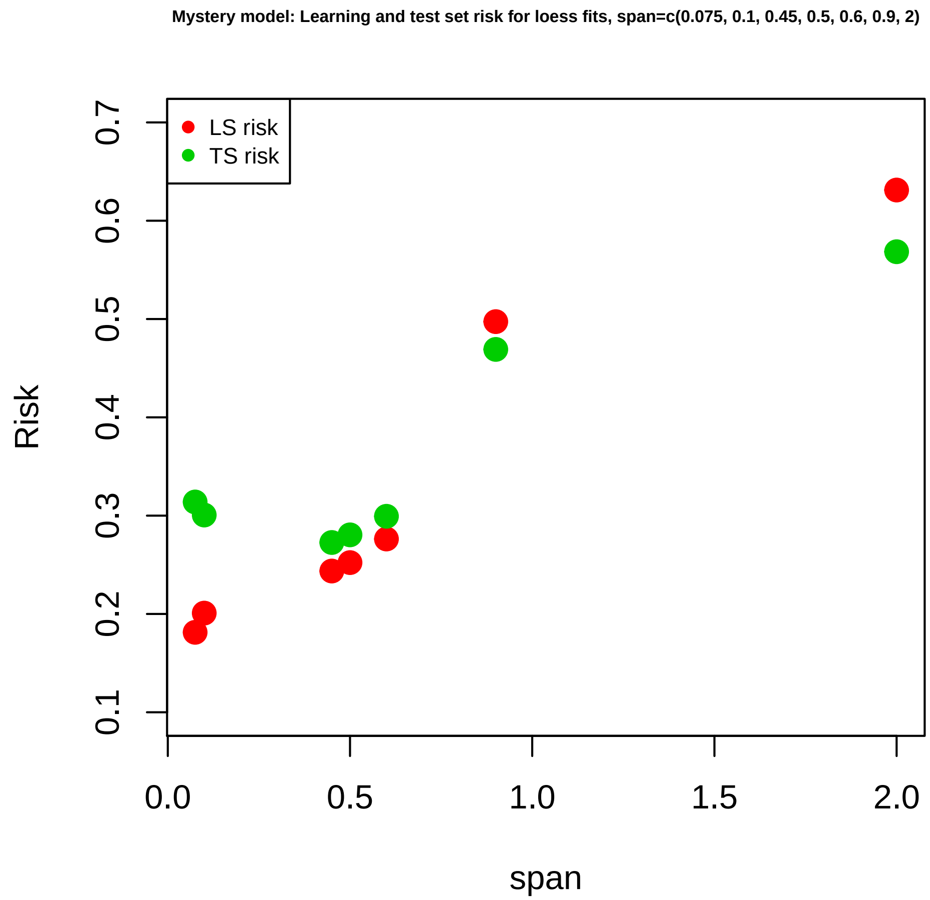



```
## Comment: From the plot, we can see that both LS empirical risk and TS empirical risk are
## decreasing as the D becomes bigger. I think D=4 is the best fit. It has low TS empirical risk,
## and also avoid posibilities of overfitting

##############
### loess ####
##############
span <- c(0.075, 0.1, 0.45,0.5,0.6, 0.9, 2)
# Create an empty matrix to store fitted Y for LS
YhatLoess <- matrix(NA, nrow(ls), length(span))
LSRiskLoess <- TSRiskLoess <- rep(NA,length(span))
for(j in 1:length(span))
{
  fit <- loess(Y ~ W, span=span[j], data=ls)
  YhatLoess[,j] <- fit$fitted
  LSRiskLoess[j] <- mean(fit$residuals^2)
  pred <- predict(fit,data.frame(W=ts$W))
  TSRiskLoess[j] <- mean((pred-ts$Y)^2, na.rm=TRUE)
}

# loess fits
plot(ls$W, ls$Y, xlim=c(-1.5, 1.5), ylim=c(-4, 6), xlab="W", ylab="Y",
     main=paste("Mystery model: loess fits, span=", deparse(span), sep=""),
     cex.main = 0.5)
matplot(ls$W, YhatLoess, type="l", lty=1:length(span),
        lwd=2, col=1:length(span), add=TRUE)
legend("topleft", paste("span=",span,sep=""), lty=1:length(span),
```

```r
       lwd=2, col=1:length(span), cex =0.5)
```




```r
## Comment: The curve fits the data the best when span = 0.45,0.5,0.6
# LS and TS risk for loess fits
matplot(span, cbind(LSRiskLoess,TSRiskLoess), ylim=c(0.1, 0.7),
        xlab="span", ylab="Risk", type="p", pch=19, col=2:3, cex=1.5,
        main=paste("Mystery model: Learning and test set risk for loess fits, span=",
                   deparse(span),sep=""),
        cex.main = 0.5)
legend("topleft", c("LS risk", "TS risk"), pch=19, col=2:3, cex=0.7)
```




```r
## Comment: From this plot, we can see that LS empirical risk is monotonically increasing as we
## increase the span. We can identify a minimum TS empirical risk clearly from the plot when span=0.45.
## Therefore, I think span=0.45 is the best fit.

lapply(fitLm, function(z) round(z$coefficients,2))
```

```
## [[1]]
## (Intercept)        W.ls
##        1.06        2.99
##
## [[2]]
## (Intercept)       W.ls1       W.ls2
##        0.61        3.00        1.22
##
## [[3]]
## (Intercept)       W.ls1       W.ls2       W.ls3
##        0.60        3.39        1.23       -0.64
##
## [[4]]
## (Intercept)       W.ls1       W.ls2       W.ls3       W.ls4
##       -0.16        3.25        8.20       -0.43       -7.98
##
## [[5]]
## (Intercept)       W.ls1       W.ls2       W.ls3       W.ls4       W.ls5
##       -0.18        3.54        8.30       -1.72       -8.09        1.14
##
## [[6]]
## (Intercept)       W.ls1       W.ls2       W.ls3       W.ls4       W.ls5
```

```
##       -0.18        3.54        8.43       -1.72       -8.46        1.12
##       W.ls6
##        0.26
##
## [[7]]
## (Intercept)       W.ls1       W.ls2       W.ls3       W.ls4       W.ls5
##       -0.18        3.44        8.39       -0.77       -8.36       -0.95
##       W.ls6       W.ls7
##        0.18        1.28
##
## [[8]]
## (Intercept)       W.ls1       W.ls2       W.ls3       W.ls4       W.ls5
##       -0.21        3.42        9.30       -0.59      -13.32       -1.36
##       W.ls6       W.ls7       W.ls8
##        8.83        1.56       -4.67
```

```
# lm fits: LS and TS risk
round(rbind(D, LSRiskLm, TSRiskLm), 4)
```

```
##            [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]
## D        1.0000 2.0000 3.0000 4.0000 5.0000 6.0000 7.0000 8.0000
## LSRiskLm 0.7747 0.6403 0.6307 0.2371 0.2351 0.2351 0.2350 0.2345
## TSRiskLm 0.7648 0.6404 0.6595 0.2658 0.2681 0.2686 0.2677 0.2689
```

```
#loess fits: LS and TS risk
round(rbind(span, LSRiskLoess, TSRiskLoess), 4)
```

```
##               [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]
## span        0.0750 0.1000 0.4500 0.5000 0.6000 0.9000 2.0000
## LSRiskLoess 0.1813 0.2009 0.2437 0.2522 0.2763 0.4973 0.6312
## TSRiskLoess 0.3139 0.3006 0.2727 0.2805 0.2994 0.4691 0.5685
```