

# Package ‘ODPR.COVID.19’

June 26, 2020

**Type** Package

**Title** Prediction region based on over-dispersed Poisson Regression

**Version** 0.9

**Date** 2020-06-26

**Author** T. Kim, B. Lieberman, G. Luta, E. Pena

**Maintainer** T. Kim <yourfault@somewhere.net>

**Description** Prediction Regions in Over-Dispersed Poisson Regression with Applications in Forecasting Number of Deaths during the Covid-19 Pandemic

**License** GNU General Public License Version 2

**RoxygenNote** 7.1.0

**NeedsCompilation** no

## R topics documented:

ODPR.COVID.19-package . . . . .	1
AnalysisByCountry2 . . . . .	2
ByCountryData . . . . .	4
ConvertToList . . . . .	5
NewtRaph . . . . .	6
NewtRaphXi . . . . .	7
OveDisPoiReg . . . . .	8
PartialSum . . . . .	10
PoissonRegOveDis . . . . .	11
UHFuns . . . . .	14
UpdateToday . . . . .	16
<b>Index</b>	<b>18</b>

---

ODPR.COVID.19-package *c("\Sexpr[results=rd,stage=build]tools:::Rd\_package\_title(\"#I\")",  
"ODPR.COVID.19")Prediction region based on over-dispersed  
Poisson Regression*

---

**Description**

`c("\Sexpr[results=rd,stage=build]tools:::Rd_package_description(\"#1\")", "ODPR.COVID.19")`Prediction Regions in Over-Dispersed Poisson Regression with Applications in Forecasting Number of Deaths during the Covid-19 Pandemic

**Details**

The DESCRIPTION file: `c("\Sexpr[results=rd,stage=build]tools:::Rd_package_DESCRIPTION(\"#1\")", "ODPR.COVID.19")`This package was not yet installed at build time.\cr `c("\Sexpr[results=rd,stage=build]tools:::Rd_package_INDEX(\"#1\")", "ODPR.COVID.19")` Index: This package was not yet installed at build time.\cr ~ ~ An overview of how to use the package, including the ~ ~ ~ most important functions ~ ~

**Author(s)**

`c("\Sexpr[results=rd,stage=build]tools:::Rd_package_author(\"#1\")", "ODPR.COVID.19")`T. Kim, B. Lieberman, G. Luta, E. Pena

Maintainer: `c("\Sexpr[results=rd,stage=build]tools:::Rd_package_maintainer(\"#1\")", "ODPR.COVID.19")`T. Kim <yourfault@somewhere.net>

---

AnalysisByCountry2      *Analysis by country*

---

**Description**

function for country-wise analysis

**Usage**

```
AnalysisByCountry2(
  CountryAbbrev = "USA",
  alldata = countrydata,
  fval = 0.14,
  lwdval1 = 5,
  lwdval2 = 3,
  cexval = 0.9,
  topplot = F,
  lowdeaths = 1
)
```

**Arguments**

CountryAbbrev	Country name
alldata	data
fval	graphical parameter
lwdval1	graphical parameter
lwdval2	graphical parameter
cexval	graphical parameter
topplot	plot or not
lowdeaths	the lowest number for deaths

**Details**

Okay

**Value**

list

**Examples**

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (CountryAbbrev = "USA", alldata = countrydata, fval = 0.14,
  lwdval1 = 5, lwdval2 = 3, cexval = 0.9, topplot = F, lowdeaths = 1)
{
  countrycodes = alldata$CountryCode
  ccode = which(countrycodes == CountryAbbrev)
  cdata = alldata$Country_Data[[ccode]]
  CountryCode = as.character(cdata$code)
  CountryID = as.character(cdata$id)
  CountryName = as.character(cdata$name)
  CountryPopn = cdata$popn
  StartDate = as.character(cdata$dateRep[1])
  len = length(cdata$dateRep)
  EndDate = as.character(cdata$dateRep[len])
  ind = which(cdata$cumdeaths >= lowdeaths)
  dateFirstCase = as.character(cdata$dateRep[ind[1]])
  dates = as.character(cdata$dateRep[ind])
  daynum = cdata$dayNum[ind]
  dailycases = cdata$cases[ind]
  dailydeaths = cdata$deaths[ind]
  cumcases = cdata$cumcases[ind]
  cumdeaths = cdata$cumdeaths[ind]
  close.screen(all.screens = T)
  if (topplot) {
    plot(daynum, dailydeaths, type = "l", lwd = lwdval1,
      cex = cexval, xlab = "Day Number", ylab = "Daily Deaths")
    scan()
    plot(daynum, cumdeaths, type = "l", lwd = lwdval1, cex = cexval,
      xlab = "Day Number", ylab = "Cumulative Deaths")
  }
  return(list(CountryCode = CountryCode, CountryID = CountryID,
    CountryName = CountryName, Population = CountryPopn,
    StartDate = StartDate, dateFirstCase = dateFirstCase,
    EndDate = EndDate, DateRep = dates, daynum = daynum,
    dailydeaths = dailydeaths, cumdeaths = cumdeaths))
}
```

ByCountryData

*County data***Description**

Function to extract county data

**Usage**

```
ByCountryData(alldat = covid19)
```

**Arguments**

alldat            input data

**Details**

Okay

**Value**

list

**Author(s)**

E. A. Pena

**Examples**

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (alldat = covid19)
{
  uniqueCode = unique(alldat$countryterritoryCode)
  uniqueID = unique(alldat$geoID)
  uniqueCountry = unique(alldat$countriesAndTerritories)
  lenCode = length(uniqueCode)
  Country_Data = vector("list", lenCode)
  n = 0
  for (uc in uniqueCode) {
    n = n + 1
    countrydat = alldat[alldat$countryterritoryCode == uc,
      ]
    temp = ConvertToList(countrydat)
    Country_Data[[n]] = temp
  }
  return(list(CountryCode = uniqueCode, CountryID = uniqueID,
    Country = uniqueCountry, Country_Data = Country_Data))
}
```

---

ConvertToList	<i>Convert to list</i>
---------------	------------------------

---

**Description**

Function for data conversion

**Usage**

```
ConvertToList(countrydat)
```

**Arguments**

countrydat      input data

**Details**

Okay

**Value**

list

**Author(s)**

E. A. Pena

**Examples**

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (countrydat)
{
  nrows = dim(countrydat)[1]
  seqrev = nrows:1
  dat = NULL
  dat$code = countrydat[1, 9]
  dat$id = countrydat[1, 8]
  dat$name = countrydat[1, 7]
  dat$popn = countrydat[1, 10]
  dat$dayNum = 1:nrows
  dat$dateRep = countrydat[seqrev, 1]
  dat$day = countrydat[seqrev, 2]
  dat$month = countrydat[seqrev, 3]
  dat$year = countrydat[seqrev, 4]
  dat$cases = countrydat[seqrev, 5]
  dat$deaths = countrydat[seqrev, 6]
  dat$cumcases = PartialSum(dat$cases)
  dat$cumdeaths = PartialSum(dat$deaths)
  return(dat)
```

```
}
```

---

NewtRaph

*Newton-Raphson Method*


---

## Description

Function for Newton-Raphson method

## Usage

```
NewtRaph(thseed, xiseed, Y, X, tol = 10^(-5))
```

## Arguments

thseed	Initial value for theta
xiseed	Initial value for xi
Y	data
X	data
tol	approximation tolerance

## Details

Okay

## Value

list

## Author(s)

E. A. Pena

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (thseed, xiseed, Y, X, tol = 10^(-5))
{
  p = length(thseed)
  kiter = 0
  th0 = thseed
  xi0 = xiseed
  OK = FALSE
  while (!OK) {
    kiter = kiter + 1
    est0 = c(th0, xi0)
```

```

    out = UHFuncs(th0, xi0, Y, X)
    U = out$U
    H = out$H
    UUt = out$UUt
    Hinv = solve(H)
    est1 = est0 - Hinv %*% U
    th0 = est1[-(p + 1)]
    xi0 = est1[(p + 1)]
    dist = sqrt(sum((est1 - est0)^2))
    if (dist < tol) {
      OK = TRUE
    }
  }
  }
  XiCov = Hinv %*% UUt %*% t(Hinv)
  return(list(kiter = kiter, th = th0, xi = xi0, U = U, H = H,
    Hinv = Hinv, UUt = UUt, XiCov = XiCov))
}

```

NewtRaphXi

*Newton-Raphson for Xi***Description**

Function for Newton-Raphson method for xi

**Usage**

```
NewtRaphXi(xiseed, theta, Y, X, tol = 10^(-5))
```

**Arguments**

xiseed	Initial value
theta	theta
Y	data
X	data
tol	approximation tolerance

**Details**

Okay

**Value**

list

**Author(s)**

E. A. Pena

## Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (xiseed, theta, Y, X, tol = 10^(-5))
{
  rho = exp(X %*% theta)
  n = length(Y)
  kiter = 0
  OK = FALSE
  xi0 = xiseed
  while (!OK) {
    kiter = kiter + 1
    g = sum((Y - rho)^2 - rho * (1 + (1 + rho)/xi0))
    gp = sum(rho * (1 + rho))/xi0^2
    xi1 = xi0 - g/gp
    if (abs(xi1 - xi0) < tol) {
      OK = TRUE
    }
    xi0 = xi1
    if (xi1 < 0) {
      OK = FALSE
      xiseed = xiseed/2
      xi0 = xiseed
      kiter = 0
    }
  }
  return(list(kiter = kiter, xi = xi1))
}
```

---

OveDisPoiReg

*Regression part*

---

## Description

Function handles the regression part

## Usage

```
OveDisPoiReg(Y, X, X0 = NULL, xival = NULL, alpha = 0.05, tol = 10^(-5))
```

## Arguments

Y	data
X	data
X0	prediction
xival	over-dispersion parameter, xi
alpha	alpha
tol	approximation tolerance



**Details**

Okay

**Value**

list

**Author(s)**

E. A. Pena

**Examples**

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (Y, X, X0 = NULL, xival = NULL, alpha = 0.05, tol = 10^(-5))
{
  p = dim(X)[2]
  n = dim(X)[1]
  if (is.null(X0)) {
    X0 = X
  }
  m = dim(X0)[1]
  Xmean = apply(X, 2, "mean")
  Xstd = apply(X, 2, "sd")
  XS = X
  X0S = X0
  for (j in 2:p) {
    XS[, j] = (X[, j] - Xmean[j])/Xstd[j]
    X0S[, j] = (X0[, j] - Xmean[j])/Xstd[j]
  }
  outglm = glm(Y ~ XS[, -1], family = poisson(link = log))
  outglmsumm = summary(outglm)
  theta = outglm$coefficients
  if (is.null(xival)) {
    xi = NewtRaphXi(1, theta, Y, XS, tol)$xi
  }
  else {
    xi = Inf
  }
  XiCov = NewtRaph(theta, xi, Y, XS, tol)$XiCov
  XiCov11 = XiCov[1:p, 1:p]
  rho0 = exp(X0S %*% theta)
  zcrit = qnorm(1 - alpha/2)
  PE = rep(0, m)
  PE2 = rep(0, m)
  for (i in 1:m) {
    PE[i] = zcrit * sqrt(rho0[i] + rho0[i] * (1 + rho0[i])/xi +
      (1/n) * (rho0[i]^2) * (t(X0S[i, ]) %*% XiCov11) %*%
      X0S[i, ])
  }
  PILow = rho0 - PE
}
```

```

    PIHig = rho0 + PE
    for (i in 1:m) {
        PILow[i] = max(0, PILow[i])
    }
    return(list(outglm = outglm, outglmsumm = outglmsumm, theta = theta,
               xi = xi, XiCov = XiCov, Xmean = Xmean, Xstd = Xstd, FitPred = data.frame(rho0,
               PILow, PIHig, X0)))
}

```

---

PartialSum

*Partial Sum*


---

## Description

Function for data partial summation

## Usage

PartialSum(x)

## Arguments

x                      data

## Details

Okay

## Value

list

## Author(s)

E. A. Pena

## Examples

```

##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (x)
{
    nlen = length(x)
    y = rep(0, nlen)
    for (i in 1:nlen) {
        y[i] = sum(x[1:i])
    }
    return(y)
}

```

---

PoissonRegOveDis

---

Main Function on Over-dispersed Poisson Regression

---

## Description

This is the main file which incorporates other functions.

## Usage

```
PoissonRegOveDis(
  daycutpred = 137,
  maxday = 154,
  ordfit = 5,
  DayEffect = TRUE,
  Country = "USA",
  xival = NULL,
  dat = NULL,
  alpha = 0.05,
  yhighval = NULL,
  targetval = 1e+05,
  LegPos = c("topright", "left"),
  ToPlot = TRUE
)
```

## Arguments

daycutpred	threshold for prediction
maxday	maximum day for the prediction
ordfit	the order of polynomial of Poisson regression
DayEffect	day effect
Country	country name
xival	over-dispersion parameter, xi
dat	data
alpha	alpha
yhighval	graphical parameter
targetval	graphical parameter
LegPos	graphical parameter
ToPlot	plot or not

## Details

Okay

## Value

list

**Author(s)**

E. A. Pena

**Examples**

```

##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (daycutpred = 137, maxday = 154, ordfit = 5, DayEffect = TRUE,
        Country = "USA", xival = NULL, dat = NULL, alpha = 0.05,
        yhighval = NULL, targetval = 1e+05, LegPos = c("topright",
        "left"), ToPlot = TRUE)
{
  if (is.null(dat)) {
    countrydata = UpdateToday()
  }
  else {
    countrydata = dat
  }
  dat = AnalysisByCountry2(Country, alldata = countrydata)
  daynum = dat$daynum
  deaths = dat$dailydeaths
  cumdeaths = dat$cumdeaths
  DateRep = dat$DateRep
  close.screen(all.screens = T)
  daynumc = daynum - daynum[1]
  X = NULL
  collab = NULL
  for (j in 0:ordfit) {
    X = cbind(X, daynumc^j)
    collab = c(collab, paste("DayNum", j))
  }
  colnames(X) = collab
  if (DayEffect) {
    dayclass = daynum%%7
    dayclass = factor(dayclass, levels = as.character(0:6),
      labels = c("Monday", "Tuesday", "Wednesday", "Thursday",
      "Friday", "Saturday", "Sunday"))
    ll = length(dayclass)
    desmat = model.matrix(~dayclass)[, -1]
    daylab = c("Tuesday", "Wednesday", "Thursday", "Friday",
      "Saturday", "Sunday")
    X = cbind(X, desmat)
    colnames(X) = c(collab, daylab)
  }
  inpred = (daynum <= daycutpred)
  numdaysused = sum(inpred)
  Y = deaths[inpred]
  X = X[inpred, ]
  X0 = NULL
  dayspred = daynum[1]:maxday
  dayspredc = dayspred - daynum[1]
  for (j in 0:ordfit) {

```

```

    X0 = cbind(X0, dayspredc^j)
  }
  if (DayEffect) {
    dayspredclass = as.factor(dayspred%%7)
    dayspredclass = factor(dayspredclass, levels = 0:6, labels = c("Monday",
      "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
      "Sunday"))
    ll = length(dayspredclass)
    desmat = model.matrix(~dayspredclass)[, -1]
    X0 = cbind(X0, desmat)
  }
  numdaystopred = maxday - daycutpred
  if (numdaystopred == 0) {
    alphaeff = alpha
  }
  else {
    alphaeff = 1 - (1 - alpha)^(1/numdaystopred)
  }
  out = OveDisPoiReg(Y, X, X0, alpha = alphaeff, xival = xival)
  outglm = out$outglm
  outglmsumm = out$outglmsumm
  thetahat = out$theta
  xihat = out$xi
  XiCov = out$XiCov
  Xmean = out$Xmean
  Xstd = out$Xstd
  FitPred = out$FitPred[, 1:3]
  if (is.null(yhighval)) {
    yhighval = max(c(deaths, FitPred[, 3]))
  }
  if (ToPlot) {
    plot(NULL, NULL, xlim = c(daynum[1], maxday), ylim = c(0,
      yhighval), xlab = "Day Number Since 12/31/2019",
      ylab = "Daily Deaths", main = paste("Daily Deaths due to Covid-19 in ",
        Country))
    points(daynum[inpred], Y, col = "black", lwd = 3)
    matlines(dayspred, FitPred, col = c("blue", rep("red",
      2)), lty = 1, pch = 1, lwd = 3)
    points(daynum[(!inpred) & (daynum <= maxday)], deaths[(!inpred) &
      (daynum <= maxday)], col = "red", lwd = 3)
    abline(v = daycutpred, col = "black", lwd = 2)
    abline(v = maxday, col = "green", lwd = 2)
  }
  lendayspred = length(dayspred)
  cumdeathspred = rep(0, lendayspred)
  cumdeathspredlow = rep(0, lendayspred)
  cumdeathspredhig = rep(0, lendayspred)
  for (i in 1:lendayspred) {
    cumdeathspred[i] = sum(FitPred[1:i, 1])
    cumdeathspredlow[i] = sum(FitPred[1:i, 2])
    cumdeathspredhig[i] = sum(FitPred[1:i, 3])
  }
  if (daycutpred < maxday) {
    daysahead = (daycutpred + 1):maxday
    lendaysahead = length(daysahead)
    lennow = length(daynum[daynum <= daycutpred])
    currentval = cumdeaths[lennow]
  }

```

```

cumdeathspredahead = rep(currentval, lendaysahead)
cumdeathspredaheadlow = rep(currentval, lendaysahead)
cumdeathspredaheadhig = rep(currentval, lendaysahead)
for (i in 1:lendaysahead) {
  cumdeathspredahead[i] = cumdeathspredahead[i] + sum(FitPred[(lennow +
    1):(lennow + i), 1])
  cumdeathspredaheadlow[i] = cumdeathspredaheadlow[i] +
    sum(FitPred[(lennow + 1):(lennow + i), 2])
  cumdeathspredaheadhig[i] = cumdeathspredaheadhig[i] +
    sum(FitPred[(lennow + 1):(lennow + i), 3])
}
maxy = max(cumdeathspredaheadhig)
}
else {
  maxy = max(cumdeaths[inpred])
}
if (ToPlot) {
  scan()
  plot(NULL, NULL, xlim = range(dayspred), ylim = c(0,
    maxy), xlab = "Day Number Since 12/31/2019", ylab = "Cumulative Deaths",
    main = paste("Cumulative Deaths Due to Covid-19 in ",
      Country))
  lines(dayspred, cumdeathspred, type = "l", col = c("blue"),
    lty = 1, pch = 1, lwd = 3)
  points(daynum[inpred], cumdeaths[inpred], col = "black",
    lwd = 3)
  points(daynum[(!inpred) & (daynum <= maxday)], cumdeaths[(!inpred) &
    (daynum <= maxday)], col = "red", lwd = 3)
  if (daycutpred < maxday) {
    matlines(daysahead, cbind(cumdeathspredaheadlow,
      cumdeathspredaheadhig), col = "magenta", lty = 1,
      pch = 2, lwd = 3)
  }
  abline(v = daycutpred, col = "black", lwd = 2)
  abline(v = maxday, col = "green", lwd = 2)
}
summpreds = NULL
if (daycutpred < maxday) {
  summpreds = data.frame(daynum = daysahead, Pred = cumdeathspredahead,
    PILow = cumdeathspredaheadlow, PIHig = cumdeathspredaheadhig)
}
return(list(Country = Country, dat = dat, outglm = outglm,
  outglmsumm = outglmsumm, daycutpred = daycutpred, numdaysused = numdaysused,
  maxday = maxday, numdaystopred = numdaystopred, ordfit = ordfit,
  alpha = alpha, alphaeff = alphaeff, thetahat = thetahat,
  xihat = xihat, XiCov = XiCov, Xmean = Xmean, Xstd = Xstd,
  summpreds = summpreds))
}

```

UHFuncs

UHFuncs

## Description

Estimation

**Usage**

```
UHFunks(th, xi, Y, X)
```

**Arguments**

th	parameter1
xi	target parameter
Y	data
X	data

**Details**

Okay

**Value**

list

**Author(s)**

E. A. Pena

**Examples**

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function (th, xi, Y, X)
{
  p = length(th)
  n = length(Y)
  U1 = rep(0, p)
  U2 = 0
  UUt = matrix(0, (p + 1), (p + 1))
  H11 = matrix(0, p, p)
  H12 = matrix(0, p, 1)
  H21 = matrix(0, 1, p)
  H22 = 0
  for (i in 1:n) {
    xtemp = as.vector(X[i, ])
    ytemp = Y[i]
    rho = as.numeric(exp(t(xtemp) %*% th))
    u1add = xtemp * (ytemp - rho)
    u2add = (ytemp - rho)^2 - rho * (1 + (1 + rho)/xi)
    u12add = c(u1add, u2add)
    U1 = U1 + u1add
    U2 = U2 + u2add
    UUt = UUt + u12add %*% t(u12add)
    H11 = H11 - (xtemp %*% t(xtemp)) * rho
    H21 = H21 - xtemp * rho * (2 * (ytemp - rho) + (1 + (1 + rho)/xi) - rho/xi^2)
    H22 = H22 + (1/xi^2) * rho * (1 + rho)
  }
}
```

```

    }
    U = c(U1, U2)/n
    H = rbind(cbind(H11, H12), cbind(H21, H22))/n
    UUt = UUt/n
    return(list(U = U, H = H, UUt = UUt))
  }

```

---

UpdateToday

*Update today*


---

### Description

function for date updating

### Usage

```
UpdateToday()
```

### Details

Okay

### Value

list

### Note

function for data updating

### Author(s)

E. A. Pena

### References

No particular reference

### See Also

###

### Examples

```

##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.

## The function is currently defined as
function ()
{
  library(utils)

```



```
library(httr)
GET("https://opendata.ecdc.europa.eu/covid19/casedistribution/csv",
    authenticate(":", ":", type = "ntlm"), write_disk(tf <- tempfile(fileext = ".csv")))
covid19 <- read.csv(tf)
countrydata = ByCountryData(covid19)
return(countrydata)
}
```

# Index

- \* **Over-dispersed**

- ODPR.COVID.19-package, [1](#)

- \* **Poisson**

- ODPR.COVID.19-package, [1](#)

- \* **Regression**

- ODPR.COVID.19-package, [1](#)

- \* **model**

- ODPR.COVID.19-package, [1](#)

AnalysisByCountry2, [2](#)

ByCountryData, [4](#)

ConvertToList, [5](#)

NewtRaph, [6](#)

NewtRaphXi, [7](#)

ODPR.COVID.19 (ODPR.COVID.19-package), [1](#)

ODPR.COVID.19-package, [1](#)

OveDisPoiReg, [8](#)

PartialSum, [10](#)

PoissonRegOveDis, [11](#)

UHFuns, [14](#)

UpdateToday, [16](#)