Oregon Department of Environmental Quality

An R-based Web Application to Search, Analyze and Display Water Quality Data in Oregon State, USA

Peter Bryant

May 4th, 2016
National Water Quality Monitoring Council
10th National Monitoring Conference
Tampa, Florida

Web Application Goals

Provide an interactive means for users to:

- – Query data from multiple databases
- – Evaluate status and trend at individual sampling stations
- – Display the results

Non-R users are able to generate information on status and trends

Water Quality Data Start Here...

...And Go Here

- DEQ databases – Current and legacy databases for grab and continuous data
- NWQMC Water Quality Portal for grab data
  - EPA STORET
  - USGS NWIS
- USGS National Water Information System for continuous data

Then What?

Use the data to:

- Determine status of water quality
- Look for trends in water quality

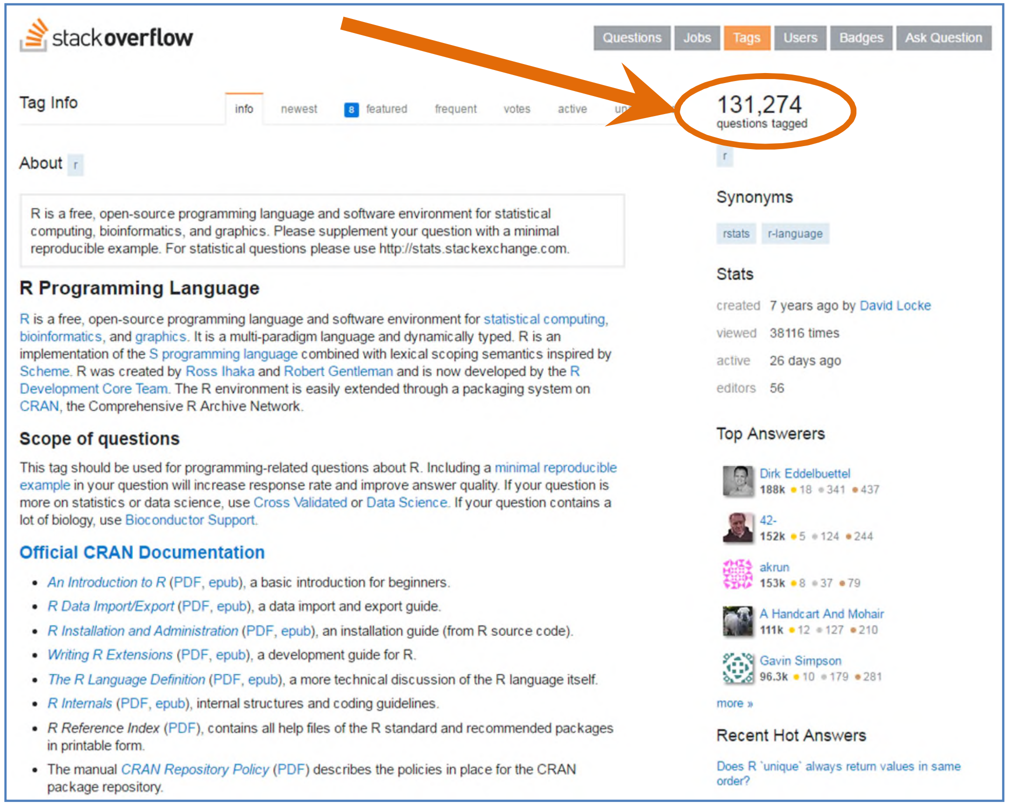
But How?

That’s a lot of places to look and potentially a lot of data to analyze

Luckily there’s R

What is R ?

- Open source statistical programming language
- It’s free (r-project.org)
- Has a large user base
- Lots of documentation and learning resources



Tag Info

131,274
questions tagged

About r

R is a free, open-source programming language and software environment for statistical computing, bioinformatics, and graphics. Please supplement your question with a minimal reproducible example. For statistical questions please use http://stats.stackexchange.com.

Synonyms

R Programming Language

R is a free, open-source programming language and software environment for statistical computing, bioinformatics, and graphics. It is a multi-paradigm language and dynamically typed. R is an implementation of the S programming language combined with lexical scoping semantics inspired by Scheme. R was created by Ross Ihaka and Robert Gentleman and is now developed by the R Development Core Team. The R environment is easily extended through a packaging system on CRAN, the Comprehensive R Archive Network.

Scope of questions

This tag should be used for programming-related questions about R. Including a minimal reproducible example in your question will increase response rate and improve answer quality. If your question is more on statistics or data science, use Cross Validated or Data Science. If your question contains a lot of biology, use Bioconductor Support.

Official CRAN Documentation

- *An Introduction to R* (PDF, epub), a basic introduction for beginners.
- *R Data Import/Export* (PDF, epub), a data import and export guide.
- *R Installation and Administration* (PDF, epub), an installation guide (from R source code).
- *Writing R Extensions* (PDF, epub), a development guide for R.
- *The R Language Definition* (PDF, epub), a more technical discussion of the R language itself.
- *R Internals* (PDF, epub), internal structures and coding guidelines.
- *R Reference Index* (PDF), contains all help files of the R standard and recommended packages in printable form.
- The manual *CRAN Repository Policy* (PDF) describes the policies in place for the CRAN package repository.

Stats

created	7 years ago by David Locke
viewed	38116 times
active	26 days ago
editors	56

Top Answerers

Dirk Eddelbuettel
188k ● 18 ● 341 ● 437

42-
152k ● 5 ● 124 ● 244

akrun
153k ● 8 ● 37 ● 79

A Handcart And Mohair
111k ● 12 ● 127 ● 210

Gavin Simpson
96.3k ● 10 ● 179 ● 281

Recent Hot Answers

Does R `unique` always return values in same order?

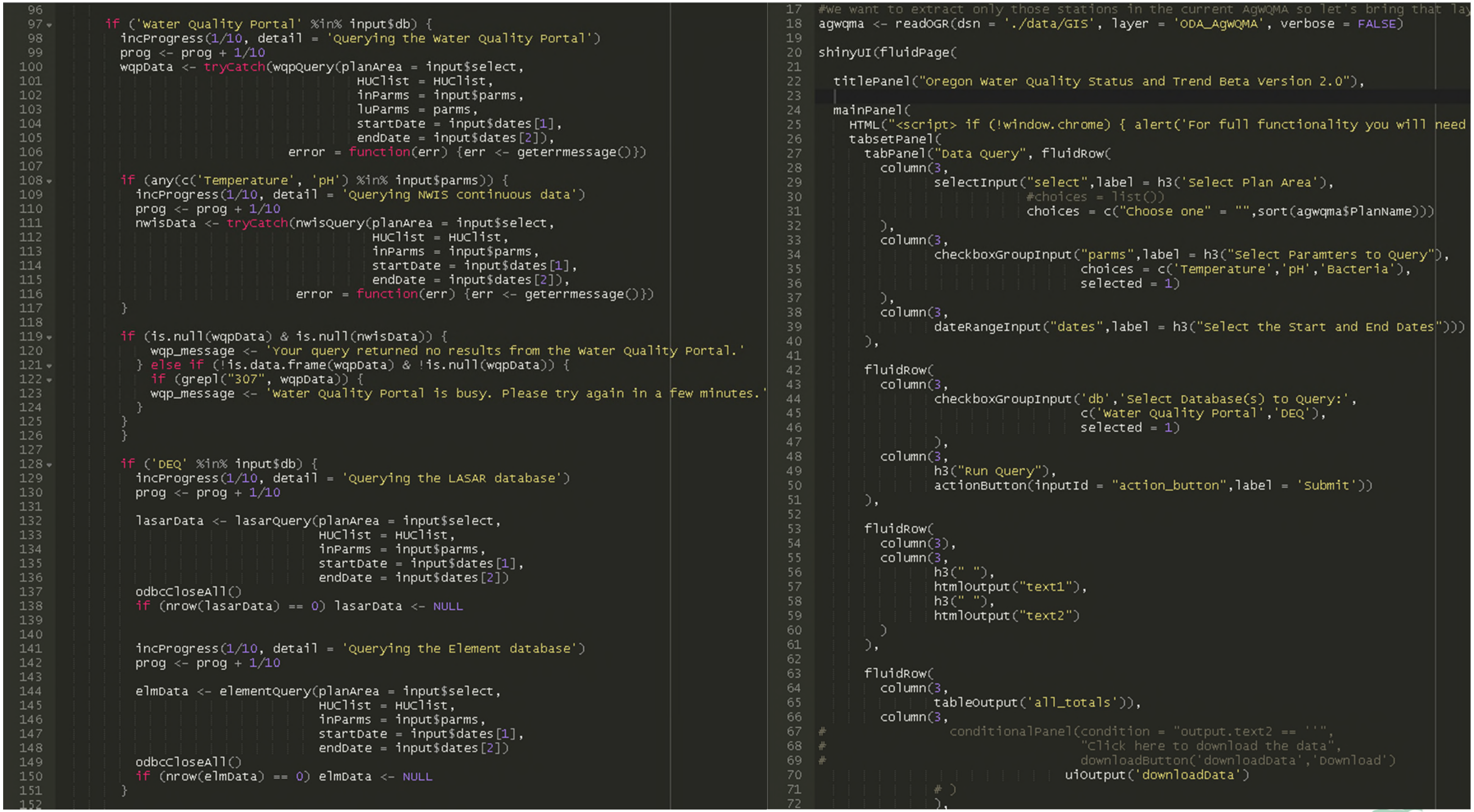
How Does R Work?

- Use a text editor to write out the analysis
- Pass the analysis code to the R software to actually run the analysis
- R users have built packages to do many different types of analysis
- A package is a collection of functions to perform an analysis

The Shiny Package

- The Shiny package is built by RStudio
  - Provides R functions that translate R code to an interactive javascript web application
  - Builds a user interface to edit settings in the analysis
  - Puts all the code to run an analysis behind the user interface

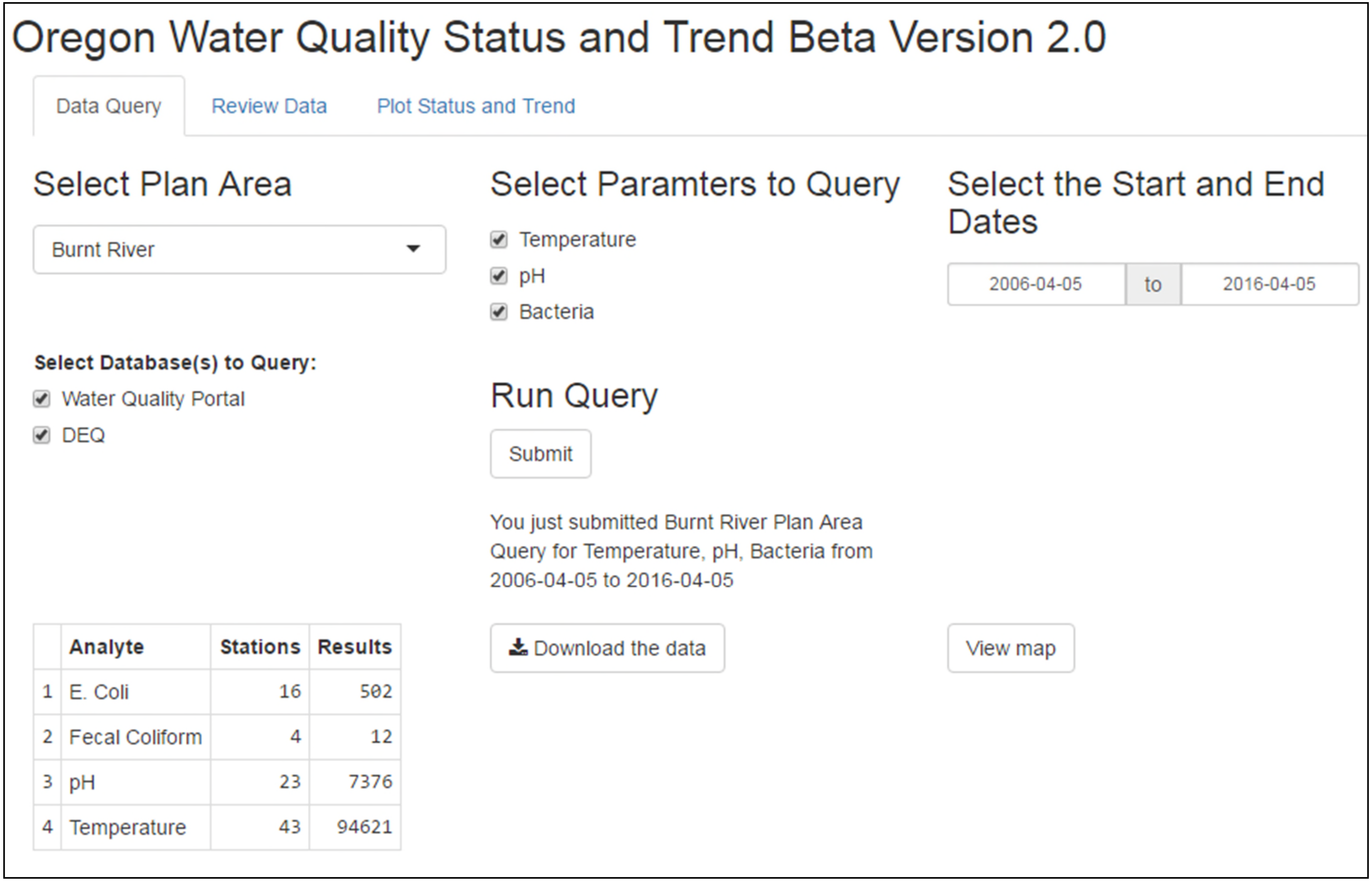
While I see this



```r

if ('Water Quality Portal' %in% input$db) {
  incProgress(1/10, detail = 'Querying the Water Quality Portal')
  prog <- prog + 1/10
  wqpData <- tryCatch(wqpQuery(planArea = input$select,
                                 HUClist = HUClist,
                                 inParms = input$parms,
                                 luParms = parms,
                                 startDate = input$dates[1],
                                 endDate = input$dates[2]),
                        error = function(err) {err <- geterrmessage()})

  if (any(c('Temperature', 'pH') %in% input$parms)) {
    incProgress(1/10, detail = 'Querying NWIS continuous data')
    prog <- prog + 1/10
    nwisData <- tryCatch(nwisQuery(planArea = input$select,
                                   HUClist = HUClist,
                                   inParms = input$parms,
                                   startDate = input$dates[1],
                                   endDate = input$dates[2]),
                         error = function(err) {err <- geterrmessage()})
  }

  if (is.null(wqpData) & is.null(nwisData)) {
      wqp_message <- 'Your query returned no results from the Water Quality Portal.'
    } else if (!is.data.frame(wqpData) & !is.null(wqpData)) {
      if (grepl("307", wqpData)) {
      wqp_message <- 'Water Quality Portal is busy. Please try again in a few minutes.'
    }
  }
  }

  if ('DEQ' %in% input$db) {
    incProgress(1/10, detail = 'Querying the LASAR database')
    prog <- prog + 1/10

    lasarData <- lasarQuery(planArea = input$select,
                            HUClist = HUClist,
                            inParms = input$parms,
                            startDate = input$dates[1],
                            endDate = input$dates[2])
    odbcCloseAll()
    if (nrow(lasarData) == 0) lasarData <- NULL


    incProgress(1/10, detail = 'Querying the Element database')
    prog <- prog + 1/10

    elmData <- elementQuery(planArea = input$select,
                            HUClist = HUClist,
                            inParms = input$parms,
                            startDate = input$dates[1],
                            endDate = input$dates[2])
    odbcCloseAll()
    if (nrow(elmData) == 0) elmData <- NULL
  }

```

```r
#We want to extract only those stations in the current AgWQMA so let's bring that lay
agwqma <- readOGR(dsn = './data/GIS', layer = 'ODA_AgWQMA', verbose = FALSE)

shinyUI(fluidPage(

  titlePanel("Oregon Water Quality Status and Trend Beta Version 2.0"),

  mainPanel(
    HTML("<script> if (!window.chrome) { alert('For full functionality you will need
    tabsetPanel(
      tabPanel("Data Query", fluidRow(
        column(3,
               selectInput("select",label = h3('Select Plan Area'),
                           #choices = list())
                           choices = c("Choose one" = "",sort(agwqma$PlanName)))
        ),
        column(3,
               checkboxGroupInput("parms",label = h3("Select Paramters to Query"),
                                  choices = c('Temperature','pH','Bacteria'),
                                  selected = 1)
        ),
        column(3,
               dateRangeInput("dates",label = h3("Select the Start and End Dates")))
      ),

      fluidRow(
        column(3,
               checkboxGroupInput('db','Select Database(s) to Query:',
                                  c('Water Quality Portal','DEQ'),
                                  selected = 1)
               ),
        column(3,
               h3("Run Query"),
               actionButton(inputId = "action_button",label = 'Submit'))
      ),

      fluidRow(
        column(3),
        column(3,
               h3(" "),
               htmlOutput("text1"),
               h3(" "),
               htmlOutput("text2")
        )
      ),

      fluidRow(
        column(3,
               tableOutput('all_totals')),
        column(3,
#                conditionalPanel(condition = "output.text2 == ''",
#                                 "Click here to download the data",
#                                 downloadButton('downloadData','Download')
                                uiOutput('downloadData')
               # )
               ),
```

You see this



Oregon Water Quality Status and Trend Beta Version 2.0

Select Plan Area

Burnt River ▾

Select Database(s) to Query:

☑ Water Quality Portal

☑ DEQ

Select Paramters to Query

☑ Temperature

☑ pH

☑ Bacteria

Run Query

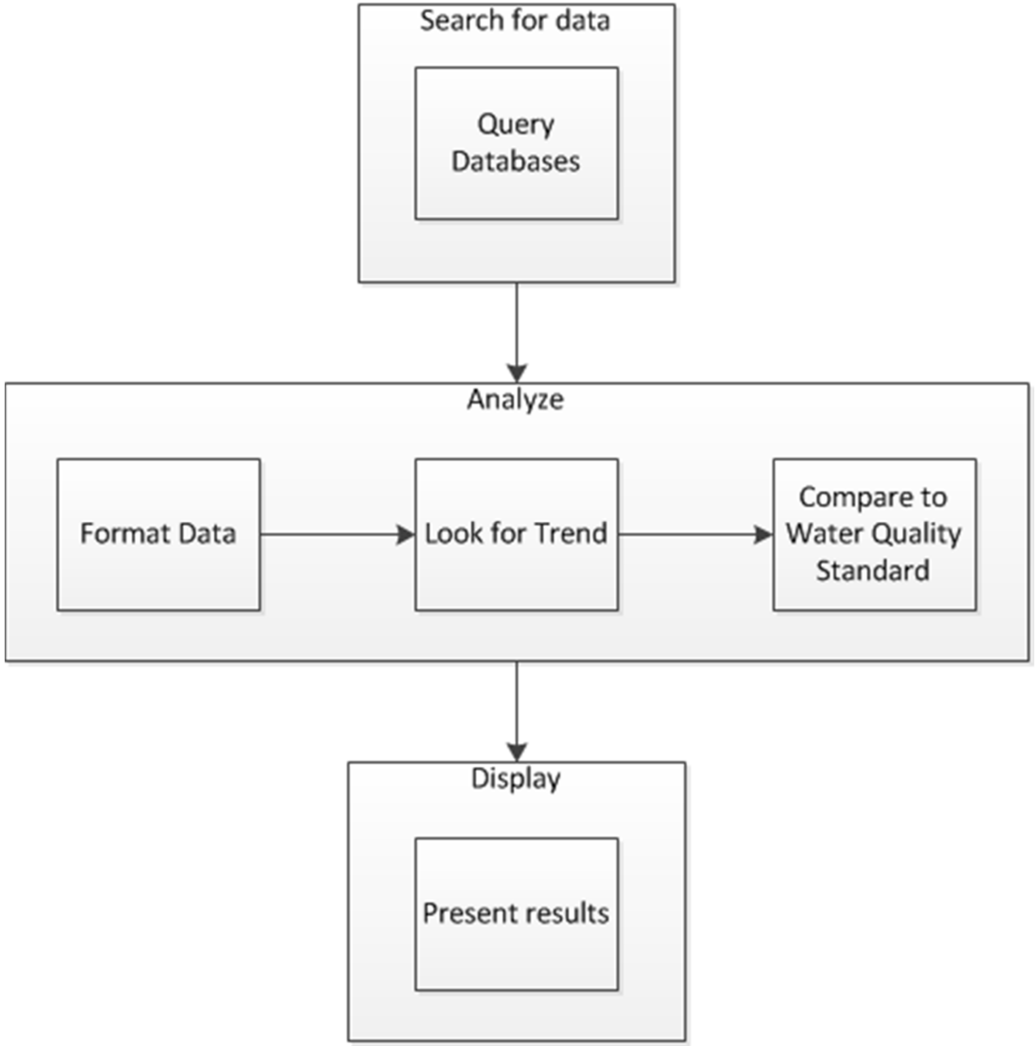
You just submitted Burnt River Plan Area Query for Temperature, pH, Bacteria from 2006-04-05 to 2016-04-05

Select the Start and End Dates

2006-04-05 to 2016-04-05

	Analyte	Stations	Results
1	E. Coli	16	502
2	Fecal Coliform	4	12
3	pH	23	7376
4	Temperature	43	94621

Application Structure



Search for data

Query Databases

Analyze

Format Data

Look for Trend

Compare to Water Quality Standard

Display

Present results

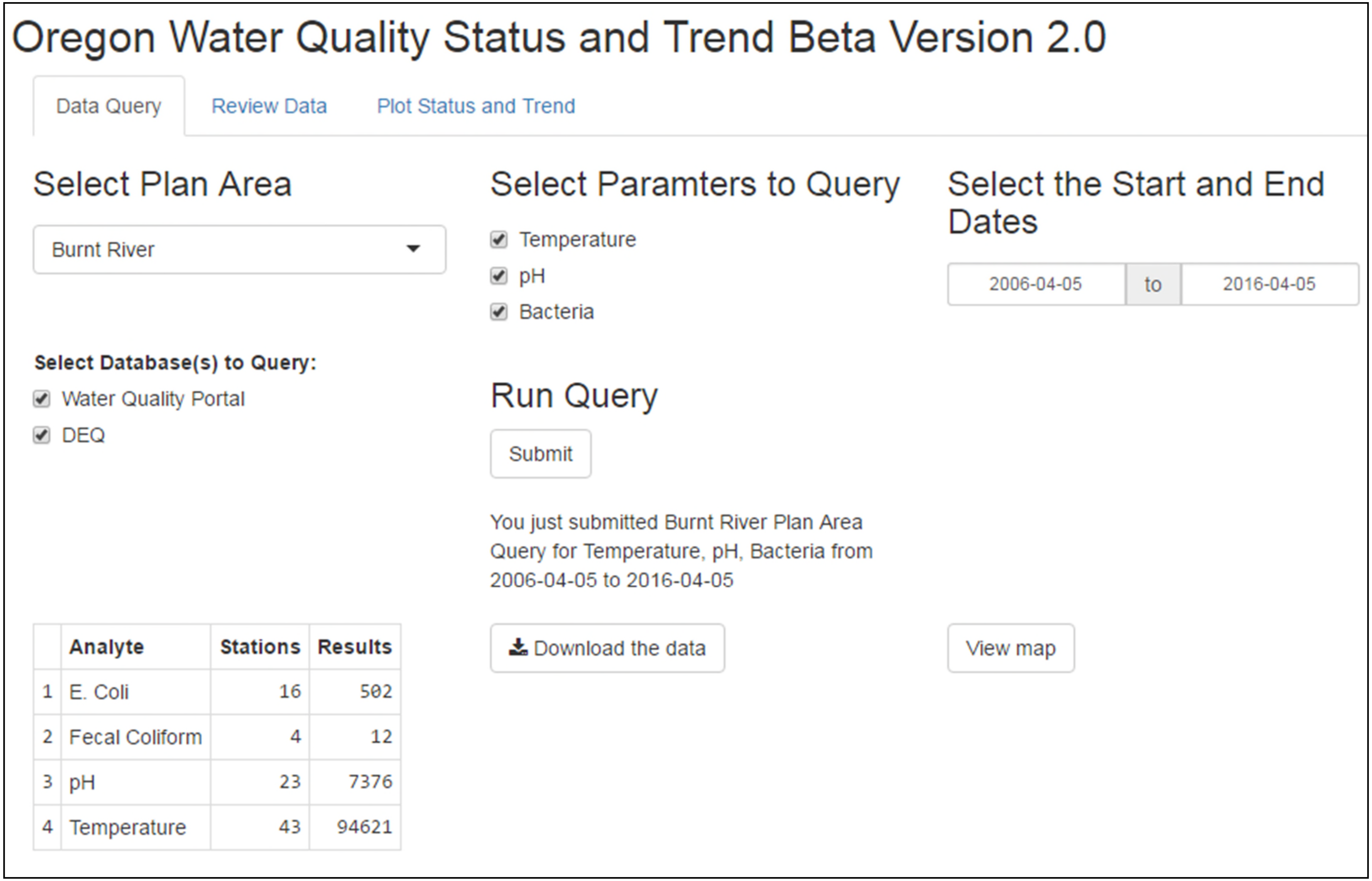
Search For Data

Query Databases

- RODBC package for DEQ data
  - Provides connection from R to SQL Server databases

- dataRetrieval package for Water Quality Portal data
  - A collection of functions to get data from USGS and EPA National Databases
  - Uses web services to communicate with the databases

Search For Data



Oregon Water Quality Status and Trend Beta Version 2.0

Select Plan Area

Burnt River ▾

Select Database(s) to Query:

☑ Water Quality Portal

☑ DEQ

Select Paramters to Query

☑ Temperature

☑ pH

☑ Bacteria

Select the Start and End Dates

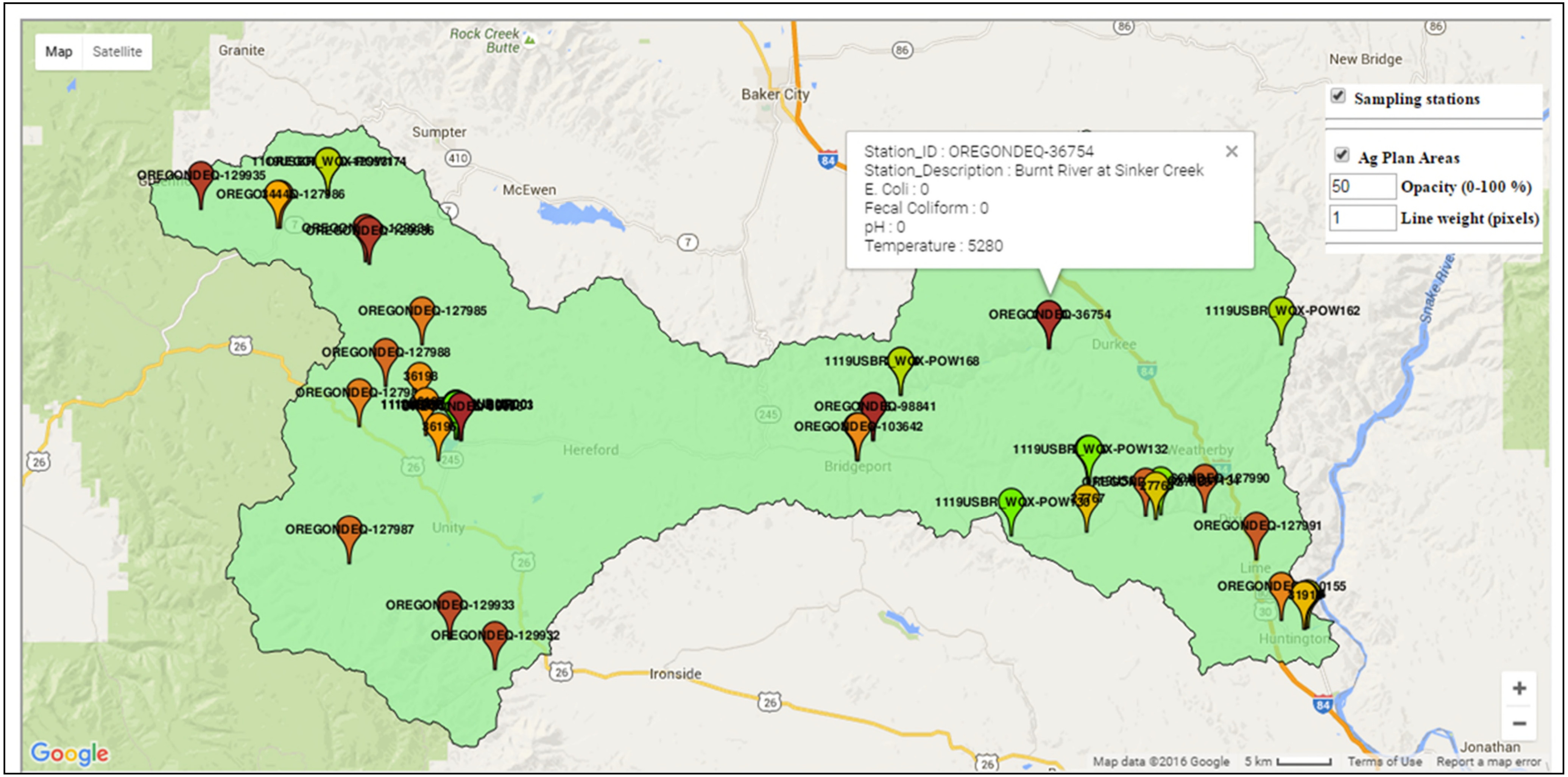
2006-04-05 to 2016-04-05

Run Query

You just submitted Burnt River Plan Area Query for Temperature, pH, Bacteria from 2006-04-05 to 2016-04-05

	Analyte	Stations	Results
1	E. Coli	16	502
2	Fecal Coliform	4	12
3	pH	23	7376
4	Temperature	43	94621

Search For Data



Station_ID : OREGONDEQ-36754
Station_Description : Burnt River at Sinker Creek
E. Coli : 0
Fecal Coliform : 0
pH : 0
Temperature : 5280

☑ Sampling stations

☑ Ag Plan Areas

50 Opacity (0-100 %)

1 Line weight (pixels)

Analyze

Format

- Each database has different field names
- Built function to combine results from each database into a single table

Analyze

Look for trend

- Uses Seasonal Kendall trend analysis from wq package

- Apply to each station for each parameter

Analyze

Compare to Water Quality Standard

Figure 230B: Salmon and Steelhead Spawning Use Designations*
North Coast Basin, Oregon
PACIFIC OCEAN
Legend

- Many of Oregon’s standards are geo-specific (e.g. Temperature and pH)
- Data are gathered from many different sources
- Station location information may not be accurate
- User is able to specify geo-specific criteria to apply to the station for analysis

Analyze

Compare to Water Quality Standard

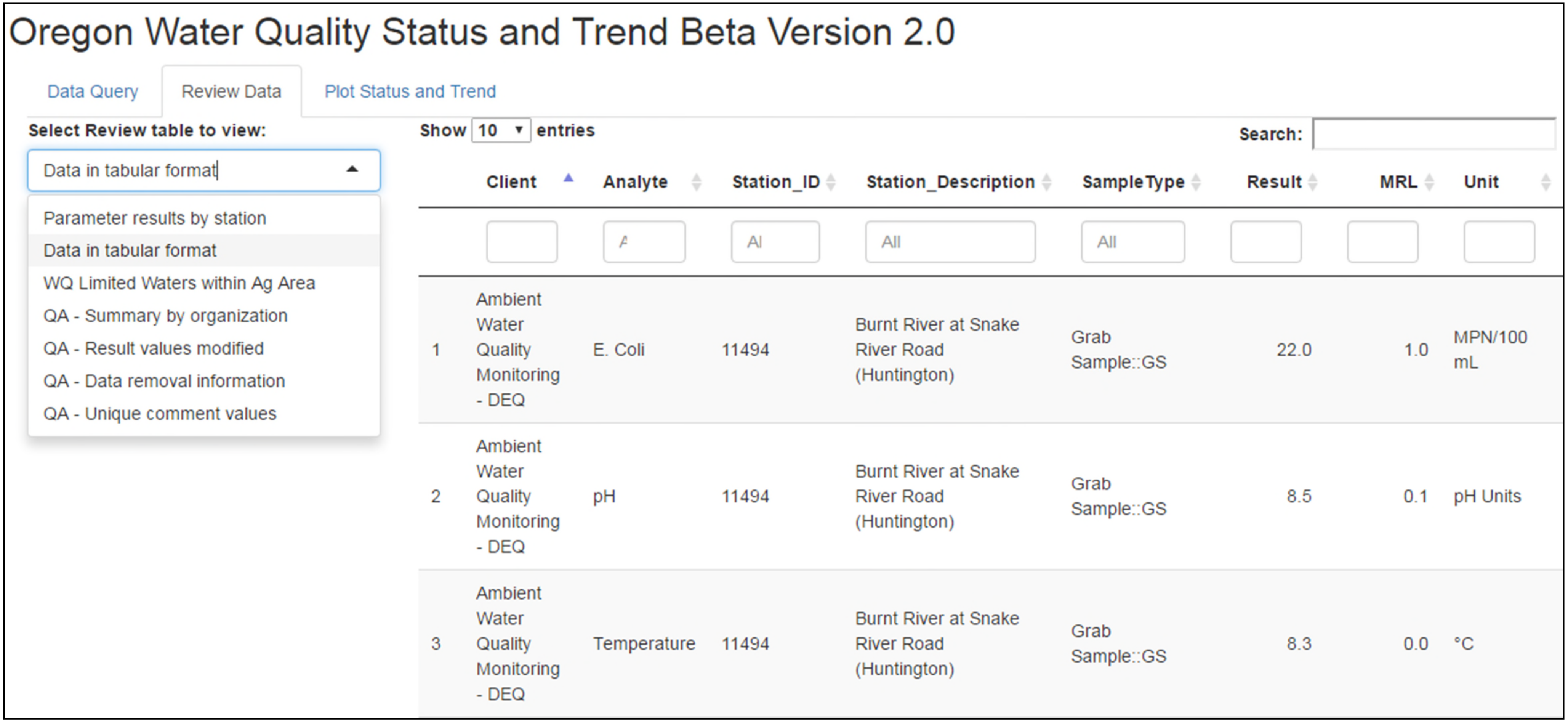
- Temperature standard is based on 7 Day Average Daily Maximum

- E. Coli and Enterococcus are based on 30-day Geometric Mean

Display

- Provide tabular summaries of formatted data
- Create interactive charts using ggplot2 package
- Provide zooming to specific time periods in the charts
- Provide download of chart to insert into report
- Construct responsive table of exceedances based on changes to selected water quality standard

Display



Oregon Water Quality Status and Trend Beta Version 2.0

Data Query
Review Data
Plot Status and Trend

Select Review table to view: Data in tabular format

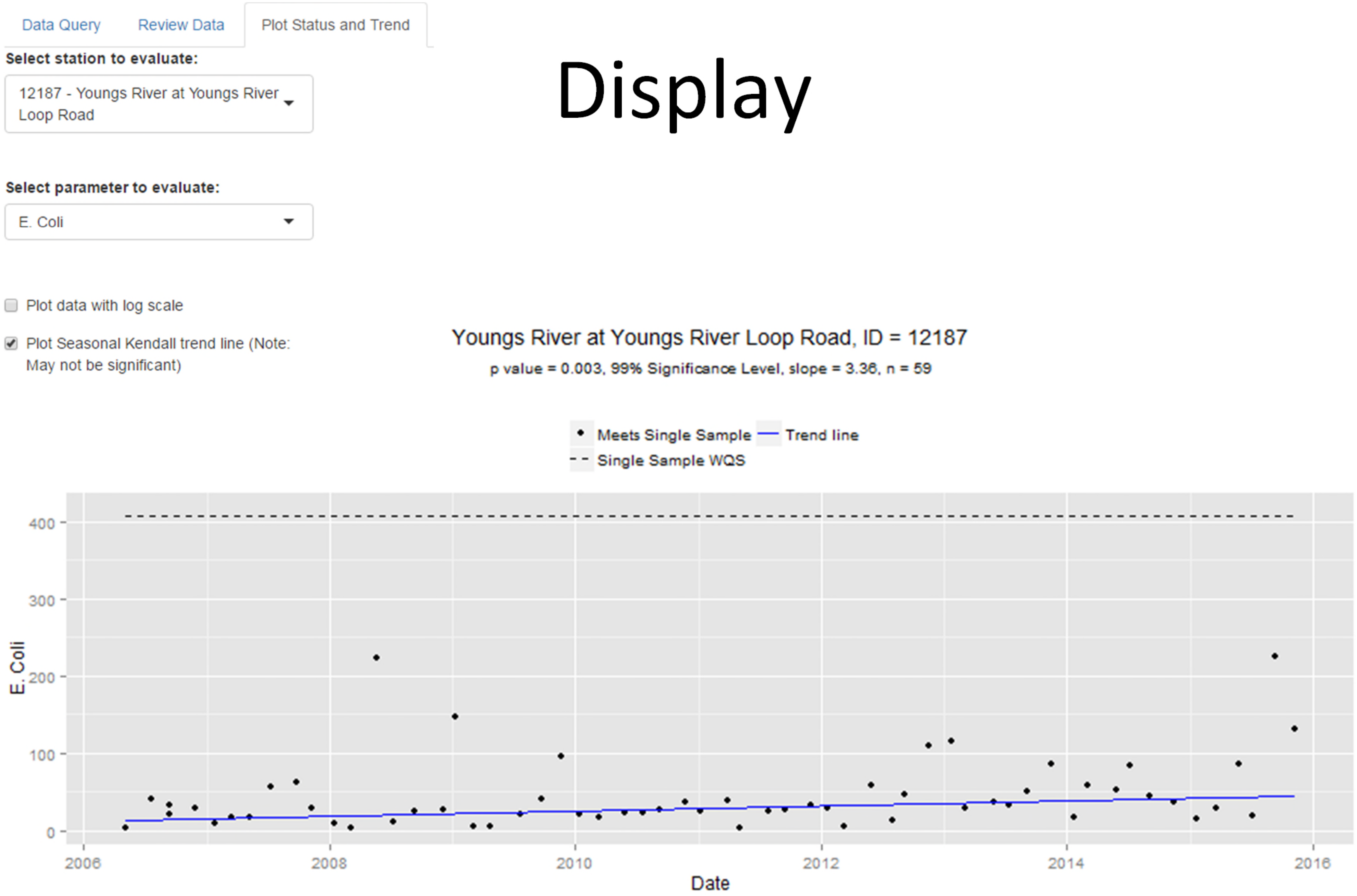
Parameter results by station
Data in tabular format
WQ Limited Waters within Ag Area
QA - Summary by organization
QA - Result values modified
QA - Data removal information
QA - Unique comment values

Show 10 entries

Search:

	Client	Analyte	Station_ID	Station_Description	SampleType	Result	MRL	Unit
		A	Al	All	All			
1	Ambient Water Quality Monitoring - DEQ	E. Coli	11494	Burnt River at Snake River Road (Huntington)	Grab Sample::GS	22.0	1.0	MPN/100 mL
2	Ambient Water Quality Monitoring - DEQ	pH	11494	Burnt River at Snake River Road (Huntington)	Grab Sample::GS	8.5	0.1	pH Units
3	Ambient Water Quality Monitoring - DEQ	Temperature	11494	Burnt River at Snake River Road (Huntington)	Grab Sample::GS	8.3	0.0	°C

Display



Select station to evaluate:

12187 - Youngs River at Youngs River Loop Road

Select parameter to evaluate:

E. Coli

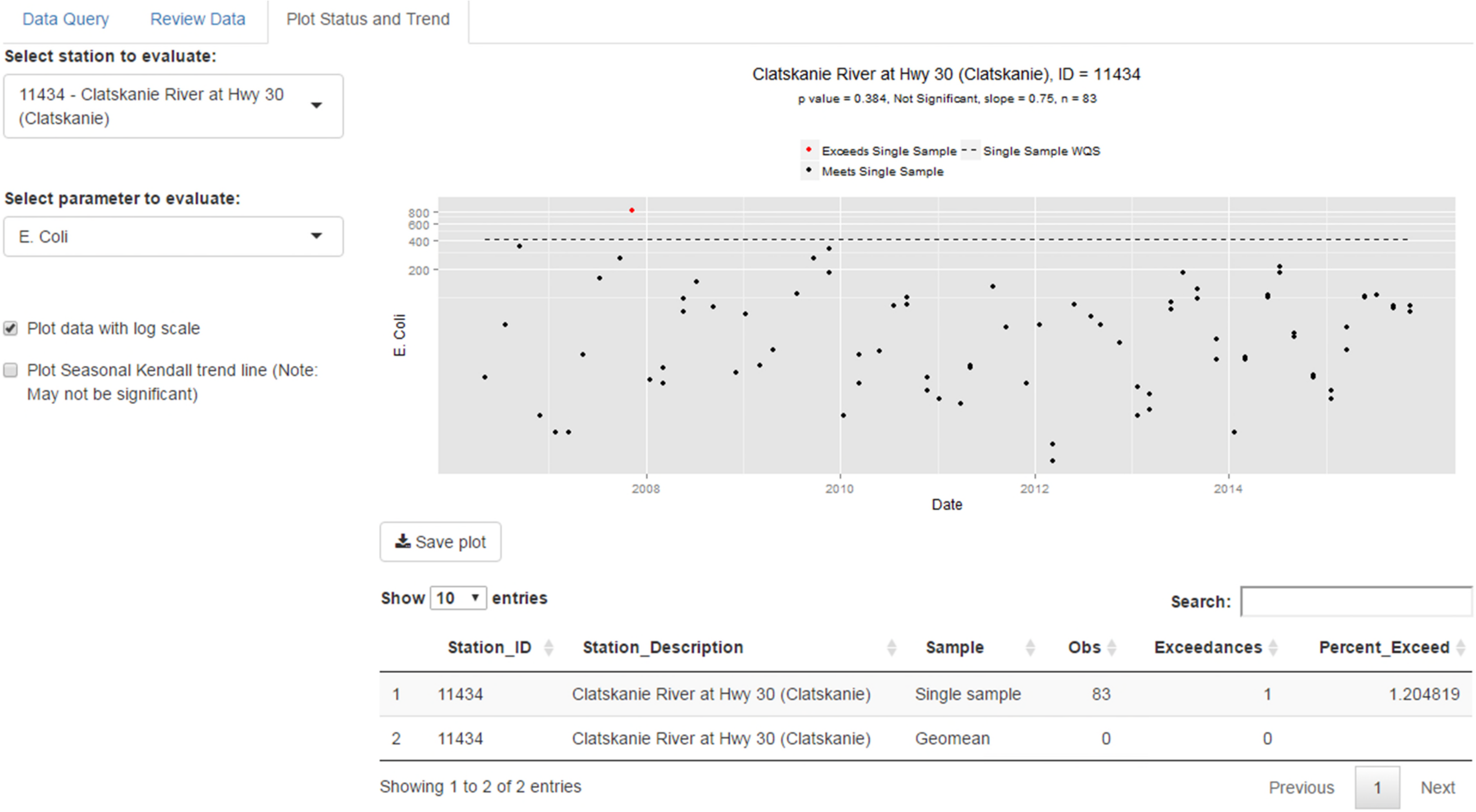
☐ Plot data with log scale

☑ Plot Seasonal Kendall trend line (Note: May not be significant)

Youngs River at Youngs River Loop Road, ID = 12187

p value = 0.003, 99% Significance Level, slope = 3.36, n = 59

Display



Select station to evaluate:

11434 - Clatskanie River at Hwy 30 (Clatskanie)

Select parameter to evaluate:

E. Coli

☑ Plot data with log scale

□ Plot Seasonal Kendall trend line (Note: May not be significant)

Clatskanie River at Hwy 30 (Clatskanie), ID = 11434

p value = 0.384, Not Significant, slope = 0.75, n = 83

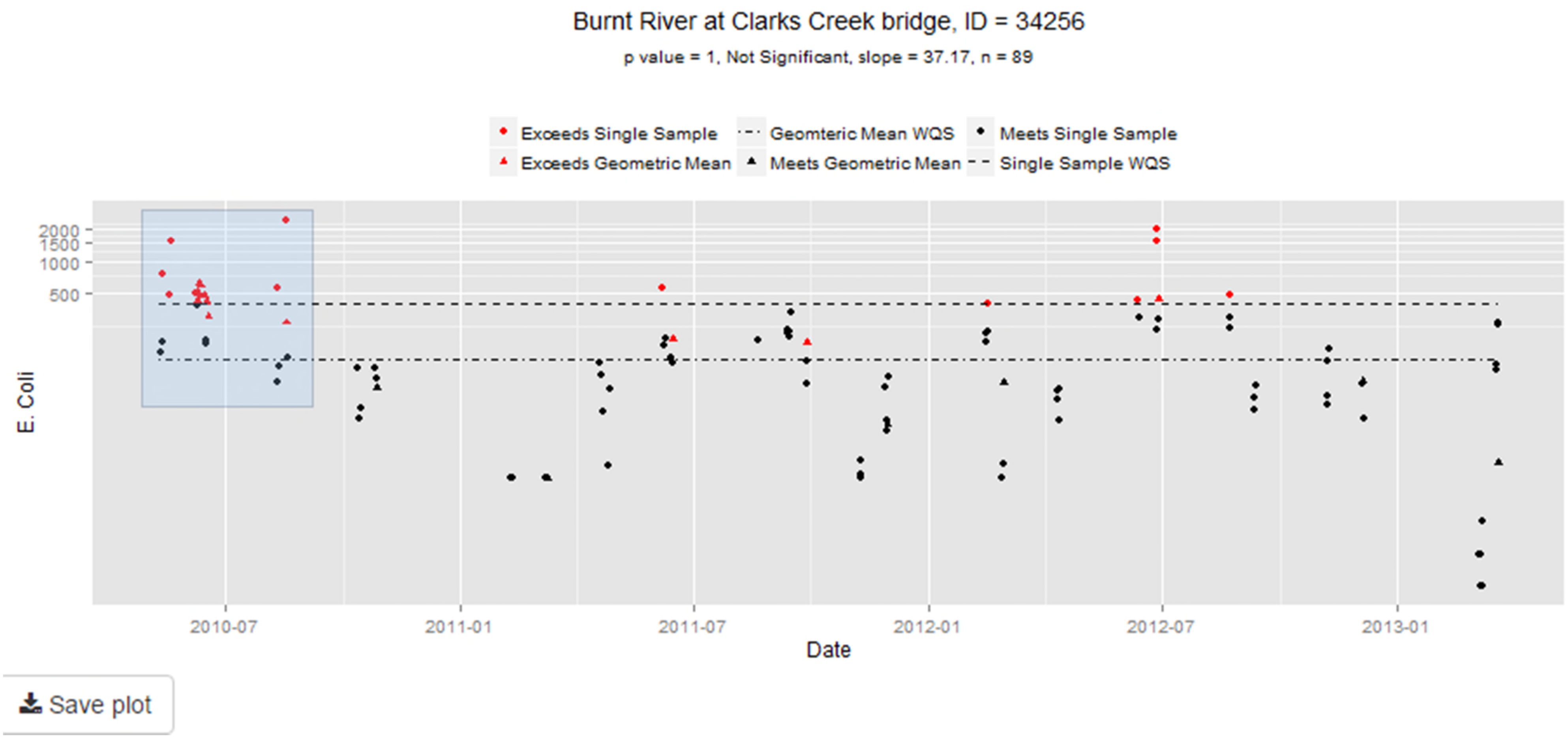
Show 10 **entries**

Search:

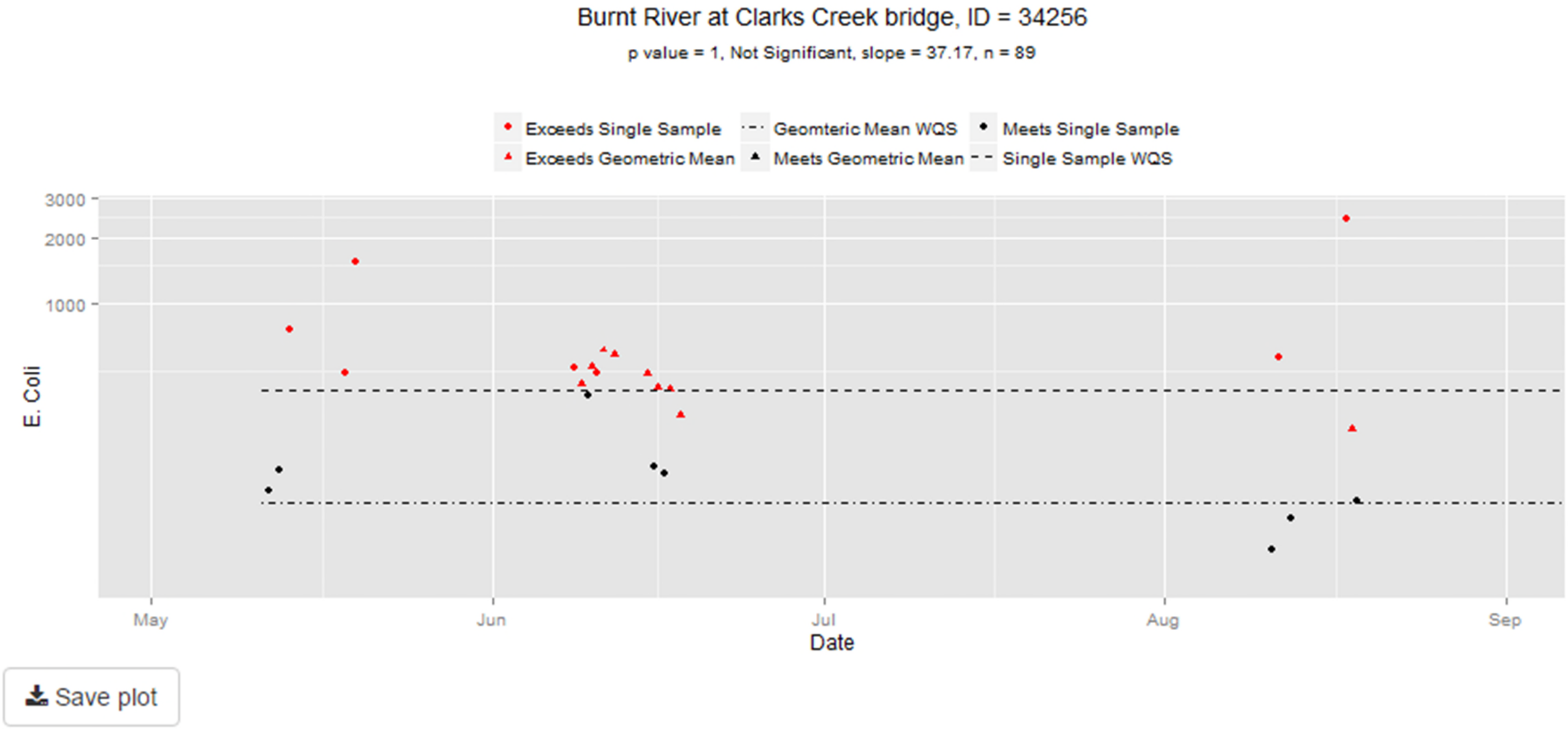
	Station_ID	Station_Description	Sample	Obs	Exceedances	Percent_Exceed
1	11434	Clatskanie River at Hwy 30 (Clatskanie)	Single sample	83	1	1.204819
2	11434	Clatskanie River at Hwy 30 (Clatskanie)	Geomean	0	0	

Showing 1 to 2 of 2 entries

Display


Burnt River at Clarks Creek bridge, ID = 34256
p value = 1, Not Significant, slope = 37.17, n = 89
Exceeds Single Sample
Exceeds Geometric Mean
Geomteric Mean WQS
Meets Geometric Mean
Meets Single Sample
Single Sample WQS
2000
1500
1000
500
E. Coli
2010-07
2011-01
2011-07
2012-01
2012-07
2013-01
Date
Save plot

Display


Burnt River at Clarks Creek bridge, ID = 34256

p value = 1, Not Significant, slope = 37.17, n = 89

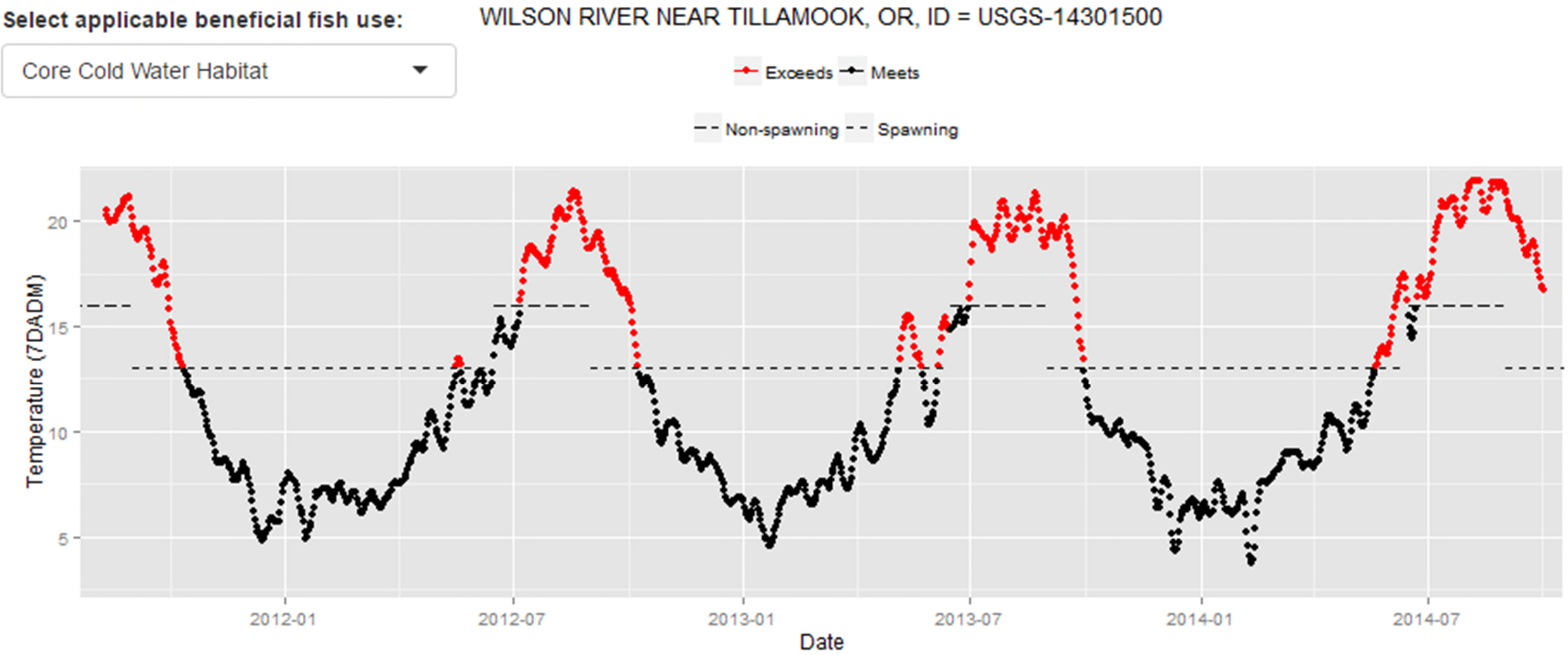
Display

Select applicable spawning time period:

September 1-June 15 ▾

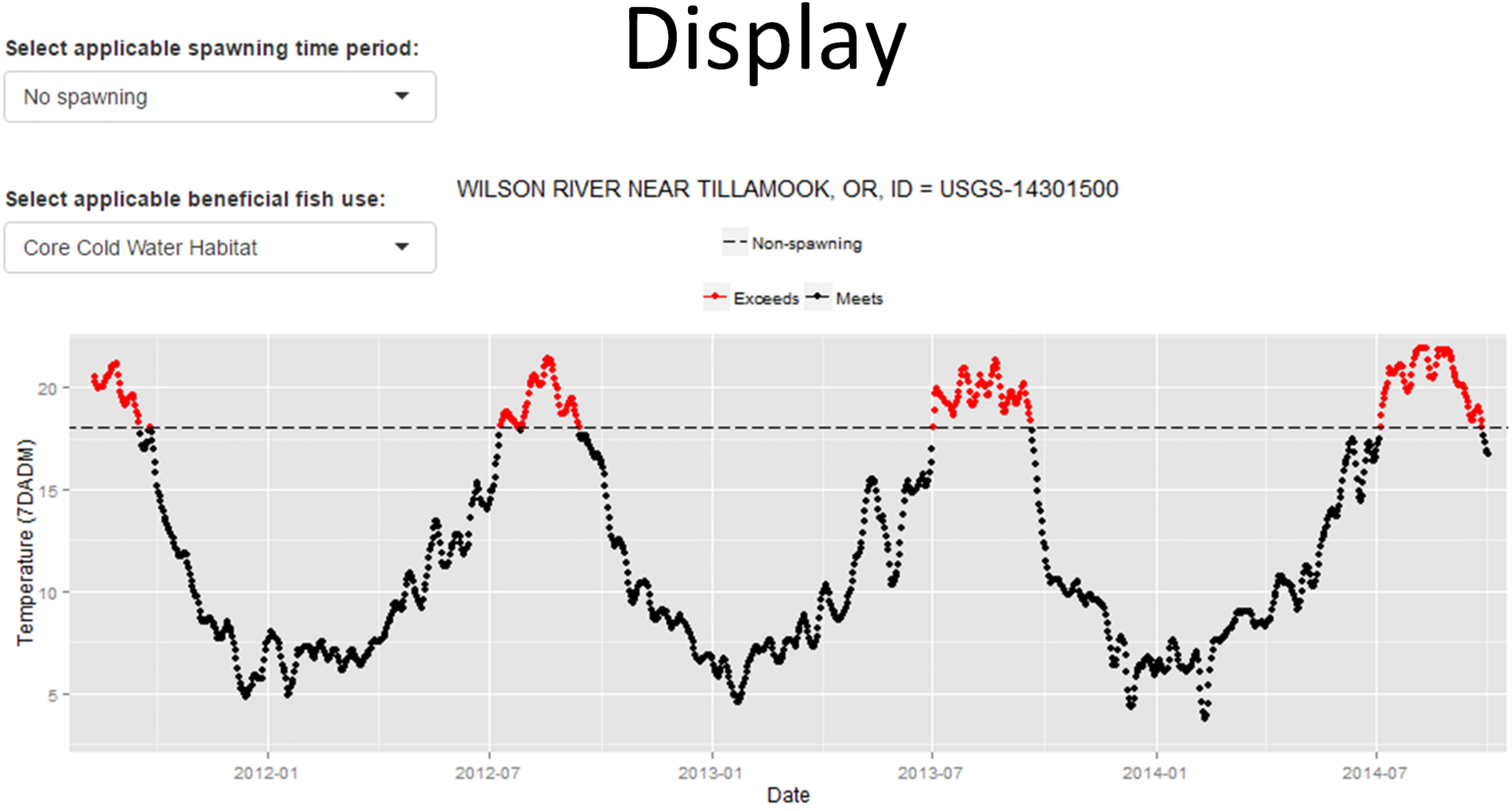
Select applicable beneficial fish use:

Core Cold Water Habitat ▾



DEQ
State of Oregon Department of Environmental Quality

Display



Select applicable spawning time period:

No spawning ▾

Select applicable beneficial fish use:

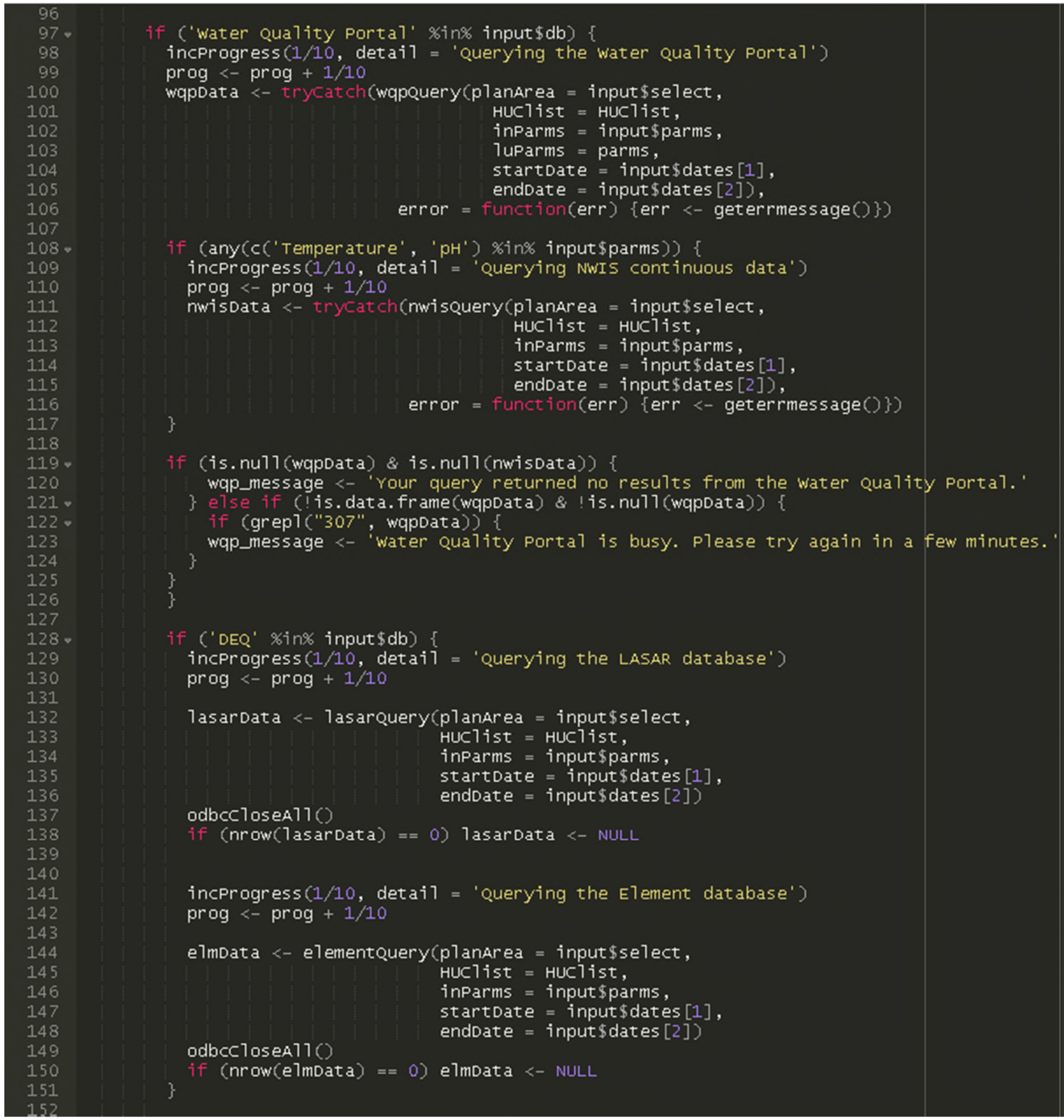
Core Cold Water Habitat ▾

Summary

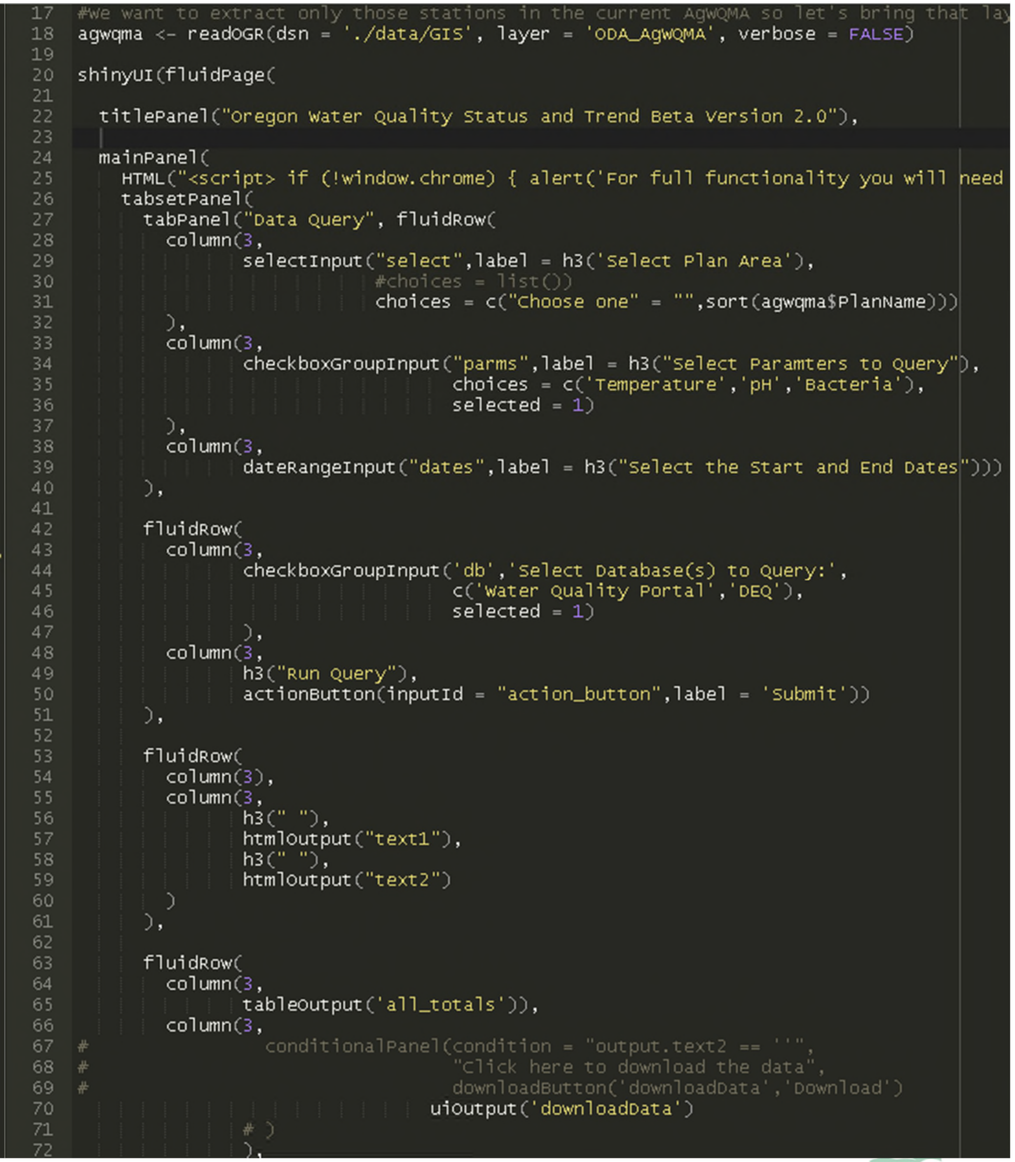
This R based web application is built using the Shiny package in R Studio and:

- Uses database queries to search for data
- R code to facilitate streamlined analysis
- Includes interactive tables and charts

If you *want* to see this



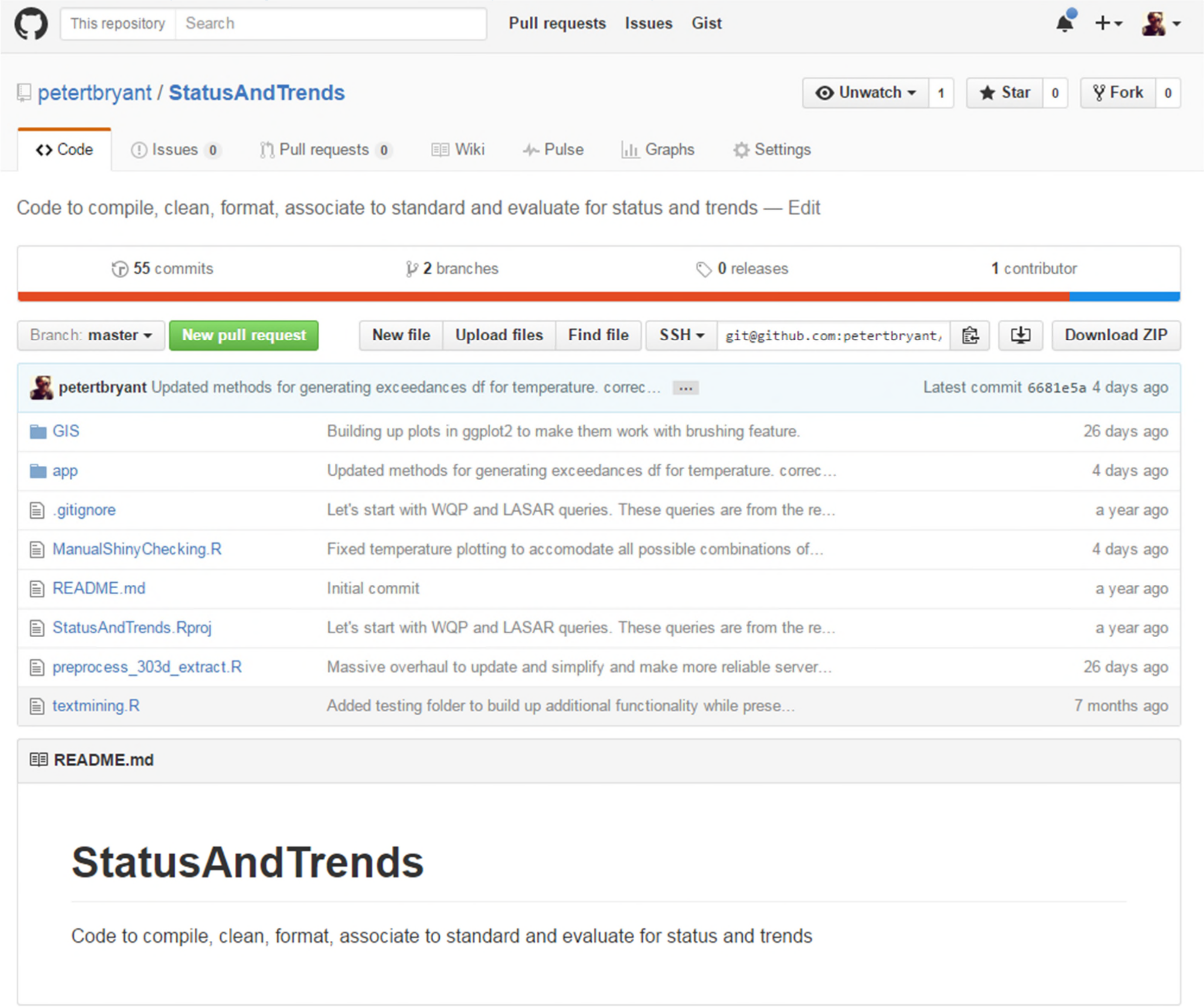
```r
if ('Water Quality Portal' %in% input$db) {
  incProgress(1/10, detail = 'Querying the Water Quality Portal')
  prog <- prog + 1/10
  wqpData <- tryCatch(wqpQuery(planArea = input$select,
                                 HUClist = HUClist,
                                 inParms = input$parms,
                                 luParms = parms,
                                 startDate = input$dates[1],
                                 endDate = input$dates[2]),
                        error = function(err) {err <- geterrmessage()})

  if (any(c('Temperature', 'pH') %in% input$parms)) {
    incProgress(1/10, detail = 'Querying NWIS continuous data')
    prog <- prog + 1/10
    nwisData <- tryCatch(nwisQuery(planArea = input$select,
                                   HUClist = HUClist,
                                   inParms = input$parms,
                                   startDate = input$dates[1],
                                   endDate = input$dates[2]),
                         error = function(err) {err <- geterrmessage()})
  }

  if (is.null(wqpData) & is.null(nwisData)) {
      wqp_message <- 'Your query returned no results from the Water Quality Portal.'
    } else if (!is.data.frame(wqpData) & !is.null(wqpData)) {
      if (grepl("307", wqpData)) {
      wqp_message <- 'Water Quality Portal is busy. Please try again in a few minutes.'
    }
  }
  }

  if ('DEQ' %in% input$db) {
    incProgress(1/10, detail = 'Querying the LASAR database')
    prog <- prog + 1/10

    lasarData <- lasarQuery(planArea = input$select,
                            HUClist = HUClist,
                            inParms = input$parms,
                            startDate = input$dates[1],
                            endDate = input$dates[2])
    odbcCloseAll()
    if (nrow(lasarData) == 0) lasarData <- NULL


    incProgress(1/10, detail = 'Querying the Element database')
    prog <- prog + 1/10

    elmData <- elementQuery(planArea = input$select,
                            HUClist = HUClist,
                            inParms = input$parms,
                            startDate = input$dates[1],
                            endDate = input$dates[2])
    odbcCloseAll()
    if (nrow(elmData) == 0) elmData <- NULL
  }
```



```r
#We want to extract only those stations in the current AgWQMA so let's bring that lay
agwqma <- readOGR(dsn = './data/GIS', layer = 'ODA_AgWQMA', verbose = FALSE)

shinyUI(fluidPage(

  titlePanel("Oregon Water Quality Status and Trend Beta Version 2.0"),

  mainPanel(
    HTML("<script> if (!window.chrome) { alert('For full functionality you will need
    tabsetPanel(
      tabPanel("Data Query", fluidRow(
        column(3,
               selectInput("select",label = h3('Select Plan Area'),
                           #choices = list())
                           choices = c("Choose one" = "",sort(agwqma$PlanName)))
        ),
        column(3,
               checkboxGroupInput("parms",label = h3("Select Paramters to Query"),
                                  choices = c('Temperature','pH','Bacteria'),
                                  selected = 1)
        ),
        column(3,
               dateRangeInput("dates",label = h3("Select the Start and End Dates")))
      ),

      fluidRow(
        column(3,
               checkboxGroupInput('db','Select Database(s) to Query:',
                                  c('Water Quality Portal','DEQ'),
                                  selected = 1)
               ),
        column(3,
               h3("Run Query"),
               actionButton(inputId = "action_button",label = 'Submit'))
      ),

      fluidRow(
        column(3),
        column(3,
               h3(" "),
               htmloutput("text1"),
               h3(" "),
               htmloutput("text2")
        )
      ),

      fluidRow(
        column(3,
               tableOutput('all_totals')),
        column(3,
#                conditionalPanel(condition = "output.text2 == ''",
#                                  "Click here to download the data",
#                                  downloadButton('downloadData','Download')
                                uiOutput('downloadData')
               # )
               ),
```

Go To This Project’s Repository

https://github.com/petertbryant/StatusAndTrends



Code to compile, clean, format, associate to standard and evaluate for status and trends — Edit

55 commits | 2 branches | 0 releases | 1 contributor

git@github.com:petertbryant,

petertbryant Updated methods for generating exceedances df for temperature. correc… … Latest commit 6681e5a 4 days ago

GIS	Building up plots in ggplot2 to make them work with brushing feature.	26 days ago
app	Updated methods for generating exceedances df for temperature. correc…	4 days ago
.gitignore	Let's start with WQP and LASAR queries. These queries are from the re…	a year ago
ManualShinyChecking.R	Fixed temperature plotting to accomodate all possible combinations of…	4 days ago
README.md	Initial commit	a year ago
StatusAndTrends.Rproj	Let's start with WQP and LASAR queries. These queries are from the re…	a year ago
preprocess_303d_extract.R	Massive overhaul to update and simplify and make more reliable server…	26 days ago
textmining.R	Added testing folder to build up additional functionality while prese…	7 months ago

README.md

StatusAndTrends

Code to compile, clean, format, associate to standard and evaluate for status and trends

Thanks to

- Oregon DEQ Laboratory Water Quality Monitoring Group and Steve Hanson, Wade Peerman, Michael Tichenor and Lori Pillsbury

- John Paul Schmit with the National Park Service whose Shiny app provided inspiration and whose Github repository helped get me started

Questions?

Contact info:

bryant.peter@deq.state.or.us

GitHub

- Github is a website where you share your R code with others online
- Github uses the version control software git
- Version control software keeps track of the changes you make to your code