# R Studio Code Essentials

# Hydrologic Data Analysis

## Session Set Up

getwd() 🡪 checks from which file R is drawing the working directory

install.packages(“PACKAGENAME”) 🡪 installs an R package into Rstudio

library(PACKAGENAME) 🡪 loads the R package into the current session/RMarkdown doc

packages <- c(“PACKAGENAME”, “2NDPACKAGENAME”, “ETC”)

🡪 alternate way of loading packages

invisible(suppressPackagesStartupMessages(lapply(packages, library, character.only = TRUE))

🡪 makes the package loading messages invisible in the final, knitted file

theme\_set(THEMENAME()) 🡪 sets the current session’s visualization theme

options(scipen = 100) 🡪 used to avoid e^x notation in ggplot values

lagosne\_get(dest\_folder = LAGOSNE:::lagos\_path(), overwrite = TRUE) 🡪 loads the LAGOS data system. do once per computer

## Load Data

Existing CSV

DATAFRAMENAME <- read.csv(“./FOLDER/SUBFOLDER/ETC/FILENAME.csv”) 🡪 pull a CSV file directly from working directory files

NWIS

DATAFRAMENAME <- readNWISdv(siteNumbers = “NUMBER”,

parameterCd = “CODE”, 🡪 =

startDate = “YYYY-MM-DD”,

endDate = “YYYY-MM-DD”) 🡪Import DAILY VALUES

readNWISqw 🡪 water quality factors

readNWISuv 🡪 high frequency data points

LAGOS

DATAFRAMENAME <- lagosne\_load() 🡪 load LAGOS data into the session

## Data Wrangling

Renaming Columns

names(DATAFRAMENAME)[COLUMN#:COLUMN2#] <- c(“NEWCOLUMNNAME”, “NEWCOLUMN2NAME”) 🡪 rename certain columns

%>% rename(NEWCOLUMNNAME = ‘OLDCOLUMNNAME’, NEWCOLUMN2NAME = ‘OLDCOLUMN2NAME’)

Add Date Columns

%>% mutate(NEWCOLUMNNAME = month(DATECOLUMNNAME)) 🡪 create a column with just the month number

%>% mutate(NEWCOLUMNNAME = year(DATECOLUMNNAME)) 🡪 create a column with just the year number

%>% mutate(NEWCOLUMNNAME = yday(DATECOLUMNNAME)) 🡪 create a column with the day of the year

%>% mutate(NEWCOLUMNNAME = as.factor(quarter(DATECOLUMNNAME, fiscal\_start = 12))) 🡪 create a column with the season

Add Any Other Columns

%>% mutate(NEWCOLUMNNAME = EQUATION)

%>% mutate(NEWCOLUMNNAME = FUNCTION(COLUMNNAME)) 🡪 both are general guides to adding new columns to the data frame

Include Only Certain Rows/Columns

%>% select(COLUMNNAME, COLUMN2NAME, ETC) 🡪 only include certain columns

%>% filter(COLUMNNAME == VALUESEENINCOLUMN & COLUMN2NAME < VALUESEENINCOLUMN2) 🡪 only include certain rows

%>% dropna() 🡪 drop any and all rows that include even one N/A value

Organize Data

%>% group\_by(COLUMNNAME) 🡪 groups the data by the values in the specified column

%>% arrange(COLUMNNAME) 🡪 arranges the data by the specified column

%>% spread(key = COLUMNNAME, value = COLUMN2NAME) 🡪 tells R to put values in column2 into values in column1

%>% summarize(COLUMNNAME = mean/max/min(COLUMNNAME)) 🡪 summarizes the specified column in the specified way

%>% summarize\_all(mean/min/max) 🡪 summarizes all columns in the specified way

Join Data Frames

NEWDATAFRAMENAME <- full\_join(DATAFRAME, DATAFRAME2, by = “COLUMNNAME”) %>%

full\_join(. , DATAFRAME3, by = “COLUMNNAME”) %>%

full\_join(. , ETC, by = “COLUMNNAME”) 🡪 join multiple data frames by a single column

NEWDATAFRAMENAME <- left\_join(DATAFRAMENAME, DATAFRAME2NAME, by “COLUMNNAME”) 🡪 joins all the columns from the RIGHT data frame to the LEFT one, with the listed column not repeated

## Data Visualization

GGPlot

PLOTNAME <-

ggplot(DATAFRAMENAME, aes(x = COLUMNNAME, y = COLUMNNAME)) +

geom\_histogram()/geom\_bar()/geom\_line()/geom\_point(alpha = POINTTRANSPARENCYINDECIMALS, color = “COLORCODE”, size = #SIZEOFPOINTS) +

geom\_vline(xintercept = #) + 🡪 add VERTICAL line on the graph

geom\_hline(yintercept = #) + 🡪 add HORIZONTAL line on the graph

scale\_x\_continuous(position = “top/bottom”) +

scale\_fill\_viridis\_d(option = “COLORSCALENAME”) +

ggtitle(“GRAPHTITLE”) +

labs(x = “X-AXISLABEL”, y = “Y-AXISLABEL”) +

theme(plot.title = element\_text(), legend.position = “none”, axis.text.y = element\_blank()) + 🡪 customize the graph title/legend/axis aesthetics

geom\_smooth() + 🡪 smooth the graph

scale\_x\_log10() + 🡪 log transform x values

scale\_y\_log10() 🡪 log transform y values

print(PLOTNAME) 🡪 prints the specified plot

DyGraphs

DyDat <- cbind(DATAFRAME, DATAFRAME2)

dygraph <- DyDat) %>%

dySeries(“DATAFRAME”, axis = “y2”) %>%

dyAxis(name = “y”, label = “AXISNAME”) %>%

dyAxis(name = “y2”, label = “OTHERAXISNAME”) %>%

dyRangeSelector() 🡪 makes an interactive dygraph

Time Series

TIMESERIES <- ts(DATAFRAMENAME[[COLUMN#]], frequency = #DAYFREQUENCY)

TSDECOMPOSED <- stl(TIMESERIES, s.window = “periodic”)

TSCOMPONENTS <- as.data.frame(TSDECOMPOSED$TSCOLUMN[,#:#]) %>%

mutate(COLUMNNAME = DATAFRAMENAME$COLUMN,

COLUMN2NAME = DATAFRAMENAME$COLUMN2) 🡪 plot all of the individual component variables of the time series

Combine Graphs

COMBINEDGRAPH <-

ggplot(PLOT) +

geom\_PLOTTYPE()

geom\_PLOT2TYPE(data = PLOT2) +

geom\_ETCTYPE(data = ETC)

COMBINEDGRAPH <-

plot\_grid(PLOT, PLOT2, PLOT3, ETC,

ncol = #COLUMNSINGRID, rel\_widths = c(#, #2, #3, #4))

## Statistical Tests

Linear Regression

LINEARREGRESSIONDATAFRAME <- lm(data = DATAFRAMENAME, DATACOLUMN ~ VARIABLECOLUMN + VARIABLE2COLUMN + ETCCOLUMN)

summary(LINEARREGRESSIONDATAFRAME) 🡪 view the regression model

vif(LINEARREGRESSIONDATAFRAME) 🡪 calculate variance inflation factors for the variables in the model

steo(LINEARREGRESSIONDATAFRAME) 🡪 calculate AIC values for the model

QQNorm

qqnorm(VECTOR, las = #); qqline(VECTOR)

qqnorm(VECTOR2, las = 3); qqline(VECTOR2)

T-Test

cat(“NAME:”, length(VECTOR), “\n”, “NAME2”, length(VECTOR2))

t.test(VECTOR, VECTOR2)