```
# Creating a dataframe
# Example: RPI Weather dataframe.

days <- c('Mon', 'Tue','Wed','Thur','Fri','Sat','Sun') #  days
temp <- c(28,30.5,32,31.2,29.3,27.9,26.4) # Temperature in F' during the winter :)
snowed <- c('T','T','F','F','T','T','T') # Snowed on that day: T = TRUE, F= FALSE
snowed
class(snowed)
help("data.frame")
```

```
'T' · 'T' · 'F' · 'F' · 'T' · 'T' · 'T'
'character'
```

```
emptyDataframe <- data.frame()
emptyDataframe

RPI_Weather_Week <- data.frame(days,temp,snowed) # creating the dataframe using the data.frame() function

RPI_Weather_Week
head(RPI_Weather_Week)
```

A data.frame: 7 × 3

days	temp	snowed
<chr>	<dbl>	<chr>
Mon	28.0	T
Tue	30.5	T
Wed	32.0	F
Thur	31.2	F
Fri	29.3	T
Sat	27.9	T
Sun	26.4	T

A data.frame: 6 × 3

	days	temp	snowed
	<chr>	<dbl>	<chr>
1	Mon	28.0	T
2	Tue	30.5	T
3	Wed	32.0	F
4	Thur	31.2	F
5	Fri	29.3	T
6	Sat	27.9	T

```

str(RPI_Weather_Week) # we can take a look at the structure of the dataframe using the str() function.

summary(RPI_Weather_Week) # summary of the dataframe using the summary() function
summary((RPI_Weather_Week$temp))

```

```
'data.frame':   7 obs. of  3 variables:
 $ days  : chr  "Mon" "Tue" "Wed" "Thur" ...
```

```
1
RPI_Weather_Week[1,] # showing the 1st row and all the columns
RPI_Weather_Week[,1] # showing the 1st coulmn and all the rows

RPI_Weather_Week[,'snowed']
RPI_Weather_Week[,'days']
RPI_Weather_Week[,'temp']
RPI_Weather_Week[1:5,c("days","temp")]
RPI_Weather_Week$temp
subset(RPI_Weather_Week,subset=snowed==TRUE)
```

A data.frame: 1 × 3

	days	temp	snowed
	<chr>	<dbl>	<chr>
1	Mon	28	T

'Mon' · 'Tue' · 'Wed' · 'Thur' · 'Fri' · 'Sat' · 'Sun'
'T' · 'T' · 'F' · 'F' · 'T' · 'T' · 'T'
'Mon' · 'Tue' · 'Wed' · 'Thur' · 'Fri' · 'Sat' · 'Sun'
28 · 30.5 · 32 · 31.2 · 29.3 · 27.9 · 26.4

A data.frame: 5 × 2

	days	temp
	<chr>	<dbl>
1	Mon	28.0
2	Tue	30.5
3	Wed	32.0
4	Thur	31.2
5	Fri	29.3

28 · 30.5 · 32 · 31.2 · 29.3 · 27.9 · 26.4

A data.frame: 0 × 3

days	temp	snowed
<chr>	<dbl>	<chr>

```
dec.snow <- order(-RPI_Weather_Week$temp)
dec.snow
# Creating Dataframes
# creating an empty dataframe
empty.DataFrame <- data.frame()

```

3 · 4 · 2 · 5 · 1 · 6 · 7

```
v1 <- 1:10
v1
letters
v2 <- letters[1:10]
v2
df <- data.frame(col_name.1 = v1,col.name.2 = v2)
df
# importing data and exporting data
# writing to a CSV file:
write.csv(df,file = 'saved_df1_Section2.csv')
df_section2 <- read.csv('saved_df1_Section2.csv')
df_section2
```

1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10
'a' · 'b' · 'c' · 'd' · 'e' · 'f' · 'g' · 'h' · 'i' · 'j' · 'k' · 'l' · 'm' · 'n' · 'o' · 'p' · 'q' · 'r' · 's' · 't' · 'u' · 'v' · 'w' · 'x' · 'y' · 'z'
'a' · 'b' · 'c' · 'd' · 'e' · 'f' · 'g' · 'h' · 'i' · 'j'

A data.frame: 10 × 2

col_name.1 <int>	col.name.2 <chr>
1	a
2	b
3	c
4	d
5	e
6	f
7	g
8	h
9	i
10	j

A data.frame: 10 × 3

X <int>	col_name.1 <int>	col.name.2 <chr>
1	1	a
2	2	b
3	3	c

```
1 nrow(df)
2 ncol(df)
3 colnames(df)
4 rownames(df)
5 str(df)
6 summary(df)
```

```
10
2
'col_name.1' · 'col.name.2'
'1' · '2' · '3' · '4' · '5' · '6' · '7' · '8' · '9' · '10'
'data.frame':   10 obs. of  2 variables:
 $ col_name.1: int  1 2 3 4 5 6 7 8 9 10
 $ col.name.2: chr  "a" "b" "c" "d" ...
  col_name.1     col.name.2
 Min.   : 1.00   Length:10
 1st Qu.: 3.25   Class :character
 Median : 5.50   Mode  :character
 Mean   : 5.50
 3rd Qu.: 7.75
 Max.   :10.00
```

```
1 df[[5,2]]
2 df[[5,'col.name.2']]
3 df[[3,'col.name.1']]
4 df[[4,'col.name.2']]
5 df[[3,'col.name.2']] <- 3000
6 df
7
```

```
'e'
'e'
NULL
'd'
```

A data.frame: 10 × 2

col_name.1 <int>	col.name.2 <chr>
1	a
2	b

```
1
```

```
persons.df <- data.frame(FirstName = 'Vasundhara', LastName ='Acharya', ID = 123456, Instituete = 'RPI')
persons.df


```

A data.frame: 1 × 4

FirstName <chr>	LastName <chr>	ID <dbl>	Instituete <chr>
Vasundhara	Acharya	123456	RPI

```
Age <- c(27,25,26,28)
Weight <- c(155,175,130,155)
Sex <- c('M','M','F','O')
Names <- c('Mike','Jason','Silvia','Dell')
people <- data.frame(Names, Age = Age, Weight = Weight, Sex = Sex)
people2 <- data.frame(Names, Age, Weight, Sex)
people2

df <- data.frame(row.names = Names, Age, Weight,Sex)
df
is.na(df)
is.data.frame(df)
# names <- data.frame(Age,Weight,Sex)
# names

mat <- matrix(1:25,nrow = 5)
mat2 <- as.data.frame(mat)
is.data.frame(mat2)

```

A data.frame: 4 × 4

```
# Lists
vec <- c(6,7,8) # vector
mat <- matrix(1:30, nrow = 5) # matrix
mat
class(vec)
class(mat)
# if you want to include different data structures in to one single variable, you can use the list()
# list() function allow us to combine different datastructure into a single variable.
my.list <- list(vec,mat,df)
my.list
# instead of having automatically numbered, we can name the item in the list as follows:
my.named.list <- list(sampleVec = vec, SampleMatrix = mat, SampleDataFrame = df)
my.named.list

# List is more like an organizational tool, you can organize various dataframes
# One advantage is, you can call the items in the list using the $ sign to call them as you call them like coulums.
my.named.list$sampleVec
my.named.list$SampleMatrix
my.named.list$SampleDataFrame
```

A matrix: 5 × 6 of type int

1	6	11	16	21	26
2	7	12	17	22	27
3	8	13	18	23	28
4	9	14	19	24	29
5	10	15	20	25	30

'numeric'
'matrix' · 'array'

1. 6 · 7 · 8
2. A matrix: 5 × 6 of type int

1	6	11	16	21	26
2	7	12	17	22	27
3	8	13	18	23	28
4	9	14	19	24	29
5	10	15	20	25	30

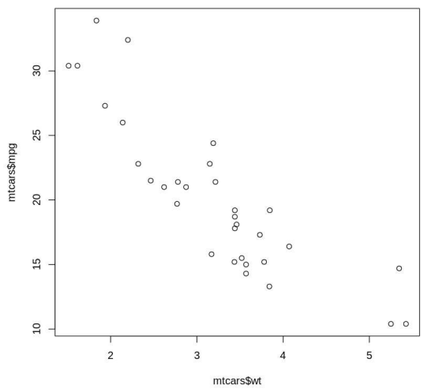
3. A data.frame: 4 × 3

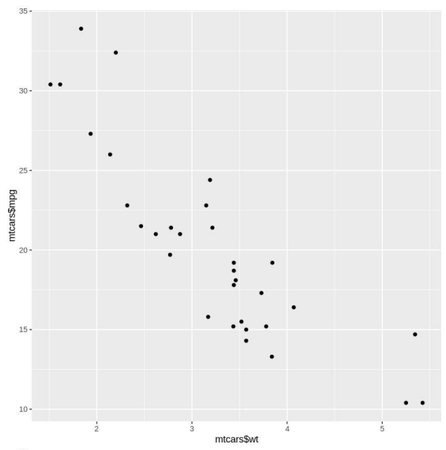
	Age	Weight	Sex

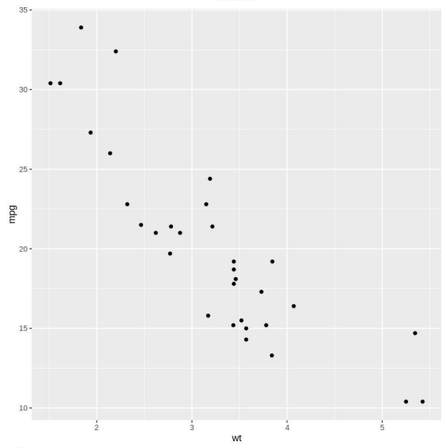
```
#########PLOTTING EXERCISES###########
data(mtcars)
```

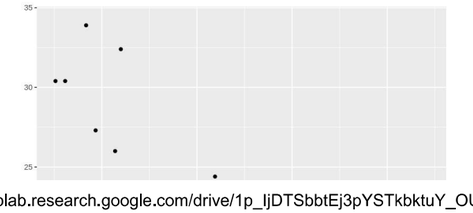
```
plot (mtcars$wt, mtcars$mpg)
library (ggplot2)
qplot (mtcars$wt, mtcars$mpg)
qplot (wt, mpg, data = mtcars)
ggplot (mtcars, aes ( x=wt,y=mpg) ) + geom_point ()
plot (pressure$temperature,pressure$pressure, type = "l")
points (pressure$temperature,pressure$pressure)
lines (pressure$temperature,pressure$pressure/2, col="red")
points (pressure $ temperature,pressure $ pressure /2, col="blue")
library (ggplot2)
qplot (pressure $temperature,pressure$pressure, geom="line")
qplot (temperature,pressure, data = pressure, geom = "line")
ggplot(pressure, aes ( x= temperature,y=pressure)) + geom_line () + geom_point()
ggplot(pressure, aes ( x= temperature, y= pressure ))+ geom_line ()+ geom_point ()
##########3WE NOTE THAT AS PRESSURE INCREASES, TEMPERATURE ALSO INCREASES
```

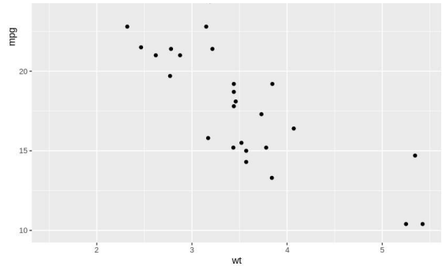
```
Warning message:
“`qplot()` was deprecated in ggplot2 3.4.0.”
```

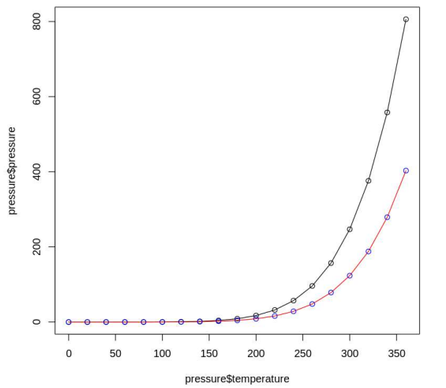


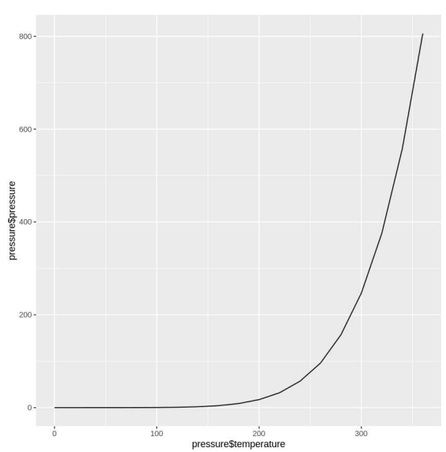


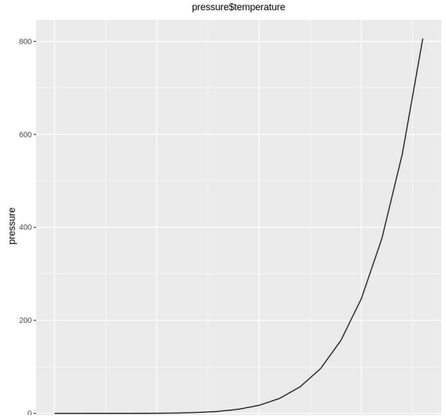


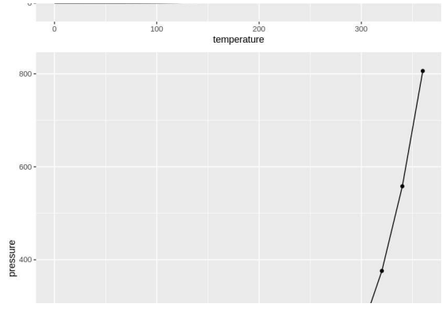



mpg
20
15
10
2
3
4
5
wt


pressure$pressure
800
600
400
200
0
0
50
100
150
200
250
300
350
pressure$temperature

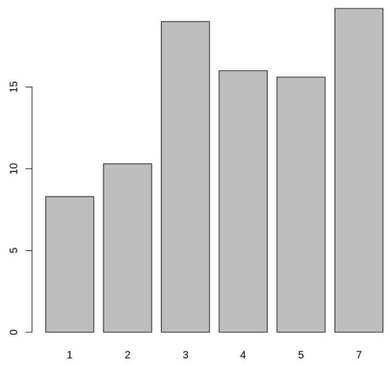

pressure$pressure
800
600
400
200
0
0
100
200
300
pressure$temperature


pressure
800
600
400
200
0

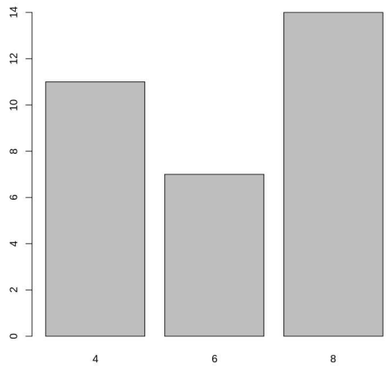


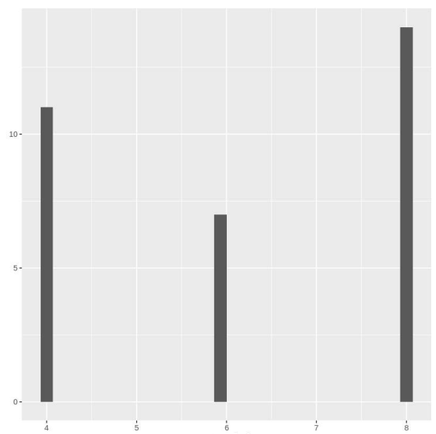

```
##################CREATING THE BAR GRAPHS###########
# Creating Bar graphs
barplot (BOD$ demand, names.arg = BOD $ Time )
table (mtcars $cyl)
barplot(table(mtcars $cyl) ) # generate a table of counts
qplot(mtcars$cyl) # cyl is continous here
qplot(factor (mtcars$cyl)) # treat cyl as discrete
# Bar graph of counts
qplot (factor (cyl), data = mtcars)
ggplot (mtcars, aes (x=factor (cyl)))+ geom_bar ()

```

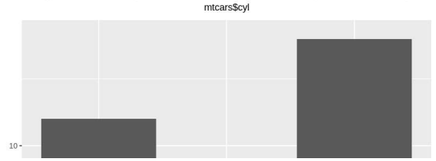
```
 4  6  8
11  7 14
```



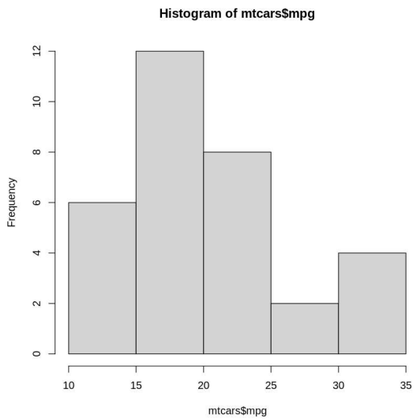

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

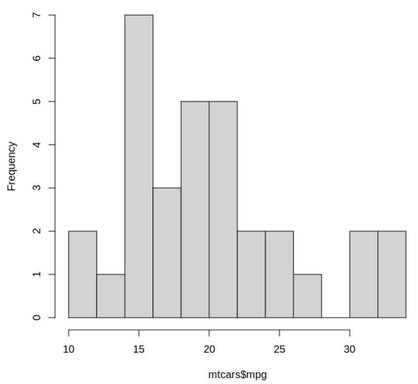


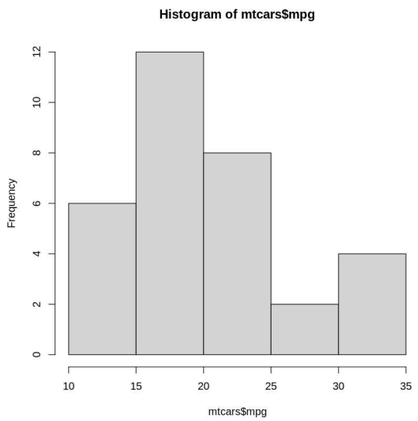




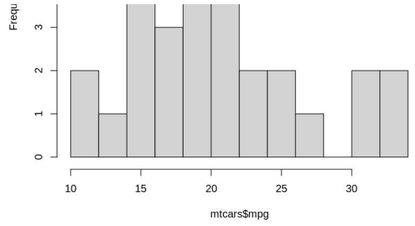

```
#######WE NOTE THAT  CAR WITH 15 miles per gallon ARE IN MORE COUNT THAN OTHERS######
hist(mtcars$mpg)
hist(mtcars$mpg, breaks = 10) # specify apprioximate number
hist (mtcars$mpg, breaks =5 )
hist (mtcars$mpg, breaks =12 )
qplot (mpg, data = mtcars, binwidth=4)
ggplot (mtcars, aes (x=mpg) ) + geom_histogram (binwidth =4 )
ggplot(mtcars, aes (X=mpg))+ geom_histogram (binwidth =5)

```

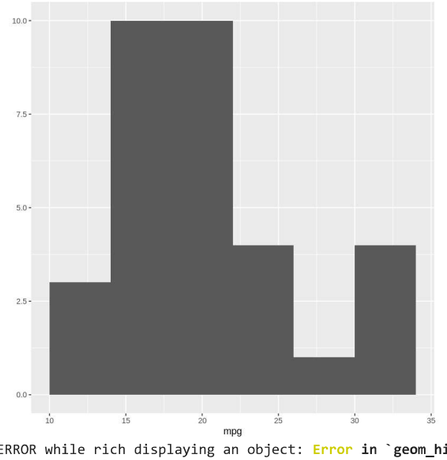

Histogram of mtcars$mpg
Frequency
0
2
4
6
8
10
12
10
15
20
25
30
35
mtcars$mpg


Histogram of mtcars$mpg
Frequency
0
1
2
3
4
5
6
7
10
15
20
25
30
mtcars$mpg


Histogram of mtcars$mpg
Frequency
0
2
4
6
8
10
12
10
15
20
25
30
35
mtcars$mpg


Histogram of mtcars$mpg
4
5
6
7






```
ERROR while rich displaying an object: Error in `geom_histogram()`:
! Problem while computing stat.
i Error occurred in the 1st layer.
Caused by error in `setup_params()`:
! `stat_bin()` requires an x or y aesthetic.
```

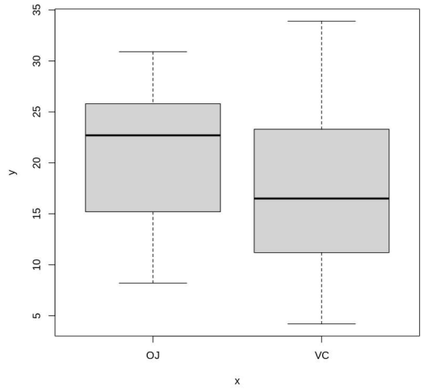
```
Traceback:
1. tryCatch(withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler), error = outer_handler)
2. tryCatchList(expr, classes, parentenv, handlers)
3. tryCatchOne(expr, names, parentenv, handlers[[1L]])
4. doTryCatch(return(expr), name, parentenv, handler)
5. withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler)
6. repr::mime2repr[[mime]](obj)
7. repr_text.default(obj)
8. paste(capture.output(print(obj)), collapse = "\n")
9. capture.output(print(obj))
10. withVisible(...elt(i))
11. print(obj)
12. print.ggplot(obj)
13. ggplot_build(x)
14. ggplot_build.ggplot(x)
15. by_layer(function(l, d) l$compute_statistic(d, layout), layers,
  .     data, "computing stat")
16. try_fetch(for (i in seq_along(data)) {
  .     out[[i]] <- f(l = layers[[i]], d = data[[i]])
  . }, error = function(cnd) {
  .     cli::cli_abort(c("Problem while {step}.", i = "Error occurred in the {ordinal(i)} layer."),
  .         call = layers[[i]]$constructor, parent = cnd)
  . })
17. tryCatch(withCallingHandlers(expr, condition = function(cnd) {
  .     {
  .         .__handler_frame__. <- TRUE
  .         .__setup_frame__. <- frame
  .         if (inherits(cnd, "message")) {
```

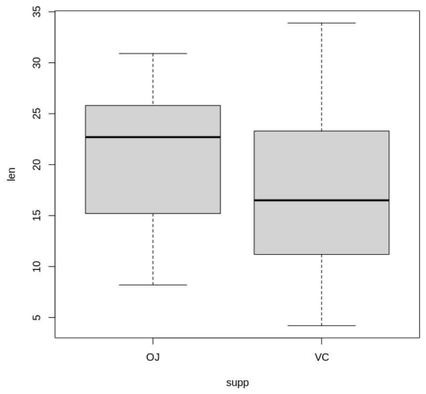
```
.            except <- c("warning", "error")
.        }
.        else if (inherits(cnd, "warning")) {
.            except <- "error"
.        }
.        else {
.            except <- ""
.        }
.    }
.    while (!is_null(cnd)) {
.        if (inherits(cnd, "error")) {
.            out <- handlers[[1L]](cnd)
.            if (!inherits(out, "rlang_zap"))
.                throw(out)
.        }
.        inherit <- .subset2(.subset2(cnd, "rlang"), "inherit")
.        if (is_false(inherit)) {
.            return()
.        }
.        cnd <- .subset2(cnd, "parent")
.    }
. }), stackOverflowError = handlers[[1L]])
18. tryCatchList(expr, classes, parentenv, handlers)
19. tryCatchOne(expr, names, parentenv, handlers[[1L]])
20. doTryCatch(return(expr), name, parentenv, handler)
21. withCallingHandlers(expr, condition = function(cnd) {
.    {
.        .__handler_frame__. <- TRUE
.        .__setup_frame__. <- frame
.        if (inherits(cnd, "message")) {
.            except <- c("warning", "error")
.        }
.        else if (inherits(cnd, "warning")) {
.            except <- "error"
.        }
.        else {
.            except <- ""
.        }
.    }
.    while (!is_null(cnd)) {
.        if (inherits(cnd, "error")) {
.            out <- handlers[[1L]](cnd)
.            if (!inherits(out, "rlang_zap"))
.                throw(out)
.        }
.        inherit <- .subset2(.subset2(cnd, "rlang"), "inherit")
.        if (is_false(inherit)) {
.            return()
.        }
.        cnd <- .subset2(cnd, "parent")
.    }
. })
22. f(l = layers[[i]], d = data[[i]])
23. l$compute_statistic(d, layout)
24. compute_statistic(..., self = self)
25. self$stat$setup_params(data, self$stat_params)
26. setup_params(..., self = self)
27. cli::cli_abort("{.fn {snake_class(self)}} requires an {.field x} or {.field y} aesthetic.")
28. rlang::abort(message, ..., call = call, use_cli_format = TRUE,
.    .frame = .frame)
29. signal_abort(cnd, .file)
30. signalCondition(cnd)
31. (function (cnd)
. {
.    {
.        .__handler_frame__. <- TRUE
.        .__setup_frame__. <- frame
.        if (inherits(cnd, "message")) {
.            except <- c("warning", "error")
.        }
.        else if (inherits(cnd, "warning")) {
.            except <- "error"
.        }
.        else {
.            except <- ""
.        }
.    }
.    while (!is_null(cnd)) {
.        if (inherits(cnd, "error")) {
.            out <- handlers[[1L]](cnd)
.            if (!inherits(out, "rlang_zap"))
.                throw(out)
.        }
.        inherit <- .subset2(.subset2(cnd, "rlang"), "inherit")
.        if (is_false(inherit)) {
```

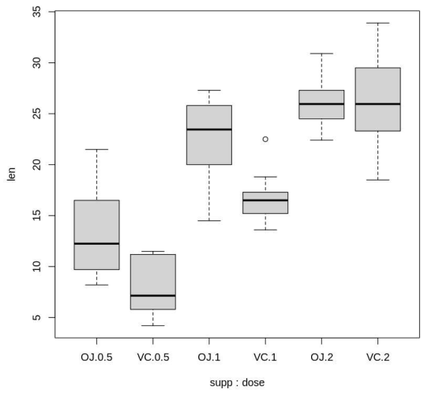
```
.         return()
.       }
.       cnd <- .subset2(cnd, "parent")
.     }
. })(structure(list(message = "`stat_bin()` requires an \033[32mx\033[39m or \033[32my\033[39m aesthetic.",
.     trace = structure(list(call = list(IRkernel::main(), kernel$run(),
.         handle_shell(), executor$execute(msg), tryCatch(evaluate(request$content$code,
.             envir = .GlobalEnv, output_handler = oh, stop_on_error = 1L),
.             interrupt = function(cond) {
.                 log_debug("Interrupt during execution")
.                 interrupted <<- TRUE
.             }, error = .self$handle_error), tryCatchList(expr,
.             classes, parentenv, handlers), tryCatchOne(tryCatchList(expr,
.             names[-nh], parentenv, handlers[-nh]), names[nh],
.             parentenv, handlers[[nh]]), doTryCatch(return(expr),
.             name, parentenv, handler), tryCatchList(expr, names[-nh],
.             parentenv, handlers[-nh]), tryCatchOne(expr, names,
.             parentenv, handlers[[1L]]), doTryCatch(return(expr),
.             name, parentenv, handler), evaluate(request$content$code,
.             envir = .GlobalEnv, output_handler = oh, stop_on_error = 1L),
.         evaluate_call(expr, parsed$src[[i]], envir = envir, enclos = enclos,
.             debug = debug, last = i == length(out), use_try = stop_on_error !=
.                 2L, keep_warning = keep_warning, keep_message = keep_message,
.             log_echo = log_echo, log_warning = log_warning, output_handler = output_handler,
.             include_timing = include_timing), handle(pv <- withCallingHandlers(withVisible(value_fun(ev$value,
.             ev$visible)), warning = wHandler, error = eHandler,
.             message = mHandler)), try(f, silent = TRUE), tryCatch(expr,
.             error = function(e) {
.                 call <- conditionCall(e)
.                 if (!is.null(call)) {
.                   if (identical(call[[1L]], quote(doTryCatch)))
.                     call <- sys.call(-4L)
.                   dcall <- deparse(call, nlines = 1L)
.                   prefix <- paste("Error in", dcall, ": ")
.                   LONG <- 75L
.                   sm <- strsplit(conditionMessage(e), "\n")[[1L]]
.                   w <- 14L + nchar(dcall, type = "w") + nchar(sm[1L],
.                     type = "w")
.                   if (is.na(w))
.                     w <- 14L + nchar(dcall, type = "b") + nchar(sm[1L],
.                       type = "b")
.                   if (w > LONG)
.                     prefix <- paste0(prefix, "\n  ")
.                 }
.                 else prefix <- "Error : "
.                 msg <- paste0(prefix, conditionMessage(e), "\n")
.                 .Internal(seterrmessage(msg[1L]))
.                 if (!silent && isTRUE(getOption("show.error.messages"))) {
.                   cat(msg, file = outFile)
.                   .Internal(printDeferredWarnings())
.                 }
.                 invisible(structure(msg, class = "try-error",
.                   condition = e))
.             }), tryCatchList(expr, classes, parentenv, handlers),
.         tryCatchOne(expr, names, parentenv, handlers[[1L]]),
.         doTryCatch(return(expr), name, parentenv, handler), withCallingHandlers(withVisible(value_fun(ev$value,
.             ev$visible)), warning = wHandler, error = eHandler,
.             message = mHandler), withVisible(value_fun(ev$value,
.             ev$visible)), value_fun(ev$value, ev$visible), prepare_mimebundle_kernel(obj,
.             .self$handle_display_error), prepare_mimebundle(obj,
.             "text/plain", error_handler = handle_display_error),
.         filter_map(mimetypes, function(mime) {
.             tryCatch(withCallingHandlers({
.                 if (!mime %in% names(repr::mime2repr))
.                   stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
.                 rpr <- repr::mime2repr[[mime]](obj)
.                 if (is.null(rpr))
.                   return(NULL)
.                 prepare_content(is.raw(rpr), rpr)
.             }, error = error_handler), error = outer_handler)
.         }), Filter(Negate(is.null), sapply(x, f, simplify = simplify)),
.         unlist(lapply(x, f)), lapply(x, f), sapply(x, f, simplify = simplify),
.         lapply(X = X, FUN = FUN, ...), FUN(X[[i]], ...), tryCatch(withCallingHandlers({
.             if (!mime %in% names(repr::mime2repr))
.                 stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
.             rpr <- repr::mime2repr[[mime]](obj)
.             if (is.null(rpr))
.                 return(NULL)
.             prepare_content(is.raw(rpr), rpr)
.         }, error = error_handler), error = outer_handler), tryCatchList(expr,
.             classes, parentenv, handlers), tryCatchOne(expr,
.             names, parentenv, handlers[[1L]]), doTryCatch(return(expr),
.             name, parentenv, handler), withCallingHandlers({
.             if (!mime %in% names(repr::mime2repr))
```

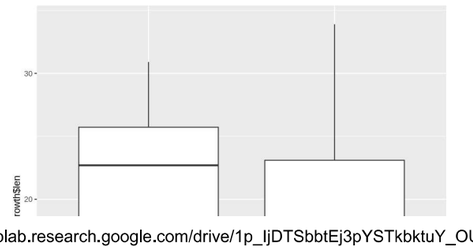
```
.                stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
.            rpr <- repr::mime2repr[[mime]](obj)
.            if (is.null(rpr))
.                return(NULL)
.            prepare_content(is.raw(rpr), rpr)
.        }, error = error_handler), repr::mime2repr[[mime]](obj),
.        repr_text.default(obj), paste(capture.output(print(obj)),
.            collapse = "\n"), capture.output(print(obj)), withVisible(...elt(i)),
.        print(obj), print.ggplot(obj), ggplot_build(x), ggplot_build.ggplot(x),
.        by_layer(function(l, d) l$compute_statistic(d, layout),
.            layers, data, "computing stat"), try_fetch(for (i in seq_along(data)) {
.            out[[i]] <- f(l = layers[[i]], d = data[[i]])
.        }, error = function(cnd) {
.            cli::cli_abort(c("Problem while {step}.", i = "Error occurred in the {ordinal(i)} layer."),
.                call = layers[[i]]$constructor, parent = cnd)
.        }), tryCatch(withCallingHandlers(expr, condition = function(cnd) {
.            {
.                .__handler_frame__. <- TRUE
.                .__setup_frame__. <- frame
.                if (inherits(cnd, "message")) {
.                  except <- c("warning", "error")
.                }
.                else if (inherits(cnd, "warning")) {
.                  except <- "error"
.                }
.                else {
.                  except <- ""
.                }
.            }
.            while (!is_null(cnd)) {
.                if (inherits(cnd, "error")) {
.                  out <- handlers[[1L]](cnd)
.                  if (!inherits(out, "rlang_zap"))
.                    throw(out)
.                }
.                inherit <- .subset2(.subset2(cnd, "rlang"), "inherit")
.                if (is_false(inherit)) {
.                  return()
.                }
.                cnd <- .subset2(cnd, "parent")
.            }
.        }), stackOverflowError = handlers[[1L]]), tryCatchList(expr,
.            classes, parentenv, handlers), tryCatchOne(expr,
.            names, parentenv, handlers[[1L]]), doTryCatch(return(expr),
.            name, parentenv, handler), withCallingHandlers(expr,
.            condition = function(cnd) {
.                {
.                  .__handler_frame__. <- TRUE
.                  .__setup_frame__. <- frame
.                  if (inherits(cnd, "message")) {
.                    except <- c("warning", "error")
.                  }
.                  else if (inherits(cnd, "warning")) {
.                    except <- "error"
.                  }
.                  else {
.                    except <- ""
.                  }
.                }
.                while (!is_null(cnd)) {
.                  if (inherits(cnd, "error")) {
.                    out <- handlers[[1L]](cnd)
.                    if (!inherits(out, "rlang_zap"))
.                      throw(out)
.                  }
.                  inherit <- .subset2(.subset2(cnd, "rlang"),
.                    "inherit")
.                  if (is_false(inherit)) {
.                    return()
.                  }
.                  cnd <- .subset2(cnd, "parent")
.                }
.            }), f(l = layers[[i]], d = data[[i]]), l$compute_statistic(d,
.            layout), compute_statistic(..., self = self), self$stat$setup_params(data,
.            self$stat_params), setup_params(..., self = self),
.        cli::cli_abort("{.fn {snake_class(self)}} requires an {.field x} or {.field y} aesthetic."),
.        rlang::abort(message, ..., call = call, use_cli_format = TRUE,
.            .frame = .frame)), parent = c(0L, 1L, 2L, 3L, 4L,
.    5L, 6L, 7L, 6L, 9L, 10L, 4L, 12L, 13L, 14L, 15L, 16L, 17L,
.    18L, 13L, 13L, 13L, 22L, 23L, 24L, 25L, 26L, 26L, 25L, 29L,
.    30L, 31L, 32L, 33L, 34L, 31L, 31L, 31L, 38L, 38L, 40L, 38L,
.    38L, 43L, 43L, 45L, 46L, 47L, 48L, 49L, 50L, 47L, 46L, 53L,
.    54L, 55L, 56L, 57L, 58L), visible = c(TRUE, TRUE, TRUE, TRUE,
.    TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
.    TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE,
```

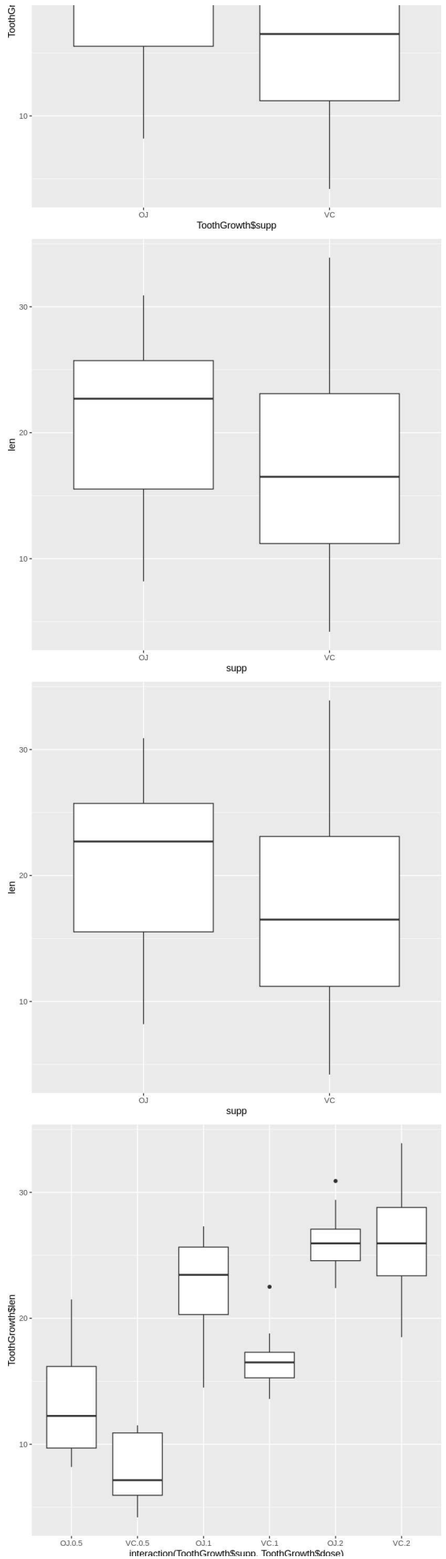
```
plot(ToothGrowth$supp, ToothGrowth$len) # using plot() function and pass it a factor of x-values and a vector of y-values.
#Formula Syntax
boxplot(len ~ supp, data = ToothGrowth) # if the tw0 vectors are in the same dataframe, you can use the formula syntax. With # this syntax you can combine t
# put interaction of two variables on x-axis,with formula syntax and two variables on the x-axis
boxplot(len ~ supp + dose, data = ToothGrowth)
# with ggplot2 you can get the same results above.
library (ggplot2)
qplot(ToothGrowth$supp, ToothGrowth$len, geom = "boxplot")
# if the two vectors are in the same dataframe, you can use the following syntax
qplot(supp, len, data = ToothGrowth, geom = "boxplot")
# in ggplot2, the above is equvalent to:
ggplot (ToothGrowth, aes( x= supp, y= len)) + geom_boxplot ()
# Using three seperate vectors
qplot (interaction(ToothGrowth$supp, ToothGrowth$dose), ToothGrowth$len, geom = "boxplot")
# You can write the samething above, get the columns from the dataframe
qplot (interaction(supp, dose), len, data = ToothGrowth, geom = "boxplot")
# Using ggplot() you can do the samething and it is equivalent to:
ggplot (ToothGrowth, aes ( x= interaction(supp, dose), y= len) ) + geom_boxplot ()
#Plotting a function curve

```


y
x
OJ
VC
5
10
15
20
25
30
35


len
supp
OJ
VC
5
10
15
20
25
30
35


len
supp : dose
OJ.0.5
VC.0.5
OJ.1
VC.1
OJ.2
VC.2
5
10
15
20
25
30
35


rowth$len
20
30


ToothGr
10
OJ
VC
ToothGrowth$supp

30
20
10
len
OJ
VC
supp

30
20
10
len
OJ
VC
supp

30
20
10
ToothGrowth$len
OJ.0.5
VC.0.5
OJ.1
VC.1
OJ.2
VC.2
interaction(ToothGrowth$supp, ToothGrowth$dose)

1