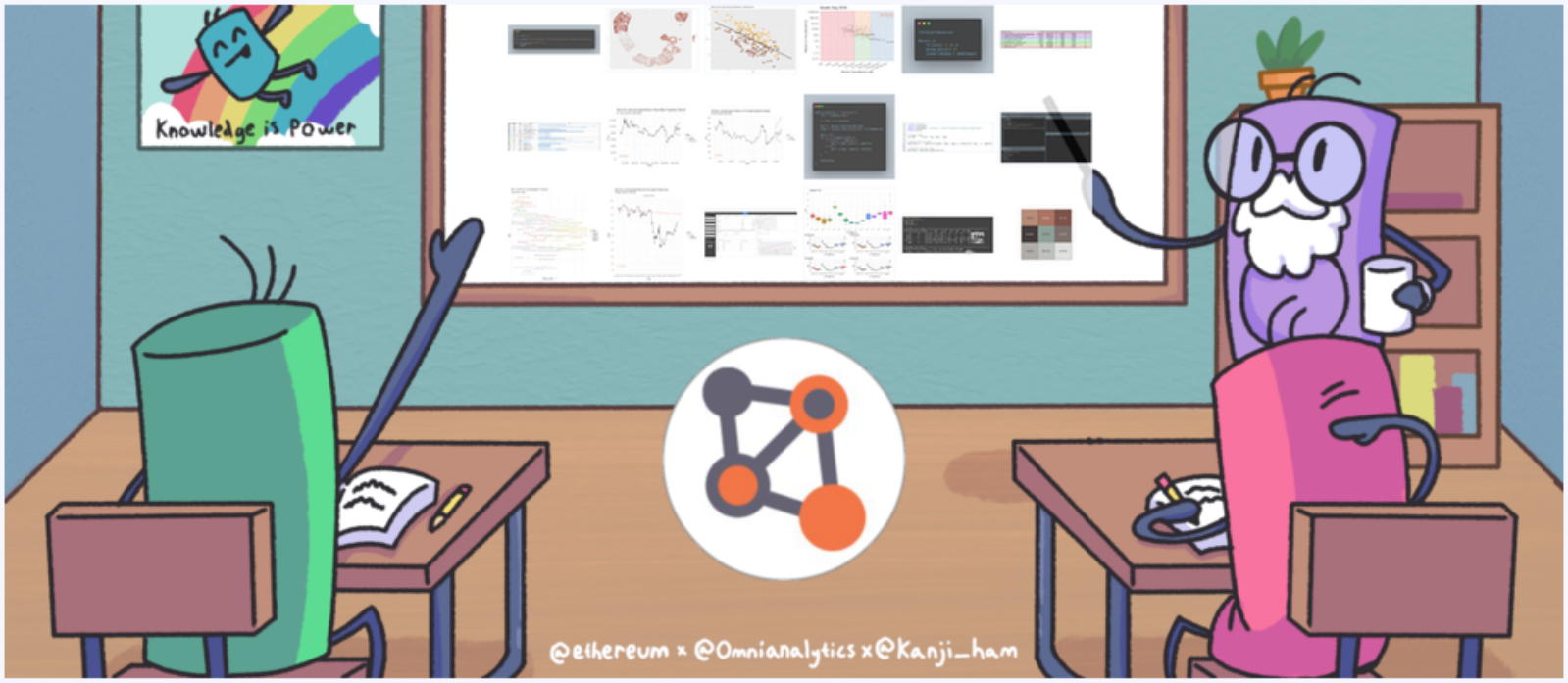
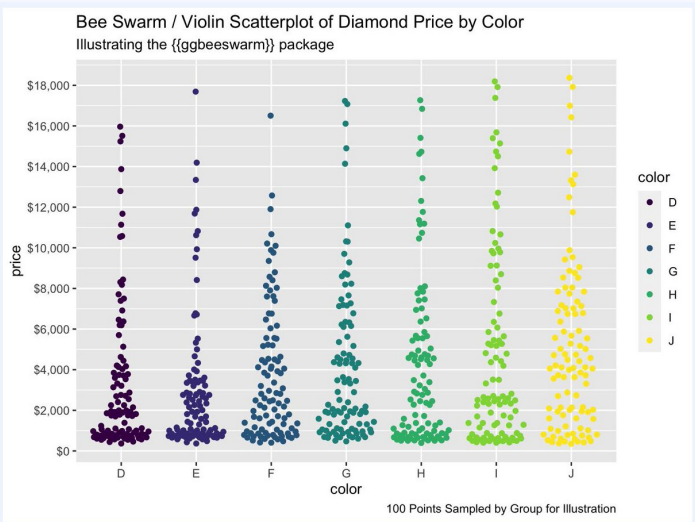
The Omniacs' Data Science Code Snippet Tweet Book Vol 1

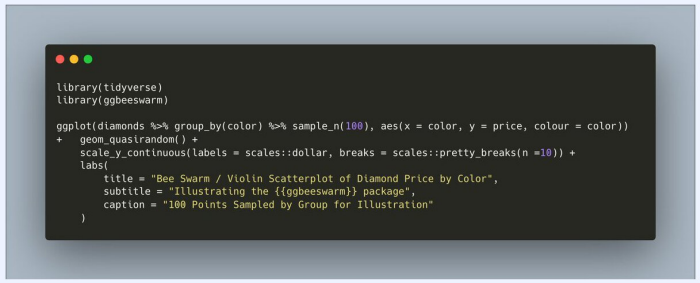


Quirkily Created by Omni Analytics Group

#rstats and #tidyverse

A #beeswarm plot, otherwise known as a Violin Scatter Plot, lets us view both distributional characteristics of the data, as well as details of the individual points. {{ ggbeeswarm }} makes producing such plots very simple! #rstats #dataviz #tidyverse



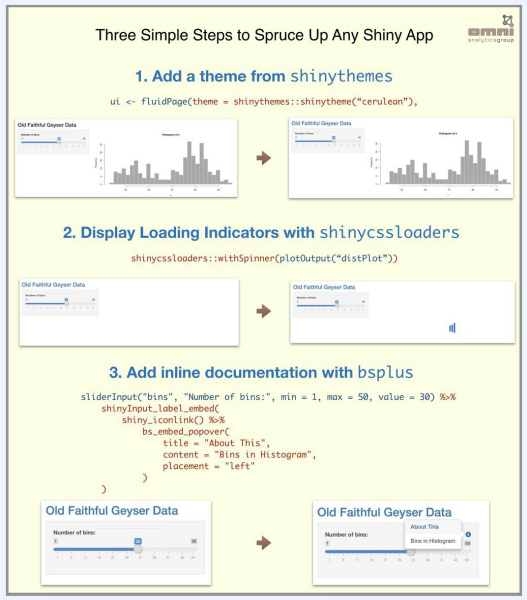


```
library(tidyverse)
library(ggbeeswarm)

ggplot(diamonds %>% group_by(color) %>% sample_n(100), aes(x = color, y = price, colour = color))
+   geom_quasirandom() +
    scale_y_continuous(labels = scales::dollar, breaks = scales::pretty_breaks(n =10)) +
    labs(
        title = "Bee Swarm / Violin Scatterplot of Diamond Price by Color",
        subtitle = "Illustrating the {{ggbeeswarm}} package",
        caption = "100 Points Sampled by Group for Illustration"
    )
```

Tweeted on: 2022-05-17 20:06:46, Retweets: 6, Likes: 22

#rstats and #tidytuesday

Spruce up an #RShiny app in three easy steps: (1) Add a theme from #shinythemes, (2) Display loading indicators with #shinycssloaders, and (3) Add inline documentation with #bsplus! #rstats #programming #tidytuesday



Tweeted on: 2019-05-21 11:12:44, Retweets: 5, Likes: 17

#rstats and #tidyverse

If you want a human-readable description of the #dplyr data pipeline steps taken in your code, simply load the #tidylog package before running it! #tidyverse #rstats #TidyTuesday

dplyr code

```
library(tidyverse)
library(tidylog)

diamonds %>%
    mutate(ppc = price / carat) %>%
    select(cut, color, clarity, ppc) %>%
    group_by(cut) %>%
    summarise(ppc = mean(ppc))
```

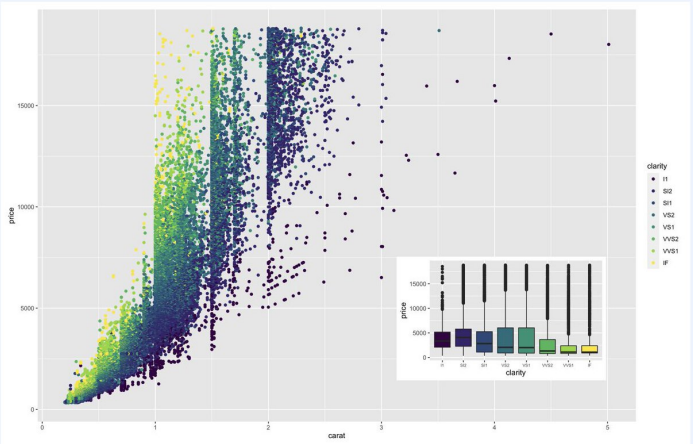
tidylog output

mutate: new variable 'ppc' with 26,693 unique values and 0% NA
select: dropped 7 variables (carat, depth, table, price, x, …)
group_by: one grouping variable (cut)
summarise: now 5 rows and 2 columns, ungrouped

Tweeted on: 2020-01-14 13:52:21, Retweets: 8, Likes: 17

#rstats and #tidyverse

The newest {{ patchwork }} release on #cran goes beyond alignment of plots to a grid, and enables support for inset plots, allowing plots to be overlaid on top of one another at the exact location specified! #rstats #dataviz #tidyverse

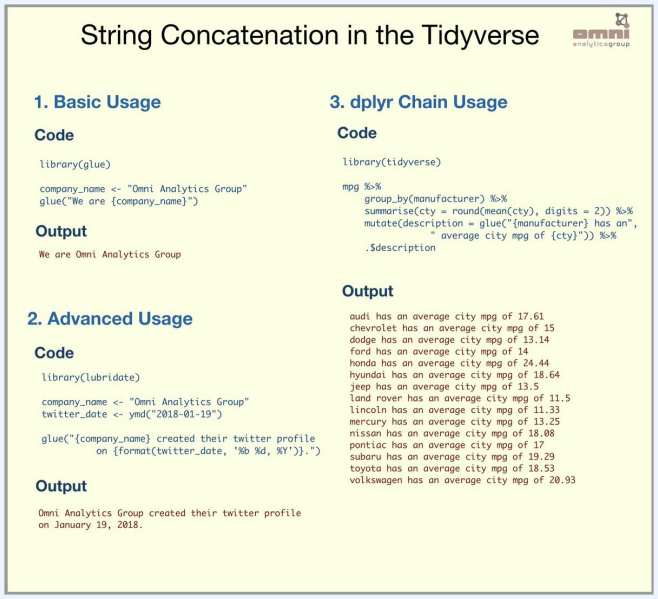



```
library(ggplot2)
library(patchwork)

ggplot(data = diamonds, aes(x = carat, y = price, color = clarity)) +
    geom_point() +
    inset_element(
        ggplot(data = diamonds, aes(x = clarity, y = price, fill = clarity)) +
            geom_boxplot() +
            theme(legend.position = "off",
                  axis.text.x = element_text(size = 6)),
        left = 0.6, bottom = 0.1, right = 0.95, top = 0.4
    )
```

Tweeted on: 2020-11-13 17:48:25, Retweets: 6, Likes: 15

#rstats and #tidyverse

For #tidytuesday, were highlighting #glue, a #tidyverse-compatible alternative to the paste() function for string concatenation in R. Have a look at some standalone examples, and how it can be used in a #dplyr chain. #rstats #datascience #programming



String Concatenation in the Tidyverse

1. Basic Usage

Code

```
library(glue)

company_name <- "Omni Analytics Group"
glue("We are {company_name}")
```

Output

```
We are Omni Analytics Group
```

2. Advanced Usage

Code

```
library(lubridate)

company_name <- "Omni Analytics Group"
twitter_date <- ymd("2018-01-19")

glue("{company_name} created their twitter profile
          on {format(twitter_date, '%b %d, %Y')}.")
```

Output

```
Omni Analytics Group created their twitter profile
on January 19, 2018.
```

3. dplyr Chain Usage

Code

```
library(tidyverse)

mpg %>%
    group_by(manufacturer) %>%
    summarise(cty = round(mean(cty), digits = 2)) %>%
    mutate(description = glue("{manufacturer} has an",
                  " average city mpg of {cty}")) %>%
    .$description
```

Output

```
audi has an average city mpg of 17.61
chevrolet has an average city mpg of 15
dodge has an average city mpg of 13.14
ford has an average city mpg of 14
honda has an average city mpg of 24.44
hyundai has an average city mpg of 18.64
jeep has an average city mpg of 13.5
land rover has an average city mpg of 11.5
lincoln has an average city mpg of 11.33
mercury has an average city mpg of 13.25
nissan has an average city mpg of 18.08
pontiac has an average city mpg of 17
subaru has an average city mpg of 19.29
toyota has an average city mpg of 18.53
volkswagen has an average city mpg of 20.93
```

Tweeted on: 2019-04-09 15:04:51, Retweets: 7, Likes: 13

#rstats and #tidyverse

Using paradigms from the {glue} package but thinking in reverse, the {unglue} #rstats package provides for some help #regex functionality, including the creation of a #dataframe automatically from semi-structured sentences that match a defined pattern. #tidyverse #tidytuesday

```
# A set of semi-structured sentences
sentences <- c("The quick brown fox jumped over the lazy dog.",
               "The slow red hamster climbed over the frenetic lizard.",
               "The handsome mauve turtle swam over the unsuspecting anchovy.",
               "The confused rainbow pigeon flew over the happy hippo.",
               "The slick colorless rstats-package imported over the neglected base.")
```

```
> unglue_data(sentences, patterns)
  adjective1     color        animal1     verb   adjective2 animal2
1      quick     brown            fox   jumped         lazy     dog
2       slow       red        hamster  climbed     frenetic  lizard
3   handsome     mauve         turtle     swam unsuspecting anchovy
4   confused   rainbow         pigeon     flew        happy   hippo
5      slick colorless rstats-package imported    neglected    base
```

```
library(unglue)

# A set of semi-structured sentences
sentences <- c("The quick brown fox jumped over the lazy dog.",
               "The slow red hamster climbed over the frenetic lizard.",
               "The handsome mauve turtle swam over the unsuspecting anchovy.",
               "The confused rainbow pigeon flew over the happy hippo.",
               "The slick colorless rstats-package imported over the neglected base.")

# Define a pattern for the data
patterns <- c("The {adjective1} {color} {animal1} {verb} over the {adjective2} {animal2}.")

# Unglue it into a data frame
unglue_data(sentences, patterns)
```

Tweeted on: 2020-06-23 16:49:42, Retweets: 6, Likes: 13

#rstats and #tidyverse

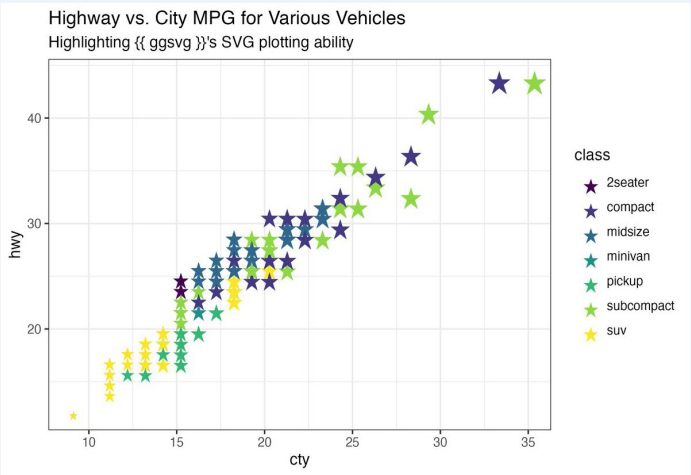
Whether due to disk space constraints, or just ease and convenience, sometimes its nice to work with archives like #zip files directly in #rstats The {{ archive }} package makes this painless, and even supports writing archives natively! #tidyverse

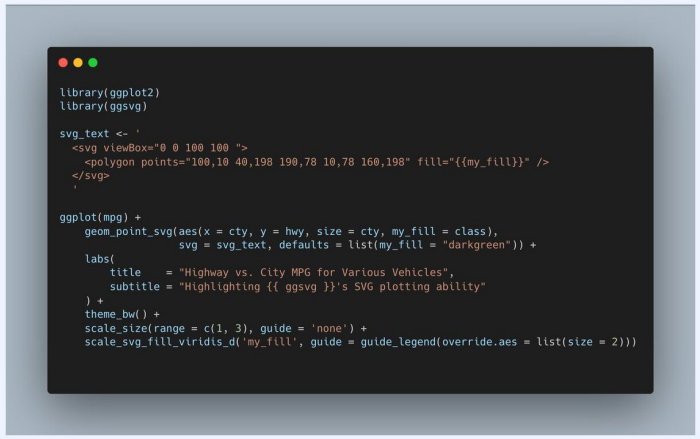
```
library(archive)
library(ggplot2)
library(readr)

data_export <- archive_write("diamonds.zip", "diamonds.csv")
write_csv(diamonds, data_export)
```

Tweeted on: 2021-11-09 15:07:25, Retweets: 8, Likes: 12

#rstats and #tidyverse

{{ ggsvg }}, another extremely useful #ggplot2 extension, allows the plotting of arbitrary SVGs as points. Here we define an SVG by plotting a polygon in the shape of a star, and color/size it according to properties of the underlying data! #rstats #tidyverse



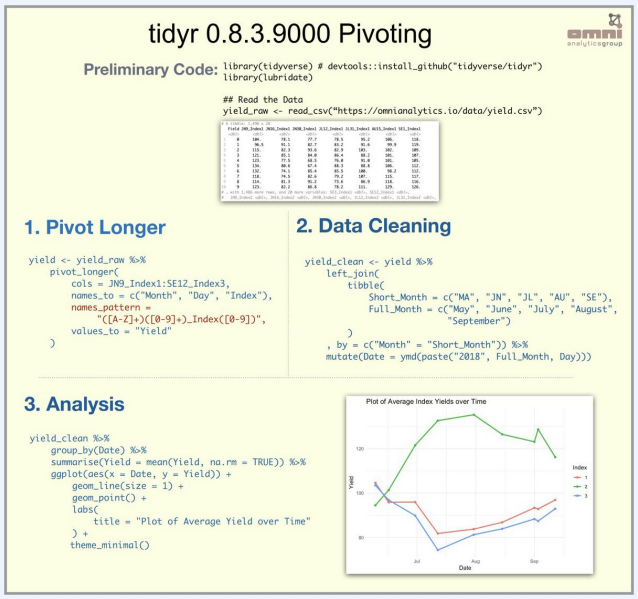


```
library(ggplot2)
library(ggsvg)

svg_text <- '
  <svg viewBox="0 0 100 100 ">
    <polygon points="100,10 40,198 190,78 10,78 160,198" fill="{{my_fill}}" />
  </svg>
  '

ggplot(mpg) +
    geom_point_svg(aes(x = cty, y = hwy, size = cty, my_fill = class),
                   svg = svg_text, defaults = list(my_fill = "darkgreen")) +
    labs(
        title    = "Highway vs. City MPG for Various Vehicles",
        subtitle = "Highlighting {{ ggsvg }}'s SVG plotting ability"
    ) +
    theme_bw() +
    scale_size(range = c(1, 3), guide = 'none') +
    scale_svg_fill_viridis_d('my_fill', guide = guide_legend(override.aes = list(size = 2)))
```

Tweeted on: 2021-12-30 18:59:52, Retweets: 5, Likes: 12

#rstats and #tidyverse

Brand new functionality in the dev version of #tidyr allows for columns containing multiple variables to be easily separated with the names_pattern argument from the pivot_longer() function! #rstats #tidyverse #tidytuesday #datascience

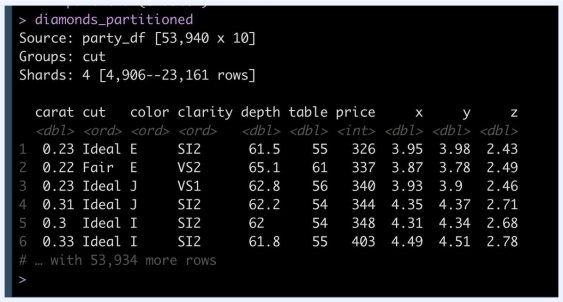


Tweeted on: 2019-04-02 18:05:39, Retweets: 9, Likes: 11

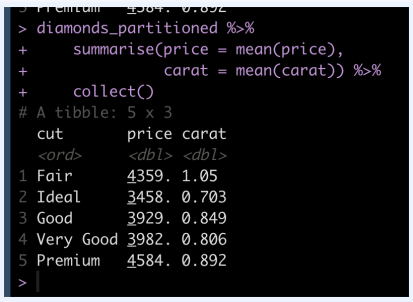
#rstats and #tidyverse

Parallel computing in the #tidyverse - {{ multidplyr }} 0.1 is now on CRAN and provides a quick and easy framework for distributing long-running computations to the different processes in a cluster, then collecting the results! #rstats

```
library(multidplyr)
library(ggplot2)

cluster <- new_cluster(4)
cluster_library(cluster, "dplyr")

diamonds_partitioned <- diamonds %>%
    group_by(cut) %>%
    partition(cluster)

diamonds_partitioned %>%
    summarise(price = mean(price),
              carat = mean(carat)) %>%
    collect()
```



```
> diamonds_partitioned
Source: party_df [53,940 x 10]
Groups: cut
Shards: 4 [4,906--23,161 rows]

  carat cut   color clarity depth table price     x     y     z
  <dbl> <ord> <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1  0.23 Ideal E     SI2      61.5    55   326  3.95  3.98  2.43
2  0.22 Fair  E     VS2      65.1    61   337  3.87  3.78  2.49
3  0.23 Ideal J     VS1      62.8    56   340  3.93  3.9   2.46
4  0.31 Ideal J     SI2      62.2    54   344  4.35  4.37  2.71
5  0.3  Ideal I     SI2      62      54   348  4.31  4.34  2.68
6  0.33 Ideal I     SI2      61.8    55   403  4.49  4.51  2.78
# … with 53,934 more rows
>
```



```
> diamonds_partitioned %>%
+     summarise(price = mean(price),
+               carat = mean(carat)) %>%
+     collect()
# A tibble: 5 x 3
  cut       price carat
  <ord>     <dbl> <dbl>
1 Fair      4359. 1.05
2 Ideal     3458. 0.703
3 Good      3929. 0.849
4 Very Good 3982. 0.806
5 Premium   4584. 0.892
>
```

Tweeted on: 2021-02-25 16:08:45, Retweets: 5, Likes: 11

#rstats and #programming

🔢 Simple steps to convert your #forloop to a parallel {{ foreach }} loop:

1 Change “for” to “foreach”

2 Change “in” to “=”

3 Assign the result to a variable

4 Add “%do%” after the iterator statement

#rstats #programming

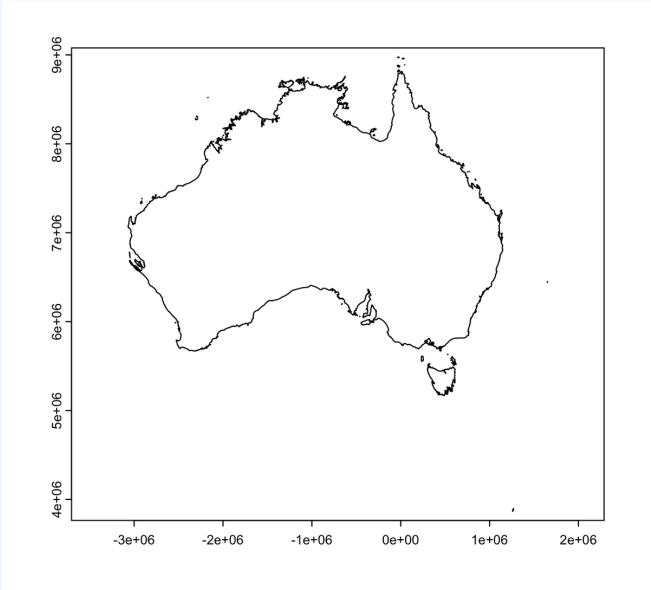
```
for (i in 1:10) {
    j = i^2 - 5

    print(j)
}
```

```
library(foreach)

res <- foreach (i = 1:10) %do% {
    j = i^2 - 5

    print(j)
}
```

Tweeted on: 2022-03-31 15:41:24, Retweets: 8, Likes: 10

#rstats and #tidyverse

For working with #rasters, we love the {{ terra }} package in #rstats - Combine that with {{ geodata }} and in just a few lines of code, we can manipulate and project geographic data from around the world! #tidyverse

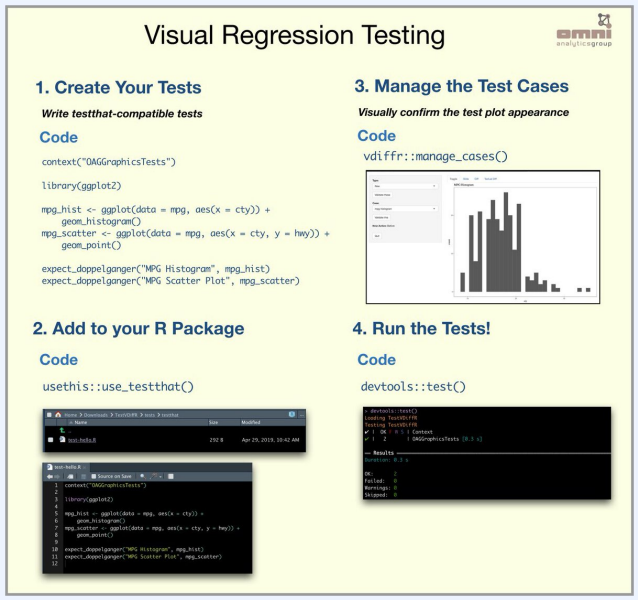



```
library(terra)
library(geodata)

w <- world(path = ".", resolution =
australia <- w[w$GID_0=="AUS", ]

prj <- "epsg:28355"

p_aus <- project(australia, prj)
plot(p_aus)
```

Tweeted on: 2022-05-25 14:24:24, Retweets: 6, Likes: 10

#rstats and #tidytuesday

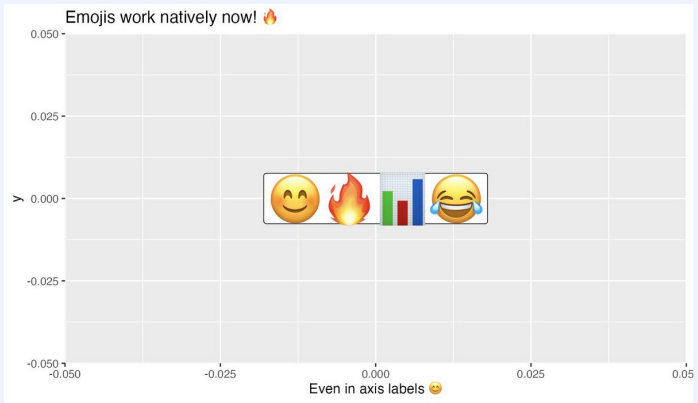
Adding #testthat tests to your #rstats package helps with proper regression testing. But @_lionelhenrys #vdiffr package extends this functionality even further, allowing you to test for regressions in visuals such as #ggplot2 plots! #rstats #tidytuesday



Tweeted on: 2019-04-30 14:41:33, Retweets: 5, Likes: 9

#rstats and #tidyverse

Configure @RStudio 1.4 to use the AGG backend, install the latest {{ ragg }} package, and like magic, emojis are natively rendered in #ggplot in the RStudio plot pane without any configuration needed! #rstats #tidyverse Read more here: https://www.tidyverse.org/blog/2021/02/modern-text-features/

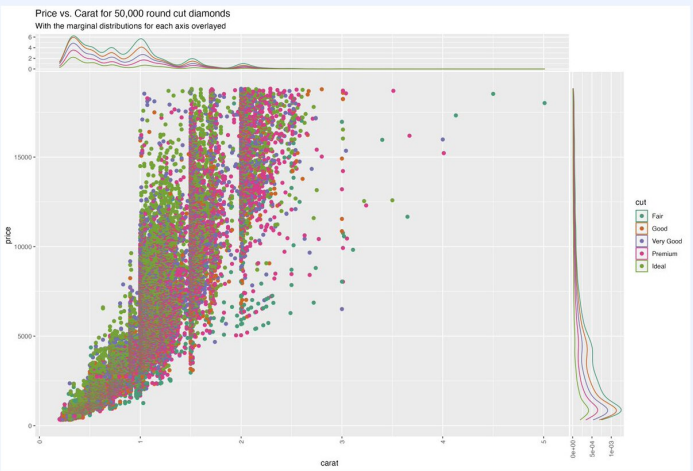



```
library(ragg)
library(ggplot2)

emojis <- "😊🔥📊😂"

p <- ggplot() +
    geom_label(
        aes(x = 0, y = 0, label = emojis),
        family = "Apple Color Emoji",
        size = 15
    ) +
    labs(
        title = "Emojis work natively now! 🔥",
        x = "Even in axis labels 😊"
    )
```

Tweeted on: 2021-02-17 20:19:17, Retweets: 6, Likes: 9

#rstats and #dataviz

Another very useful {{ ggplot2 }} extension is {{ ggside }}, which allows for a quick and easy #visualization of the marginal distributions of your variables side by side with your plot. #rstats #dataviz #TidyTuesday



```
library(tidyverse)
library(ggside)

ggplot(diamonds, aes(carat, price, colour = cut)) +
    geom_point(size = 2) +
    scale_colour_brewer(palette = "Dark2") +
    geom_xsidedensity(aes(y = after_stat(density)), position = "stack") +
    geom_ysidedensity(aes(x = after_stat(density)), position = "stack") +
    theme(axis.text.x = element_text(angle = 90, vjust = .5)) +
    labs(
        title = "Price vs. Carat for 50,000 round cut diamonds",
        subtitle = "With the marginal distributions for each axis overlayed"
    )
```

Tweeted on: 2021-05-25 14:31:19, Retweets: 4, Likes: 9

#rstats and #tidyverse

The #python package {{ polars }} is an extremely efficient library for #dataframe manipulation. Better yet, it has a syntax for data summaries that should be quite familiar to most #rstats and #tidyverse users!

```
In [6]: df
Out[6]:
shape: (5, 8)
```

cut	carat_sum	depth_sum	table_sum	price_sum	x_sum	y_sum	z_sum
str	f64	f64	f64	i64	f64	f64	f64
Good	695.2099999999986	5.80361000000001e4	5.48416999999999e4	3194260	5241.49	5255.079999999999	3261.829999999977
Very Good	1623.1599999999958	1.48152599999999 2e5	1.3929329999999993e5	7715165	1.3037919999999998e4	1.3111710000000001e4	8097.589999999965
Ideal	2257.4999999999513	2.40763099999974e5	2.18441000000000003e5	10138238	2.03571200000000006e4	2.04213300000000016e4	1.25690099999999978e4
Fair	191.8799999999999	1.41836000000000 6e4	1.32977e4	824838	1323.6300000000006	1312.2399999999998	833.7600000000001
Premium	1677.3699999999856	1.42968099999983e5	1.373676e5	8270443	1.30570900000000017e4	1.298987e4	7966.539999999967

```
In [2]: df
Out[2]:
shape: (53940, 10)
```

carat	cut	color	clarity	...	price	x	y	z
f64	str	str	str		i64	f64	f64	f64
0.23	Ideal	E	SI2	...	326	3.95	3.98	2.43
0.21	Premium	E	SI1	...	326	3.89	3.84	2.31
0.23	Good	E	VS1	...	327	4.05	4.07	2.31
0.29	Premium	I	VS2	...	334	4.2	4.23	2.63
...	...	...	...	...	...	...	...	...
0.72	Ideal	D	SI1	...	2757	5.75	5.76	3.5
0.72	Good	D	SI1	...	2757	5.69	5.75	3.61
0.7	Very Good	D	SI1	...	2757	5.66	5.68	3.56
0.86	Premium	H	SI2	...	2757	6.15	6.12	3.74
0.75	Ideal	D	SI2	...	2757	5.83	5.87	3.64

```
import polars as pl

df = pl.read_csv("https://raw.githubusercontent.com/tidyverse/ggplot2/main/data-raw/diamonds.csv")

df = (df.filter(pl.col("color") == "E")
      .groupby("cut")
      .sum())
```

Tweeted on: 2021-11-23 20:13:03, Retweets: 3, Likes: 9

#rstats and #tidyverse

An essential package in our #rstats analysis toolkit, {{ tidyr }} gets a new 1.2.0 release today. Were still unpacking the new features, but one cool one - the ability to expand the names in pivot operations, to retain missing factor levels automatically! #tidyverse

```
> months <- c("January", "February", "March", "April",
+             "May", "June", "July", "August",
+             "September", "October", "November", "December")
>
> m_series <- tibble(
+     month = factor(months[-c(2, 10)], levels = months),
+     value = sample(1:10)
+ )
>
> pivot_wider(m_series, names_from = month,
+             values_from = value, names_expand = TRUE)
# A tibble: 1 × 12
  January February March April   May  June  July August September October November December
    <int>    <int> <int> <int> <int> <int> <int>  <int>     <int>   <int>    <int>    <int>
1       6       NA     3    10     8     5     9      7         2      NA        1        4
> |
```

```
library(tidyr)

months <- c("January", "February", "March", "April",
            "May", "June", "July", "August",
            "September", "October", "November", "December")

m_series <- tibble(
    month = factor(months[-c(2, 10)], levels = months),
    value = sample(1:10)
)

pivot_wider(m_series, names_from = month,
            values_from = value, names_expand = TRUE)
])
}
```

Tweeted on: 2022-02-03 01:21:42, Retweets: 7, Likes: 9

#rstats and #tidyverse

The latest release of #santoku 🔪 for #rstats adds even more functionality for chopping vectors, including support for dates! Here we easily chop the airquality data into Early, Middle, and Late categories based on the date of observation. #tidyverse #tidytuesday

```r
library(santoku)
library(lubridate)
library(tidyverse)

airquality %>%
    mutate(Date = ymd(paste0("1973-", Month, "-", Day))) %>%
    mutate(Cut = chop_quantiles(Date, c(0, .33, .66, 1), labels = c("Early", "Middle", "Late"))) %>%
    group_by(Cut) %>%
    slice(1:5)
```

```
# A tibble: 15 x 8
# Groups:   Cut [3]
   Ozone Solar.R  Wind  Temp Month   Day Date       Cut
   <int>   <int> <dbl> <int> <int> <int> <date>     <fct>
 1    41     190   7.4    67     5     1 1973-05-01 Early
 2    36     118   8      72     5     2 1973-05-02 Early
 3    12     149  12.6    74     5     3 1973-05-03 Early
 4    18     313  11.5    62     5     4 1973-05-04 Early
 5    NA      NA  14.3    56     5     5 1973-05-05 Early
 6    NA     150   6.3    77     6    21 1973-06-21 Middle
 7    NA      59   1.7    76     6    22 1973-06-22 Middle
 8    NA      91   4.6    76     6    23 1973-06-23 Middle
 9    NA     250   6.3    76     6    24 1973-06-24 Middle
10    NA     135   8      75     6    25 1973-06-25 Middle
11    NA     222   8.6    92     8    10 1973-08-10 Late
12    NA     137  11.5    86     8    11 1973-08-11 Late
13    44     192  11.5    86     8    12 1973-08-12 Late
14    28     273  11.5    82     8    13 1973-08-13 Late
15    65     157   9.7    80     8    14 1973-08-14 Late
```

Tweeted on: 2020-06-09 22:42:56, Retweets: 10, Likes: 8

#rstats and #tidyverse

The latest {{ dbplyr }} 2.0 release adds compatibilty with {{ dplyr }}s 1.0 release features, and drastically improves the #sql translation. Heres some #rstats #tidyverse code and the corresponding #sql automatically generated!

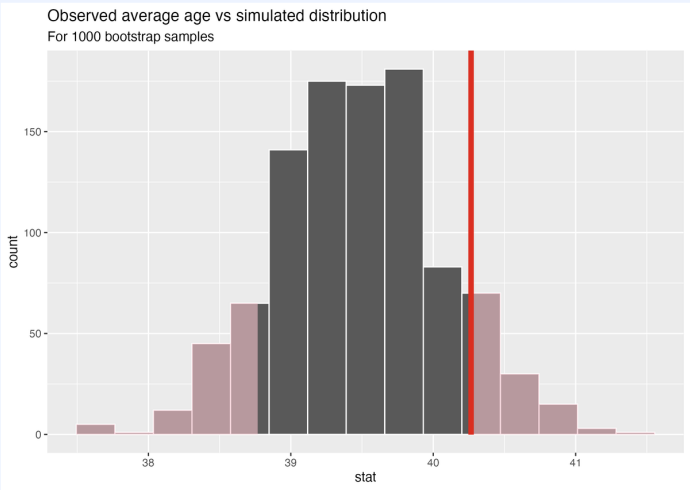
```r
library(dbplyr)
library(tidyverse)

lf <- tbl_lazy(mpg)

lf %>%
    mutate(model = toupper(model)) %>%
    group_by(model) %>%
    summarise(across(c(cty, hwy), mean, na.rm = TRUE)) %>%
    arrange(desc(cty))
```

```sql
SELECT `model`, AVG(`cty`) AS `cty`, AVG(`hwy`) AS `hwy`
FROM (SELECT `manufacturer`, UPPER(`model`) AS `model`, `displ`, `year`, `cyl`, `trans`, `drv`, `cty`,
`hwy`, `fl`, `class`
FROM `df`) `q01`
GROUP BY `model`
ORDER BY `cty` DESC
```

Tweeted on: 2020-11-05 19:41:09, Retweets: 4, Likes: 8

#rstats and #tidyverse

Powerful and easy-to-use new #rstats package for #tidyverse-compliant statistical inference! The {{ infer }} package makes it dead simple to perform a number of statistical inference tasks, including simulation from a null-distribution https://www.tidyverse.org/blog/2021/08/infer-1-0-0/



```
library(infer)
library(ggplot2)

point_estimate <- gss %>%
    specify(response = age) %>%
    calculate(stat = "mean")

gss %>%
    specify(response = age) %>%
    hypothesize(null = "point", mu = 39.5) %>%
    generate(reps = 1000, type = "bootstrap") %>%
    calculate(stat = "mean") %>%
    visualize() +
    shade_p_value(obs_stat = point_estimate, direction = "two-sided") +
    labs(
        title = "Observed average age vs simulated distribution",
        subtitle = "For 1000 bootstrap samples"
    )
```

Tweeted on: 2021-08-17 22:16:16, Retweets: 5, Likes: 8

#rstats and #python

Its now even easier to import #rstats #dataframes into #python. With {{ pyreadr }}, RData files can be natively imported and converted into a #pandas dataframe like so!

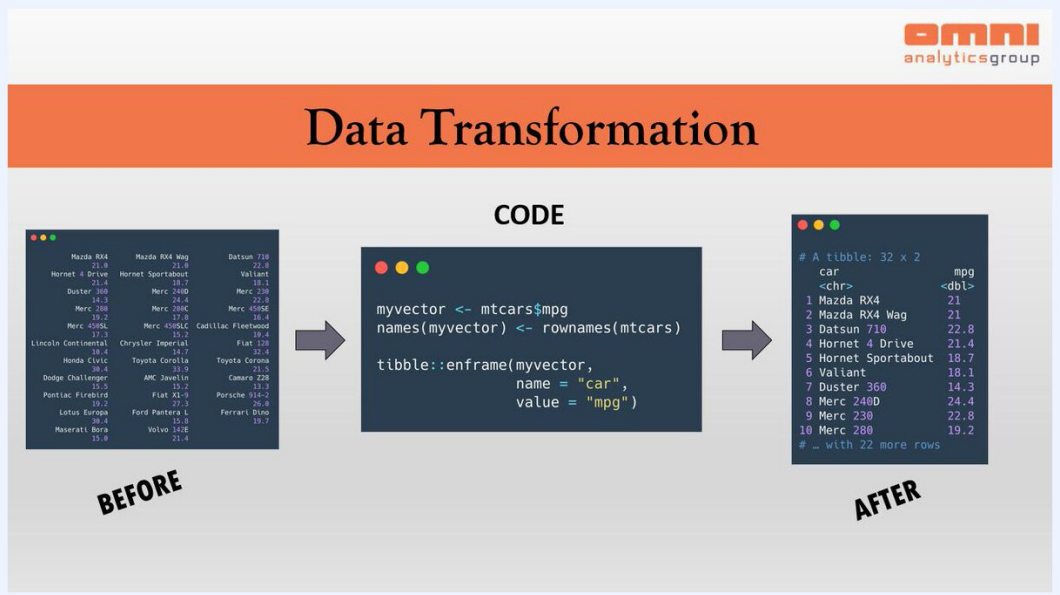
```
[In [6]: mpg_df
Out[6]:
    manufacturer   model  displ  year  cyl       trans drv  cty  hwy fl    class
0           audi      a4    1.8  1999    4    auto(l5)   f   18   29  p  compact
1           audi      a4    1.8  1999    4  manual(m5)   f   21   29  p  compact
2           audi      a4    2.0  2008    4  manual(m6)   f   20   31  p  compact
3           audi      a4    2.0  2008    4    auto(av)   f   21   30  p  compact
4           audi      a4    2.8  1999    6    auto(l5)   f   16   26  p  compact
..           ...     ...    ...   ...  ...         ...  ..  ...  ... ..      ...
229   volkswagen  passat    2.0  2008    4    auto(s6)   f   19   28  p  midsize
230   volkswagen  passat    2.0  2008    4  manual(m6)   f   21   29  p  midsize
231   volkswagen  passat    2.8  1999    6    auto(l5)   f   16   26  p  midsize
232   volkswagen  passat    2.8  1999    6  manual(m5)   f   18   26  p  midsize
233   volkswagen  passat    3.6  2008    6    auto(s6)   f   17   26  p  midsize

[234 rows x 11 columns]
```

```
import pyreadr

mpg = pyreadr.read_r('Downloads/Snapshot/mpg.RData')
mpg_df = mpg["mpg"]
```

Tweeted on: 2021-12-08 16:35:54, Retweets: 5, Likes: 8

#rstats and #programming

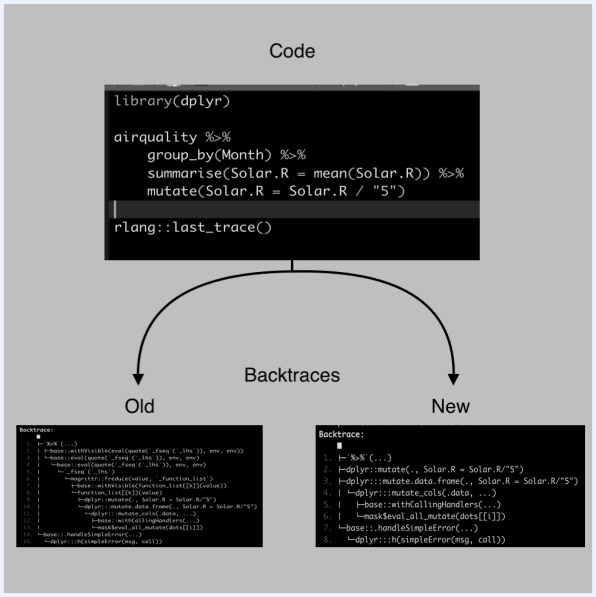
Our #rstats #programming #tipoftheday - Did you know you can use the enframe() function from the #tibble package to easily convert named vectors and lists to a two column #dataframe? Heres some sample code!



Tweeted on: 2019-05-23 13:26:59, Retweets: 6, Likes: 7

#rstats and #debugging

Were looking forward to the latest {{ magrittr }} 2.0 release for #rstats - In addition to evaluation changes and speed improvements, the new version cleans up the error backtrace to ease the process of #debugging (Left: Old backtrace, Right: New)



Tweeted on: 2020-09-03 16:19:42, Retweets: 2, Likes: 7

#rstats and #tidyverse

Hot off the presses, {{ dplyr }} 1.0.4 implements two new functions which perform column-wise filtering across a data frame - if_all() checks that all columns satisfy the given condition, while if_any() checks that any one of them does. #rstats #tidyverse

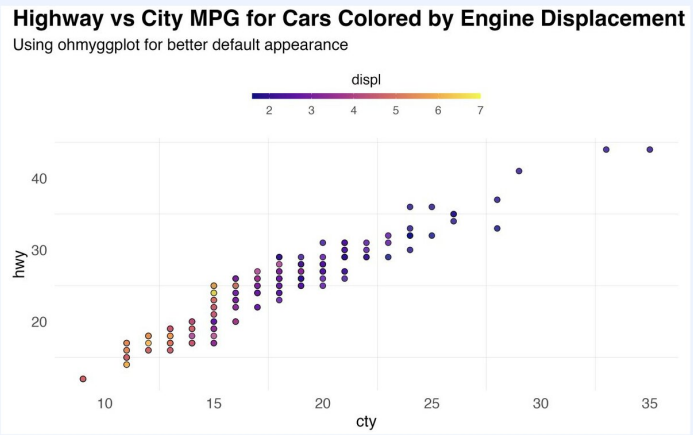
```
> mtcars %>%
+     filter(if_all(c(gear, carb), `>`, 4))
               mpg cyl disp  hp drat   wt qsec vs am gear carb
Ferrari Dino  19.7   6  145 175 3.62 2.77 15.5  0  1    5    6
Maserati Bora 15.0   8  301 335 3.54 3.57 14.6  0  1    5    8
> mtcars %>%
+     filter(if_any(c(gear, carb), `>`, 4))
                mpg cyl  disp  hp drat    wt qsec vs am gear carb
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
>
```

```r
library(dplyr)

mtcars %>%
    filter(if_all(c(gear, carb), `>`, 4))

mtcars %>%
    filter(if_any(c(gear, carb), `>`, 4))
```

Tweeted on: 2021-02-03 16:31:53, Retweets: 5, Likes: 7

#rstats and #dataviz

Came across @KejunYings package {{ ohmyggplot }}, which sets certain aesthetic and stylistic defaults for #ggplot2 that make the default plot appearance much nicer with only one single initialization line. Check it out! #rstats #dataviz https://github.com/albert-ying/ohmyggplot

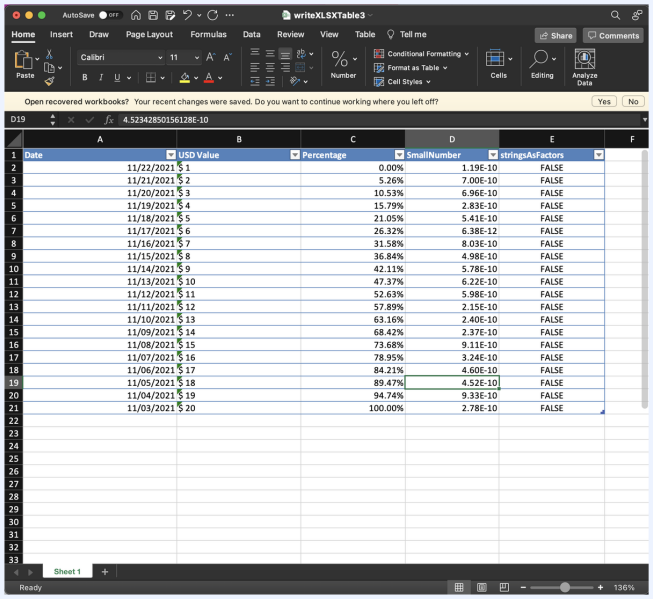


```
library(ggplot2)
library(ohmyggplot)

oh_my_ggplot()

p <- ggplot(mpg) +
    geom_point(aes(x = cty, y = hwy, fill = displ), alpha = 0.8) +
    theme(legend.position = "top") +
    labs(title = "Highway vs City MPG for Cars Colored by Engine Displacement",
         subtitle = "Using ohmyggplot for better default appearance") +
    better_fill_legend
p
```

Tweeted on: 2021-11-11 18:47:20, Retweets: 5, Likes: 7

#rstats and #dataframes

The {{ openxlsx }} package allows easy creation of #excel file from #rstats #dataframes. Compared to existing solutions, it maintains the same relative performance, adds many styling/formatting options, while also removing a dependency on #Java.



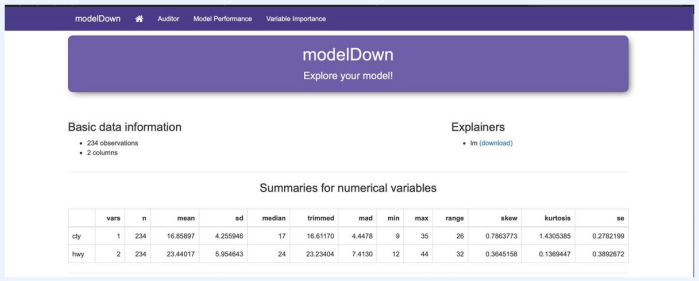
Date	USD Value	Percentage	SmallNumber	stringsAsFactors
11/22/2021	$ 1	0.00%	1.19E-10	FALSE
11/21/2021	$ 2	5.26%	7.00E-10	FALSE
11/20/2021	$ 3	10.53%	6.96E-10	FALSE
11/19/2021	$ 4	15.79%	2.83E-10	FALSE
11/18/2021	$ 5	21.05%	5.41E-10	FALSE
11/17/2021	$ 6	26.32%	6.38E-12	FALSE
11/16/2021	$ 7	31.58%	8.03E-10	FALSE
11/15/2021	$ 8	36.84%	4.98E-10	FALSE
11/14/2021	$ 9	42.11%	5.78E-10	FALSE
11/13/2021	$ 10	47.37%	6.22E-10	FALSE
11/12/2021	$ 11	52.63%	5.98E-10	FALSE
11/11/2021	$ 12	57.89%	2.15E-10	FALSE
11/10/2021	$ 13	63.16%	2.40E-10	FALSE
11/09/2021	$ 14	68.42%	2.37E-10	FALSE
11/08/2021	$ 15	73.68%	9.11E-10	FALSE
11/07/2021	$ 16	78.95%	3.24E-10	FALSE
11/06/2021	$ 17	84.21%	4.60E-10	FALSE
11/05/2021	$ 18	89.47%	4.52E-10	FALSE
11/04/2021	$ 19	94.74%	9.33E-10	FALSE
11/03/2021	$ 20	100.00%	2.78E-10	FALSE

```
library(tidyverse)
library(openxlsx)

options("openxlsx.borderColour" = "#F07649")
options("openxlsx.dateFormat" = "mm/dd/yyyy")
options("openxlsx.numFmt" = NULL) ## For default style rounding of numeric columns

df <- tibble("Date" = Sys.Date()-0:19,
             "USD Value" = paste("$",1:20),
             "Percentage" = seq(0, 1, length.out=20),
             "SmallNumber" = runif(20) / 1E9,  stringsAsFactors = FALSE)

class(df$Percentage) <- "percentage"
class(df$SmallNumber) <- "scientific"

write.xlsx(df, file = "writeXLSXTable3.xlsx", asTable = TRUE)
```

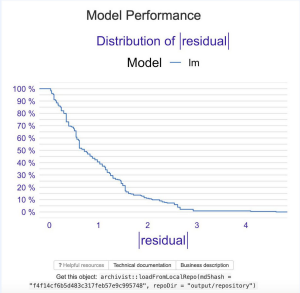
Tweeted on: 2021-12-20 18:10:52, Retweets: 3, Likes: 7

#rstats and #modeldown

A brand new member of the *down series of #rstats packages, #modeldown quickly and easily creates a #bootstrap-based #HTML page to showcase model fit and performance results!



	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
cty	1	234	16.85897	4.255946	17	16.61170	4.4478	9	35	26	0.7863773	1.4305385	0.2782199
hwy	2	234	23.44017	5.954643	24	23.23404	7.4130	12	44	32	0.3645158	0.1369447	0.3892672




```
# Load libraries
library(tidyverse)
library(modelDown) # devtools::install_github("MI2DataLab/modelDown")
library(DALEX)

# Create model
lm_model <- lm(cty ~ hwy, data = mpg)

# Define the explainer
explainer_lm <- explain(lm_model, data = mpg %>% select(cty, hwy), y = mpg$cty)

# Generate the HTML Page
modelDown::modelDown(explainer_lm)
```

Tweeted on: 2019-06-19 12:24:38, Retweets: 2, Likes: 6

#rstats and #tidyverse

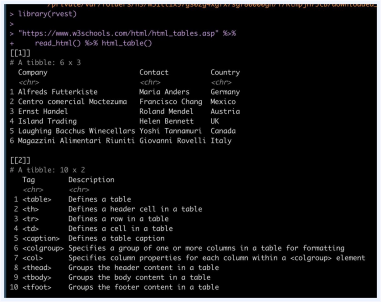
An #rstats pop-quiz: Provide a succinct human readable description of what this #dplyr chain is computing. #tidyverse

```
library(tidyverse)
library(nycflights13)
library(lubridate)

flights %>%
    mutate(Date = ymd(paste(year, month, day, sep = "-"))) %>%
    group_by(Date, carrier) %>%
    arrange(desc(dep_delay)) %>%
    slice(1) %>%
    group_by(carrier) %>%
    summarise(distance = mean(distance))
```

Tweeted on: 2020-07-30 15:26:57, Retweets: 4, Likes: 6

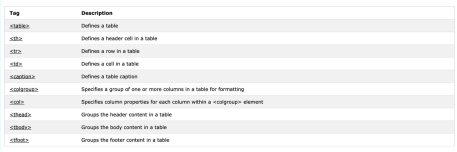
#rstats and #tidyverse

Our go-to tool for #webscraping, the brand new release of {{ rvest }} 1.0 improves the compatibility and performance of both table and text extraction from #html pages! #rstats #tidyverse



```
> library(rvest)
>
> "https://www.w3schools.com/html/html_tables.asp" %>%
+   read_html() %>% html_table()
[[1]]
# A tibble: 6 x 3
  Company                      Contact         Country
  <chr>                        <chr>           <chr>
1 Alfreds Futterkiste          Maria Anders    Germany
2 Centro comercial Moctezuma   Francisco Chang Mexico
3 Ernst Handel                 Roland Mendel   Austria
4 Island Trading               Helen Bennett   UK
5 Laughing Bacchus Winecellars Yoshi Tannamuri Canada
6 Magazzini Alimentari Riuniti Giovanni Rovelli Italy

[[2]]
# A tibble: 10 x 2
   Tag       Description
   <chr>     <chr>
 1 <table>   Defines a table
 2 <th>      Defines a header cell in a table
 3 <tr>      Defines a row in a table
 4 <td>      Defines a cell in a table
 5 <caption> Defines a table caption
 6 <colgroup> Specifies a group of one or more columns in a table for formatting
 7 <col>     Specifies column properties for each column within a <colgroup> element
 8 <thead>   Groups the header content in a table
 9 <tbody>   Groups the body content in a table
10 <tfoot>   Groups the footer content in a table
```

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy



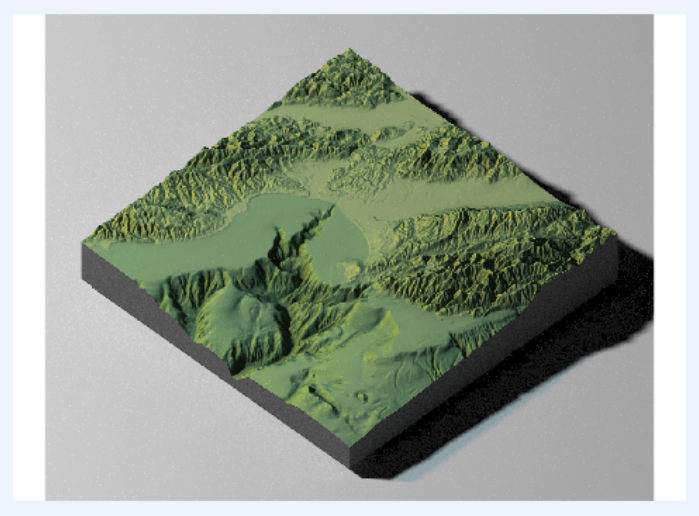
Tag	Description
<table>	Defines a table
<th>	Defines a header cell in a table
<tr>	Defines a row in a table
<td>	Defines a cell in a table
<caption>	Defines a table caption
<colgroup>	Specifies a group of one or more columns in a table for formatting
<col>	Specifies column properties for each column within a <colgroup> element
<thead>	Groups the header content in a table
<tbody>	Groups the body content in a table
<tfoot>	Groups the footer content in a table

```
library(rvest)

"https://www.w3schools.com/html/html_tables.asp" %>%
    read_html() %>% html_table()
```

Tweeted on: 2021-03-12 15:26:31, Retweets: 5, Likes: 6

#rstats and #GitHub

Were following the #GitHub README for the {{ rayshader }} #rstats package to learn how to render complex scenes in 3D, as well as export high quality snapshots from the rendered scene! More here: https://github.com/tylermorganwall/rayshader



```
library(rayshader)

montshadow = ray_shade(montereybay, zscale = 50, lambert = FALSE)
montamb = ambient_shade(montereybay, zscale = 50)
montereybay %>%
    sphere_shade(zscale = 10, texture = "imhof1") %>%
    add_shadow(montshadow, 0.5) %>%
    add_shadow(montamb, 0) %>%
    plot_3d(montereybay, zscale = 50, fov = 0, theta = -45, phi = 45,
            windowsize = c(1000, 800), zoom = 0.75,
            water = TRUE, waterdepth = 0, wateralpha = 0.5, watercolor = "lightblue",
            waterlinecolor = "white", waterlinealpha = 0.5)

render_highquality(clamp_value=10,samples=256)
```

Tweeted on: 2021-07-27 18:50:51, Retweets: 2, Likes: 6

#rstats and #python

A common data manipulation task in #python is to quickly take two list objects and construct a #dictionary. The zip() function makes this seamless. For our #rstats followers, how would you produce a named list from two distinct lists/vectors in a similar fashion? 🤔

```
my_list = ["apple", "orange", "cherry"]
my_properties = [{"color": "red", "size": "medium"},
                 {"color": "orange", "size": "medium"},
                 {"color": "red", "size": "small"}]

my_dict = dict(zip(my_list, my_properties))

Out[45]:
{'apple': {'color': 'red', 'size': 'medium'},
 'orange': {'color': 'orange', 'size': 'medium'},
 'cherry': {'color': 'red', 'size': 'small'}}
```

Tweeted on: 2021-11-19 18:16:27, Retweets: 3, Likes: 6

#rstats and #tidyverse

A major new release of #rstats, version 4.1, is now available. Among other features, it introduces a native pipe operator that behaves much the same way as the {{ magrittr }} pipe in the #tidyverse. (Left: Old, Right: New)

```
library(tidyverse)

mtcars %>%
    filter(wt > 3) %>%
    group_by(cyl) %>%
    summarise(mpg = mean(mpg))
```

```
library(tidyverse)

mtcars |>
    filter(wt > 3) |>
    group_by(cyl) |>
    summarise(mpg = mean(mpg))
```

Tweeted on: 2021-05-20 21:54:00, Retweets: 4, Likes: 5

#rstats and #tidyverse

The first public release of the new {{ clock }} package provides a set of utilities for manipulating #datetimes in #rstats, based on the #date library in C++. Here we group a set of airquality observations by month and compute summaries over the numeric variables. #tidyverse

```
# A tibble: 5 x 7
  Month      Ozone Solar.R  Wind  Temp   Day  Year
  <date>     <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl>
1 1973-05-01  23.6    181. 11.6   65.5  16    1973
2 1973-06-01  29.4    190. 10.3   79.1  15.5  1973
3 1973-07-01  59.1    216.  8.94  83.9  16    1973
4 1973-08-01  60.0    172.  8.79  84.0  16    1973
5 1973-09-01  31.4    167. 10.2   76.9  15.5  1973
```

```r
library(tidyverse)
library(clock)

airquality %>%
    mutate(Year = 1973,
           Date = date_build(Year, Month, Day),
           Month = date_group(Date, "month")) %>%
    group_by(Month) %>%
    summarise(across(where(is.numeric), mean, na.rm = TRUE))
```

Tweeted on: 2021-04-05 16:34:15, Retweets: 5, Likes: 4

#rstats and #dataframe

As its name suggests, the {{ arsenal }} package provides, well, an arsenal of tools for large scale summaries, such as variable summary tables, #dataframe comparison, and exporting results to html! #rstats

	4 (N=77)	5 (N=4)	6 (N=78)	8 (N=70)	Total (N=229)	p value
drv						< 0.001
4	23 (29.9%)	0 (0.0%)	32 (41.0%)	48 (68.6%)	103 (45.0%)	
f	54 (70.1%)	4 (100.0%)	42 (53.8%)	1 (1.4%)	101 (44.1%)	
r	0 (0.0%)	0 (0.0%)	4 (5.1%)	21 (30.0%)	25 (10.9%)	
trans						< 0.001
auto(av)	1 (1.3%)	0 (0.0%)	3 (3.8%)	0 (0.0%)	4 (1.7%)	
auto(l3)	2 (2.6%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	2 (0.9%)	
auto(l4)	24 (31.2%)	0 (0.0%)	29 (37.2%)	30 (42.9%)	83 (36.2%)	
auto(l5)	5 (6.5%)	0 (0.0%)	15 (19.2%)	17 (24.3%)	37 (16.2%)	
auto(l6)	0 (0.0%)	0 (0.0%)	2 (2.6%)	4 (5.7%)	6 (2.6%)	
auto(s4)	2 (2.6%)	0 (0.0%)	0 (0.0%)	1 (1.4%)	3 (1.3%)	
auto(s5)	1 (1.3%)	0 (0.0%)	1 (1.3%)	1 (1.4%)	3 (1.3%)	
auto(s6)	4 (5.2%)	2 (50.0%)	5 (6.4%)	5 (7.1%)	16 (7.0%)	
manual(m5)	32 (41.6%)	2 (50.0%)	18 (23.1%)	5 (7.1%)	57 (24.9%)	
manual(m6)	6 (7.8%)	0 (0.0%)	5 (6.4%)	7 (10.0%)	18 (7.9%)	

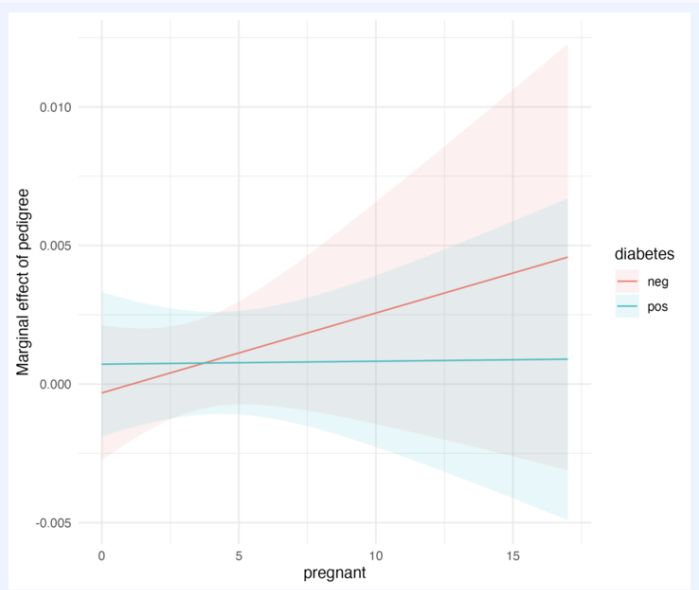
```
library(arsenal)
library(tidyverse)

mpg_copy <- mpg %>%
    select(-cty) %>%
    mutate(cty_other = hwy * 2) %>%
    slice(-c(1:5))

comparedf(mpg, mpg_copy)

tbl <- tableby(cyl ~ drv + trans, data=mpg_copy)
write2html(tbl, "output.html")
```

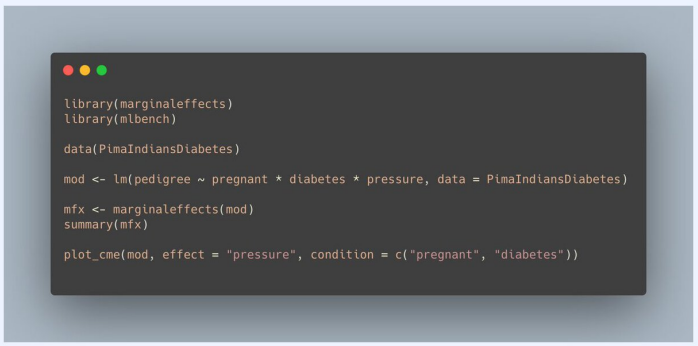
```
Compare Object

Function Call:
comparedf(x = mpg, y = mpg_copy)

Shared: 10 non-by variables and 229 observations.
Not shared: 2 variables and 5 observations.

Differences found in 10/10 variables compared.
0 variables compared have non-identical attributes.
>
```

Tweeted on: 2021-06-15 16:26:11, Retweets: 3, Likes: 4

#rstats and #dataviz

Now on #CRAN, {{ marginaleffects }} allows #rstats users to extract estimates of the marginal effect of a variable conditioned on the values of other variables, and visualize them quickly and seamlessly! #dataviz



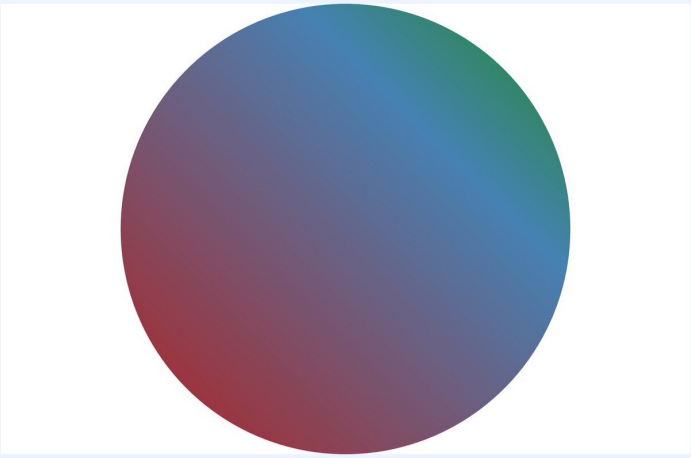


```
library(marginaleffects)
library(mlbench)

data(PimaIndiansDiabetes)

mod <- lm(pedigree ~ pregnant * diabetes * pressure, data = PimaIndiansDiabetes)

mfx <- marginaleffects(mod)
summary(mfx)

plot_cme(mod, effect = "pressure", condition = c("pregnant", "diabetes"))
```

Tweeted on: 2021-10-07 14:24:06, Retweets: 3, Likes: 4

#rstats and #tidyverse

#DidYouKnow that you can draw elegant #color gradients using the #rstats graphic engine? If you update to the latest version of R, plus install companion packages {{ ragg }} and {{ svglite }}, you can produce this! See more in the #tidyverse blog: https://www.tidyverse.org/blog/2022/02/new-graphic-features/



```
library(grid)

grid.circle(
    gp = gpar(
        fill = linearGradient(
            c("firebrick", "steelblue", "forestgreen"),
            stops = c(0, 0.7, 1)
        ),
        col = NA
    )
)
```

Tweeted on: 2022-03-08 21:38:37, Retweets: 3, Likes: 4

#rstats and #programming

#rstats #programming #tipoftheday: If you need to dynamically grow a #vector but know its maximum size in advance, pre-initialize it! Compare the #microbenchmark runtimes of the following two code samples - a 100x speedup!

Dynamically Growing Vectors

```
slow_expr <- function() {
    result <- NULL
    for (i in 1:100000) {
        result <- c(result, i)
    }
}
```

```
fast_expr <- function() {
    result <- integer(100000)
    for (i in 1:100000) {
        result[i] <- i
    }
}
```

microbenchmark::microbenchmark(slow_expr(), unit = "us")

```
Unit: microseconds
      expr      min      lq    mean   median       uq      max neval
slow_expr() 31677.34 38995.9 42249.1 40620.51 44659.66 86773.35   100
```

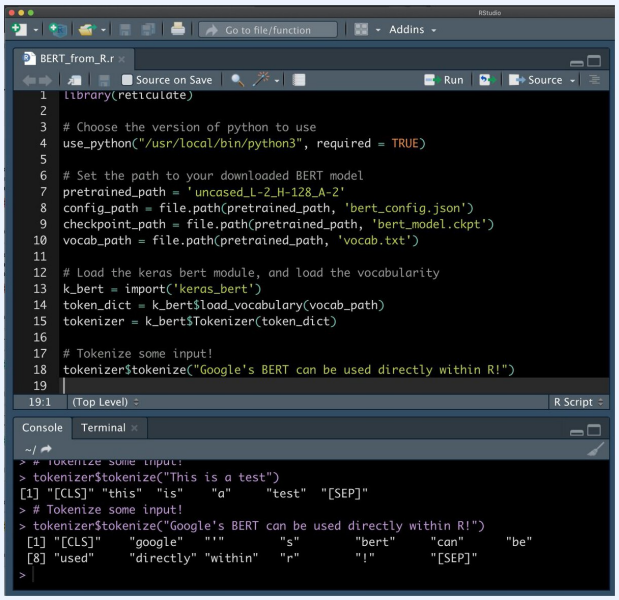
microbenchmark::microbenchmark(fast_expr(), unit = "us")

```
Unit: microseconds
       expr     min       lq     mean  median       uq     max neval
fast_expr() 319.639 351.5635 390.7448 376.057 405.8195 711.525   100
```

Tweeted on: 2019-04-17 14:53:37, Retweets: 4, Likes: 3

#rstats and #python

#didyouknow that you dont need #python in order to interact with @Googles #BERT? Using the #Keras-bert pip package and the reticulate #rstats package, we can load, train, and predict with BERT directly in R!




```
library(reticulate)
# Choose the version of python to use
use_python("/usr/local/bin/python3", required = TRUE)
# Set the path to your downloaded BERT model
pretrained_path = 'uncased_L-2_H-128_A-2'
config_path = file.path(pretrained_path, 'bert_config.json')
checkpoint_path = file.path(pretrained_path, 'bert_model.ckpt')
vocab_path = file.path(pretrained_path, 'vocab.txt')
# Load the keras bert module, and load the vocabularity
k_bert = import('keras_bert')
token_dict = k_bert$load_vocabulary(vocab_path)
tokenizer = k_bert$Tokenizer(token_dict)
# Tokenize some input!
tokenizer$tokenize("Google's BERT can be used directly within R!")
```

Tweeted on: 2020-05-26 19:59:02, Retweets: 2, Likes: 3

#rstats and #data

The @Socrata API platform is used by non-profits and government organizations all over the world to provide programmatic access to open #data. You can use the {{RSocrata}} #rstats package to seamlessly access this data (example shown: Mesa, AZ police reports)

```
library(RSocrata)

mesa <- read.socrata("https://data.mesaaz.gov/resource/39rt-2rfj.json")

View(mesa)
```

crime_id	crime_type	report_date	report_month	report_year	oc
20160010880	WARRANT ARREST-FOJ ARIZ JURISD	2016-01-01	Jan	2016	2(
20160057001	PROPERTY-LOST	2016-01-01	Jan	2016	2(
20160010024	PUBLIC SEXUAL INDEC-CONTACT	2016-01-01	Jan	2016	2(
20160010034	DUI - LIQUOR - DRUGS	2016-01-01	Jan	2016	2(
20160010095	MISSING JUVENILE	2016-01-01	Jan	2016	2(
20160010182	CRIMINAL DAMAGE-DEFACE - DV	2016-01-01	Jan	2016	2(
20160010262	INFORMATION-ALCOHOL RELATED	2016-01-01	Jan	2016	2(
20160010274	DEATH-NATURAL	2016-01-01	Jan	2016	2(
20160010275	WARRANT ARREST-FOJ ARIZ JURISD	2016-01-01	Jan	2016	2(
20160010287	PROPERTY-FOUND	2016-01-01	Jan	2016	2(
20160010303	DANGEROUS DRUG-POSS-USE	2016-01-01	Jan	2016	2(
20160010402	DISORDERLY CONDUCT-FIGHTING - DV	2016-01-01	Jan	2016	2(
20160010403	WARRANT ARREST-MESA CITY COURT	2016-01-01	Jan	2016	2(
20160010493	FAIL TO STOP/UNATTENDED VEHICLE - COLLISION	2016-01-01	Jan	2016	2(
20160010500	WARRANT ARREST-MESA CITY COURT	2016-01-01	Jan	2016	2(

Tweeted on: 2020-07-27 23:19:10, Retweets: 5, Likes: 3

#rstats and #tidyverse

With a new #svg rendering engine and a streamlined codebase extending {{ testthat }} 3, the 1.0 release of {{ vdiffr }} makes testing for regressions in graphical output even easier, helping ensure our #ggplot visualizations are as we expect. #rstats #tidyverse

```
library(testthat)
library(vdiffr)
library(ggplot2)

test_that("Check histogram appearance", {
    p <- ggplot(mpg) + geom_histogram(aes(x = cty))
    expect_doppelganger("CTY Histogram", p)
})
```

```
> test_that("Check histogram appearance", {
+     p <- ggplot(mpg) + geom_histogram(aes(x = cty))
+     expect_doppelganger("CTY Histogram", p)
+ })
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
Can't compare snapshot to reference when testing interactively
ℹ Run `devtools::test()` or `testthat::test_file()` to see changes
ℹ New path: /var/folders/x6/13byt17d36g2zh_dx587dbzm0000gn/T//RtmpfzpjX8/cty-histogram176a949a
43223.svg
Test passed 🐹
```

Tweeted on: 2021-06-25 21:53:01, Retweets: 3, Likes: 3

#rstats and #compsci

An #rstats pop-quiz: This function solves a popular #compsci algorithm, for vectors of length 2^n. What comp sci algorithm is it? Can you generalize the code to handle any arbitrary length vector of numeric values? 🤔

```r
mystery_function <- function(x) {
    ind <- length(x) %/% 2

    if (ind == 0) return(x)

    left <- mystery_function(x[0:ind])
    right <- mystery_function(x[(ind + 1):length(x)])

    new <- c()
    for (i in 1:length(left)) {
        if (left[i] < right[i]) {
            new <- c(new, left[i], right[i])
        } else {
            new <- c(new, right[i], left[i])
        }
    }

    return(new)
}
```

Tweeted on: 2021-11-29 22:11:41, Retweets: 4, Likes: 3

#rstats and #tidyverse

New release of {{ dtplyr }}, detailed in the #tidyverse blog. The package, which allows #dplyr verbs to be used with #datatable objects, gains support for a wider variety of verbs, combining #datatables speed with #dplyrs elegant syntax. #rstats

```
# A tibble: 53,940 × 11
   carat cut       color clarity depth table price     x     y     z   PPC
   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl> <dbl>
 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43 1417.
 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31 1552.
 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31 1422.
 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63 1152.
 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75 1081.
 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48 1400
 7  0.24 Very Good I     VVS1     62.3    57   336  3.95  3.98  2.47 1400
 8  0.26 Very Good H     SI1      61.9    55   337  4.07  4.11  2.53 1296.
 9  0.22 Fair      E     VS2      65.1    61   337  3.87  3.78  2.49 1532.
10  0.23 Very Good H     VS1      59.4    61   338  4     4.05  2.39 1470.
# … with 53,930 more rows
```

```
library(dtplyr)
library(tidyverse)

dt <- data.table::data.table(diamonds)

dt %>%
    mutate(PPC = price / carat) %>%
    replace_na(list(PPC = 0)) %>%
    as_tibble()
```

Tweeted on: 2021-12-07 15:48:09, Retweets: 4, Likes: 3

#rstats and #python

While we do the majority of our random #sampling in #rstats, its useful to know that #python has its own module called random, where you can quickly and easily sample without (random.sample) and with (random.choices) replacement.

```
In [12]: import random
    ...:
    ...: L = [1, 2, 3, 4, 5]

In [13]: random.sample(L, 5)
Out[13]: [4, 1, 2, 3, 5]

In [14]: random.choices(L, k=5)
Out[14]: [3, 1, 5, 1, 3]
```

Tweeted on: 2022-04-20 16:37:33, Retweets: 5, Likes: 3

#rstats

Minor but important update in the release of #rstats 4.1.2 that not only fixes several segfaults, but also includes performance improvements: In particular, a try() statement in some cases is now much faster, like in this block. Technical details here: https://stat.ethz.ch/pipermail/r-devel/2021-September/081101.html

```
my_list <- rep(list(airquality), 1000)

my_function <- function(x) {
    I(x)
}

try({
    do.call(my_function, list(x = my_list))
})
```

Tweeted on: 2021-11-10 18:38:51, Retweets: 3, Likes: 2

#rstats and #tidyverse

The 💁‍♀️ {{ maybe }} 💁‍♀️ package implements a #monad which allows for functions to be written which gracefully handle invalid inputs, like this safe_log() function! {{ maybe }} this will be useful to us in our #rstats analysis! #tidyverse

```
library(maybe)

safe_log <- function(x, base = exp(1)) {
    if (x < 0) nothing() else just(log(x, base = base))
}

safe_log(5)
# Just
# [1] 1.609438

safe_log(-5)
# Nothing>
```

Tweeted on: 2022-02-01 17:46:27, Retweets: 5, Likes: 2

#rstats and #TalkLikeAPirateDay

Want to talk like a pirate today for #TalkLikeAPirateDay? Just add "ARRRRRrrrrrrrr!" to all your text strings with #rstats

```
> regular_text<-"Make me talk like a pirate!"
> pirate_text<-paste0(c(regular_text,"ARRRRrrrr!"),collapse =" ")
> pirate_text
[1] "Make me talk like a pirate! ARRRRrrrr!"
```

Tweeted on: 2019-09-19 14:15:30, Retweets: 3, Likes: 1

#rstats and #tidyverse

(3) The position of new columns generated by mutate() can now be specified upon creation, with a simple .before and .after argument. #rstats #tidyverse #TidyTuesday

```
library(dplyr)
library(ggplot2)

mpg %>%
    mutate(cty_squared = cty^2, .after = cty) %>%
    mutate(manu_model = paste(manufacturer, model), .before = model)
```

```
# A tibble: 234 x 13
   manufacturer manu_model  model  displ  year   cyl trans  drv     cty cty_squared  hwy fl    class
   <chr>        <chr>       <chr>  <dbl> <int> <int> <chr>  <chr> <int>       <dbl> <int> <chr> <chr>
 1 audi         audi a4     a4       1.8  1999     4 auto(… f        18         324    29 p     comp…
 2 audi         audi a4     a4       1.8  1999     4 manua… f        21         441    29 p     comp…
 3 audi         audi a4     a4       2    2008     4 manua… f        20         400    31 p     comp…
 4 audi         audi a4     a4       2    2008     4 auto(… f        21         441    30 p     comp…
 5 audi         audi a4     a4       2.8  1999     6 auto(… f        16         256    26 p     comp…
 6 audi         audi a4     a4       2.8  1999     6 manua… f        18         324    26 p     comp…
 7 audi         audi a4     a4       3.1  2008     6 auto(… f        18         324    27 p     comp…
 8 audi         audi a4 qu… a4 qu…   1.8  1999     4 manua… 4        18         324    26 p     comp…
 9 audi         audi a4 qu… a4 qu…   1.8  1999     4 auto(… 4        16         256    25 p     comp…
10 audi         audi a4 qu… a4 qu…   2    2008     4 manua… 4        20         400    28 p     comp…
# … with 224 more rows
```

Tweeted on: 2020-03-17 17:55:40, Retweets: 4, Likes: 6

#rstats and #tidyverse

(2) Eliminating the need for the select_if() function, select() now allows direct selection of columns based on their type, like here where we select all the character columns from the mpg data #rstats #tidyverse #tidytuesday

```
library(dplyr)
library(ggplot2)

mpg %>%
    select(is.character)
```

```
# A tibble: 234 x 6
   manufacturer model      trans      drv   fl    class
   <chr>        <chr>      <chr>      <chr> <chr> <chr>
 1 audi         a4         auto(l5)   f     p     compact
 2 audi         a4         manual(m5) f     p     compact
 3 audi         a4         manual(m6) f     p     compact
 4 audi         a4         auto(av)   f     p     compact
 5 audi         a4         auto(l5)   f     p     compact
 6 audi         a4         manual(m5) f     p     compact
 7 audi         a4         auto(av)   f     p     compact
 8 audi         a4 quattro manual(m5) 4     p     compact
 9 audi         a4 quattro auto(l5)   4     p     compact
10 audi         a4 quattro manual(m6) 4     p     compact
# … with 224 more rows
```

Tweeted on: 2020-03-17 17:55:40, Retweets: 2, Likes: 6