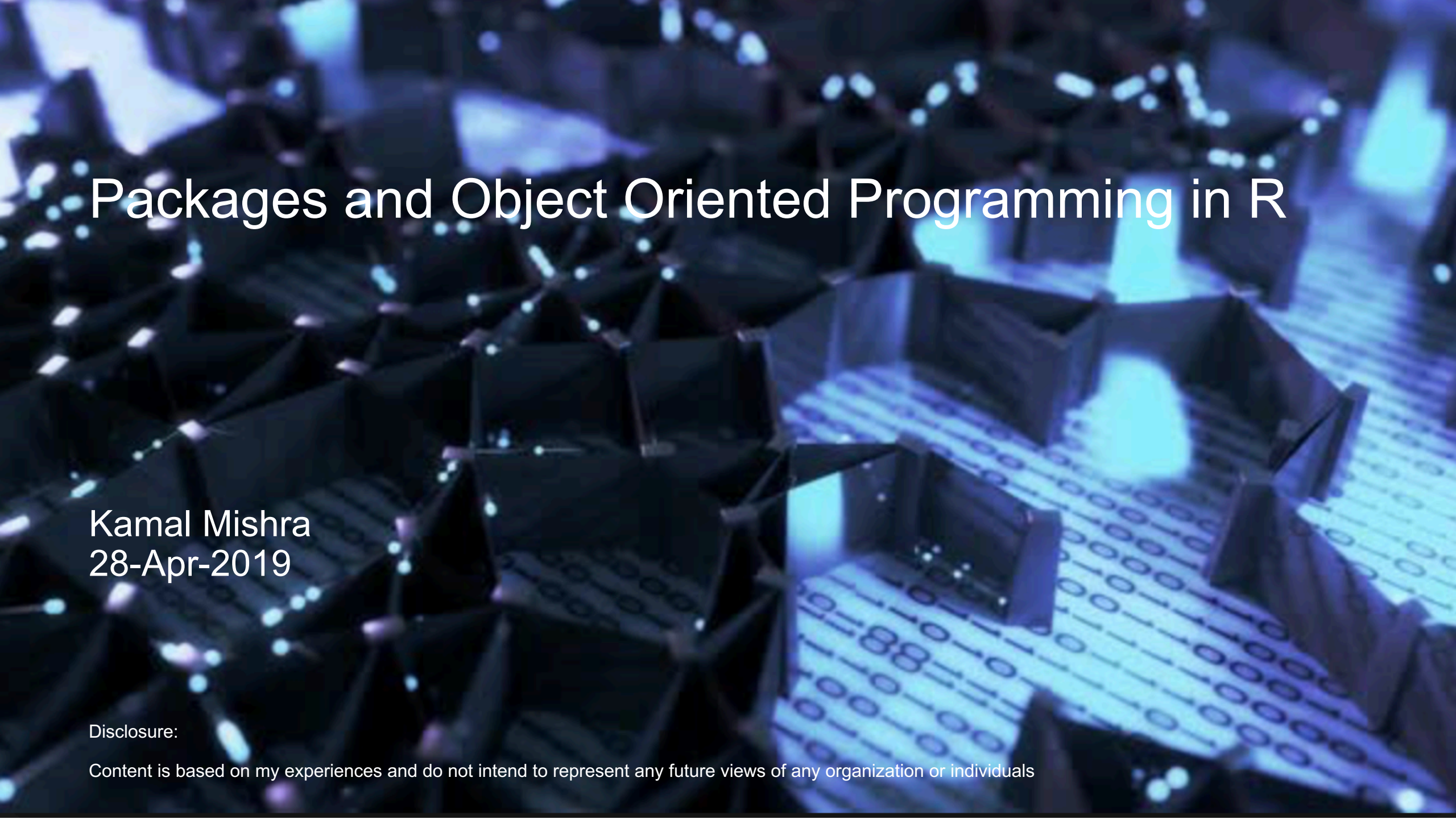


Packages and Object Oriented Programming in R

Kamal Mishra
28-Apr-2019

Disclosure:

Content is based on my experiences and do not intend to represent any future views of any organization or individuals

Disclaimer

Disclaimer – The contents of this document are to best of my knowledge and experiences. Some data and names MAY BE tweaked to take care of data privacy and business sensitivity aspects. The information provided is purely to highlight experience gathered with clear business impact created as part of opportunities and NO WAY RELATES TO ANY ORGANIZATION.

Safe Harbor

This document contains forward-looking statements within the meaning of Section 27A of the Securities Act of 1933, as amended, and Section 21E of the Securities Exchange Act of 1934, as amended. The forward-looking statements contained herein are subject to certain risks and uncertainties that could cause actual results to differ materially from those reflected in the forward-looking statements. I undertake no duty to update any forward-looking statements.

Content

- Introduction / About me
- Motivation
- Packages in R
  - Top Packages to look at
  - Snapshots on how they are used / popular
  - By usage areas / features
- Object Oriented Programming in R

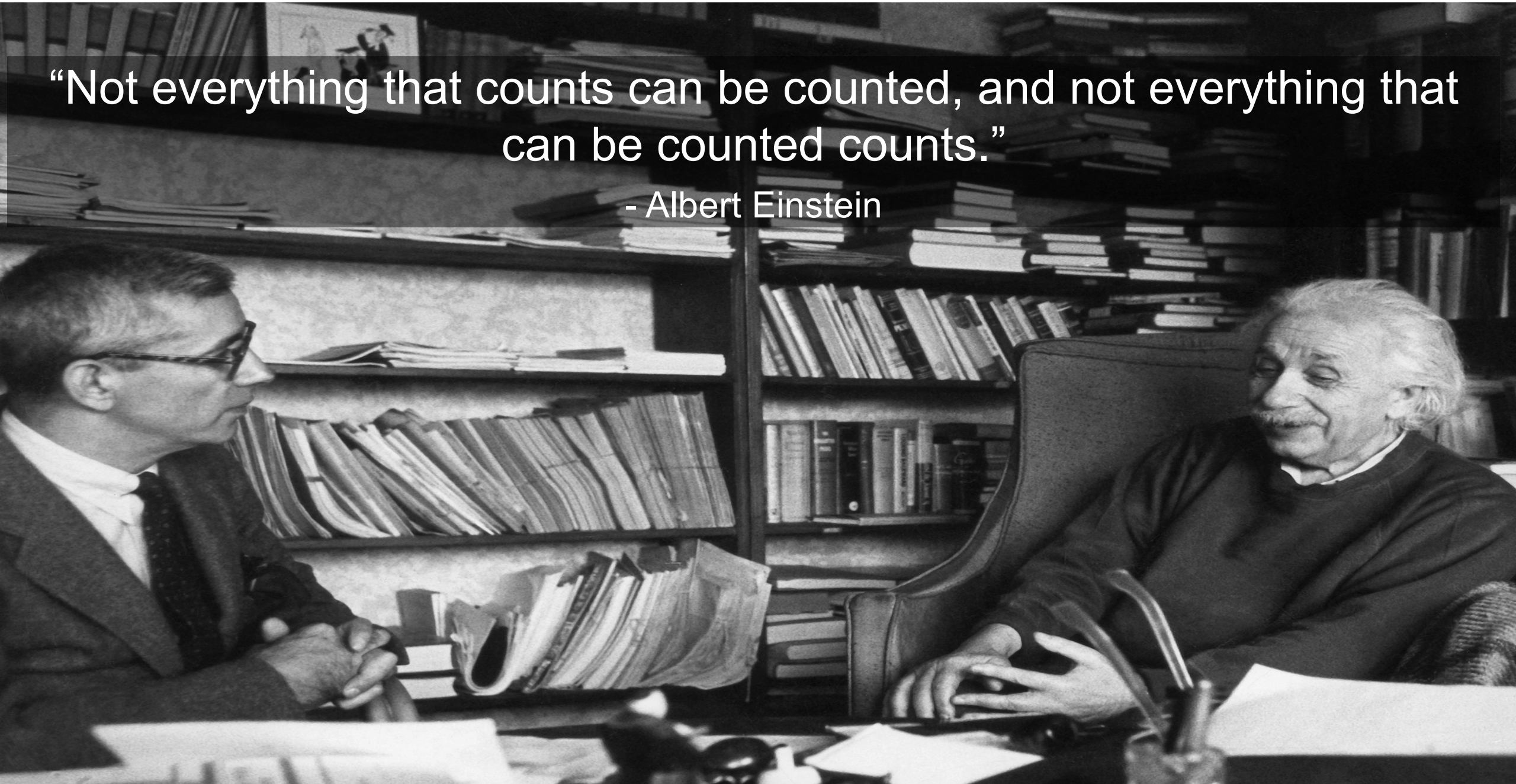
Disclosure:

Content is based on my experiences and do not intend to represent any future views of any organization or individuals

About Me

- Academia
- Professional Experience
- Interests

Motivation


“Not everything that counts can be counted, and not everything that can be counted counts.”
- Albert Einstein

Top data manipulation packages

DATA MANIPULATION

Package / Library	Commits	Contributors	Features
dplyr	4 354	136	• powerful library for data wrangling • works with local data frames and remote database tables • precise and simple command syntax
data.table	3 211	43	• quick aggregation of large data • laconic flexible syntax and a wide suite of useful functions • friendly file reader and parallel file writer
lubridate	1 427	45	• a set of functions to work with date and time format • easy and fast parsing of date-time data • expanded mathematical operations on time data
jsonlite	908	11	• robust and quick parsing JSON objects in R • great tool for interacting with web APIs and building pipelines • functions to stream, validate, and prettify JSON data

Source: KDNuggets

dplyr

- Mainly around data manipulation
- Core 5 functions
  - **Select** certain columns of data
  - **Filter** your data to select specific rows
  - **Arrange** the rows of your data into an order
  - **Mutate** your data frame to contain new columns
  - **Summarize** chunks of data in some way
- Other functions – sample, group by, pipe

Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**
Extract rows that meet logical criteria.

dplyr::**distinct(iris)**
Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**
Randomly select fraction of rows.

dplyr::**sample_n(iris, 10, replace = TRUE)**
Randomly select n rows.

dplyr::**slice(iris, 10:15)**
Select rows by position.

dplyr::**top_n(storms, 2, date)**
Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&,|,!,xor,any,all	Boolean operators

Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**
Select columns by name or helper function.

Helper functions for select - ?select

select(iris, contains("."))
Select columns whose name contains a character string.

select(iris, ends_with("Length"))
Select columns whose name ends with a character string.

select(iris, everything())
Select every column.

select(iris, matches(".t."))
Select columns whose name matches a regular expression.

select(iris, num_range("x", 1:5))
Select columns named x1, x2, x3, x4, x5.

select(iris, one_of(c("Species", "Genus")))
Select columns whose names are in a group of names.

select(iris, starts_with("Sepal"))
Select columns whose name starts with a character string.

select(iris, Sepal.Length:Petal.Width)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, -Species)
Select all columns except Species.

dplyr (contd.)

Summarise Data

dplyr::**summarise(iris, avg = mean(Sepal.Length))**
Summarise data into single row of values.
dplyr::**summarise_each(iris, funs(mean))**
Apply summary function to each column.
dplyr::**count(iris, Species, wt = Sepal.Length)**
Count number of rows with each unique value of variable (with or without weights).

Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

dplyr::**first**
First value of a vector.
dplyr::**last**
Last value of a vector.
dplyr::**nth**
Nth value of a vector.
dplyr::**n**
# of values in a vector.
dplyr::**n_distinct**
# of distinct values in a vector.
IQR
IQR of a vector.
min
Minimum value in a vector.
max
Maximum value in a vector.
mean
Mean value of a vector.
median
Median value of a vector.
var
Variance of a vector.
sd
Standard deviation of a vector.

Group Data

dplyr::**group_by(iris, Species)**
Group data into rows with the same value of Species.
dplyr::**ungroup(iris)**
Remove grouping information from data frame.

iris %>% group_by(Species) %>% summarise(…)
Compute separate summary row for each group.

Make New Variables

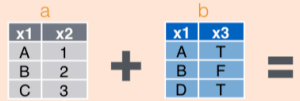
dplyr::**mutate(iris, sepal = Sepal.Length + Sepal. Width)**
Compute and append one or more new columns.
dplyr::**mutate_each(iris, funs(min_rank))**
Apply window function to each column.
dplyr::**transmute(iris, sepal = Sepal.Length + Sepal. Width)**
Compute one or more new columns. Drop original columns.

Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

dplyr::**lead**
Copy with values shifted by 1.
dplyr::**lag**
Copy with values lagged by 1.
dplyr::**dense_rank**
Ranks with no gaps.
dplyr::**min_rank**
Ranks. Ties get min rank.
dplyr::**percent_rank**
Ranks rescaled to [0, 1].
dplyr::**row_number**
Ranks. Ties got to first value.
dplyr::**ntile**
Bin vector into n buckets.
dplyr::**between**
Are values between a and b?
dplyr::**cume_dist**
Cumulative distribution.
dplyr::**cumall**
Cumulative `all`
dplyr::**cumany**
Cumulative `any`
dplyr::**cummean**
Cumulative `mean`
cumsum
Cumulative `sum`
cummax
Cumulative `max`
cummin
Cumulative `min`
cumprod
Cumulative `prod`
pmax
Element-wise `max`
pmin
Element-wise `min`

iris %>% group_by(Species) %>% mutate(…)
Compute new variables by group.

Combine Data Sets



Mutating Joins

dplyr::**left_join(a, b, by = "x1")**
Join matching rows from b to a.

dplyr::**right_join(a, b, by = "x1")**
Join matching rows from a to b.

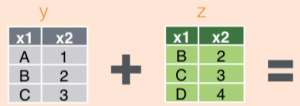
dplyr::**inner_join(a, b, by = "x1")**
Join data. Retain only rows in both sets.

dplyr::**full_join(a, b, by = "x1")**
Join data. Retain all values, all rows.

Filtering Joins

dplyr::**semi_join(a, b, by = "x1")**
All rows in a that have a match in b.

dplyr::**anti_join(a, b, by = "x1")**
All rows in a that do not have a match in b.



Set Operations

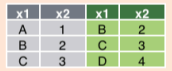
dplyr::**intersect(y, z)**
Rows that appear in both y and z.

dplyr::**union(y, z)**
Rows that appear in either or both y and z.

dplyr::**setdiff(y, z)**
Rows that appear in y but not z.

Binding

dplyr::**bind_rows(y, z)**
Append z to y as new rows.



dplyr::**bind_cols(y, z)**
Append z to y as new columns.
Caution: matches rows by position.

data.table

- Enhanced data.frame
- For data manipulation
- Core functions
  - File reader and writer
  - Aggregations
  - Updates
- Internally optimizes expressions of the form DT[col == value] or DT[col %in% values]

Creating A `data.table`

```
> set.seed(45L)
> DT <- data.table(V1=c(1L,2L),
                   V2=LETTERS[1:3],
                   V3=round(rnorm(4),4),
                   V4=1:12)
```

Create a `data.table` and call it `DT`

Subsetting Rows Using `i`

```
> DT[3:5,]
> DT[3:5]
> DT[V2=="A"]
> DT[V2 %in% c("A","C")]
```

Select 3rd to 5th row
Select 3rd to 5th row
Select all rows that have value `A` in column `V2`
Select all rows that have value `A` or `C` in column `V2`

Manipulating on Columns in `j`

```
> DT[,V2]
[1] "A" "B" "C" "A" "B" "C" ...
> DT[,.(V2,V3)]
> DT[,sum(V1)]
[1] 18
> DT[,.(sum(V1),sd(V3))]
   V1       V2
1: 18 0.4546055
> DT[,.(Aggregate=sum(V1),
        Sd.V3=sd(V3))]
   Aggregate     Sd.V3
1:        18 0.4546055
> DT[,.(V1,Sd.V3=sd(V3))]

> DT[,.(print(V2),
        plot(V3),
        NULL)]
```

Return `V2` as a vector

Return `V2` and `V3` as a `data.table`
Return the sum of all elements of `V1` in a vector
Return the sum of all elements of `V1` and the std. dev. of `V3` in a `data.table`

The same as the above, with new names

Select column `V2` and compute std. dev. of `V3`, which returns a single value and gets recycled
Print column `V2` and plot `V3`

Doing `j` by Group

```
> DT[,.(V4.Sum=sum(V4)),by=V1]
   V1 V4.Sum
1:  1     36
2:  2     42
> DT[,.(V4.Sum=sum(V4)),
        by=.(V1,V2)]
> DT[,.(V4.Sum=sum(V4)),
        by=sign(V1-1)]
   sign V4.Sum
1:    0     36
2:    1     42
> DT[,.(V4.Sum=sum(V4)),
        by=.(V1.01=sign(V1-1))]
> DT[1:5,.(V4.Sum=sum(V4)),
           by=V1]
> DT[,.N,by=V1]
```

Calculate sum of `V4` for every group in `V1`

Calculate sum of `V4` for every group in `V1` and `V2`
Calculate sum of `V4` for every group in `sign(V1-1)`

The same as the above, with new name for the variable you're grouping by
Calculate sum of `V4` for every group in `V1` after subsetting on the first 5 rows
Count number of rows for every group in `V1`

data.table (contd.)

Adding/Updating Columns By Reference in j Using :=

```
> DT[,V1:=round(exp(V1),2)]
> DT
     V1 V2      V3 V4
1: 2.72  A -0.1107  1
2: 7.39  B -0.1427  2
3: 2.72  C -1.8893  3
4: 7.39  A -0.3571  4
...
> DT[,c("V1","V2"):=list(round(exp(V1),2),
                         LETTERS[4:6])]
> DT[,':='(V1=round(exp(V1),2),
           V2=LETTERS[4:6])][]
        V1 V2      V3 V4
1:   15.18  D -0.1107  1
2: 1619.71  E -0.1427  2
3:   15.18  F -1.8893  3
4: 1619.71  D -0.3571  4
> DT[,V1:=NULL]
> DT[,c("V1","V2"):=NULL]
> Cols.chosen=c("A","B")
> DT[,Cols.Chosen:=NULL]

> DT[,(Cols.Chosen):=NULL]
```

V1 is updated by what is after :=
Return the result by calling DT

Columns V1 and V2 are updated by what is after :=
Alternative to the above one. With [], you print the result to the screen

Remove V1
Remove columns V1 and V2

Delete the column with column name Cols.chosen

Delete the columns specified in the variable Cols.chosen

Indexing And Keys

```
> setkey(DT,V2)
> DT["A"]
   V1 V2      V3 V4
1:  1  A -0.2392  1
2:  2  A -1.6148  4
3:  1  A  1.0498  7
4:  2  A  0.3262 10
> DT[c("A","C")]
> DT["A",mult="first"]

> DT["A",mult="last"]

> DT[c("A","D")]
   V1 V2      V3 V4
1:  1  A -0.2392  1
2:  2  A -1.6148  4
3:  1  A  1.0498  7
4:  2  A  0.3262 10
5: NA  D      NA NA
> DT[c("A","D"),nomatch=0]
   V1 V2      V3 V4
1:  1  A -0.2392  1
2:  2  A -1.6148  4
3:  1  A  1.0498  7
4:  2  A  0.3262 10
> DT[c("A","C"),sum(V4)]

> DT[c("A","C"),
     sum(V4),
     by=.EACHI]
   V2 V1
1:  A 22
2:  C 30
> setkey(DT,V1,V2)
> DT[.(2,"C")]
   V1 V2      V3 V4
1:  2  C  0.3262  6
2:  2  C -1.6148 12
> DT[.(2,c("A","C"))]
   V1 V2      V3 V4
1:  2  A -1.6148  4
2:  2  A  0.3262 10
3:  2  C  0.3262  6
4:  2  C -1.6148 12
```

A key is set on V2; output is returned invisibly
Return all rows where the key column (set to V2) has the value A

Return all rows where the key column (V2) has value A or C
Return first row of all rows that match value A in key column V2

Return last row of all rows that match value A in key column V2

Return all rows where key column V2 has value A or D

Return all rows where key column V2 has value A or D

Return total sum of V4, for rows of key column V2 that have values A or C

Return sum of column V4 for rows of V2 that have value A, and anohter sum for rows of V2 that have value C

Sort by V1 and then by V2 within each group of V1 (invisible)
Select rows that have value 2 for the first key (V1) and the value C for the second key (V2)

Select rows that have value 2 for the first key (V1) and within those rows the value A or C for the second key (V2)

lubridate

Date-times

2017-11-28 12:00:00

A **date-time** is a point on the timeline, stored as the number of seconds since 1970-01-01 00:00:00 UTC

*dt <- **as_datetime**(1511870400)*
"2017-11-28 12:00:00 UTC"

2017-11-28

A **date** is a day stored as the number of days since 1970-01-01

*d <- **as_date**(17498)*
"2017-11-28"

12:00:00

An hms is a **time** stored as the number of seconds since 00:00:00

*t <- hms::**as.hms**(85)*
00:01:25

PARSE DATE-TIMES (Convert strings or numbers to date-times)

1. Identify the order of the year (**y**), month (**m**), day (**d**), hour (**h**), minute (**m**) and second (**s**) elements in your data.
2. Use the function below whose name replicates the order. Each accepts a wide variety of input formats.

2017-11-28T14:02:00	**ymd_hms**(), **ymd_hm**(), **ymd_h**(). *ymd_hms("2017-11-28T14:02:00")*
2017-22-12 10:00:00	**ydm_hms**(), **ydm_hm**(), **ydm_h**(). *ydm_hms("2017-22-12 10:00:00")*
11/28/2017 1:02:03	**mdy_hms**(), **mdy_hm**(), **mdy_h**(). *mdy_hms("11/28/2017 1:02:03")*
1 Jan 2017 23:59:59	**dmy_hms**(), **dmy_hm**(), **dmy_h**(). *dmy_hms("1 Jan 2017 23:59:59")*
20170131	**ymd**(), **ydm**(). *ymd(20170131)*
July 4th, 2000	**mdy**(), **myd**(). *mdy("July 4th, 2000")*
4th of July '99	**dmy**(), **dym**(). *dmy("4th of July '99")*
2001: Q3	**yq**() Q for quarter. *yq("2001: Q3")*
2:01	hms::**hms**() Also lubridate::**hms**(), **hm**() and **ms**(), which return periods.* *hms::hms(sec = 0, min= 1, hours = 2)*

GET AND SET COMPONENTS

Use an accessor function to get a component.
d ## "2017-11-28"
day(d) ## 28

Assign into an accessor function to change a component in place.
day(d) <- 1
d ## "2017-11-01"

2018-01-31 11:59:59	**date**(x) Date component. *date(dt)*
2018-01-31 11:59:59	**year**(x) Year. *year(dt)* **isoyear**(x) The ISO 8601 year. **epiyear**(x) Epidemiological year.
2018-01-31 11:59:59	**month**(x, label, abbr) Month. *month(dt)*
2018-01-31 11:59:59	**day**(x) Day of month. *day(dt)* **wday**(x,label,abbr) Day of week. **qday**(x) Day of quarter.
2018-01-31 11:59:59	**hour**(x) Hour. *hour(dt)*
2018-01-31 11:59:59	**minute**(x) Minutes. *minute(dt)*
2018-01-31 11:59:59	**second**(x) Seconds. *second(dt)*
	**week**(x) Week of the year. *week(dt)* **isoweek**() ISO 8601 week. **epiweek**() Epidemiological week.
	**quarter**(x, with_year = FALSE) Quarter. *quarter(dt)*
	**semester**(x, with_year = FALSE) Semester. *semester(dt)*
	**am**(x) Is it in the am? *am(dt)* **pm**(x) Is it in the pm? *pm(dt)*
	dst(x) Is it daylight savings? *dst(d)*
	leap_year(x) Is it a leap year? *leap_year(d)*
	update(object, ..., simple = FALSE) *update(dt, mday = 2, hour = 1)*

Stamp Date-times

stamp() Derive a template from an example string and return a new function that will apply the template to date-times. Also **stamp_date**() and **stamp_time**().

1. Derive a template, create a function
   sf <- stamp("Created Sunday, Jan 17, 1999 3:34")
2. Apply the template to dates
   sf(ymd("2010-04-05"))
   ## [1] "Created Monday, Apr 05, 2010 00:00"

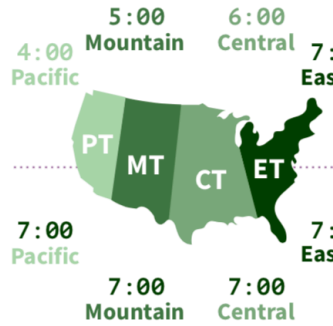
Tip: use a date with day > 12

Time Zones

R recognizes ~600 time zones. Each encodes the time zone, Daylight Savings Time, and historical calendar variations for an area. R assigns *one* time zone per vector.

Use the **UTC** time zone to avoid Daylight Savings.

OlsonNames() Returns a list of valid time zone names. *OlsonNames()*



with_tz(time, tzone = "") Get the **same date-time** in a new time zone (a new clock time). *with_tz(dt, "US/Pacific")*

force_tz(time, tzone = "") Get the **same clock time** in a new time zone (a new date-time). *force_tz(dt, "US/Pacific")*

lubridate (contd.)

Math with Date-times — Lubridate provides three classes of timespans to facilitate math with dates and date-times

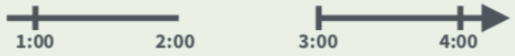
Math with date-times relies on the **timeline**, which behaves inconsistently. Consider how the timeline behaves during:

A normal day

nor <- ymd_hms("2018-01-01 01:30:00",tz="US/Eastern")

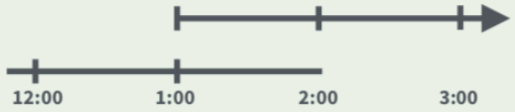
The start of daylight savings (spring forward)

gap <- ymd_hms("2018-03-11 01:30:00",tz="US/Eastern")



The end of daylight savings (fall back)

lap <- ymd_hms("2018-11-04 00:30:00",tz="US/Eastern")

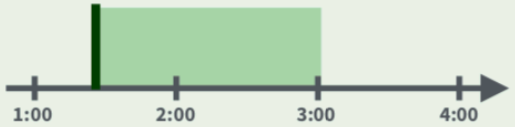


Leap years and leap seconds

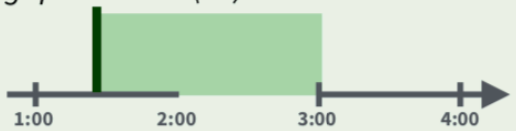
leap <- ymd("2019-03-01")

Periods track changes in clock times, which ignore time line irregularities.

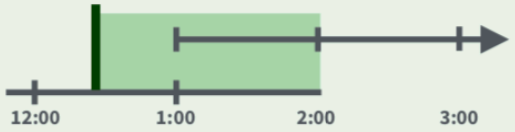
nor + minutes(90)



gap + minutes(90)



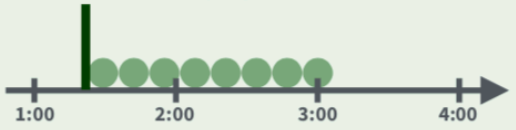
lap + minutes(90)



leap + years(1)

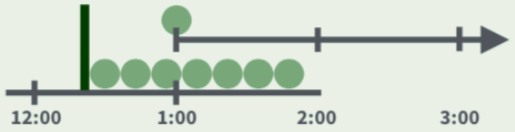
Durations track the passage of physical time, which deviates from clock time when irregularities occur.

nor + dminutes(90)

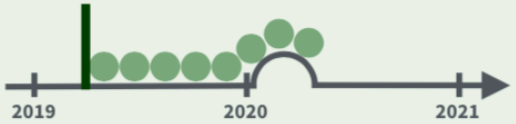


gap + dminutes(90)

lap + dminutes(90)

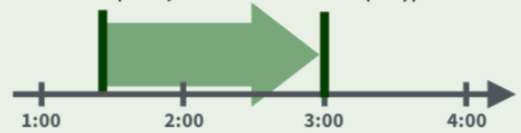


leap + dyears(1)



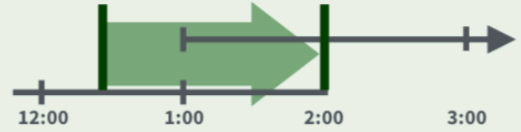
Intervals represent specific intervals of the timeline, bounded by start and end date-times.

interval(nor, nor + minutes(90))

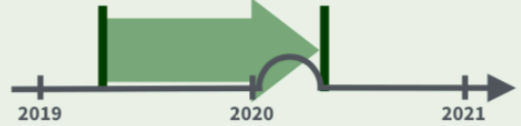


interval(gap, gap + minutes(90))

interval(lap, lap + minutes(90))



interval(leap, leap + years(1))



Top graphic display packages

GRAPHIC DISPLAYS

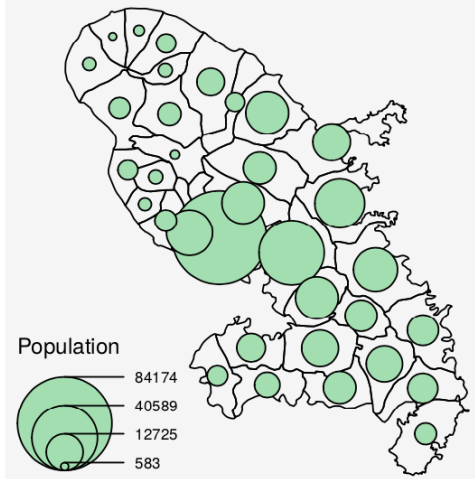
Package / Library	Commits	Contributors	Features
	3 903	133	• powerful implementation of the grammar of graphics visualization • developed static graphics system • takes care of plot specifications
Corrplot	299	8	• abilities to visualize correlation matrices and confidence intervals • contains algorithms to do matrix reordering • flexible appearance details settings
lattice	132	0	• high-level visualization system • emphasis on multivariate data • efficiently copes with nonstandard requirements

Source: KDNuggets

Thematic maps with cartography

Use cartography with spatial objects from sf or sp packages to create thematic maps.

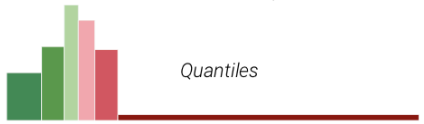
```
library(cartography)
library(sf)
mtq <- st_read("martinique.shp")
plot(st_geometry(mtq))
propSymbolsLayer(x = mtq, var = "P13_POP",
 legend.title.txt = "Population",
 col = "#a7dfb4")
```



Classification

Available methods are: quantile, equal, q6, fisher-jenks, mean-sd, sd, geometric progression...

```
bks1 <- getBreaks(v = var, nclass = 6,
 method = "quantile")
bks2 <- getBreaks(v = var, nclass = 6,
 method = "fisher-jenks")
pal <- carto.pal("green.pal",3, "wine.pal", 3)
hist(var, breaks = bks1, col = pal)
```




```
hist(var, breaks = bks2, col = pal)
```

Symbology

In most functions the x argument should be an sf object. sp objects are handled through spdf and df arguments.

Choropleth
```
choroLayer(x = mtq, var = "myvar",
 method = "quantile", nclass = 8)
```

Typology
```
typoLayer(x = mtq, var = "myvar")
```

Proportional Symbols
```
propSymbolsLayer(x = mtq, var = "myvar",
 inches = 0.1, symbols = "circle")
```

Colorized Proportional Symbols (relative data)
```
propSymbolsChoroLayer(x = mtq, var = "myvar",
 var2 = "myvar2")
```

Colorized Proportional Symbols (qualitative data)
```
propSymbolsTypoLayer(x = mtq, var = "myvar",
 var2 = "myvar2")
```

Double Proportional Symbols
```
propTrianglesLayer(x = mtq, var1 = "myvar",
 var2 = "myvar2")
```

OpenStreetMap Basemap (see rosm package)
```
tiles <- getTiles(x = mtq, type = "osm")
tilesLayer(tiles)
```

Isopleth (see SpatialPosition package)
```
smoothLayer(x = mtq, var = "myvar",
 typefct = "exponential", span = 500,
 beta = 2)
```

Discontinuities
```
discLayer(x = mtq.borders, df = mtq,
 var = "myvar", threshold = 0.5
```

Flows
```
propLinkLayer(x = mtq_link, df = mtq_df,
 var = "fij")
```

Dot Density
```
dotDensityLayer(x = mtq, var = "myvar")
```

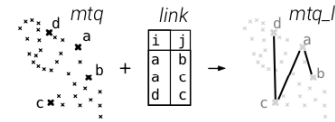
Labels
```
labelLayer(x = mtq, txt = "myvar",
 halo = TRUE, overlap = FALSE)
```

Transformations

Polygons to Grid
```
mtq_grid <- getGridLayer(x = mtq, cellsize = 3.6e+07,
  type = "hexagonal", var = "myvar")
```

Grids layers can be used by choroLayer() or propSymbolsLayer().

Points to Links
```
mtq_link <- getLinkLayer(x = mtq, df = link)
```

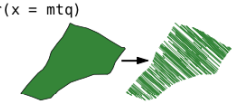


Links layers can be used by *LinkLayer().

Polygons to Borders
```
mtq_border <- getBorders(x = mtq)
```

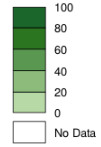
Borders layers can be used by discLayer() function

Polygons to Pencil Lines
```
mtq_pen <- getPencilLayer(x = mtq)
```



Legends

legendChoro()




```
legendChoro(pos = "topleft",
 title.txt = "legendChoro()",
 breaks = c(0,20,40,60,80,100),
 col = carto.pal("green.pal", 5),
 nodata = TRUE, nodata.txt = "No Data")
```

legendTypo()


```
legendTypo(title.txt = "legendTypo()",
 col = c("peru", "skyblue", "gray77"),
 categ = c("type 1", "type 2", "type 3"),
 nodata = FALSE)
```

legendCirclesSymbols()


```
legendCirclesSymbols(var = c(10,100),
 title.txt = "legendCirclesSymbols()",
 col = "#a7dfb4ff", inches = 0.3)
```

See also legendSquaresSymbols(), legendBarsSymbols(), legendGradLines(), legendPropLines() and legendPropTriangles().

Map Layout

North Arrow:
```
north(pos = "topright")
```

Scale Bar:
```
barscale(size = 5)
```

Full Layout:
```
layoutLayer(
 title = "Martinique",
 tabtitle = TRUE,
 frame = TRUE,
 author = "Author",
 sources = "Sources",
 north = TRUE,
 scale = 5)
```

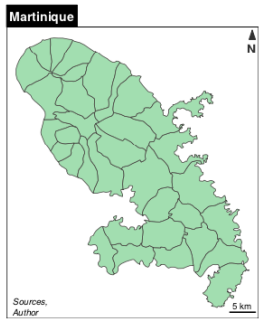
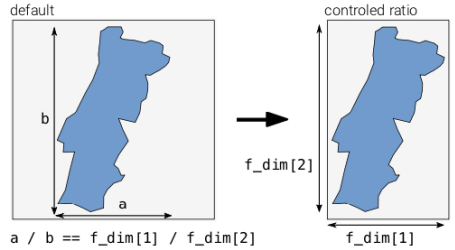


Figure Dimensions

Get figure dimensions based on the dimension ratio of a spatial object, figure margins and output resolution.

```
f_dim <- getFigDim(x = sf_obj, width = 500,
  mar = c(0,0,0,0))
png("fig.png", width = 500, height = f_dim[2])
par(mar = c(0,0,0,0))
plot(sf_obj, col = "#729fcf")
dev.off()
```



Color Palettes

```
carto.pal(pal1 = "blue.pal", n1 = 5,
 pal2 = sand.pal, n2 = 3)
```

```
display.carto.all(n = 8)
```

Top html widget packages

HTML WIDGETS

Package / Library	Commits	Contributors	Features
plotly	2 989	26	• rich features and plenty of available charts • web-based toolbox for building visualizations • abilities to make ggplot2 graphics interactive
ggvis	2 159	21	• implementation of an interactive grammar of graphic • incorporates shiny reactive programming model and dplyr grammar of data transformation
DT DataTables	1 919	21	• displays R matrices and data frames as interactive HTML tables • creates sortable tables with a minimum of code • many useful features and styling options for tables
rCharts	638	11	• interactive JS charts from R • tools for creation, customization, and sharing

Source: KDNuggets

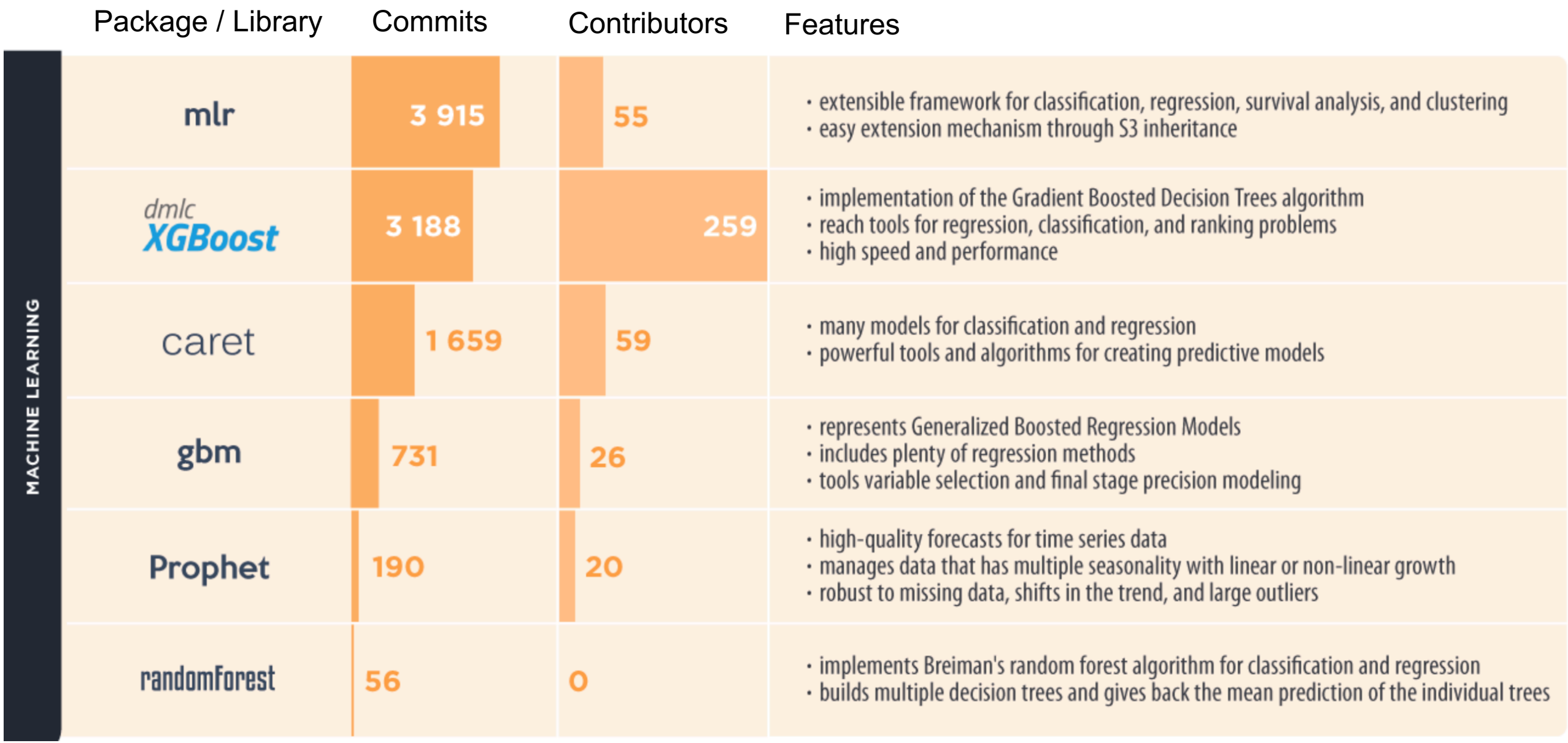
Top packages for reproducible research

REPRODUCIBLE RESEARCH

Package / Library	Commits	Contributors	Features
	5 467	96	• transparent tool for easy dynamic report generation in R • enables integration of R code into LaTeX, LyX, HTML, Markdown, AsciiDoc, and reStructuredText documents
	2 297	56	• next generation implementation of R Markdown based on pandoc • many static and dynamic output formats • abilities to define new formats for custom publishing requirements
slidify	302	7	• generates reproducible html5 slides from r markdown • allows embedded code chunks and mathematical formulas • rich sharing and customizing opportunities

Source: KDNuggets

Top machine learning packages



Source: KDNuggets

Other Machine Learning specific

- Boruta - A wrapper algorithm for all-relevant feature selection
- Arules - Mining Association Rules and Frequent Itemsets
- Forecast – Timeseries forecasting using ARIMA, ETS, STLM, TBATS, and neural network models
- Anomalize - Tidy Anomaly Detection using Twitter's AnomalyDetection method
- AnomalyDetection - AnomalyDetection R package from Twitter
- e1071 - Misc Functions of the Department of Statistics (e1071)
- Gbm - Generalized Boosted Regression Models
- Glmnet - Lasso and elastic-net regularized generalized linear models
- MXNet - MXNet brings flexible and efficient GPU computing and state-of-art deep learning to R
- CausalImpact - Causal inference using Bayesian structural time-series models

H2O packages

- Package for running H2O via its REST API from within R
- To communicate with a H2O instance, the version of the R package must match the version of H2O
- When connecting to a new H2O cluster, it is necessary to re-run the initializer

- Supports standard statistical models such as GLM, K-means, RF etc
  - For example, to run GLM, invoke h2o.glm with the H2O parsed data and parameters (response variable, error distribution, etc...) as arguments.
  - The operation will be done on the server associated with the data object where H2O is running, not within the R environment
- R only saves named objects, which uniquely identify data set, model etc on server (R → Rest API → returns JSON file format → R console ouput)
- Reference: H2O on Github - https://github.com/h2oai/h2o-3
- Reference: H2O documentation - http://docs.h2o.ai

Packages to search the web

- Rcurl – general network (HTTP/FTP…) client interface for R
- Curl – flexible web client for R
- Httr – user friendly Rcurl wrapper
- Rfacebook – access to facebook API via R
- Plumber – A library to expose existing R code as web API
- RSiteCatalyst – R client library for Adobe Analytics
- Shiny – simple interactive web applications with R

Database management

- RODBC – ODBC database access for R
- DBI – common interface between R and DBMS
- Elastic – wrapper for elastic search HTTP API
- Mongolite – streaming mongo client for R
- Roracle – OCI based Oracle database interface for R
- RPostgreSQL – R interface to PostgreSQL database system
- RSQLite – SQLite interface for R
- RNeo4j – Neo4j graph database driver

NLP specific

- text2vec – Fast Text Mining Framework for Vectorization and Word Embeddings
- tm – A comprehensive text mining framework for R
- openNLP – Apache OpenNLP Tools Interface.
- koRpus – An R Package for Text Analysis
- zipfR – Statistical models for word frequency distributions
- NLP - Basic functions for Natural Language Processing
- LDAvis - Interactive visualization of topic models
- SnowballC - Snowball stemmers based on the C libstemmer UTF-8 library
- Tidytext - Implementing tidy principles of Hadley Wickham to text mining

Packages for optimization

- lpSolve – Interface to Lp_solve to Solve Linear/Integer Programs
- Minqa - Derivative-free optimization algorithms by quadratic approximation
- Nloptr - NLopt is a free/open-source library for nonlinear optimization
- Ompr - Model mixed integer linear programs in an algebraic way directly in R
- Rglpk - R/GNU Linear Programming Kit Interface
- ROI - The R Optimization Infrastructure (‘ROI’) is a sophisticated framework for handling optimization problems in R

Bioinformatics and Biostatistics

- Bioconductor - Tools for the analysis and comprehension of high-throughput genomic data
- Genetics - Classes and methods for handling genetic data
- Gap - An integrated package for genetic data analysis of both population and family data
- Ape - Analyses of Phylogenetics and Evolution
- Pheatmap - Pretty heatmaps made easy

Packages for other languages

- rJava – low level R to java interface
- rPython – package allowing R to call Python
- Rpy2 – Python interface for R
- Runr – Run Julia and Bash from R
- RinRuby – a ruby library that integrates R interpreter in ruby
- R.matlab – read and write of MAT files together with R-to-Matlab connectivity
- RcppOctave – seamless interface to Octave and Matlab
- RSPerl – A bidirectional interface for calling R from Perl and Perl from R
- V8 – embedded javascript engine

Packages for Computer Vision

- magick – importing / converting to/from all formats / basic image manipulation
- imageR – image processing library based on “CImg” (interpolation, resizing, filtering, fourier transformations, denoising, gradients, blurring)
- OpenImageR – an image processing toolkit (hashing, edge detection, manipulation)

- R has options and good at interfacing
  - Rvision/ROpenCVLite – OpenCV from R
  - APIs also exist for traditional computer vision (Google vision API, Microsoft and IBM cognitive services)
  - On top of deep learning tools like Keras (kerasR package) and tensorflow (tensorflowR package) etc.

Object Oriented Programming in R

Base R provides 3 OOP systems:

- **S3 (informal implementation of functional OOP)**
- **S4 (more of a formal and rigorous rewrite of S3)**
- **RC (Reference Class, implements encapsulated OO, special type of R4 objects)**

Other OOP systems by CRAN packages:

- **R6 (simple to use, uses simpler S3, simpler mechanism for cross package subclassing, faster than RC)**
- **R.oo (some formalism on top of S3, possibility of mutable S3 objects)**
- **Proto (based on idea of prototypes)**

sloop package

```r
library(sloop)

otype(1:10)  # Numbers from 1 to 10
# "base"

otype(airquality) # New York Air Quality Measurements
# "S3"

otype(sleep) # Student's Sleep Data
# "S3"

mle_test_obj <- stats4::mle(function(y = 5)(y + 2)^ 2)
otype(mle_test_obj)
# "S4"
```

Sloop – is the helper package for OOP

Object types, Classes can be found out

Base package vs OO package

```r
is.object(1:10)
# FALSE

is.object(airquality)
# TRUE

is.object(mle_test_obj)
# TRUE
```

Difference is that OO objects have a "class" attribute

```r
attr(1:10,"class")
# NULL

attr(airquality, "class")
# "data.frame"
```

S3

- R's first and simplest OO system
- The only OO system used in base and stats packages
- S3 has no formal definition of a class - to make an object an instance of a class, you simply set the class attribute. You can do that during creation with structure(), or after the fact with class<-():

```
# Create and assign class in one step
x <- structure(list(), class = "my_class")

# Create, then set class
x <- list()
class(x) <- "my_class"
```

- You can determine the class of an S3 object with class(x), and see if an object is an instance of a class using inherits(x, "classname")

```
class(x)
#> [1] "my_class"
inherits(x, "my_class")
#> [1] TRUE
inherits(x, "your_class")
#> [1] FALSE
```

S4

- Classes are explicitly created
- To create a new class, we use the function setClass from the methods package

```
library(methods)
Stack <- setClass("Stack")
```

- This creates a new class called "Stack".
- We have not specified any attributes of the class
- Hence S4 will assume that it is an abstract class that is not supposed to be instantiated and we will get an error if we try.
- Instead we can create a vector based stack class with 2 arguments – slots and contains
- The slots argument is a list of attributes that objects of the class should have. Here specify that it should contain a vector called elements. The contains argument specifies which super classes the new class should have. We make VectorStack a subclass of Stack

```
VectorStack <- setClass("VectorStack",
                        slots = c(
                          elements = "vector"
                        ),
                        contains = "Stack")
(vs <- VectorStack())
## An object of class "VectorStack"
## Slot "elements":
## logical(0)
```

R6

- 2 special properties
  - Uses the encapsulated OOP paradigm, which means that methods belong to objects, not generics, and you call them like object$method()
  - R6 objects are mutable, which means that they are modified in place, and hence have reference semantics
- R6 is very similar to a base OOP system called reference classes or RC for short

TradeOff

- Once you've understood S3 and familiar with it, S4 is not too difficult to pick up, since underlying ideas are same
- S4 is just more formal, more verbose and protocol oriented
- If you are a larger team and want to collaborate, you can prefer S4
- S4 tends to require more upfront design than S3
- Example: Bioconductor package (large team effort where S4 is used to good effect)
- Bioconductor packages are not required to use S4, but most will because the key data structures (e.g. SummarizedExperiment, IRanges, DNAStringSet) are built using S4

- R6 is a profoundly different OO system from S3 and S4 because it is built on encapsulated objects, rather than generic functions
- R6 objects have also reference semantics, i.e. they can be modified in place

References

- R6 documentation: https://r6.r-lib.org
- Cheat sheets reference (across most areas): https://www.rstudio.com
- Cartography GitHub reference - https://github.com/riatelab/cartography
- Quick reference guide in R - https://www.statmethods.net
- Advanced R – the online version of advanced R book by Hadley wickham - http://adv-r.had.co.nz
- CRAN contributed documentation / free books on R - https://cran.r-project.org/other-docs.html
- The Art of R programming by Orielly - http://shop.oreilly.com/product/9781593273842.do
- Readings in applied data science (Stats 337 Stanford) - https://github.com/hadley/stats337
- Corner detection libraries - https://github.com/jcayzac/F9-Corner-Detection-Library