Coding Resources (general)

keyword searches (*stack overflow*)
slack channel / colleagues
platform websites, ebooks
in-lab code reviews

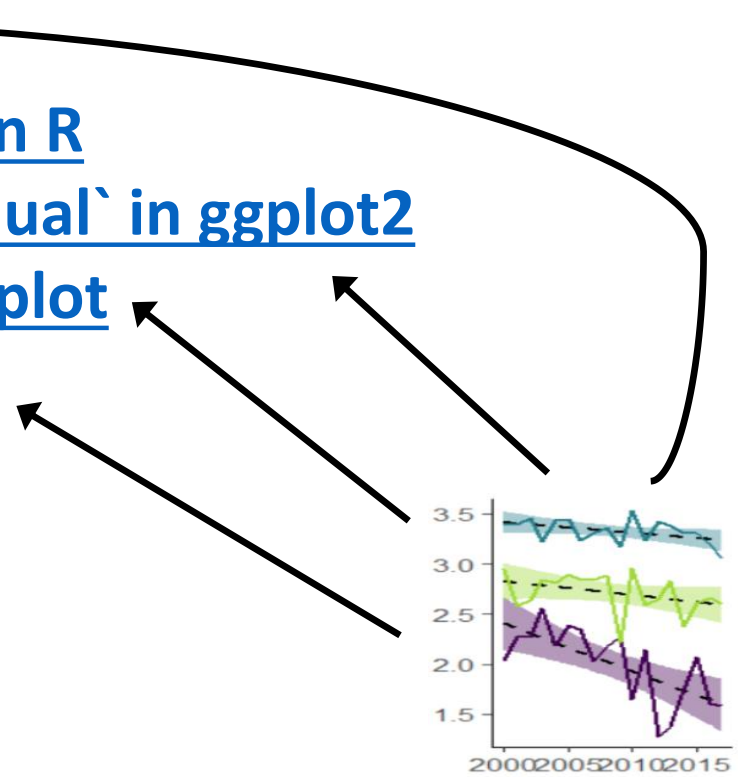
Ries Lab Meeting January 2022, Elise Larsen (eal109@Georgetown.edu)

Recent web queries:

stack overflow:

- How to use superscript with ggplot2
- Extracting the last n characters from a string in R
- Manually colouring plots with `scale_fill_manual` in ggplot2
- Construct a manual legend for a complicated plot
- Adding images below x-axis labels in ggplot2

sthda.com



Recent slack notes:

1. Function for “not in”:

   because sometimes I forget it: `%notin%` = function(x,y) !(x %in% y)

1. Dealing with decimal places

   WTF ggplot why would I want contour labels with 12 digits?

   soln: geom_text_contour(aes(label=**scaleFUN(..level..)**)

   also: label_number(accuracy = 0.01)

   label = label_number(suffix = "\u00b0C") for °C

1. Dealing with dates: applying regular argument params to function

   as.mdy<-function(x) {as.Date(x,format="%m/%d/%Y") }

   mutate_at(vars(end_with("Date")), as.mdy))

Resources from Rstudio.com

https://www.rstudio.com/resources/cheatsheets/

https://rmarkdown.rstudio.com/lesson-1.html

https://www.rstudio.com/resources/webinars/

https://www.rstudio.com/resources/rstudioglobal-2021/

Webinars & Videos

Data Science Hangout

Cheatsheets

Books

Education

Certified Partners

In-Person Workshops

RStudio Documentation

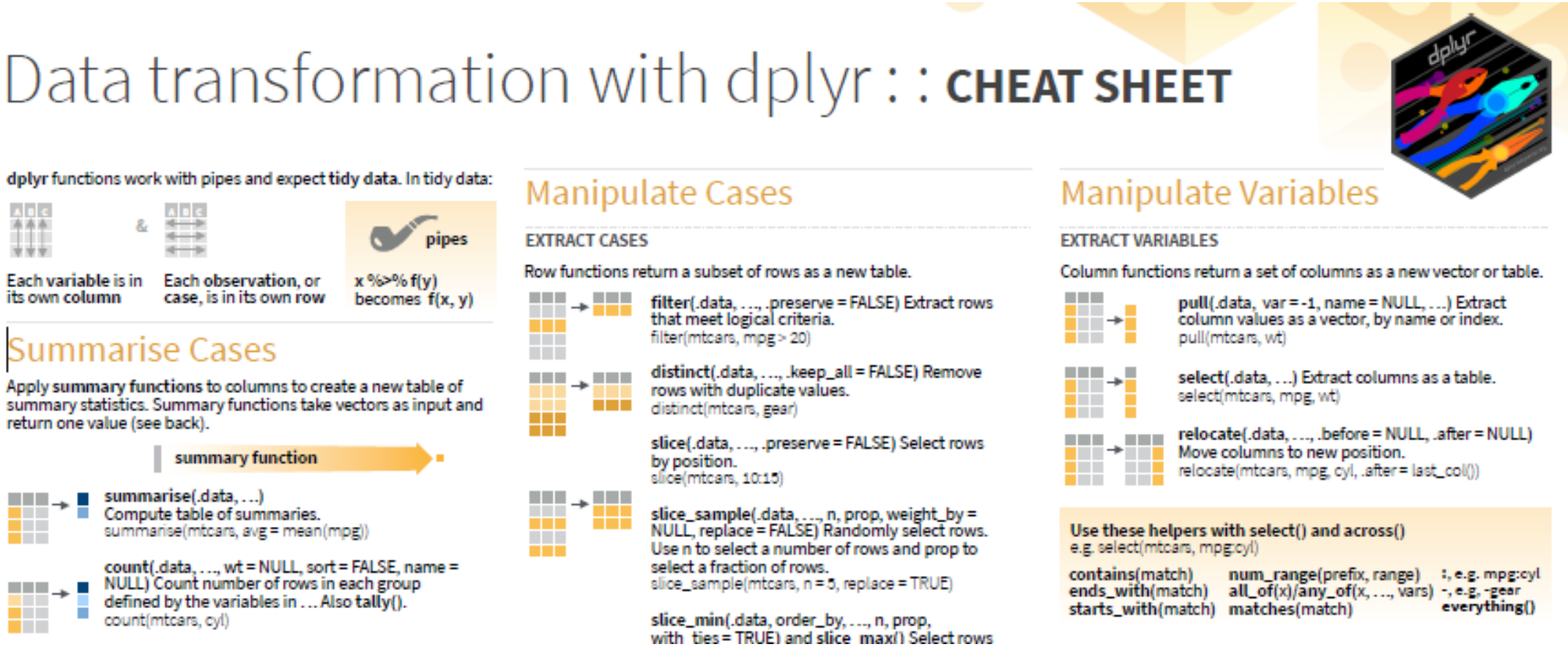
RStudio Blog

R Views Blog

AI Blog

Tidyverse Blog

cheatsheets for R



Data transformation with dplyr : : CHEAT SHEET

dplyr functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

pipes

x %>% f(y) becomes f(x, y)

Summarise Cases

Apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function

summarise(.data, ...)
Compute table of summaries.
summarise(mtcars, avg = mean(mpg))

count(.data, ..., wt = NULL, sort = FALSE, name = NULL) Count number of rows in each group defined by the variables in ... Also **tally**().
count(mtcars, cyl)

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

filter(.data, ..., .preserve = FALSE) Extract rows that meet logical criteria.
filter(mtcars, mpg > 20)

distinct(.data, ..., .keep_all = FALSE) Remove rows with duplicate values.
distinct(mtcars, gear)

slice(.data, ..., .preserve = FALSE) Select rows by position.
slice(mtcars, 10:15)

slice_sample(.data, ..., n, prop, weight_by = NULL, replace = FALSE) Randomly select rows. Use n to select a number of rows and prop to select a fraction of rows.
slice_sample(mtcars, n = 5, replace = TRUE)

slice_min(.data, order_by, ..., n, prop, with_ties = TRUE) and **slice_max**() Select rows

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1, name = NULL, ...) Extract column values as a vector, by name or index.
pull(mtcars, wt)

select(.data, ...) Extract columns as a table.
select(mtcars, mpg, wt)

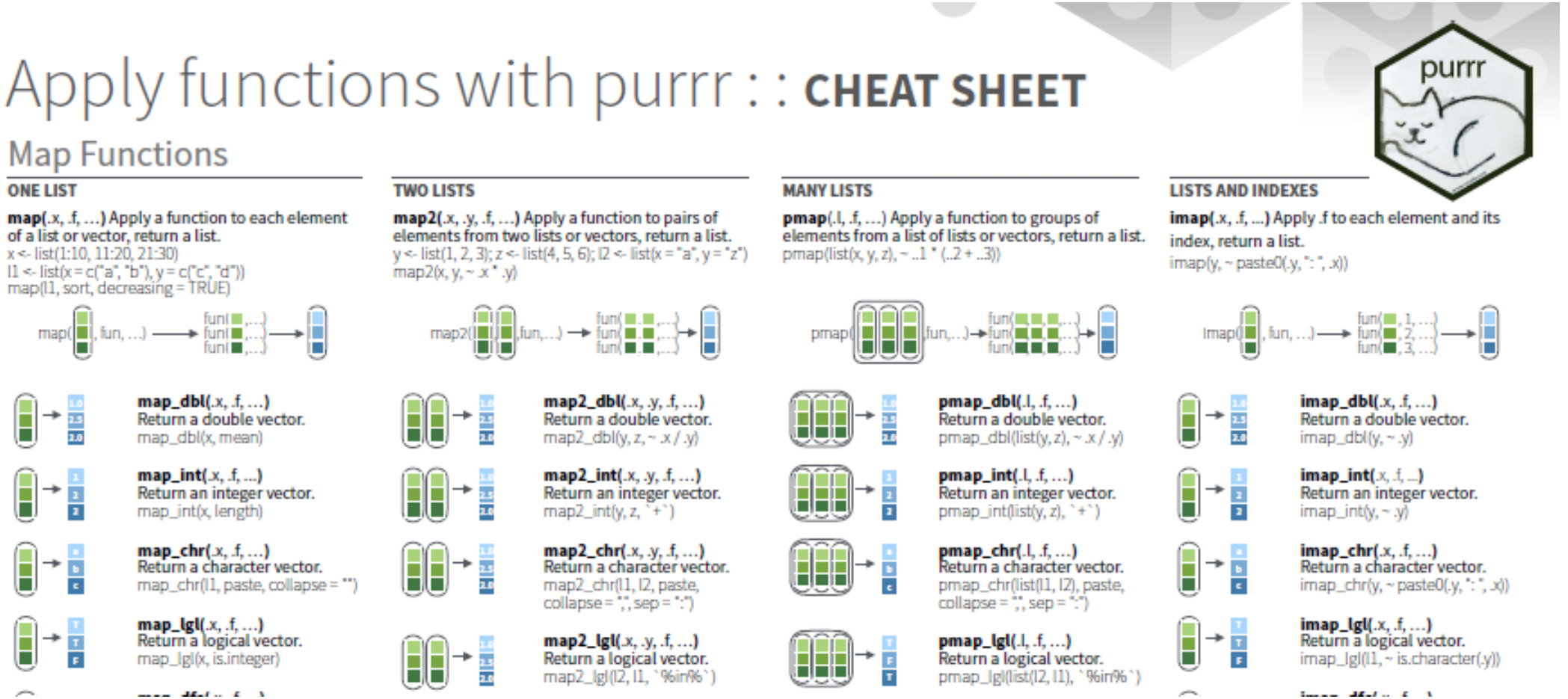
relocate(.data, ..., .before = NULL, .after = NULL) Move columns to new position.
relocate(mtcars, mpg, cyl, .after = last_col())

Use these helpers with select() and across()
e.g. select(mtcars, mpg:cyl)

contains(match)
ends_with(match)
starts_with(match)
num_range(prefix, range)
all_of(x)/**any_of**(x, ..., vars)
matches(match)
:, e.g. mpg:cyl
-, e.g. -gear
everything()

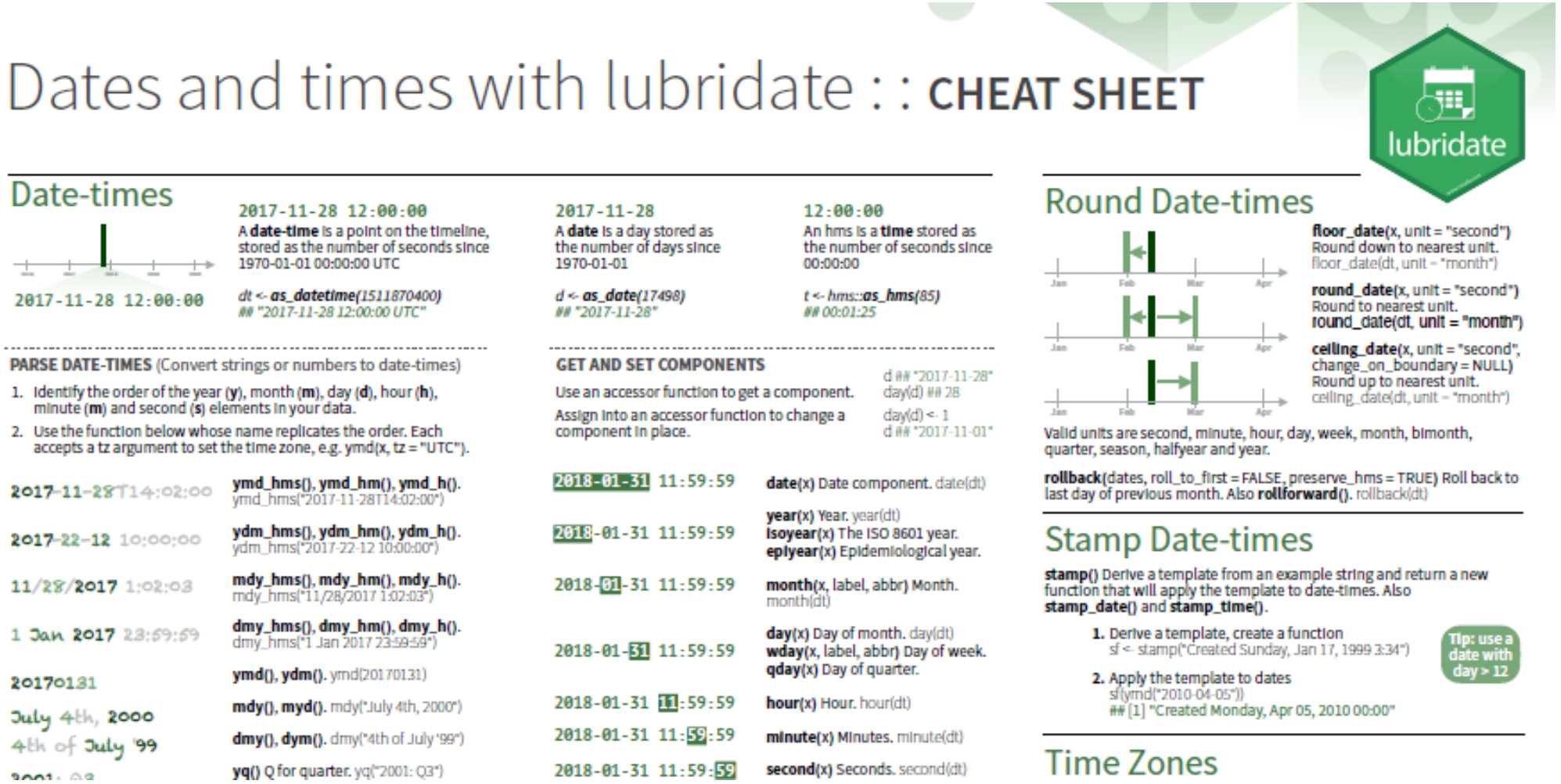
cheatsheets for R

https://www.rstudio.com/resources/cheatsheets/



cheatsheets for R

https://www.rstudio.com/resources/cheatsheets/



Dates and times with lubridate : : CHEAT SHEET

lubridate

Date-times

2017-11-28 12:00:00

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 tz argument to set the time zone, e.g. ymd(x, tz = "UTC").

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")

yq() Q for quarter. yq("2001: Q3")

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)

Round Date-times

floor_date(x, unit = "second") Round down to nearest unit. floor_date(dt, unit = "month")

round_date(x, unit = "second") Round to nearest unit. round_date(dt, unit = "month")

ceiling_date(x, unit = "second", change_on_boundary = NULL) Round up to nearest unit. ceiling_date(dt, unit = "month")

Valid units are second, minute, hour, day, week, month, bimonth, quarter, season, halfyear and year.

rollback(dates, roll_to_first = FALSE, preserve_hms = TRUE) Roll back to last day of previous month. Also **rollforward()**. rollback(dt)

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()**.

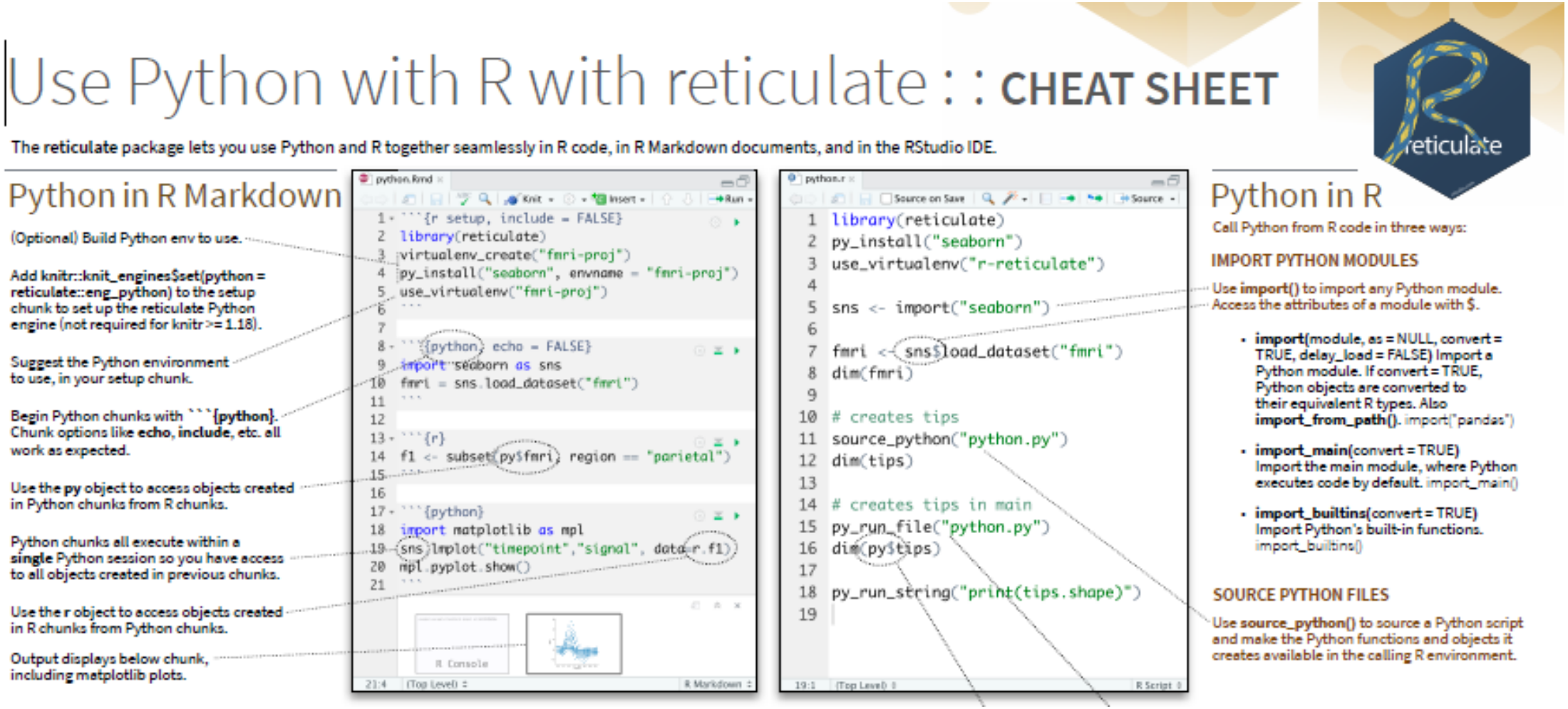
1. Derive a template, create a function
   sf <- stamp("Created Sunday, Jan 17, 1999 3:34")
2. Apply the template to dates
   sf(ymd("2010-04-05"))
   ## [1] "Created Monday, Apr 05, 2010 00:00"

Tip: use a date with day > 12

Time Zones

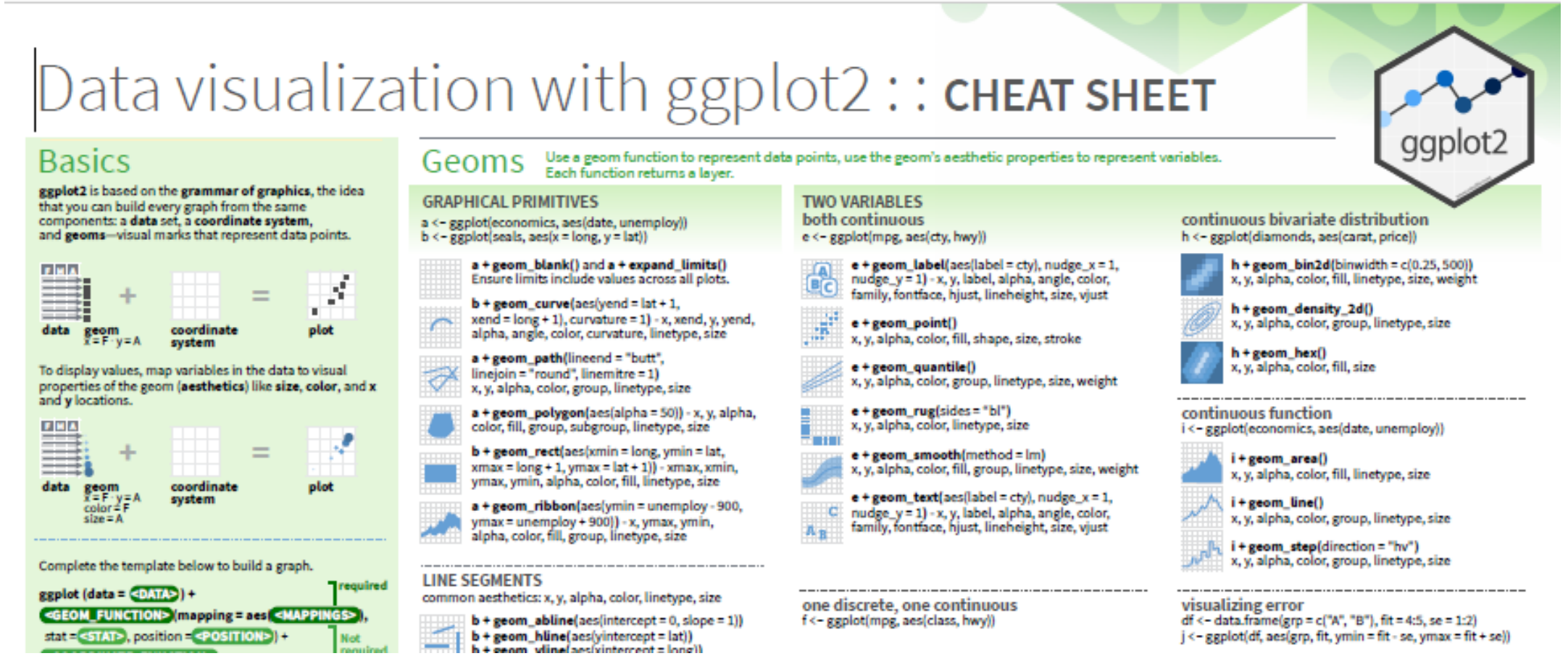
cheatsheets for R

https://www.rstudio.com/resources/cheatsheets/



cheatsheets for R

https://www.rstudio.com/resources/cheatsheets/



Data visualization with ggplot2 : : CHEAT SHEET

ggplot2

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and **geoms**—visual marks that represent data points.

data + geom (x = F, y = A) = coordinate system + plot

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

data + geom (x = F, y = A, color = F, size = A) = coordinate system + plot

Complete the template below to build a graph.

```
ggplot(data = <DATA>) +                                   required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
stat = <STAT>, position = <POSITION>) +                   Not required
```

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

a + geom_blank() and **a + expand_limits()**
Ensure limits include values across all plots.

b + geom_curve(aes(yend = lat + 1, xend = long + 1), curvature = 1) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend = "butt", linejoin = "round", linemitre = 1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(alpha = 50)) - x, y, alpha, color, fill, group, subgroup, linetype, size

b + geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin = unemploy - 900, ymax = unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept = 0, slope = 1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

TWO VARIABLES

both continuous

```
e <- ggplot(mpg, aes(cty, hwy))
```

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_point()
x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile()
x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl")
x, y, alpha, color, linetype, size

e + geom_smooth(method = lm)
x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1) - x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

one discrete, one continuous

```
f <- ggplot(mpg, aes(class, hwy))
```

continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))
```

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density_2d()
x, y, alpha, color, group, linetype, size

h + geom_hex()
x, y, alpha, color, fill, size

continuous function

```
i <- ggplot(economics, aes(date, unemploy))
```

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

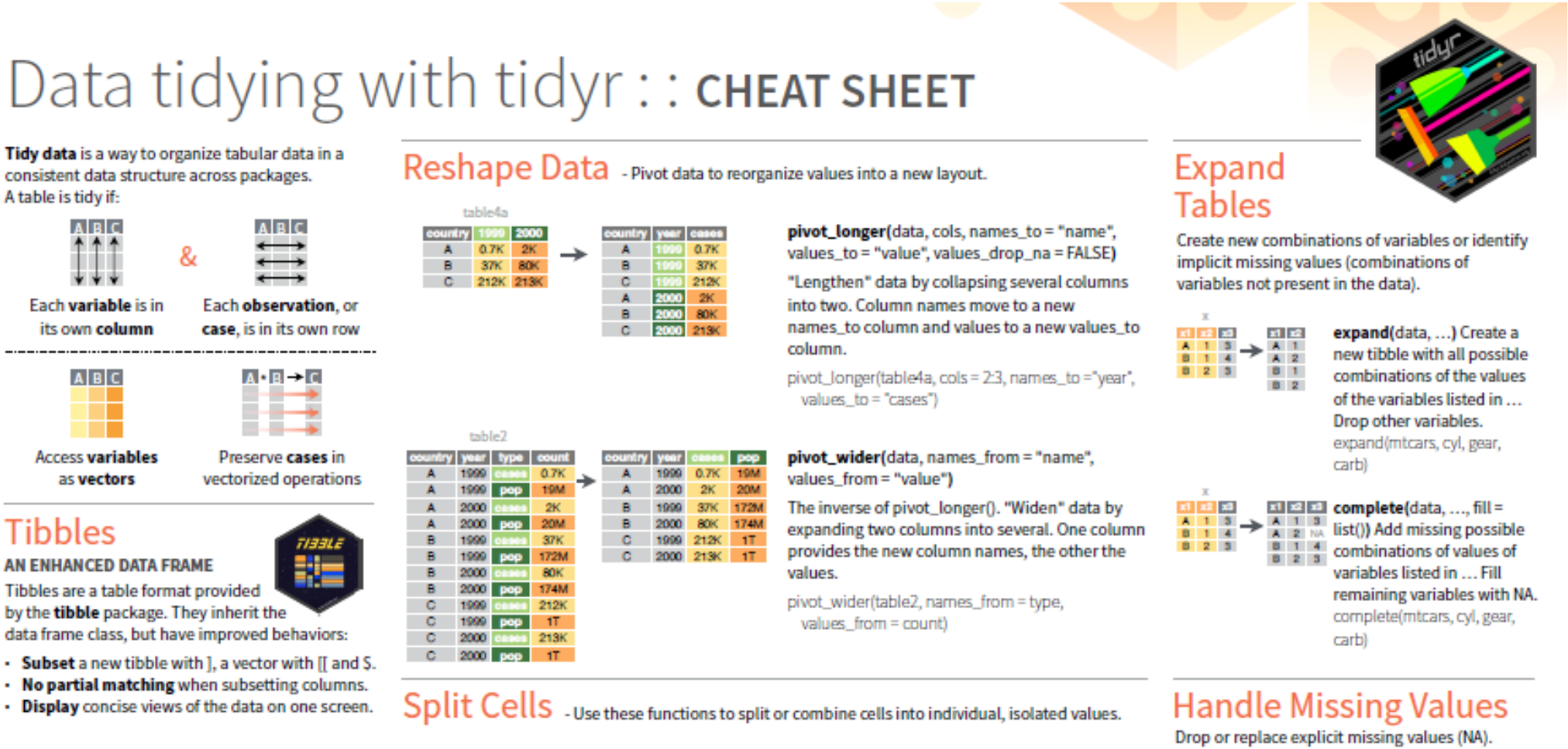
i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

```
df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))
```

cheatsheets for R

https://www.rstudio.com/resources/cheatsheets/



Contributed Cheatsheets

These cheatsheets have been generously contributed by R users.

Advanced R

Environments, data structures, functions, subsetting and more, by Arianne Colton and Sean Chen. Updated February 2016.

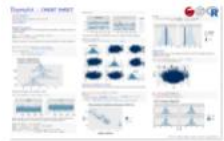
DOWNLOAD

Base R

Vectors, matrices, lists, data frames, functions and more in base R, by Mhairi McNeill. Updated March 2015.

DOWNLOAD

bayesplot



Plotting for Bayesian Models in R, by Edward A. Roualdes. Updated May 2020.

DOWNLOAD

BCEA

Bayesian cost effective analysis in R, by Gianluca Baio. Updated February 2021.

DOWNLOAD

caret

Modeling and machine learning in R with the caret package, by Max Kuhn. Updated September 2017.

DOWNLOAD

cartography

Thematic maps with spatial objects, by Timothée Giraud. Updated July 2018.

DOWNLOAD

collapse

data.table

DeclareDesign

jfa

Work with bayesian and classical statistical audit samples, by Koen Derks. Updated September 2021.

DOWNLOAD

labelled

Manipulate labelled data, by Joseph Larmarange. Updated June 2020.

DOWNLOAD

LaTeX

A reference to the LaTeX typesetting language, useful in combination with knitr and R Markdown, by Winston Chang. Updated January 2018.

DOWNLOAD

leaflet

Interactive maps in R with leaflet, by Kejia Shi. Updated May 2017.

DOWNLOAD

Machine Learning Modelling

A tabular guide to machine learning algorithms in R, by Arnaud Amsellem. Updated March 2018.

DOWNLOAD

mlr

The mlr package offers a unified interface to R's machine learning capabilities, by Aaron Cooley. Updated February 2018.

DOWNLOAD

mosaic

The mosaic package is for teaching mathematics, statistics, computation and

nardl

The nardl package estimates the nonlinear cointegrating autoregressive distributed

nimble

Hierarchical statistical models that extend BUGS and JAGS, by the Nimble

Visualizations with ggplot2 (K. Woo, accidental aRt)

- Data first grammar of graphics

- Common problems: mapping mishaps, scale snafus, theme threats
    -theme customization always defaults to most specific input

- ggplot2-book.org

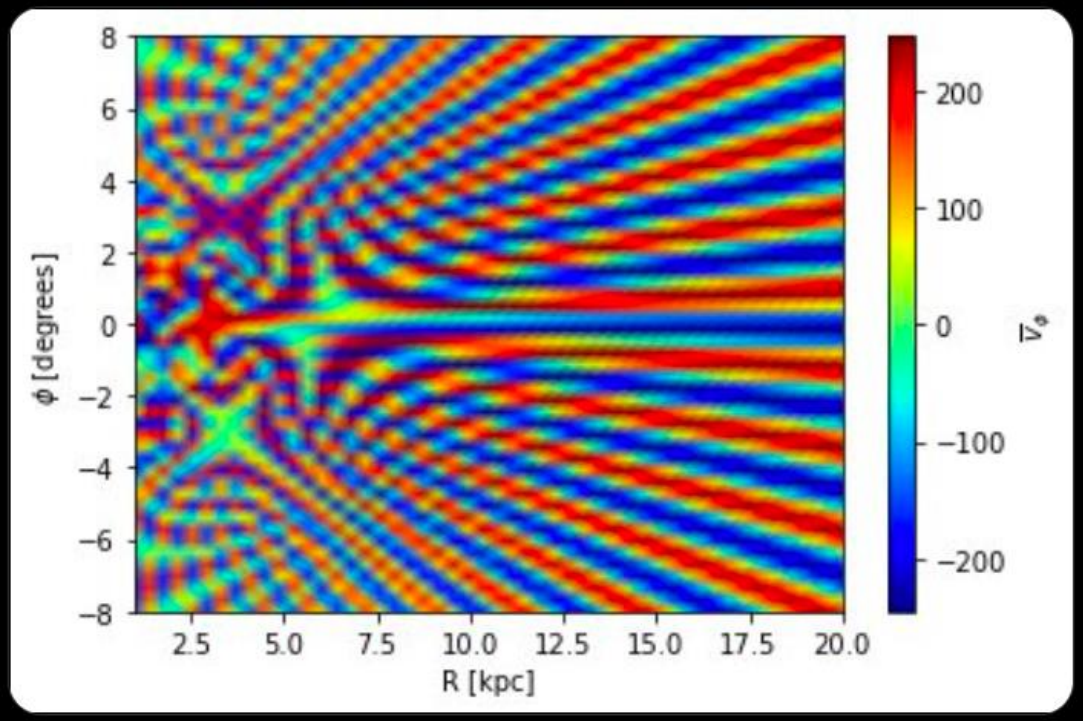
https://www.rstudio.com/resources/rstudioglobal-2021/always-look-on-the-bright-side-of-plots/

@accidental_aRt

accidental aRt Retweeted

Pau Ramos @PBR117 · Jun 15, 2021

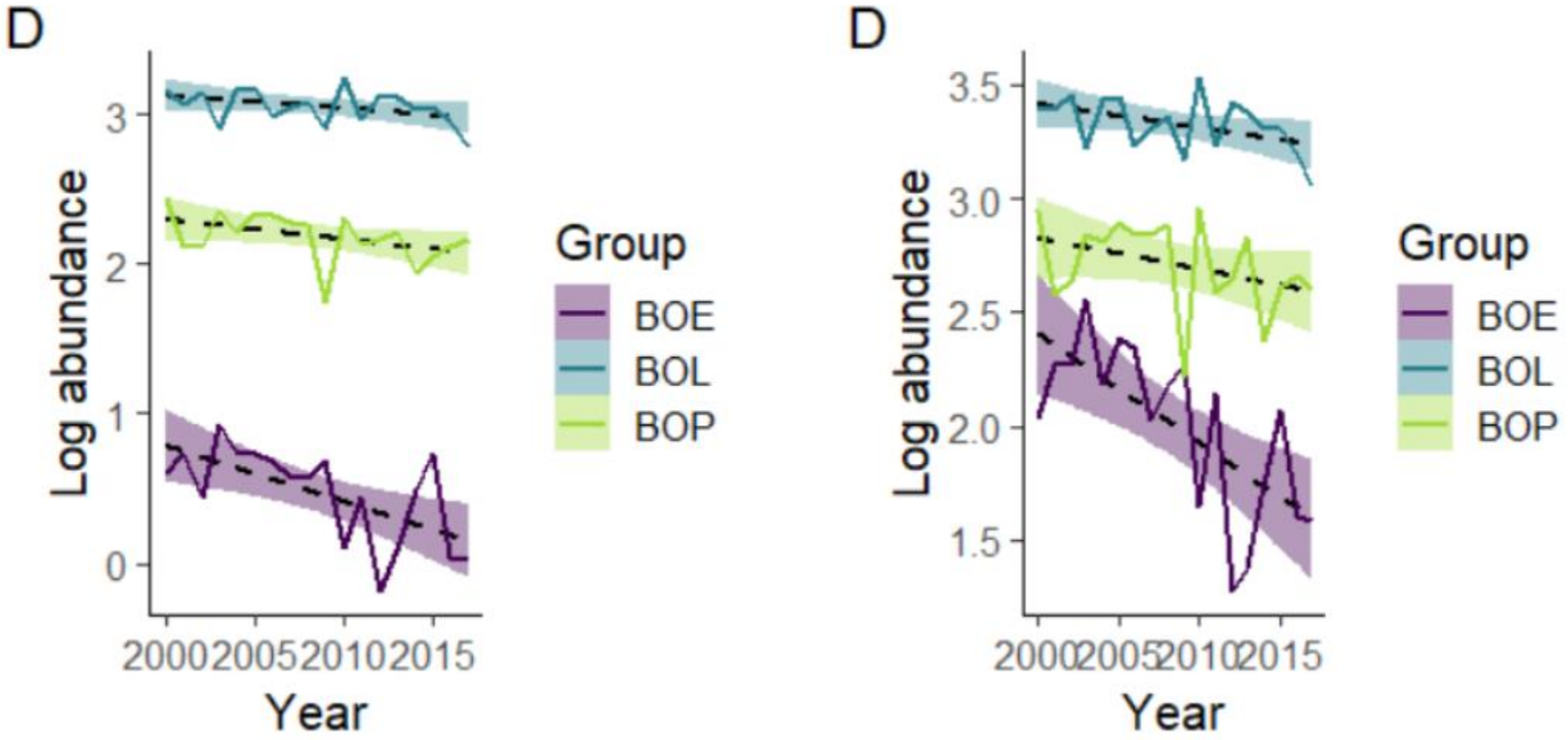
Everything in this plot is wrong (literally everything) and yet...There is some beauty in it! I'd say it qualifies as @accidental_aRt



Kanishka Misra @kanishkamisra · Jan 25, 2021

"Definitely Accidental" art @accidental_aRt

Same underlying data … different order of operations



Dealing with code maintenance in the light of version changes (H. Wickham)

- Reframing frustrations with broken code & documentation
- Using best applicable functions (avoiding off-label uses)
- Tools to isolate environment to specific package versions

https://www.rstudio.com/resources/rstudioglobal-2021/maintaining-the-house-the-tidyverse-built/

Tools to isolate environment to specific package versions

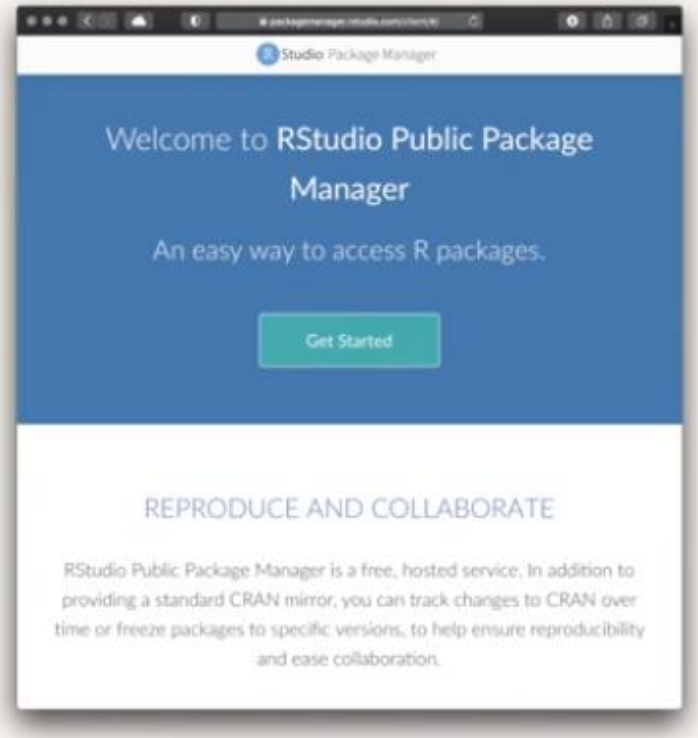
One option is to use renv


```
# Isolate this project
renv::init()

# Capture any recent package changes
renv::snapshot()

# Elsewhere, restore exact environment
renv::restore()
```

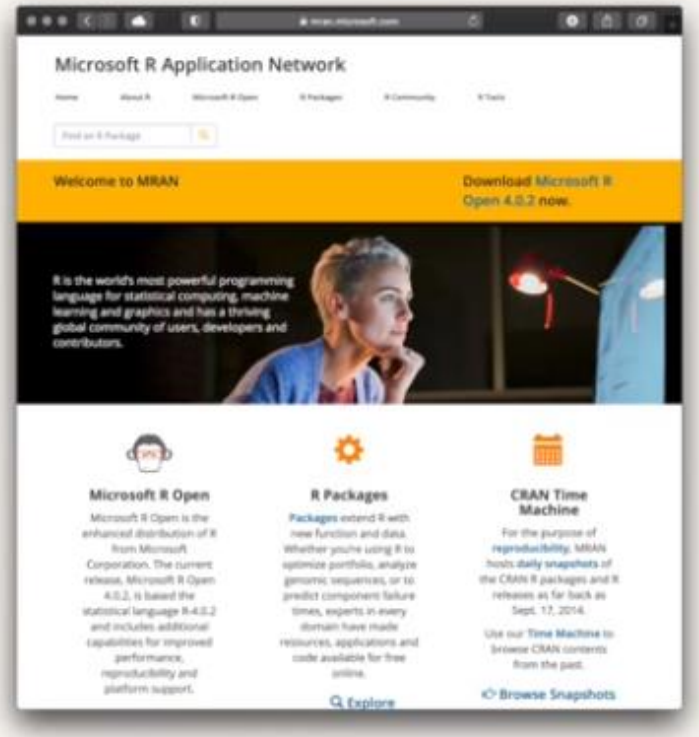
https://rstudio.github.io/renv/index.html

Tools to isolate environment to specific package versions

Another option is to use a CRAN time capsule



https://packagemanager.rstudio.com



https://mran.microsoft.com

Possible “homework” ideas for next discussion:

Watch 1 video from Rstudio or other instructional site

Report back on main points / key tips

20 minute R plot

- Pick a dataset using data()
- Define a question / hypothesis
- Consider filters, aggregations, transformations
- Create a relevant data visualization

Share our code to explore different approaches