Stat405

Dates and times

Hadley Wickham

Lubridate problems

- Try:
- `Sys.setlocale("LC_ALL", "en")`
- `Sys.setenv("LANGUAGE" = "en")`

1. Date-times
2. Time spans
3. Math with date-times

Homeworks

- Should be your own work
- You can work together on them, but you should not have identical code (particularly if it's wrong!)
- Substantially similar work will not be graded. Future penalties will be more severe.

Next week

- I'll be away
- Barrett will be teaching tables and tidy data

Date-times in R

Riddle

When does:

one year != 365 days?

one day != 24 hours?

one minute != 60 seconds?

Time is a deceptively complicated system to work with. To master it, we are going to learn about 4 new objects:

1. instants
2. intervals
3. periods
4. durations

Instant

An instant refers to a specific moment of time.

Instants are combination of years, months, days, hours, minutes, seconds, and time zone.

e.g, 2011-03-28 08:44:32 CST

```
# lubridate turns strings into instants with
# functions that have y, m, and d in their
# names. Use the function whose name matches
# the order of the elements in the date

ymd("2011-03-28")
mdy("03/28/2011")
dmy("28032011")
ymd_hms("2011:03:28 08:50:30")

#order matters
dmy("05-03-2011")
mdy("05-03-2011")
```

```
library(ggplot2)
library(lubridate)

emails <- read.csv("email.csv",
  stringsAsFactors = FALSE)
# This is all the emails I've sent 2011-10-18
# to 2012-03-12 along, with an estimate of how many
# words I wrote

str(emails)
# Note that R does not automatically parse dates
```

Your turn

Parse the column of dates in `emails.csv` into date time objects.

```
emails$time <- ymd_hms(emails$time)
emails$time <- ymd_hms(emails$time, tz = "")
```

Manipulating instants

```
# compare
now() > ymd("2011-03-29")
today() > ymd("2011-03-29")

# round
floor_date(now(), "month")
ceiling_date(now(), "month")
round_date(now(), "month")

# extract
year(now())
```

Date component	Accessor
Year	`year()`
Month	`month()`
Week	`week()`
Day of year	`yday()`
Day of month	`mday()`
Day of week	`wday()`
Hour	`hour()`
Minute	`minute()`
Second	`second()`
Time Zone	`tz()`

```
now()
year(now())
hour(now())

day(now())
yday(now())
wday(now())
wday(now(), label = TRUE)
wday(now(), label = TRUE, abbr = FALSE)

month(now(), label = TRUE, abbr = FALSE)
```

Which day of the week were you born?

```
now()
year(now())
hour(now())

day(now())
yday(now())
wday(now())
wday(now(), label = TRUE)
wday(now(), label = TRUE, abbr = FALSE)

month(now(), label = TRUE, abbr = FALSE)
```

Your turn

At which hour of the day am I most likely to respond to your email?

```
# Can't really answer that, because don't know
# how many emails I'm receiving, but can at least
# figure out when most emails are sent:

qplot(hour(time), data = emails, binwidth = 1)

# OR
library(plyr)
emails$hour <- hour(emails$time)
hours <- count(emails, "hour")
qplot(hour, freq, data = hours, geom = "line")

emails$wday <- wday(emails$time, label = TRUE)
hours <- count(emails, c("hour", "wday"))
qplot(hour, freq, data = hours, geom = "line",
  colour = wday)
```

```
# Accessor functions can also be used to change the
# elements of a date-time

time <- now()
year(time) <- 1999
hour(time) <- 23

day(time) <- 45

tz(time) <- "UTC"
```

```
# Accessor functions can also be used to change the
# elements of a date-time

time <- now()
year(time) <- 1999
hour(time) <- 23

day(time) <- 45

tz(time) <- "UTC"
```

What day of the week your birthday will be on in 2012?

Rounding instants

We can round an instant to a specified unit using floor_date(), ceiling_date(), round_date()

e.g.

```
round_date(now(), "day")
round_date(now(), "month")
```

Your turn

Use `ddply` or `count` to calculate how many emails I've sent per day since Jan 1. Plot the number of emails per day over time.

Compare the number of emails sent by day of the week.

```
# just the recent emails
recent <- subset(emails, time >= ymd("2012-01-01"))

# binning into days
recent$day <- floor_date(recent$time, "day")

# calculating daily totals
daily <- ddply(recent, "day", summarise,
  words = sum(words), emails = length(day))
qplot(day, emails, data = daily, geom = "line")
qplot(wday(day, label = T), emails, data = daily,
  geom = "boxplot")
```

Time spans

Consider

Suppose we want to use R to set our alarm

```
ymd_hms("2011-01-01 08:30:00", tz = "") +
ddays(0:30)
```

how about

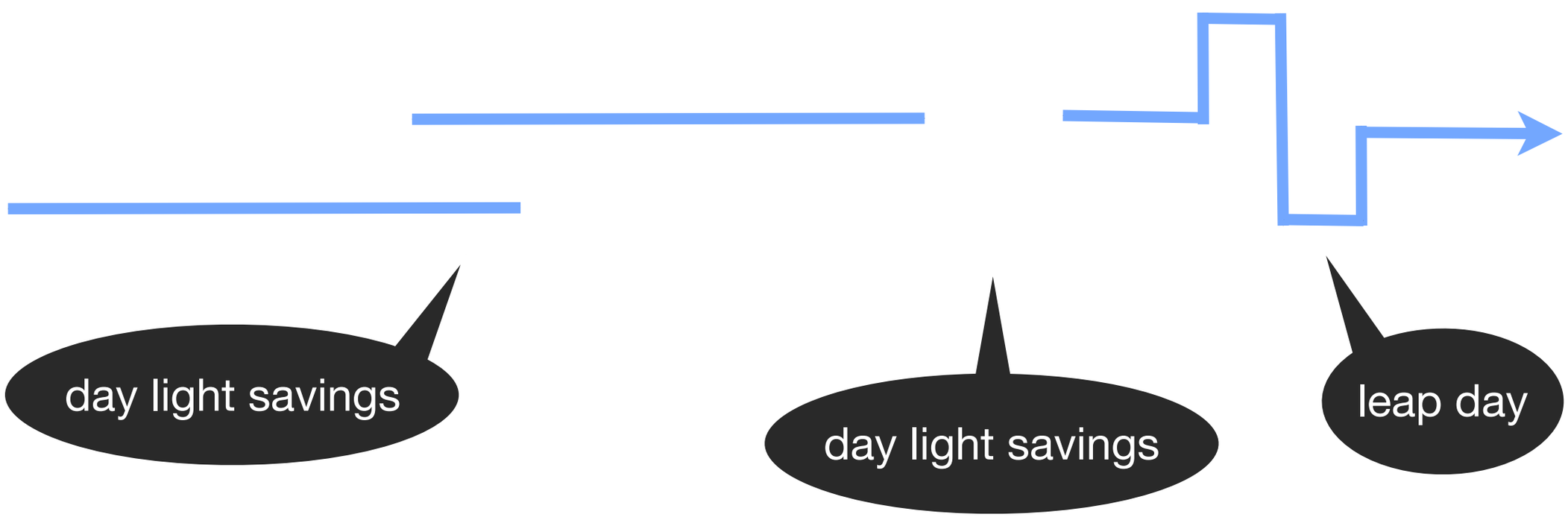
```
ymd_hms("2011-03-01 08:30:00", tz = "") +
ddays(0:30)
```

What went wrong?

The number line

The number line

The time line (according to clock times)



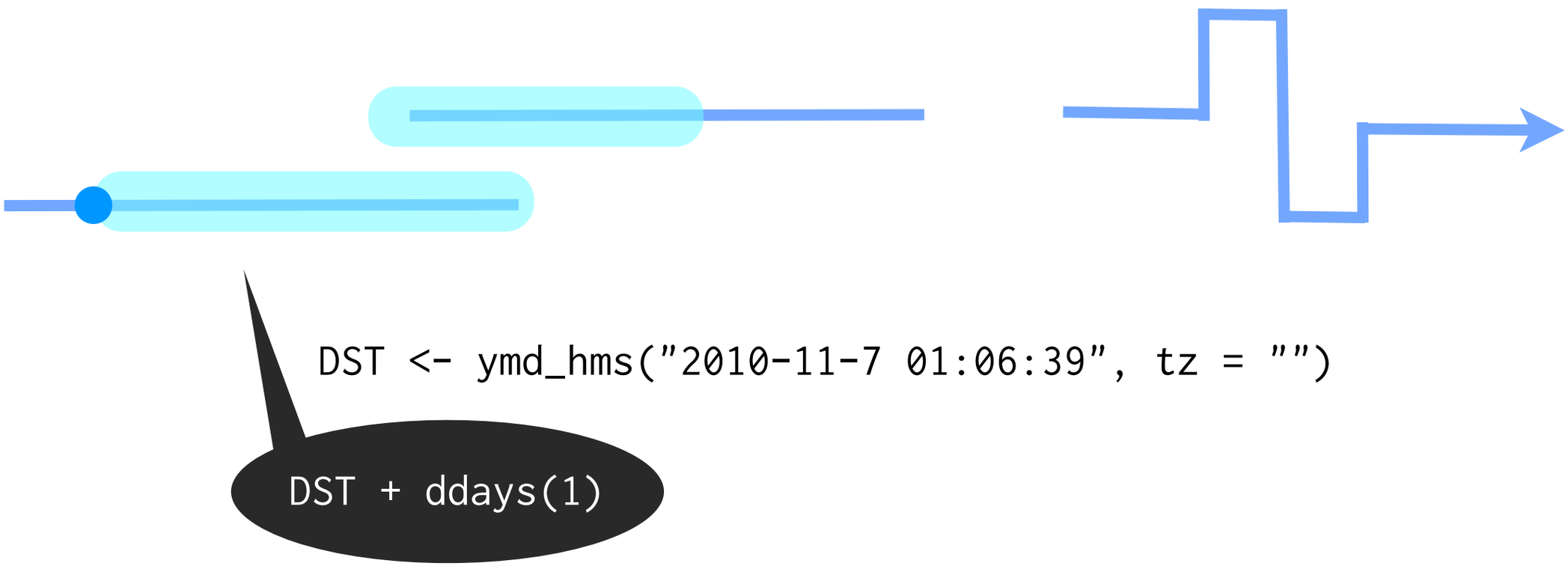
3 types of time spans

We can still do math on the time line, as long as we're specific about how we want to measure time.

1. durations (exact times)
2. periods (clock times)
3. intervals (specific times)

durations

Durations measure the exact amount of seconds that pass between two time points.

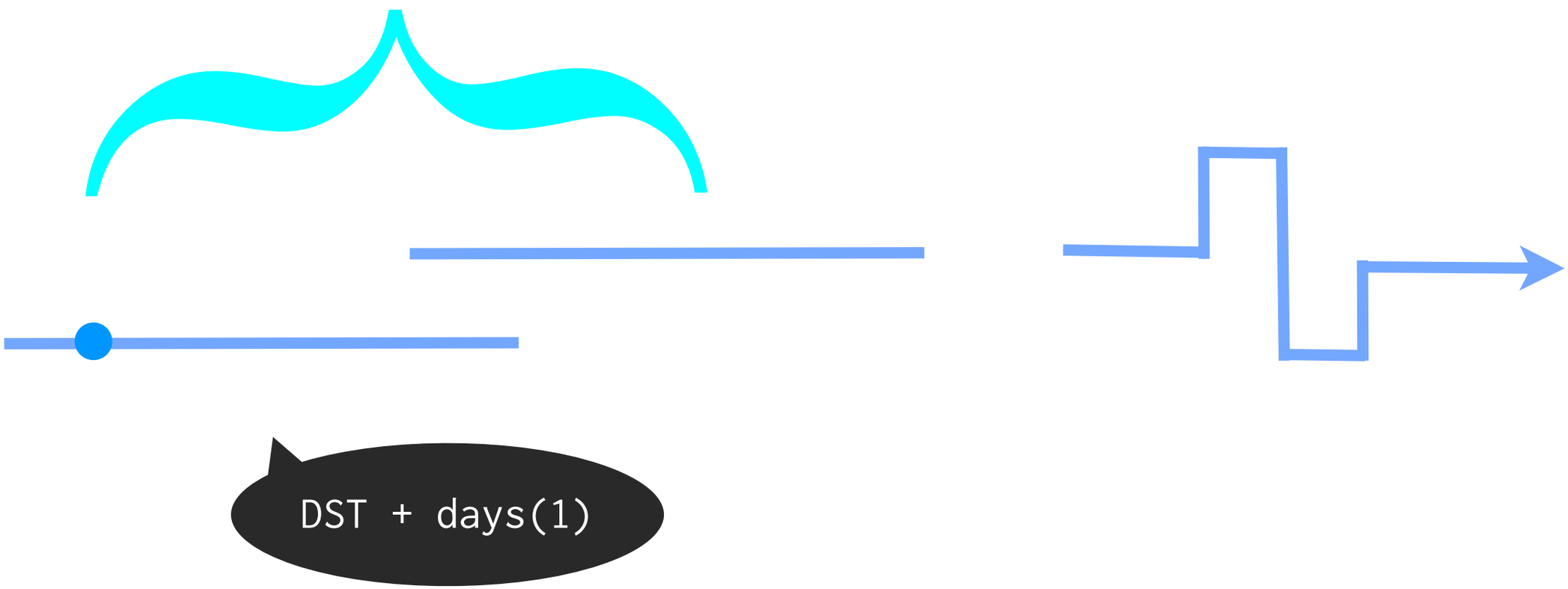


```
# Use d + the plural of of a unit of time to create
# a duration.

dminutes(5)
dhours(278)
dmonths(4) # why doesn't this work?
```

periods

Periods measure time spans in units larger than seconds. Periods pay no attention to how many sub-units occur during the unit of measurement.



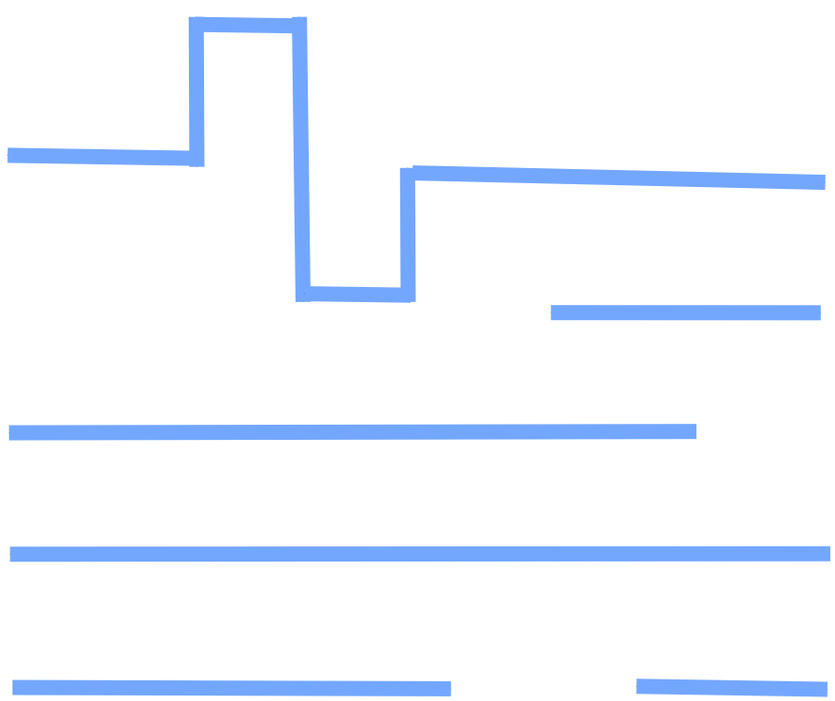
```
# Use the plural of a time unit to create a period.

minutes(5)
hours(278)
months(4) # months are not a problem
years(1:10)
```

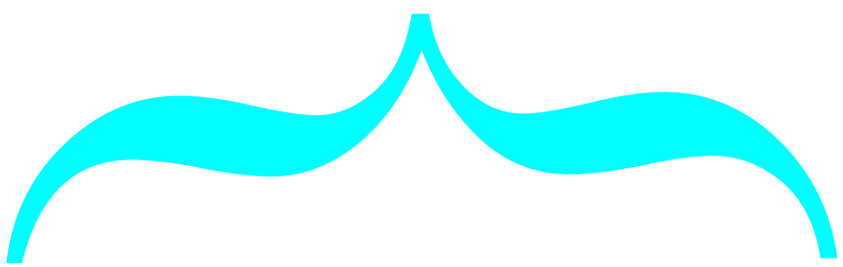
Why use periods?

No surprises.

`2010-11-01 00:00:00 + months(1)` will always equal 2010-12-01 00:00:00 no matter how many leap seconds, leap days or changes in DST occur in between

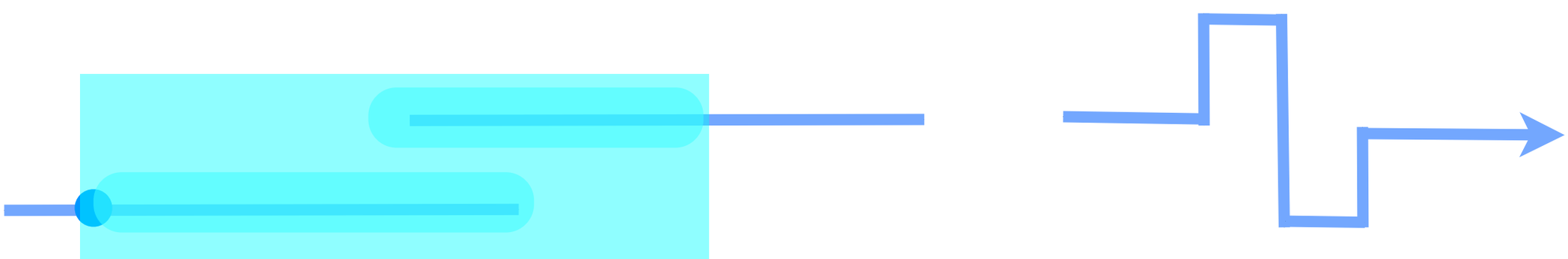


=



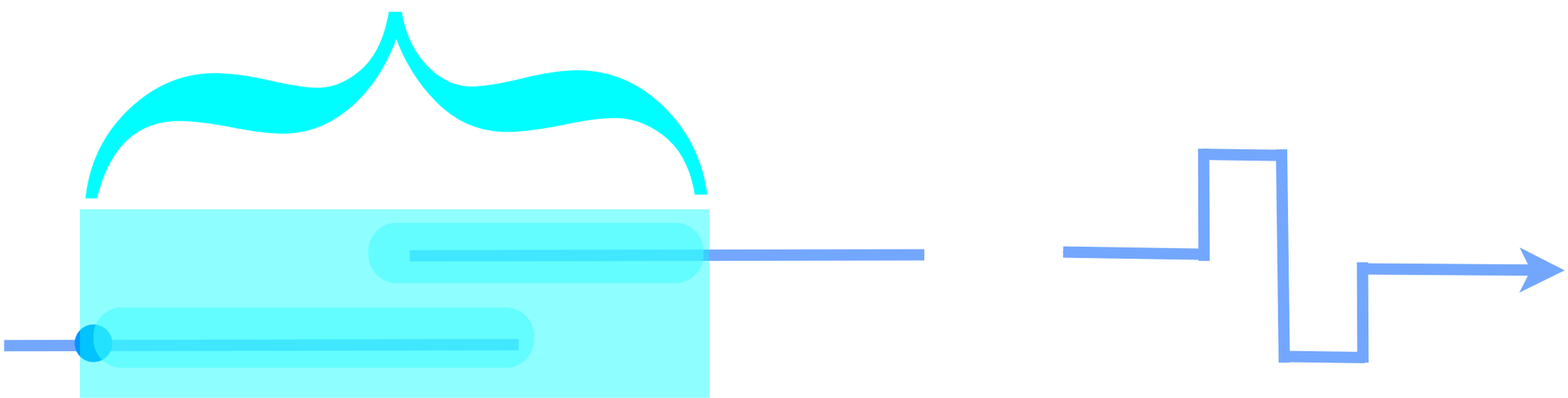
intervals

Intervals measure a time span by recording its endpoints. Since we know when the time span occurs, we can calculate the lengths of all the units involved.



intervals

Intervals measure a time span by recording its endpoints. Since we know when the time span occurs, we can calculate the lengths of all the units involved.



```
# To create an interval, use the special %--%
# operator:
int <- ymd("2010-01-01") %--% ymd("2009-01-01")

# Access and set the endpoints with int_start() and
# int_end().
int_start(int)
int_end(int) <- ymd("2010-03-14")

# Intervals can be accurately converted to either
# periods or durations with as.period() and
# as.duration()
```

Math with date-times

```
birthday <- ymd("1979-10-14")
life <- birthday %--% now()
life / ddays(1)

life / days(1)
life %/% days(1)
```

Your turn

Calculate your age in minutes. In hours. In days. How old is someone who was born on June 4th, 1958?

```
dob <- ymd("1979-10-14")
(dob %--% now()) %/% days(1)
(dob %--% now()) %/% months(1)
(dob %--% now()) %/% hours(1)
(dob %--% now()) %/% minutes(1)
```

Challenge

Write a function that takes a date and returns the last day of the month the date occurs in.

```
date <- today()
date - days(day(date) - 1) + months(1) - days(1)

# Note that date arithmetic is not commutative!
date - days(day(date) - 2) + months(1)

# OR

ceiling_date(date, "month") - days(1)
```

More info at:

http://www.jstatsoft.org/v40/i03/