R's weirdnesses are fun & useful

Rich FitzJohn

richfitz

Statistics programs generally include **distributions**

Statistics programs generally include **statistical tests**

Statistics programs generally include **plotting**

Statistics programs don't often include **blog generators**

`cran.r-project.org/package=blogdown`

Statistics programs don't often include **webservers**

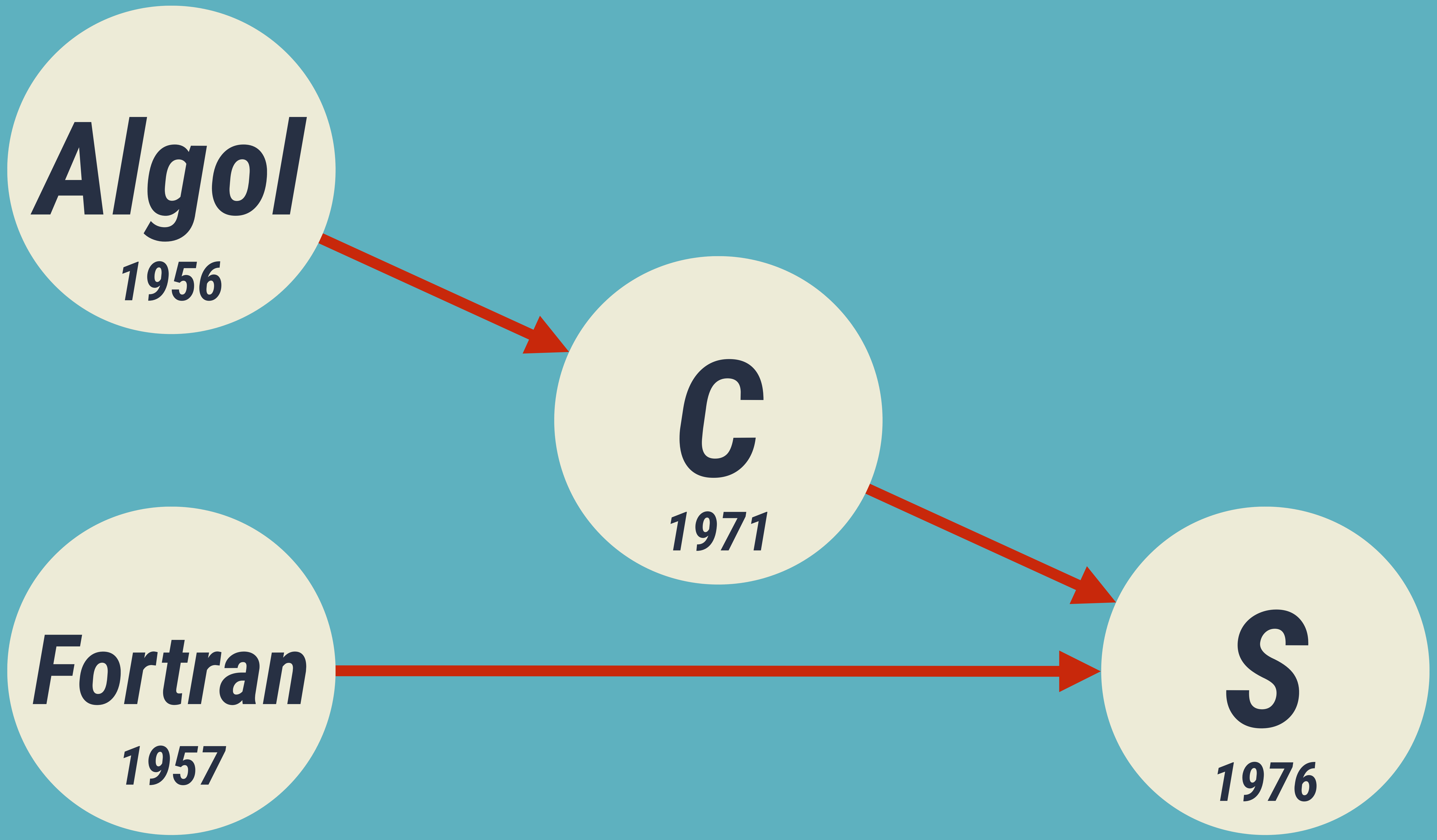
`cran.r-project.org/package=httpuv`

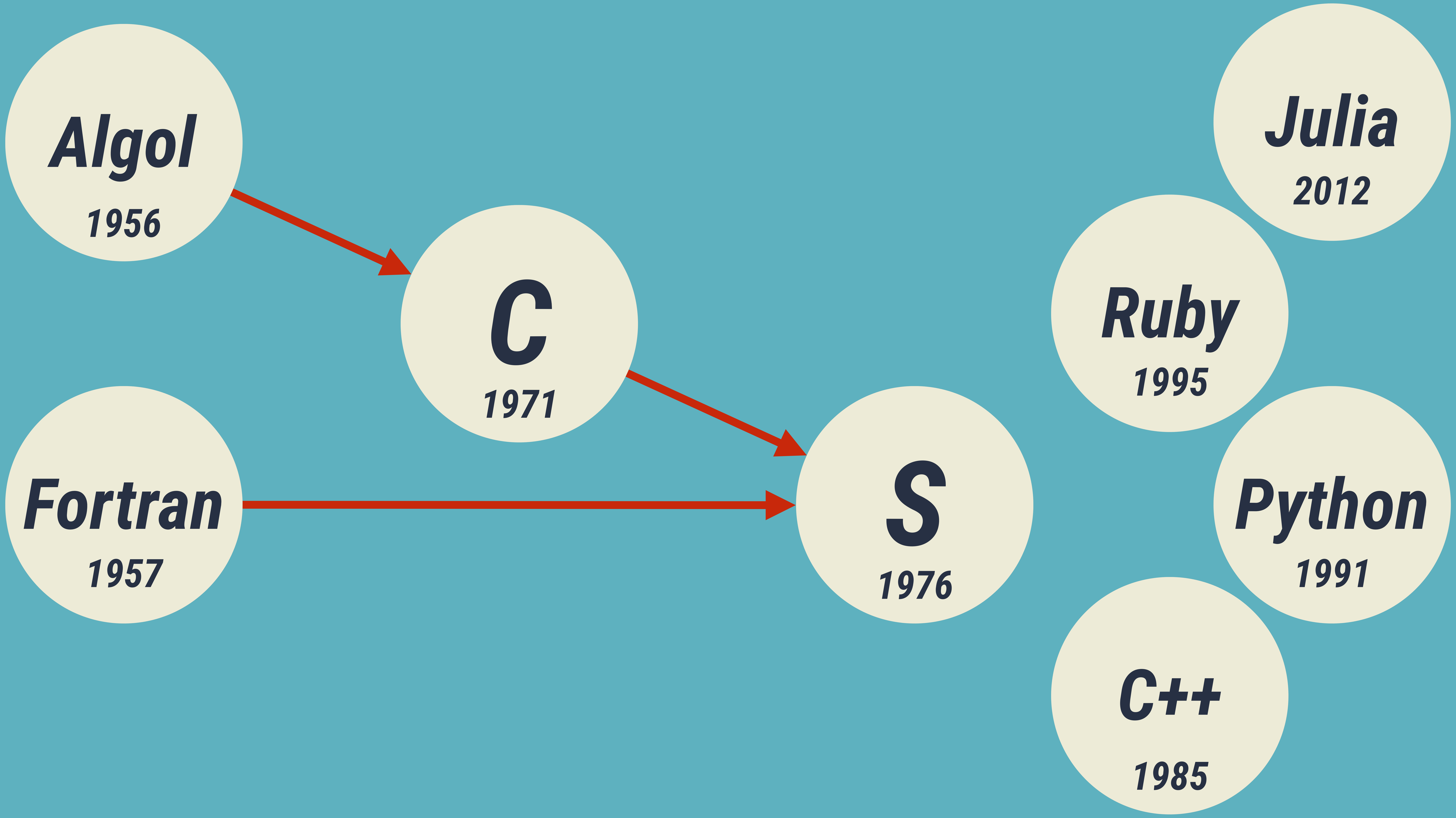
Statistics programs don't often include **minecraft clients**

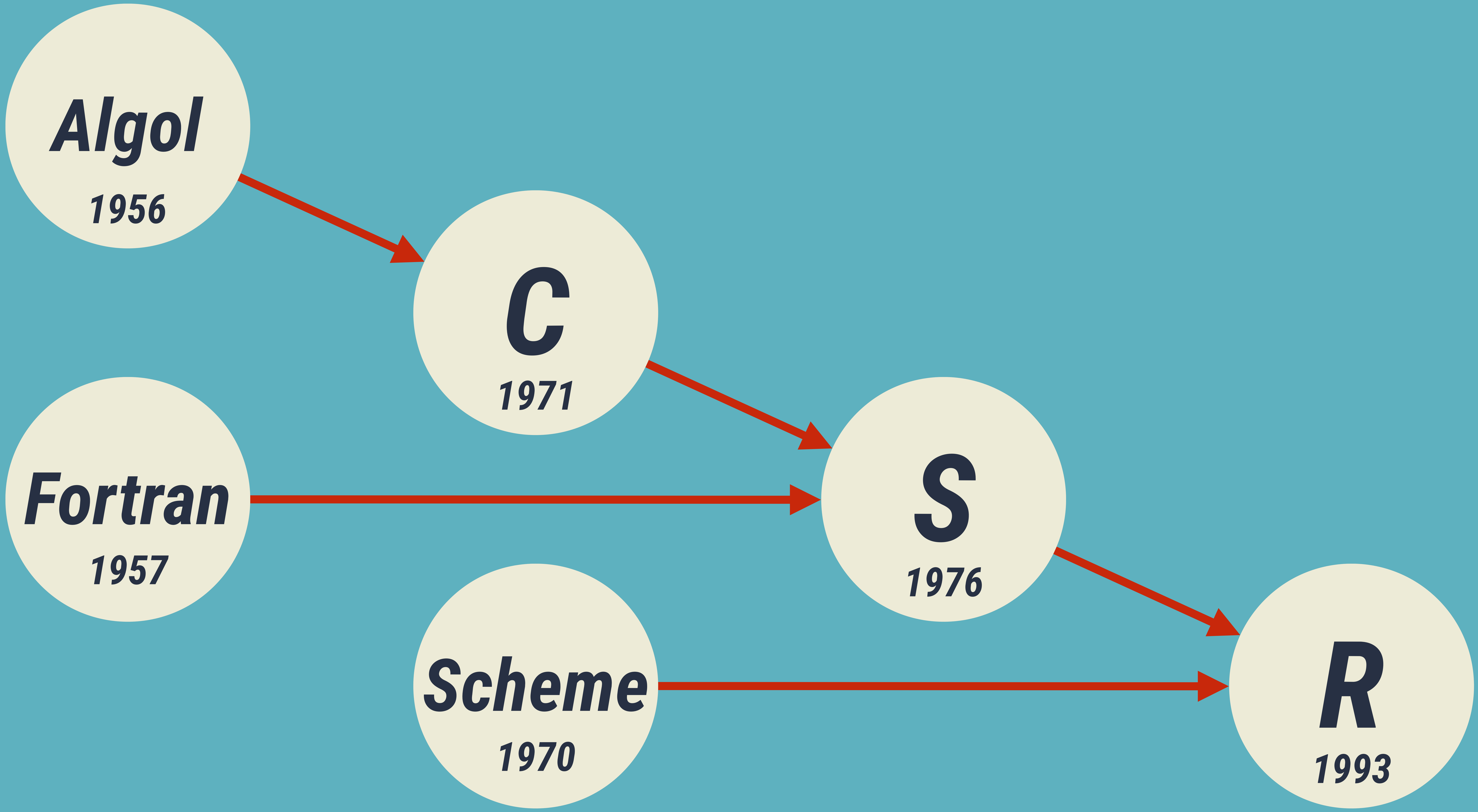
github.com/ropenscilabs/miner

Statistics programs don't often include **metaprogramming**

DATA
YOUR SCIENTISTS
WERE SO
PREOCCUPIED
WITH WHETHER OR
NOT THEY COULD
THEY DIDN’T STOP TO THINK IF THEY SHOULD


Algol
1956
C
1971
Fortran
1957
S
1976


Algol
1956
C
1971
Fortran
1957
S
1976
Julia
2012
Ruby
1995
Python
1991
C++
1985


Algol
1956
C
1971
Fortran
1957
S
1976
Scheme
1970
R
1993

R's weirdnesses are fun & useful

Rich FitzJohn

richfitz

GREAT MARLBOROUGH STREET
ARGYLL PLACE
LITTLE ARGYLL ST
PUMP
WORK HOUSE
PORTLAND STREET
PORTLAND MEWS
BENTINCK STREET
WARDOUR MEWS
ST ANNS COURT
RICHMOND BUILDINGS
RICHMOND MEWS
BREWERY YARD
EDWARD STREET
DUCK LANE
HAM YARD
BROAD STREET
DUFOURS PLACE
MARSHALL STREET
MARSHALL ST.
WEST STREET
TYLER COURT
LOWNDES CT
SOUTH ROW
CARNABY STREET
GREEN DRAGON YARD
LITTLE MARLBOROUGH ST
TYLER STREET
PEMS PLACE
CROSS STREET
CROSS ST.
MARLBOROUGH ROW
FOUBERTS PL.
KING STREET
CAMBRIDGE ST
BREWERY
NEW STREET
HOPKINS STREET
COCK CT.
PULTENEY CT.
HUSBAND ST
WILLIAM AND MARY YARD
TYLERS CT.
BLACK LION CT.
MEARDS STREET
MEARDS COURT
PETER STREET
WALKERS COURT
OLD COMPTON
LITTLE WINDMILL STREET
GREAT PULTENEY STREET
BRIDLE STREET
LITTLE PULTENEY STREET
GREAT CROWN CT.
CT. CROWN CT.
UP. RUPERT ST
GOLDEN SQUARE
SILVER STREET
UP JAMES ST
UP JOHN ST
LOW. JAMES ST
WARWICK STREET
BEAK ST.
CHAPEL PLACE
SAVILLE YARD
REGENT STREET
OLD BURLINGTON MEWS
MILL STREET
MILL ST
NEW BURLINGTON STREET
NEW BURLINGTON MEWS
BOYLE ST.
CONDUIT STREET
MADDOX STREET
POLLEN ST.
ARCHER STREET
PLOUGH YARD
RICHMOND STREET
QUEENS HEAD COURT
SMITHS COURT
SMITHS YARD
RUPERT
PRINCE
DEAN STREET
BERWICK STREET
WARDOUR STREET
CARLISLE
POLAND STREET

Encryption

Differential equations

Docker

Encryption

Encrypt and save csv

```
write.csv(mydata, "secret.csv")
```

Encrypt and save csv

```
tmp <- tempfile()
write.csv(mydata, tmp)
```

Encrypt and save csv

```
tmp <- tempfile()
write.csv(mydata, tmp)
bytes <- readBin(tmp, ...)
enc <- sodium::data_encrypt(bytes, key)
```

Encrypt and save csv

```
tmp <- tempfile()
write.csv(mydata, tmp)
bytes <- readBin(tmp, ...)
enc <- sodium::data_encrypt(bytes, key)
enc
 [1] a7 8e 31 99 3b 7b ac 58 4e 35 37 79
[13] 53 10 4c fe 5e 78 de 4e 4d 25 77 26
```

Encrypt and save csv

```
tmp <- tempfile()
write.csv(mydata, tmp)
bytes <- readBin(tmp, ...)
enc <- sodium::data_encrypt(bytes, key)
writeBin(enc, "secret.csv")
file.remove(tmp)
```

Decrypt and read csv

```
enc <- readBin("secret.csv", ...)
bytes <- sodium::data_decrypt(enc, key)
tmp <- tempfile()
writeBin(bytes, tmp)
mydata <- read.csv(tmp)
file.remove(tmp)
```

A simpler interface

```r
cyphr::encrypt(write.csv(mydata, "secret.csv"), key)



mydata <- cyphr::decrypt(read.csv("secret.csv"), key)
```

A simpler interface

```
cyphr::encrypt(write.csv(mydata, "secret.csv"), key)
# Write mydata to temp file using write.csv
# Encrypt temp file contents to "secret.csv" using key
# Delete temp file
```

A simpler interface

```
cyphr::encrypt(write.csv(mydata, "secret.csv"), key)
# Decide on a temporary file tmp
# Detect filename is second argument "secret.csv"
# Rewrite expression as write.csv(mydata, tmp)
# Evaluate new expression (in same environment as old)
# Read in tmp as bytes
# Encrypt the contents with cyphr::encrypt(bytes, key)
# Save encrypted data as secret.csv
# Delete the temporary file tmp
```

Expressions are data

```
as.list(quote(saveRDS(mydata, "secret.rds")))
[[1]]
saveRDS

[[2]]
mydata

[[3]]
[1] "secret.rds"
```

A simpler interface

```
cyphr::encrypt(write.csv(mydata, "secret.csv"), key)
# Write mydata to temp file using write.csv
# Encrypt temp file to "secret.csv" using key
# Delete temp file

mydata <- cyphr::decrypt(read.csv("secret.csv"), key)
# Decrypt "secret.csv" into temp file using key
# Read mydata from temp file using read.csv
# Delete temp file
```

A simpler interface

```
cyphr::encrypt(saveRDS(mydata, "secret.rds"), key)
# Write mydata to temp file using saveRDS
# Encrypt temp file to "secret.rds" using key
# Delete temp file

mydata <- cyphr::decrypt(readRDS("secret.rds"), key)
# Decrypt "secret.rds" into temp file using key
# Read mydata from temp file using readRDS
# Delete temp file
```

Encrypting an analysis

```
mydata <- read.csv("secret.csv")

newdata <- my_analysis_function(mydata)

saveRDS(newdata, "export.rds")
```

Encrypting an analysis

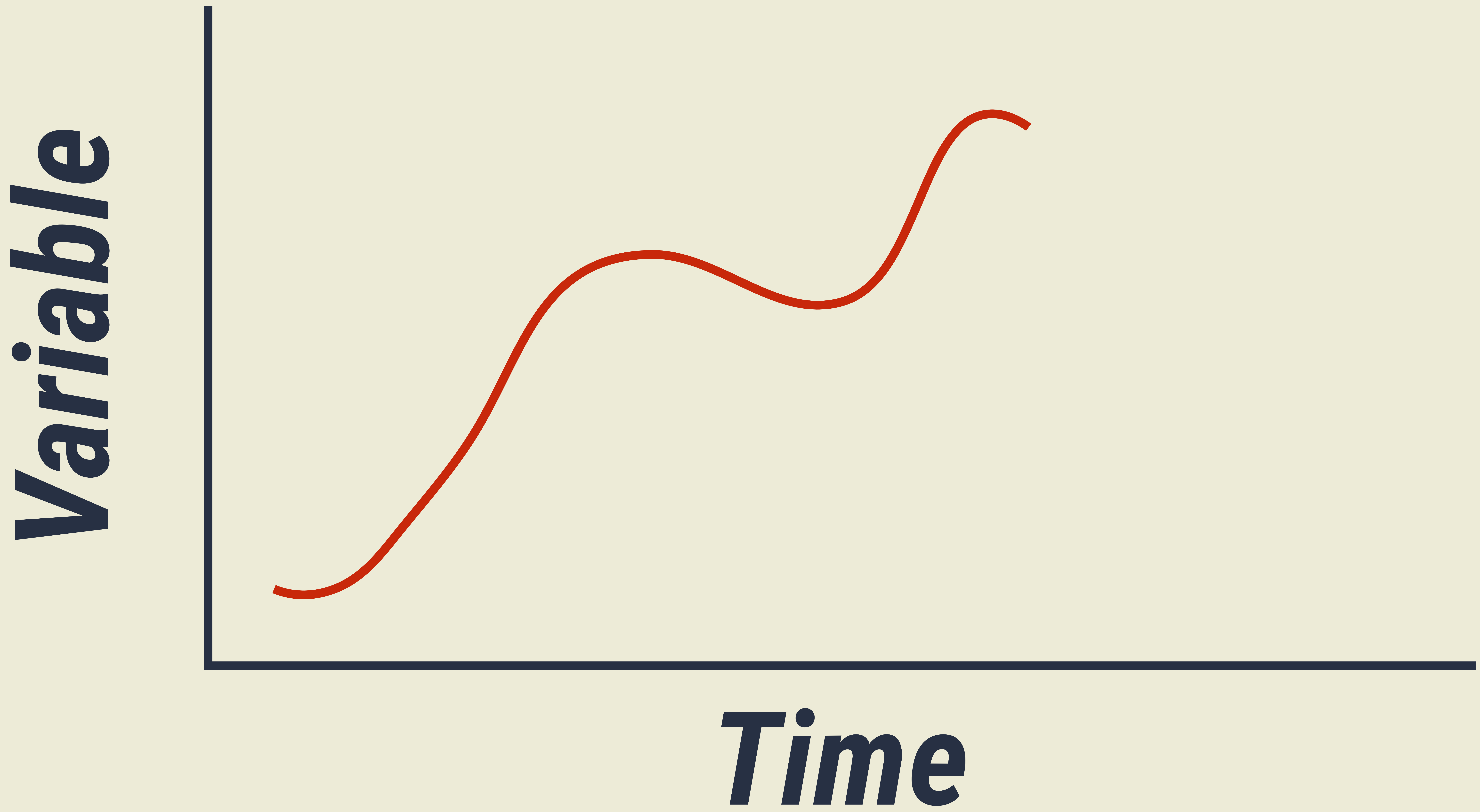
```
mydata <- cyphr::decrypt(read.csv("secret.csv"), key)

newdata <- my_analysis_function(mydata)

cyphr::encrypt(saveRDS(newdata, "export.rds"), key)
```

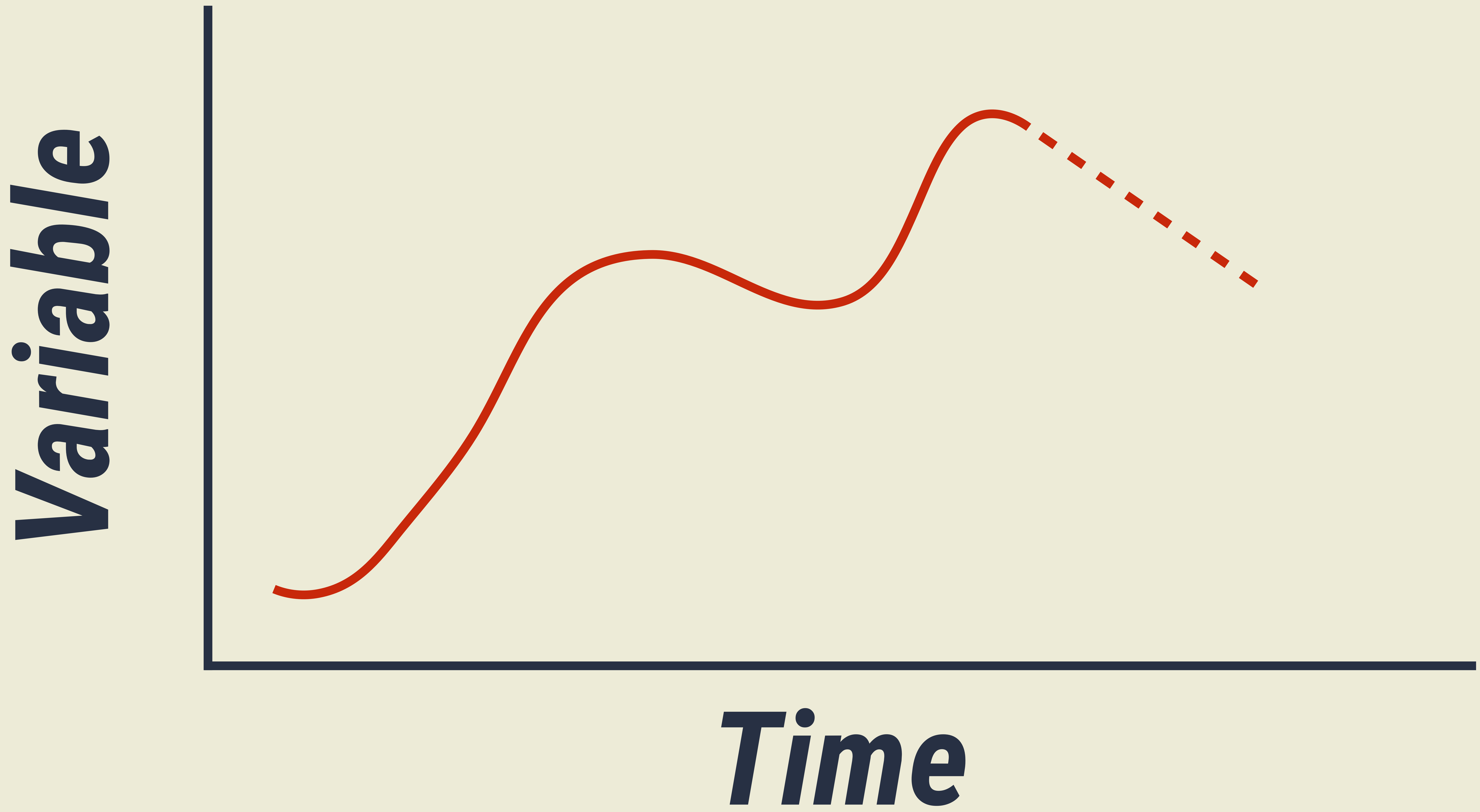
MARMITE
YEAST EXTRACT
MARMITE
CONTAINS B VITAMINS · 10
VEGETARIAN
WARNING

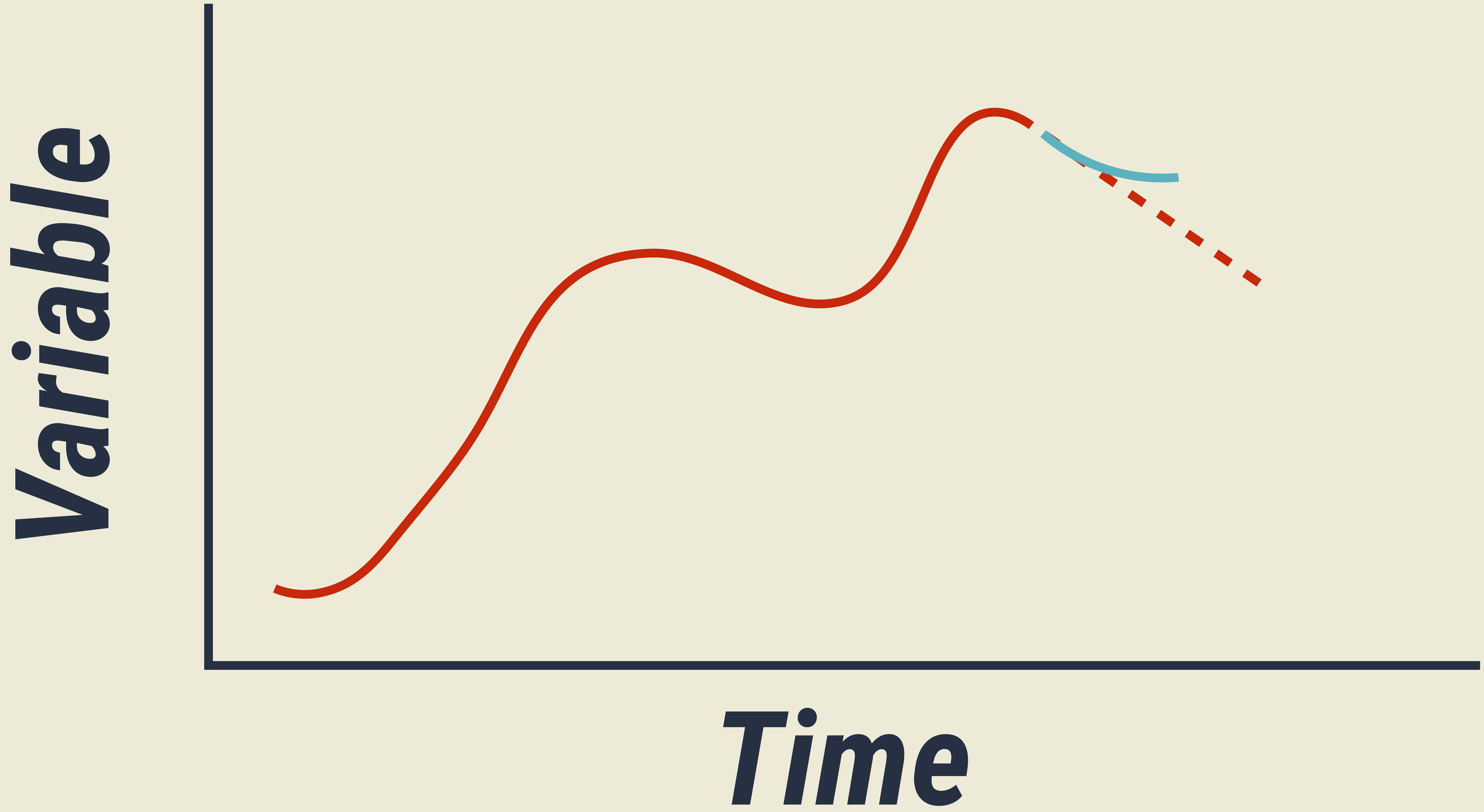
Differential equations

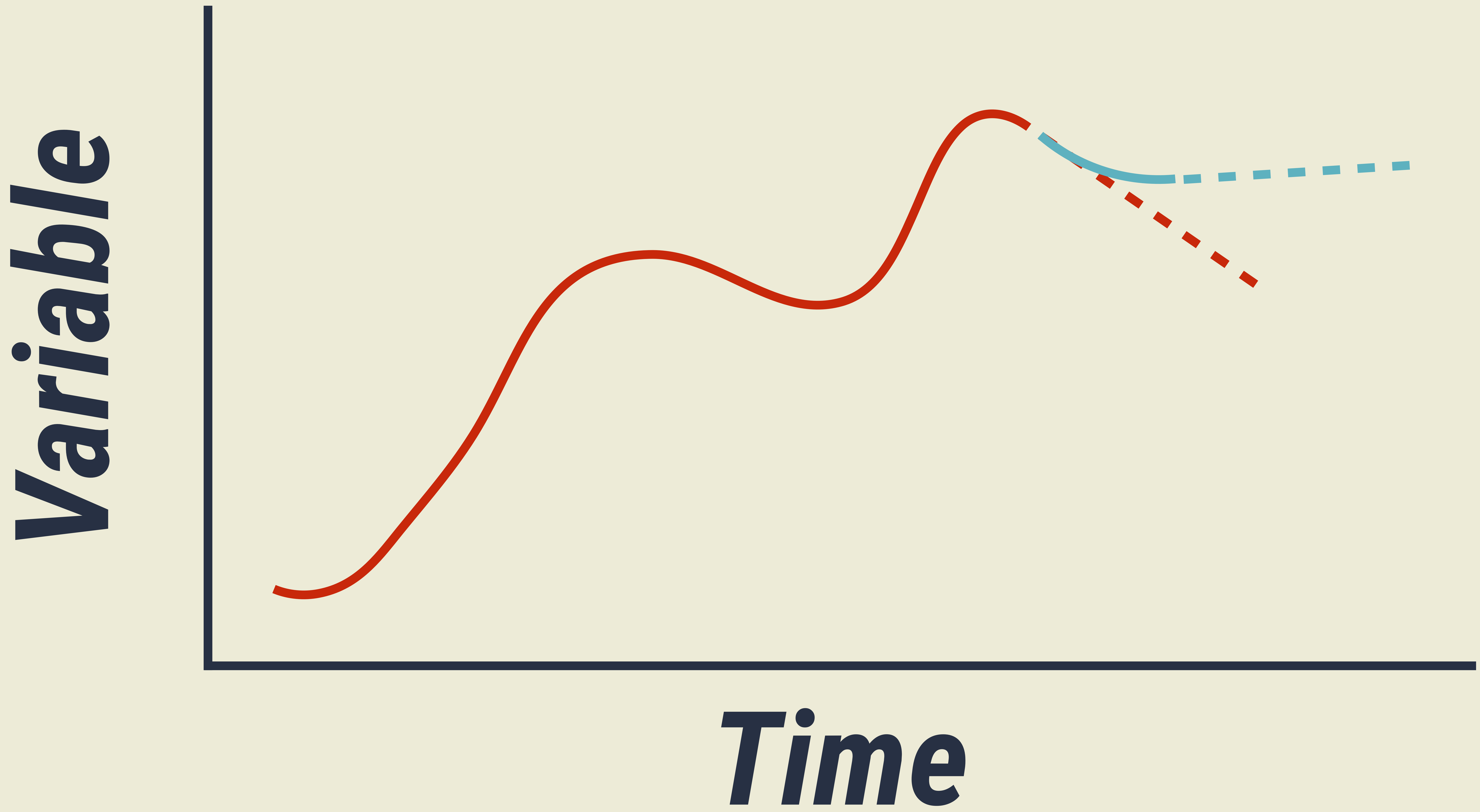
$$\begin{tikzcd} S \arrow[r] & I \arrow[r] & R \end{tikzcd}$$

$$\frac{dS}{dt} \qquad \frac{dI}{dt} \qquad \frac{dR}{dt}$$


Variable
Time


Variable
Time


Variable
Time



Variable
Time

easy

or

fast

easy

```
lorenz <- function(t, y, parms)
{
  sigma <- parms[1]
  R <- parms[2]
  b <- parms[3]
  y1 <- y[1]
  y2 <- y[2]
  y3 <- y[3]
  list(c(sigma * (y2 - y1),
         R * y1 - y2 - y1 * y3,
         -b * y3 + y1 * y2))
}
```

easy

```r
lorenz <- function(t, y, parms)
{
  sigma <- parms[1]
  R <- parms[2]
  b <- parms[3]
  y1 <- y[1]
  y2 <- y[2]
  y3 <- y[3]
  list(c(sigma * (y2 - y1),
         R * y1 - y2 - y1 * y3,
         -b * y3 + y1 * y2))
}
```

```r
deSolve::ode(t, y, lorenz)
```

```c
void initmod(void (* odeparms)(int *, double *)) {
  int N = 3;
  odeparms(&N, parms);
}

void lorenz(int *n, double *t, double *y, double *dydt, double *yout, int *ip)
{
  double sigma = parms[0];
  double R = parms[1];
  double b = parms[2];
  double y1 = y[0];
  double y2 = y[1];
  double y3 = y[2];
  dydt[0] = sigma * (y2 - y1);
  dydt[1] = R * y1 - y2 - y1 * y3;
  dydt[2] = -b * y3 + y1 * y2;
}
```

fast

```c
void initmod(void (* odeparms)(int *, double *)) {
  int N = 3;
  odeparms(&N, parms);
}

void lorenz(int *n, double *t, double *y, double *dydt, double *yout, int *ip)
{
  double sigma = parms[0];
  double R = parms[1];
  double b = parms[2];
  double y1 = y[0];
  double y2 = y[1];
  double y3 = y[2];
  dydt[0] = sigma * (y2 - y1);
  dydt[1] = R * y1 - y2 - y1 * y3;
  dydt[2] = -b * y3 + y1 * y2;
}
```

fast

```r
deSolve::ode(t, y, "lorenz", initfunc = "initmod", dllname = "lorenz")
```

```c
void initmod(void (* odeparms)(int *, double *)) {
  int N = 3;
  odeparms(&N, parms);
}

void lorenz(int *n, double *t, double *y, double *dydt, double *yout, int *ip)
{
  double sigma = parms[0];
  double R = parms[1];
  double b = parms[2];
  double y1 = y[0];
  double y2 = y[1];
  double y3 = y[2];
  dydt[0] = sigma * (y2 - y1);
  dydt[1] = R * y1 - y2 - y1 * y3;
  dydt[2] = -b * y3 + y1 * y2;
}
```

```r
lorenz <- function(t, y, parms)
{
  sigma <- parms[1]
  R <- parms[2]
  b <- parms[3]
  y1 <- y[1]
  y2 <- y[2]
  y3 <- y[3]
  list(c(sigma * (y2 - y1),
         R * y1 - y2 - y1 * y3,
         -b * y3 + y1 * y2))
}
```

```c
void initmod(void (* odeparms)(int *, double *)) {
  int N = 3;
  odeparms(&N, parms);
}

void lorenz(int *n, double *t, double *y, double *dydt, double *yout, int *ip)
{
  double sigma = parms[0];
  double R = parms[1];
  double b = parms[2];
  double y1 = y[0];
  double y2 = y[1];
  double y3 = y[2];
  dydt[0] = sigma * (y2 - y1);
  dydt[1] = R * y1 - y2 - y1 * y3;
  dydt[2] = -b * y3 + y1 * y2;
}
```

```r
lorenz <- function(t, y, parms)
{
  sigma <- parms[1]
  R <- parms[2]
  b <- parms[3]
  y1 <- y[1]
  y2 <- y[2]
  y3 <- y[3]
  list(c(sigma * (y2 - y1),
         R * y1 - y2 - y1 * y3,
         -b * y3 + y1 * y2))
}
```

```c
void initmod(void (* odeparms)(int *, double *)) {
  int N = 3;
  odeparms(&N, parms);
}

void lorenz(int *n, double *t, double *y, double *dydt, double *yout, int *ip)
{
  double sigma = parms[0];
  double R = parms[1];
  double b = parms[2];
  double y1 = y[0];
  double y2 = y[1];
  double y3 = y[2];
  dydt[0] = sigma * (y2 - y1);
  dydt[1] = R * y1 - y2 - y1 * y3;
  dydt[2] = -b * y3 + y1 * y2;
}
```

```r
lorenz <- function(t, y, parms)
{
  sigma <- parms[1]
  R <- parms[2]
  b <- parms[3]
  y1 <- y[1]
  y2 <- y[2]
  y3 <- y[3]
  list(c(sigma * (y2 - y1),
         R * y1 - y2 - y1 * y3,
         -b * y3 + y1 * y2))
}
```

```c
void initmod(void (* odeparms)(int *, double *)) {
  int N = 3;
  odeparms(&N, parms);
}

void lorenz(int *n, double *t, double *y, double *dydt, double *yout, int *ip)
{
  double sigma = parms[0];
  double R = parms[1];
  double b = parms[2];
  double y1 = y[0];
  double y2 = y[1];
  double y3 = y[2];
  dydt[0] = sigma * (y2 - y1);
  dydt[1] = R * y1 - y2 - y1 * y3;
  dydt[2] = -b * y3 + y1 * y2;
}
```

```r
lorenz <- function(t, y, parms)
{
  sigma <- parms[1]
  R <- parms[2]
  b <- parms[3]
  y1 <- y[1]
  y2 <- y[2]
  y3 <- y[3]
  list(c(sigma * (y2 - y1),
         R * y1 - y2 - y1 * y3,
         -b * y3 + y1 * y2))
}
```

```c
void initmod(void (* odeparms)(int *, double *)) {
  int N = 3;
  odeparms(&N, parms);
}

void lorenz(int *n, double *t, double *y, double *dydt, double *yout, int *ip)
{
  double sigma = parms[0];
  double R = parms[1];
  double b = parms[2];
  double y1 = y[0];
  double y2 = y[1];
  double y3 = y[2];
  dydt[0] = sigma * (y2 - y1);
  dydt[1] = R * y1 - y2 - y1 * y3;
  dydt[2] = -b * y3 + y1 * y2;
}
```

```r
lorenz <- function(t, y, parms)
{
  sigma <- parms[1]
  R <- parms[2]
  b <- parms[3]
  y1 <- y[1]
  y2 <- y[2]
  y3 <- y[3]
  list(c(sigma * (y2 - y1),
         R * y1 - y2 - y1 * y3,
         -b * y3 + y1 * y2))
}
```

```
void initmod(void (* odeparms)(int *, double *)) {
  int N = 3;
  odeparms(&N, parms);
}

void lorenz(int *n, double *t, double *y, double *dydt, double *yout, int *ip)
{
  double sigma = parms[0];
  double R = parms[1];
  double b = parms[2];
  double y1 = y[0];
  double y2 = y[1];
  double y3 = y[2];
  dydt[0] = sigma * (y2 - y1);
  dydt[1] = R * y1 - y2 - y1 * y3;
  dydt[2] = -b * y3 + y1 * y2;
}
```

easy

or

fast

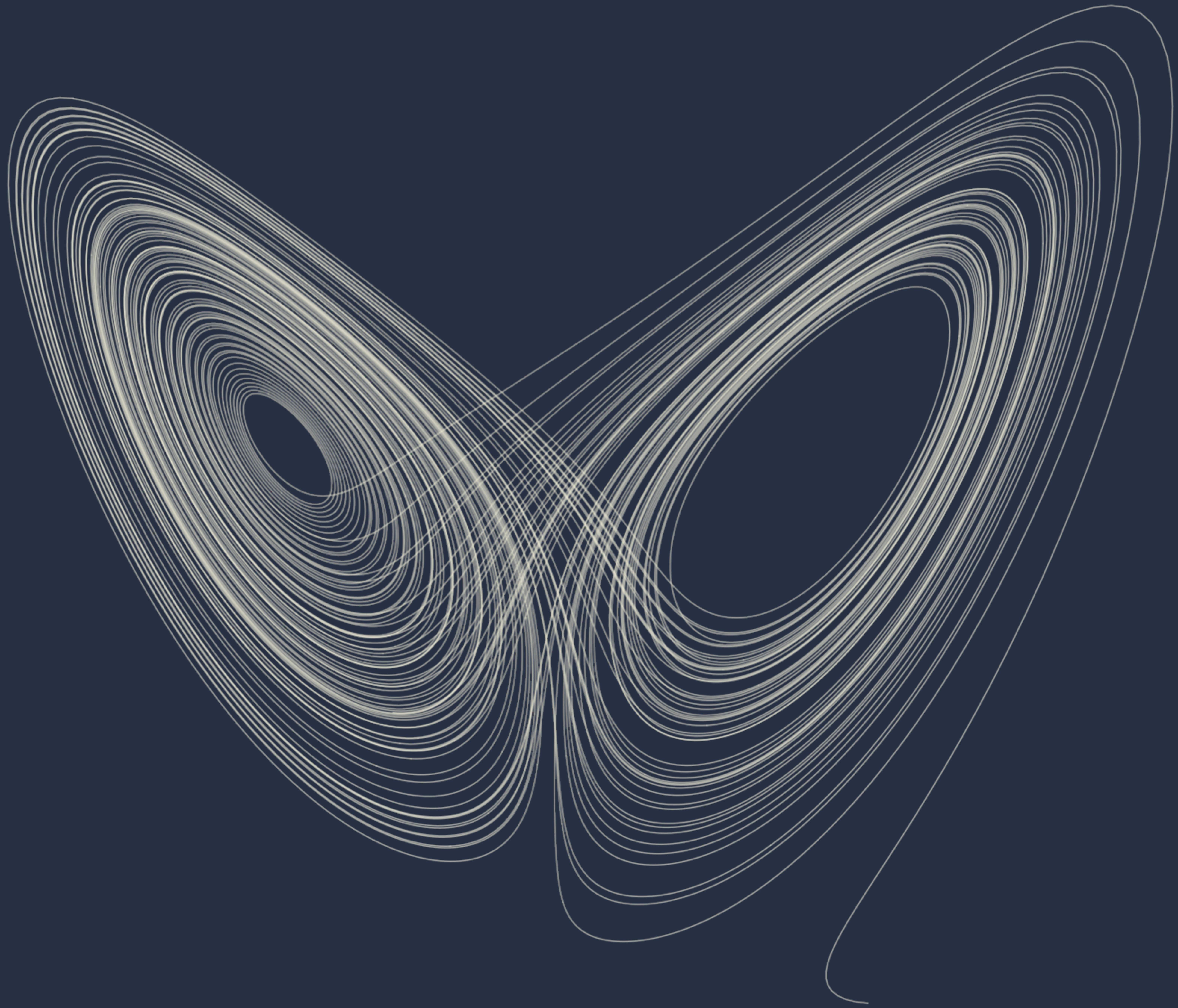
easy
and
fast

```r
lorenz <- odin::odin({
  ## Derivatives
  deriv(y1) <- sigma * (y2 - y1)
  deriv(y2) <- R * y1 - y2 - y1 * y3
  deriv(y3) <- -b * y3 + y1 * y2

  ## Initial conditions
  initial(y1) <- 10.0
  initial(y2) <- 1.0
  initial(y3) <- 1.0

  ## parameters
  sigma <- user()
  R     <- user()
  b     <- user()
})
```

odin

odin

```r
lorenz <- odin::odin({
  ...
  sigma <- user()
  R     <- user()
  b     <- user()
})

model <- lorenz(sigma = 10.0,
                R = 28.0,
                b = 8 / 3)
t <- seq(0, 50, length.out = 10000)
y <- model$run(t)
```



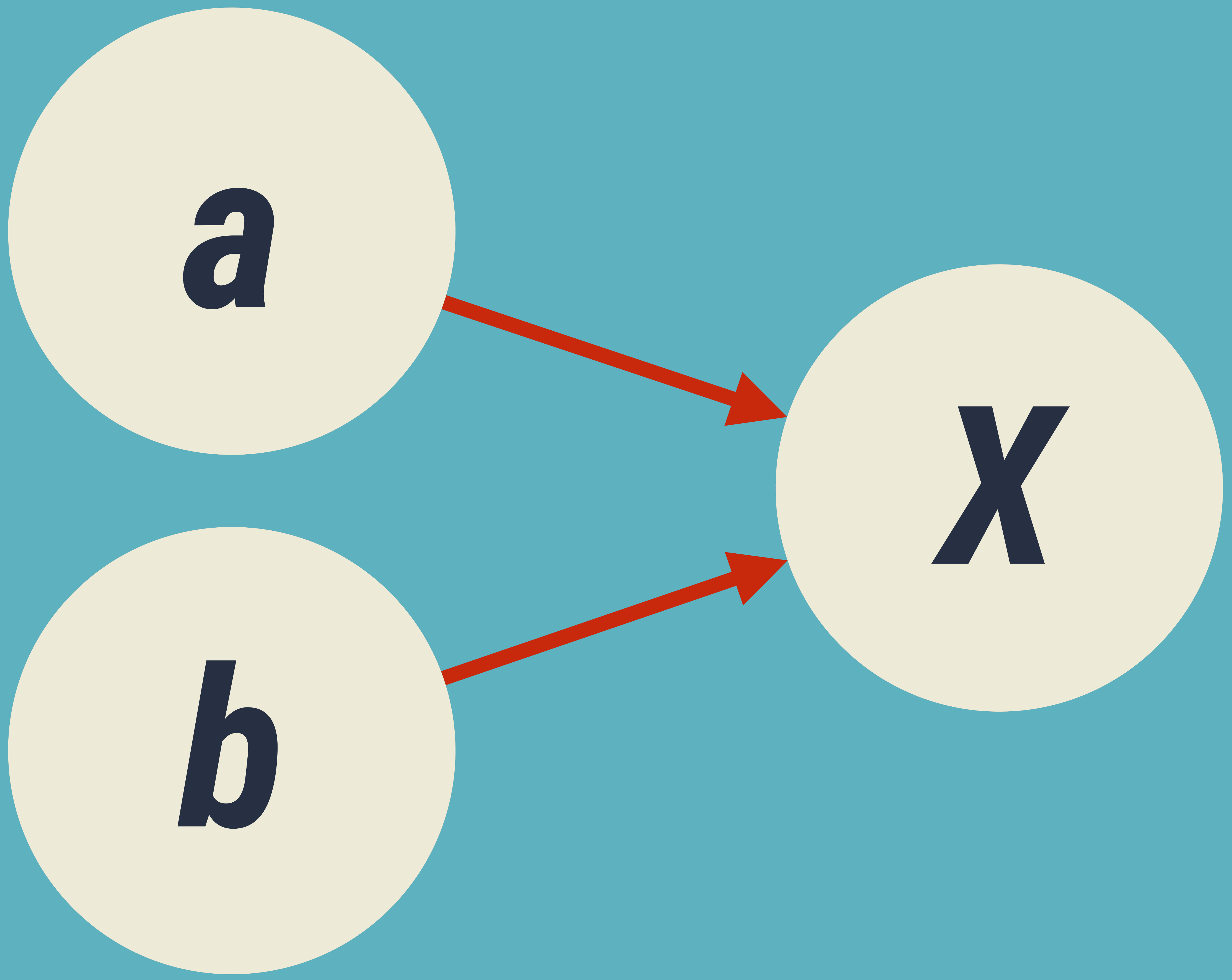
Rewriting expressions

```
deriv(y1) <- sigma * (y2 - y1)

list(`<-`,
     deriv(y1),
     sigma * (y2 - y1))

dydt[0] = sigma * (y2 - y1);
```

Rewriting expressions

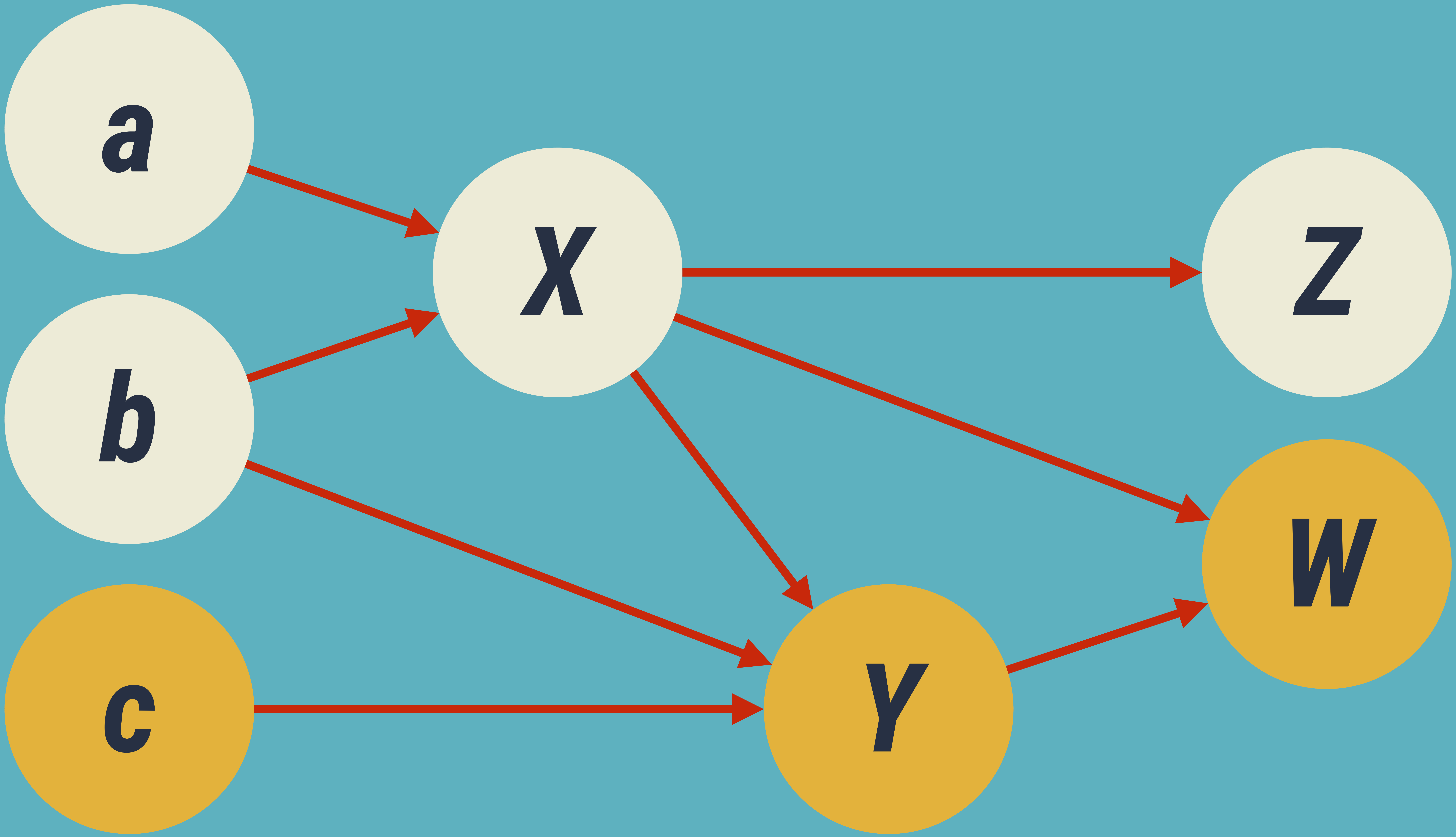
```
deriv(y1[]) <- sigma * (y2[i] - y1[i])

list(`<-`,
     deriv(y1[]),
     sigma * (y2[i] - y1[i]))

for (size_t i = 0; i < len_y1; ++i) {
  dydt[i] = sigma * (y2[i] - y1[i]);
}
```

```r
lorenz <- odin::odin({
  ## Derivatives
  deriv(y1) <- sigma * (y2 - y1)
  deriv(y2) <- R * y1 - y2 - y1 * y3
  deriv(y3) <- -b * y3 + y1 * y2

  ## Initial conditions
  initial(y1) <- 10.0
  initial(y2) <- 1.0
  initial(y3) <- 1.0

  ## parameters
  sigma <- user()
  R     <- user()
  b     <- user()
})
```



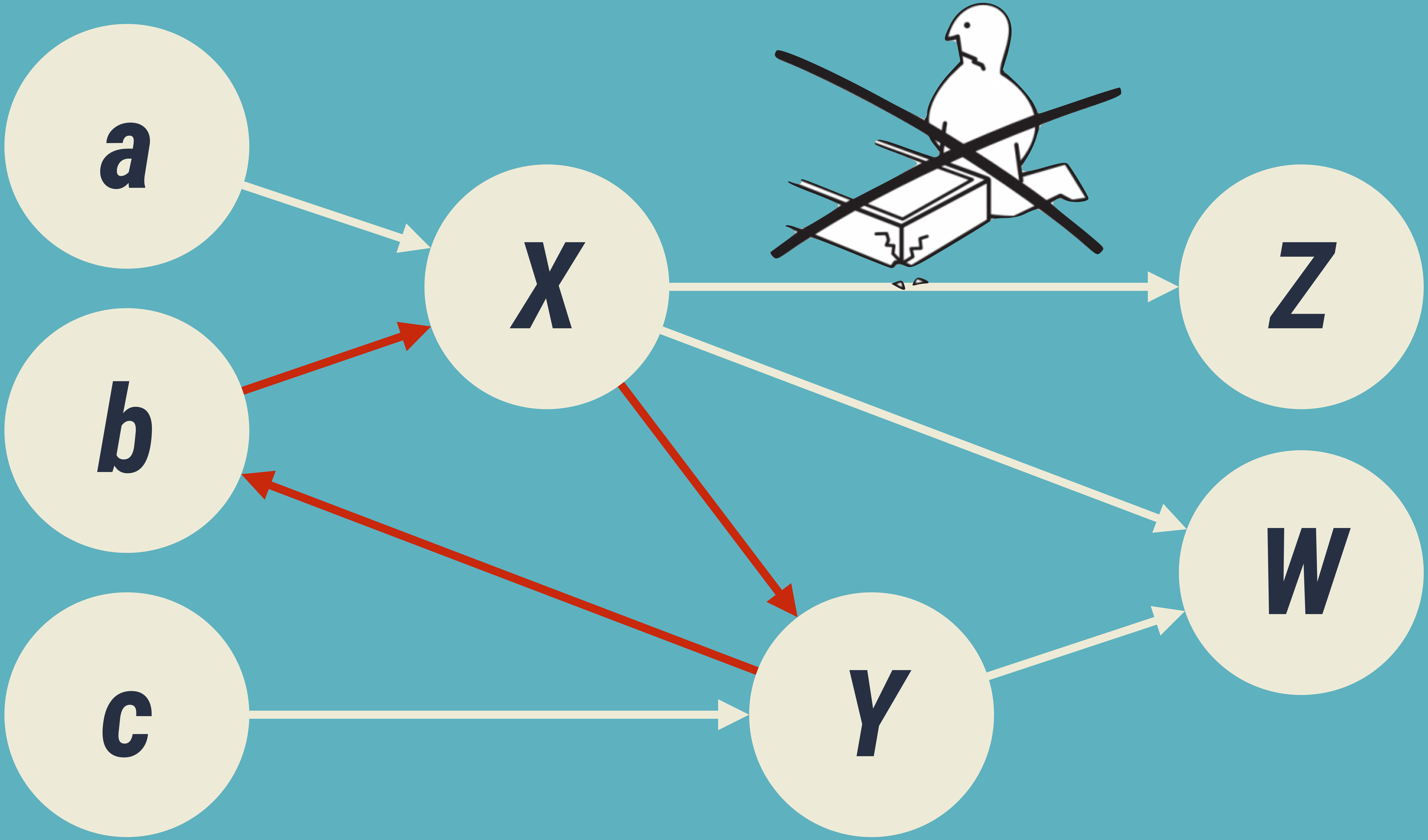
$$X = a + b$$

$$\begin{tikzcd}
a \arrow[dr] & & & \\
& X \arrow[rr] \arrow[rrdd] \arrow[rddd] & & Z \\
b \arrow[ur] \arrow[rrdd] & & & \\
& & & W \\
c \arrow[rr] & & Y \arrow[ur] &
\end{tikzcd}$$



$$\begin{tikzcd}
a \arrow[dr] & & & \\
& X \arrow[rr] \arrow[rrdd] \arrow[rddd] & & Z \\
b \arrow[ur] \arrow[rrdd] & & & \\
& & & W \\
c \arrow[rr] & & Y \arrow[ru] &
\end{tikzcd}$$

$$\begin{tikzcd}
a \arrow[dr, phantom] \arrow[r] & X \arrow[rr] \arrow[rrd] \arrow[rdd] & & Z \\
b \arrow[ur] & & & W \\
c \arrow[rr] & & Y \arrow[ull] \arrow[ur] &
\end{tikzcd}$$


a
b
c
X
Y
Z
W

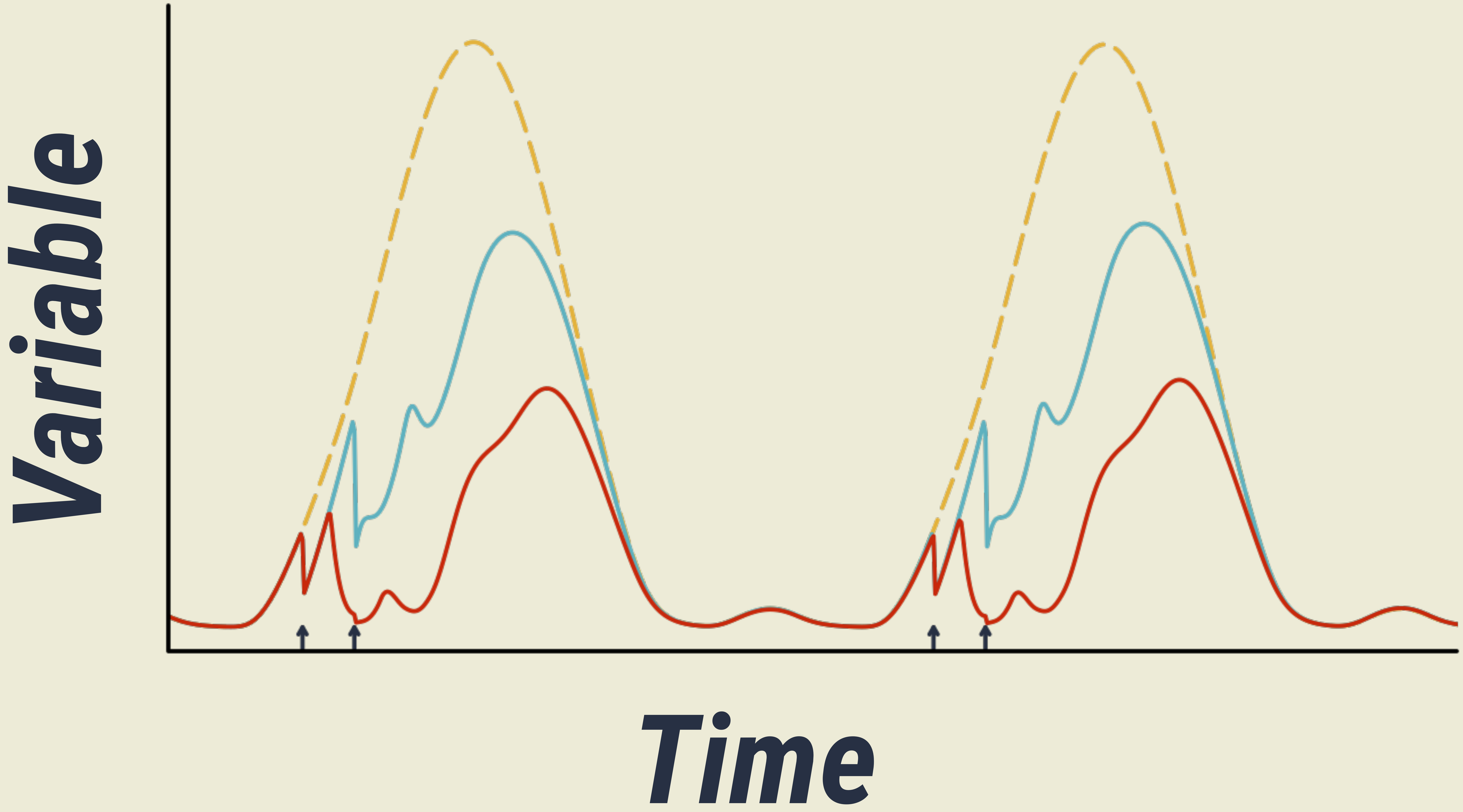
$$\begin{tikzcd} S \arrow[r] & I \arrow[r] & R \end{tikzcd}$$

$$\frac{dS}{dt} \qquad \frac{dI}{dt} \qquad \frac{dR}{dt}$$

```r
sir <- odin::odin({
  deriv(S) <- -beta * S * I / N
  deriv(I) <- beta * S * I / N - gamma * I
  deriv(R) <- gamma * I

  initial(S) <- 1000
  initial(I) <- 1
  initial(R) <- 0

  N <- S + I + R

  beta <- 0.2
  gamma <- 0.1
})
```

```
deriv(S) <- -beta * S * I / N
             N <- S + I + R
```

$$\begin{tikzcd}
S \arrow[dr, dashed] \arrow[drr] & & \\
I \arrow[r, dashed] \arrow[rr, bend left] & N \arrow[r] & \frac{dS}{dt} \\
R \arrow[ur, dashed] & beta \arrow[ur] &
\end{tikzcd}$$


Variable
Time

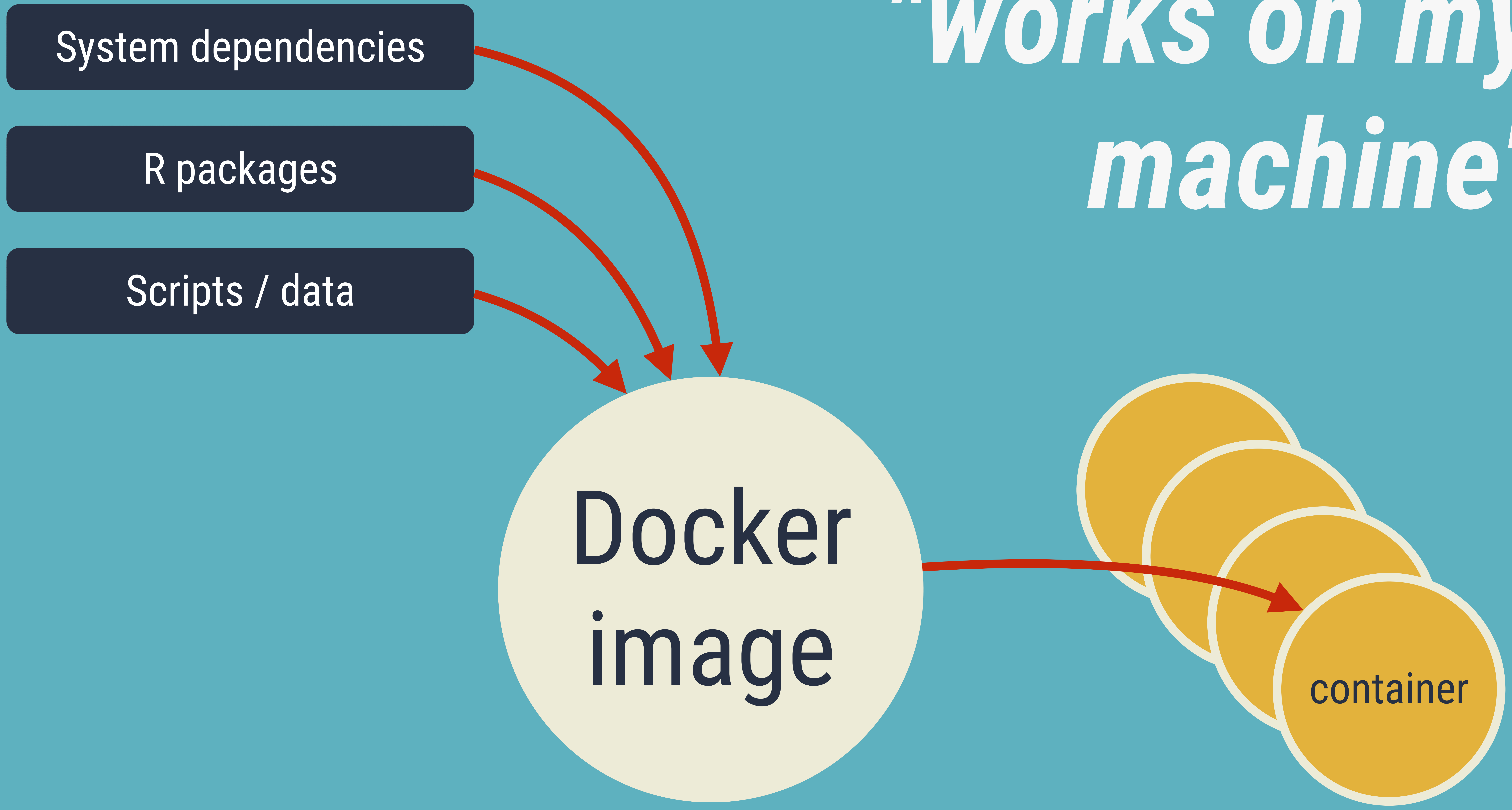
Docker

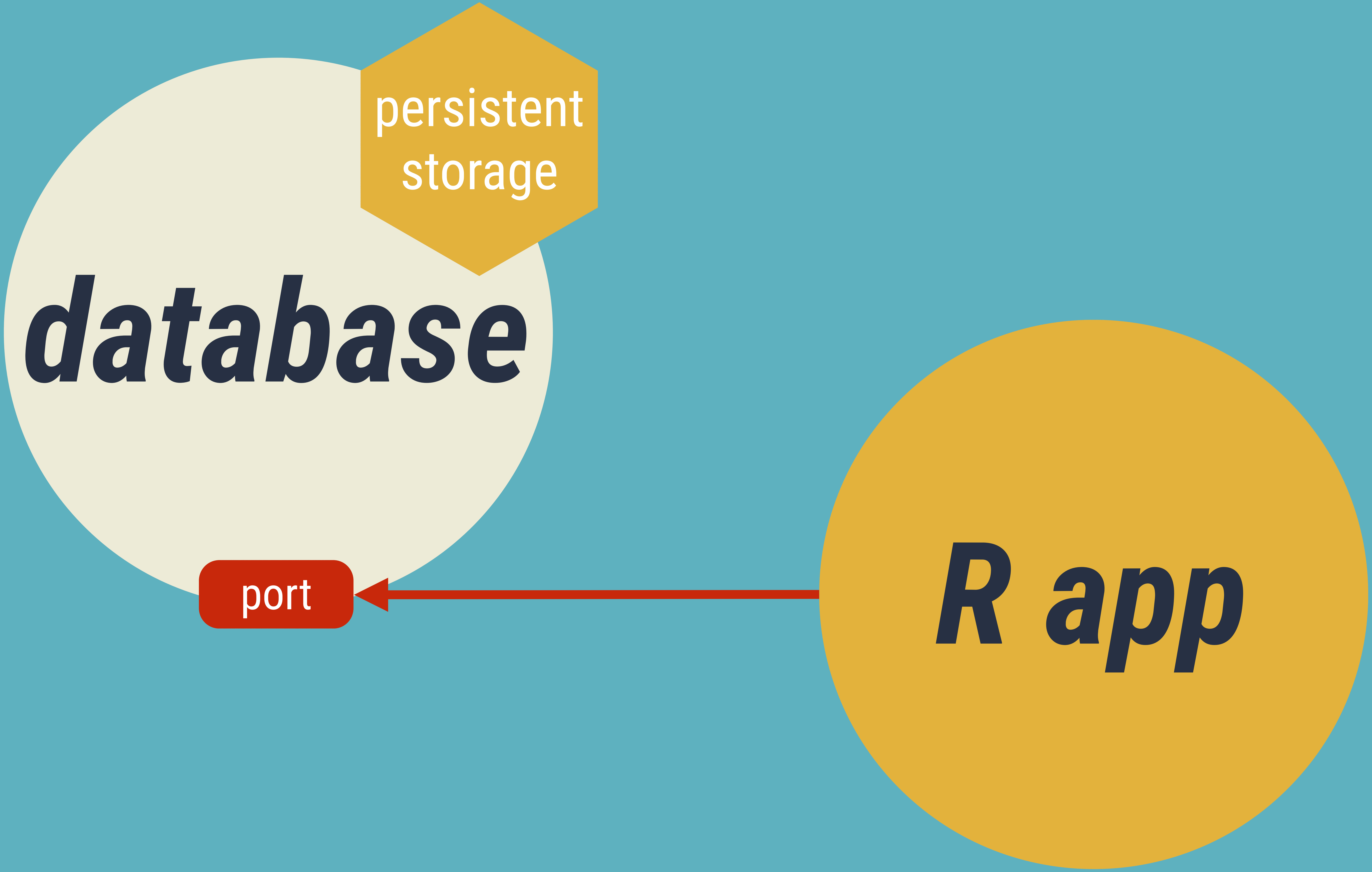
what is docker?

Docker is a software technology providing operating-system-level virtualization also known as containers, promoted by the company Docker, Inc. Docker provides an additional layer of abstraction and automation of operating-system-level virtualization on Windows and Linux. Docker uses the resource isolation features of the Linux kernel such as cgroups and kernel namespaces, and a union-capable file system such as OverlayFS and others to allow independent "containers" to run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines.

why use docker?

"works on my machine"




persistent storage
database
port
R app

persistent storage
database
port
port
shiny app
proxy
port

E4

CHINA SHIPPING LINE
CSCL GLOBE
HONG KONG
EUROMAX
UASC
CHINA SHIPPING
HAMBURG SÜD
HANJIN

swagger

```
/containers/create:
  post:
    summary: "Create a container"
    consumes:
       - "application/json"
    parameters:
       - name: "name"
         in: "query"
         description: "Assign the specified name to the container."
         type: "string"
    responses:
       201:
         description: "Container created successfully"
         schema:
           type: "object"
          description: "OK response to ContainerCreate operation"
          properties:
             Id:
               description: "The ID of the created container"
               type: "string"
```

where

```
/containers/create:
  post:
    summary: "Create a container"
    consumes:
       - "application/json"
    parameters:
       - name: "name"
         in: "query"
         description: "Assign the specified name to the container."
         type: "string"
    responses:
       201:
         description: "Container created successfully"
         schema:
           type: "object"
          description: "OK response to ContainerCreate operation"
          properties:
             Id:
               description: "The ID of the created container"
               type: "string"
```

parameters

```
/containers/create:
  post:
    summary: "Create a container"
    consumes:
       - "application/json"
    parameters:
       - name: "name"
         in: "query"
         description: "Assign the specified name to the container."
         type: "string"
    responses:
       201:
         description: "Container created successfully"
         schema:
           type: "object"
          description: "OK response to ContainerCreate operation"
          properties:
             Id:
               description: "The ID of the created container"
               type: "string"
```

returning

```
/containers/create:
  post:
    summary: "Create a container"
    consumes:
       - "application/json"
    parameters:
       - name: "name"
         in: "query"
         description: "Assign the specified name to the container."
         type: "string"
    responses:
       201:
         description: "Container created successfully"
         schema:
           type: "object"
          description: "OK response to ContainerCreate operation"
          properties:
             Id:
               description: "The ID of the created container"
               type: "string"
```

```
/containers/create:
  post:
    summary: "Create a container"
    consumes:
      - "application/json"
    parameters:
      - name: "name"
        in: "query"
        description: "Assign the specified name to the container."
        type: "string"
    responses:
      201:
        description: "Container created successfully"
        schema:
          type: "object"
         description: "OK response to ContainerCreate operation"
         properties:
            Id:
              description: "The ID of the created container"
              type: "string"
```

90 methods

10,000 lines

12 versions

```
if tmpfs:
    if version_lt(version, '1.22'):
        raise host_config_version_error('tmpfs', '1.22')
    self["Tmpfs"] = convert_tmpfs_mounts(tmpfs)

if userns_mode:
    if version_lt(version, '1.23'):
        raise host_config_version_error('userns_mode', '1.23')
    self['UsernsMode'] = userns_mode

if pids_limit:
    if version_lt(version, '1.23'):
        raise host_config_version_error('pids_limit', '1.23')
    self["PidsLimit"] = pids_limit

if isolation:
    if version_lt(version, '1.24'):
        raise host_config_version_error('isolation', '1.24')
    self['Isolation'] = isolation

if auto_remove:
    if version_lt(version, '1.25'):
        raise host_config_version_error('auto_remove', '1.25')
    self['AutoRemove'] = auto_remove
```

if else hell

How to write a function

```
add <- function(a, b) {
  a + b
}
```

How to **build** a function

```r
add <- function(a, b) {
  a + b
}
args <- alist(a =, b =)
body <- quote(a + b)
add <- as.function(c(args, body))
```

How to draw an Owl.

"A fun and creative guide for beginners"

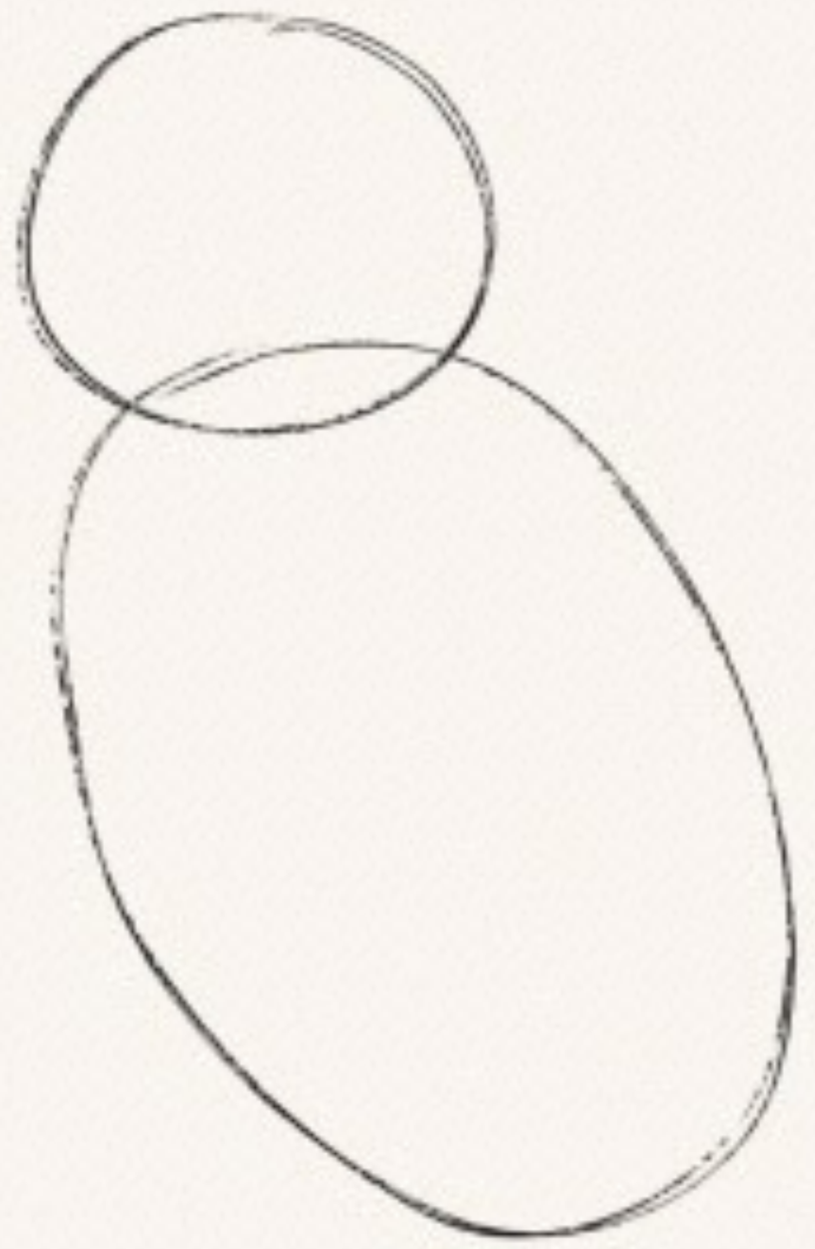


Fig 1. Draw two circles

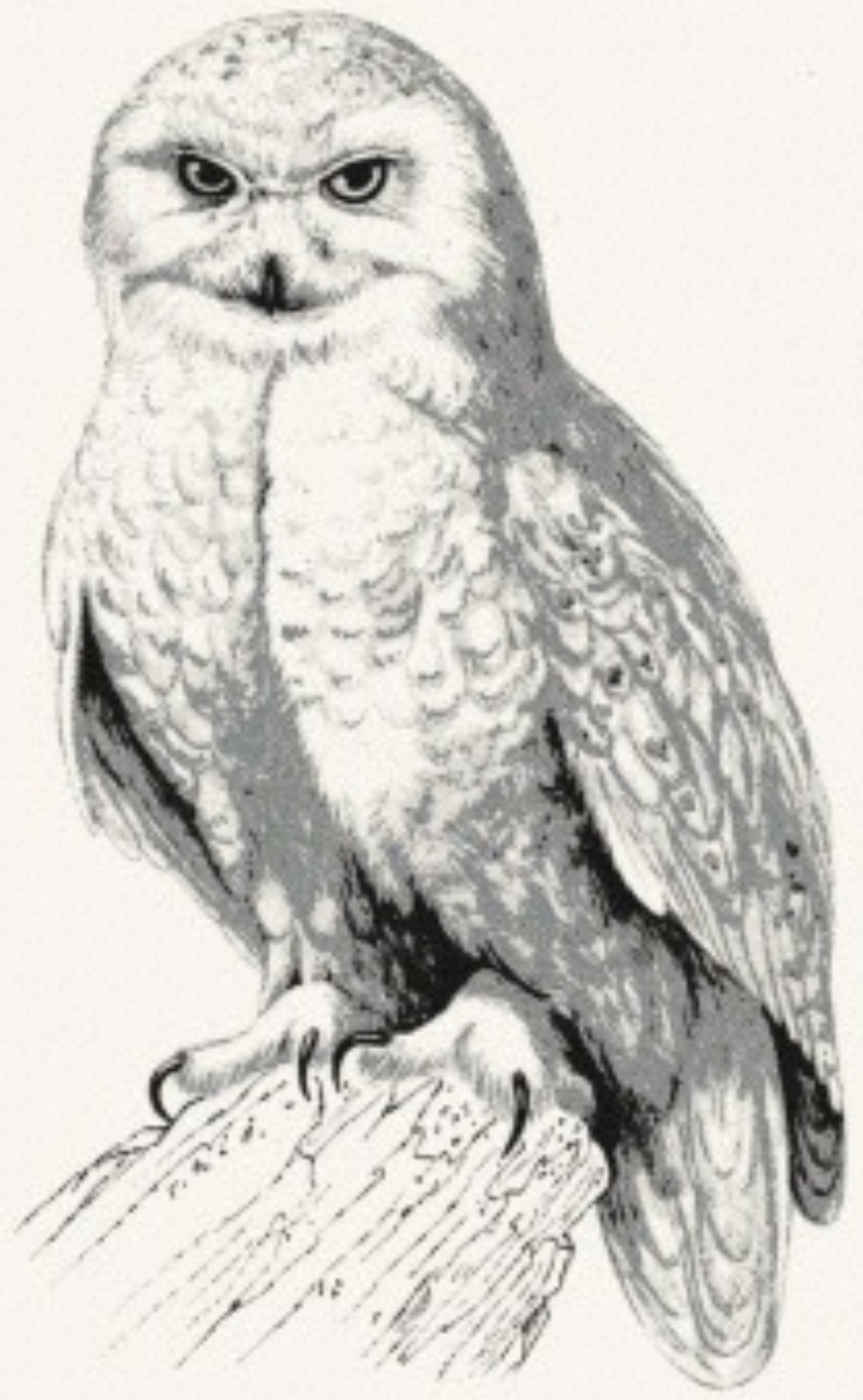


Fig 2. Draw the rest of the damn Owl

stevedore

```
docker <- stevedore::docker_client()
```

Stevedore

```
docker <- stevedore::docker_client(
    api_version = "1.35")
```

Testing packages

1. Install database
2. Configure & set up passwords
3. Use database in package tests
4. Make sure you clean up properly!

```
echo mysql-server mysql-server/root_password password $MYSQL_PASSWORD | \
        debconf-set-selections
echo mysql-server mysql-server/root_password_again password $MYSQL_PASSWORD | \
        debconf-set-selections
apt-get install -y mysql-server

systemctl stop mysql
mv /var/lib/mysql /mnt/data/mysql
ln -s /mnt/data/mysql /var/lib/mysql

echo "alias /var/lib/mysql/ -> /mnt/data/mysql," >> \
    /etc/apparmor.d/tunables/alias
sudo systemctl restart apparmor
systemctl start mysql

mysql -u root -p$MYSQL_PASSWORD -e 'show databases;'| grep teamcity > /dev/null
if [ "$?" = "1" ]; then
    cat > /tmp/database-setup.sql <<EOF
CREATE DATABASE $TEAMCITY_DB_NAME DEFAULT CHARACTER SET utf8 COLLATE utf8_bin;
CREATE USER '$TEAMCITY_DB_USER'@'%' IDENTIFIED BY '$TEAMCITY_DB_PASS';
GRANT ALL ON $TEAMCITY_DB_NAME.* TO '$TEAMCITY_DB_USER'@'%';
EOF
    mysql -u root -p$MYSQL_PASSWORD < /tmp/database-setup.sql
    rm /tmp/database-setup.sql
fi
```

Testing packages

```r
env <- c("POSTGRES_PASS" = "s3cret!")
db <- docker$containers$run("postgres", ports = "2222:5432",
                            rm = TRUE, detach = TRUE,
                            env = env)
```

Testing packages

```
env <- c("POSTGRES_PASS" = "s3cret!")
db <- docker$containers$run("postgres", ports = "2222:5432",
                            rm = TRUE, detach = TRUE,
                            env = env)
con <- dbConnect(Postgres(), host = "localhost", port = 2222,
                 user = "postgres", password = "s3cret!")
dbWriteTable(con, "table", mydata)
```

Testing packages

```
env <- c("POSTGRES_PASS" = "s3cret!")
db <- docker$containers$run("postgres", ports = "2222:5432",
                            rm = TRUE, detach = TRUE,
                            env = env)
con <- dbConnect(Postgres(), host = "localhost", port = 2222,
                 user = "postgres", password = "s3cret!")
dbWriteTable(con, "table", mydata)
dbGetQuery(con, "SELECT * FROM table LIMIT 20")
```

Testing packages

```r
env <- c("POSTGRES_PASS" = "s3cret!")
db <- docker$containers$run("postgres", ports = "2222:5432",
                            rm = TRUE, detach = TRUE,
                            env = env)
con <- dbConnect(Postgres(), host = "localhost", port = 2222,
                 user = "postgres", password = "s3cret!")
dbWriteTable(con, "table", mydata)
dbGetQuery(con, "SELECT * FROM table LIMIT 20")
db$stop()
```

Encryption

Differential equations

Docker

Encryption

cyphr github.com/ropensci/cyphr

Differential equations

odin github.com/mrc-ide/odin

Docker

stevedore github.com/richfitz/stevedore

R's weirdnesses are fun & useful

Rich FitzJohn

richfitz