staypuft: object validation and serialization

& should this even be a package?

Scott Chamberlain (@sckottie)

pain point: serialization

converting data in one format to another format

especially painful when complex

other languages have good ideas

marshmallow - a Python library

marshmallow

> *A lightweight library for converting complex objects to and from simple Python datatypes.*

An example with marshmallow

```python
from datetime import date
from marshmallow import Schema, fields, pprint

class ArtistSchema(Schema):
    name = fields.Str()

class AlbumSchema(Schema):
    title = fields.Str()
    release_date = fields.Date()
    artist = fields.Nested(ArtistSchema())

bowie = dict(name='David Bowie')
album = dict(artist=bowie, title='Hunky Dory', release_date=date(1971, 12, 17))

schema = AlbumSchema()
result = schema.dump(album)
# { 'artist': {'name': 'David Bowie'},
#   'release_date': '1971-12-17',
#   'title': 'Hunky Dory'}

album = dict(artist=bowie, title='Hunky Dory', release_date="2020-04-14")
schema.dump(album)
# ValidationError: {'release_date': ['"2020-04-14" cannot be formatted as a date.']
```

back to R

similar art in R

- assertr (assertions for analysis pipeline)
- validate (very similar to assertr AFAICT)
- errorlocate (find errors in datasets)

- any others?

ropensci/staypuft

An example with staypuft

```r
library(staypuft)
MySchema <- Schema$new("MySchema",
  num = fields$integer(),
  uuid = fields$uuid(),
  foo = fields$boolean()
)
x <- list(num=5, uuid="9a5f6bba-4101-48e9-a7e3-b5ac456a04b5", foo=TRUE)

# all good
MySchema$dump_json(x)
#> {"name":["Jane Doe"],"title":["Howdy doody"],"num":[5.5], ...

# invalid uuid
z <- x
z$uuid <- "foo-bar"
MySchema$load(z)
#> Error: ValidationError: Not a valid UUID.

# invalid boolean
w <- x
w$foo <- "stuff"
MySchema$load(x)
#> Error: ValidationError: Not a valid boolean.
```

Use case: convert each thing to an S3 class

```r
z <- Schema$new("ArtistSchema",
  name = fields$character(),
  role = fields$character(data_key="foo_bar"),
  post_load = {
    function(x) structure(x, class = "Artist")
  },
  unknown = "exclude"
)

print.Artist <- function(x) {
  cat("Artist\n")
  cat(sprintf("  name/role: %s/%s\n", x$name, x$role))
}

artists <- list(
  list(name="David Bowie", foo_bar="lead", instrument="voice"),
  list(name="Michael Jackson", foo_bar="lead", instrument="voice")
)
json <- jsonlite::toJSON(artists)
z$load_json(json, simplifyVector = FALSE, many = TRUE)
#> [[1]]
#> Artist
#>   name/role: David Bowie/lead
#>
#> [[2]]
#> Artist
#>   name/role: Michael Jackson/lead
```

why?/use cases

- data validation: lots of potential users
- remote data sources can change: schemas help validate and catch changes
- use in scripts (most researchers): help raise issues with scripts as time goes on and data inputs change
- using R with plumbr or similar: convert data to serve to API or consume from API request bodies

To do

- Nested data works - but needs more testing
- Add more 'field' types: url, email, (domain specific types)
- Add support for user-defined fields
- Probably add an easier to use interface, less R6'y

wait …
should this even be a package
though?

When should I not make a pkg?

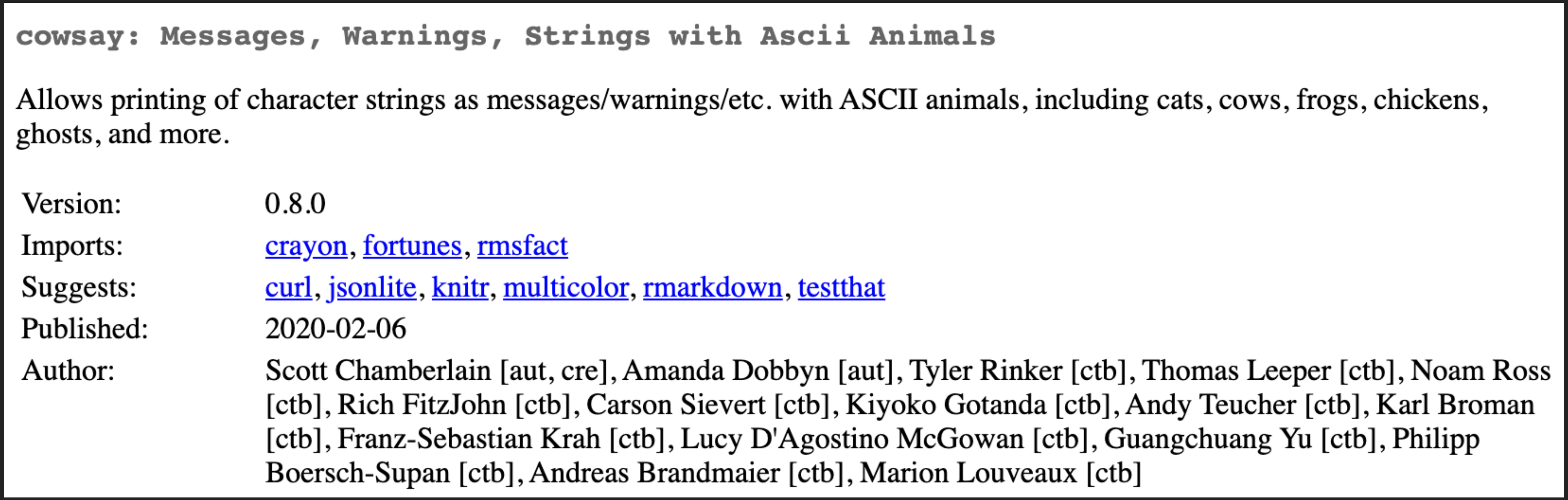
- the pkg doesn't solve actual use cases
- there's significant overlap with existing solutions
    - and maintainers are responsive
- there's higher priority/lowering hanging fruit

Use cases

For `staypuft`, likely many users

Everyone deals with objects in R

& I'm not against sillyness

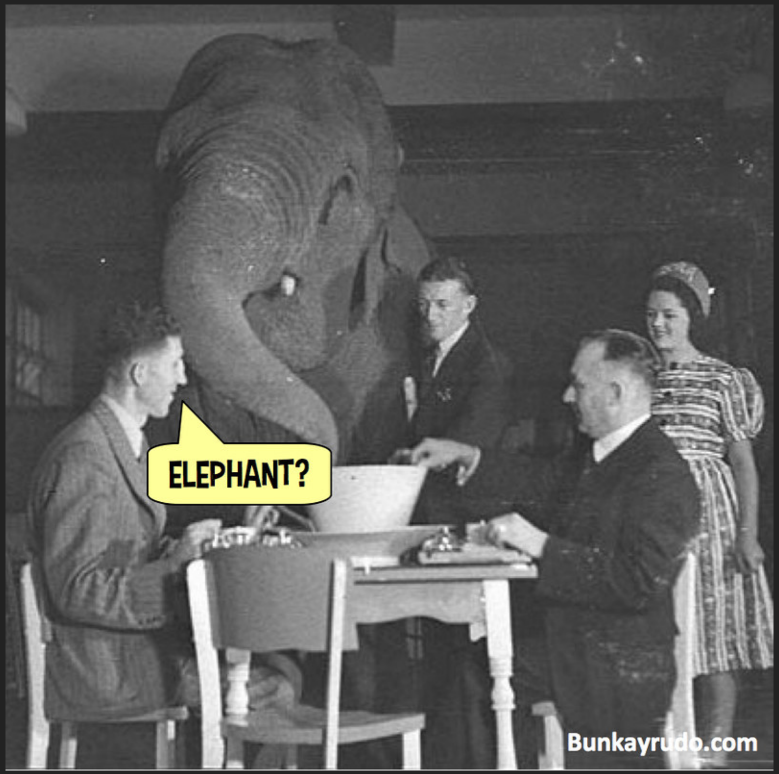


cowsay: Messages, Warnings, Strings with Ascii Animals

Allows printing of character strings as messages/warnings/etc. with ASCII animals, including cats, cows, frogs, chickens, ghosts, and more.

Version:	0.8.0
Imports:	crayon, fortunes, rmsfact
Suggests:	curl, jsonlite, knitr, multicolor, rmarkdown, testthat
Published:	2020-02-06
Author:	Scott Chamberlain [aut, cre], Amanda Dobbyn [aut], Tyler Rinker [ctb], Thomas Leeper [ctb], Noam Ross [ctb], Rich FitzJohn [ctb], Carson Sievert [ctb], Kiyoko Gotanda [ctb], Andy Teucher [ctb], Karl Broman [ctb], Franz-Sebastian Krah [ctb], Lucy D'Agostino McGowan [ctb], Guangchuang Yu [ctb], Philipp Boersch-Supan [ctb], Andreas Brandmaier [ctb], Marion Louveaux [ctb]

elephant in the room ...



[illegible]

S4 e.g.

```
setClass("BMI", representation(weight="numeric", size="numeric"))
new("BMI", weight="Hello", size=1.84)
#> Error in validObject(.Object) :
#>   invalid class “BMI” object: invalid object for slot "weight"

#>   in class "BMI": got class "character",
#>   should be or extend class "numeric"
```

But I think staypuft use cases are sufficiently different

higher priority/lower hanging fruit

- I've got many other packages
- Many of which have many users
- What if new package has a huge impact though?
  - How would I know?

So...

staypuft future is unclear

if you're interested:

ropensci/staypuft

scotttalks.info/staypuft