The big picture

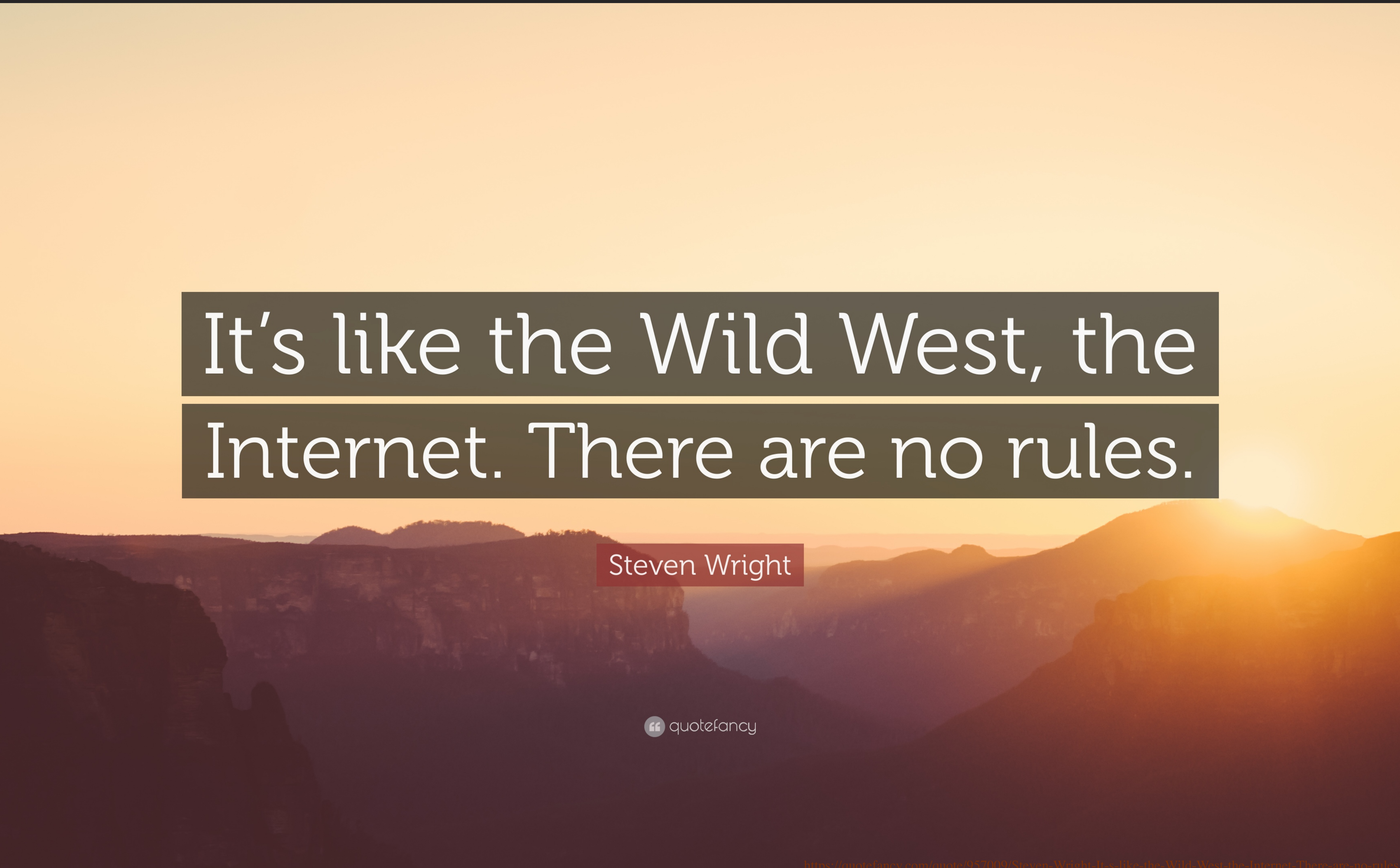
useR - July 2019

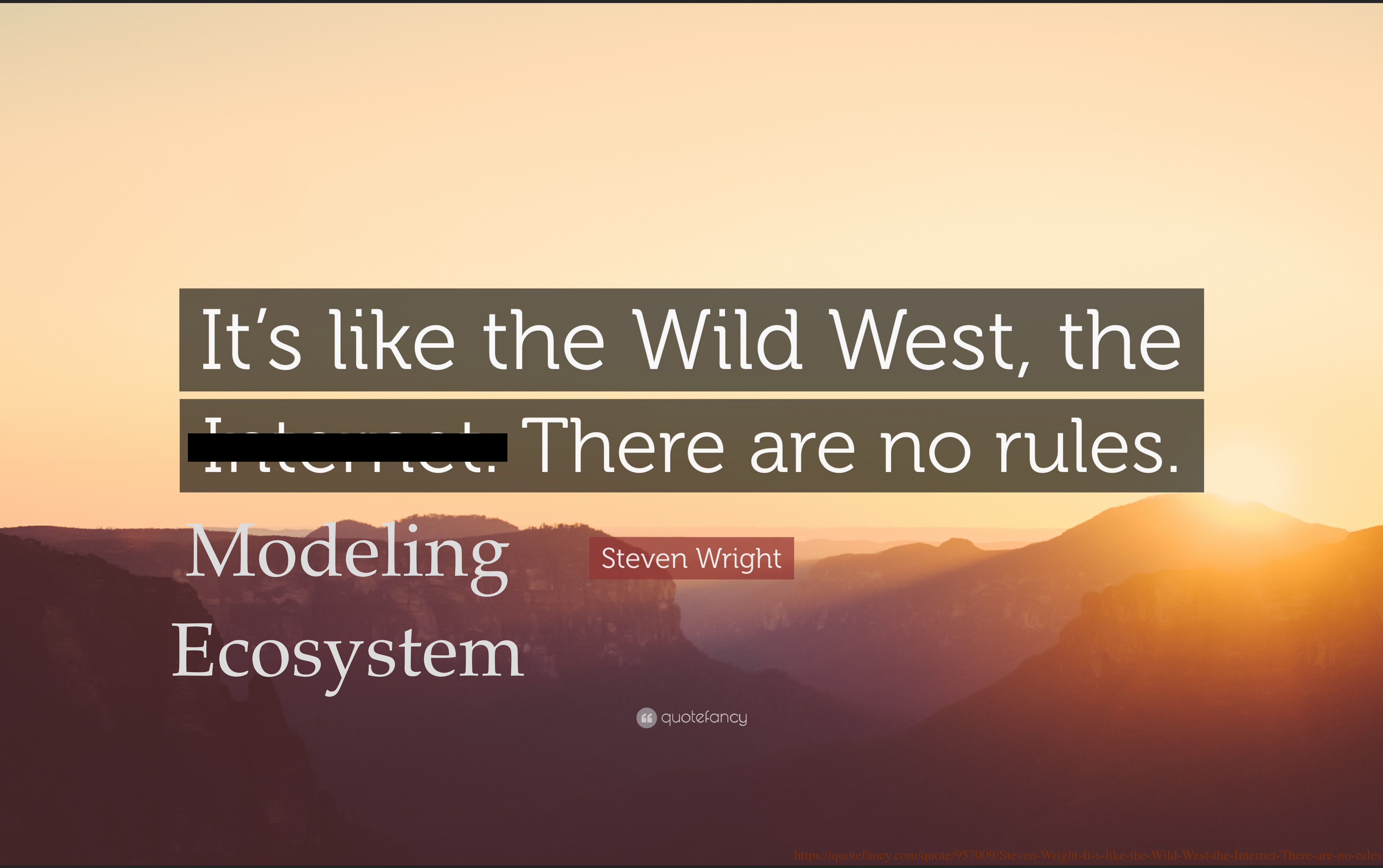
RStudio

The modeling ecosystem

Case study

The R Modeling Ecosystem


It’s like the Wild West, the Internet. There are no rules.
Steven Wright
quotefancy
https://quotefancy.com/quote/957009/Steven-Wright-It-s-like-the-Wild-West-the-Internet-There-are-no-rules


It's like the Wild West, the
[redacted] There are no rules.
Steven Wright
Modeling
Ecosystem
quotefancy

But I thought R was good at modeling?

But I thought R was good at modeling?

It is.

But I thought R was good at modeling?

It is.

But…

A problem of consistency

Function	**Package**	**Code**
`lda`	`MASS`	`predict(obj)`
`glm`	`stats`	`predict(obj, type = "response")`
`gbm`	`gbm`	`predict(obj, type = "response", n.trees)`
`mda`	`mda`	`predict(obj, type = "posterior")`
`rpart`	`rpart`	`predict(obj, type = "prob")`
`Weka`	`RWeka`	`predict(obj, type = "probability")`
`logitboost`	`LogitBoost`	`predict(obj, type = "raw", nIter)`
`pamr.train`	`pamr`	`pamr.predict(obj, type = "posterior", threshold)`

A problem of consistency

Function	Package	Code
[illegible]	MASS	predict(obj)
[illegible]	[illegible]	predict(obj, type = "response")
gbm	[illegible]	[illegible] type = "response", n.trees)
mda	mda	[illegible]ior")
rpart	rpart	predict(obj, [illegible]
Weka	RWeka	predict(obj, type = "probabi[illegible]
logitboost	LogitBoost	predict(obj, type = "raw", nIter)
pamr.train	pamr	pamr.predict(obj, type = "posterior", threshold)

predict(obj, type = “prob”)

tidymodels

```r
library(parsnip)
train <- mtcars[1:20,]
test <- mtcars[21:32,]
```

```r
linear_reg(penalty = .01) %>%
  set_engine("glmnet") %>%
  fit(mpg ~ cyl + disp, train) %>%
  predict(test)
#> # A tibble: 12 x 1
#>    .pred
#>    <dbl>
#>  1  26.5
#>  2  15.1
#>  3  15.5
#>  4  14.4
#>  5  13.2
#>  6  27.5
#>  7  26.5
#>  8  27.1
#>  9  14.4
#> 10  22.6
#> 11  15.6
#> 12  26.5
```

```r
rand_forest(mode = "regression") %>%
  set_engine("ranger") %>%
  fit(mpg ~ cyl + disp, train) %>%
  predict(test)
#> # A tibble: 12 x 1
#>    .pred
#>    <dbl>
#>  1  25.0
#>  2  15.6
#>  3  15.7
#>  4  15.6
#>  5  15.5
#>  6  30.2
#>  7  25.0
#>  8  26.7
#>  9  15.6
#> 10  21.5
#> 11  15.7
#> 12  25.0
```

Lets talk about xgboost

Lets talk about xgboost

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

Lets talk about xgboost

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

https://en.wikipedia.org/wiki/Gradient_boosting

Lets talk about xgboost

Lets talk about
xgboost

Written by C developers as an R
interface to a C library

```r
library(xgboost)
library(dplyr)

outcome <- pull(iris, Species)
predictors <- select(iris, -Species)

head(outcome)
#> [1] setosa setosa setosa setosa setosa setosa
#> Levels: setosa versicolor virginica

head(predictors)
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width
#> 1          5.1         3.5          1.4         0.2
#> 2          4.9         3.0          1.4         0.2
#> 3          4.7         3.2          1.3         0.2
#> 4          4.6         3.1          1.5         0.2
#> 5          5.0         3.6          1.4         0.2
#> 6          5.4         3.9          1.7         0.4
```

```r
model ← xgboost(
  data      = predictors,
  label     = outcome,
  objective = "multi:softprob"
)
```

```r
model <- xgboost(
  data      = predictors,
  label     = outcome,
  objective = "multi:softprob"
)

#> Warning in xgb.get.DMatrix(data, label, missing,
#> weight): xgboost: label will be ignored.

#> Error in xgb.get.DMatrix(data, label, missing,
#> weight): xgboost doesn't support data.frame as
#> input. Convert it to matrix first.
```

```r
model ← xgboost(
  data      = as.matrix(predictors),
  label     = outcome,
  objective = "multi:softprob"
)
```

```r
model <- xgboost(
  data      = as.matrix(predictors),
  label     = outcome,
  objective = "multi:softprob"
)

#> Error in check.booster.params(params, ...):
#> 'num_class' > 1 parameter must be set for
#> multiclass classification
```

```r
model <- xgboost(
  data      = as.matrix(predictors),
  label     = outcome,
  objective = "multi:softprob"
)

#> Error in check.booster.params(params, ...):
#> 'num_class' > 1 parameter must be set for
#> multiclass classification
```

num_class set the number of classes.
To use only with multiclass objectives.

```r
model <- xgboost(
  data      = as.matrix(predictors),
  label     = outcome,
  objective = "multi:softprob",
  num_class = 3
)
```

```r
model <- xgboost(
  data      = as.matrix(predictors),
  label     = outcome,
  objective = "multi:softprob",
  num_class = 3
)

#> Error in xgb.train(params, dtrain, nrounds,
#> watchlist, verbose = verbose, :
#> argument "nrounds" is missing, with no default
```

```r
model <- xgboost(
  data      = as.matrix(predictors),
  label     = outcome,
  objective = "multi:softprob",
  num_class = 3,
  nrounds   = 10
)
```

```r
model <- xgboost(
  data      = as.matrix(predictors),
  label     = outcome,
  objective = "multi:softprob",
  num_class = 3,
  nrounds   = 10
)

#> Error in xgb.iter.update(bst$handle, dtrain,
#> iteration - 1, obj): [12:15:39] amalgamation/../
#> src/objective multiclass_obj.cu:110:
#> SoftmaxMultiClassObj: label must be in
#> [0, num_class).
#>
#> Stack trace returned 10 entries:
#> [bt] (0) 0   xgboost.so
#> 0x000000010f7bb23c dmlc::StackTrace(unsigned long)
#> + 460
#> ..BLAHHHHHHH
```

```r
model <- xgboost(
  data        = as.matrix(predictors),
  label       = outcome,
  objective = "multi:softprob",
  num_class = 3,
  nrounds   = 10
)

#> Error in xgb.iter.update(bst$handle, dtrain,
#> iteration - 1, obj): [12:15:39] amalgamation/../
#> src/objective multiclass_obj.cu:110:
#> SoftmaxMultiClassObj: label must be in
#> [0, num_class).
#>
#> Stack trace returned 10 entries:
#> [bt] (0) 0   xgboost.so
#> 0x000000010f7bb23c dmlc::StackTrace(unsigned long)
#> + 460
#>  ..BLAHHHHHHH
```

```r
model <- xgboost(
  data      = as.matrix(predictors),
  label     = as.numeric(outcome) - 1L,
  objective = "multi:softprob",
  num_class = 3,
  nrounds   = 10
)

#> [1]  train-merror:0.020000
#> [2]  train-merror:0.026667
#> [3]  train-merror:0.020000
#> [4]  train-merror:0.020000
#> [5]  train-merror:0.013333
#> [6]  train-merror:0.013333
#> [7]  train-merror:0.013333
#> [8]  train-merror:0.013333
#> [9]  train-merror:0.013333
#> [10] train-merror:0.013333
```

Constructive feedback 🤔

Why can’t predictors be a data frame?

Why force me to specify num_class?

Why do I always have to set nrounds?

Why can’t outcome be a factor?

Why does outcome have to be a 0-based integer?

Constructive feedback 🤔

Why can’t predictors be a data frame?
Principle: Familiar interface

Why force me to specify num_class?
Principle: Good defaults

Why do I always have to set nrounds?
Principle: Good defaults

Why can’t outcome be a factor?
Principle: Familiar interface

Why does outcome have to be a 0-based integer?
Principle: Familiar interface

```r
predict(model, as.matrix(predictors))

#>   [1] 0.95302665 0.02447842 0.02249493 0.95302665 0.02447842
#>   [6] 0.95302665 0.02447842 0.02249493 0.95302665 0.02447842
#>    ...
```

```r
predict(model, as.matrix(predictors))

#>   [1] 0.95302665 0.02447842 0.02249493 0.95302665 0.02447842
#>   [6] 0.95302665 0.02447842 0.02249493 0.95302665 0.02447842
#>    ...


0.95302665 + 0.02447842 + 0.02249493
#> [1] 1
```

```r
# Or `predict(model, as.matrix(predictors), reshape = TRUE)`
matrix(predict(model, as.matrix(predictors)), ncol = 3)

#>                 [,1]       [,2]       [,3]
#>   [1,] 0.95302665 0.02447842 0.02249493
#>   [2,] 0.95302665 0.02447842 0.02249493
#>   [3,] 0.95302665 0.02447842 0.02249493
#>   [4,] 0.95302665 0.02447842 0.02249493
#>    ..   ..   ..   ..
```

Constructive feedback 🤔

Why can't newdata be a data frame?
Principle: Familiar interface

Why is the default multiclass prob output a vector?
Principle: Predictable output

Note: If factors were allowed, the column names of the output could actually be meaningful!
Principle: User burden

In summary:

“I feel like I’m working for xgboost rather than having xgboost work for me.”

What makes a
modeling package “good”?

What makes a
modeling package “~~good~~”?
intuitive to use

Intuitive modeling packages:

Familiar interfaces

Good defaults

Familiar output types

Standardized arguments

Intuitive modeling packages:

Familiar interfaces

Good defaults

Familiar output types

Standardized arguments

https://tidymodels.github.io/model-implementation-principles/

Intuitive modeling packages:

Think about the R <u>interface</u> in addition to the <u>implementation</u>

Intuitive modeling packages:

Think about the R <ins>interface</ins> in addition to the <ins>implementation</ins>

Why doesn’t everyone do this?

Why doesn't everyone do this?

High barrier of
required knowledge

Why doesn’t everyone do this?

High barrier of required knowledge

Low amount of tooling

Why doesn't everyone do this?

High barrier of required knowledge

Low amount of tooling

Why doesn’t everyone do this?

High barrier of
required knowledge

Low amount
of tooling

hardhat

“A toolkit for the construction of modeling packages”

- Opinionated
- Handles preprocessing
- Robust at prediction time
- S3 method advice
- Validation