Modeling in the Tidyverse

Max Kuhn (RStudio)

Modeling in R

- R has always had a rich set of modeling tools that it inherited from S. For example, the formula interface has made it simple to specify potentially complex model structures.

- *R has cutting edge models*. Many researchers in various domains use R as their primary computing environment and their work often results in R packages.

- *It is easy to port or link to other applications*. R doesn't try to be everything to everyone. If you prefer models implemented in C, C++, `tensorflow`, `keras`, `python`, `stan`, or `Weka`, you can access these applications without leaving R.

However, there is a huge *consistency problem*. For example:

- There are two primary methods for specifying what terms are in a model. Not all models have both.
- 99% of model functions automatically generate dummy variables.
- Sparse matrices can be used (unless the can't).
- Many package developers don't know much about the language and omit OOP and other core R components.

Two examples follow...

Between-Package Inconsistency

Syntax for computing predicted class probabilities:

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")`

What We Need

Unless you are doing a simple one-off data analysis, the lack of consistency between, and sometimes within, R packages can be very frustrating.

If we could agree on a set of common conventions for interfaces, return values, and other components, everyone's life would be easier.

Once we agree on conventions, **two challenges** are:

- As of 8/2018, there are over 12K R packages on CRAN. How do we "harmonize" these without breaking everything?

- How can we guide new R users (or people unfamiliar with R) in making good choices in their modeling packages?

These prospective and retrospective problems will be addressed in a minute.

The Tidyverse

The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

The principles of the tidyverse:

1. Reuse existing data structures.
2. Compose simple functions with the pipe.
3. Embrace functional programming.
4. Design for humans.

This results in more specific conventions around interfaces, function naming, etc. For example:

```
## [1] "glue_col"      "glue_collapse"
## [3] "glue_data"     "glue_data_col"
## [5] "glue_data_sql" "glue_sql"
```

There is also the notion of tidy data:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

Based on these ideas, we can create modeling packages that have predictable results and are a pleasure to use.

Tidymodels

`tidymodels` is a collection of modeling packages that live in the tidyverse and are designed in the same way.

My goals for tidymodels are:

1. Encourage empirical validation and good methodology.

2. Smooth out diverse interfaces.

3. Build highly reusable infrastructure.

4. Enable a wider variety of methodologies.

The `tidymodels` packages address the *retrospective* and *prospective* issues. We are also developing a set of principles and templates to make *prospective* (new R packages) easy to create.

Current Modeling Packages

- `broom` takes the messy output of built-in functions in R, such as `lm`, `nls`, or `t.test`, and turns them into tidy data frames.

- `dials` has tools for creating and validating tuning parameter values.

- `infer` is a modern approach to statistical inference.

- `recipes` is a general data preprocessor with a modern interface. It can create model matrices that incorporate feature engineering, imputation, and other tools.

- `rsample` has infrastructure for *resampling* data so that models can be assessed and empirically validated.

- `tidyposterior` can be used to compare models using resampling and Bayesian analysis.

- `tidytext` contains tidy tools for quantitative text analysis, including basic text summarization, sentiment analysis, and text modeling.

- `yardstick` contains tools for evaluating models (e.g. accuracy, RMSE, etc.)

More on the way... blog post.

Loading the Meta-Package

```
library(tidymodels)
```

```
## ── Attaching packages ──────────────────────────────────────── tidymodels 0.0.1 ──
```

```
## ✔ ggplot2    3.0.0           ✔ recipes    0.1.3.9000
## ✔ tibble     1.4.2           ✔ broom      0.5.0
## ✔ purrr      0.2.5           ✔ yardstick  0.0.1.9000
## ✔ dplyr      0.7.6           ✔ infer      0.3.1
## ✔ rsample    0.0.2
```

```
## ── Conflicts ──────────────────────────────────────── tidymodels_conflicts() ──
## ✖ purrr::accumulate() masks foreach::accumulate()
## ✖ dplyr::collapse()   masks glue::collapse()
## ✖ rsample::fill()     masks tidyr::fill()
## ✖ dplyr::filter()     masks stats::filter()
## ✖ dplyr::lag()        masks stats::lag()
## ✖ recipes::prepper()  masks rsample::prepper()
## ✖ recipes::step()     masks stats::step()
## ✖ purrr::when()       masks foreach::when()
```

broom Example

Model fit from `?lm`

```
summary(lm.D9)$coefficients
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.032      0.220   22.85 9.55e-15
## groupTrt      -0.371      0.311   -1.19 2.49e-01
```

```
broom::tidy(lm.D9)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    5.03      0.220      22.9 9.55e-15
## 2 groupTrt      -0.371     0.311     -1.19 2.49e- 1
```

Find the differences...

Recipes for Preprocessing Data

`recipes` provides a `dplyr`-like utility for

- Defining roles of variables in a model (e.g. outcome, predictor, etc).
- One or more *steps* are specified that do various types of operations, such as centering, imputation, feature extraction, term specification, re-encodings, etc.

Using a recipe is a stage-wise process:

recipe()	{define}
⬇	⬇
prep()	{estimate}
⬇	⬇
bake()/juice()	{apply}

```r
library(caret)
data(BloodBrain)
bbb_data <- bbbDescr %>% mutate(log_ratio = logBBB)

rec <- recipe(log_ratio ~ ., data = bbb_data) %>%
  step_nzv(all_predictors()) %>%
  step_corr(all_predictors(), threshold = 0.75) %>%
  step_YeoJohnson(all_predictors()) %>%
  step_interact(~ nbasic:rotatablebonds) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors()) %>%
  step_pca(all_predictors(), num = 3)

rec <- prep(rec, training = bbb_data)

bake(rec, bbb_data %>% slice(1:3))
```

```
## # A tibble: 3 x 4
##   log_ratio   PC1    PC2    PC3
##       <dbl> <dbl>  <dbl>  <dbl>
## 1      1.08  2.67  -7.88  -1.58
## 2      -0.4 -4.14  -2.15  -2.09
## 3      0.22  1.08   1.86   1.91
```

Evaluating Hypotheses via `infer`

```r
library(caret)
data(BloodBrain)

dat <-
  data.frame(
    mol_weight = bbbDescr$mw,
    log_ratio = logBBB
  )

set.seed(3555)
perms <-
  dat %>%
  specify(log_ratio ~ mol_weight) %>%
  hypothesize(null = "independence") %>%
  generate(reps = 5000, type = "permute") %>%
  calculate(stat = "correlation", method = "spearman")

observed <-
  dat %>%
  specify(log_ratio ~ mol_weight) %>%
  calculate(stat = "correlation", method = "spearman")

perms %>% get_pvalue(obs_stat = observed, direction = "two_sided")
```
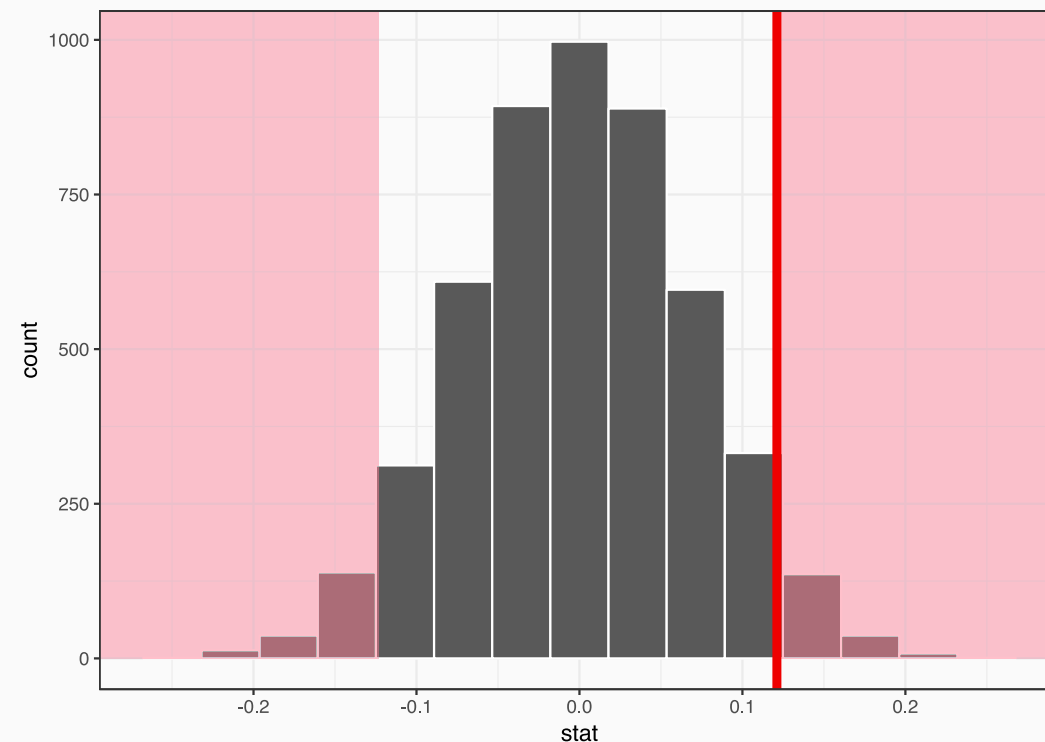
```
## # A tibble: 1 x 1
##   p_value
##     <dbl>
## 1  0.0854
```

```r
perms %>%
  visualize(obs_stat = observed, direction = "two_sided")
```

Principles of Modeling Packages

We are in the process of developing a set of *guidelines* for making good modeling packages. For example:

- Separate the interface that the **modeler** uses from the code to do the computations. They serve two very different purposes.

- Have multiple interfaces (e.g. formula, x/y, etc).

- The *user-facing interface* should use the most appropriate data structures for the data (as opposed to the computations). For example, factor outcomes versus 0/1 indicators and data frames versus matrices.

- `type = "prob"` for class probabilities 😄.

- Use S3 methods.

- The `predict` method should give standardized, predictable results.

Rather than try to make methodologists into software developers, we will provide GitHub repositories with template packages that can be used to meet these guidelines (along with documentation and examples on *why*).

Next Steps

- Hash out the principles of modeling functions. Let me know if you'd like to contribute.

- Packages on the horizon:

  - `parsnip`: a unified interface to models. This should significantly reduce the amount of syntactical minutia that you'll need to memorize by having one standardized model function across different packages and by harmonizing the parameter names across models.

  - `embed`: an add-on package for `recipes`. This can be used to efficiently encode high-cardinality categorical predictors using supervised methods such as likelihood encodings and entity embeddings.

  - A pipeline(ish) structure that can contain specifications for a model, recipe, feature filter, and post-processing. This will easily enable a *data analysis process*.

More details are at `https://github.com/orgs/tidymodels/projects`

Thanks!

Extra Slides

Comparing Models Using Resampling and Bayes

`tidyposter` and `rsample` can be used to make comparisons between and within types of models.

A model is resampled and it's performance metrics (e.g. R^2, ROC AUC, etc.) can be used as the *outcome* in a Bayesian meta-model. From this, posteriors for the differences can be computed.

Let's say that I have these two models:

```
coxph(Surv(time, status) ~ ph.ecog + age + sex, data = lung)
coxph(Surv(time, status) ~ ph.ecog + age,       data = lung)
```
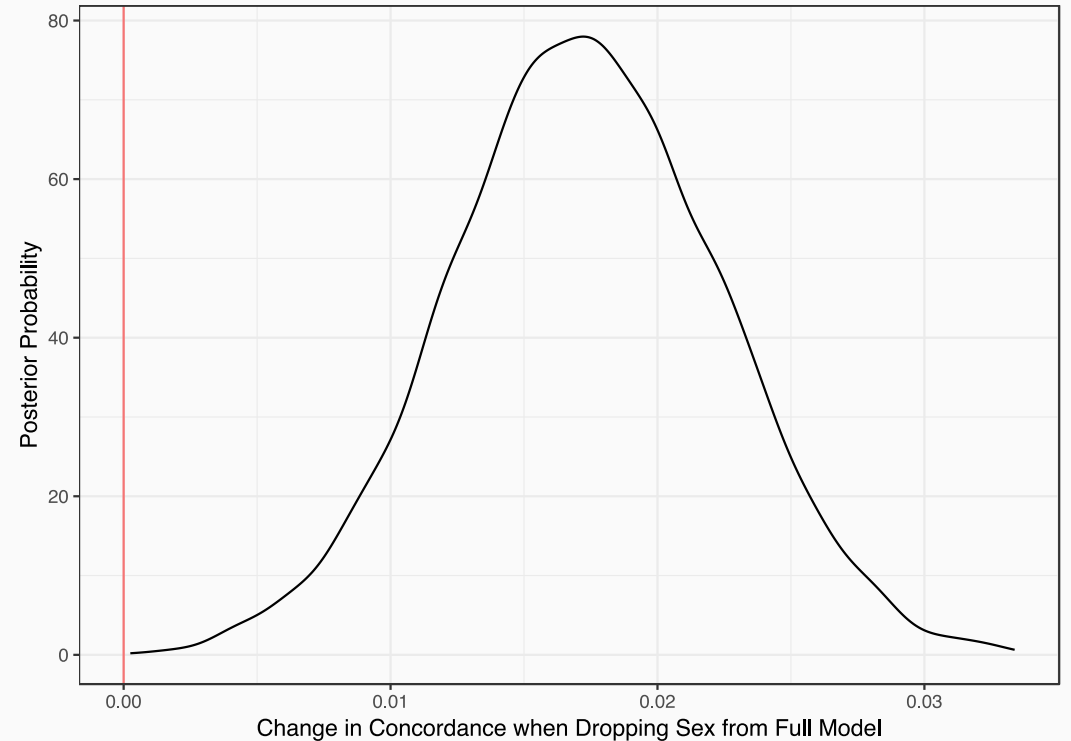
I can compare them using a standard hierarchical model comparison or a simple Wald-type test. On this case, the Wald p-value is 0.00099. That doesn't tell me much about the *effect size*.

Suppose I look at the change in the standard **concordance statistic** between the two models? What is the distribution of the *change in concordance* when I remove `sex`?

Cross-validation can be used to compute this difference on out-of-sample data and a Bayesian model can be used to compute the posterior for the difference.

Results

```
## #  10-fold cross-validation repeated 10 times using stratif
## # A tibble: 100 x 5
##    id       id2     full_model without_sex diff
##  * <chr>    <chr>        <dbl>       <dbl> <chr>
##  1 Repeat01 Fold01       0.701       0.687 worse  by 0.01
##  2 Repeat01 Fold02       0.570       0.554 worse  by 0.02
##  3 Repeat01 Fold03       0.731       0.702 worse  by 0.03
##  4 Repeat01 Fold04       0.599       0.550 worse  by 0.05
##  5 Repeat01 Fold05       0.594       0.528 worse  by 0.07
##  6 Repeat01 Fold06       0.742       0.663 worse  by 0.08
##  7 Repeat01 Fold07       0.525       0.514 worse  by 0.01
##  8 Repeat01 Fold08       0.714       0.674 worse  by 0.04
##  9 Repeat01 Fold09       0.491       0.549 better by 0.06
## 10 Repeat01 Fold10       0.764       0.692 worse  by 0.07
## # ... with 90 more rows
```


A positive difference would imply that `sex` is important in explaining the outcome.