Probability theory can be fun and simple with dependent types (Yet another formal theory of probabilities in Coq)

Reynald Affeldt, Alessandro Bruni, Pierre Roux, Takafumi Saikawa

Workshop on Alignment of Proof Systems and Machine Learning
Vienna, Austria, March 25th-26th 2024

The Coq proof assistant

- Dependently typed programming language/proof checker
- Notable (historical) uses:
  - CompCert, a verified C compiler [Leroy et al., 2016]
  - Four Color Theorem [Gonthier, 2005]
  - Odd Order Theorem [Gonthier et al., 2013]
- More recent (and relevant) uses:
  - Verified perceptrons [Bagnall and Stewart, 2019]
  - Probabilistic languages with applications to machine learning [Tassarotti and Tristan, 2023]

An overview of existing formalizations in Coq

InfoTheo [Infotheo, 2018]

- Formalizes *finite probabilities*; used for information theory, error-correcting codes, robust statistics

coq-proba [Tassarotti, 2023]

- Used to *reason about programs*

FormalML [The FormalML development team, 2023]

- Contains *advanced theorems* in probability theory

MathComp-Analysis [Affeldt et al., 2024a]

- Contains a rich formalization of *measure theory* and *Lebesgue integrals* to build upon

A proof engineering effort

Mathematical Components

29 followers https://math-comp.github.io/math-...

math-comp Public
Mathematical Components
Coq 541 109 98 34 Updated 7 minutes ago

analysis Public
Mathematical Components compliant Analysis Library
Coq 176 40 72 (1 issue needs help) 38 Updated 2 hours ago

real-closed Public
Theorems for Real Closed Fields
Coq 12 10 5 3 Updated 5 hours ago

hierarchy-builder Public
High level commands to declare a hierarchy based on packed classes
Prolog 90 MIT 19 64 13 Updated 10 hours ago

Motivation

Recent developments in MathComp-Analysis

- Asymptotic reasoning + Landau notations → differentiability [Affeldt et al., 2020]
- Lebesgue integrals (2021-2023)
- Fundamental theorem of calculus (2023)
- Probability theory (2023)

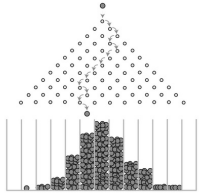
Applications

- Verified probabilistic programming languages [Affeldt et al., 2023, Saito and Affeldt, 2023]
- Verified worst-case failure probability of real-time systems [Markovic et al., 2023]
- *Future*: verified robust statistics [Daukantas et al., 2021, Affeldt et al., 2024b]
- *Future*: verified machine learning [Affeldt et al., 2024c]

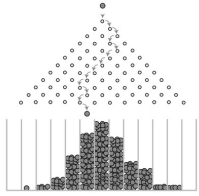
An example: Bernoulli sampling [Rajani, 2019]

Theorem (Bernoulli sampling)

Given independent 0-1 random variables X_i, with $X = \sum_{i=1}^{n} X_i$, $Pr(X_i = 1) = p$, and $\bar{X} = \frac{X}{n}$, if $n \geq \frac{3}{\theta^2} \ln(\frac{2}{\delta})$, then $Pr(|\bar{X} - p| \leq \theta) \geq 1 - \delta$.

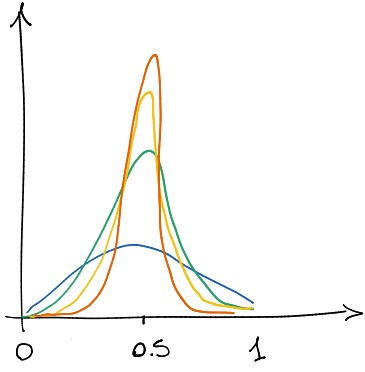


An example: Bernoulli sampling [Rajani, 2019]



Theorem (Bernoulli sampling)

Given independent 0-1 random variables X_i*, with* $X = \sum_{i=1}^{n} X_i$*,* $Pr(X_i = 1) = p$*, and* $\bar{X} = \frac{X}{n}$*, if* $n \geq \frac{3}{\theta^2} \ln(\frac{2}{\delta})$*, then* $Pr(|\bar{X} - p| \leq \theta) \geq 1 - \delta$.

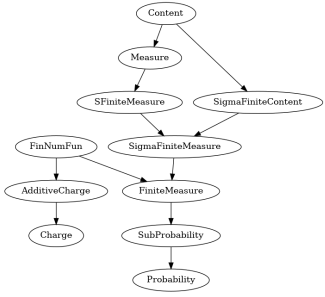


Simple and general: inherit from measure theory with Hierarchy Builder

Definition (Measure)

A measure $\mu : T \to \overline{\mathbb{R}}$ satisfies:

1. $\mu(\emptyset) = 0$ (measure-0)
2. $0 \leq \mu(A)$ for any set A (non-negativity)
3. $\mu(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i)$ (σ-additivity)

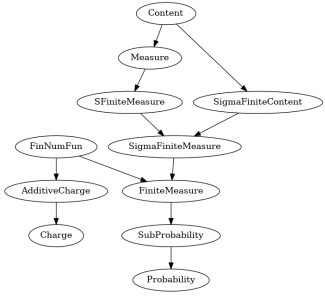


Simple and general: inherit from measure theory with Hierarchy Builder

Definition (Measure)

A measure $\mu : T \to \overline{\mathbb{R}}$ satisfies:

1. $\mu(\emptyset) = 0$ (measure-0)
2. $0 \leq \mu(A)$ for any set A (non-negativity)
3. $\mu(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i)$ (σ-additivity)



Definition (Probability measure)

A probability measure additionally implements the following interface:

```
HB.mixin Record isProbability d (T : measurableType d)
  (R : realType) (P : set T -> \bar R) :=
    { probability_setT : P setT = 1 }.
```

…and fun: random variables and expectations

```
Context d (T : measurableType d) (R : realType) (P : probability T R).
```

Definition (Random variables)

A random variable is neither random, nor a variable. It's a measurable function from `T` to `R`.

```
Definition random_variable := {mfun T >-> R}.
Notation "{ 'RV' P >-> R }" := (@random_variable _ _ R P).
```

…and fun: random variables and expectations

```
Context d (T : measurableType d) (R : realType) (P : probability T R).
```

Definition (Random variables)

A random variable is neither random, nor a variable. It's a measurable function from `T` to `R`.

```
Definition random_variable := {mfun T >-> R}.
Notation "{ 'RV' P >-> R }" := (@random_variable _ _ R P).
```

Definition (Expectation)

Expectation of X with the measure P can be expressed as the Lebesgue integral $\int X \, dP$:

```
Definition expectation (X : {RV P >-> R}) := \int[P]_w (X w)%:E.
```

Recovering discreteness

Discrete (random) variables

Discrete random variables additionally implement the following interface:

```
HB.mixin Record MeasurableFun_isDiscrete d (T : measurableType d) (R : realType)
  (X : T -> R) of @MeasurableFun d T R X := { countable_range : countable (range X) }.
```

Recovering discreteness

Discrete (random) variables

Discrete random variables additionally implement the following interface:

```
HB.mixin Record MeasurableFun_isDiscrete d (T : measurableType d) (R : realType)
  (X : T -> R) of @MeasurableFun d T R X := { countable_range : countable (range X) }.
```

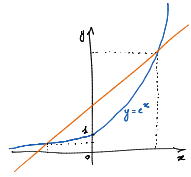
Discrete sums

When `X : {RV P >-> R}` (the type of discrete random variables), we build a function `dRV_enum` to enumerate its values a_k, and a function `enum_prob` to enumerate the probabilities p_k, so that the distribution can be written as $\sum_k c_k \delta_{a_k}$, where δ is the Dirac measure. In Coq this becomes:

```
Lemma distribution_dRV A : measurable A ->
  distribution P X A = \sum_(k <oo) enum_prob X k * \d_(dRV_enum X k) A.
```

(More) formal adventures in convex spaces

[Infotheo, 2018] shows that probability theory benefits from a theory of convex spaces:



Convex function

```
Definition convex_function (R : realType) (D : set R) (f : R -> R) :=
  forall t : {i01 R}, {in D &, forall (x y : R), f (x <| t |> y) <= f x <| t |> f y}.
```

Exponentials are convex

```
Lemma convex_expR : convex_function setT expR.
Lemma convex_powR p : 1 <= p -> convex_function `[0, +oo[ (fun x : R => powR x p).
```

Moments: exponential expectations

```
Definition mmt_gen_fun (X : {RV P >-> R}) (t : R) := 'E_P[expR \o t \o* X].
```

Applications of convexity: Hölder and Minkowski and L_p -spaces

We are building a theory of L_p -spaces. For that purpose we prove Hölder's and Minkowski's inequalities, which are also generally applicable to probabilities:

Hölder

```
Lemma hoelder (f g : T -> R) (p q : R) : measurable_fun setT f -> measurable_fun setT g ->
  0 < p -> 0 < q -> p^-1 + q^-1 = 1 ->
    'N_1 [f \* g] <= 'N_p [f] * 'N_q [g].
```

(Here `\+` and `\*` are pointwise addition and multiplication, and `N_p [f]` is the p-norm of `f`)

Applications of convexity: Hölder and Minkowski and L_p -spaces

We are building a theory of L_p -spaces. For that purpose we prove Hölder's and Minkowski's inequalities, which are also generally applicable to probabilities:

Hölder

```
Lemma hoelder (f g : T -> R) (p q : R) : measurable_fun setT f -> measurable_fun setT g ->
  0 < p -> 0 < q -> p^-1 + q^-1 = 1 ->
    'N_1 [f \* g] <= 'N_p [f] * 'N_q [g].
```

Minkowski

```
Lemma minkowski f g p : measurable_fun setT f -> measurable_fun setT g -> 1 <= p ->
  'N_p%:E[f \+ g] <= 'N_p%:E[f] + 'N_p%:E[g].
```

(Here `\+` and `\*` are pointwise addition and multiplication, and `N_p [f]` is the p-norm of `f`)

More useful lemmas: Markov, Chernoff, Chebyshev and Cantelli

```
Lemma markov (X : {RV P >-> R}) (f : R -> R) (eps : R) : (0 < eps) ->
    measurable_fun [set: R] f -> (forall r, 0 <= r -> 0 <= f r) ->
    {in Num.nneg &, {homo f : x y / x <= y}} ->
  (f eps)%:E * P [set x | eps%:E <= `| (X x)%:E | ] <=
    'E_P[f \o (fun x => `| x |) \o X].

Lemma chernoff (X : {RV P >-> R}) (r a : R) : (0 < r) ->
  P [set x | X x >= a] <= mmt_gen_fun X r * (expR (- (r * a)))%:E.

Lemma chebyshev (X : {RV P >-> R}) (eps : R) : (0 < eps) ->
  P [set x | (eps <= `| X x - fine ('E_P[X])|) ] <= (eps ^- 2)%:E * 'V_P[X].

Lemma cantelli (X : {RV P >-> R}) (lambda : R) :
    P.-integrable setT (EFin \o X) -> P.-integrable setT (EFin \o (X ^+ 2)) ->
    (0 < lambda) ->
  P [set x | lambda%:E <= (X x)%:E - 'E_P[X]] <=
    (fine 'V_P[X] / (fine 'V_P[X] + lambda^2))%:E.
```

Our experiment: Bernoulli sampling [Rajani, 2019]

Theorem

Given independent 0-1 random variables X_i, with $X = \sum_{i=1}^{n} X_i$, $Pr(X_i = 1) = p$, and $\bar{X} = \frac{X}{n}$, if $n \geq \frac{3}{\theta^2} \ln(\frac{2}{\delta})$, then $Pr(|\bar{X} - p| \leq \theta) \geq 1 - \delta$.

becomes:

```
Theorem sampling (X_ : seq {RV P >-> R}) (theta delta p : R) :
  let n := size X_ in let X' x := ((\sum_(Xi in X_) Xi) x) / n%:R in
  is_bernoulli_trial X n -> 0 < p <= 1 -> 0 < delta <= 1 ->
  0 < theta < p -> 0 < n -> 3 / theta^+2 * ln(2 / delta) <= n%:R
  -> P [set i | `| X' i - p | <= theta] >= 1 - delta%:E.
```

Bibliography I

Reynald Affeldt, Cyril Cohen, Marie Kerjean, Assia Mahboubi, Damien Rouhling, and Kazuhiko Sakaguchi. Competing inheritance paths in dependent type theory: A case study in functional analysis. In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part II*, volume 12167 of *Lecture Notes in Computer Science*, pages 3–20. Springer, 2020. doi: 10.1007/978-3-030-51054-1_1. URL `https://doi.org/10.1007/978-3-030-51054-1_1`.

Reynald Affeldt, Cyril Cohen, and Ayumu Saito. Semantics of probabilistic programs using s-finite kernels in Coq. In *12th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2023), Boston, MA, USA, January 16–17, 2023*, pages 3–16. ACM, 2023. doi: 10.1145/3573105.3575691. URL `https://doi.org/10.1145/3573105.3575691`.

Reynald Affeldt, Yves Bertot, Alessandro Bruni, Cyril Cohen, Marie Kerjean, Assia Mahboubi, Damien Rouhling, Pierre Roux, Kazuhiko Sakaguchi, Zachary Stone, Pierre-Yves Strub, and Laurent Théry. MathComp-Analysis: Mathematical components compliant analysis library. `https://github.com/math-comp/analysis`, 2024a. Since 2017. Version 1.0.0.

Reynald Affeldt, Alessandro Bruni, Clark Barrett, Ieva Daukantas, Harun Khan, Takafumi Saikawa, and Carsten Schürmann. Robust mean estimation by all means. 2024b. Under submission.

Reynald Affeldt, Alessandro Bruni, Ekaterina Komendantskaya, Natalia Ślusarz, and Kathrin Stark. Taming differentiable logics with coq formalisation. 2024c. Under submission.

Bibliography II

Alexander Bagnall and Gordon Stewart. Certifying the true error: Machine learning in coq with verified generalization guarantees. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 2662–2669. AAAI Press, 2019. doi: 10.1609/AAAI.V33I01.33012662. URL `https://doi.org/10.1609/aaai.v33i01.33012662`.

Ieva Daukantas, Alessandro Bruni, and Carsten Schürmann. Trimming data sets: a verified algorithm for robust mean estimation. In *23rd International Symposium on Principles and Practice of Declarative Programming (PPDP 2021), Tallinn, Estonia, September 6–8, 2021*, pages 17:1–17:9. ACM, 2021. doi: 10.1145/3479394.3479412. URL `https://doi.org/10.1145/3479394.3479412`.

Georges Gonthier. A computer-checked proof of the Four Color Theorem. Technical report, Inria, March 2005. URL `https://inria.hal.science/hal-04034866`.

Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, François Garillot, Stéphane Le Roux, Assia Mahboubi, Russell O'Connor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Théry. A machine-checked proof of the odd order theorem. In *4th International Conference on Interactive Theorem Proving (ITP 2013), Rennes, France, July 22–26, 2013*, volume 7998 of *Lecture Notes in Computer Science*, pages 163–179. Springer, 2013. doi: 10.1007/978-3-642-39634-2_14. URL `https://doi.org/10.1007/978-3-642-39634-2_14`.

Bibliography III

Infotheo. Infotheo: A Coq formalization of information theory and linear error-correcting codes. `https://github.com/affeldt-aist/infotheo`, 2018. Authors: Reynald Affeldt, Manabu Hagiwara, Jonas Sénizergues, Jacques Garrigue, Kazuhiko Sakaguchi, Taku Asai, Takafumi Saikawa, and Naruomi Obata. Last stable release: 0.6.1 (2023).

Xavier Leroy, Sandrine Blazy, Daniel Kästner, Bernhard Schommer, Markus Pister, and Christian Ferdinand. CompCert - A Formally Verified Optimizing Compiler. In *ERTS 2016: Embedded Real Time Software and Systems, 8th European Congress*, Toulouse, France, January 2016. SEE. URL `https://inria.hal.science/hal-01238879`.

Filip Markovic, Pierre Roux, Sergey Bozhko, Alessandro V. Papadopoulos, and Björn B. Brandenburg. CTA: A correlation-tolerant analysis of the deadline-failure probability of dependent tasks. In *IEEE Real-Time Systems Symposium, RTSS 2023, Taipei, Taiwan, December 5-8, 2023*, pages 317–330. IEEE, 2023. doi: 10.1109/RTSS59052.2023.00035. URL `https://doi.org/10.1109/RTSS59052.2023.00035`.

Samir Rajani. Applications of chernoff bounds. `http://math.uchicago.edu/~may/REU2019/REUPapers/Rajani.pdf`, 2019. The University of Chicago Mathematics REU 2019.

Ayumu Saito and Reynald Affeldt. Experimenting with an intrinsically-typed probabilistic programming language in coq. In *21st Asian Symposium on Programming Languages and Systems (APLAS 2023), Taipei, Taiwan, November 26–29, 2023*, volume 14405 of *Lecture Notes in Computer Science*, pages 182–202. Springer, 2023. doi: 10.1007/978-981-99-8311-7_9. URL `https://doi.org/10.1007/978-981-99-8311-7_9`.

Bibliography IV

Joseph Tassarotti. A probability theory library for the Coq theorem prover. `https://github.com/jtassarotti/coq-proba`, 2023. Since 2020.

Joseph Tassarotti and Jean-Baptiste Tristan. Verified density compilation for a probabilistic programming language. *Proc. ACM Program. Lang.*, 7(PLDI):615–637, 2023. doi: 10.1145/3591245. URL `https://doi.org/10.1145/3591245`.

The FormalML development team. FormalML: Formalization of machine learning theory with applications to program synthesis. `https://github.com/IBM/FormalML`, 2023. Since 2019.