Concentration Inequalities

Emin Karayel and Yong Kiam Tan*

March 17, 2025

Abstract

Concentration inequalities provide bounds on how a random variable (or a sum/composition of random variables) deviate from their expectation, usually based on moments/independence of the variables.

The most important concentration inequalities (the Markov, Chebyshev, and Hoelder inequalities and the Chernoff bounds) are already part of HOL-Probability. This entry collects more advanced results, such as Bennett's/Bernstein's Inequality, Bienaymé's Identity, Cantelli's Inequality, the Efron-Stein Inequality, McDiarmid's Inequality, and the Paley-Zygmund Inequality.

Contents

1 Preliminary results

theory *Concentration-Inequalities-Preliminary*
imports *Lp.Lp*
begin

Version of Cauchy-Schwartz for the Lebesgue integral:

*The authors contributed equally to this work.

lemma *cauchy-schwartz*:
fixes *f g* :: - ⇒ *real*
assumes *f* ∈ *borel-measurable M g* ∈ *borel-measurable M*
assumes *integrable M* (λ*x*. (*f x*) ^2) *integrable M* (λ*x*. (*g x*) ^2)
shows *integrable M* (λ*x*. *f x* ∗ *g x*) (**is** *?A*)
(∫ *x*. *f x* ∗ *g x* ∂*M*) ≤ (∫ *x*. (*f x*)^2 ∂*M*) *powr* (*1*/*2*) ∗ (∫ *x*. (*g x*)^ *2* ∂*M*) *powr* (*1*/*2*)
(**is** *?L* ≤ *?R*)
proof −
show *0*:*?A*
using *assms* **by** (*intro Holder-inequality*(*1*)[**where** *p*=*2* **and** *q*=*2*]) *auto*

have *?L* ≤ (∫ *x*. |*f x* ∗ *g x*| ∂*M*)
using *0* **by** (*intro integral-mono*) *auto*
also have ... ≤ (∫ *x*. |*f x*| *powr* *2* ∂*M*) *powr* (*1*/*2*) ∗ (∫ *x*. |*g x*| *powr* *2* ∂*M*) *powr* (*1*/*2*)
using *assms* **by** (*intro Holder-inequality*(*2*)) *auto*
also have ... = *?R* **by** *simp*
finally show *?L* ≤ *?R* **by** *simp*
qed

Generalization of *prob-space.indep-vars-iff-distr-eq-PiM′*:

lemma (**in** *prob-space*) *indep-vars-iff-distr-eq-PiM″*:
fixes *I* :: *′i set* **and** *X* :: *′i* ⇒ *′a* ⇒ *′b*
assumes *rv*: ⋀*i*. *i* ∈ *I* ⟹ *random-variable* (*M′ i*) (*X i*)
shows *indep-vars M′ X I* ⟷
distr M (Π*M* *i*∈*I*. *M′ i*) (λ*x*. λ*i*∈*I*. *X i x*) = (Π*M* *i*∈*I*. *distr M* (*M′ i*) (*X i*))
proof (*cases I* = {})
case *True*
have *0*: *indicator A* (λ-. *undefined*) = *emeasure* (*count-space* {λ-. *undefined*}) *A* (**is** *?L* = *?R*)
if *A* ⊆ {λ-. *undefined*} **for** *A* :: (*′i* ⇒ *′b*) *set*
proof −
have *1*:*A* ≠ {} ⟹ *A* = {λ-. *undefined*}
using *that* **by** *auto*

have *?R* = *of-nat* (*card A*)
using *finite-subset that* **by** (*intro emeasure-count-space-finite that*) *auto*
also have ... = *?L*
using *1* **by** (*cases A* = {}) *auto*
finally show *?thesis* **by** *simp*
qed

have *distr M* (Π*M* *i*∈*I*. *M′ i*) (λ*x*. λ*i*∈*I*. *X i x*) =
distr M (*count-space* {λ-. *undefined*}) (λ-. (λ-. *undefined*))
unfolding *True PiM-empty* **by** (*intro distr-cong*) (*auto simp*:*restrict-def*)

```
  also have ... = return (count-space {λ-. undefined}) (λ-. undefined)
    by (intro distr-const) auto
  also have ... = count-space ({λ-. undefined} :: ('i ⇒ 'b) set)
    by (intro measure-eqI) (auto simp:0)
  also have ... = (Π_M i∈I. distr M (M' i) (X i))
    unfolding True PiM-empty by simp
  finally have distr M (Π_M i∈I. M' i) (λx. λi∈I. X i x)=(Π_M i∈I.
distr M (M' i) (X i)) ⟷ True
    by simp
  also have ... ⟷ indep-vars M' X I
   unfolding indep-vars-def by (auto simp add: space-PiM indep-sets-def)
(auto simp add:True)
  finally show ?thesis by simp
next
  case False
  thus ?thesis
    by (intro indep-vars-iff-distr-eq-PiM' assms) auto
qed

lemma proj-indep:
  assumes ⋀i. i ∈ I ⟹ prob-space (M i)
  shows prob-space.indep-vars (PiM I M) M (λi ω. ω i) I
proof −
  interpret prob-space (PiM I M)
    by (intro prob-space-PiM assms)

  have distr (Pi_M I M) (Pi_M I M) (λx. restrict x I) = PiM I M
    by (intro distr-PiM-reindex assms) auto
  also have ... =  Pi_M I (λi. distr (Pi_M I M) (M i) (λω. ω i))
    by (intro PiM-cong refl distr-PiM-component[symmetric] assms)
  finally have
    distr (Pi_M I M) (Pi_M I M) (λx. restrict x I) = Pi_M I (λi. distr
(Pi_M I M) (M i) (λω. ω i))
    by simp
  thus indep-vars M (λi ω. ω i) I
    by (intro iffD2[OF indep-vars-iff-distr-eq-PiM'']) simp-all
qed

lemma forall-Pi-to-PiE:
  assumes ⋀x. P x = P (restrict x I)
  shows (∀ x ∈ Pi I A. P x) = (∀ x ∈ PiE I A. P x)
  using assms by (simp add:PiE-def Pi-def set-eq-iff, force)

lemma PiE-reindex:
  assumes inj-on f I
  shows PiE I (A ∘ f) = (λa. restrict (a ∘ f) I) ‘ PiE (f ‘ I) A (is
?lhs = ?g ‘ ?rhs)
proof −
  have ?lhs ⊆ ?g‘ ?rhs
```

proof (*rule subsetI*)
fix x
assume a:$x \in Pi_E\ I\ (A \circ f)$
define y **where** y-def: $y = (\lambda k.\ if\ k \in f\ `\ I\ then\ x\ (the\text{-}inv\text{-}into\ I\ f\ k)\ else\ undefined)$
have b:$y \in PiE\ (f\ `\ I)\ A$
using a *assms the-inv-into-f-eq*[*OF assms*]
by (*simp add*: y-*def PiE-iff extensional-def*)
have c: $x = (\lambda a.\ restrict\ (a \circ f)\ I)\ y$
using a *assms the-inv-into-f-eq extensional-arb*
by (*intro ext*, *simp add*:y-*def PiE-iff*, *fastforce*)
show $x \in ?g\ `\ ?rhs$ **using** b c **by** *blast*
qed
moreover have $?g\ `\ ?rhs \subseteq ?lhs$
by (*rule image-subsetI*, *simp add*:*Pi-def PiE-def*)
ultimately show *?thesis* **by** *blast*
qed

context *prob-space*
begin

lemma *indep-sets-reindex*:
assumes *inj-on f I*
shows *indep-sets A* $(f\ `\ I) =$ *indep-sets* $(\lambda i.\ A\ (f\ i))\ I$
proof −
have a: $\bigwedge J\ g.\ J \subseteq I \Longrightarrow (\prod j \in f\ `\ J.\ g\ j) = (\prod j \in J.\ g\ (f\ j))$
by (*metis assms prod.reindex-cong subset-inj-on*)

have b:$J \subseteq I \Longrightarrow (\Pi_E\ i \in J.\ A\ (f\ i)) = (\lambda a.\ restrict\ (a \circ f)\ J)\ `\ PiE\ (f\ `\ J)\ A$ **for** J
using *assms inj-on-subset*
by (*subst PiE-reindex*[*symmetric*]) *auto*

have c:$\bigwedge J.\ J \subseteq I \Longrightarrow$ *finite* $(f\ `\ J) =$ *finite* J
by (*meson assms finite-image-iff inj-on-subset*)

show *?thesis*
by (*simp add*:*indep-sets-def all-subset-image a c*) (*simp-all add*:*forall-Pi-to-PiE b*)
qed

lemma *indep-vars-reindex*:
assumes *inj-on f I*
assumes *indep-vars* $M'\ X'\ (f\ `\ I)$
shows *indep-vars* $(M' \circ f)\ (\lambda k\ \omega.\ X'\ (f\ k)\ \omega)\ I$
using *assms* **by** (*simp add*:*indep-vars-def2 indep-sets-reindex*)

lemma *indep-vars-cong-AE*:
assumes $AE\ x\ in\ M.\ (\forall i \in I.\ X'\ i\ x = Y'\ i\ x)$

assumes *indep-vars M′ X′ I*
assumes $\bigwedge i.\ i \in I \Longrightarrow$ *random-variable* (*M′ i*) (*Y′ i*)
shows *indep-vars M′ Y′ I*
proof −
have *a*: *AE x in M*. (λ*i*∈*I*. *Y′ i x*) = (λ*i*∈*I*. *X′ i x*)
by (*rule AE-mp*[*OF assms*(*1*)], *rule AE-I2*, *simp cong*:*restrict-cong*)
have *b*: $\bigwedge i.\ i \in I \Longrightarrow$ *random-variable* (*M′ i*) (*X′ i*)
using *assms*(*2*) **by** (*simp add*:*indep-vars-def2*)
have *c*: $\bigwedge x.\ x \in I \Longrightarrow$ *AE xa in M*. *X′ x xa* = *Y′ x xa*
by (*rule AE-mp*[*OF assms*(*1*)], *rule AE-I2*, *simp*)

have *distr M* (*Pi*$_M$ *I M′*) (λ*x*. λ*i*∈*I*. *Y′ i x*) = *distr M* (*Pi*$_M$ *I M′*) (λ*x*. λ*i*∈*I*. *X′ i x*)
by (*intro distr-cong-AE measurable-restrict a b assms*(*3*)) *auto*
also have ... = *Pi*$_M$ *I* (λ*i*. *distr M* (*M′ i*) (*X′ i*))
using *assms b* **by** (*subst indep-vars-iff-distr-eq-PiM′′*[*symmetric*]) *auto*
also have ... = *Pi*$_M$ *I* (λ*i*. *distr M* (*M′ i*) (*Y′ i*))
by (*intro PiM-cong distr-cong-AE c assms*(*3*) *b*) *auto*
finally have *distr M* (*Pi*$_M$ *I M′*) (λ*x*. λ*i*∈*I*. *Y′ i x*) = *Pi*$_M$ *I* (λ*i*. *distr M* (*M′ i*) (*Y′ i*))
by *simp*

thus *?thesis*
using *assms*(*3*)
by (*subst indep-vars-iff-distr-eq-PiM′′*) *auto*
qed

end

Integrability of bounded functions on finite measure spaces:

lemma *bounded-const*: *bounded* ((λ*x*. (*c*::*real*)) ‘ *T*)
by (*intro boundedI*[**where** *B*=*norm c*]) *auto*

lemma *bounded-exp*:
fixes *f* :: *′a* ⇒ *real*
assumes *bounded* ((λ*x*. *f x*) ‘ *T*)
shows *bounded* ((λ*x*. *exp* (*f x*)) ‘ *T*)
proof −
obtain *m* **where** *norm* (*f x*) ≤ *m* **if** *x* ∈ *T* **for** *x*
using *assms* **unfolding** *bounded-iff* **by** *auto*

thus *?thesis*
by (*intro boundedI*[**where** *B*=*exp m*]) *fastforce*
qed

lemma *bounded-mult-comp*:
fixes *f* :: *′a* ⇒ *real*
assumes *bounded* (*f* ‘ *T*) *bounded* (*g* ‘ *T*)

shows *bounded* $((\lambda x.\ (f\ x) * (g\ x))\ `\ T)$
proof −
obtain *m1* **where** *norm* $(f\ x) \leq m1\ m1 \geq 0$ **if** $x \in T$ **for** x
using *assms* **unfolding** *bounded-iff* **by** *fastforce*
moreover obtain *m2* **where** *norm* $(g\ x) \leq m2\ m2 \geq 0$ **if** $x \in T$ **for** x
using *assms* **unfolding** *bounded-iff* **by** *fastforce*

ultimately show *?thesis*
by (*intro boundedI*[**where** *B*=*m1* * *m2*]) (*auto intro*!: *mult-mono simp*:*abs-mult*)
qed

lemma *bounded-sum*:
fixes $f :: 'i \Rightarrow 'a \Rightarrow real$
assumes *finite I*
assumes $\bigwedge i.\ i \in I \Longrightarrow bounded\ (f\ i\ `\ T)$
shows *bounded* $((\lambda x.\ (\sum i \in I.\ f\ i\ x))\ `\ T)$
using *assms* **by** (*induction I*) (*auto intro*:*bounded-plus-comp bounded-const*)

lemma *bounded-pow*:
fixes $f :: 'a \Rightarrow real$
assumes *bounded* $((\lambda x.\ f\ x)\ `\ T)$
shows *bounded* $((\lambda x.\ (f\ x)\hat{}n)\ `\ T)$
proof −
obtain *m* **where** *norm* $(f\ x) \leq m$ **if** $x \in T$ **for** x
using *assms* **unfolding** *bounded-iff* **by** *auto*
hence *norm* $((f\ x)\hat{}n) \leq m\hat{}n$ **if** $x \in T$ **for** x
using *that* **unfolding** *norm-power* **by** (*intro power-mono*) *auto*
thus *?thesis* **by** (*intro boundedI*[**where** *B*=$m\hat{}n$]) *auto*
qed

lemma *bounded-sum-list*:
fixes $f :: 'i \Rightarrow 'a \Rightarrow real$
assumes $\bigwedge y.\ y \in set\ ys \Longrightarrow bounded\ (f\ y\ `\ T)$
shows *bounded* $((\lambda x.\ (\sum y \leftarrow ys.\ f\ y\ x))\ `\ T)$
using *assms* **by** (*induction ys*) (*auto intro*:*bounded-plus-comp bounded-const*)

lemma (**in** *finite-measure*) *bounded-int*:
fixes $f :: 'i \Rightarrow 'a \Rightarrow real$
assumes *bounded* $((\lambda\ x.\ f\ (fst\ x)\ (snd\ x))\ `\ (T \times space\ M))$
shows *bounded* $((\lambda x.\ (\int \omega.\ (f\ x\ \omega)\ \partial M))\ `\ T)$
proof −
obtain *m* **where** $\bigwedge x\ y.\ x \in T \Longrightarrow y \in space\ M \Longrightarrow norm\ (f\ x\ y) \leq m$
using *assms* **unfolding** *bounded-iff* **by** *auto*
hence *m*:$\bigwedge x\ y.\ x \in T \Longrightarrow y \in space\ M \Longrightarrow norm\ (f\ x\ y) \leq max\ m\ 0$
by *fastforce*

```
  have norm (∫ ω. (f x ω) ∂M) ≤ max m 0 * measure M (space M)
(is ?L ≤ ?R) if x ∈ T for x
  proof −
    have ?L ≤ (∫ ω. norm (f x ω) ∂M) by simp
    also have ... ≤ (∫ ω. max m 0 ∂M)
      using that m by (intro integral-mono′) auto
    also have ... = ?R
      by simp
    finally show ?thesis by simp
  qed
  thus ?thesis
    by (intro boundedI[where B=max m 0 * measure M (space M)])
auto
qed

lemmas bounded-intros =
  bounded-minus-comp bounded-plus-comp bounded-mult-comp bounded-sum
finite-measure.bounded-int
  bounded-const bounded-exp bounded-pow bounded-sum-list

lemma (in prob-space) integrable-bounded:
  fixes f :: - ⇒ ('b :: {banach,second-countable-topology})
  assumes bounded (f ‘ space M)
  assumes f ∈ M →M borel
  shows integrable M f
proof −
  obtain m where norm (f x) ≤ m if x ∈ space M for x
    using assms(1) unfolding bounded-iff by auto
  thus ?thesis
    by (intro integrable-const-bound[where B=m] AE-I2 assms(2))
qed

lemma integrable-bounded-pmf:
  fixes f :: - ⇒ ('b :: {banach,second-countable-topology})
  assumes bounded (f ‘ set-pmf M)
  shows integrable (measure-pmf M) f
proof −
  obtain m where norm (f x) ≤ m if x ∈ set-pmf M for x
    using assms(1) unfolding bounded-iff by auto
  thus ?thesis by (intro measure-pmf.integrable-const-bound[where
B=m] AE-pmfI) auto
qed

end
```

2 Bennett's Inequality

In this section we verify Bennett's inequality [1] and a (weak) version of Bernstein's inequality as a corollary. Both inequalities give concentration bounds for sums of independent random variables. The statement and proofs follow a summary paper by Boucheron et al. [2].

theory *Bennett-Inequality*
imports *Concentration-Inequalities-Preliminary*
begin

context *prob-space*
begin

lemma *indep-vars-Chernoff-ineq-ge*:
assumes *I*: *finite I*
assumes *ind*: *indep-vars* (λ -. *borel*) *X I*
assumes *sge*: $s \geq 0$
assumes *int*: $\bigwedge i.\ i \in I \Longrightarrow$ *integrable M* (λ*x*. *exp* (*s* ∗ *X i x*))
shows *prob* {*x* ∈ *space M*. (∑ *i* ∈*I*. *X i x* − *expectation* (*X i*)) ≥ *t*} ≤
exp (−*s*∗*t*) ∗
(∏ *i*∈*I*. *expectation* (λ*x*. *exp*(*s* ∗ (*X i x* − *expectation* (*X i*)))))
proof (*cases s* = 0)
case [*simp*]: *True*
thus *?thesis*
by (*simp add*: *prob-space*)
next
case *False*
then have *s*: *s* > 0 **using** *sge* **by** *auto*

have [*measurable*]: $\bigwedge i.\ i \in I \Longrightarrow$ *random-variable borel* (*X i*)
using *ind* **unfolding** *indep-vars-def* **by** *blast*

have *indep1*: *indep-vars* (λ-. *borel*)
(λ*i* ω. *exp* (*s* ∗ (*X i* ω − *expectation* (*X i*)))) *I*
apply (*intro indep-vars-compose*[*OF ind*, *unfolded o-def*])
by *auto*

define *S* **where** *S* = (λ*x*. (∑ *i* ∈*I*. *X i x* − *expectation* (*X i*)))

have *int1*: $\bigwedge i.\ i \in I \Longrightarrow$
integrable M (λω. *exp* (*s* ∗ (*X i* ω − *expectation* (*X i*))))
by (*auto simp add*: *algebra-simps exp-diff int*)

have *eprod*: $\bigwedge x.$ *exp* (*s* ∗ *S x*) = (∏ *i*∈*I*. *exp*(*s* ∗ (*X i x* − *expectation* (*X i*))))

```
    unfolding S-def
  by (simp add: assms(1) exp-sum vector-space-over-itself.scale-sum-right)

 from indep-vars-integrable[OF I indep1 int1]
 have intS: integrable M (λx. exp (s * S x))
   unfolding eprod by auto

 then have si: set-integrable M (space M) (λx. exp (s * S x))
   unfolding set-integrable-def
   apply (intro integrable-mult-indicator)
   by auto

 from Chernoff-ineq-ge[OF s si]
 have prob {x ∈ space M. S x ≥ t} ≤ exp (- s * t) * (∫ x∈space M.
exp (s * S x)∂M)
   by auto

 also have (∫ x∈space M. exp (s * S x)∂M) = expectation (λx. exp(s
* S x))
    unfolding set-integral-space[OF intS] by auto

 also have ... = expectation (λx. ∏ i∈I. exp(s * (X i x - expectation
(X i))))
    unfolding S-def
  by (simp add: assms(1) exp-sum vector-space-over-itself.scale-sum-right)
 also have ... = (∏ i∈I. expectation (λx. exp(s * (X i x - expectation
(X i)))))
    apply (intro indep-vars-lebesgue-integral[OF I indep1 int1]) .
 finally show ?thesis
   unfolding S-def
   by auto
qed

definition bennett-h::real ⇒ real
 where bennett-h u = (1 + u) * ln (1 + u) - u

lemma exp-sub-two-terms-eq:
 fixes x :: real
 shows exp x - x - 1 = (∑ n. x^(n+2) / fact (n+2))
   summable (λn. x^(n+2) / fact (n+2))
proof -
 have (∑ i<2. inverse (fact i) * x ^ i) = 1 + x
   by (simp add:numeral-eq-Suc)
 thus exp x - x - 1 = (∑ n. x^(n+2) / fact (n+2))
   unfolding exp-def
   apply (subst suminf-split-initial-segment[where k = 2])
   by (auto simp add: summable-exp divide-inverse-commute)
 have summable (λn. x^n / fact n)
   by (simp add: divide-inverse-commute summable-exp)
```

```
  then have summable (λn. x^(Suc (Suc n)) / fact (Suc (Suc n)))
    apply (subst summable-Suc-iff)
    apply (subst summable-Suc-iff)
    by auto
  thus summable (λn. x^(n+2) / fact (n+2)) by auto
qed

lemma psi-mono:
  defines f ≡ (λx. (exp x − x − 1) − x^2 / 2)
  assumes xy: a ≤ (b::real)
  shows f a ≤ f b
proof −
  have 1: (f has-real-derivative (exp x − x − 1)) (at x) for x
    unfolding f-def
    by (auto intro!: derivative-eq-intros)

  have 2: ⋀x. x ∈ {a..b} ⟹ 0 ≤ exp x − x − 1
    by (smt (verit) exp-ge-add-one-self)

  from deriv-nonneg-imp-mono[OF 1 2 xy]
  show ?thesis by auto
qed


lemma psi-inequality:
  assumes le: x ≤ (y::real) y ≥ 0
  shows y^2 * (exp x − x − 1) ≤ x^2 * (exp y − y − 1)
proof −

  have x: exp x − x − 1 = (∑ n. (x^(n+2) / fact (n+2)))
    summable (λn. x^(n+2) / fact (n+2))
    using exp-sub-two-terms-eq .

  have y: exp y − y − 1 = (∑ n. (y^(n+2) / fact (n+2)))
    summable (λn. y^(n+2) / fact (n+2))
    using exp-sub-two-terms-eq .


   have l:y^2 * (exp x − x − 1) = (∑ n. y^2 * (x^(n+2) / fact
(n+2)))
    using x
    apply (subst suminf-mult)
    by auto
  have ls: summable (λn. y^2 * (x^(n+2) / fact (n+2)))
    by (intro summable-mult[OF x(2)])

   have r:x^2 * (exp y − y − 1) = (∑ n. x^2 * (y^(n+2) / fact
(n+2)))
    using y
```

```
    apply (subst suminf-mult)
    by auto
  have rs: summable (λn. x^2 * (y^(n+2) / fact (n+2)))
    by (intro summable-mult[OF y(2)])

  have |x| ≤ |y| ∨ |y| < |x| by auto
  moreover {
    assume |x| ≤ |y|
    then have x^ n ≤ y ^n for n
   by (smt (verit, ccfv-threshold) bot-nat-0.not-eq-extremum le power-0
real-root-less-mono real-root-power-cancel root-abs-power)
    then have (x^2 * y^2) * x^n ≤ (x^2 * y^2) * y^n for n
      by (simp add: mult-left-mono)
    then have y^2 * (x ^(n + 2)) ≤ x^2 * (y ^(n + 2)) for n
    by (metis (full-types) ab-semigroup-mult-class.mult-ac(1) mult.commute
power-add)
    then have y^2 * (x ^(n + 2)) / fact (n+2)≤ x^2 * (y ^(n + 2))
/ fact (n+2) for n
      by (meson divide-right-mono fact-ge-zero)
    then have (∑n. y^2 * (x^(n+2) / fact (n+2))) ≤ (∑n. x^2 *
(y^(n+2) / fact (n+2)))
      apply (intro suminf-le[OF - ls rs])
      by auto
    then have y^2 * (exp x − x − 1) ≤ x^2 * (exp y − y − 1)
    using l r by presburger
  }
  moreover {
    assume ineq: |y| < |x|

    from psi-mono[OF assms(1)]
    have (exp x − x − 1) − x^2 /2 ≤ (exp y − y − 1) − y^2/2 .

    then have y^2 * ((exp x − x − 1) − x^2 /2) ≤ x^2 * ((exp y −
y − 1) − y^2/2)
    by (smt (verit, best) ineq diff-divide-distrib exp-lower-Taylor-quadratic
le(1) le(2) mult-nonneg-nonneg one-less-exp-iff power-zero-numeral prob-space.psi-mono
prob-space-completion right-diff-distrib zero-le-power2)

    then have y^2 * (exp x − x − 1) ≤ x^2 * (exp y − y − 1)
      by (simp add: mult.commute right-diff-distrib)
  }
  ultimately show ?thesis by auto
qed


lemma bennett-inequality-1:
  assumes I: finite I
  assumes ind: indep-vars (λ -. borel) X I
  assumes intsq: ⋀i. i ∈ I ⟹ integrable M (λx. (X i x)^2)
```

assumes *bnd*: $\bigwedge i.\ i \in I \Longrightarrow AE\ x\ in\ M.\ X\ i\ x \leq 1$
assumes *t*: $t \geq 0$
defines $V \equiv (\sum i \in I.\ expectation(\lambda x.\ X\ i\ x\hat{}2))$
shows $prob\ \{x \in space\ M.\ (\sum i \in I.\ X\ i\ x - expectation\ (X\ i)) \geq t\} \leq$
$exp\ (-V * bennett\text{-}h\ (t\ /\ V))$
proof (*cases* $V = 0$)
case *True*
then show *?thesis*
by *auto*
next
case *f*: *False*
have $V \geq 0$
unfolding *V-def*
apply (*intro sum-nonneg integral-nonneg-AE*)
by *auto*
then have *Vpos*: $V > 0$ **using** *f* **by** *auto*

define $l :: real$ **where** $l = ln(1 + t\ /\ V)$
then have *l*: $l \geq 0$
using *t Vpos* **by** *auto*
have *rv*[*measurable*]: $\bigwedge i.\ i \in I \Longrightarrow random\text{-}variable\ borel\ (X\ i)$
using *ind* **unfolding** *indep-vars-def* **by** *blast*

define ψ **where** $\psi = (\lambda x{::}real.\ exp(x) - x - 1)$

have *rw*: $exp\ y = 1 + y + \psi\ y$ **for** *y*
unfolding ψ*-def* **by** *auto*

have *ebnd*: $\bigwedge i.\ i \in I \Longrightarrow$
$AE\ x\ in\ M.\ exp\ (l * X\ i\ x) \leq exp\ l$
apply (*drule bnd*)
using *l* **by** (*auto simp add*: *mult-left-le*)

have *int*: $\bigwedge i.\ i \in I \Longrightarrow integrable\ M\ (\lambda x.\ (X\ i\ x))$
using *rv intsq square-integrable-imp-integrable* **by** *blast*

have *intl*: $\bigwedge i.\ i \in I \Longrightarrow integrable\ M\ (\lambda x.\ (l * X\ i\ x))$
using *int* **by** *blast*

have *intexpl*: $\bigwedge i.\ i \in I \Longrightarrow integrable\ M\ (\lambda x.\ exp\ (l * X\ i\ x))$
apply (*intro integrable-const-bound*[**where** $B = exp\ l$])
using *ebnd* **by** *auto*

have *intpsi*: $\bigwedge i.\ i \in I \Longrightarrow integrable\ M\ (\lambda x.\ \psi\ (l * X\ i\ x))$
unfolding ψ*-def*
using *intl intexpl* **by** *auto*

```
  have **: ⋀i. i ∈ I ⟹
    expectation (λx. ψ (l * X i x)) ≤ ψ l * expectation (λx. (X i x)^2)
  proof −
    fix i assume i: i ∈ I
    then have AE x in M. l * X i x ≤ l
      using ebnd by auto
    then have AE x in M. l^2 * ψ (l * X i x) ≤ (l * X i x)^2 * ψ l
      using psi-inequality[OF - l] unfolding ψ-def
      by auto
    then have AE x in M. l^2 * ψ (l * X i x) ≤ l^2 * (ψ l * (X i
x)^2)
      by (auto simp add: field-simps)
    then have AE x in M. ψ (l * X i x) ≤ ψ l * (X i x)^2
     by (smt (verit, best) AE-cong ψ-def exp-eq-one-iff mult-cancel-left
mult-eq-0-iff mult-left-mono zero-eq-power2 zero-le-power2)
    then have AE x in M. 0 ≤ ψ l * (X i x)^2 − ψ (l * X i x)
      by auto
    then have expectation (λx. ψ l * (X i x)^2 + (− ψ (l * X i x)))
≥ 0
      by (simp add: integral-nonneg-AE)
    also have expectation (λx. ψ l * (X i x)^2 + (− ψ (l * X i x))) =
        ψ l * expectation (λx. (X i x)^2) − expectation (λx. ψ (l * X i
x))
      apply (subst Bochner-Integration.integral-add)
      using intpsi[OF i] intsq[OF i] by auto
    finally show expectation (λx. ψ (l * X i x)) ≤ ψ l * expectation
(λx. (X i x)^2)
      by auto
  qed


  then have *: ⋀i. i ∈ I ⟹
      expectation (λx. exp (l * X i x)) ≤
     exp (l * expectation (X i)) * exp (ψ l * expectation (λx. X i x^2))
  proof −
    fix i
    assume iI: i ∈ I
    have expectation (λx. exp (l * X i x)) =
      1 + l * expectation (λx. X i x) +
       expectation (λx. ψ (l * X i x))
      unfolding rw
      apply (subst Bochner-Integration.integral-add)
      using iI intl intpsi apply auto[2]
      apply (subst Bochner-Integration.integral-add)
      using intl iI prob-space by auto
    also have ... = l * expectation (X i) + 1 + expectation (λx. ψ (l
* X i x))
      by auto
    also have ... ≤ 1 + l * expectation (X i) + ψ l * expectation (λx.
```

```
X i x^2)
      using **[OF iI] by auto
    also have ... ≤ exp (l * expectation (X i)) * exp (ψ l * expectation
(λx. X i x^2))
      by (simp add: is-num-normalize(1) mult-exp-exp)
    finally show expectation (λx. exp (l * X i x)) ≤
     exp (l * expectation (X i)) * exp (ψ l * expectation (λx. X i x^2))
.
  qed

  have (∏i∈I. expectation (λx. exp (l * (X i x)))) ≤
    (∏i∈I. exp (l * expectation (X i)) * exp (ψ l * expectation (λx.
X i x^2)))
    by (auto intro!: prod-mono simp add: *)
  also have ... =
   (∏i∈I. exp (l * expectation (X i))) * (∏i∈I. exp (ψ l * expectation
(λx. X i x^2)))
    by (auto simp add: prod.distrib)
  finally have **:
    (∏i∈I. expectation (λx. exp (l * (X i x)))) ≤
    (∏i∈I. exp (l * expectation (X i))) * exp (ψ l * V)
    by (simp add: V-def I exp-sum sum-distrib-left)

  from indep-vars-Chernoff-ineq-ge[OF I ind l intexpl]
  have prob {x ∈ space M. (∑ i ∈ I. X i x − expectation (X i)) ≥ t}
≤
    exp (− l * t) *
     (∏i∈I. expectation (λx. exp (l * (X i x − expectation (X i)))))
     by auto
  also have (∏i∈I. expectation (λx. exp (l * (X i x − expectation (X
i))))) =
    (∏i∈I. expectation (λx. exp (l * (X i x))) * exp (− l * expectation
(X i)))
   by (auto intro!: prod.cong simp add: field-simps exp-diff exp-minus-inverse)
  also have ... =
     (∏i∈I. exp (− l * expectation (X i))) * (∏i∈I. expectation (λx.
exp (l * (X i x))))
    by (auto simp add: prod.distrib)
  also have ... ≤
     (∏i∈I. exp (− l * expectation (X i))) * ((∏i∈I. exp (l * expec-
tation (X i))) * exp (ψ l * V))
    apply (intro mult-left-mono[OF **])
    by (meson exp-ge-zero prod-nonneg)
  also have ... = exp (ψ l * V)
    apply (simp add: prod.distrib [symmetric])
   by (smt (verit, ccfv-threshold) exp-minus-inverse prod.not-neutral-contains-not-neutral)
  finally have
    prob {x ∈ space M. (∑ i ∈ I. X i x − expectation (X i)) ≥ t} ≤
    exp (ψ l * V − l * t)
```

by (*simp add*:*mult-exp-exp*)
also have $\psi\ l * V - l * t = -V * \textit{bennett-h}\ (t\ /\ V)$
unfolding *ψ-def l-def bennett-h-def*
apply (*subst exp-ln*)
subgoal by (*smt* (*verit*) *Vpos divide-nonneg-nonneg t*)
by (*auto simp add*: *algebra-simps*)
finally show *?thesis* **.**
qed

lemma *real-AE-le-sum*:
assumes $\bigwedge i.\ i \in I \Longrightarrow AE\ x\ in\ M.\ f\ i\ x \leq (g\ i\ x::real)$
shows $AE\ x\ in\ M.\ (\sum i{\in}I.\ f\ i\ x) \leq (\sum i{\in}I.\ g\ i\ x)$
proof (*cases*)
assume *finite I*
with *AE-finite-allI*[*OF this assms*] **have** *0*:$AE\ x\ in\ M.\ (\forall\, i{\in}I.\ f\ i\ x \leq g\ i\ x)$ **by** *auto*
show *?thesis* **by** (*intro eventually-mono*[*OF 0*] *sum-mono*) *auto*
qed *simp*

lemma *real-AE-eq-sum*:
assumes $\bigwedge i.\ i \in I \Longrightarrow AE\ x\ in\ M.\ f\ i\ x = (g\ i\ x::real)$
shows $AE\ x\ in\ M.\ (\sum i{\in}I.\ f\ i\ x) = (\sum i{\in}I.\ g\ i\ x)$
proof −
have *1*: $AE\ x\ in\ M.\ (\sum i{\in}I.\ f\ i\ x) \leq (\sum i{\in}I.\ g\ i\ x)$
apply (*intro real-AE-le-sum*)
apply (*drule assms*)
by *auto*
have *2*: $AE\ x\ in\ M.\ (\sum i{\in}I.\ g\ i\ x) \leq (\sum i{\in}I.\ f\ i\ x)$
apply (*intro real-AE-le-sum*)
apply (*drule assms*)
by *auto*
show *?thesis*
using *1 2*
by *auto*
qed

theorem *bennett-inequality*:
assumes *I*: *finite I*
assumes *ind*: *indep-vars* (*λ -. borel*) *X I*
assumes *intsq*: $\bigwedge i.\ i \in I \Longrightarrow \textit{integrable}\ M\ (\lambda x.\ (X\ i\ x)\hat{}2)$
assumes *bnd*: $\bigwedge i.\ i \in I \Longrightarrow AE\ x\ in\ M.\ X\ i\ x \leq B$
assumes *t*: $t \geq 0$
assumes *B*: $B > 0$
defines $V \equiv (\sum i \in I.\ \textit{expectation}\ (\lambda x.\ X\ i\ x\hat{}2))$
shows $\textit{prob}\ \{x \in \textit{space}\ M.\ (\sum i \in I.\ X\ i\ x - \textit{expectation}\ (X\ i)) \geq t\} \leq$
$\textit{exp}\ (-\ V\ /\ B\hat{}2 * \textit{bennett-h}\ (t * B\ /\ V))$
proof −

define Y **where** $Y = (\lambda i\ x.\ X\ i\ x\ /\ B)$

from *indep-vars-compose*[*OF ind*, **where** $Y = \lambda i\ x.\ x/\ B$]
have *1*: *indep-vars* (λ-. *borel*) $Y\ I$
unfolding *Y-def* **by** (*auto simp add*: *o-def*)
have *2*: $\bigwedge i.\ i \in I \Longrightarrow$ *integrable* M $(\lambda x.\ (Y\ i\ x)^2)$
unfolding *Y-def* **apply** (*drule intsq*)
by (*auto simp add*: *field-simps*)
have *3*: $\bigwedge i.\ i \in I \Longrightarrow$ *AE x in M*. $Y\ i\ x \leq 1$
unfolding *Y-def* **apply** (*drule bnd*)
using B **by** *auto*
have *4*:$0 \leq t\ /\ B$ **using** $t\ B$ **by** *auto*

have *rw1*: $(\sum i{\in}I.\ Y\ i\ x - expectation\ (Y\ i)) =$
$(\sum i{\in}I.\ X\ i\ x - expectation\ (X\ i))\ /\ B$ **for** x
unfolding *Y-def*
by (*auto simp*: *diff-divide-distrib sum-divide-distrib*)

have *rw2*: *expectation* $(\lambda x.\ (Y\ i\ x)^2) =$
expectation $(\lambda x.\ (X\ i\ x)^2)\ /\ B\hat{}2$ **for** i
unfolding *Y-def*
by (*simp add*: *power-divide*)

have *rw3*:$-\ (\sum i{\in}I.\ expectation\ (\lambda x.\ (X\ i\ x)^2)\ /\ B\hat{}2) = -\ V\ /\ B\hat{}2$
unfolding *V-def*
by (*auto simp add*: *sum-divide-distrib*)

have $t\ /\ B\ /\ (\sum i{\in}I.\ expectation\ (\lambda x.\ (X\ i\ x)^2)\ /\ B\hat{}2) =$
$t\ /\ B\ /\ (V\ /\ B\hat{}2)$
unfolding *V-def*
by (*auto simp add*: *sum-divide-distrib*)
then have *rw4*: $t\ /\ B\ /\ (\sum i{\in}I.\ expectation\ (\lambda x.\ (X\ i\ x)^2)\ /\ B\hat{}2) =$
$t * B\ /\ V$
by (*simp add*: *power2-eq-square*)
have *prob* $\{x \in space\ M.\ t \leq (\sum i{\in}I.\ X\ i\ x - expectation\ (X\ i))\} =$
$prob\{x \in space\ M.\ t\ /\ B \leq (\sum i{\in}I.\ X\ i\ x - expectation\ (X\ i))\ /\ B\}$
by (*smt* (*verit*, *best*) B *Collect-cong divide-cancel-right divide-right-mono*)
also have ... $\leq$
exp $(-\ V\ /\ B^2 *$
bennett-h $(t * B\ /\ V))$
using *bennett-inequality-1*[*OF I 1 2 3 4*]
unfolding *rw1 rw2 rw3 rw4* .
finally show *?thesis* .
qed

lemma *bennett-h-bernstein-bound*:
assumes $x \geq 0$
shows *bennett-h* $x \geq x\hat{}2 \;/\; (2 * (1 + x \;/\; 3))$
proof −
have *eq*:$x\hat{}2 \;/\; (2 * (1 + x \;/\; 3)) = 3/2 * x - 9/2 * (x \;/\; (x+3))$
using *assms*
by (*sos* (() & ()))

define g **where** $g = (\lambda x.\ \textit{bennett-h}\ x - (3/2 * x - 9/2 * (x \;/\; (x+3))))$

define g' **where** $g' = (\lambda x{::}\textit{real}.$
$\textit{ln}(1 + x) + \ 27 \;/\; (2 * (x+3)\hat{}2) - \ 3 \;/\; 2)$
define g'' **where** $g'' = (\lambda x{::}\textit{real}.$
$1 \;/\; (1 + x) - 27 \;/\; (x+3)\hat{}3)$

have $54 \;/\; ((2 * x + 6)\hat{}2) = 27 \;/\; (2 * (x + 3)^2)$ (**is** *?L* = *?R*)
for x :: *real*
proof −
have *?L* = $54 \;/\; (2\hat{}2 * (x + 3)\hat{}2)$
unfolding *power-mult-distrib*[*symmetric*] **by** (*simp add*:*algebra-simps*)
also have ... = *?R* **by** *simp*
finally show *?thesis* **by** *simp*
qed

hence *1*: $x \geq 0 \Longrightarrow (g\ \textit{has-real-derivative}\ (g'\ x))\ (\textit{at}\ x)$ **for** x
unfolding *g-def g'-def bennett-h-def* **by** (*auto intro*!: *derivative-eq-intros simp*:*power2-eq-square*)
have *2*: $x \geq 0 \Longrightarrow (g'\ \textit{has-real-derivative}\ (g''\ x))\ (\textit{at}\ x)$ **for** x
unfolding *g'-def g''-def*
apply (*auto intro*!: *derivative-eq-intros*)[*1*]
by (*sos* (() & ()))

have *gz*: $g\ 0 = 0$
unfolding *g-def bennett-h-def* **by** *auto*
have *g1z*: $g'\ 0 = 0$
unfolding *g'-def* **by** *auto*

have *p2*: $g''\ x\ \geq 0$ **if** $x \geq 0$ **for** x
proof −
have $27 * (1+x) \leq (x+3)\hat{}3$
using *that* **unfolding** *power3-eq-cube* **by** (*auto simp*:*algebra-simps*)
hence $27 \;/\; (x + 3)\ \hat{}\ 3 \leq 1 \;/\; (1+x)$
using *that* **by** (*subst frac-le-eq*) (*auto intro*!:*divide-nonpos-pos*)
thus *?thesis* **unfolding** *g''-def* **by** *simp*
qed

from *deriv-nonneg-imp-mono*[*OF 2 p2* -]

```
  have x ≥ 0 ⟹ g' x ≥ 0 for x using g1z
    by (metis atLeastAtMost-iff)

  from deriv-nonneg-imp-mono[OF 1 this -]
  have x ≥ 0 ⟹ g x ≥ 0 for x using gz
    by (metis atLeastAtMost-iff)

  thus ?thesis
  using assms eq g-def by force
qed

lemma sum-sq-exp-eq-zero-imp-zero:
  assumes finite I i ∈ I
  assumes intsq: integrable M (λx. (X i x)^2)
  assumes (∑ i ∈ I. expectation (λx. X i x^2)) = 0
  shows AE x in M. X i x = (0::real)
proof −
  have (∀ i ∈I. expectation (λx. X i x^2) = 0)
    using assms
    apply (subst sum-nonneg-eq-0-iff[symmetric])
    by auto
  then have expectation (λx. X i x^2) = 0
    using assms(2) by blast
  thus ?thesis
    using integral-nonneg-eq-0-iff-AE[OF intsq]
    by auto
qed

corollary bernstein-inequality:
  assumes I: finite I
  assumes ind: indep-vars (λ -. borel) X I
  assumes intsq: ⋀i. i ∈ I ⟹ integrable M (λx. (X i x)^2)
  assumes bnd: ⋀i. i ∈ I ⟹ AE x in M. X i x ≤ B
  assumes t: t ≥ 0
  assumes B: B > 0
  defines V ≡ (∑ i ∈ I. expectation (λx. X i x^2))
  shows prob {x ∈ space M. (∑ i ∈ I. X i x − expectation (X i)) ≥
t} ≤
    exp (− (t^2 / (2 * (V + t * B / 3))))
proof (cases V = 0)
  case True
  then have 1:⋀i. i ∈ I ⟹ AE x in M. X i x = 0
    unfolding V-def
    using sum-sq-exp-eq-zero-imp-zero
    by (metis I intsq)
  then have 2:⋀i. i ∈ I ⟹ expectation (X i) = 0
    using integral-eq-zero-AE by blast

  have AE x in M. (∑ i ∈ I. X i x − expectation (X i)) = (∑ i ∈ I.
```

```
0)
      apply (intro real-AE-eq-sum)
      using 1 2
      by auto
  then have *: AE x in M. (∑ i ∈ I. X i x − expectation (X i)) = 0
    by force

  moreover {
    assume t > 0
    then have prob {x ∈ space M. (∑ i ∈ I. X i x − expectation (X
i)) ≥ t} = 0
      apply (intro prob-eq-0-AE)
      using * by auto
    then have ?thesis by auto
  }
  ultimately show ?thesis
    apply (cases t = 0) using t by auto
next
  case f: False
  have V ≥ 0
    unfolding V-def
    apply (intro sum-nonneg  integral-nonneg-AE)
    by auto
  then have V: V > 0 using f by auto

  have t * B / V ≥ 0 using t B V by auto
  from bennett-h-bernstein-bound[OF this]
  have (t * B / V)^2 / (2 * (1 + t * B / V / 3))
    ≤ bennett-h (t * B / V) .

  then have (− V / B^2) * bennett-h (t * B / V) ≤
    (− V / B^2) * ((t * B / V)^2 / (2 * (1 + t * B / V / 3)))
    apply (subst mult-left-mono-neg)
    using B V by auto
  also have ... =
     ((− V / B^2) * (t * B / V)^2) / (2 * (1 + t * B / V / 3))
    by auto
  also have  ((− V / B^2) * (t * B / V)^2) = −(t^2) / V
    using V B by (auto simp add: field-simps power2-eq-square)
  finally have *: (− V / B^2) * bennett-h (t * B / V) ≤
     −(t^2)  / (2 * (V + t * B  / 3))
    using V by (auto simp add: field-simps)

  from bennett-inequality[OF assms(1−6)]
  have prob {x ∈ space M. (∑ i ∈ I. X i x − expectation (X i)) ≥ t}
≤
    exp (− V / B^2 * bennett-h (t * B / V))
    using V-def by auto
  also have ... ≤ exp (− (t^2/ (2 * (V + t * B  / 3))))
```

using ∗
by *auto*
finally show *?thesis* **.**
qed

end

end

3 Bienaymé's identity

Bienaymé's identity [5, §17] can be used to deduce the variance of a sum of random variables, if their co-variance is known. A common use-case of the identity is the computation of the variance of the mean of pair-wise independent variables.

theory *Bienaymes-Identity*
imports *Concentration-Inequalities-Preliminary*
begin

context *prob-space*
begin

lemma *variance-divide*:
fixes $f :: 'a \Rightarrow real$
assumes *integrable M f*
shows *variance* $(\lambda\omega.\ f\ \omega\ /\ r) =$ *variance f* $/\ r\hat{}2$
using *assms*
by (*subst Bochner-Integration.integral-divide*[*OF assms*(*1*)])
(*simp add*:*diff-divide-distrib*[*symmetric*] *power2-eq-square algebra-simps*)

definition *covariance* **where**
covariance f g = *expectation* $(\lambda\omega.\ (f\ \omega -$ *expectation f*$) * (g\ \omega -$ *expectation g*$))$

lemma *covariance-eq*:
fixes $f :: 'a \Rightarrow real$
assumes $f \in$ *borel-measurable M* $g \in$ *borel-measurable M*
assumes *integrable M* $(\lambda\omega.\ f\ \omega\hat{}2)$ *integrable M* $(\lambda\omega.\ g\ \omega\hat{}2)$
shows *covariance f g* = *expectation* $(\lambda\omega.\ f\ \omega * g\ \omega) -$ *expectation f* $*$ *expectation g*
proof −
have *integrable M f* **using** *square-integrable-imp-integrable assms* **by** *auto*
moreover have *integrable M g* **using** *square-integrable-imp-integrable assms* **by** *auto*
ultimately show *?thesis*
using *assms cauchy-schwartz*(*1*)[**where** *M*=*M*]
by (*simp add*:*covariance-def algebra-simps prob-space*)

qed

lemma *covar-integrable*:
fixes $f\ g :: 'a \Rightarrow real$
assumes $f \in borel\text{-}measurable\ M\ g \in borel\text{-}measurable\ M$
assumes $integrable\ M\ (\lambda\omega.\ f\ \omega\widehat{}2)\ integrable\ M\ (\lambda\omega.\ g\ \omega\widehat{}2)$
shows $integrable\ M\ (\lambda\omega.\ (f\ \omega - expectation\ f) * (g\ \omega - expectation\ g))$
proof −
have $integrable\ M\ f$ **using** *square-integrable-imp-integrable assms* **by** *auto*
moreover have $integrable\ M\ g$ **using** *square-integrable-imp-integrable assms* **by** *auto*
ultimately show *?thesis* **using** *assms cauchy-schwartz*(*1*)[**where** *M*=*M*] **by** (*simp add*: *algebra-simps*)
qed

lemma *sum-square-int*:
fixes $f :: 'b \Rightarrow 'a \Rightarrow real$
assumes *finite I*
assumes $\bigwedge i.\ i \in I \Longrightarrow f\ i \in borel\text{-}measurable\ M$
assumes $\bigwedge i.\ i \in I \Longrightarrow integrable\ M\ (\lambda\omega.\ f\ i\ \omega\widehat{}2)$
shows $integrable\ M\ (\lambda\omega.\ (\sum i \in I.\ f\ i\ \omega)^2)$
proof −
have $integrable\ M\ (\lambda\omega.\ \sum i{\in}I.\ \sum j{\in}I.\ f\ j\ \omega * f\ i\ \omega)$
using *assms*
by (*intro Bochner-Integration.integrable-sum cauchy-schwartz*(*1*)[**where** *M*=*M*], *auto*)
thus *?thesis*
by (*simp add*:*power2-eq-square sum-distrib-left sum-distrib-right*)
qed

theorem *bienaymes-identity*:
fixes $f :: 'b \Rightarrow 'a \Rightarrow real$
assumes *finite I*
assumes $\bigwedge i.\ i \in I \Longrightarrow f\ i \in borel\text{-}measurable\ M$
assumes $\bigwedge i.\ i \in I \Longrightarrow integrable\ M\ (\lambda\omega.\ f\ i\ \omega\widehat{}2)$
shows
$variance\ (\lambda\omega.\ (\sum i \in I.\ f\ i\ \omega)) = (\sum i \in I.\ (\sum j \in I.\ covariance\ (f\ i)\ (f\ j)))$
proof −
have $a{:}\bigwedge i\ j.\ i \in I \Longrightarrow j \in I \Longrightarrow$
$integrable\ M\ (\lambda\omega.\ (f\ i\ \omega - expectation\ (f\ i)) * (f\ j\ \omega - expectation\ (f\ j)))$
using *assms covar-integrable* **by** *simp*
have $variance\ (\lambda\omega.\ (\sum i \in I.\ f\ i\ \omega)) = expectation\ (\lambda\omega.\ (\sum i{\in}I.\ f\ i\ \omega - expectation\ (f\ i))^2)$
using *square-integrable-imp-integrable*[*OF assms*(*2*,*3*)]
by (*simp add*: *Bochner-Integration.integral-sum sum-subtractf*)

also have ... = *expectation* ($\lambda\omega$. ($\sum i \in I$. ($\sum j \in I$.
($f\ i\ \omega$ − *expectation* ($f\ i$)) ∗ ($f\ j\ \omega$ − *expectation* ($f\ j$)))))
by (*simp add*: *power2-eq-square sum-distrib-right sum-distrib-left mult.commute*)
also have ... = ($\sum i \in I$. ($\sum j \in I$. *covariance* ($f\ i$) ($f\ j$)))
using *a* **by** (*simp add*: *Bochner-Integration.integral-sum covariance-def*)
finally show *?thesis* **by** *simp*
qed

lemma *covar-self-eq*:
fixes $f :: 'a \Rightarrow real$
shows *covariance f f* = *variance f*
by (*simp add*:*covariance-def power2-eq-square*)

lemma *covar-indep-eq-zero*:
fixes $f\ g :: 'a \Rightarrow real$
assumes *integrable M f*
assumes *integrable M g*
assumes *indep-var borel f borel g*
shows *covariance f g* = *0*
proof −
have *a*:*indep-var borel* ((λt. *t* − *expectation f*) ∘ *f*) *borel* ((λt. *t* − *expectation g*) ∘ *g*)
by (*rule indep-var-compose*[*OF assms*(*3*)], *auto*)

have *b*:*expectation* ($\lambda\omega$. (*f* ω − *expectation f*) ∗ (*g* ω − *expectation g*)) = *0*
using *a assms* **by** (*subst indep-var-lebesgue-integral*, *auto simp add*:*comp-def prob-space*)

thus *?thesis* **by** (*simp add*:*covariance-def*)
qed

lemma *bienaymes-identity-2*:
fixes $f :: 'b \Rightarrow 'a \Rightarrow real$
assumes *finite I*
assumes $\bigwedge i.\ i \in I \Longrightarrow f\ i \in$ *borel-measurable M*
assumes $\bigwedge i.\ i \in I \Longrightarrow$ *integrable M* ($\lambda\omega$. *f i* $\omega\hat{}2$)
shows *variance* ($\lambda\omega$. ($\sum i \in I$. *f i* ω)) =
($\sum i \in I$. *variance* (*f i*)) + ($\sum i \in I$. $\sum j \in I - \{i\}$. *covariance* (*f i*) (*f j*))
proof −
have *variance* ($\lambda\omega$. ($\sum i \in I$. *f i* ω)) = ($\sum i \in I$. $\sum j \in I$. *covariance* (*f i*) (*f j*))
by (*simp add*: *bienaymes-identity*[*OF assms*(*1*,*2*,*3*)])
also have ... = ($\sum i \in I$. *covariance* (*f i*) (*f i*) + ($\sum j \in I - \{i\}$. *covariance* (*f i*) (*f j*)))
using *assms* **by** (*subst sum.insert*[*symmetric*], *auto simp add*:*insert-absorb*)

also have ... = ($\sum i \in I$. *variance* (*f i*)) + ($\sum i \in I$. ($\sum j \in I - \{i\}$. *covariance* (*f i*) (*f j*)))
by (*simp add*: *covar-self-eq*[*symmetric*] *sum.distrib*)
finally show *?thesis* **by** *simp*
qed

theorem *bienaymes-identity-pairwise-indep*:
fixes *f* :: *'b* ⇒ *'a* ⇒ *real*
assumes *finite I*
assumes ⋀*i*. *i* ∈ *I* ⟹ *f i* ∈ *borel-measurable M*
assumes ⋀*i*. *i* ∈ *I* ⟹ *integrable M* (λω. *f i* ω^2)
assumes ⋀*i j*. *i* ∈ *I* ⟹ *j* ∈ *I* ⟹ *i* ≠ *j* ⟹ *indep-var borel* (*f i*) *borel* (*f j*)
shows *variance* (λω. ($\sum i \in I$. *f i* ω)) = ($\sum i \in I$. *variance* (*f i*))
proof −
have ⋀*i j*. *i* ∈ *I* ⟹ *j* ∈ *I* − {*i*} ⟹ *covariance* (*f i*) (*f j*) = *0*
using *covar-indep-eq-zero assms*(*4*) *square-integrable-imp-integrable*[*OF assms*(*2*,*3*)] **by** *auto*
hence *a*:($\sum i \in I$. $\sum j \in I - \{i\}$. *covariance* (*f i*) (*f j*)) = *0*
by *simp*
thus *?thesis* **by** (*simp add*: *bienaymes-identity-2*[*OF assms*(*1*,*2*,*3*)])
qed

lemma *indep-var-from-indep-vars*:
assumes *i* ≠ *j*
assumes *indep-vars* (λ-. *M'*) *f* {*i*, *j*}
shows *indep-var M'* (*f i*) *M'* (*f j*)
proof −
have *a*:*inj* (*case-bool i j*) **using** *assms*(*1*)
by (*simp add*: *bool.case-eq-if inj-def*)
have *b*:*range* (*case-bool i j*) = {*i*,*j*}
by (*simp add*: *UNIV-bool insert-commute*)
have *c*:*indep-vars* (λ-. *M'*) *f* (*range* (*case-bool i j*)) **using** *assms*(*2*) *b* **by** *simp*

have *True* = *indep-vars* (λ*x*. *M'*) (λ*x*. *f* (*case-bool i j x*)) *UNIV*
using *indep-vars-reindex*[*OF a c*]
by (*simp add*:*comp-def*)
also have ... = *indep-vars* (λ*x*. *case-bool M' M' x*) (λ*x*. *case-bool* (*f i*) (*f j*) *x*) *UNIV*
by (*rule indep-vars-cong*, *auto simp*:*bool.case-distrib bool.case-eq-if*)
also have ... = *?thesis*
by (*simp add*: *indep-var-def*)
finally show *?thesis* **by** *simp*
qed

lemma *bienaymes-identity-pairwise-indep-2*:
fixes *f* :: *'b* ⇒ *'a* ⇒ *real*
assumes *finite I*

assumes $\bigwedge i.\ i \in I \Longrightarrow f\ i \in$ *borel-measurable M*
assumes $\bigwedge i.\ i \in I \Longrightarrow$ *integrable M* $(\lambda\omega.\ f\ i\ \omega\hat{}2)$
assumes $\bigwedge J.\ J \subseteq I \Longrightarrow$ *card J* $= 2 \Longrightarrow$ *indep-vars* $(\lambda$ *-. borel*$)\ f\ J$
shows *variance* $(\lambda\omega.\ (\sum i \in I.\ f\ i\ \omega)) = (\sum i \in I.$ *variance* $(f\ i))$
using *assms*(*4*)
by (*intro bienaymes-identity-pairwise-indep*[*OF assms*(*1*,*2*,*3*)] *indep-var-from-indep-vars*, *auto*)

lemma *bienaymes-identity-full-indep*:
fixes $f :: 'b \Rightarrow 'a \Rightarrow$ *real*
assumes *finite I*
assumes $\bigwedge i.\ i \in I \Longrightarrow f\ i \in$ *borel-measurable M*
assumes $\bigwedge i.\ i \in I \Longrightarrow$ *integrable M* $(\lambda\omega.\ f\ i\ \omega\hat{}2)$
assumes *indep-vars* $(\lambda$ *-. borel*$)\ f\ I$
shows *variance* $(\lambda\omega.\ (\sum i \in I.\ f\ i\ \omega)) = (\sum i \in I.$ *variance* $(f\ i))$
by (*intro bienaymes-identity-pairwise-indep-2*[*OF assms*(*1*,*2*,*3*)] *indep-vars-subset*[*OF assms*(*4*)])
auto

end

end

4 Cantelli's Inequality

Cantelli's inequality [3] is an improvement of Chebyshev's inequality for one-sided tail bounds.

theory *Cantelli-Inequality*
imports *HOL−Probability.Probability*
begin

context *prob-space*
begin

lemma *cantelli-arith*:
assumes $a > (0::real)$
shows $(V + (V\ /\ a)\hat{}2)\ /\ (a + (V\ /\ a))\hat{}2 = V\ /\ (a\ \hat{}2 + V)$ (**is** *?L = ?R*)
proof −
have *?L* $= ((V * a\hat{}2 + V\hat{}2)\ /\ a\hat{}2)\ /\ ((a\hat{}2 + V)\hat{}2/a\hat{}2)$
using *assms* **by** (*intro arg-cong2*[**where** *f*=(/)]) (*simp-all add:field-simps power2-eq-square*)
also have ... $= (V * a^2 + V^2)/\ (a^2 + V)^2$
using *assms* **unfolding** *divide-divide-times-eq* **by** *simp*
also have ... $= V * (a\hat{}2 + V)\ /\ (a\hat{}2 + V)\hat{}2$
by (*intro arg-cong2*[**where** *f*=(/)]) (*simp-all add*: *algebra-simps power2-eq-square*)
also have ... = *?R* **by** (*simp add:power2-eq-square*)

```
  finally show ?thesis by simp
qed

theorem cantelli-inequality:
  assumes [measurable]: random-variable borel Z
  assumes intZsq: integrable M (λz. Z z^2)
  assumes a: a > 0
  shows prob {z ∈ space M. Z z − expectation Z ≥ a} ≤
    variance Z / (a^2 + variance Z)
proof −
  define u where u = variance Z / a
  have u: u ≥ 0
    unfolding u-def
    by (simp add: a divide-nonneg-pos)
  define Y where Y = (λz. Z z + (−expectation Z))
  have random-variable borel (λz. |Y z + u|)
    unfolding Y-def
    by auto
  then have ev: {z ∈ space M. a + u ≤ |Y z + u|} ∈ events
    by auto

  have intZ:integrable M Z
    apply (subst square-integrable-imp-integrable[OF - intZsq])
    by auto
  then have i1: integrable M (λz. (Z z − expectation Z + u)^2)
    unfolding power2-sum power2-diff using intZsq
    by auto

  have intY:integrable M Y
    unfolding Y-def using intZ by auto
  have intYsq:integrable M (λz. Y z^2)
    unfolding Y-def power2-sum using intZsq intZ by auto

  have expectation Y = 0
    unfolding Y-def
    apply (subst Bochner-Integration.integral-add[OF intZ])
    using prob-space by auto

  then have expectation (λz. (Y z + u)^2) =
    expectation (λz. (Y z)^2) + u^2
    unfolding power2-sum
    apply (subst Bochner-Integration.integral-add[OF - -])
    using intY intYsq apply auto[2]
    apply (subst Bochner-Integration.integral-add[OF - -])
    using intY intYsq apply auto[2]
    using prob-space by auto
  then have ∗: expectation (λz. (Y z + u)^2) = variance Z + u^2
    unfolding Y-def by auto
```

```
  have
    prob {z ∈ space M. Z z − expectation Z ≥ a} =
    prob {z ∈ space M. Y z + u ≥ a + u}
    apply (intro arg-cong[where f = prob])
    using Y-def by auto
  also have ... ≤ prob {z ∈ space M. a + u ≤ |Y z + u|}
    apply (intro finite-measure-mono[OF - ev])
    by auto

  also have ... ≤ expectation (λz. (Y z + u)^2) / (a + u)^2
    apply (intro second-moment-method)
    unfolding Y-def using a u i1 by auto
  also have ... = ((variance Z) + u^2) / (a + u)^2
    using ∗ by auto
  also have ... = variance Z / (a ^2 + variance Z)
    unfolding u-def using a by (auto intro!: cantelli-arith)
  finally show ?thesis .
qed


corollary cantelli-inequality-neg:
  assumes [measurable]: random-variable borel Z
  assumes intZsq: integrable M (λz. Z z^2)
  assumes a: a > 0
  shows prob {z ∈ space M. Z z − expectation Z ≤ −a} ≤
    variance Z / (a^2 + variance Z)
proof −
  define nZ where [simp]: nZ = (λz. −Z z)
  have vnZ: variance nZ = variance Z
    unfolding nZ-def
    by (auto simp add: power2-commute)

  have 1: random-variable borel nZ by auto
  have 2: integrable M (λz. (nZ z)^2)
    using intZsq by auto
  from cantelli-inequality[OF 1 2 a]
  have prob {z ∈ space M. a ≤ nZ z − expectation nZ} ≤
    variance nZ / (a^2 + variance nZ)
    by auto
  thus ?thesis unfolding vnZ apply auto[1]
    by (smt (verit, del-insts) Collect-cong)
qed

end

end
```

5 Efron-Stein Inequality

In this section we verify the Efron-Stein inequality. The verified theorem is stated as Efron-Stein inequality for non-symmetric functions by Steele [8]. However most textbook refer to this version as "the Efron-Stein inequality". The original result that was shown by Efron and Stein is a tail bound for the variance of a symmetric functions of i.i.d. random variables [4].

theory *Efron-Stein-Inequality*
imports *Concentration-Inequalities-Preliminary*
begin

theorem *efron-stein-inequality-distr*:
fixes f :: - ⇒ *real*
assumes *finite I*
assumes ⋀*i*. $i \in I \Longrightarrow$ *prob-space* (*M i*)
assumes *integrable* (*PiM I M*) (λ*x*. *f x*^2) **and** *f-meas*: $f \in$ *borel-measurable* (*PiM I M*)
shows *prob-space.variance* (*PiM I M*) *f* ≤
(∑ *i*∈*I*. (∫ *x*. (*f* (λ*j*. *x* (*j*,*False*)) − *f* (λ*j*. *x* (*j*, *j*=*i*)))^2 ∂*PiM* (*I*×*UNIV*) (*M* ∘ *fst*))) / *2*
(**is** *?L* ≤ *?R*)
proof −
let *?M* = *PiM* (*I*×(*UNIV*::*bool set*)) (*M* ∘ *fst*)

have *prob*: *prob-space* (*PiM I M*)
using *assms*(*2*) **by** (*intro prob-space-PiM*) *auto*

interpret *prob-space ?M*
using *assms*(*2*) **by** (*intro prob-space-PiM*) *auto*

define *n* **where** *n* = *card I*

obtain *q* :: - ⇒ *nat* **where** *q*:*bij-betw q I* {..<*n*}
unfolding *n-def* **using** *ex-bij-betw-finite-nat*[*OF assms*(*1*)] *atLeast0LessThan*
by *auto*

let *?φ* = (λ*n x*. *f* (λ*j*. *x* (*j*, *q j* < *n*)))
let *?τ* = (λ*n x*. *f* (λ*j*. *x* (*j*, *q j* = *n*)))
let *?σ* = (λ*x*. *f* (λ*j*. *x* (*j*, *False*)))
let *?χ* = (λ*x*. *f* (λ*j*. *x* (*j*, *True*)))

have *meas-1*: (λω. *f* (*g* ω)) ∈ *borel-measurable ?M*
if *g* ∈ *Pi*$_M$ (*I* × *UNIV*) (*M* ∘ *fst*) →$_M$ *Pi*$_M$ *I M* **for** *g*
using *that* **by** (*intro measurable-compose*[*OF - f-meas*])

have *meas-2*: (λ*x j*. *x* (*j*, *h j*)) ∈ *?M* →$_M$ *Pi*$_M$ *I M* **for** *h*
proof −

```
    have ?thesis ⟷ (λx. (λj ∈ I. x (j, h j))) ∈ ?M →M PiM I M
      by (intro measurable-cong) (auto simp:space-PiM PiE-def exten-
sional-def)
    also have ... ⟷ True
      unfolding eq-True
     by (intro measurable-restrict measurable-PiM-component-rev) auto
    finally show ?thesis by simp
  qed

  have int-1: integrable ?M (λx. (g x − h x)^2)
    if integrable ?M (λx. (g x)^2)  integrable ?M (λx. (h x)^2)
    and g ∈ borel-measurable ?M h ∈ borel-measurable ?M
    for g h :: - ⇒ real
  proof −
    have integrable ?M (λx. (g x)^2 + (h x)^2 − 2 ∗ (g x ∗ h x))
    using that by (intro Bochner-Integration.integrable-add Bochner-Integration.integrable-diff
          integrable-mult-right cauchy-schwartz(1))
    thus ?thesis by (simp add:algebra-simps power2-eq-square)
  qed

 note meas-rules = borel-measurable-add borel-measurable-times borel-measurable-diff
    borel-measurable-power meas-1 meas-2

  have f-int: integrable (PiM I M) f
   by (intro finite-measure.square-integrable-imp-integrable[OF - f-meas
assms(3)]
        prob-space.finite-measure prob)
  moreover have integrable (PiM I M) (λx. f (restrict x I)) = inte-
grable (PiM I M) f
   by (intro Bochner-Integration.integrable-cong) (auto simp:space-PiM)
  ultimately have f-int-2: integrable (PiM I M) (λx. f (restrict x I))
by simp

  have cong: (∫ x. g (λj∈I. x (j, h j)) ∂?M) = (∫ x. g (λj. x (j, h j))
∂?M) (is ?L1 = ?R1)
    for g :: - ⇒ real and h
     by (intro Bochner-Integration.integral-cong arg-cong[where f=g]
refl)
       (auto simp add:space-PiM PiE-def extensional-def restrict-def)

  have lift: (∫ x. g x ∂PiM I M) = (∫ x. g (λj. x (j, h j)) ∂?M) (is
?L1 = ?R1)
    if g ∈ borel-measurable (PiM I M)
    for g :: - ⇒ real and h
  proof −
    let ?J = (λi. (i, h i)) ‘ I
    have ?R1 = (∫ x. g (λj ∈ I. x (j, h j)) ∂?M)
      by (intro cong[symmetric])
    also have ... = (∫ x. g x ∂distr ?M (PiM I (λi. (M∘fst) (i, h i)))
```

```
(λx. (λj ∈ I. x (j, h j))))
      using that
       by (intro integral-distr[symmetric] measurable-restrict measur-
able-component-singleton) auto
    also have ... = (∫ x. g x ∂PiM I (λi. (M ∘ fst) (i, h i)))
      using assms(2)
       by (intro arg-cong2[where f=integral^L] refl distr-PiM-reindex
inj-onI) auto
    also have ... = ?L1
      by auto
    finally show ?thesis
      by simp
  qed

  have lift-int: integrable ?M (λx. g (λj. x (j, h j))) if integrable (Pi_M
I M) g
    for g :: - ⇒ real and h
  proof −
    have 0:integrable (distr ?M (PiM I (λi. (M∘fst) (i, h i))) (λx. (λj
∈ I. x (j, h j)))) g
       using that assms(2) by (subst distr-PiM-reindex) (auto in-
tro:inj-onI)
    have integrable ?M (λx. g (λj∈I. x (j, h j)))
      by (intro integrable-distr[OF - 0] measurable-restrict measur-
able-component-singleton) auto
   moreover have integrable ?M (λx. g (λj∈I. x (j, h j))) ⟷ ?thesis
     by (intro Bochner-Integration.integrable-cong refl arg-cong[where
f=g] ext)
       (auto simp:PiE-def space-PiM extensional-def)
    ultimately show ?thesis
      by simp
  qed

   note int-rules = cauchy-schwartz(1) int-1 lift-int assms(3) f-int
f-int-2

  have (∫ x. g x ∂?M) = (∫ x. g (λ(j,v). x (j, v ≠ h j)) ∂?M) (is ?L1
= ?R1)
    if g ∈ borel-measurable ?M for g :: - ⇒ real and h
  proof −
    have ?L1 = (∫ x. g x ∂distr ?M (PiM (I×UNIV) (λi. (M ∘ fst)
(fst i, snd i ≠ h (fst i)))))
      (λx.(λi ∈ I×UNIV. x (fst i, snd i ≠ h (fst i))) ))
     by (subst distr-PiM-reindex) (auto intro:inj-onI assms(2) simp:comp-def)
    also have ... = (∫ x. g (λi ∈ I×UNIV. x (fst i, snd i ≠ h (fst i)))
∂?M)
       using that by (intro integral-distr measurable-restrict measur-
able-component-singleton)
        (auto simp:comp-def)
```

```
    also have ... = ?R1
      by (intro Bochner-Integration.integral-cong refl arg-cong[where
f=g] ext)
       (auto simp add:space-PiM PiE-def extensional-def restrict-def)
    finally show ?thesis
      by simp
  qed

  hence switch: (∫ x. g x ∂?M) = (∫ x. h x ∂?M)
    if ⋀x. h x = g (λ(j,v). x (j, v ≠ u j)) g ∈ borel-measurable ?M
    for g h :: - ⇒ real and u
    using that by simp

  have 1: (∫ x. (?σ x) * (?φ i x − ?φ (i+1) x) ∂?M) ≤ (∫ x. (?σ x
− ?τ i x)^2 ∂?M) / 2
    (is ?L1 ≤ ?R1)
    if i < n for i
  proof −
    have ?L1 = (∫ x. (?τ i x) * (?φ (i+1) x − ?φ i x) ∂?M)
      by (intro switch[of - - (λj. q j = i)] arg-cong2[where f=(*)]
          arg-cong2[where f=(−)] arg-cong[where f=f] ext meas-rules)
(auto intro:arg-cong)
     hence ?L1 = (?L1 + (∫ x. (?τ i x) * (?φ (i+1) x − ?φ i x)
∂?M)) / 2
      by simp
    also have ... = (∫ x. (?σ x) * (?φ i x − ?φ(i+1) x) + (?τ i x) *
(?φ(i+1) x − ?φ i x) ∂?M)/2
     by (intro Bochner-Integration.integral-add[symmetric] arg-cong2[where
f=(/)] refl
          int-rules meas-rules)
     also have ... = (∫ x. (?σ x − ?τ i x) * (?φ i x − ?φ(i+1) x)
∂?M)/2
     by (intro arg-cong2[where f=(/)] Bochner-Integration.integral-cong)
        (auto simp:algebra-simps)
    also have ...≤((∫ x. (?σ x−?τ i x)^2 ∂?M)powr(1/2)*(∫ x.(?φ i
x−?φ(i+1)x)^2 ∂?M) powr (1/2))/2
      by (intro divide-right-mono cauchy-schwartz meas-rules int-rules)
auto
     also have ...=((∫ x. (?σ x−?τ i x)^2 ∂?M)powr(1/2)*(∫ x.(?σ
x−?τ i x)^2 ∂?M) powr (1/2))/2
    by (intro arg-cong2[where f=(/)] arg-cong2[where f=(*)] arg-cong2[where
f=(powr)] refl
       switch[of - - (λj. q j < i)] arg-cong2[where f=power] arg-cong2[where
f=(−)]
         arg-cong[where f=f] ext meas-rules) (auto intro:arg-cong)
    also have ... = (∫ x. (?σ x−?τ i x)^2 ∂?M)/2
      by (simp add:powr-add[symmetric])
    finally show ?thesis by simp
  qed
```

have *indep-vars* $(M \circ fst)$ $(\lambda i\ \omega.\ \omega\ i)$ $(I \times UNIV)$
using *assms(2)* **by** (*intro proj-indep*) *auto*
hence *2*:*indep-var* $(Pi_M\ (I \times \{False\})\ (M \circ fst))$ $(\lambda x.\ \lambda j \in I \times \{False\}.\ x\ j)$
$(Pi_M\ (I \times \{True\})\ (M \circ fst))$ $(\lambda x.\ \lambda j \in I \times \{True\}.\ x\ j)$
by (*intro indep-var-restrict*[**where** $I = I \times UNIV$]) *auto*
have *indep-var*
$(Pi_M\ I\ M)$ $((\lambda x.\ (\lambda i \in I.\ x\ (i,\ False))) \circ (\lambda x.\ (\lambda j \in I \times \{False\}.\ x\ j)))$
$(Pi_M\ I\ M)$ $((\lambda x.\ (\lambda i \in I.\ x\ (i,\ True))) \circ (\lambda x.\ (\lambda j \in I \times \{True\}.\ x\ j)))$
by (*intro indep-var-compose*[*OF 2*] *measurable-restrict measurable-PiM-component-rev*) *auto*
hence *indep-var* $(Pi_M\ I\ M)$ $(\lambda x.\ (\lambda j \in I.\ x\ (j,\ False)))$ $(Pi_M\ I\ M)$ $(\lambda x.\ (\lambda j \in I.\ x\ (j,\ True)))$
unfolding *comp-def* **by** (*simp add*:*restrict-def cong*:*if-cong*)

hence *indep-var borel* $(f \circ (\lambda x.\ (\lambda j \in I.\ x\ (j,\ False))))$ *borel* $(f \circ (\lambda x.\ (\lambda j \in I.\ x\ (j,\ True))))$
by (*intro indep-var-compose*[*OF - assms*(*4*,*4*)]) *auto*
hence *indep*:*indep-var borel* $(\lambda x.\ f\ (\lambda j \in I.\ x\ (j,\ False)))$ *borel* $(\lambda x.\ f\ (\lambda j \in I.\ x\ (j,\ True)))$
by (*simp add*:*comp-def*)

have *3*: $\omega\ (j,\ q\ j = q\ i) = \omega\ (j,\ j = i)$ **if**
$\omega \in PiE\ (I \times UNIV)\ (\lambda i.\ space\ (M\ (fst\ i)))$ $i \in I$ **for** $i\ j\ \omega$
proof (*cases* $j \in I$)
case *True*
hence $(q\ j = q\ i) = (j = i)$
using *that inj-onD*[*OF bij-betw-imp-inj-on*[*OF q*]] **by** *blast*
thus *?thesis* **by** *simp*
next
case *False*
hence $\omega\ (j,\ a) = undefined$ **for** *a*
using *that* **unfolding** *PiE-def extensional-def* **by** *simp*
thus *?thesis* **by** *simp*
qed

have *?L* $= (\int x.\ (f\ x)\hat{}2\ \partial PiM\ I\ M) - (\int x.\ (f\ x)\ \partial PiM\ I\ M)\hat{}2$
by (*intro prob-space.variance-eq f-int assms*(*3*) *prob*)
also have ... $= (\int x.\ (f\ x)\hat{}2\ \partial PiM\ I\ M) - (\int x.\ f\ x\ \partial PiM\ I\ M) * (\int x.\ f\ x\ \partial PiM\ I\ M)$
by (*simp add*:*power2-eq-square*)
also have ... $= (\int x.\ (?\sigma\ x)\hat{}2\ \partial ?M) - (\int x.\ ?\sigma\ x\ \partial ?M) * (\int x.\ ?\chi\ x\ \partial ?M)$
by (*intro arg-cong2*[**where** $f = (-)$] *lift arg-cong2*[**where** $f = (*)$] *meas-rules f-meas*)
also have ... $= (\int x.\ (?\sigma\ x)\hat{}2\ \partial ?M) - (\int x.\ f\ (\lambda j \in I.\ x\ (j, False))$

$\partial ?M) * (\int x.\ f(\lambda j \in I.\ x(j, True))\ \partial ?M)$
by (*intro arg-cong2*[**where** $f=(-)$] *arg-cong2*[**where** $f=(*)$] *cong*[*symmetric*] *refl*)
also have ... = $(\int x.\ (?\sigma\ x) \hat{}2\ \partial ?M) - (\int x.\ f\ (\lambda j \in I.\ x\ (j, False)) *$ $f(\lambda j \in I.\ x(j, True))\ \ \partial ?M)$
by (*intro arg-cong2*[**where** $f=(-)$] *indep-var-lebesgue-integral*[*symmetric*] *refl int-rules indep*)
also have ... = $(\int x.\ (?\sigma\ x) * (?\varphi\ 0\ x)\ \partial ?M) - (\int x.\ (?\sigma\ x) * (?\varphi$ $n\ x)\ \ \partial ?M)$
using *bij-betw-apply*[*OF q*] **by** (*intro arg-cong2*[**where** $f=(-)$] *arg-cong2*[**where** $f=(*)$] *ext*
arg-cong[**where** $f=f$] *Bochner-Integration.integral-cong*)
(*auto simp:space-PiM power2-eq-square PiE-def extensional-def*)
also have ... = $(\sum i < n.\ (\int x.\ (?\sigma\ x) *\ \ (?\varphi\ i\ x)\ \ \partial ?M) -\ \ (\int x.$ $(?\sigma\ x) *\ \ (?\varphi\ (Suc\ i)\ x)\ \partial ?M))$
unfolding *power2-eq-square* **by** (*intro sum-lessThan-telescope'*[*symmetric*])
also have ... = $(\sum i < n.\ (\int x.\ (?\sigma\ x) *\ \ (?\varphi\ i\ x) - (?\sigma\ x) *\ \ (?\varphi$ $(Suc\ i)\ x)\ \partial ?M))$
by (*intro sum.cong Bochner-Integration.integral-diff*[*symmetric*] *int-rules meas-rules*) *auto*
also have ... = $(\sum i < n.\ (\int x.\ (?\sigma\ x) * (?\varphi\ i\ x -\ ?\varphi\ (i+1)\ x)$ $\partial ?M))$
by (*simp-all add:power2-eq-square algebra-simps*)
also have ... $\leq (\sum i< n.\ ((\int x.\ (?\sigma\ x -\ ?\tau\ i\ x) \hat{}2\ \partial ?M))\ /\ 2)$
by (*intro sum-mono 1*) *auto*
also have ... = $(\sum i \in I.\ ((\int x.\ (f\ (\lambda j.\ x\ (j, False)) - f\ (\lambda j.\ x\ (j, q$ $j=q\ i))) \hat{}2\ \partial ?M))/\ 2)$
by (*intro sum.reindex-bij-betw*[*OF q, symmetric*])
also have ... = $(\sum i \in I.\ ((\int x.\ (f\ (\lambda j.\ x\ (j, False)) -\ \ f\ (\lambda j.\ x\ (j, q$ $j=q\ i))) \hat{}2\ \partial ?M)))/2$
unfolding *sum-divide-distrib*[*symmetric*] **by** *simp*
also have ... = *?R*
using *inj-onD*[*OF bij-betw-imp-inj-on*[*OF q*]]
by (*intro arg-cong2*[**where** $f=(/)$] *arg-cong2*[**where** $f=(-)$] *arg-cong2*[**where** $f=power$]
arg-cong[**where** $f=f$] *Bochner-Integration.integral-cong sum.cong refl ext 3*)
(*auto simp add:space-PiM*)
finally show *?thesis*
by *simp*
qed

theorem (**in** *prob-space*) *efron-stein-inequality-classic*:
fixes f :: - $\Rightarrow$ *real*
assumes *finite I*
assumes *indep-vars* $(M' \circ fst)\ X\ (I \times (UNIV :: bool\ set))$
assumes $f \in$ *borel-measurable* $(PiM\ I\ M')$
assumes *integrable M* $(\lambda \omega.\ f\ (\lambda i \in I.\ X\ (i, False)\ \omega) \hat{}2)$
assumes $\bigwedge i.\ i \in I \Longrightarrow distr\ M\ (M'\ i)\ (X\ (i,\ True)) = distr\ M\ (M'$

```
i) (X (i, False))
  shows variance (λω. f (λi∈I. X (i,False) ω)) ≤
    (∑ j ∈ I. expectation (λω. (f (λi∈I. X (i,False) ω) − f (λi∈I. X
(i, i=j) ω))^2))/2
    (is ?L ≤ ?R)
proof −
  let ?D = distr M (PiM I M') (λω. λi∈I. X (i, False) ω)

  let ?M = PiM I (λi. distr M (M' i) (X (i,False)))
   let ?N = PiM (I × (UNIV::bool set)) ((λi. distr M (M' i) (X
(i,False))) ∘ fst)

  have rv: random-variable (M' i) (X (i, j)) if i ∈ I for i j
    using assms(2) that unfolding indep-vars-def by auto

  have proj-meas: (λx j. x (j, h j)) ∈ Pi_M (I × UNIV) (M' ∘ fst)
→_M Pi_M I M'
    for h :: - ⇒ bool
  proof −
    have ?thesis ⟷ (λx. (λj ∈ I. x (j, h j))) ∈ Pi_M (I × UNIV)
(M' ∘ fst) →_M Pi_M I M'
      by (intro measurable-cong) (auto simp:space-PiM PiE-def exten-
sional-def)
    also have ... ⟷ True
      unfolding eq-True
     by (intro measurable-restrict measurable-PiM-component-rev) auto
    finally show ?thesis by simp
  qed

  note meas-rules = borel-measurable-add borel-measurable-times borel-measurable-diff
proj-meas
   borel-measurable-power assms(3) measurable-restrict measurable-compose[OF
- assms(3)]

  have indep-vars ((M' ∘ fst) ∘ (λi. (i, False))) (λi. X (i, False)) I
   by (intro indep-vars-reindex indep-vars-subset[OF assms(2)] inj-onI)
auto
  hence indep-vars M' (λi. X (i, False)) I by (simp add: comp-def)
  hence 0:?D = PiM I (λi. distr M (M' i) (X (i,False)))
    by (intro iffD1[OF indep-vars-iff-distr-eq-PiM'] rv)

  have distr M (M' (fst x)) (X (fst x, False)) = distr M (M' (fst x))
(X x)
    if x ∈ I × UNIV for x
    using that assms(5) by (cases x, cases snd x) auto

  hence 1: ?N = PiM (I × UNIV) (λi. distr M ((M' ∘ fst) i) (X i))
    using assms(3) by (intro PiM-cong refl) (simp add:comp-def)
  also have ... = distr M (PiM (I × UNIV) (M' ∘ fst)) (λx. λi∈I ×
```

UNIV. *X i x*)
using *rv* **by** (*intro iffD1*[*OF indep-vars-iff-distr-eq-PiM''*, *symmetric*] *assms*(*2*)) *auto*
finally have *2*:*?N* = *distr M* (*PiM* (*I* × *UNIV*) (*M'* ∘ *fst*)) (λ*x*. λ*i*∈*I* × *UNIV*. *X i x*)
by *simp*

have *3*: *integrable* (*Pi*$_M$ *I* (λ*i*. *distr M* (*M' i*) (*X* (*i*, *False*)))) (λ*x*. (*f x*)^2)
unfolding *0*[*symmetric*] **by** (*intro iffD2*[*OF integrable-distr-eq*] *meas-rules assms rv*)

have *?L* = (∫ *x*. (*f x* − *expectation* (λω. *f* (λ*i*∈*I*. *X* (*i*,*False*) ω)))^2 ∂*?D*)
using *rv* **by** (*intro integral-distr*[*symmetric*] *meas-rules measurable-restrict*) *auto*
also have ... = *prob-space.variance ?D f*
by (*intro arg-cong*[**where** *f*=*integral*^{L} *?D*] *arg-cong2*[**where** *f*=(−)] *arg-cong2*[**where** *f*=*power*]
refl ext integral-distr[*symmetric*] *measurable-restrict rv assms*(*3*))
also have ... = *prob-space.variance ?M f*
unfolding *0* **by** *simp*
also have ... ≤ (∑ *i*∈*I*. (∫ *x*. (*f* (λ*j*. *x* (*j*, *False*)) − *f* (λ*j*. *x* (*j*, *j* = *i*)))^2 ∂*?N*)) / *2*
using *assms*(*3*) **by** (*intro efron-stein-inequality-distr prob-space-distr rv assms*(*1*) *3*) *auto*
also have ... = (∑ *i*∈*I*. *expectation* (λω. (*f* (λ*j*. (λ*i*∈*I*×*UNIV*. *X i* ω) (*j*, *False*)) −
f (λ*j*. (λ*i*∈*I*×*UNIV*. *X i* ω) (*j*, *j*=*i*)))^2)) / *2*
using *rv* **unfolding** *2*
by (*intro sum.cong arg-cong2*[**where** *f*=(/)] *integral-distr refl meas-rules*) *auto*
also have ... = *?R*
by (*simp add*:*restrict-def*)
finally show *?thesis*
by *simp*
qed

end

6 McDiarmid's inequality

In this section we verify McDiarmid's inequality [6, Lemma 1.2]. In the source and also further sources sometimes refer to the result as the "independent bounded differences" inequality.

theory *McDiarmid-Inequality*
imports *Concentration-Inequalities-Preliminary*
begin

lemma *Collect-restr-cong*:
assumes $A = B$
assumes $\bigwedge x.\ x \in A \Longrightarrow P\ x = Q\ x$
shows $\{x \in A.\ P\ x\} = \{x \in B.\ Q\ x\}$
using *assms* **by** *auto*

lemma *ineq-chain*:
fixes $h :: nat \Rightarrow real$
assumes $\bigwedge i.\ i < n \Longrightarrow h\ (i+1) \leq h\ i$
shows $h\ n \leq h\ 0$
using *assms* **by** (*induction n*) *force+*

lemma *restrict-subset-eq*:
assumes $A \subseteq B$
assumes *restrict f B = restrict g B*
shows *restrict f A = restrict g A*
using *assms* **unfolding** *restrict-def* **by** (*meson subsetD*)

Bochner Integral version of Hoeffding's Lemma using *interval-bounded-random-variable.Hoeffdin*

lemma (**in** *prob-space*) *Hoeffdings-lemma-bochner*:
assumes $l > 0$ **and** *E0*: *expectation f = 0*
assumes *random-variable borel f*
assumes *AE x in M. f x* $\in$ {*a..b*::*real*}
shows *expectation* $(\lambda x.\ exp\ (l * f\ x)) \leq exp\ (l^2 * (b - a)^2\ /\ 8)$ (**is** *?L* $\leq$ *?R*)
proof −
interpret *interval-bounded-random-variable M f a b*
using *assms* **by** (*unfold-locales*) *auto*

have *integrable M* $(\lambda x.\ exp\ (l * f\ x))$
using *assms*(*1*,*3*,*4*) **by** (*intro integrable-const-bound*[**where** *B*=*exp* $(l * b)$]) *simp-all*

hence *ennreal* (*?L*) = $(\int^+ x.\ exp\ (l * f\ x)\ \partial M)$
by (*intro nn-integral-eq-integral*[*symmetric*]) *auto*
also have ... $\leq$ *ennreal* (*?R*)
by (*intro Hoeffdings-lemma-nn-integral-0 assms*)
finally have *0*:*ennreal* (*?L*) $\leq$ *ennreal ?R*
by *simp*
show *?thesis*
proof (*cases ?L* $\geq$ *0*)
case *True*
thus *?thesis* **using** *0* **by** *simp*
next
case *False*
hence *?L* $\leq$ *0* **by** *simp*
also have ... $\leq$ *?R* **by** *simp*
finally show *?thesis* **by** *simp*

```
  qed
qed

lemma (in prob-space) Hoeffdings-lemma-bochner-2:
  assumes l > 0 and E0: expectation f = 0
  assumes random-variable borel f
  assumes ⋀x y. {x,y} ⊆ space M ⟹ |f x − f y| ≤ (c::real)
  shows   expectation (λx. exp (l * f x)) ≤ exp (l^2 * c^2 / 8) (is ?L
≤ ?R)
proof −
  define a :: real where a = (INF x ∈ space M. f x)
  define b :: real where b = a+c

  obtain ω where ω:ω ∈ space M using not-empty by auto
  hence 0:f ‘ space M ≠ {} by auto
  have 1: c = b − a unfolding b-def by simp

  have bdd-below (f ‘ space M)
    using ω assms(4) unfolding abs-le-iff
   by (intro bdd-belowI[where m=f ω − c]) (auto simp add:algebra-simps)
   hence f x ≥ a if x ∈ space M for x unfolding a-def by (intro
cINF-lower that)
  moreover have f x ≤ b if x-space: x ∈ space M for x
  proof (rule ccontr)
    assume ¬(f x ≤ b)
    hence a:f x > a + c unfolding b-def by simp
    have f y ≥ f x − c if y ∈ space M for y
        using that x-space assms(4) unfolding abs-le-iff by (simp
add:algebra-simps)
    hence f x − c ≤ a unfolding a-def using cInf-greatest[OF 0] by
auto
    thus False using a by simp
  qed
  ultimately have f x ∈ {a..b} if x ∈ space M for x using that by
auto
  hence AE x in M. f x ∈ {a..b} by simp
 thus ?thesis unfolding 1 by (intro Hoeffdings-lemma-bochner assms(1,2,3))
qed

lemma (in prob-space) Hoeffdings-lemma-bochner-3:
  assumes expectation f = 0
  assumes random-variable borel f
  assumes ⋀x y. {x,y} ⊆ space M ⟹ |f x − f y| ≤ (c::real)
  shows   expectation (λx. exp (l * f x)) ≤ exp (l^2 * c^2 / 8) (is ?L
≤ ?R)
proof −
  consider (a) l > 0 | (b) l = 0 | (c) l < 0
    by argo
  then show ?thesis
```

proof (*cases*)
case *a* **thus** *?thesis* **by** (*intro Hoeffdings-lemma-bochner-2 assms*) *auto*
next
case *b* **thus** *?thesis* **by** *simp*
next
case *c*
have *?L* = *expectation* (λ*x*. *exp* ((−*l*) ∗ (−*f x*))) **by** *simp*
also have ... ≤ *exp* ((−*l*)^2 ∗ $c^2/8$) **using** *c assms* **by** (*intro Hoeffdings-lemma-bochner-2*) *auto*
also have ... = *?R* **by** *simp*
finally show *?thesis* **by** *simp*
qed
qed

Version of *product-sigma-finite.product-integral-singleton* without the condition that $M\ i$ has to be sigma finite for all i:

lemma *product-integral-singleton*:
fixes *f* :: - ⇒ -::{*banach*, *second-countable-topology*}
assumes *sigma-finite-measure* (*M i*)
assumes *f* ∈ *borel-measurable* (*M i*)
shows (∫ *x*. *f* (*x i*) ∂(*PiM* {*i*} *M*)) = (∫ *x*. *f x* ∂(*M i*)) (**is** *?L* = *?R*)
proof −
define *M'* **where** *M' j* = (*if j*=*i then M i else count-space* {*undefined*})
for *j*

interpret *product-sigma-finite M'*
using *assms*(*1*) **unfolding** *product-sigma-finite-def M'-def*
by (*auto intro*!:*sigma-finite-measure-count-space-finite*)

have *?L* = ∫ *x*. *f* (*x i*) ∂(*PiM* {*i*} *M'*)
by (*intro Bochner-Integration.integral-cong PiM-cong*) (*simp-all add*:*M'-def*)
also have ... = (∫ *x*. *f x* ∂(*M' i*))
using *assms*(*2*) **by** (*intro product-integral-singleton*) (*simp add*:*M'-def*)
also have ... = *?R*
by (*intro Bochner-Integration.integral-cong PiM-cong*) (*simp-all add*:*M'-def*)
finally show *?thesis* **by** *simp*
qed

Version of *product-sigma-finite.product-integral-fold* without the condition that $M\ i$ has to be sigma finite for all i:

lemma *product-integral-fold*:
fixes *f* :: - ⇒ -::{*banach*, *second-countable-topology*}
assumes ⋀*i*. *i* ∈ *I* ∪ *J* ⟹ *sigma-finite-measure* (*M i*)
assumes *I* ∩ *J* = {}
assumes *finite I*

assumes *finite J*
assumes *integrable* (*PiM* (*I* ∪ *J*) *M*) *f*
shows (∫ *x*. *f x* ∂*PiM* (*I* ∪ *J*) *M*) = (∫ *x*. (∫ *y*. *f* (*merge I J*(*x*,*y*)) ∂*PiM J M*) ∂*PiM I M*) (**is** *?L* = *?R*)
and *integrable* (*PiM I M*) (λ*x*. (∫ *y*. *f* (*merge I J*(*x*,*y*)) ∂*PiM J M*)) (**is** *?I*)
and *AE x in PiM I M*. *integrable* (*PiM J M*) (λ*y*. *f* (*merge I J*(*x*,*y*))) (**is** *?T*)
proof −
define *M′* **where** *M′ i* = (*if i* ∈ *I* ∪ *J then M i else count-space* {*undefined*}) **for** *i*

interpret *product-sigma-finite M′*
using *assms*(*1*) **unfolding** *product-sigma-finite-def M′-def*
by (*auto intro*!:*sigma-finite-measure-count-space-finite*)

interpret *pair-sigma-finite* Pi_M *I M′* Pi_M *J M′*
using *assms*(*3*,*4*) *sigma-finite* **unfolding** *pair-sigma-finite-def* **by** *blast*

have *0*: *integrable* (Pi_M (*I* ∪ *J*) *M′*) *f* = *integrable* (Pi_M (*I* ∪ *J*) *M*) *f*
by (*intro Bochner-Integration.integrable-cong PiM-cong*) (*simp-all add*:*M′-def*)

have *?L* = (∫ *x*. *f x* ∂*PiM* (*I* ∪ *J*) *M′*)
by (*intro Bochner-Integration.integral-cong PiM-cong*) (*simp-all add*:*M′-def*)
also have ... = (∫ *x*. (∫ *y*. *f* (*merge I J* (*x*,*y*)) ∂*PiM J M′*) ∂*PiM I M′*)
using *assms*(*5*) **by** (*intro product-integral-fold assms*(*2*,*3*,*4*)) (*simp add*:*0*)
also have ... = *?R*
by (*intro Bochner-Integration.integral-cong PiM-cong*) (*simp-all add*:*M′-def*)
finally show *?L* = *?R* **by** *simp*

have *integrable* (Pi_M (*I* ∪ *J*) *M′*) *f* = *integrable* (*PiM I M′* ⨂$_M$ *PiM J M′*) (λ*x*. *f* (*merge I J x*))
using *assms*(*5*) **apply** (*subst distr-merge*[*OF assms*(*2*,*3*,*4*),*symmetric*])
by (*intro integrable-distr-eq*) (*simp-all add*:*0*[*symmetric*])
hence *1*:*integrable* (*PiM I M′* ⨂$_M$ *PiM J M′*) (λ*x*. *f* (*merge I J x*))
using *assms*(*5*) *0* **by** *simp*

hence *integrable* (*PiM I M′*) (λ*x*. (∫ *y*. *f* (*merge I J*(*x*,*y*)) ∂*PiM J M′*)) (**is** *?I′*)
by (*intro integrable-fst′*) *auto*
moreover have *?I′* = *?I*

by (*intro Bochner-Integration.integrable-cong PiM-cong ext Bochner-Integration.integral-cong*)
(*simp-all add:M'-def*)
ultimately show *?I*
by *simp*

have *AE x in* Pi_M *I M'. integrable* (Pi_M *J M'*) (λ*y*. *f* (*merge I J* (*x*, *y*))) (**is** *?T'*)
by (*intro AE-integrable-fst'*[*OF 1*])
moreover have *?T' = ?T*
by (*intro arg-cong2*[**where** *f=almost-everywhere*] *PiM-cong ext Bochner-Integration.integrable-cong*)
(*simp-all add:M'-def*)
ultimately show *?T*
by *simp*
qed

lemma *product-integral-insert*:
fixes *f* :: - ⇒ -::{*banach*, *second-countable-topology*}
assumes ⋀*k*. *k* ∈ {*i*} ∪ *J* ⟹ *sigma-finite-measure* (*M k*)
assumes *i* ∉ *J*
assumes *finite J*
assumes *integrable* (*PiM* (*insert i J*) *M*) *f*
shows (∫ *x*. *f x* ∂*PiM* (*insert i J*) *M*) = (∫ *x*. (∫ *y*. *f* (*y*(*i* := *x*)) ∂*PiM J M*) ∂*M i*) (**is** *?L = ?R*)
proof −
note *meas-cong = iffD1*[*OF measurable-cong*]

have *integrable* (*PiM* {*i*} *M*) (λ*x*. (∫ *y*. *f* (*merge* {*i*} *J* (*x*,*y*)) ∂*PiM J M*))
using *assms* **by** (*intro product-integral-fold*) *auto*
hence *0*:(λ*x*. (∫ *y*. *f* (*merge* {*i*} *J* (*x*,*y*)) ∂*PiM J M*)) ∈ *borel-measurable* (*PiM* {*i*} *M*)
using *borel-measurable-integrable* **by** *simp*
have *1*:(λ*x*. (∫ *y*. *f* (*y*(*i* := (*x i*))) ∂*PiM J M*)) ∈ *borel-measurable* (*PiM* {*i*} *M*)
by (*intro meas-cong*[*OF - 0*] *Bochner-Integration.integral-cong arg-cong*[**where** *f=f*])
(*auto simp add:space-PiM merge-def fun-upd-def PiE-def extensional-def*)
have (λ*x*. (∫ *y*. *f* (*y*(*i* := (λ*i*∈{*i*}. *x*) *i*)) ∂*PiM J M*)) ∈ *borel-measurable* (*M i*)
by (*intro measurable-compose*[*OF - 1*, **where** *f*=(λ*x*. (λ*i*∈{*i*}. *x*))] *measurable-restrict*) *auto*
hence *2*:(λ*x*. (∫ *y*. *f* (*y*(*i* := *x*)) ∂*PiM J M*)) ∈ *borel-measurable* (*M i*) **by** *simp*

have *?L* = (∫ *x*. *f x* ∂*PiM* ({*i*} ∪ *J*) *M*) **by** *simp*
also have ... = (∫ *x*. (∫ *y*. *f* (*merge* {*i*} *J* (*x*,*y*)) ∂*PiM J M*) ∂*PiM* {*i*} *M*)

using *assms(2,4)* **by** (*intro product-integral-fold assms(1,3)*) *auto*
also have ... = ($\int$ *x*. ($\int$ *y*. *f* (*y*(*i* := (*x i*))) *∂PiM J M*) *∂PiM* {*i*} *M*)
by (*intro Bochner-Integration.integral-cong refl arg-cong*[**where** *f*=*f*])
(*auto simp add*:*space-PiM merge-def fun-upd-def PiE-def extensional-def*)
also have ... = *?R*
using *assms(1,4)* **by** (*intro product-integral-singleton assms(1) 2*) *auto*
finally show *?thesis* **by** *simp*
qed

lemma *product-integral-insert-rev*:
fixes *f* :: - ⇒ -::{*banach*, *second-countable-topology*}
assumes ⋀*k*. *k* ∈ {*i*} ∪ *J* ⟹ *sigma-finite-measure* (*M k*)
assumes *i* ∉ *J*
assumes *finite J*
assumes *integrable* (*PiM* (*insert i J*) *M*) *f*
shows ($\int$ *x*. *f x ∂PiM* (*insert i J*) *M*) = ($\int$ *y*. ($\int$ *x*. *f* (*y*(*i* := *x*)) *∂M i*) *∂PiM J M*) (**is** *?L* = *?R*)
proof −
have *?L* = ($\int$ *x*. *f x ∂PiM* (*J* ∪ {*i*}) *M*) **by** *simp*
also have ... = ($\int$ *x*. ($\int$ *y*. *f* (*merge J* {*i*} (*x*,*y*)) *∂PiM* {*i*} *M*) *∂PiM J M*)
using *assms(2,4)* **by** (*intro product-integral-fold assms(1,3)*) *auto*
also have ... = ($\int$ *x*. ($\int$ *y*. *f* (*x*(*i* := (*y i*))) *∂PiM* {*i*} *M*) *∂PiM J M*)
unfolding *merge-singleton*[*OF assms(2)*]
by (*intro Bochner-Integration.integral-cong refl arg-cong*[**where** *f*=*f*])
(*metis PiE-restrict assms(2) restrict-upd space-PiM*)
also have ... = *?R*
using *assms(1,4)* **by** (*intro Bochner-Integration.integral-cong product-integral-singleton*) *auto*
finally show *?thesis* **by** *simp*
qed

lemma *merge-empty*[*simp*]:
merge {} *I* (*y*,*x*) = *restrict x I*
merge I {} (*y*,*x*) = *restrict y I*
unfolding *merge-def restrict-def* **by** *auto*

lemma *merge-cong*:
assumes *restrict x1 I* = *restrict x2 I*
assumes *restrict y1 J* = *restrict y2 J*
shows *merge I J* (*x1*,*y1*) = *merge I J* (*x2*,*y2*)
using *assms* **unfolding** *merge-def restrict-def*
by (*intro ext*) (*smt* (*verit*, *best*) *case-prod-conv*)

lemma *restrict-merge*:
restrict (*merge I J x*) *K* = *merge* (*I* ∩ *K*) (*J* ∩ *K*) *x*
unfolding *restrict-def merge-def* **by** (*intro ext*) (*auto simp*:*case-prod-beta*)

lemma *map-prod-measurable*:
assumes $f \in M \rightarrow_M M'$
assumes $g \in N \rightarrow_M N'$
shows *map-prod f g* ∈ $M \bigotimes_M N \rightarrow_M M' \bigotimes_M N'$
using *assms* **by** (*subst measurable-pair-iff*) *simp*

lemma *mc-diarmid-inequality-aux*:
fixes *f* :: (*nat* ⇒ *'a*) ⇒ *real*
fixes *n* :: *nat*
assumes ⋀*i*. *i* < *n* ⟹ *prob-space* (*M i*)
assumes ⋀*i x y*. *i*<*n* ⟹ {*x*,*y*} ⊆ *space* (*PiM* {..<*n*} *M*) ⟹ (∀ *j*∈{..<*n*}−{*i*}. *x j*=*y j*) ⟹ |*f x*−*f y*|≤*c i*
assumes *f-meas*: *f* ∈ *borel-measurable* (*PiM* {..<*n*} *M*) **and** *ε-gt-0*: *ε* >*0*
shows $\mathcal{P}(\omega$ *in PiM* {..<*n*} *M*. *f ω* − (∫ *ξ*. *f ξ* ∂*PiM* {..<*n*} *M*) ≥ *ε*) ≤ *exp* (−(*2*∗*ε*^*2*)/(∑ *i*<*n*. (*c i*)^*2*))
(**is** *?L* ≤ *?R*)
proof −
define *h* **where** *h k* = (λ*ξ*. (∫ *ω*. *f* (*merge* {..<*k*} {*k*..<*n*} (*ξ*, *ω*)) ∂*PiM* {*k*..<*n*} *M*)) **for** *k*

define *t* :: *real* **where** *t* = *4* ∗ *ε* / (∑ *i*<*n*. (*c i*)^*2*)

define *V* **where** *V i ξ* = *h* (*Suc i*) *ξ* − *h i ξ* **for** *i ξ*

obtain *x0* **where** *x0*:*x0* ∈ *space* (*PiM* {..<*n*} *M*)
using *prob-space.not-empty*[*OF prob-space-PiM*] *assms*(*1*) **by** *fast-force*

have *delta*: |*f x* − *f y*| ≤ *c i* **if** *i* < *n*
x ∈ *PiE* {..<*n*} (λ*i*. *space* (*M i*)) *y* ∈ *PiE* {..<*n*} (λ*i*. *space* (*M i*))
restrict x ({..<*n*}−{*i*}) = *restrict y* ({..<*n*}−{*i*})
for *x y i*
proof (*rule assms*(*2*)[*OF that*(*1*)], *goal-cases*)
case *1*
then show *?case* **using** *that*(*2*,*3*) **unfolding** *space-PiM* **by** *auto*
next
case *2*
then show *?case* **using** *that*(*4*) **by** (*intro ballI*) (*metis restrict-apply'*)
qed

have *c-ge-0*: *c j* ≥ *0* **if** *j* < *n* **for** *j*
proof −

```
    have 0 ≤ |f x0 − f x0| by simp
   also have ... ≤ c j using x0 unfolding space-PiM by (intro delta
that) auto
    finally show ?thesis by simp
  qed
  hence sum-c-ge-0: (∑ i<n. (c i)^2) ≥ 0 by (meson sum-nonneg
zero-le-power2)

  hence t-ge-0: t ≥ 0 using ε-gt-0 unfolding t-def by simp

  note borel-rules =
   borel-measurable-sum measurable-compose[OF - borel-measurable-exp]
    borel-measurable-times

  note int-rules =
    prob-space-PiM assms(1) borel-rules
    prob-space.integrable-bounded bounded-intros
  have h-n: h n ξ = f ξ if ξ ∈ space (PiM {..<n} M) for ξ
  proof −
    have h n ξ = (∫ ω. f (λi∈{..<n}. ξ i) ∂PiM {} M)
      unfolding h-def using leD
     by (intro Bochner-Integration.integral-cong PiM-cong arg-cong[where
f=f] restrict-cong)
       auto
    also have ... = f (restrict ξ {..<n})
      unfolding PiM-empty by simp
    also have ... = f ξ
      using that unfolding space-PiM PiE-def
      by (simp add: extensional-restrict)
    finally show ?thesis
      by simp
  qed

  have h-0: h 0 ξ = (∫ ω. f ω ∂PiM {..<n} M) for ξ
   unfolding h-def by (intro Bochner-Integration.integral-cong PiM-cong
refl)
     (simp-all add:space-PiM atLeast0LessThan)

  have h-cong: h j ω = h j ξ if restrict ω {..<j} = restrict ξ {..<j}
for j ω ξ
    using that unfolding h-def
     by (intro Bochner-Integration.integral-cong refl arg-cong[where
f=f] merge-cong) auto

  have h-meas: h i ∈ borel-measurable (PiM I M) if i ≤ n {..<i} ⊆ I
for i I
  proof −
    have 0: {..<n} = {..<i} ∪ {i..<n}
      using that(1) by auto
```

have *1*: *merge* {..<*i*} {*i*..<*n*} = *merge* {..<*i*} {*i*..<*n*} ∘ *map-prod* (λ*x*. *restrict x* {..<*i*}) *id*
unfolding *merge-def map-prod-def restrict-def comp-def*
by (*intro ext*) (*auto simp*:*case-prod-beta*′)

have *merge* {..<*i*} {*i*..<*n*} ∈ Pi_M *I M* ⨂$_M$ Pi_M {*i*..<*n*} *M* →$_M$ Pi_M {..<*n*} *M*
unfolding *0* **by** (*subst 1*) (*intro measurable-comp*[*OF - measurable-merge*] *map-prod-measurable*
measurable-ident measurable-restrict-subset that(*2*))
hence (λ*x*. *f* (*merge* {..<*i*} {*i*..<*n*} *x*)) ∈ *borel-measurable* (Pi_M *I M* ⨂$_M$ Pi_M {*i*..<*n*} *M*)
by (*intro measurable-compose*[*OF - f-meas*])
thus *?thesis*
unfolding *h-def* **by** (*intro sigma-finite-measure.borel-measurable-lebesgue-integral prob-space-imp-sigma-finite prob-space-PiM assms*(*1*)) (*auto simp*:*case-prod-beta*′)
qed

have *merge-space-aux*:*merge* {..<*j*} {*j*..<*n*} *u* ∈ (Π$_E$ *i*∈{..<*n*}. *space* (*M i*))
if *j* ≤ *n fst u* ∈ *Pi* {..<*j*} (λ*i*. *space* (*M i*)) *snd u* ∈ *Pi* {*j*..<*n*} (λ*i*. *space* (*M i*))
for *u j*
proof −
have *merge* {..<*j*} {*j*..<*n*} (*fst u*, *snd u*) ∈ (*PiE* ({..<*j*} ∪ {*j*..<*n*}) (λ*i*. *space* (*M i*)))
using *that* **by** (*intro iffD2*[*OF PiE-cancel-merge*]) *auto*
also have ... = (Π$_E$ *i*∈{..<*n*}. *space* (*M i*))
using *that* **by** (*intro arg-cong2*[**where** *f*=*PiE*] *refl*) *auto*
finally show *?thesis* **by** *simp*
qed

have *merge-space*:*merge* {..<*j*} {*j*..<*n*} (*u*, *v*) ∈ (Π$_E$ *i*∈{..<*n*}. *space* (*M i*))
if *j* ≤ *n u* ∈ *PiE* {..<*j*} (λ*i*. *space* (*M i*)) *v* ∈ *PiE* {*j*..<*n*} (λ*i*. *space* (*M i*))
for *u v j*
using *that* **by** (*intro merge-space-aux*) (*simp-all add*:*PiE-def*)

have *delta*′: |*f x* − *f y*| ≤ (∑ *i*<*n*. *c i*)
if *x* ∈ *PiE* {..<*n*} (λ*i*. *space* (*M i*)) *y* ∈ *PiE* {..<*n*} (λ*i*. *space* (*M i*)) **for** *x y*
proof −
define *m* **where** *m i* = *merge* {..<*i*} {*i*..<*n*} (*x*,*y*) **for** *i*

have *0*: *z* ∈ *Pi I* (λ*i*. *space* (*M i*)) **if** *z* ∈ *PiE* {..<*n*} (λ*i*. *space* (*M i*))

```
    I ⊆ {..<n} for z I
    using that unfolding PiE-def by auto

  have 3: {..<Suc i} ∩ ({..<n} − {i}) = {..<i}
    {Suc i..<n} ∩ ({..<n} − {i}) = {Suc i..<n}
    {..<i} ∩ ({..<n} − {i}) = {..<i}
    {i..<n} ∩ ({..<n} − {i}) = {Suc i..<n}
    if i < n for i
    using that by auto

  have |f x − f y| = |f (m n) − f(m 0)|
    using that unfolding m-def by (simp add:atLeast0LessThan)
  also have ... = |∑ i < n. f (m (Suc i)) − f (m i)|
    by (subst sum-lessThan-telescope) simp
  also have ... ≤ (∑ i < n. |f (m (Suc i)) − f (m i)|)
    by simp
  also have ... ≤ (∑ i < n. c i)
   using that unfolding m-def by (intro delta sum-mono merge-space-aux
0 subsetI)
      (simp-all add:restrict-merge 3)
  finally show ?thesis
    by simp
 qed

 have norm (f x) ≤ norm (f x0) + sum c {..<n} if x ∈ space (Pi_M
{..<n} M) for x
 proof −
  have |f x − f x0| ≤ sum c {..<n}
    using x0 that unfolding space-PiM by (intro delta') auto
  thus ?thesis
    by simp
 qed
 hence f-bounded: bounded (f ‘ space (PiM {..<n} M))
  by (intro boundedI[where B=norm (f x0) + (∑ i<n. c i)]) auto

 have f-merge-bounded:
   bounded ((λω. (f (merge {..<j} {j..<n} (u, ω)))) ‘ space (Pi_M
{j..<n} M))
  if j ≤ n u ∈ PiE {..<j} (λi. space (M i)) for u j
 proof −
  have (λω. merge {..<j} {j..<n} (u, ω)) ‘ space (Pi_M {j..<n} M)
⊆ space (Pi_M {..<n} M)
    using that unfolding space-PiM
    by (intro image-subsetI merge-space) auto
  thus ?thesis
   by (subst image-image[of f,symmetric]) (intro bounded-subset[OF
f-bounded] image-mono)
 qed
```

have *f-merge-meas-aux*:
(λω. *f* (*merge* {..<*j*} {*j*..<*n*} (*u*, ω))) ∈ *borel-measurable* (*Pi*$_M$ {*j*..<*n*} *M*)
if *j* ≤ *n* *u* ∈ *Pi* {..<*j*} (λ*i*. *space* (*M i*)) **for** *j u*
proof −

have *0*: {..<*n*} = {..<*j* } ∪ {*j* ..<*n*}
using *that*(*1*) **by** *auto*

have *1*: *merge* {..<*j*} {*j*..<*n*} (*u*,ω) = *merge* {..<*j*} {*j*..<*n*} (*restrict u* {..<*j*},ω) **for** ω
by (*intro merge-cong*) *auto*

have (λω. *merge* {..<*j*} {*j*..<*n*} (*u*, ω)) ∈ *Pi*$_M$ {*j*..<*n*} *M* →$_M$ *Pi*$_M$ {..<*n*} *M*
using *that* **unfolding** *0 1*
by (*intro measurable-compose*[*OF* - *measurable-merge*] *measurable-Pair1′*)
(*simp add*:*space-PiM*)
thus *?thesis*
by (*intro measurable-compose*[*OF* - *f-meas*])
qed

have *f-merge-meas*: (λω. *f* (*merge* {..<*j*} {*j*..<*n*} (*u*, ω))) ∈ *borel-measurable* (*Pi*$_M$ {*j*..<*n*} *M*)
if *j* ≤ *n* *u* ∈ *PiE* {..<*j*} (λ*i*. *space* (*M i*)) **for** *j u*
using *that* **unfolding** *PiE-def* **by** (*intro f-merge-meas-aux*) *auto*

have *h-bounded*: *bounded* (*h i* ‘ *space* (*PiM I M*))
if *h-bounded-assms*: *i* ≤ *n* {..<*i*} ⊆ *I* **for** *i I*
proof −
have *merge* {..<*i*} {*i*..<*n*} *x* ∈ *space* (*Pi*$_M$ {..<*n*} *M*)
if *x* ∈ (Π$_E$ *i*∈*I*. *space* (*M i*)) × (Π$_E$ *i*∈{*i*..<*n*}. *space* (*M i*)) **for** *x*
using *that h-bounded-assms* **unfolding** *space-PiM* **by** (*intro merge-space-aux*)
(*auto simp*: *PiE-def mem-Times-iff*)
hence *bounded* ((λ*x*. *f* (*merge* {..<*i*} {*i*..<*n*} *x*)) ‘
((Π$_E$ *i*∈*I*. *space* (*M i*)) × (Π$_E$ *i*∈{*i*..<*n*}. *space* (*M i*))))
by (*subst image-image*[*of f*,*symmetric*])
(*intro bounded-subset*[*OF f-bounded*] *image-mono image-subsetI*)
thus *?thesis*
using *that* **unfolding** *h-def*
by (*intro prob-space.finite-measure finite-measure.bounded-int int-rules*)
(*auto simp*:*space-PiM PiE-def*)
qed

have *V-bounded*: *bounded* (*V i* ‘ *space* (*PiM I M*))

```
    if i < n {..<i+1} ⊆ I for i I
    using that unfolding V-def by (intro bounded-intros h-bounded)
auto

  have V-upd-bounded: bounded ((λx. V j (ξ(j := x))) ‘ space (M j))
    if V-upd-bounded-assms: ξ ∈ space (Pi_M {..<j} M) j < n for j ξ
  proof −
    have ξ(j := v) ∈ space (Pi_M {..<j + 1} M) if v ∈ space (M j)
for v
      using V-upd-bounded-assms that unfolding space-PiM PiE-def
extensional-def Pi-def by auto
    thus ?thesis
      using that unfolding image-image[of V j (λx. (ξ(j := x))),symmetric]
        by (intro bounded-subset[OF V-bounded[of j {..<j+1}]] that
image-mono) auto
  qed

  have h-step: h j ω = ∫ τ. h (j+1) (ω (j := τ)) ∂ M j (is ?L1 =
?R1)
    if ω ∈ space (PiM {..<j} M) j < n for j ω
  proof −
    have 0: (λx. f (merge {..<j} {j..<n} (ω, x))) ∈ borel-measurable
(Pi_M {j..<n} M)
      using that unfolding space-PiM by (intro f-merge-meas) auto

    have 1: insert j {Suc j..<n} = {j..<n}
      using that by auto

    have 2: bounded ((λx.(f (merge {..<j} {j..<n} (ω, x)))) ‘ space
(Pi_M {j..<n} M))
      using that by (intro f-merge-bounded) (simp-all add: space-PiM)

    have ?L1 = (∫ ξ. f (merge {..<j} {j..<n} (ω, ξ)) ∂PiM (insert j
{j+1..<n}) M)
    unfolding h-def using that by (intro Bochner-Integration.integral-cong
refl PiM-cong) auto
    also have ...=(∫ τ.(∫ ξ. f (merge {..<j} {j..<n} (ω, (ξ(j := τ))))
∂PiM {j+1..<n} M)∂M j)
    using that(1,2) 0 1 2 by (intro product-integral-insert prob-space-imp-sigma-finite
assms(1)
        int-rules f-merge-meas) (simp-all)
    also have ... = ?R1
      using that(2) unfolding h-def
     by (intro Bochner-Integration.integral-cong arg-cong[where f=f]
ext) (auto simp:merge-def)
    finally show ?thesis
      by simp
  qed
```

have *V-meas*: $V\ i \in$ *borel-measurable* $(PiM\ I\ M)$ **if** $i < n\ \{..<i+1\} \subseteq I$ **for** $i\ I$
unfolding *V-def* **using** *that* **by** (*intro borel-measurable-diff h-meas*) *auto*

have *V-upd-meas*: $(\lambda x.\ V\ j\ (\xi(j := x))) \in$ *borel-measurable* $(M\ j)$
if $j < n\ \xi \in$ *space* $(Pi_M\ \{..<j\}\ M)$ **for** $j\ \xi$
using *that* **by** (*intro measurable-compose*[*OF - V-meas*[**where** *I=insert j {..<j}*]]
measurable-component-update) *auto*

have *V-cong*:
$V\ j\ \omega = V\ j\ \xi$ **if** *restrict* $\omega\ \{..<(j+1)\} =$ *restrict* $\xi\ \{..<(j+1)\}$ **for** $j\ \omega\ \xi$
using *that restrict-subset-eq*[*OF - that*] **unfolding** *V-def*
by (*intro arg-cong2*[**where** *f=(−)*] *h-cong*) *simp-all*

have *exp-V*: $(\int \omega.\ V\ j\ (\xi(j := \omega))\ \partial M\ j) = 0$ (**is** *?L1 = 0*)
if $j < n\ \xi \in$ *space* $(PiM\ \{..<j\}\ M)$ **for** $j\ \xi$
proof −

have *fun-upd* $\xi\ j$ ‘ *space* $(M\ j) \subseteq$ *space* $(Pi_M\ (insert\ j\ \{..<j\})\ M)$
using *that* **unfolding** *space-PiM* **by** (*intro image-subsetI PiE-fun-upd*) *auto*
hence *0*:*bounded* $((\lambda x.\ h\ (Suc\ j)\ (\xi(j := x)))$ ‘ *space* $(M\ j))$
unfolding *image-image*[*of h* $(Suc\ j)\ \lambda x.\ \xi(j := x)$,*symmetric*] **using** *that*
by (*intro bounded-subset*[*OF h-bounded*[**where** *i=j+1* **and** *I={..<j+1}*]] *image-mono*)
(*auto simp*:*lessThan-Suc*)

have *1*:$(\lambda x.\ h\ (Suc\ j)\ (\xi(j := x))) \in$ *borel-measurable* $(M\ j)$
using *h-meas that* **by** (*intro measurable-compose*[*OF - h-meas*[**where** *I=insert j {..<j}*]]
measurable-component-update) *auto*

have *?L1* $= (\int \omega.\ h\ (Suc\ j)\ (\xi(j := \omega)) - h\ j\ \xi\ \partial M\ j)$
unfolding *V-def*
by (*intro Bochner-Integration.integral-cong arg-cong2*[**where** *f=(−)*] *refl h-cong*) *auto*
also have *...* $= (\int \omega.\ h\ (Suc\ j)\ (\xi(j := \omega))\ \partial M\ j) - (\int \omega.\ h\ j\ \xi\ \partial M\ j)$
using *that* **by** (*intro Bochner-Integration.integral-diff int-rules 0 1*) *auto*
also have *... = 0*
using *that*(*1*) *assms*(*1*) *prob-space.prob-space* **unfolding** *h-step*[*OF that*(*2*,*1*)] **by** *auto*
finally show *?thesis*
by *simp*

qed

have *var-V*: $|V\ j\ x - V\ j\ y| \leq c\ j$ (**is** *?L1* $\leq$ *?R1*)
if *var-V-assms*: $j < n$ $\{x,y\} \subseteq$ *space* (*PiM* $\{..<j+1\}$ *M*)
restrict x $\{..<j\}$ = *restrict y* $\{..<j\}$ **for** *x y j*
proof −
have *x-ran*: $x \in$ *PiE* $\{..<j+1\}$ $(\lambda i.$ *space* $(M\ i))$ **and** *y-ran*: $y \in$ *PiE* $\{..<j+1\}$ $(\lambda i.$ *space* $(M\ i))$
using *that*(*2*) **by** (*simp-all add*:*space-PiM*)

have *0*: $j+1 \leq n$
using *that* **by** *simp*

have *?L1* = $|h\ (Suc\ j)\ x - h\ j\ y - (h\ (Suc\ j)\ y - h\ j\ y)|$
unfolding *V-def* **by** (*intro arg-cong*[**where** *f*=*abs*] *arg-cong2*[**where** *f*=(−)] *refl h-cong that*)
also have ... = $|h\ (j+1)\ x - h\ (j+1)\ y|$
by *simp*
also have ... =
$|(\int \omega.\ f(merge\ \{..<j+1\}\ \{j+1..<n\}\ (x,\omega)) - f(merge\ \{..<j+1\}\ \{j+1..<n\}\ (y,\omega))\ \partial PiM\ \{j+1..<n\}\ M)|$
using *that* **unfolding** *h-def* **by** (*intro arg-cong*[**where** *f*=*abs*] *f-merge-meas*[*OF 0*] *x-ran*
Bochner-Integration.integral-diff[*symmetric*] *int-rules f-merge-bounded*[*OF 0*] *y-ran*) *auto*
also have ... $\leq$
$(\int \omega.\ |f(merge\ \{..<j+1\}\ \{j+1..<n\}\ (x,\omega)) - f(merge\ \{..<j+1\}\ \{j+1..<n\}\ (y,\omega))|\ \partial PiM\ \{j+1..<n\}\ M)$
by (*intro integral-abs-bound*)
also have ... $\leq (\int \omega.\ c\ j\ \partial PiM\ \{j+1..<n\}\ M)$
proof (*intro Bochner-Integration.integral-mono' delta int-rules c-ge-0 ballI merge-space 0*)
fix ω **assume** $\omega \in$ *space* $(Pi_M\ \{j+1..<n\}\ M)$
have $\{..<j+1\} \cap (\{..<n\} - \{j\}) = \{..<j\}$
using *that* **by** *auto*
thus *restrict* (*merge* $\{..<j+1\}$ $\{j+1..<n\}$ (x, ω)) $(\{..<n\}-\{j\})$ =
restrict (*merge* $\{..<j+1\}$ $\{j+1..<n\}$ (y, ω)) $(\{..<n\}-\{j\})$
using *that*(*1,3*) *less-antisym* **unfolding** *restrict-merge* **by** (*intro merge-cong refl*) *auto*
qed (*simp-all add*: *space-PiM that*(*1*) *x-ran*[*simplified*] *y-ran*[*simplified*])
also have ... = *c j*
by (*auto intro*!:*prob-space.prob-space prob-space-PiM assms*(*1*))
finally show *?thesis* **by** *simp*
qed

have $f\ \xi - (\int \omega.\ f\ \omega\ \partial(PiM\ \{..<n\}\ M)) = (\sum i < n.\ V\ i\ \xi)$ **if** $\xi \in$ *space* (*PiM* $\{..<n\}$ *M*) **for** ξ
using *that* **unfolding** *V-def* **by** (*subst sum-lessThan-telescope*)

(*simp add*: *h-0 h-n*)
hence *?L* = $\mathcal{P}$(*ξ in PiM* {..<*n*} *M*. (∑ *i* < *n*. *V i ξ*) ≥ *ε*)
by (*intro arg-cong2*[**where** *f*=*measure*] *refl Collect-restr-cong arg-cong2*[**where** *f*=(≤)]) *auto*
also have ... ≤ $\mathcal{P}$(*ξ in PiM* {..<*n*} *M*. *exp*(*t* ∗ (∑ *i* < *n*. *V i ξ*)) ≥ *exp* (*t* ∗ *ε*))
proof (*intro finite-measure.finite-measure-mono subsetI prob-space.finite-measure int-rules*)
show {*ξ* ∈ *space* (*Pi*$_M$ {..<*n*} *M*). *exp* (*t* ∗ *ε*) ≤ *exp* (*t* ∗ (∑ *i*<*n*. *V i ξ*))} ∈ *sets* (*Pi*$_M$ {..<*n*} *M*)
using *V-meas* **by** *measurable*
qed (*auto intro*!:*mult-left-mono*[*OF - t-ge-0*])
also have ... ≤ (∫ *ξ*. *exp*(*t*∗(∑ *i* < *n*. *V i ξ*)) ∂*PiM* {..<*n*} *M*)/ *exp* (*t*∗*ε*)
by (*intro integral-Markov-inequality-measure*[**where** *A*={}] *int-rules V-bounded V-meas*) *auto*
also have ... = *exp*(*t*^2 ∗ (∑ *i*∈{*n*..<*n*}. *c i*^2)/8−*t*∗*ε*)∗(∫ *ξ*. *exp*(*t*∗(∑ *i* < *n*. *V i ξ*)) ∂*PiM* {..<*n*} *M*)
by (*simp add*:*exp-minus inverse-eq-divide*)
also have ... ≤ *exp*(*t*^2 ∗ (∑ *i*∈{*0*..<*n*}. *c i*^2)/8−*t*∗*ε*)∗(∫ *ξ*. *exp*(*t*∗(∑ *i* < *0*. *V i ξ*)) ∂*PiM* {..<*0*} *M*)
proof (*rule ineq-chain*)
fix *j* **assume** *a*:*j* < *n*
let *?L1* = *exp* (*t*^{2}∗(∑ *i*=*j*+*1*..<*n*. (*c i*)^2)/8−*t*∗*ε*)
let *?L2* = *?L1* ∗ (∫ *ξ*. *exp* (*t* ∗ (∑ *i*<*j*+*1*. *V i ξ*)) ∂*PiM* {..<*j*+*1*} *M*)

note *V-upd-meas* = *V-upd-meas*[*OF a*]

have *?L2* = *?L1* ∗ (∫ *ξ*. *exp* (*t* ∗ (∑ *i*<*j*. *V i ξ*)) ∗ *exp*(*t* ∗ *V j ξ*) ∂*PiM* (*insert j* {..<*j*}) *M*)
by (*simp add*:*algebra-simps exp-add lessThan-Suc*)
also have ... = *?L1* ∗
(∫ *ξ*. (∫ *ω*. *exp* (*t* ∗ (∑ *i*<*j*. *V i* (*ξ*(*j* := *ω*)))) ∗ *exp*(*t* ∗ *V j* (*ξ*(*j* := *ω*))) ∂*M j*) ∂*PiM* {..<*j*} *M*)
using *a* **by** (*intro product-integral-insert-rev arg-cong2*[**where** *f*=(∗)] *int-rules*
prob-space-imp-sigma-finite V-bounded V-meas) *auto*
also have ... =*?L1*∗(∫ *ξ*.(∫ *ω*. *exp* (*t*∗(∑ *i*<*j*. *V i ξ*))∗*exp*(*t*∗*V j* (*ξ*(*j* := *ω*))) ∂*M j*) ∂*PiM* {..<*j*} *M*)
by (*intro arg-cong2*[**where** *f*=(∗)] *Bochner-Integration.integral-cong*
arg-cong[**where** *f*=*exp*] *sum.cong V-cong restrict-fupd*) *auto*
also have ... =*?L1*∗(∫ *ξ*. *exp* (*t*∗(∑ *i*<*j*. *V i ξ*))∗(∫ *ω*. *exp*(*t*∗*V j* (*ξ*(*j* := *ω*))) ∂*M j*) ∂*PiM* {..<*j*} *M*)
using *a* **by** (*intro arg-cong2*[**where** *f*=(∗)] *Bochner-Integration.integral-cong refl*
Bochner-Integration.integral-mult-right V-upd-meas V-upd-bounded int-rules) *auto*
also have ... ≤ *?L1* ∗ ∫ *ξ*. *exp* (*t*∗(∑ *i*<*j*. *V i ξ*))∗ *exp* (*t*^2 ∗ *c*

j^2/8) ∂PiM {..<j} M
proof (*intro mult-left-mono integral-mono'*)
fix *ξ* **assume** *c*:*ξ* ∈ *space* (*Pi*$_M$ *{..<j} M*)
hence *b*:*ξ* ∈ *PiE {..<j}* (*λi. space* (*M i*))
unfolding *space-PiM* **by** *simp*
moreover have *ξ*(*j* := *v*) ∈ *PiE {..<j+1}* (*λi. space* (*M i*)) **if** *v* ∈ *space* (*M j*) **for** *v*
using *b that* **unfolding** *PiE-def extensional-def Pi-def* **by** *auto*
ultimately show *LINT ω|M j. exp* (*t* ∗ *V j* (*ξ*(*j* := *ω*))) ≤ *exp* (*t*^{2} ∗ (*c j*)^2 / *8*)
using *V-upd-meas*[*OF c*]
by (*intro prob-space.Hoeffdings-lemma-bochner-3 exp-V var-V a int-rules*)
(*auto simp*: *space-PiM*)
next
show *integrable* (*Pi*$_M$ *{..<j} M*) (*λx. exp* (*t* ∗ (∑ *i*<*j. V i x*)) ∗ *exp* (*t*^{2} ∗ (*c j*)^2 / *8*))
using *a* **by** (*intro int-rules V-bounded V-meas*) *auto*
qed *auto*
also have *... = ?L1* ∗ ((∫ *ξ. exp* (*t*∗(∑ *i*<*j. V i ξ*)) *∂PiM {..<j} M*) ∗ *exp* (*t^2* ∗ *c j^2/8*))
proof (*subst Bochner-Integration.integral-mult-left*)
show *integrable* (*Pi*$_M$ *{..<j} M*) (*λξ. exp* (*t* ∗ (∑ *i*<*j. V i ξ*)))
using *a* **by** (*intro int-rules V-bounded V-meas*) *auto*
qed *auto*
also have *... =*
exp (*t*^{2}∗(∑ *i*∈*insert j {j+1..<n}*. (*c i*)^2)/*8*−*t*∗*ε*) ∗ (∫ *ξ. exp* (*t* ∗ (∑ *i*<*j. V i ξ*))*∂PiM {..<j} M*)
by (*simp-all add*:*exp-add*[*symmetric*] *field-simps*)
also have *...=exp* (*t*^{2}∗(∑ *i*=*j*..<*n*. (*c i*)^2)/*8*−*t*∗*ε*) ∗ (∫ *ξ. exp* (*t* ∗ (∑ *i*<*j. V i ξ*))*∂PiM {..<j} M*)
using *a* **by** (*intro arg-cong2*[**where** *f*=(∗)] *arg-cong*[**where** *f*=*exp*] *refl arg-cong2*
[**where** *f*=(−)] *arg-cong2*[**where** *f*=(/)] *sum.cong*) *auto*
finally show *?L2*≤*exp*(*t*^{2}∗(∑ *i*=*j*..<*n*. (*c i*)^2)/*8*−*t*∗*ε*)∗(∫ *ξ. exp* (*t*∗(∑ *i*<*j. V i ξ*))*∂PiM {..<j} M*)
by *simp*
qed
also have *... = exp*(*t^2* ∗ (∑ *i*<*n. c i^2*)/*8*−*t*∗*ε*) **by** (*simp add*:*PiM-empty atLeast0LessThan*)
also have *... = exp*(*t* ∗ ((*t* ∗ (∑ *i*<*n. c i^2*)/*8*)−*ε*)) **by** (*simp add*:*algebra-simps power2-eq-square*)
also have *... = exp*(*t* ∗ (−*ε*/*2*)) **using** *sum-c-ge-0* **by** (*auto simp add*:*divide-simps t-def*)
also have *... = ?R* **unfolding** *t-def* **by** (*simp add*:*field-simps power2-eq-square*)
finally show *?thesis* **by** *simp*
qed

theorem *mc-diarmid-inequality-distr*:

fixes $f :: ('i \Rightarrow 'a) \Rightarrow real$
assumes *finite I*
assumes $\bigwedge i.\ i \in I \Longrightarrow$ *prob-space* $(M\ i)$
assumes $\bigwedge i\ x\ y.\ i \in I \Longrightarrow \{x,y\} \subseteq$ *space* $(PiM\ I\ M) \Longrightarrow (\forall\, j{\in}I{-}\{i\}.$
$x\ j{=}y\ j) \Longrightarrow |f\ x{-}f\ y|{\leq}c\ i$
assumes *f-meas*: $f \in$ *borel-measurable* $(PiM\ I\ M)$ **and** ε*-gt-0*: $\varepsilon > 0$
shows $\mathcal{P}(\omega\ in\ PiM\ I\ M.\ f\ \omega - (\int \xi.\ f\ \xi\ \partial PiM\ I\ M) \geq \varepsilon) \leq exp$
$(-(2{*}\varepsilon\hat{}2)/(\sum i{\in}I.\ (c\ i)\hat{}2))$
(**is** *?L* $\leq$ *?R*)
proof −
define *n* **where** $n =$ *card I*
let *?q* = *from-nat-into I*
let *?r* = *to-nat-on I*
let *?f* = $(\lambda\xi.\ f\ (\lambda i{\in}I.\ \xi\ (?r\ i)))$

have *q*: *bij-betw ?q* {..<*n*} *I* **unfolding** *n-def* **by** (*intro bij-betw-from-nat-into-finite assms*(*1*))
have *r*: *bij-betw ?r I* {..<*n*} **unfolding** *n-def* **by** (*intro to-nat-on-finite assms*(*1*))

have [*simp*]: *?q* (*?r x*) = *x* **if** $x \in I$ **for** *x*
by (*intro from-nat-into-to-nat-on that countable-finite assms*(*1*))

have [*simp*]: *?r* (*?q x*) = *x* **if** $x < n$ **for** *x*
using *bij-betw-imp-surj-on*[*OF r*] *that* **by** (*intro to-nat-on-from-nat-into*) *auto*

have *a*: $\bigwedge i.\ i \in \{..{<}n\} \Longrightarrow$ *prob-space* $((M \circ ?q)\ i)$
unfolding *comp-def* **by** (*intro assms*(*2*) *bij-betw-apply*[*OF q*])

have *b*:*PiM I M* = *PiM I* $(\lambda i.\ (M \circ ?q)\ (?r\ i))$ **by** (*intro PiM-cong*) (*simp-all add*:*comp-def*)
also have ... = *distr* (*PiM* {..<*n*} $(M \circ ?q)$) (*PiM I* $(\lambda i.\ (M \circ ?q)$ (*?r i*))) $(\lambda\omega.\ \lambda n{\in}I.\ \omega\ (?r\ n))$
using *r* **unfolding** *bij-betw-def* **by** (*intro distr-PiM-reindex*[*symmetric*] *a*) *auto*
finally have *c*: *PiM I M* = *distr* (*PiM*{..<*n*}$(M{\circ}?q)$) (*PiM I* $(\lambda i.(M{\circ}?q)(?r\ i)))$ $(\lambda\omega.\ \lambda n{\in}I.\ \omega\ (?r\ n))$
by *simp*

have *d*: $(\lambda n{\in}I.\ x\ (?r\ n)) \in$ *space* $(Pi_M\ I\ M)$ **if** *4*:$x \in$ *space* $(Pi_M$ {..<*n*}$)\ (M \circ ?q))$ **for** *x*
proof −
have *x* (*?r i*) ∈ *space* (*M i*) **if** $i \in I$ **for** *i*
proof −
have *?r i* ∈ {..<*n*} **using** *bij-betw-apply*[*OF r*] *that* **by** *simp*
hence *x* (*?r i*) ∈ *space* $((M \circ ?q)\ (?r\ i))$ **using** *that 4 PiE-mem* **unfolding** *space-PiM* **by** *blast*
thus *?thesis* **using** *that* **unfolding** *comp-def* **by** *simp*

```
  qed
  thus ?thesis unfolding space-PiM PiE-def by auto
 qed

 have ?L = \mathcal{P}(ω in PiM {..<n} (M ∘ ?q). ?f ω − (∫ ξ. f ξ ∂PiM I M) ≥ ε)
 proof (subst c, subst measure-distr, goal-cases)
  case 1 thus ?case
    by (intro measurable-restrict measurable-component-singleton bij-betw-apply[OF r])
 next
   case 2 thus ?case unfolding b[symmetric] by (intro measurable-sets-Collect[OF f-meas]) auto
 next
  case 3 thus ?case using d by (intro arg-cong2[where f=measure] refl) (auto simp:vimage-def)
 qed
 also have ... = \mathcal{P}(ω in PiM {..<n} (M ∘ ?q). ?f ω − (∫ ξ. ?f ξ ∂PiM {..<n} (M ∘ ?q)) ≥ ε)
 proof (subst c, subst integral-distr, goal-cases)
  case (1 ω) thus ?case
    by (intro measurable-restrict measurable-component-singleton bij-betw-apply[OF r])
 next
  case (2 ω) thus ?case unfolding b[symmetric] by (rule f-meas)
 next
  case 3 thus ?case by simp
 qed
 also have ... ≤ exp (−(2*ε^2)/(∑ i<n. (c (?q i))^2))
 proof (intro mc-diarmid-inequality-aux ε-gt-0, goal-cases)
  case (1 i) thus ?case by (intro a) auto
 next
  case (2 i x y)
  have x (?r j) = y (?r j) if j ∈ I − {?q i} for j
  proof −
   have ?r j ∈ {..<n} − {i} using that bij-betw-apply[OF r] by auto
   thus ?thesis using 2 by simp
  qed
  hence ∀ j∈I − {?q i}. (λi∈I. x (?r i)) j = (λi∈I. y (?r i)) j by auto
  thus ?case using 2 d by (intro assms(3) bij-betw-apply[OF q]) auto
 next
  case 3
  have (λx. x (?r i)) ∈ Pi_M {..<n} (M ∘ ?q) →_M M i if i ∈ I for i
  proof −
   have 0:M i = (M ∘ ?q) (?r i) using that by (simp add: comp-def)
   show ?thesis unfolding 0 by (intro measurable-component-singleton
```

bij-betw-apply[*OF r*] *that*)
qed
thus *?case* **by** (*intro measurable-compose*[*OF - f-meas*] *measurable-restrict*)
qed
also have ... = *?R* **by** (*subst sum.reindex-bij-betw*[*OF q*]) *simp*
finally show *?thesis* **by** *simp*
qed

lemma (**in** *prob-space*) *mc-diarmid-inequality-classic*:
fixes *f* :: ('*i* ⇒ '*a*) ⇒ *real*
assumes *finite I*
assumes *indep-vars N X I*
assumes ⋀*i x y*. *i* ∈ *I* ⟹ {*x*,*y*} ⊆ *space* (*PiM I N*) ⟹ (∀ *j*∈*I*−{*i*}. *x j*=*y j*) ⟹ |*f x*−*f y*|≤*c i*
assumes *f-meas*: *f* ∈ *borel-measurable* (*PiM I N*) **and** ε*-gt-0*: ε > *0*
shows *P*(ω *in M*. *f* (λ*i*∈*I*. *X i* ω) − (∫ ξ. *f* (λ*i*∈*I*. *X i* ξ) ∂*M*) ≥ ε) ≤ *exp* (−(*2*∗ε^*2*)/(∑ *i*∈*I*. (*c i*)^*2*))
(**is** *?L* ≤ *?R*)
proof −
note *indep-imp* = *iffD1*[*OF indep-vars-iff-distr-eq-PiM'*]
let *?O* = λ*i*. *distr M* (*N i*) (*X i*)
have *a*:*distr M* (*Pi*$_M$ *I N*) (λ*x*. λ*i*∈*I*. *X i x*) = *Pi*$_M$ *I ?O*
using *assms*(*2*) **unfolding** *indep-vars-def* **by** (*intro indep-imp*[*OF - assms*(*2*)]) *auto*

have *b*: *space* (*PiM I ?O*) = *space* (*PiM I N*)
by (*metis* (*no-types*, *lifting*) *a space-distr*)

have (λ*i*∈*I*. *X i* ω) ∈ *space* (*Pi*$_M$ *I N*) **if** ω ∈ *space M* **for** ω
using *assms*(*2*) *that* **unfolding** *indep-vars-def measurable-def space-PiM* **by** *auto*

hence *?L* = *P*(ω *in M*. (λ*i*∈*I*. *X i* ω)∈ *space* (*Pi*$_M$ *I N*)∧ *f* (λ*i*∈*I*. *X i* ω)−(∫ ξ. *f* (λ*i*∈*I*. *X i* ξ) ∂*M*)≥ε)
by (*intro arg-cong2*[**where** *f*=*measure*] *Collect-restr-cong refl*) *auto*
also have ... = *P*(ω *in distr M* (*Pi*$_M$ *I N*) (λ*x*. λ*i*∈*I*. *X i x*). *f* ω − (∫ ξ. *f* (λ*i*∈*I*. *X i* ξ) ∂*M*) ≥ ε)
proof (*subst measure-distr*, *goal-cases*)
case *1* **thus** *?case* **using** *assms*(*2*) **unfolding** *indep-vars-def* **by** (*intro measurable-restrict*) *auto*
next
case *2* **thus** *?case* **unfolding** *space-distr* **by** (*intro measurable-sets-Collect*[*OF f-meas*]) *auto*
next
case *3* **thus** *?case* **by** (*simp-all add*:*Int-def conj-commute*)
qed
also have ... = *P*(ω *in PiM I ?O*. *f* ω − (∫ ξ. *f* (λ*i*∈*I*. *X i* ξ) ∂*M*) ≥ ε)

unfolding *a* **by** *simp*
also have ... = $\mathcal{P}$(*ω in PiM I ?O. f ω − (∫ ξ. f ξ ∂ distr M* (Pi_M *I N*) (*λx. λi∈I. X i x*)) ≥ *ε*)
proof (*subst integral-distr*[*OF - f-meas*], *goal-cases*)
case (*1 ω*) **thus** *?case* **using** *assms*(*2*) **unfolding** *indep-vars-def* **by** (*intro measurable-restrict*)*auto*
next
case *2* **thus** *?case* **by** *simp*
qed
also have ... = $\mathcal{P}$(*ω in PiM I ?O. f ω − (∫ ξ. f ξ ∂* Pi_M *I ?O*) ≥ *ε*) **unfolding** *a* **by** *simp*
also have ... ≤ *?R*
using *f-meas assms*(*2*) *b* **unfolding** *indep-vars-def*
by (*intro mc-diarmid-inequality-distr prob-space-distr assms*(*1*) *ε-gt-0 assms*(*3*)) *auto*
finally show *?thesis* **by** *simp*
qed

end

7 Paley-Zygmund Inequality

This section proves slight improvements of the Paley-Zygmund Inequality [7]. Unfortunately, the improvements are on Wikipedia with no citation.

theory *Paley-Zygmund-Inequality*
imports *Lp.Lp*
begin

context *prob-space*
begin

theorem *paley-zygmund-inequality-holder*:
assumes *p*: *1 <* (*p*::*real*)
assumes *rv*: *random-variable borel Z*
assumes *intZp*: *integrable M* (*λz. |Z z| powr p*)
assumes *t*: *ϑ ≤ 1*
assumes *ZAEpos*: *AE z in M. Z z ≥ 0*
shows
(*expectation* (*λx. |Z x − ϑ ∗ expectation Z| powr p*) *powr* (*1 / (p−1)*)) ∗
prob {*z ∈ space M. Z z > ϑ ∗ expectation Z*}
≥ ((*1−ϑ*) *powr* (*p / (p−1)*)) ∗ *expectation Z powr* (*p / (p−1)*))
proof −
have *intZ*: *integrable M Z*
apply (*subst bound-L1-Lp*[*OF - rv intZp*])
using *p* **by** *auto*

define *eZ* **where** *eZ* = *expectation Z*
have *eZ* ≥ *0*
unfolding *eZ-def*
using *ZAEpos intZ integral-ge-const prob-Collect-eq-1* **by** *auto*

have *ezp*: *expectation* (λ*x*. |*Z x* − ϑ ∗ *eZ*| *powr p*) ≥ *0*
by (*meson Bochner-Integration.integral-nonneg powr-ge-zero*)

have *expectation* (λ*z*. *Z z* − ϑ ∗ *eZ*) = *expectation* (λ*z*. *Z z* + (− ϑ ∗ *eZ*))
by *auto*
moreover have ... = *expectation Z* + *expectation* (λ*z*. − ϑ ∗ *eZ*)
apply (*subst Bochner-Integration.integral-add*)
using *intZ* **by** *auto*
moreover have ... = *eZ* + (− ϑ ∗ *eZ*)
apply (*subst lebesgue-integral-const*)
using *eZ-def prob-space* **by** *auto*
ultimately have ∗: *expectation* (λ*z*. *Z z* − ϑ ∗ *eZ*) = *eZ* − ϑ ∗ *eZ*
by *linarith*

have *ev*: {*z* ∈ *space M*. ϑ ∗ *eZ* < *Z z*} ∈ *events*
using *rv* **unfolding** *borel-measurable-iff-greater*
by *auto*

define *q* **where** *q* = *p* / (*p*−*1*)

have *sqI*:(*indicat-real E x*) *powr q* = *indicat-real E* (*x*::'*a*) **for** *E x*
unfolding *q-def*
by (*metis indicator-simps*(*1*) *indicator-simps*(*2*) *powr-0 powr-one-eq-one*)

have *bm1*: (λ*z*. (*Z z* − ϑ ∗ *eZ*)) ∈ *borel-measurable M*
using *borel-measurable-const borel-measurable-diff rv* **by** *blast*
have *bm2*: (λ*z*. *indicat-real* {*z* ∈ *space M*. *Z z* > ϑ ∗ *eZ*} *z*) ∈ *borel-measurable M*
using *borel-measurable-indicator ev* **by** *blast*
have *integrable M* (λ*x*. |*Z x* + (−ϑ ∗ *eZ*)| *powr p*)
apply (*intro Minkowski-inequality*[*OF - rv - intZp*])
using *p* **by** *auto*
then have *int1*: *integrable M* (λ*x*. |*Z x* − ϑ ∗ *eZ*| *powr p*)
by *auto*

have *integrable M*
(λ*x*. *1* ∗ *indicat-real* {*z* ∈ *space M*. ϑ ∗ *eZ* < *Z z*} *x*)
apply (*intro integrable-real-mult-indicator*[*OF ev*])
by *auto*

then have *int2*: *integrable M*
(λ*x*. |*indicat-real* {*z* ∈ *space M*. ϑ ∗ *eZ* < *Z z*} *x*| *powr q*)
by (*auto simp add*: *sqI*)

have *pq*:*p* > (*0*::*real*) *q* > *0* *1*/*p* + *1*/*q* = *1*
unfolding *q-def* **using** *p* **by** (*auto simp*:*divide-simps*)
from *Holder-inequality*[*OF pq bm1 bm2 int1 int2*]
have *hi*: *expectation* (λ*x*. (*Z x* − *ϑ* ∗ *eZ*) ∗ *indicat-real* {*z* ∈ *space M*. *ϑ* ∗ *eZ* < *Z z*} *x*)
≤ *expectation* (λ*x*. |*Z x* − *ϑ* ∗ *eZ*| *powr p*) *powr* (*1* / *p*) ∗
expectation (λ*x*. |*indicat-real* {*z* ∈ *space M*. *ϑ* ∗ *eZ* < *Z z*} *x*| *powr q*) *powr* (*1* / *q*)
by *auto*

have *eZ* − *ϑ* ∗ *eZ* ≤
expectation (λ*z*. (*Z z* − *ϑ* ∗ *eZ*) ∗ *indicat-real* {*z* ∈ *space M*. *Z z* > *ϑ* ∗ *eZ*} *z*)
unfolding ∗[*symmetric*]
apply (*intro integral-mono*)
using *intZ ev* **apply** *auto*[*1*]
apply (*auto intro*!: *integrable-real-mult-indicator simp add*: *intZ ev*)[*1*]
unfolding *indicator-def of-bool-def*
by (*auto simp add*: *mult-nonneg-nonpos2*)

also have ... ≤
expectation (λ*x*. |*Z x* − *ϑ* ∗ *eZ*| *powr p*) *powr* (*1* / *p*) ∗
expectation (λ*x*. *indicat-real* {*z* ∈ *space M*. *ϑ* ∗ *eZ* < *Z z*} *x*) *powr* (*1* / *q*)
using *hi* **by** (*auto simp add*: *sqI*)

finally have *eZ* − *ϑ* ∗ *eZ* ≤
expectation (λ*x*. |*Z x* − *ϑ* ∗ *eZ*| *powr p*) *powr* (*1* / *p*) ∗
expectation (λ*x*. *indicat-real* {*z* ∈ *space M*. *ϑ* ∗ *eZ* < *Z z*} *x*) *powr* (*1* / *q*)
by *auto*

then have (*eZ* − *ϑ* ∗ *eZ*) *powr q* ≤
(*expectation* (λ*x*. |*Z x* − *ϑ* ∗ *eZ*| *powr p*) *powr* (*1* / *p*) ∗
expectation (λ*x*. *indicat-real* {*z* ∈ *space M*. *ϑ* ∗ *eZ* < *Z z*} *x*) *powr* (*1* / *q*)) *powr q*
by (*smt* (*verit*, *ccfv-SIG*) ‹*0* ≤ *eZ*› *mult-left-le-one-le powr-mono2 pq*(*2*) *right-diff-distrib*′ *t zero-le-mult-iff*)

also have ... =
(*expectation* (λ*x*. |*Z x* − *ϑ* ∗ *eZ*| *powr p*) *powr* (*1* / *p*)) *powr q* ∗
(*expectation* (λ*x*. *indicat-real* {*z* ∈ *space M*. *ϑ* ∗ *eZ* < *Z z*} *x*) *powr* (*1* / *q*)) *powr q*
using *powr-ge-zero powr-mult* **by** *presburger*
also have ... =
(*expectation* (λ*x*. |*Z x* − *ϑ* ∗ *eZ*| *powr p*) *powr* (*1* / *p*)) *powr q* ∗
(*expectation* (λ*x*. *indicat-real* {*z* ∈ *space M*. *ϑ* ∗ *eZ* < *Z z*} *x*))

by (*smt* (*verit*, *ccfv-SIG*) *Bochner-Integration.integral-nonneg divide-le-eq-1-pos indicator-pos-le nonzero-eq-divide-eq p powr-one powr-powr q-def*)
also have ... =
(*expectation* (λ*x*. |*Z x* − ϑ ∗ *eZ*| *powr p*) *powr* (*1* / (*p*−*1*))) ∗
(*expectation* (λ*x*. *indicat-real* {*z* ∈ *space M*. ϑ ∗ *eZ* < *Z z*} *x*))
by (*smt* (*verit*, *ccfv-threshold*) *divide-divide-eq-right divide-self-if p powr-powr q-def times-divide-eq-left*)
also have ... =
(*expectation* (λ*x*. |*Z x* − ϑ ∗ *eZ*| *powr p*) *powr* (*1* / (*p*−*1*))) ∗
prob {*z* ∈ *space M*. *Z z* > ϑ ∗ *eZ*}
by (*simp add*: *ev*)

finally have *1*: (*eZ* − ϑ ∗ *eZ*) *powr q* ≤
(*expectation* (λ*x*. |*Z x* − ϑ ∗ *eZ*| *powr p*) *powr* (*1* / (*p*−*1*))) ∗
prob {*z* ∈ *space M*. *Z z* > ϑ ∗ *eZ*} **by** *linarith*

have (*eZ* − ϑ ∗ *eZ*) *powr q* = ((*1* − ϑ) ∗ *eZ*) *powr q*
by (*simp add*: *mult.commute right-diff-distrib*)
also have ... = (*1* − ϑ) *powr q* ∗ *eZ powr q*
by (*simp add*: ‹*0* ≤ *eZ*› *powr-mult t*)
finally show *?thesis* **using** *1 eZ-def q-def* **by** *force*
qed

corollary *paley-zygmund-inequality*:
assumes *rv*: *random-variable borel Z*
assumes *intZsq*: *integrable M* (λ*z*. (*Z z*)^*2*)
assumes *t*: ϑ ≤ *1*
assumes *Zpos*: ⋀*z*. *z* ∈ *space M* ⟹ *Z z* ≥ *0*
shows
(*variance Z* + (*1*−ϑ)^*2* ∗ (*expectation Z*)^*2*) ∗
prob {*z* ∈ *space M*. *Z z* > ϑ ∗ *expectation Z*}
≥ (*1*−ϑ)^*2* ∗ (*expectation Z*)^*2*
proof −
have *ZAEpos*: *AE z in M*. *Z z* ≥ *0*
by (*simp add*: *Zpos*)

define *p* **where** *p* = (*2*::*real*)
have *p1*: *1* < *p* **using** *p-def* **by** *auto*
have *integrable M* (λ*z*. |*Z z*| *powr p*) **unfolding** *p-def*
using *intZsq* **by** *auto*

from *paley-zygmund-inequality-holder*[*OF p1 rv this t ZAEpos*]
have (*1* − ϑ) *powr* (*p* / (*p* − *1*)) ∗ (*expectation Z powr* (*p* / (*p* − *1*)))
≤ *expectation* (λ*x*. |*Z x* − ϑ ∗ *expectation Z*| *powr p*) *powr* (*1* / (*p* − *1*)) ∗
prob {*z* ∈ *space M*. ϑ ∗ *expectation Z* < *Z z*} .

```
  then have hi: (1 − ϑ)^2 ∗ (expectation Z)^2
    ≤ expectation (λx. (Z x − ϑ ∗ expectation Z)^2) ∗
       prob {z ∈ space M. ϑ ∗ expectation Z < Z z}
    unfolding p-def by (auto simp add: Zpos t)

  have intZ: integrable M Z
    apply (subst square-integrable-imp-integrable[OF rv intZsq])
    by auto

  define eZ where eZ = expectation Z
  have eZ ≥ 0
    unfolding eZ-def
    using Bochner-Integration.integral-nonneg Zpos by blast

  have ezp: expectation (λx. |Z x − ϑ ∗ eZ| powr p) ≥ 0
    by (meson Bochner-Integration.integral-nonneg powr-ge-zero)

  have expectation (λz. Z z − ϑ ∗ eZ) = expectation (λz. Z z + (− ϑ
∗ eZ))
    by auto
  also have ... = expectation Z + expectation (λz. − ϑ ∗ eZ)
    apply (subst Bochner-Integration.integral-add)
    using intZ by auto
  also have ... = eZ + (− ϑ ∗ eZ)
    apply (subst lebesgue-integral-const)
    using eZ-def prob-space by auto
  finally have ∗: expectation (λz. Z z − ϑ ∗ eZ) = eZ − ϑ ∗ eZ
    by linarith
  have variance Z =
     variance (λz. (Z z − ϑ ∗ eZ))
    using ∗ eZ-def by auto
  also have ... =
    expectation (λz. (Z z − ϑ ∗ eZ)^2)
    − (expectation (λx. Z x − ϑ ∗ eZ))^2
    apply (subst variance-eq)
    by (auto simp add: intZ power2-diff intZsq)
  also have ... = expectation (λz. (Z z − ϑ ∗ eZ)^2) − ((1−ϑ)^2 ∗
eZ^2)
    unfolding ∗ by (auto simp:algebra-simps power2-eq-square)
  finally have veq: expectation (λz. (Z z − ϑ ∗ eZ)^2) = (variance Z
+ (1−ϑ)^2 ∗ eZ^2)
    by linarith
  thus ?thesis
    using hi by (simp add: eZ-def)
qed

end

end
```

References

[1] G. Bennett. Probability inequalities for the sum of independent random variables. *Journal of the American Statistical Association*, 57(297):33–45, 1962.

[2] S. Boucheron, G. Lugosi, and O. Bousquet. Concentration inequalities. In O. Bousquet, U. von Luxburg, and G. Rätsch, editors, *Advanced Lectures on Machine Learning, ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003, Revised Lectures*, volume 3176 of *Lecture Notes in Computer Science*, pages 208–240. Springer, 2003.

[3] F. P. Cantelli. Sui confini della probabilita. In *Atti del Congresso Internazionale dei Matematici: Bologna del 3 al 10 de settembre di 1928*, pages 47–60, 1929.

[4] B. Efron and C. Stein. The Jackknife Estimate of Variance. *The Annals of Statistics*, 9(3):586 – 596, 1981.

[5] M. Loève. *Probability Theory I*, chapter Sums of Independent Random Variables, pages 235–279. Springer New York, New York, NY, 1977.

[6] C. McDiarmid. *Surveys in Combinatorics, 1989: Invited Papers at the Twelfth British Combinatorial Conference*, chapter On the method of bounded differences, pages 148 – 188. London Mathematical Society Lecture Note Series. Cambridge University Press, 1989.

[7] R. E. Paley and A. Zygmund. A note on analytic functions in the unit circle. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 28, pages 266–272. Cambridge University Press, 1932.

[8] J. M. Steele. An Efron-Stein Inequality for Nonsymmetric Statistics. *The Annals of Statistics*, 14(2):753 – 758, 1986.