Supplemental Data.

	***K_1*, Iter 2 to Iter 3**		***K_1*, Iter 3 to Iter 4**		***k_2*, Iter 2 to Iter 3**		***k_2*, Iter 3 to Iter 4**	
	Indirect	**Direct**	**Indirect**	**Direct**	**Indirect**	**Direct**	**Indirect**	**Direct**
Amygdala	-2.1	-2.2	-1.3	-1.0	-0.4	-1.0	-1.9	-1.2
Caudate	-6.4	-6.9	-4.6	-4.7	-1.5	-0.5	-2.7	-1.1
Cerebellum	-2.6	-2.0	-1.4	-0.8	-2.1	-1.0	-1.4	-0.3
Frontal Lobe	-3.3	-2.8	-1.9	-1.4	-1.9	-1.1	-1.4	-0.4
Hippocampus	-3.7	-2.8	-3.1	-2.2	-3.3	-0.3	-3.0	-0.4
Occipital Lobe	-2.1	-1.7	-1.1	-0.8	-1.2	-1.0	-0.9	-0.4
Putamen	-3.1	-3.2	-1.9	-1.9	-2.0	-1.0	-1.9	-0.7
Thalamus	-3.2	-2.8	-1.9	-1.5	-3.7	-1.3	-2.5	-0.8

Table S1. Regional percent change in K_1 and k_2 from iteration 2 to 3, and iteration 3 to 4, for the 100%-count **[^{11}C]AFM** dataset.

	***K_1*, Iter 2 to Iter 3**		***K_1*, Iter 3 to Iter 4**		***k_2*, Iter 2 to Iter 3**		***k_2*, Iter 3 to Iter 4**	
	Indirect	**Direct**	**Indirect**	**Direct**	**Indirect**	**Direct**	**Indirect**	**Direct**
Amygdala	-2.8	-2.6	-1.9	-1.9	-1.6	-0.5	-2.1	-0.6
Caudate	-8.4	-8.3	-5.1	-5.0	-0.6	-0.2	-1.1	-0.6
Cerebellum	-1.9	-1.8	-1.0	-0.8	-1.5	-0.9	-1.1	-0.5
Frontal Lobe	-2.5	-2.4	-1.5	-1.4	-1.0	-0.7	-0.8	-0.3
Hippocampus	-2.7	-2.0	-2.0	-1.5	-2.4	0.1	-2.6	-0.3
Occipital Lobe	-2.1	-2.0	-1.1	-1.0	-1.2	-1.0	-0.7	-0.6
Putamen	-3.7	-3.8	-2.3	-2.4	-1.0	-1.2	-0.9	-0.6
Thalamus	-3.7	-3.6	-2.1	-2.0	-1.5	-1.1	-1.0	-0.5

Table S2. Regional percent change in K_1 and k_2 from iteration 2 to 3, and iteration 3 to 4, for the 100%-count **[^{11}C]UCB-J** dataset.

	100%		20%		10%		5%	
	Indirect	Direct	Indirect	Direct	Indirect	Direct	Indirect	Direct
Amygdala	-2	-1	1	0	14	4	25	0
Caudate	-5	-3	-1	-2	3	-2	12	-3
Cerebellum	2	0	4	1	6	2	7	3
Frontal Lobe	0	-1	3	0	5	0	7	1
Hippocampus	2	0	6	1	10	2	15	2
Occipital Lobe	2	1	4	1	6	1	9	2
Putamen	0	1	3	1	13	4	20	1
Thalamus	0	0	3	1	7	1	18	2

Table S3. Regional **mean** percent bias in $\boldsymbol{K_1}$ for the **simulated** dataset at each count level, by the indirect and direct methods (results given for iteration 2). Regional mean was computed from the average K_1 image across replicates.

	100%		20%		10%		5%	
	Indirect	Direct	Indirect	Direct	Indirect	Direct	Indirect	Direct
Amygdala	314	-8	1166	27	1259	112	1427	875
Caudate	45	-6	574	9	1225	51	1483	693
Cerebellum	4	2	314	3	770	5	1448	95
Frontal Lobe	1	0	116	0	420	1	996	35
Hippocampus	5	4	341	2	791	8	1771	284
Occipital Lobe	1	1	123	1	420	1	921	32
Putamen	69	-3	830	48	1022	71	1752	900
Thalamus	46	1	717	25	1294	60	1441	668

Table S4. Regional **mean** percent bias in $\boldsymbol{V_T}$ for the **simulated** dataset at each count level, by the indirect and direct methods (results given for iteration 2). Regional mean was computed from the average V_T image across replicates.

	100%		20%		10%		5%	
	Indirect	Direct	Indirect	Direct	Indirect	Direct	Indirect	Direct
Amygdala	-2	-11	1188	0	1044	-2	1260	226
Caudate	-2	-10	17	-6	86	1	258	46
Cerebellum	1	1	8	2	15	3	26	7
Frontal Lobe	0	0	3	0	7	1	15	4
Hippocampus	1	3	5	1	15	3	47	14
Occipital Lobe	0	1	3	1	7	0	15	4
Putamen	-2	-7	45	0	36	-2	1497	151
Thalamus	3	-2	26	0	109	5	101	44

Table S5. Regional **median** percent bias in $\boldsymbol{V_T}$ for the **simulated** dataset at each count level, by the indirect and direct methods (results given for iteration 2). Regional median was computed from the average V_T image across replicates.

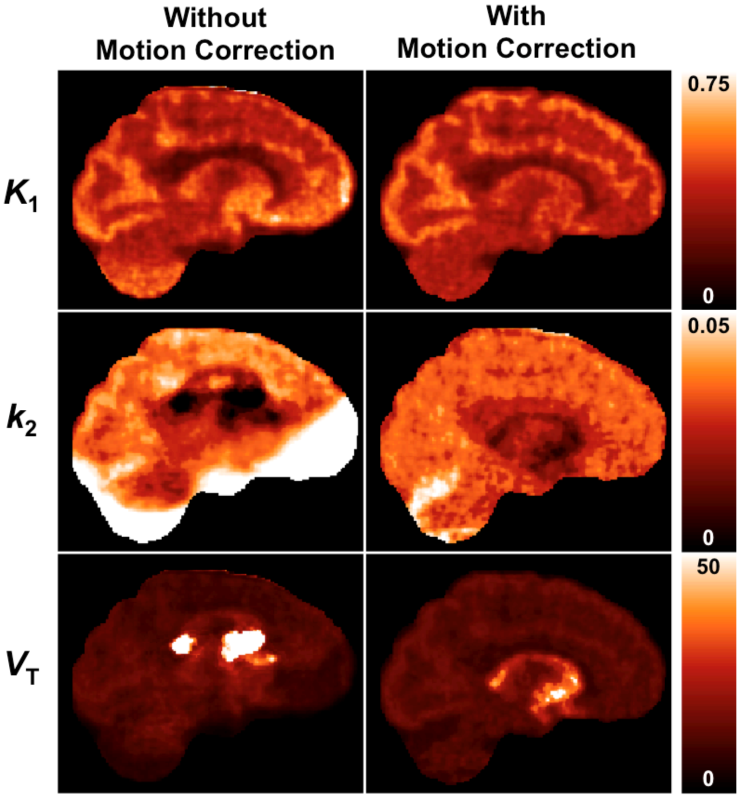

Figure S1. Parametric images generated by direct reconstruction for a full-count simulated dataset, with and without event-by-event motion correction. This simulation includes more severe motion than that presented in the main manuscript. Results are shown at iteration 2 (20 subsets/iteration).

	K_1			k_2			V_T		
	Without MC	**With MC**	**% Difference**	**Without MC**	**With MC**	**% Difference**	**Without MC**	**With MC**	**% Difference**
Amygdala	0.34	0.32	-5	0.012	0.011	-15	27.9	32.5	17
Caudate	0.39	0.41	4	0.014	0.014	0	29.3	30.3	4
Cerebellum	0.32	0.33	2	0.034	0.033	-1	9.7	10.1	4
Frontal Lobe	0.42	0.44	7	0.029	0.028	-3	14.6	15.9	9
Hippocampus	0.29	0.29	1	0.018	0.018	-2	16.6	16.8	1
Occipital Lobe	0.43	0.44	2	0.029	0.029	2	15.1	15.2	0
Putamen	0.46	0.50	7	0.014	0.014	1	35.5	38.2	8
Thalamus	0.41	0.44	7	0.014	0.013	-2	32.1	34.8	8

Table S6. Regional mean parameter estimates from direct reconstruction with and without motion correction (MC) for the **[^{11}C]AFM** dataset. Results are given for one replicate at the 20% count level, at iteration 2 (20 subsets/iteration).

	K_1			k_2			V_T		
	Without MC	**With MC**	**% Difference**	**Without MC**	**With MC**	**% Difference**	**Without MC**	**With MC**	**% Difference**
Amygdala	0.29	0.24	-17	0.028	0.013	-54	10.3	19.0	85
Caudate	0.39	0.37	-3	0.020	0.017	-15	20.1	22.6	12
Cerebellum	0.37	0.34	-7	0.033	0.020	-40	11.5	18.2	58
Frontal Lobe	0.40	0.43	6	0.029	0.018	-38	15.2	25.5	68
Hippocampus	0.31	0.24	-21	0.031	0.016	-50	10.5	16.8	60
Occipital Lobe	0.43	0.41	-4	0.022	0.017	-25	20.0	25.6	28
Putamen	0.44	0.44	-1	0.025	0.016	-34	18.0	27.8	54
Thalamus	0.39	0.39	0	0.028	0.020	-29	13.7	19.8	44

Table S7. Regional mean parameter estimates from direct reconstruction with and without motion correction (MC) for the **[^{11}C]UCB-J** dataset. Results are given for one replicate at the 20% count level, at iteration 2 (20 subsets/iteration).

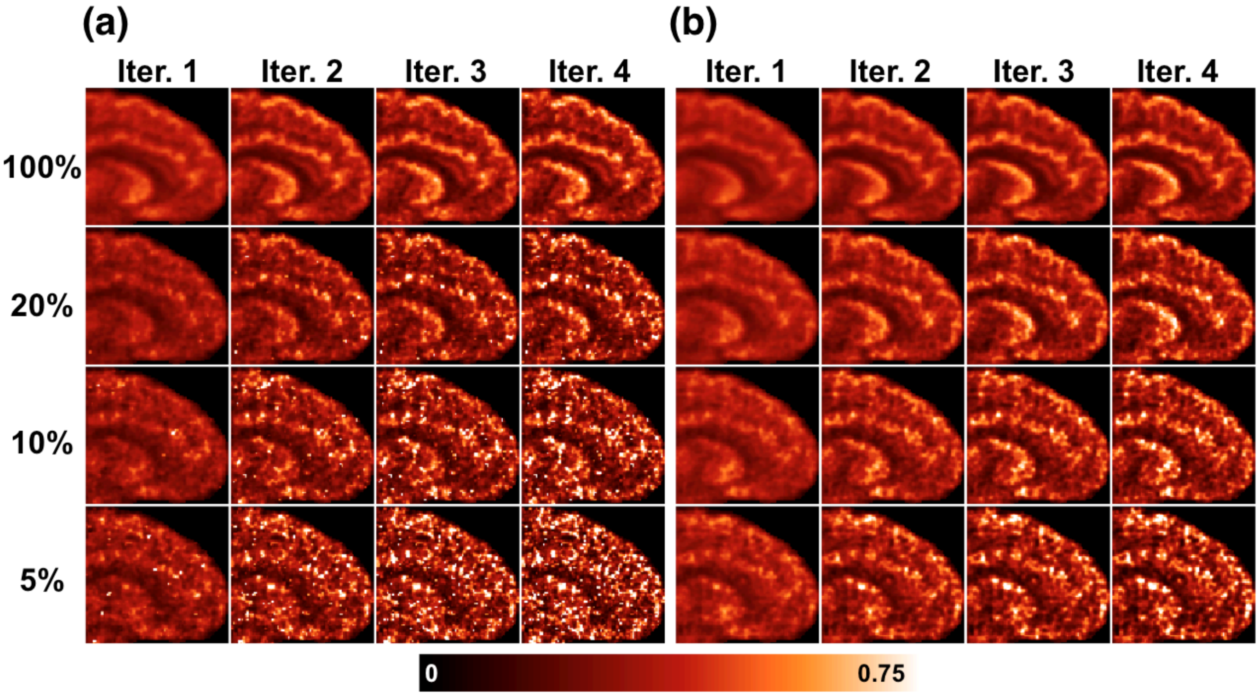

Figure S2. K_1 parametric maps of replicate of human [^{11}C]AFM dataset per count level (rows), across 4 iterations (columns). (a) Generated by the indirect method. (b) Generated by the direct method.

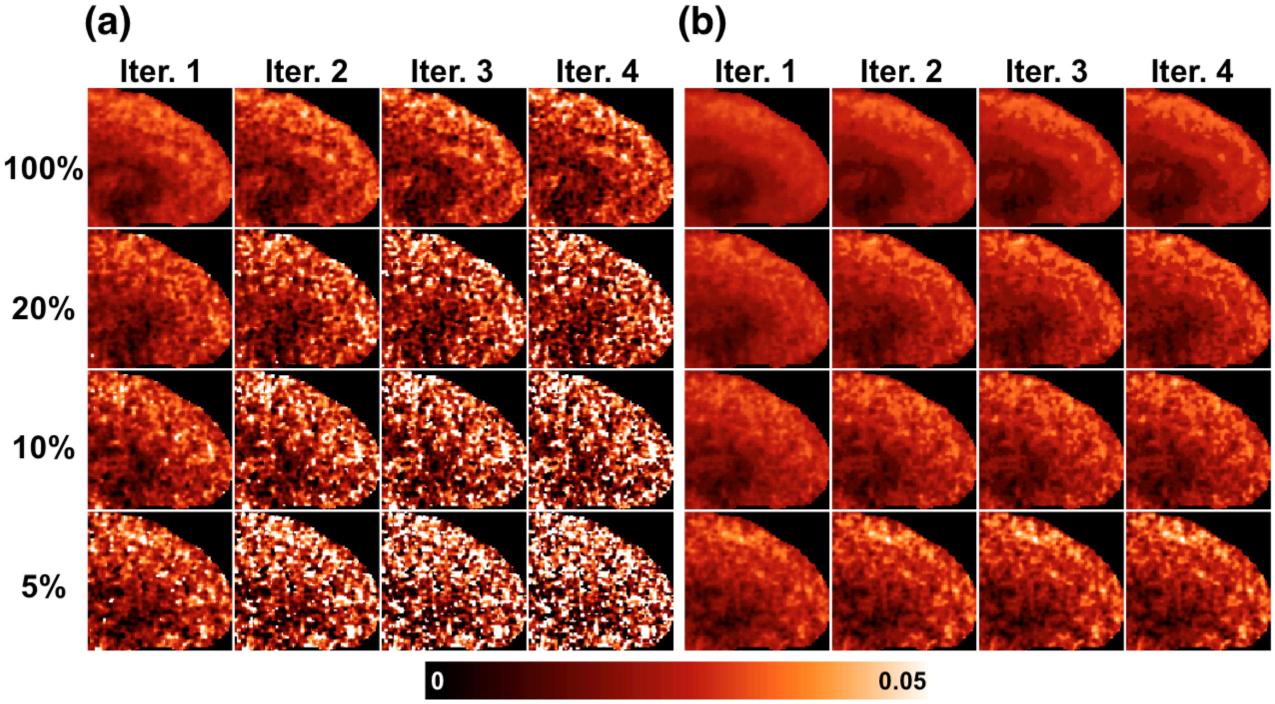

Figure S3. k_2 parametric maps of replicate of human [^{11}C]AFM dataset per count level (rows), across 4 iterations (columns). (a) Generated by the indirect method. (b) Generated by the direct method.

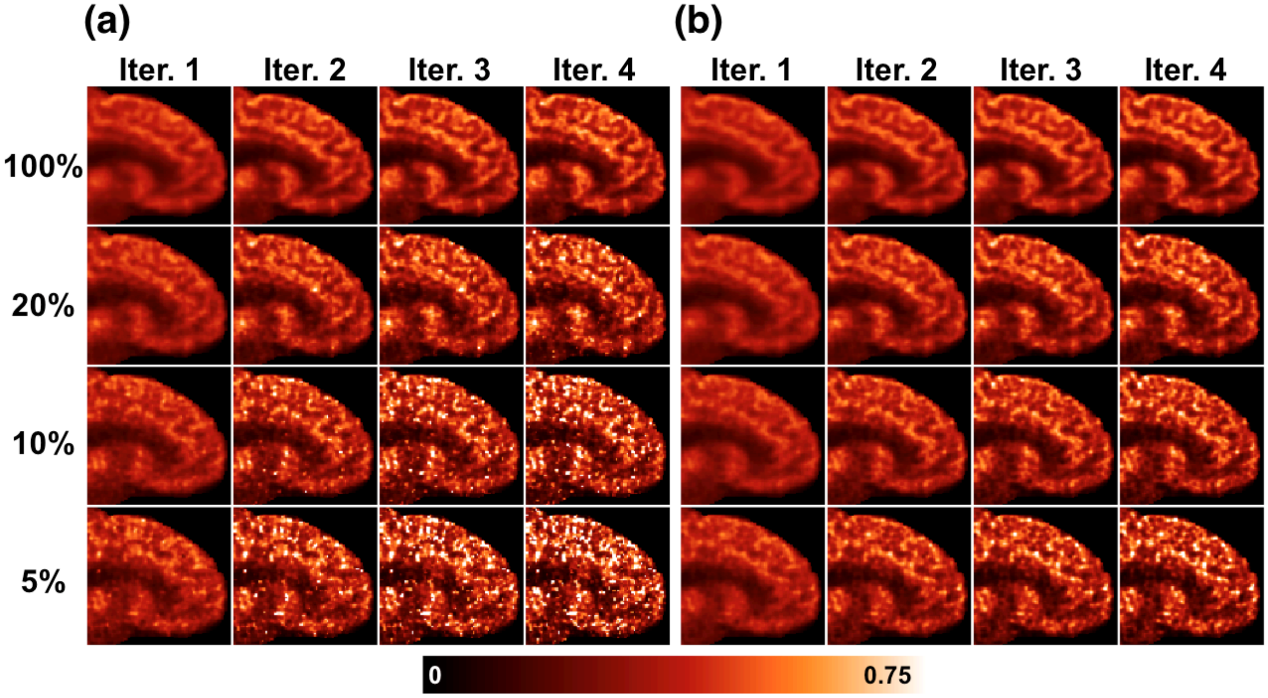

Figure S4. K_1 parametric maps of replicate of human [^{11}C]UCB-J dataset per count level (rows), across 4 iterations (columns). (a) Generated by the indirect method. (b) Generated by the direct method.

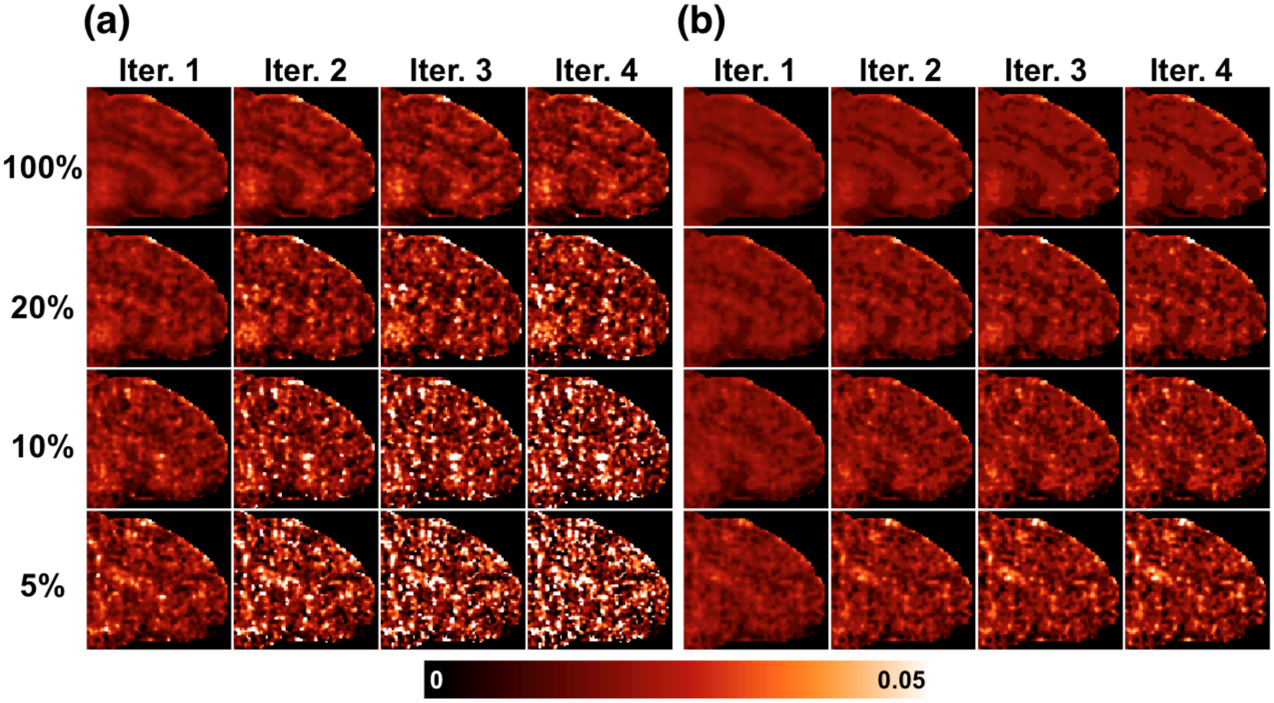

Figure S5. k_2 parametric maps of replicate of human [^{11}C]UCB-J dataset per count level (rows), across 4 iterations (columns). (a) Generated by the indirect method. (b) Generated by the direct method.

	100%		20%		10%		5%	
	Indirect	**Direct**	**Indirect**	**Direct**	**Indirect**	**Direct**	**Indirect**	**Direct**
Amygdala	0.31	0.32	0.34	0.33	0.36	0.33	0.41	0.34
Caudate	0.41	0.40	0.41	0.40	0.43	0.40	0.44	0.40
Cerebellum	0.35	0.33	0.36	0.33	0.36	0.33	0.37	0.34
Frontal Lobe	0.45	0.44	0.46	0.44	0.47	0.45	0.48	0.45
Hippocampus	0.31	0.29	0.32	0.29	0.34	0.29	0.36	0.30
Occipital Lobe	0.45	0.43	0.45	0.43	0.47	0.44	0.48	0.44
Putamen	0.51	0.49	0.50	0.49	0.52	0.50	0.55	0.50
Thalamus	0.46	0.44	0.46	0.44	0.47	0.44	0.49	0.44

Table S8. Regional mean $\boldsymbol{K_1}$ for the **[^{11}C]AFM** dataset at each count level, by the indirect and direct methods (results given for iteration 2). Regional mean and standard deviation were computed from the average K_1 image across replicates.

	100%		20%		10%		5%	
	Indirect	**Direct**	**Indirect**	**Direct**	**Indirect**	**Direct**	**Indirect**	**Direct**
Amygdala	141	33.8	449	36.6	399	33.3	618	52.9
Caudate	31.6	30.6	164	31.1	271	30.3	438	32.1
Cerebellum	9.27	10.1	27.6	10.1	59.4	10.2	113	10.4
Frontal Lobe	14.8	15.9	26.1	15.9	62.5	16.0	125	16.1
Hippocampus	15.5	16.9	62.0	16.7	112	16.7	202	16.9
Occipital Lobe	14.1	15.2	25.4	15.1	58.2	15.1	132	15.2
Putamen	38.1	38.8	228	39.5	376	41.1	483	40.9
Thalamus	35.9	35.1	166.6	35.1	281.7	36.3	448	39.8

Table S9. Regional mean $\boldsymbol{V_T}$ for the **[^{11}C]AFM** dataset at each count level, by the indirect and direct methods (results given for iteration 2). Regional mean and standard deviation were computed from the average V_T image across replicates.

	100%		20%		10%		5%	
	Indirect	**Direct**	**Indirect**	**Direct**	**Indirect**	**Direct**	**Indirect**	**Direct**
Amygdala	0.23	0.23	0.24	0.24	0.24	0.24	0.25	0.24
Caudate	0.38	0.38	0.38	0.37	0.38	0.38	0.39	0.38
Cerebellum	0.34	0.34	0.35	0.34	0.35	0.34	0.36	0.34
Frontal Lobe	0.43	0.43	0.43	0.43	0.44	0.43	0.45	0.43
Hippocampus	0.24	0.24	0.24	0.24	0.25	0.24	0.26	0.24
Occipital Lobe	0.41	0.41	0.42	0.41	0.42	0.41	0.43	0.41
Putamen	0.43	0.43	0.43	0.43	0.44	0.43	0.46	0.44
Thalamus	0.40	0.39	0.40	0.38	0.40	0.39	0.41	0.39

Table S10. Regional mean $\boldsymbol{K_1}$ for the **[^{11}C]UCB-J** dataset at each count level, by the indirect and direct methods (results given for iteration 2). Regional mean and standard deviation were computed from the average K_1 image across replicates.

	100%		20%		10%		5%	
	Indirect	**Direct**	**Indirect**	**Direct**	**Indirect**	**Direct**	**Indirect**	**Direct**
Amygdala	18.5	18.6	20.2	18.7	49.2	20.0	176	41.3
Caudate	23.3	22.4	26.1	22.6	40.1	22.9	104	25.7
Cerebellum	21.0	18.6	26.0	19.1	48.1	19.7	103	23.5
Frontal Lobe	25.4	25.0	30.5	25.9	54.3	26. 7	120	30.6
Hippocampus	17.2	16.6	30.4	17.5	71.0	20.1	172	42.1
Occipital Lobe	25.2	25.2	28.4	25.5	53.4	26.2	120	29.7
Putamen	27.8	27.8	33.3	28.2	55.3	28.9	138	31.8
Thalamus	19.2	19.5	22.30	19.8	29.6	19.9	76.6	20.8

Table S11. Regional mean $\boldsymbol{V_T}$ for the **[^{11}C]UCB-J** dataset at each count level, by the indirect and direct methods (results given for iteration 2). Regional mean and standard deviation were computed from the average V_T image across replicates.

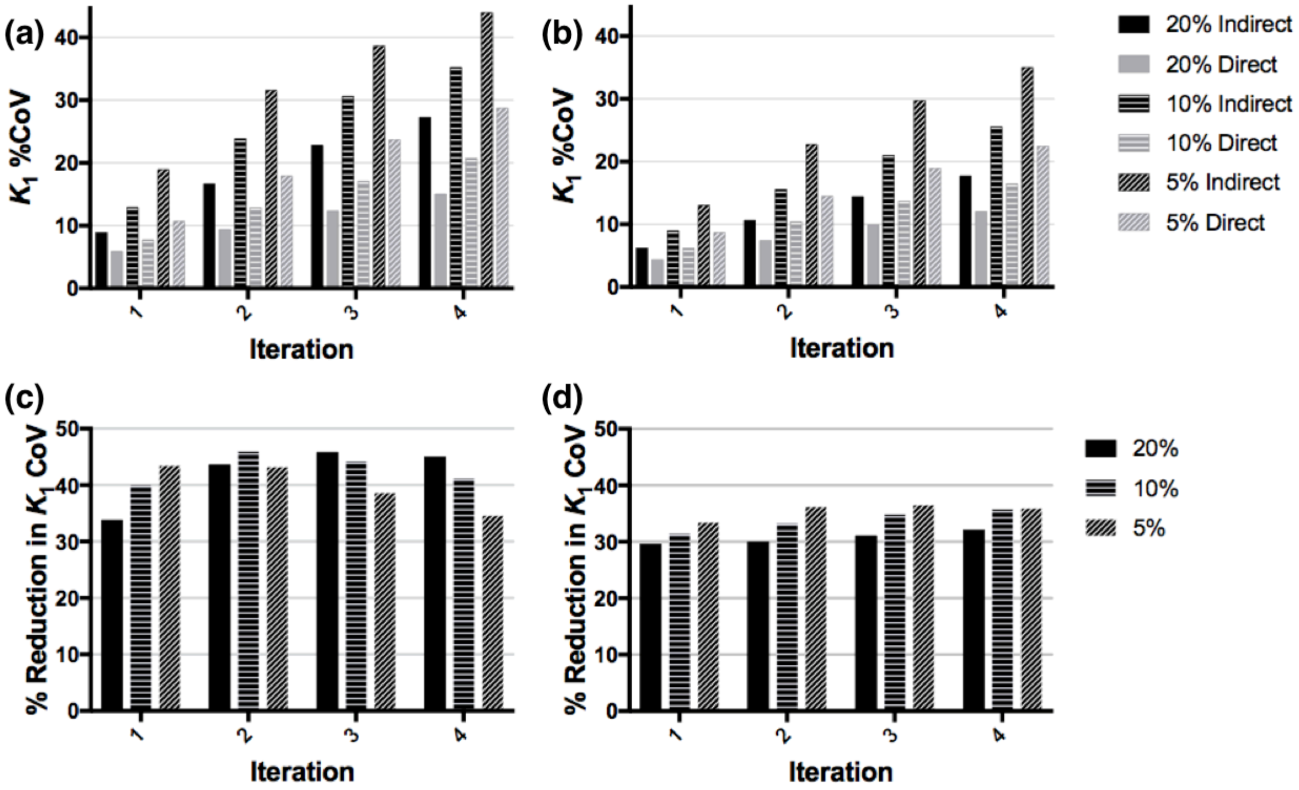

Figure S6. (a,b) Regional K_1 %CoV (in the frontal cortex) by iteration and count level for the indirect and direct methods, for (a) [^{11}C]AFM and (b) [^{11}C]UCB-J. (c,d) Percent reduction in regional K_1 CoV of the direct method relative to the indirect method, by iteration and count level, for (c) [^{11}C]AFM and (d) [^{11}C]UCB-J.

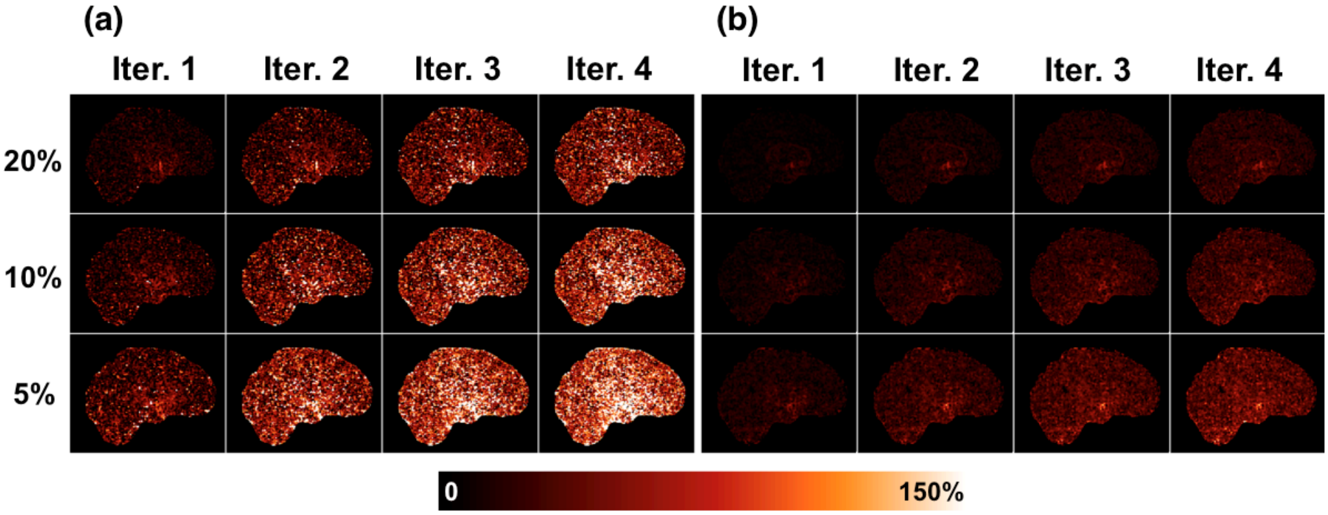

Figure S7. V_T %CoV maps of replicates of human [^{11}C]AFM data per count level (rows), across 4 iterations (columns). (a) Generated by the indirect method. (b) Generated by the direct method.

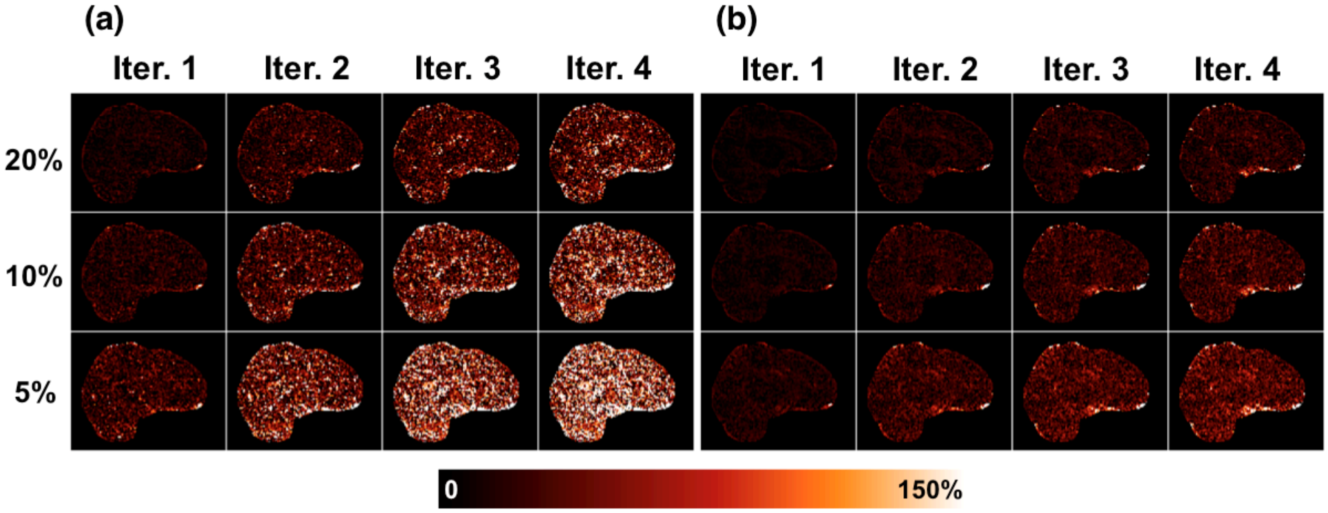

Figure S8. V_T %CoV maps of replicate human [^{11}C]UCB-J data per count level (rows), across 4 iterations (columns). (a) Generated by the indirect method. (b) Generated by the direct method.

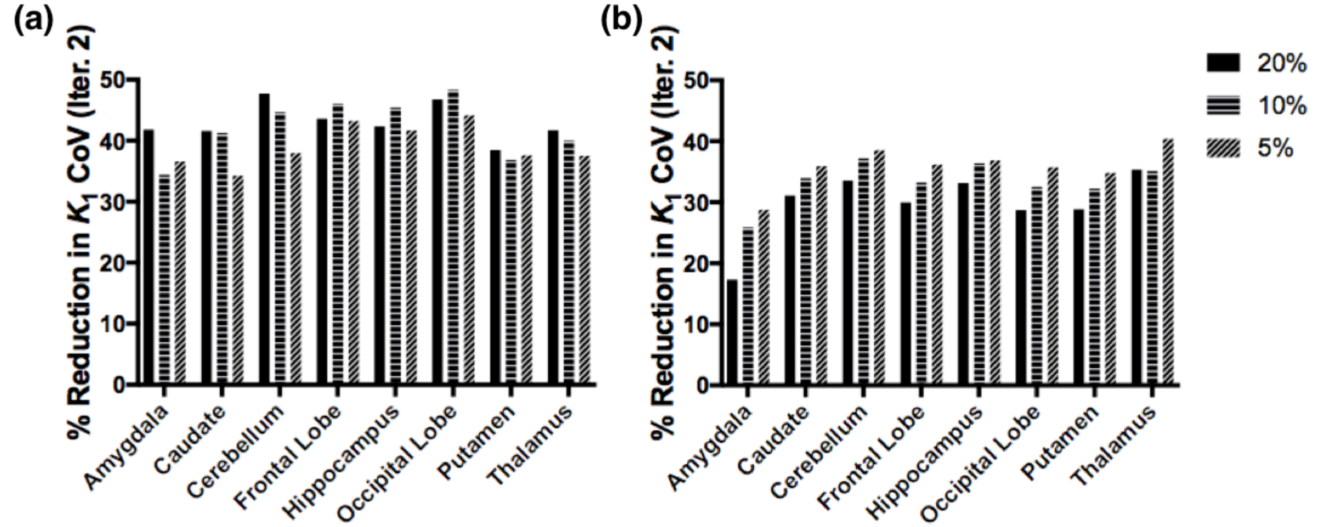

Figure S9. Percent reduction in regional K_1 CoV of the direct method relative to the indirect method by count level and region at iteration 2, for (a) [^{11}C]AFM and (b) [^{11}C]UCB-J.

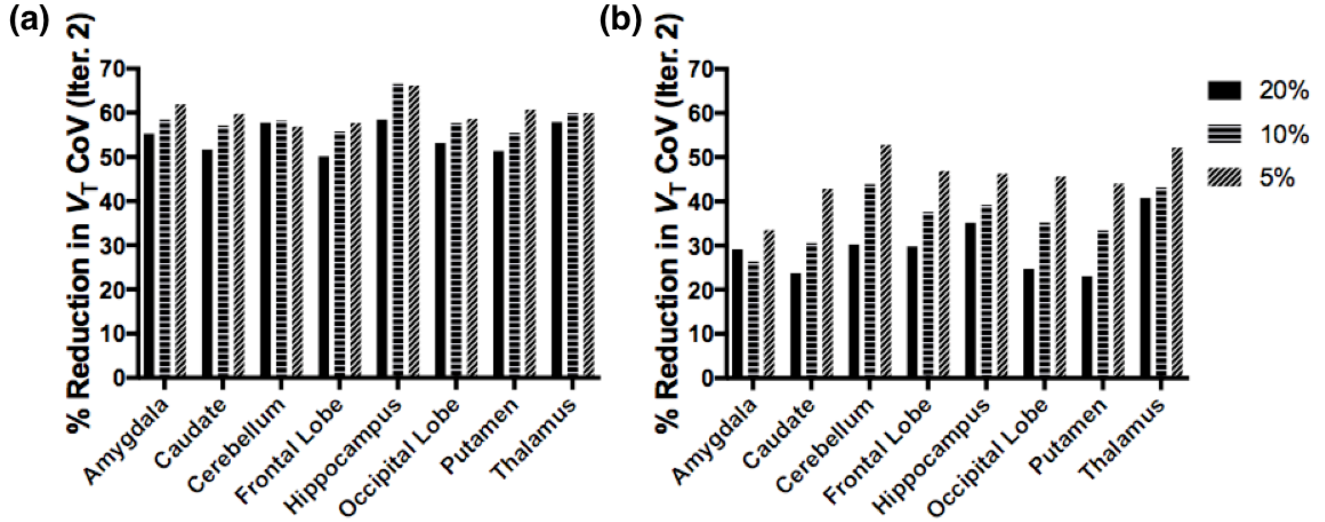

Figure S10. Percent reduction in regional V_T CoV of the direct method relative to the indirect method by count level and region at iteration 2, for (a) [^{11}C]AFM and (b) [^{11}C]UCB-J.

	Simulation		[^{11}C]AFM		[^{11}C]UCB-J	
	K_1	V_T	K_1	V_T	K_1	V_T
Indirect	1.7	2.1	1.9	2.2	2.1	2.5
Direct	2.1	2.2	2.0	1.9	1.9	2.0

Table S12. Ratio of parameter CoV at 5% count level to CoV at 20% count level at iteration 2, averaged across regions.

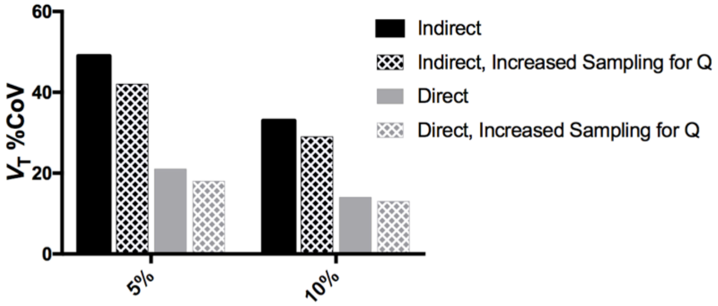

Figure S11. Impact of increased sampling of LORs in estimation of the sensitivity image on V_{T} CoV. For the 5% count level, LOR sampling was quadrupled; for the 10% count level, LOR sampling was doubled. Results are shown at iteration 2.

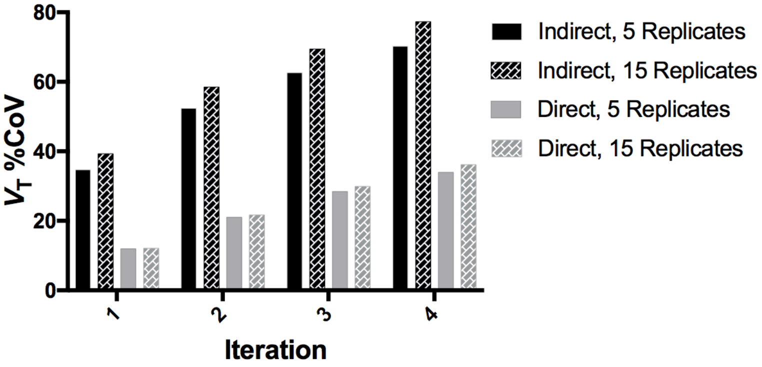

Figure S12. Regional V_{T} %CoV estimates for the simulated data, at the 5% count level, estimated from either 5 or 15 replicates for the indirect and direct methods.

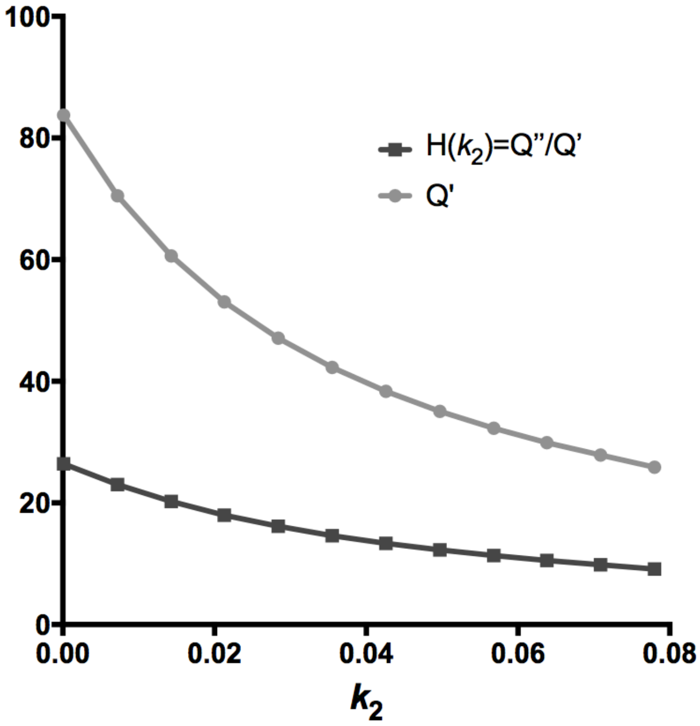

Figure S13. Example plots of H($k_{2,j}$) (used in k_2 update) and Q' (used in K_1 update), for a single voxel from one acquisition. Each symbol denotes a k_2 value at which Q' and Q'' images were computed. These curves are representative; the general monotonic shape of these curves is similar for all voxels.

Appendix A: Invertibility of $\boldsymbol{H(k_2)}$

In the derivation in the main text, an important requirement is that the function $H(k_2)$ is invertible. Since this function is intrinsically positive, this requirement can be met by demonstrating that the function is monotonically decreasing over the prescribed range of k_2.

We denote the numerator and denominator of $H(k_{2,j})$ as $N(k_{2,j})$ and $D(k_{2,j})$, respectively:

$$H(k_{2,j}) = \frac{\sum_{it\tau}(t-\tau)c_{ijt}L_tP_\tau e^{-k_{2,j}(t-\tau)}}{\sum_{it\tau} c_{ijt}L_tP_\tau e^{-k_{2,j}(t-\tau)}} = \frac{N(k_{2,j})}{D(k_{2,j})}$$

since $N(k_{2j}) = -\frac{dD(k_{2j})}{dk_{2j}}$, then $H(k_{2j}) = -\frac{d\ln(D(k_{2j}))}{dk_{2j}}$ and $\frac{dH(k_{2j})}{dk_{2j}} = -\frac{d^2\ln(D(k_{2j}))}{{dk_{2j}}^2}$

So demonstrating that $H(k_{2,j})$ is decreasing, is equivalent to showing that $\ln\left(D(k_{2,j})\right)$ is convex.

Applying the quotient rule to differentiate $H(k_{2,j})$ leads to:

$$\frac{dH(k_{2,j})}{dk_{2,j}}D(k_{2,j})^2 = \frac{dN(k_{2,j})}{dk_{2,j}}D(k_{2,j}) - \frac{dD(k_{2,j})}{dk_{2,j}}N(k_{2,j})$$

Since $N(k_{2,j}) = -\frac{dD(k_{2,j})}{dk_{2,j}}$, we have:

$$\frac{dH(k_{2,j})}{dk_{2,j}}D(k_{2,j})^2 = \frac{dN(k_{2,j})}{dk_{2,j}}D(k_{2,j}) + N(k_{2,j})^2$$

Written out, this is:

$$\frac{dH(k_{2,j})}{dk_{2,j}}D(k_{2,j})^2 = -\left(\sum_{it\tau}(t-\tau)^2c_{ijt}L_tP_\tau e^{-k_{2,j}(t-\tau)}\right)\left(\sum_{it\tau}c_{ijt}L_tP_\tau e^{-k_{2,j}(t-\tau)}\right) + \left(\sum_{it\tau}(t-\tau)c_{ijt}L_tP_\tau e^{-k_{2,j}(t-\tau)}\right)^2$$

Collapsing the sums over the lines of response *i*:

$$\frac{dH(k_{2,j})}{dk_{2,j}}D(k_{2,j})^2 = -\left(\sum_{t\tau}(t-\tau)^2Q_{jt}L_tP_\tau e^{-k_{2,j}(t-\tau)}\right)\left(\sum_{t\tau}Q_{jt}L_tP_\tau e^{-k_{2,j}(t-\tau)}\right) + \left(\sum_{t\tau}(t-\tau)Q_{jt}L_tP_\tau e^{-k_{2,j}(t-\tau)}\right)^2$$

where $Q_{jt} = \sum_i \ c_{ijt}$ is the value of the sensitivity image for voxel *j* at time *t*.

Now, change variables by defining $\Delta t = t - \tau$. Since $\tau \in [0, t]$, then $\Delta t \in [0, t]$, and:

$$\frac{dH(k_{2,j})}{dk_{2,j}}D(k_{2,j})^2 = -\left(\sum_{t,\Delta t}\Delta t^2Q_{jt}L_tP_{t-\Delta t}e^{-k_{2,j}\Delta t}\right)\left(\sum_{t,\Delta t}Q_{jt}L_tP_{t-\Delta t}e^{-k_{2,j}\Delta t}\right) + \left(\sum_{t,\Delta t}\Delta tQ_{jt}L_tP_{t-\Delta t}e^{-k_{2,j}\Delta t}\right)^2$$

Define:

$$W_{j,\Delta t} = e^{-k_{2,j}\Delta t} \sum_{t} Q_{jt} L_t P_{t-\Delta t}$$

$W_{j,\Delta t}$ is a positive quantity. Then:

$$\frac{dH(k_{2,j})}{dk_{2,j}} D(k_{2,j})^2 = -\left(\sum_{\Delta t} \Delta t^2 W_{j,\Delta t}\right)\left(\sum_{\Delta t} W_{j,\Delta t}\right) + \left(\sum_{\Delta t} \Delta t W_{j,\Delta t}\right)^2$$

This expression is of the form

$$\frac{dH(k_{2,j})}{dk_{2,j}} D(k_{2,j})^2 = -\left(\sum_{\Delta t} u_{\Delta t}{}^2\right)\left(\sum_{\Delta t} v_{\Delta t}{}^2\right) + \left(\sum_{\Delta t} u_{\Delta t} v_{\Delta t}\right)^2$$

with

$$v_{\Delta t} = \sqrt{W_{j,\Delta t}}$$

and

$$u_{\Delta t} = \Delta t \sqrt{W_{j,\Delta t}}$$

Thus, $\frac{dH(k_{2,j})}{dk_{2,j}} D(k_{2,j})^2 \leq 0$, due to the Cauchy-Schwarz inequality $\left(\sum_i u_i v_i\right)^2 \leq \left(\sum_i u_i{}^2\right)\left(\sum_i v_i{}^2\right)$. So $\frac{dH(k_{2,j})}{dk_{2,j}} \leq 0$ and $H(k_{2,j})$ is monotonically decreasing.

Moreover, the Cauchy-Schwarz inequality becomes an equality if and only if:

1) $v_{\Delta t} = \lambda u_{\Delta t}$, which is not possible here, or
2) $u_{\Delta t}$ or $v_{\Delta t}$ is all zeros, which here means $W_{j,\Delta t} = 0$ for all Δt, i.e., $Q_{jt} = 0$ for all t, which means the voxel j is outside the field of view.

Therefore in practice, $H(k_{2,j})$ is strictly monotonically decreasing for all voxels of interest.

Finally, at the beginning of this appendix we noted that demonstrating that $H(k_{2,j})$ is decreasing is equivalent to showing that $\ln\left(D(k_{2,j})\right)$ is convex. With the notation introduced above, $\ln\left(D(k_{2,j})\right) = \ln\left(\sum_{\Delta t} W_{j,\Delta t}\right)$, which is the logarithm of a sum of (positively) weighted (decreasing) exponentials. It is a classical result that the logarithm of a sum of exponentials is convex, and this demonstration of the monotonicity of $H(k_{2,j})$ was inspired by the demonstration of that classical result.

Appendix B: Simulation Methodology

I. Overview

The ability to simulate PET list mode data is essential for validating list mode reconstruction algorithms. Most existing tools for PET list mode simulation use Monte Carlo methods to generate random positron emissions one-by-one, subjecting each to a sequence of explicitly modeled physical effects to determine which will ultimately be detected. Some examples of open source software packages for Monte Carlo simulations include GATE [1] and PeneloPET [2]. While such packages provide realistic modeling of physical effects, the potential mismatch between the physics model used in the simulation and that used in the reconstruction can be confounding when trying to validate new reconstruction algorithms, independent of the modeling of physical effects. The motivation for providing a new simulation tool in the MOLAR framework was to enforce perfectly consistent physics modeling between the simulation and reconstruction, to disentangle the validation of the physics model (not the focus of this manuscript) from that of the other details of the reconstruction algorithm such as motion correction, direct parametric implementations, regularization, etc.

The primary input to any simulation is a representation of the desired ground truth activity distribution, which may be time-varying. For static PET, such input might simply be a single emission image. For dynamic PET, the input can be provided as a sequence of emission images; or a set of parametric images, along with the corresponding input function and model definition; or an image of integer labels denoting regions, along with a set of time activity curves corresponding to each region.

Given such input, the expected value of the emissions at a particular time along any line-of-response (LOR) defined by a pair of detectors (and possibly a time-of-flight bin) can be computed via a forward projection model, which could include attenuation, normalization, decay, dead time, resolution, scatter, randoms, etc. In theory, one could compute the expected value for every possible LOR at every time step of the simulation, and simulate activity along each with a random variate drawn from a Poisson distribution with parameter set to the corresponding expected value. However, such an approach is not computationally feasible for simulating modern systems with large numbers of detectors, especially in the presence of rapid changes in activity, either due to tracer kinetics or motion. Therefore, we developed an approach that randomly samples a small fraction of the total possible LORs at each time step, computes the expected number of events for these LORs, adjusts the computed expected values with an undersampling correction factor, and then generates an event list by randomly including some of the selected LORs, so that the simulated list mode data follow a Poisson distribution.

II. Theory and Implementation

Distribution of LOR Selection Frequency. Let I be the total number of possible LORs in the system, and S be the number of simulated LORs. Then the undersampling factor U is I/S (i.e. the fraction of LORs sampled is $1/U$).

Because we would like to sample with replacement (both for ease of implementation and for realism), it is possible that LOR i can be picked more than once, i.e., n_i times. Let N_i be a random variable representing the number of times LOR i is selected; N_i will be binomially distributed, with the probability of picking any particular LOR once, $1/I$ (assuming uniform LOR sampling). If I is large enough, then the distribution of N_i is well-approximated as Poisson with parameter $1/U$:

$$\Pr(N_i = n_i) = \frac{(1/U)^{n_i} e^{-1/U}}{n_i!} \tag{S1}$$

.

Distribution of Events. Once a LOR is selected, a random variable must be used to decide if this randomly selected LOR should be included in the simulated list mode dataset. Given that the LOR selection distribution is Poisson, the next step is to determine a suitable probability distribution to draw from for each LOR such that the simulated data will have Poisson statistics. In other words, once an LOR has been selected and the expected value $\hat{y}$ has been computed by forward projection, we need to model the yes/no decision of whether to write an event packet corresponding to that LOR to the list mode file. Clearly, if the computed expected value $\hat{y}$ =0, no event should be written, and the greater $\hat{y}$ is, the higher the probability of "keeping" that event should be.

The process of drawing from a Poisson distribution for LOR selection (corresponding to random variable *N*), and, for each time the LOR is chosen, subsequently drawing from a second distribution that will somehow be parameterized by $\hat{y}$ (let the corresponding random variable for this be *X*), can be modeled with a Compound Poisson distribution. This is the distribution of *Z*, the counts on a particular LOR.

While we haven't yet established the distribution of *X*, we can write the distribution of *Z* conditioned on *N*:

$$Z|N = \sum_{n=1}^{N} X_n, \tag{S2}$$

where X_1, X_2, X_3 ... are i.i.d. random variables, each with sample space of 1 (include the LOR), or 0 (discard the LOR). This says, to get the distribution of counts Z_i on LOR *i* given that LOR *i* was picked a certain number of times, add up the probability of writing an event packet for LOR *i*, once for each time it was picked.

Note that the joint distribution of *Z* and *N* isPr(*Z*,*N*) = Pr(*Z*|*N*)Pr(*N*) (from the chain rule of probability); then the Compound Poisson Distribution is the distribution of *Z*, which can be found by marginalizing the joint distribution Pr(*Z*,*N*) over *N*:

$$\Pr(Z) = \sum_{0}^{\infty} \Pr\,(Z|N = n)\Pr\,(N = n) \tag{S3}$$

As stated above, the count data, which will be denoted by random variable *Z*, should be Poisson. So, given *N* ~ Poisson and *Z* ~ Poisson, we would like to identify the distribution of *X*.

Conveniently, it can be shown that *X* is Bernoulli-distributed with parameter $U\hat{y}$:

$$\Pr(Z = z) = \sum_{n=z}^{\infty} \Pr(Z = z|N = n)\Pr(N = n)$$

$$= \sum_{n=z}^{\infty}\left(\sum_{i=1}^{n}(U\hat{y})^{z}(1-U\hat{y})^{1-z}\right)\left(\frac{(1/U)^{n}e^{-1/U}}{n!}\right)$$

The sum of n Bernoulli trials with identical probabilities is binomial, so:

$$= \sum_{n=z}^{\infty}\left(\frac{n!}{z!\,(n-z)!}(U\hat{y})^{z}(1-U\hat{y})^{n-z}\right)\left(\frac{(1/U)^{n}e^{-1/U}}{n!}\right)$$

$$= \frac{(U\hat{y})^{z}e^{-1/U}}{z!}\sum_{n=z}^{\infty}\frac{(1/U)^{n}}{(n-z)!}(1-U\hat{y})^{n-z}$$

$$= \frac{(U\hat{y})^{z}e^{-1/U}}{z!}\sum_{n=0}^{\infty}\frac{(1/U)^{(z+n)}}{(z+n-z)!}(1-U\hat{y})^{z+n-z}$$

$$= \frac{(U\hat{y})^{z}e^{-\frac{1}{U}}(1/U)^{z}}{z!}\sum_{n=0}^{\infty}\frac{(1/U)^{n}}{n!}(1-U\hat{y})^{n}$$

$$= \frac{(U\hat{y})^{z}e^{-\frac{1}{U}}(1/U)^{z}}{z!}\cdot e^{1/U(1-U\hat{y})}$$

$$= \frac{e^{-\hat{y}}\hat{y}^{z}}{z!}.\ \blacksquare \tag{S4}$$

In other words, each time a LOR is selected, the decision to include it in the simulated list mode file should be based on a random realization from a Bernoulli random variable with parameter $U\hat{y}_i$.

Therefore, to simulate a random variable from a Poisson distribution with expected value $\hat{y}_i$ for a given (randomly selected) LOR *i*, we can draw from a Bernoulli distribution with parameter $U\hat{y}_i$. This parameter represents a probability, so $U\hat{y}_i \leq 1$. Because $\hat{y}_i$ is determined by the ground truth emission image we are trying to simulate, the sampling rate of LORs must be set high enough (*U* must be small enough) that we never encounter a $U\hat{y}_i > 1$. Given maximum expected value $\hat{y}_{\max}$ we must sample at least ($\hat{y}_{\max} \times I$) LORs.

LOR Sampling Scheme. We began this Appendix with the declaration that it is impractical to compute expected counts for every possible LOR for every time increment of a simulation based on a modern PET system. However, if we sample too few LORs, we risk having a $U\hat{y}_i > 1$ for one or more LORs, which is not valid as a parameter of the Bernoulli distribution.

A particularly disagreeable scenario would be one where the average $\hat{y}$ across all LORs for a given simulation is fairly low, giving us hope that we can use a relatively large *U* (giving us computational efficiency); but then we happen upon a small handful of LORs with diabolically large ŷ values; in that case, the maximal appropriate value for *U* must be small, so that $U\hat{y}_i < 1$ for all LORs. This suggests that we would prefer non-uniform sampling of LORs, to sequester "rogue" LORs demanding heavy sampling (low *U*) from the locales of projection space where a more gentle touch would do (high *U*).

So, rather than using a global undersampling factor *U*, we can parcellate projection space into *P* "parcels" of LORs. LORs will be grouped such that each parcel contains LORs with similar $\hat{y}$ values.

Parcel p will have undersampling factor U_p, which is the number of LORs in parcel p (I_p), divided by the number of LORs sampled from parcel p (S_p). Each S_p will be defined by the largest $\hat{y}$ in parcel p, so that the Bernoulli parameter is less than 1:

$$S_p > \hat{y}_{\mathrm{max},p} \times I_p \ . \qquad \text{(S5)}$$

In practice, this inequality is enforced by setting S_p such that $U_p \times \hat{y}_{\mathrm{max},p}$ equals a value slightly less than 1 (e.g. 0.9). There must be sufficient LORs per parcel to ensure that the LOR selection distribution for a given parcel remains Poisson. This should be a fairly easy assumption to uphold, given the large number of LORs in the system and the relatively small number of parcels we'd use in practice (on the order of 10-20).

In determining the parcellation, since we would like not to forward project every single possible LOR in the system (which would be one unprofitable way to determine where the large $\hat{y}$ values are), the parcellation is determined using only a very coarse sinogram representation of the system. Each detector position in the system can be defined by its axial slice and angle around the detector ring. Detectors are grouped into bins of similar slice and angle. Then our "coarse sinogram" is represented as a 4-dimensional array, where the first two dimensions are the angle and slice bins of detector 1, and the last two dimensions are the angle and slice bins of detector 2 (for the LOR subtended by detectors 1 and 2). Note, this array will be diagonally symmetric: [Detector 1, Detector 2] refers to the same LOR as does [Detector 2, Detector 1].

To initialize the parcellation at the start of a simulation, we first define the axial and angular sampling desired for the coarse sinogram (reasonable values for these would be ~100 angles and ~20 axial slices). To get approximate $\hat{y}_b$ for each sinogram bin b, simple ray-tracing can be used to forward project the ground truth emission image. For simulations without randoms and scatter, sinogram bins which do not contain LORs that intersect the object will not be assigned to a parcel, i.e., we don't waste time forward projecting LORs that have $\hat{y} = 0$. (The emission image can be blurred to ensure that sinogram bins on the very edge of the object will still be included in the simulation.) Parcels can then be defined, for instance, by dividing the range of non-zero $\hat{y}_b$ values evenly into P segments. S_p, the "quota" for LORs to sample from each parcel is computed using Equation S5, with a margin left such that all Bernoulli parameters are strictly less than 1. The parcellation does not have to be perfect - even a rough idea of where the counts are helps to sample more efficiently.

For each time step of a simulation, the emission image is determined from user-defined input. An LOR is selected by randomly generating two numbers corresponding to detector indices. After checking that the LOR is valid, the parcel p that the LOR belongs to is determined. If the sampling quota S_p for this parcel has already been met, the LOR is simply ignored and a new one selected. If the quota has not been met, a forward projection with all desired physical modeling is performed to get the exact $\hat{y}$ for that LOR. The quotas are based on approximate $\hat{y}$ values, so during simulation we might encounter a $\hat{y}$ value that is too high ($U_p\hat{y}_i > 1$); in this case, the quota is dynamically adjusted to account for this. As long as there exists a quota that has not been met, LORs are randomly generated. As the simulation runs, we keep track of the maximum $\hat{y}$ value encountered in each bin of the coarse sinogram. This information is used to hone the parcellation at regular intervals throughout the simulation, which is particularly useful for simulating dynamic PET, where count rates change in time and space.

III. Summary of Simulation Algorithm

1. Expected Value Approximation. For a fairly low-resolution sinogram representation, the maximal expected value of LORs in each sinogram bin is estimated, assuming that LORs within a sinogram bin will have similar expected values.
2. Parcellation. Sinogram bins are combined into *P* "parcels" of similar expected values. Currently, the range of maximal expected values is divided linearly between the minimum and the maximum.
3. Quota Assignment. The number of LORs that would need to be sampled to produce expected values that are <1 is estimated per parcel; these are the "quotas" for each parcel, per timestamp.

(Steps 1-3 are repeated periodically for dynamic simulations.)

4. LOR Generation. LORs are randomly selected (with replacement) until all quotas are met, on a per-timestamp basis. If we pick an LOR that belongs to a parcel for which the quota is already met, we throw it out; other LORs are kept.
5. Forward Projection. For all LORs that were kept, we perform a forward projection including all desired modeling (attenuation, motion, etc). This gives us an "exact" expected value for each LOR. Along the way, we check that the expected value corrected for undersampling (at the current undersampling rate for this LOR's parcel) does not exceed 1. If it does, we dynamically adjust the quota upwards accordingly.
6. Correction for Undersampling. Once all quotas have been met for a timestamp, the expected value of each randomly sampled LOR is adjusted to correct for the final undersampling rate of the parcel to which that LOR belongs.
7. Bernoulli Trials. Using the corrected expected value of each LOR as the parameter of a Bernoulli distribution, we generate a random variate for each. Whenever we obtain a value of 1, an event packet for this LOR is written to the list mode file.

(Steps 4-7 are repeated for all timestamps in the desired simulation duration.)

IV. Parallelization Strategy

As with MOLAR reconstructions, MOLAR simulations are parallelized to run on a high-performance computing cluster using MPI. Each process has an associated intermediate list mode file, to which it writes the events as it simulates them. Each timestamp of the simulation is handled by exactly one MPI process. At the end of the simulation, the process-specific list mode files are interleaved into one final list mode file containing all events from all timestamps; this step is performed serially on process 0. This scheme is more time and memory efficient than requiring periodic communication between slave and master nodes to coordinate the writing of a single file without intermediate writes. The final step of list mode interleaving (for typical count levels and numbers of processors) takes on the order of a couple minutes.

1. Jan S, Santin G, Strul D, Staelens S, Assie K, Autret D, et al. GATE: a simulation toolkit for PET and SPECT. Physics in medicine and biology. 2004;49:4543.

2. España S, Herraiz J, Vicente E, Vaquero JJ, Desco M, Udías JM. PeneloPET, a Monte Carlo PET simulation tool based on PENELOPE: features and validation. Physics in medicine and biology. 2009;54:1723.