



Athinoula A.

**Martinos**  
**Center**  
For Biomedical Imaging



MASSACHUSETTS  
GENERAL HOSPITAL



HARVARD  
MEDICAL SCHOOL



Massachusetts  
Institute of  
Technology

# Multiparametric mapping with Pulseq

Shohei Fujita

BRAIN (Bilgic Reconstruction Acquisition for Imaging Neuroscience) Lab

# Objectives

1. Demonstrate the actual workflow of implementing and executing custom pulse sequences in Pulseq
2. Introduce an application of Pulseq for cross-platform, multiparametric mapping

## Background

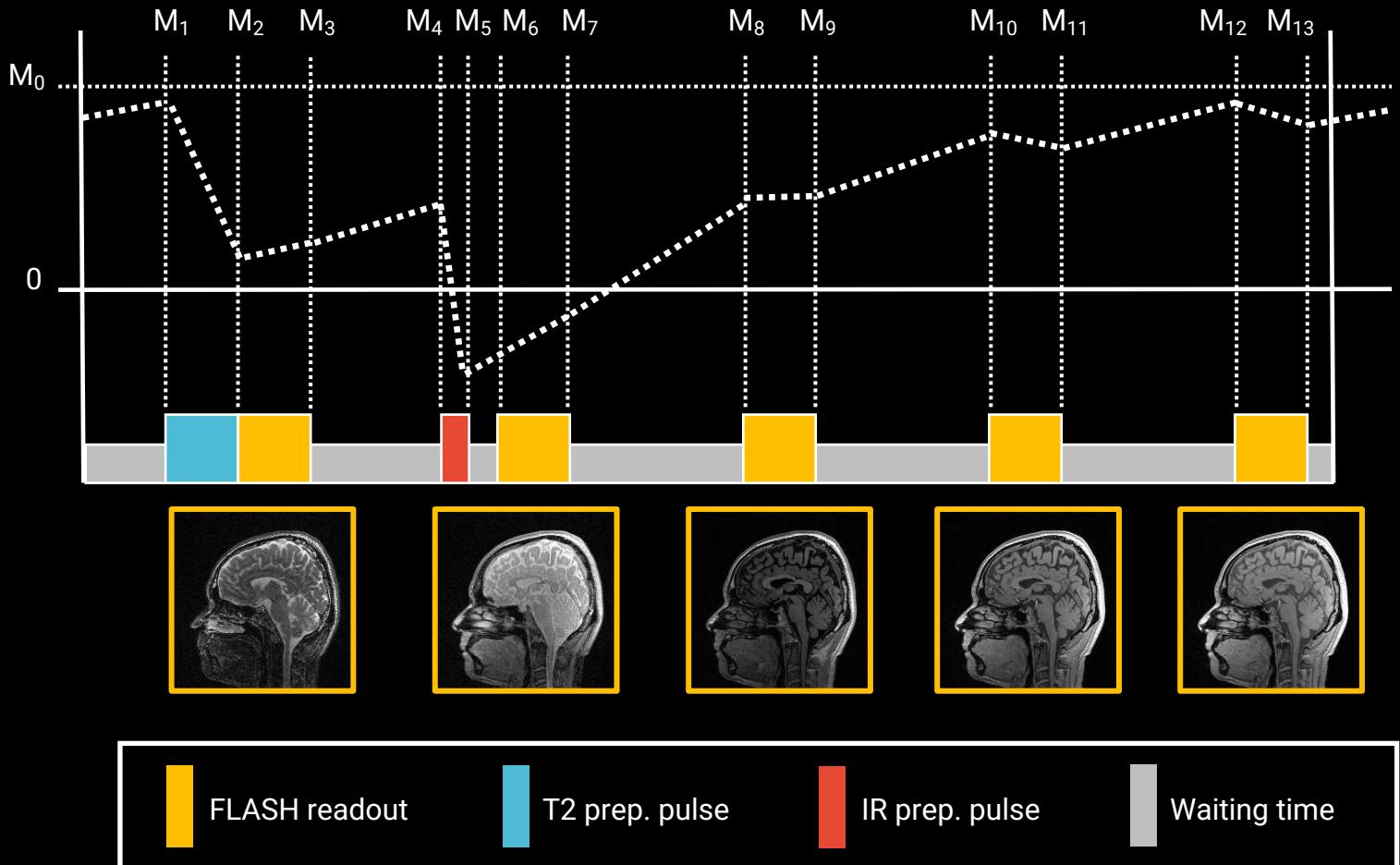
- Multi-site data cannot be naively pooled together for analysis due to large inter-scanner differences

Address the unmet need for a cross-platform, multiparametric technique to facilitate data harmonization across different sites

# 3D-QALAS: Rapid whole-brain T1 and T2 mapping

3D-QALAS<sup>1,2</sup>:

- 3D-quantification using an interleaved Look–Locker acquisition sequence with a T2 preparation pulse
- Multi-acquisition 3D gradient echo
- Signal modeling-based parameter fitting on the five source images to derive quantitative maps

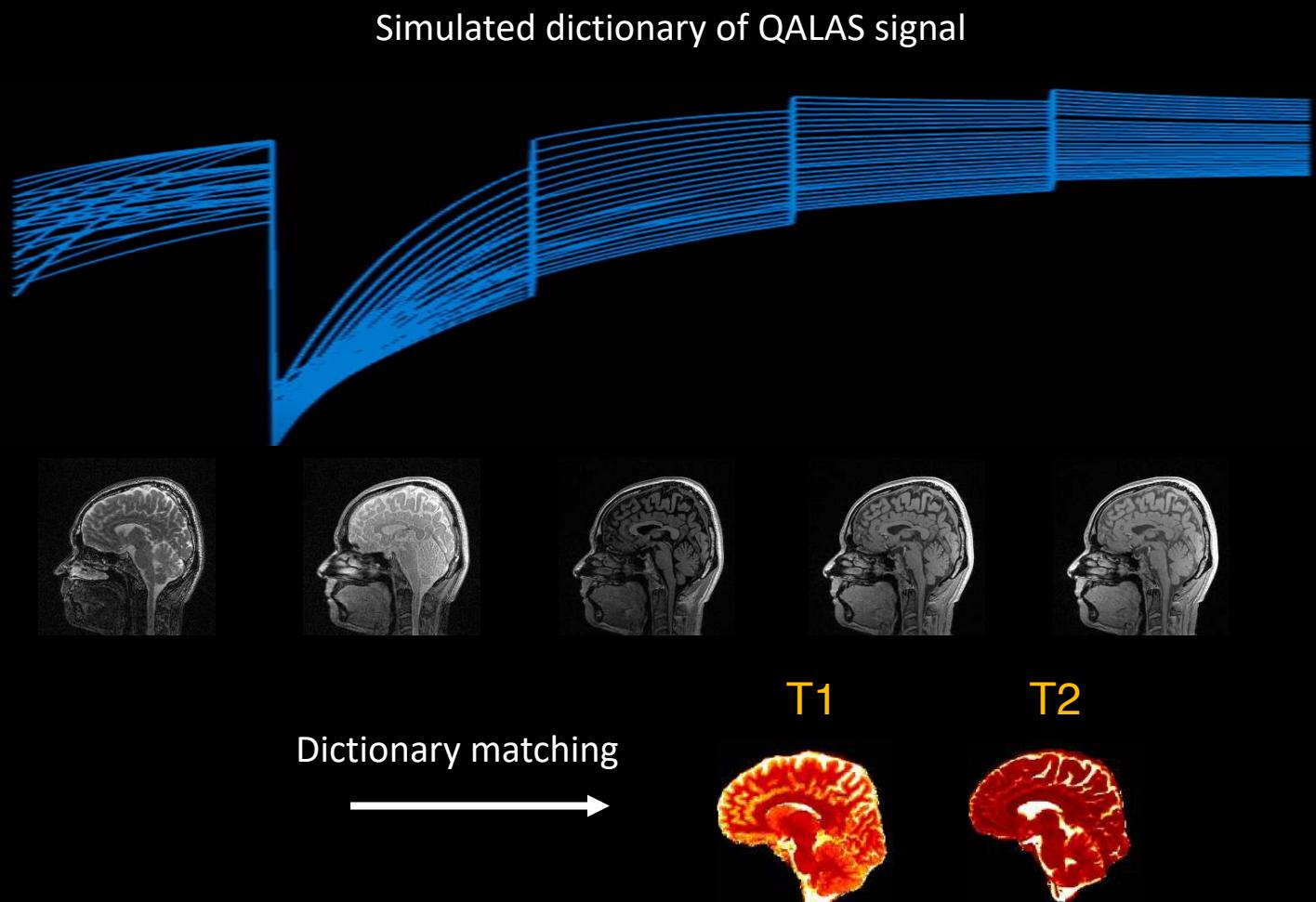


[1] Kvernby et al. *J Cardiovasc Magn Reson* 2014; [2] Fujita et al. *Magn Reson Imaging* 2019

# 3D-QALAS: Rapid whole-brain T1 and T2 mapping

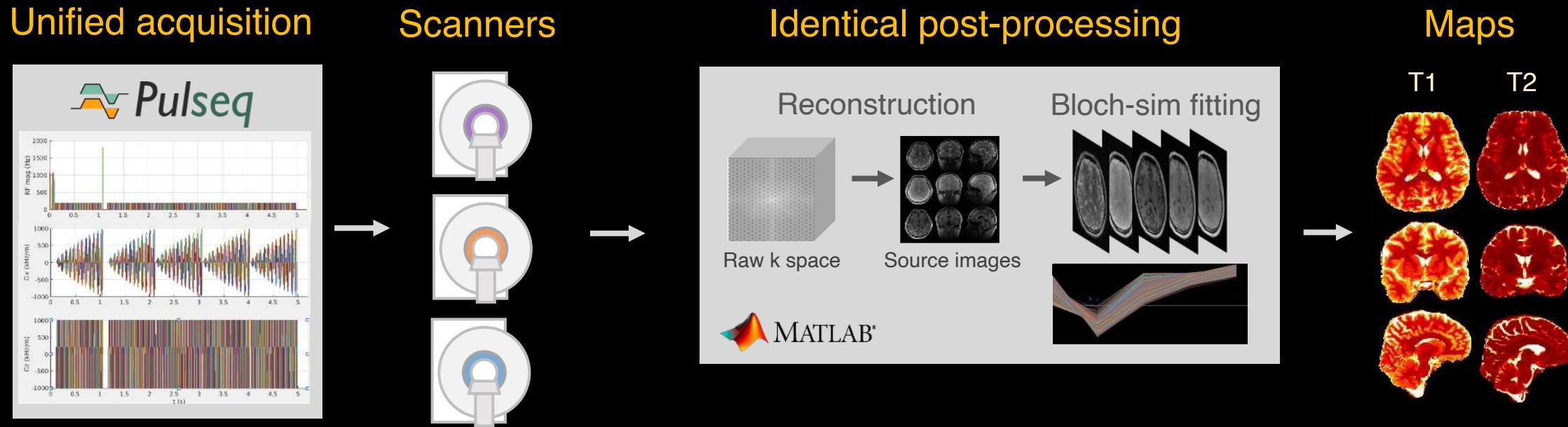
3D-QALAS<sup>1,2</sup>:

- 3D-quantification using an interleaved Look–Locker acquisition sequence with a T2 preparation pulse
- Multi-acquisition 3D gradient echo
- Signal modeling-based parameter fitting on the five source images to derive quantitative maps



[1] Kvernby et al. *J Cardiovasc Magn Reson* 2014; [2] Fujita et al. *Magn Reson Imaging* 2019

# Study overview



Site	Manufacturer	Scanner model	Software version	Evaluation
Site A	Siemens Healthcare, Erlangen, Germany	Prisma	VE11C	-
Site A	Siemens Healthcare, Erlangen, Germany	Prisma	XA30	Cross-software
Site A	Siemens Healthcare, Erlangen, Germany	Skyra	VE11C	Cross-scanner
Site B	Siemens Healthcare, Erlangen, Germany	Prisma	VE11C	Cross-site
Site C	GE HealthCare, WI, USA	Premier XT	MR30	Cross-vendor

# Sequence implementation on Pulseq



# Sequence implementation on Pulseq: writeSeq.m

```
%-----  
% Define high-level parameters  
%-----  
  
% Set system parameters  
sys = mr.opts('MaxGrad', 32, 'GradUnit', 'mT/m', ...  
    'MaxSlew', 160, 'SlewUnit', 'T/m/s', ...  
    'rfDeadTime', 100e-6, ...  
    'rfRingdownTime', 60e-6, ...  
    'adcDeadTime', 40e-6, ...  
    'adcRasterTime', 2e-6, ...  
    'gradRasterTime', 20e-6, ...  
    'blockDurationRaster', 20e-6, ...  
    'B0', 3.0); |  
  
gradRasterTime (Siemens) = 10us  
gradRasterTime (GE) = 4us  
  
% Create sequence object and define FOV, resolution, and other high-level  
seq = mr.Sequence(sys); % Create a new sequence object  
fov=[192 168 168]*1e-3; % Define FOV and resolution  
N = [192 168 56];  
  
Nx = N(1);  
Ny = N(2);  
Nz = N(3);  
  
bandwidth = 340; % Hz/pixel  
flip_angle = 4; % degrees  
rfSpoilingInc=117; % RF spoiling increment  
dwell = 16e-6;  
Tread = dwell*Nx;
```

# Sequence implementation on Pulseq: writeSeq.m

```
%-----  
% Prepare sequence blocks  
%-----  
  
deltak = 1 ./ fov;  
gx = mr.makeTrapezoid('x',sys,'Amplitude',Nx*deltak(1)/Tread,'FlatTime',ceil(Tread/sys.gradRasterTime)*sys.g  
gxPre = mr.makeTrapezoid('x',sys,'Area',-gx.area/2); % Gx prewinder  
gxSpoil = mr.makeTrapezoid('x',sys,'Area',gx.area); % Gx spoiler  
  
% Make trapezoids for inner loop to save computation  
  
gyPre = mr.makeTrapezoid('y','Area',deltak(2)*(Ny/2),'Duration',mr.calcDuration(gxPre), 'system',sys);  
gzPre = mr.makeTrapezoid('z','Area',deltak(3)*(Nz/2),'Duration',mr.calcDuration(gxPre), 'system',sys);  
  
gyReph = mr.makeTrapezoid('y','Area',deltak(2)*(Ny/2),'Duration',mr.calcDuration(gxSpoil), 'system',sys);  
gzReph = mr.makeTrapezoid('z','Area',deltak(3)*(Nz/2),'Duration',mr.calcDuration(gxSpoil), 'system',sys);  
  
stepsY=((traj(:,2)-1)-Ny/2)/Ny*2;  
stepsZ=((traj(:,1)-1)-Nz/2)/Nz*2;  
  
% Create non-selective pulse  
rf = mr.makeBlockPulse(flip_angle*pi/180, sys, 'Duration', 0.1e-3);
```

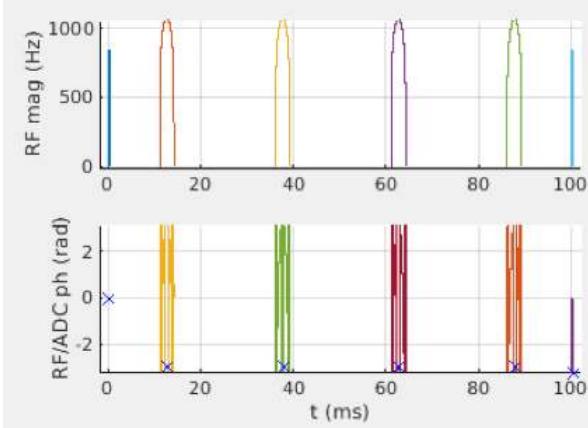
# Sequence implementation on Pulseq: writeSeq.m

```
% Prep pulses
% T2 prep and IR prep pulse are imported from external txt files
% txt file is in mag and phase, while mr.makeArbitraryRf assumes real and
t2prep = readmatrix('csv_imports/T2prep.txt');
Re = t2prep(:,1) .* cos(t2prep(:,2));
Im = t2prep(:,1) .* sin(t2prep(:,2));
t2prep_pulse = mr.makeArbitraryRf((Re+Im*1i).', 380.4*pi/180, 'system',sy

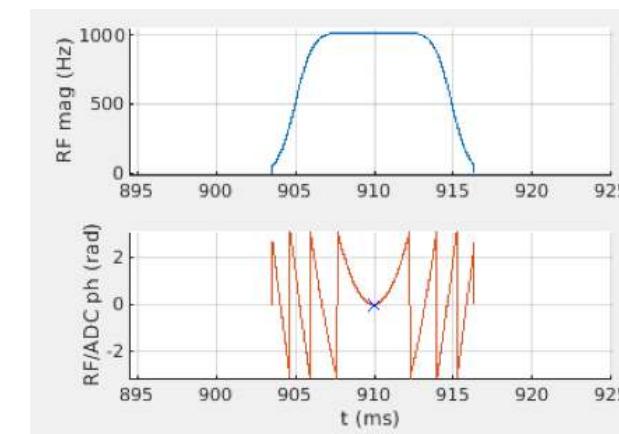
IRprep = readmatrix('csv_imports/IRprep.txt');
Re = IRprep(:,1) .* cos(IRprep(:,2));
Im = IRprep(:,1) .* sin(IRprep(:,2));
IRprep_pulse = mr.makeArbitraryRf((Re+Im*1i).', 850*pi/180, 'system',sys,
```

mr.makeArbitraryRF

T2prep



IR



# Sequence implementation on Pulseq: writeSeq.m

```
%-----  
% Build sequence  
%-----  
  
for iZ = 1:nTR  
    rf_phase=0;  
    rf_inc=0;  
  
    for iY = 1:nETL  
  
        % Calculate index for ky-kz look up table  
        index = iY + (iZ-1)*nETL;  
  
        % RF spoiling  
        rf.phaseOffset=rf_phase/180*pi;  
        adc.phaseOffset=rf_phase/180*pi;  
        rf_inc=mod(rf_inc+rfSpoilingInc, 360.0);  
        rf_phase=mod(rf_phase+rf_inc, 360.0);      %increment RF phase  
  
        % Excitation  
        if iY == 1  
            seq.addBlock(rf,mr.makeLabel('SET', 'LIN', segmentID)); % segment  
        else  
            seq.addBlock(rf);  
        end  
  
        % Encoding  
        seq.addBlock(gxPre, ...  
                    mr.scaleGrad(gyPre,-stepsY(index)), ...  
                    mr.scaleGrad(gzPre,-stepsZ(index)));    % Gz, Gy blips, Gx pre-win
```

# Sequence implementation on Pulseq: timing check, PNS check

```
%-----
% Check timing and write sequence
%-----

% check whether the timing of the sequence is correct
[ok, error_report]=seq.checkTiming;

if (ok)
    fprintf('Timing check passed successfully\n');
else
    fprintf('Timing check failed! Error listing follows:\n');
    fprintf([error_report{:}]);
    fprintf('\n');
end

% PNS check
[pns_ok, pns_n, pns_c, tpns]=seq.calcPNS('/autofs/cluster/berkin/fujita/m'

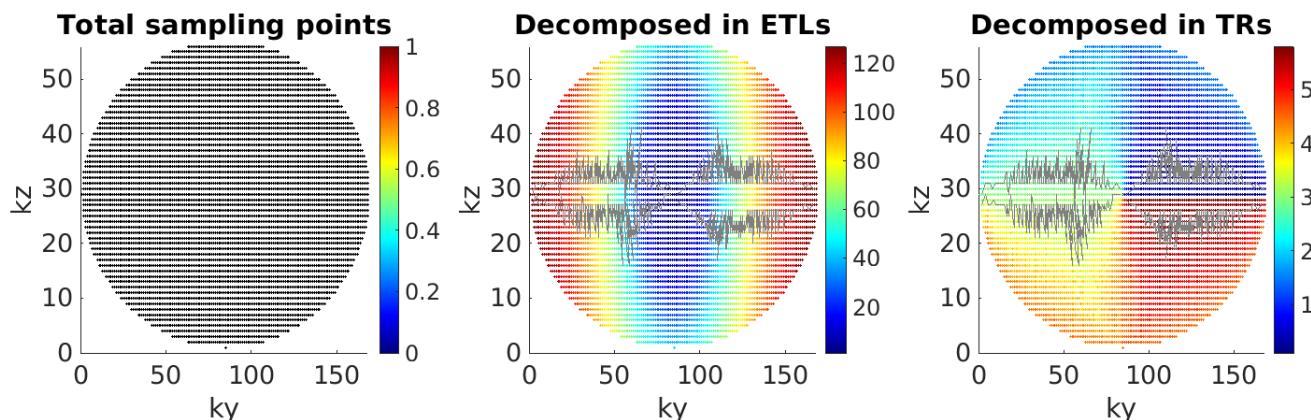
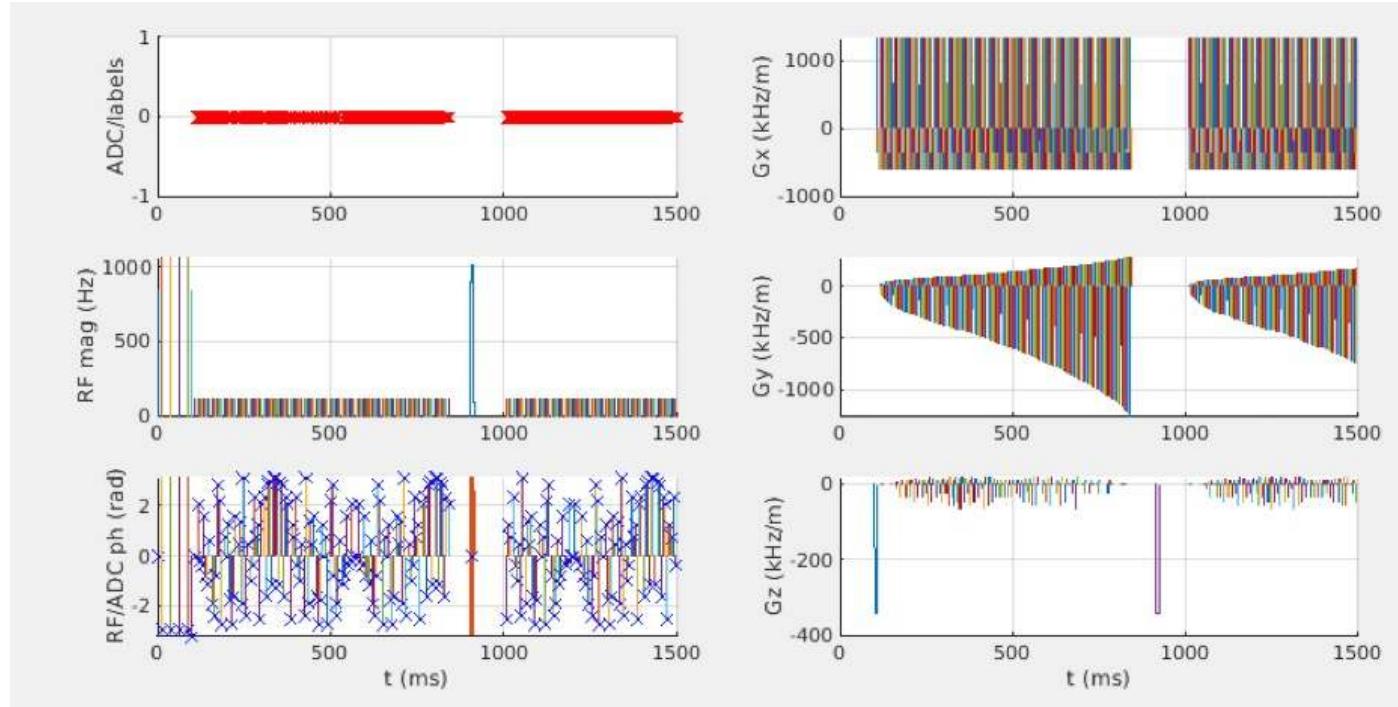
if (pns_ok)
    fprintf('PNS check passed successfully\n');
else
    fprintf('PNS check failed! The sequence will probably be stopped by t'
end

% Set definitions for recon
seq.setDefinition('FOV', fov);
seq.setDefinition('Matrix', N);
seq.setDefinition('nETL', nETL);
seq.setDefinition('nTR', nTR);
seq.setDefinition('traj_z', traj(:,1));
seq.setDefinition('traj_y', traj(:,2));

% plot
seq.plot('TimeRange',[0 1.5], 'timeDisp', 'ms');

% Write to pulseq file
filename = strrep(mfilename, 'write', '');
seq.write([filename,'.seq']);
```

# Sequence implementation on Pulseq



FOV: 192x168x168

Matrix: 192x168x56

TR: 4500ms

TE: 2.3ms

Echo spacing: 5.8ms

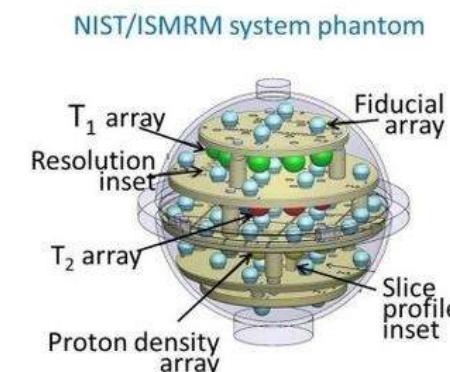
FA: 4°

ETL: 127

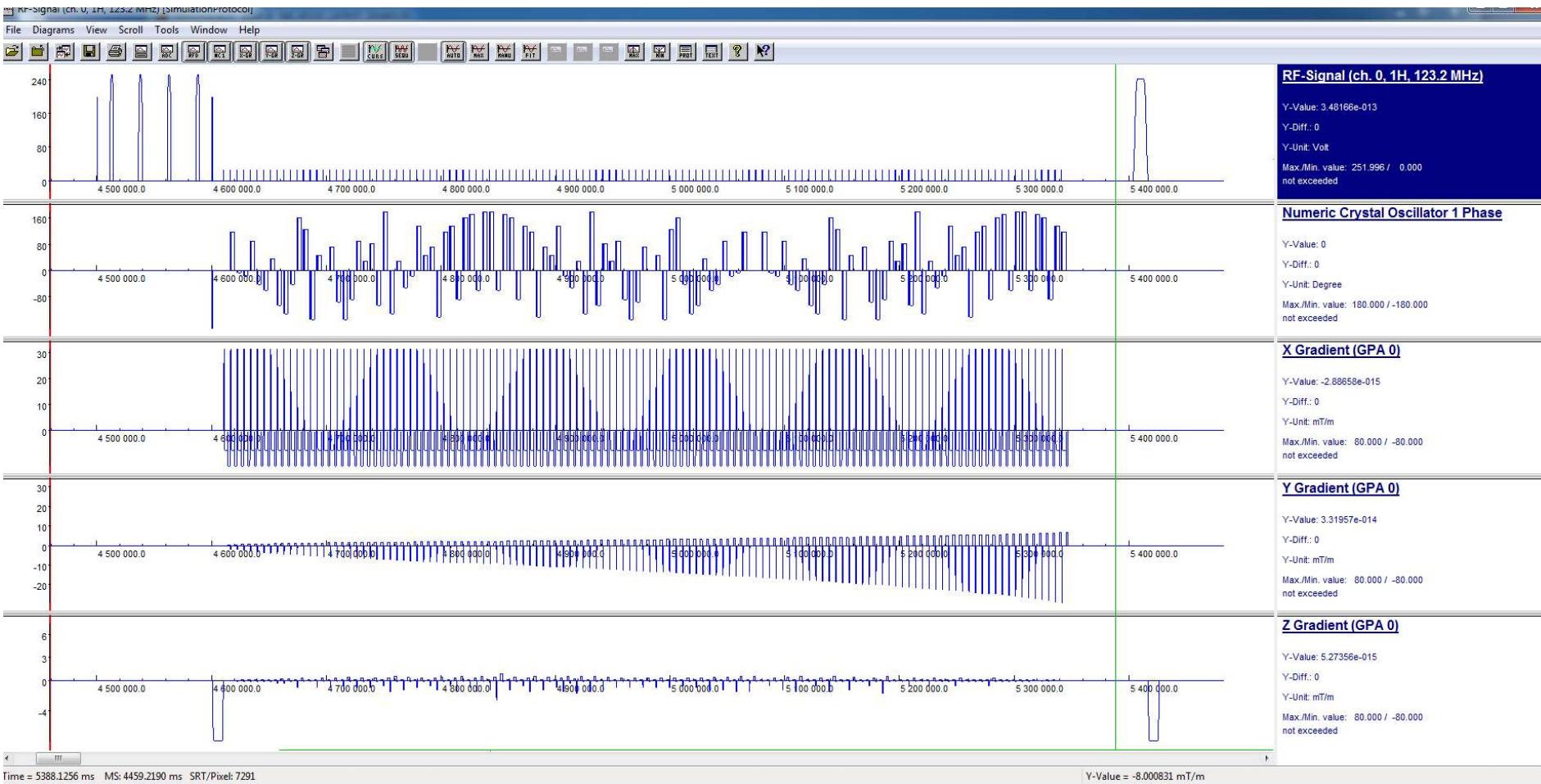
TA: 4:21

Center-out elliptical ordering

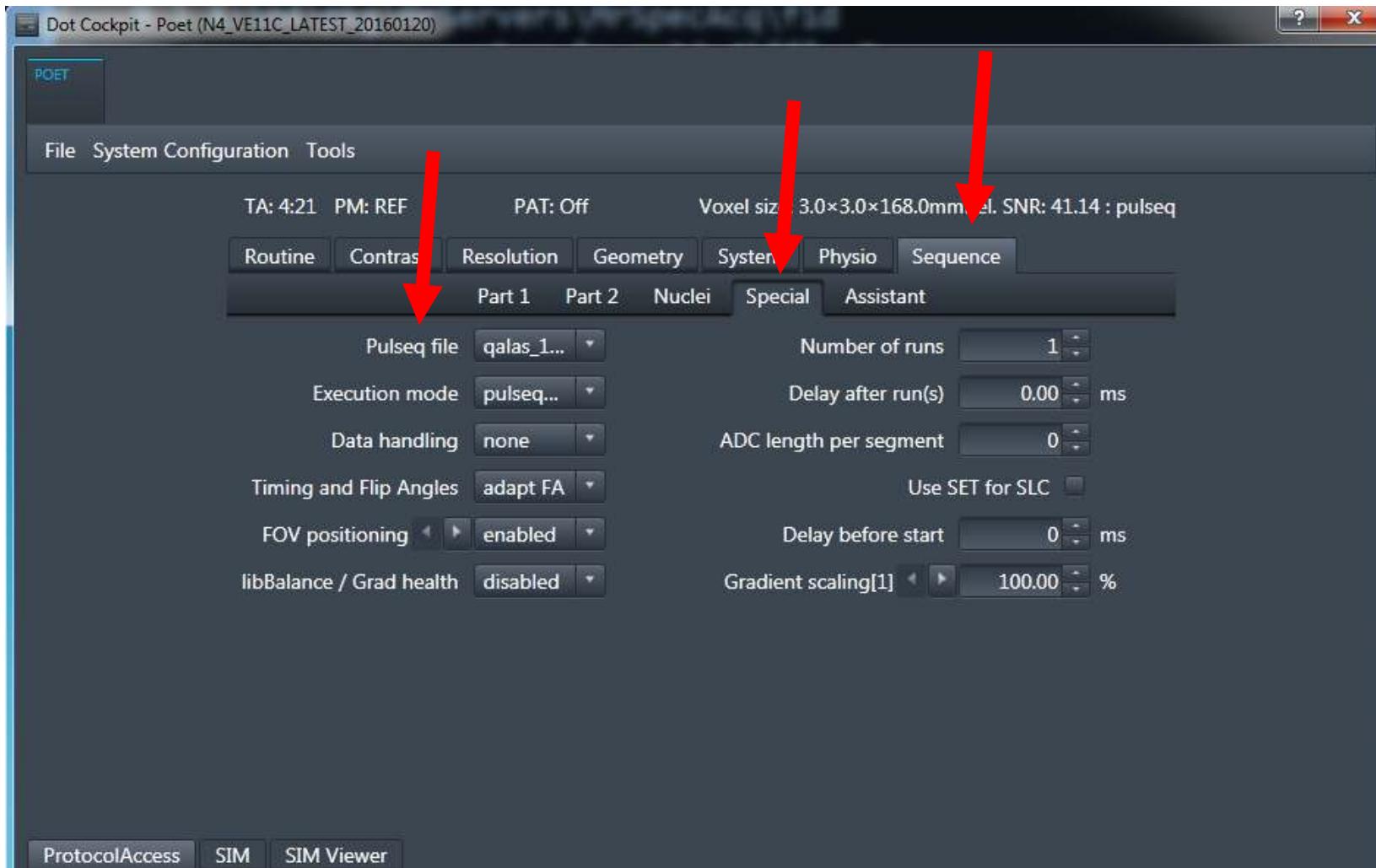
Cartesian sampling



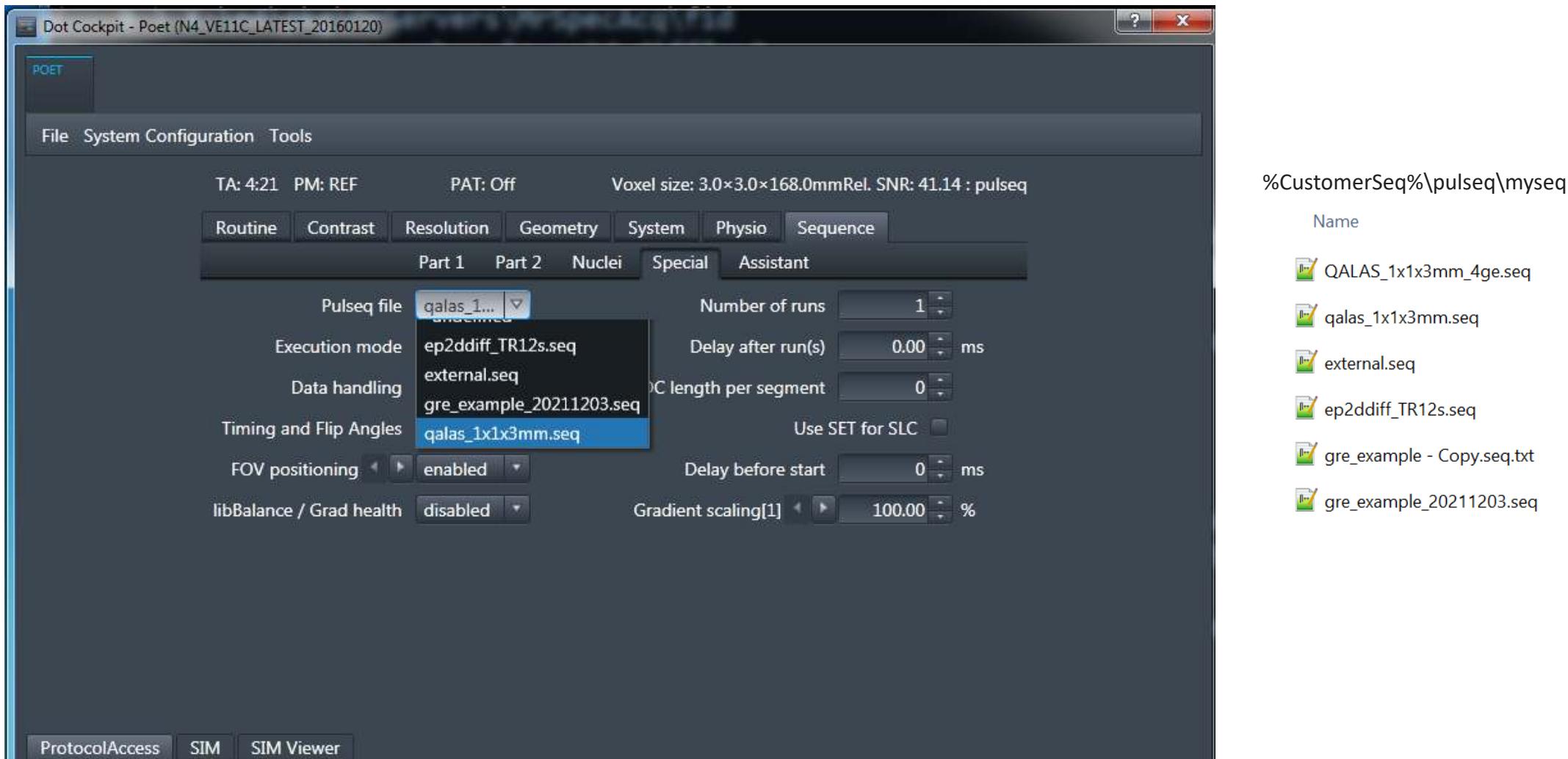
# Before going to the scanner... simulating the .seq file in POET



# Running your sequence on the scanner



# Running your sequence on the scanner



# Exporting and reconstructing your pulseq data

Siemens

Twix



meas\_MID\*.dat

```
data_file_path =  
'/autofs/cluster/berkin/fujita/testscan/archive/meas  
_MID00465_FID38558_pulseq_qalas.dat';
```

```
twix_obj = mapVBVD2(data_file_path);  
data_unsorted = twix_obj{end}.image.unsorted();
```

```
% size(data_unsorted): adc_len, ncoil, readouts
```



Data sorted in the order it was acquired  
Sort using your seq file

GE

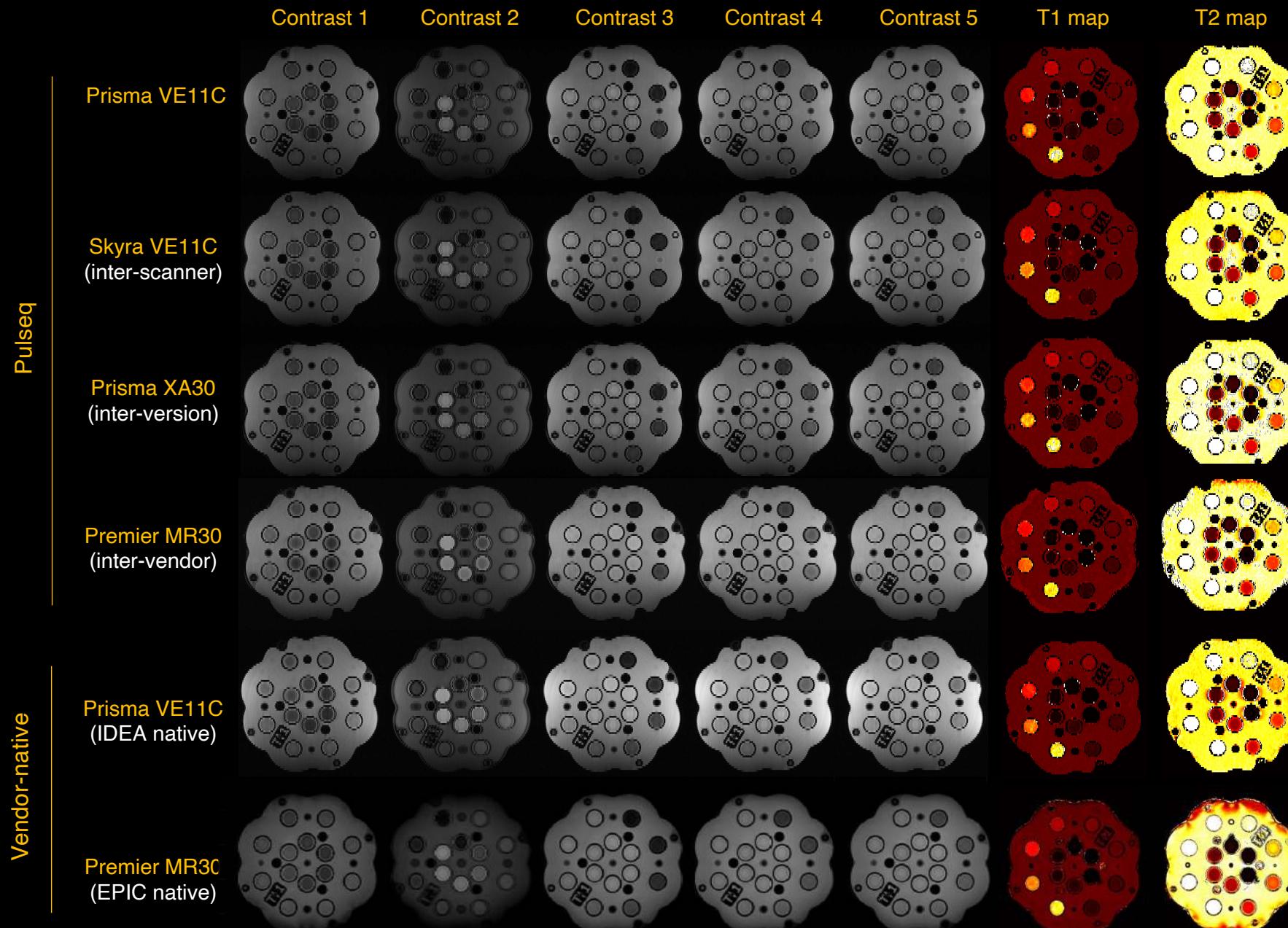
```
pfile = 'P06144.7';
```

```
data_unsorted =  
toppe.utils.loadpfile(pfile, [], [], [], 'acqOrder', true);
```

```
% size(data_unsorted): adc_len, ncoil, readouts
```

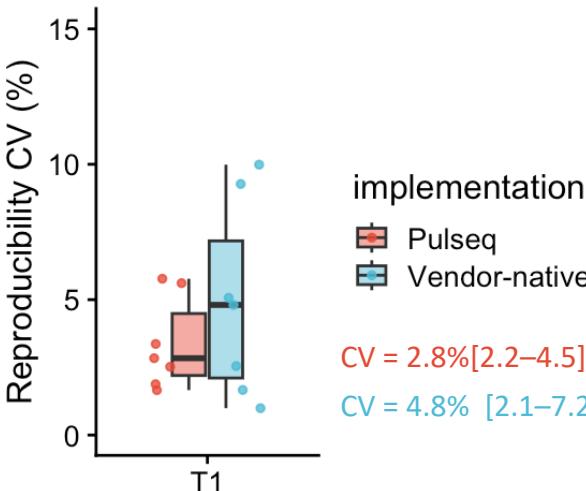


# Cross-platform multiparametric mapping

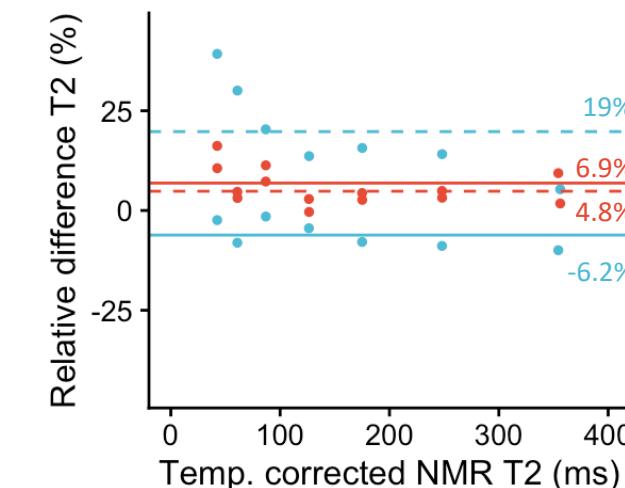
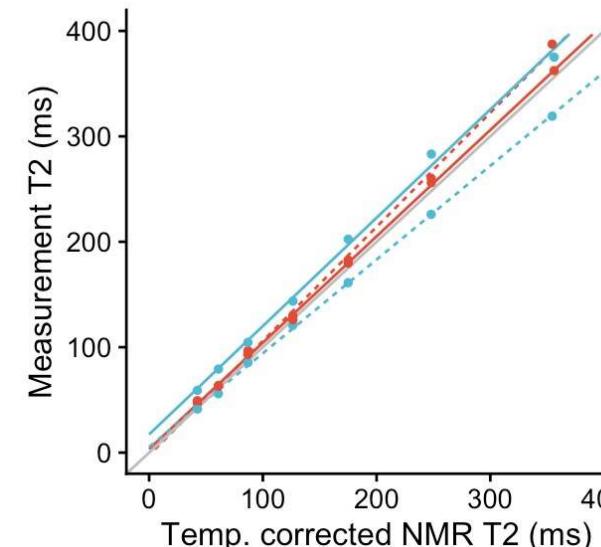
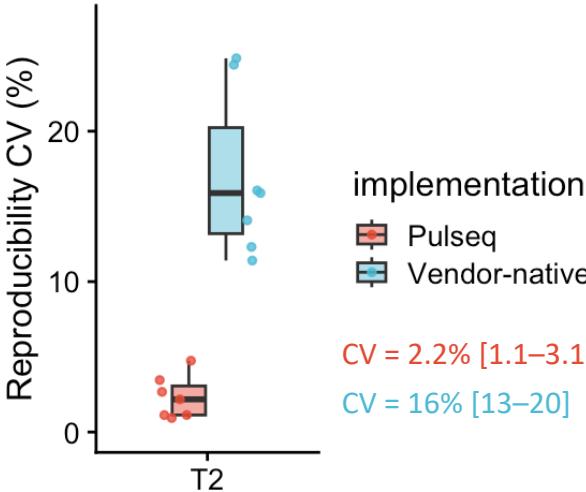
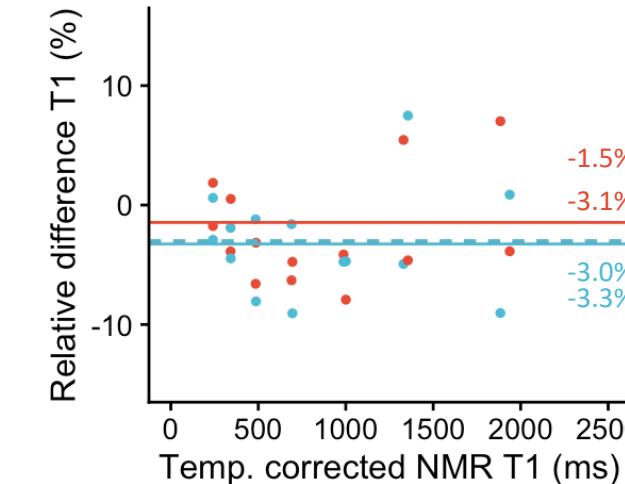
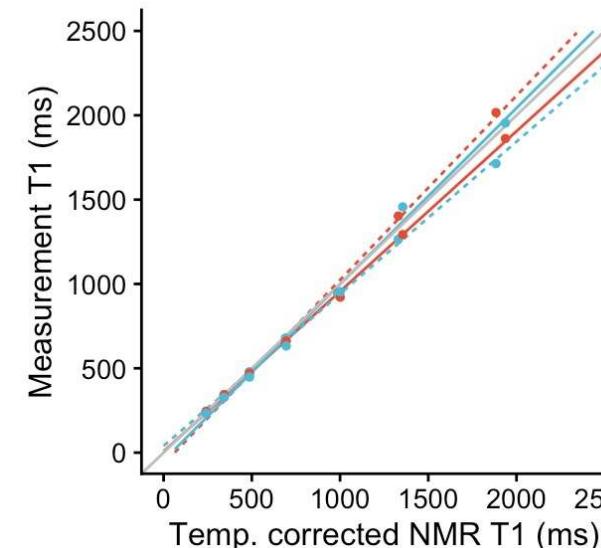


# Identical implementation and reconstruction pipeline improves cross-vendor reproducibility

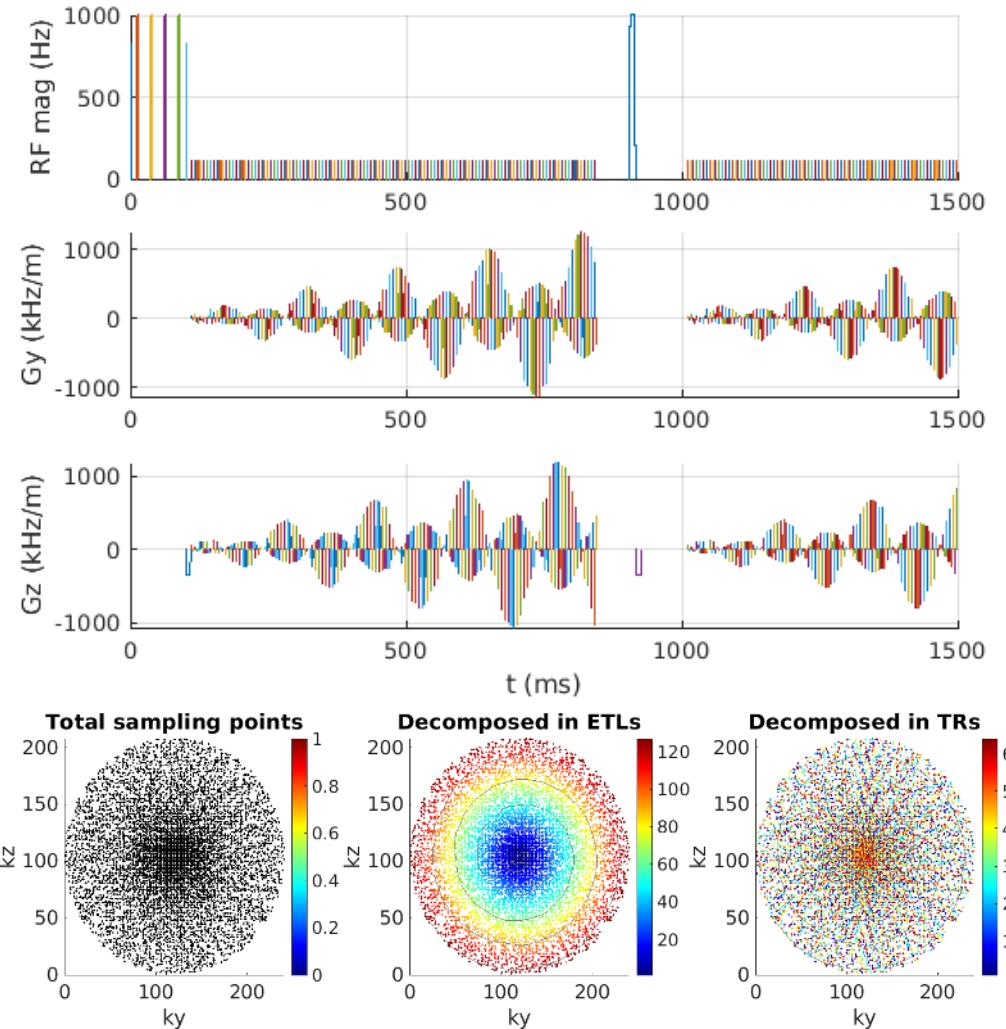
(A) Cross-vendor reproducibility



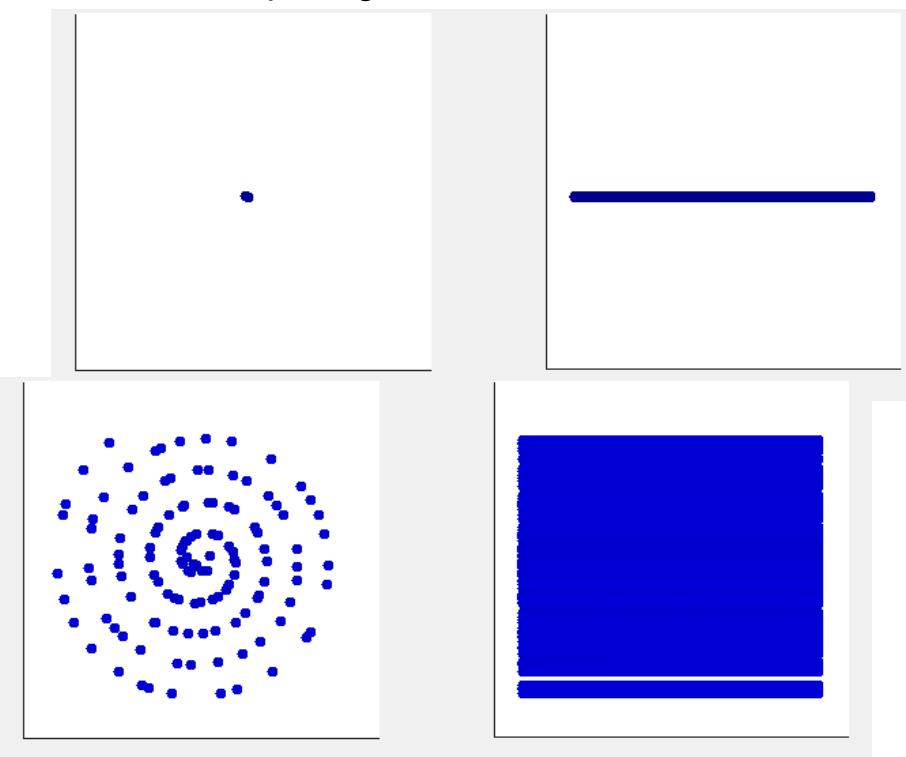
(B) Comparison of Pulseq- and vendor-native implementation



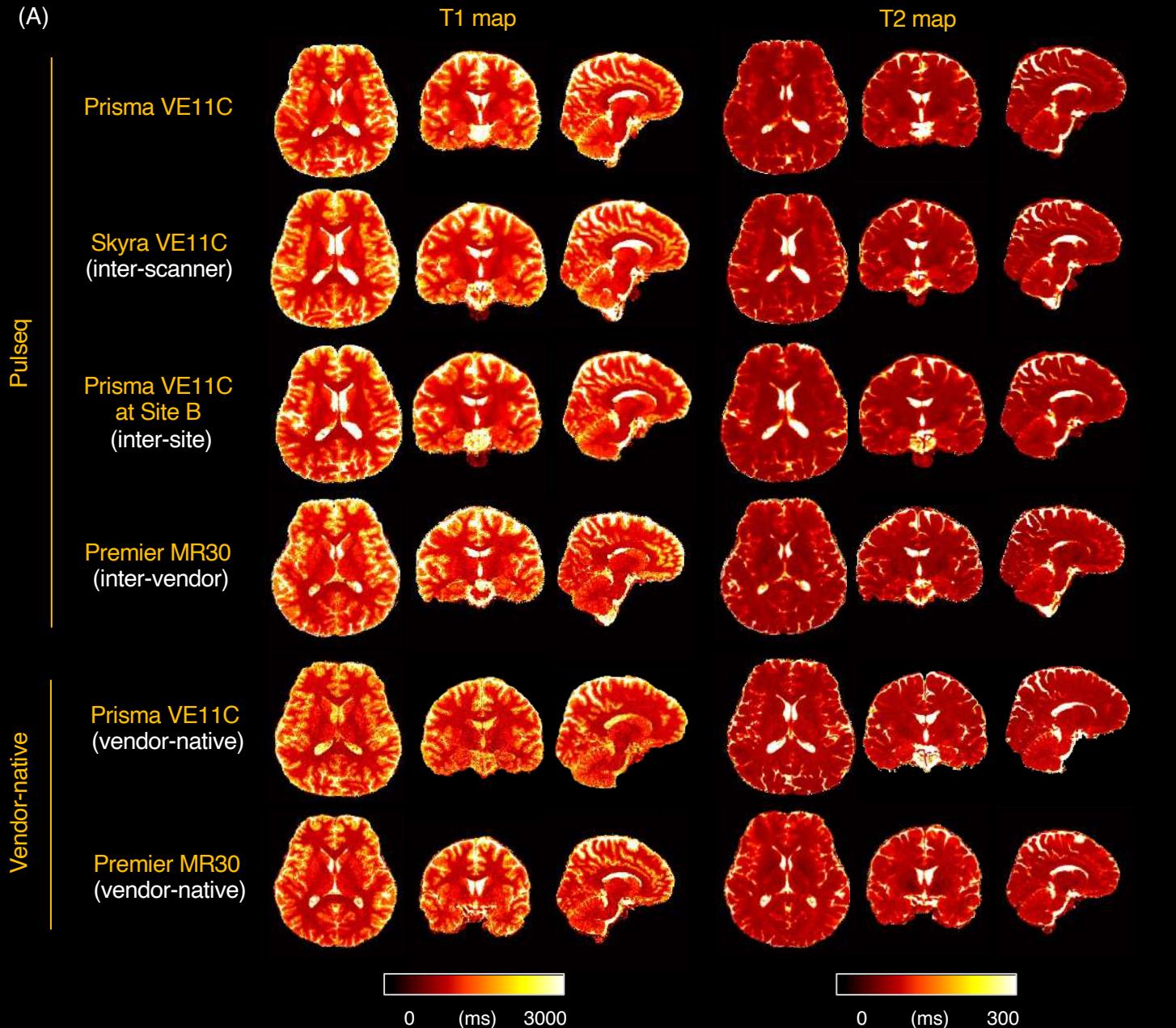
# Implementation for in vivo acquisition



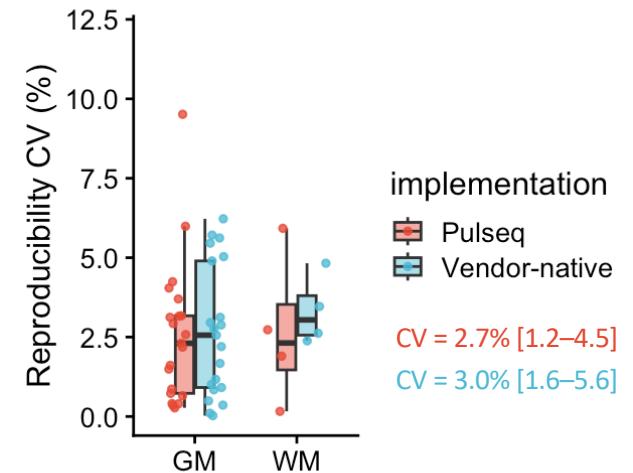
FOV: 256x240x208 (ADNI-compliant)  
Matrix: 256x240x208  
TR: 4500ms  
TE: 2.3ms  
Echo spacing: 5.8ms



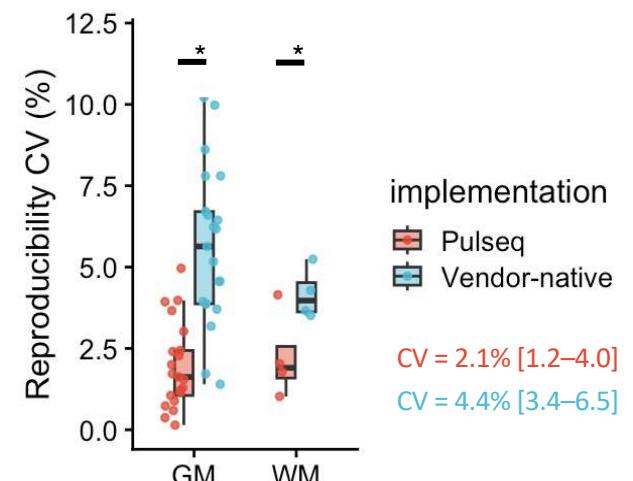
# Vendor-agnostic implementation improves reproducibility



(B) Brain structure T1 value reproducibility



(C) Brain structure T2 value reproducibility



# Summary

- Workflow for using custom sequences implemented in Pulseq was demonstrated
- An application of multiparametric mapping using Pulseq was demonstrated
- Pulseq implementation exhibited:
  - Higher reproducibility than vendor-native implementations
  - ADNI-compliant field-of-view sizes with 1mm isotropic resolution within 5 minutes, while maintaining a cross-platform coefficient of variation below 4%.



Athinoula A.

# Martinos Center

For Biomedical Imaging



MASSACHUSETTS  
GENERAL HOSPITAL



HARVARD  
MEDICAL SCHOOL



Massachusetts  
Institute of  
Technology

# Thank you!

Shohei Fujita

BRAIN (Bilgic Reconstruction Acquisition for Imaging Neuroscience) Lab