# Bayesian analysis of Neuroimaging data with R

Chris Hammill
Jason Lerch

2018-06-14

# Outline

1. Quick overview of algorithms and datasets used today
2. Anatomy and hierarchies
3. Massively univariate classical statistics
4. Meet Reverend Thomas
5. A simple model
6. A more involved model
7. Diagnostics
8. A complex example (if we have time)
9. …
10. Profit?

# Following Along

- To follow along with today's session you will need to either:
- run R with `c("knitr", "tidyverse", "forcats", "ggplot2", "broom", "data.tree", "treemap", "rstanarm", "bayesplot", "pheatmap", "ggrepel")` on your laptop
- use R from our singularity container

# Get these slides and code

- From a terminal run:

```
git clone https://github.com/Mouse-Imaging-Centre/Scinet-SS-bayesian-neuroimaging
cd Scinet-SS-bayesian-neuroimaging-R

## cp in some extra files <use scp if you're working on your own laptop>
cp /bb/scinet/course/ss2018/3_bm/7_mrir/ADNI2_BL_MAGeT_Hippocampus_subfields.csv
cp /bb/scinet/course/ss2018/3_bm/7_mrir/flathierarchy.Rds .
cp /bb/scinet/course/ss2018/3_bm/7_mrir/twolevelhierarchy.Rds .
```

# Singularity set up

- First ssh to niagara and set your port to forward

```
ssh -L8787:localhost:<your-port> <you>@niagara.scinet.utoronto.ca
ssh -L<your-port>:localhost:<your-port> tds01
```

- To use the singularity container you will need to set an environment variable with a temporary RStudio password. You will also need a temporary directory on SCRATCH, this assumes you don't have a directory with this name already

```
export RSTUDIO_PASSWORD="dont_use_this_fake_password"

mkdir $SCRATCH/tmp
module load singularity
```

# Run the container

- Run the container , can be sent to the background

```
module load singularity/2.5.1
singularity exec \
  --bind /bb/scinet/course/ss2018/3_bm/7_mrir/rstudio_auth.sh:/usr/lib/rstudio-se
  --bind $SCRATCH/tmp:/tmp \
  --bind $SCRATCH:$SCRATCH \
  /bb/scinet/course/ss2018/3_bm/7_mrir/RMINC-develop-2018-06-13-6793fefb8a5a.img
  rserver \
  --auth-none 0 \
  --auth-pam-helper-path rstudio_auth.sh \
  --www-port <your-port>
```

- Then navigate to localhost:8787 in your browser, use your scinet username and `$RSTUDIO_PASSWORD$`

# Optional step to compile the slides

- You'll need the package "xaringan" to render the slides
- On singularity this is slightly involved, you'll need a library directory

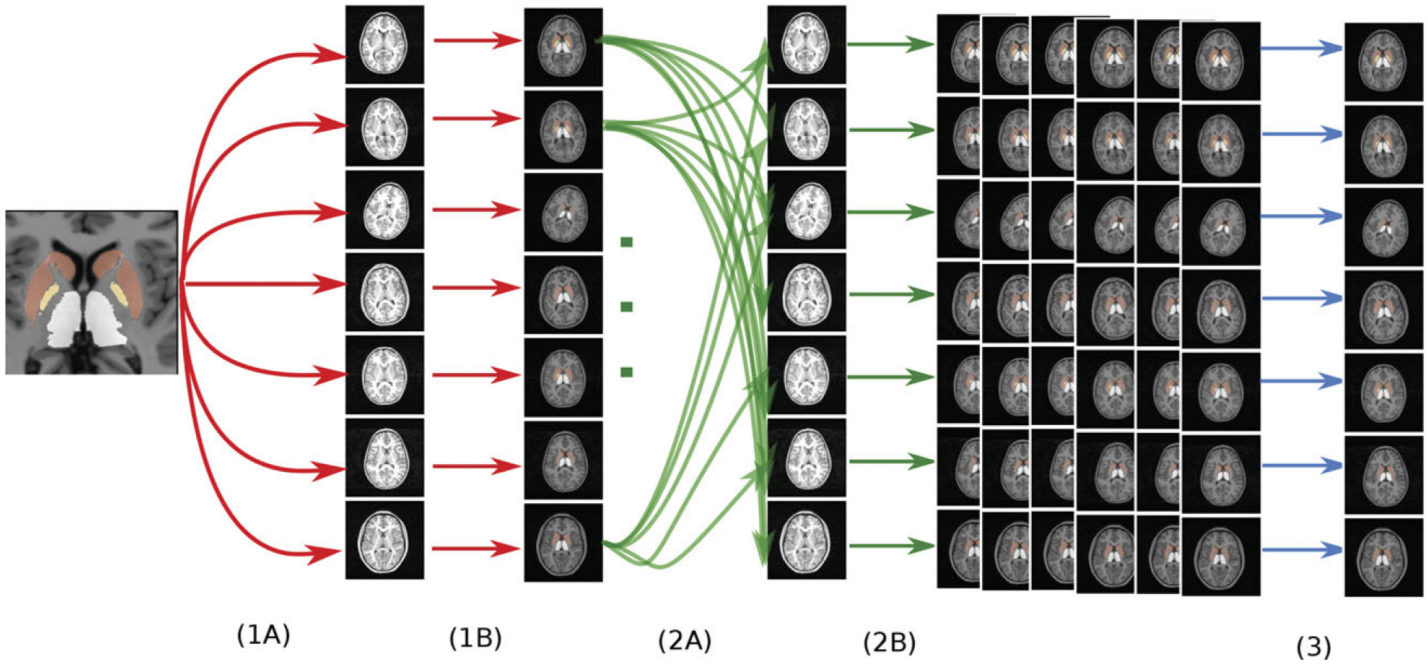
```
dir.create("~/tmp_rlibs")
.libPaths(new = "~/tmp_rlibs")
install.packages("xaringan")
```

# The dataset: ADNI

- baseline scans from ADNI2
- multi-site initiative to study Alzheimer's and Mild Cognitive Impairment

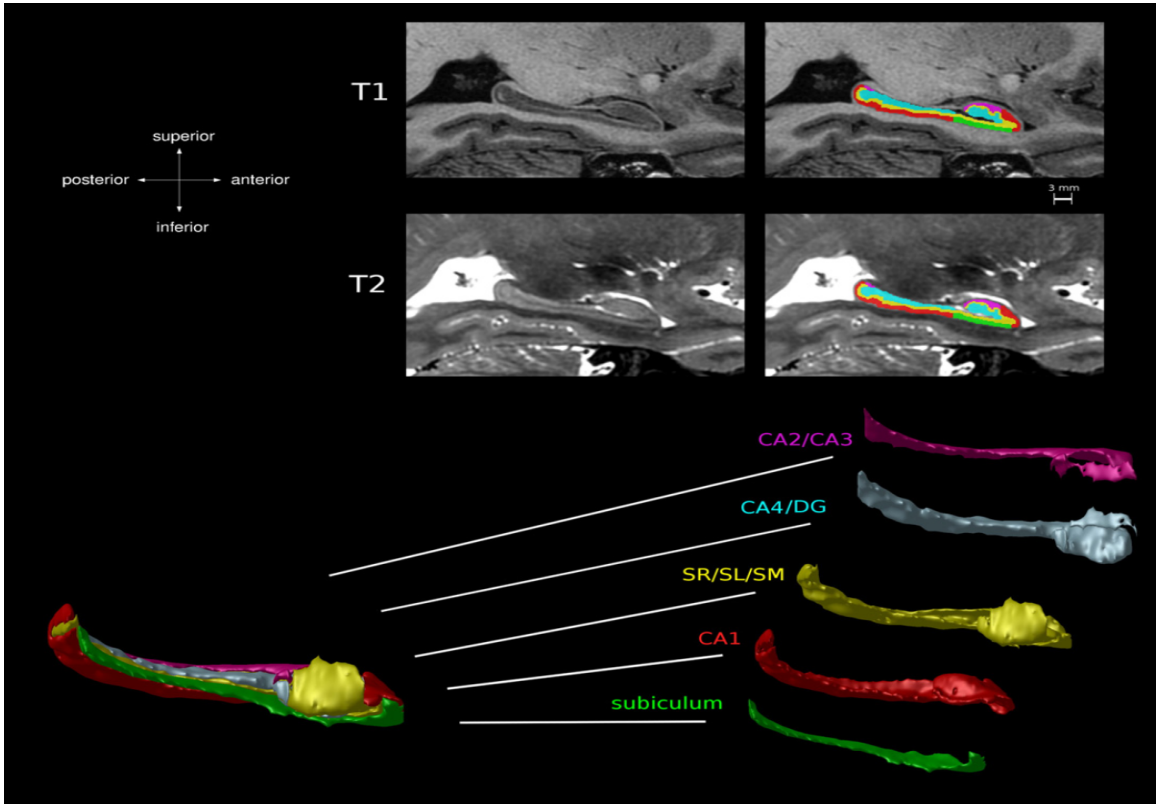
With much gratitude to Nikhil Bhagwat of the CoBrA Lab (PI: Chakravarty) for providing us with processed data.

# MAGeT - the algorithm




Chakravarty MM, Steadman P, van Eede MC, Calcott RD, Gu V, Shaw P, Raznahan A, Collins DL, Lerch JP. Performing label-fusion-based segmentation using multiple automatically generated templates. Hum Brain Mapp. 2013 Oct;34(10):2635–54.

# MAGeT - the atlas




Pipitone J, Park MTM, Winterburn J, Lett TA, Lerch JP, Pruessner JC, Lepage M, Voineskos AN, Chakravarty MM, Alzheimer's Disease Neuroimaging Initiative. Multi-atlas segmentation of the whole hippocampus and subfields using multiple automatically generated templates. Neuroimage. 2014 Nov 1;101:494–512.

Winterburn JL, Pruessner JC, Chavez S, Schira MM, Lobaugh NJ, Voineskos AN, Chakravarty MM. A novel in vivo atlas of human hippocampal subfields using high-resolution 3 T magnetic resonance imaging. Neuroimage. 2013 Jul

# Load the data

Read in the processed files

```
suppressMessages(library(tidyverse))
adni <- read.csv("ADNI2_BL_MAGeT_Hippocampus_subfields.csv")
names(adni)
```

```
##  [1] "X"           "L_CA1"       "L_subiculum" "L_CA4DG"     "L_CA2CA3"
##  [6] "L_stratum"   "L_Alv"       "L_Fimb"      "L_Fornix"    "L_Mam"
## [11] "R_CA1"       "R_subiculum" "R_CA4DG"     "R_CA2CA3"    "R_stratum"
## [16] "R_Alv"       "R_Fimb"      "R_Fornix"    "R_Mam"       "PTID"
## [21] "VISCODE"     "ImageUID"    "ORIGPROT"    "COLPROT"     "AGE"
## [26] "APOE4"       "PTGENDER"    "DX_bl"       "MMSE"        "ADAS11"
## [31] "ADAS13"
```

# Reorganize the diagnosis label

```
library(forcats)
adni <- adni %>%
  mutate(DX_bl=fct_relevel(DX_bl,
                           "CN", "SMC", "EMCI", "LMCI", "AD"))
```

# Data organization

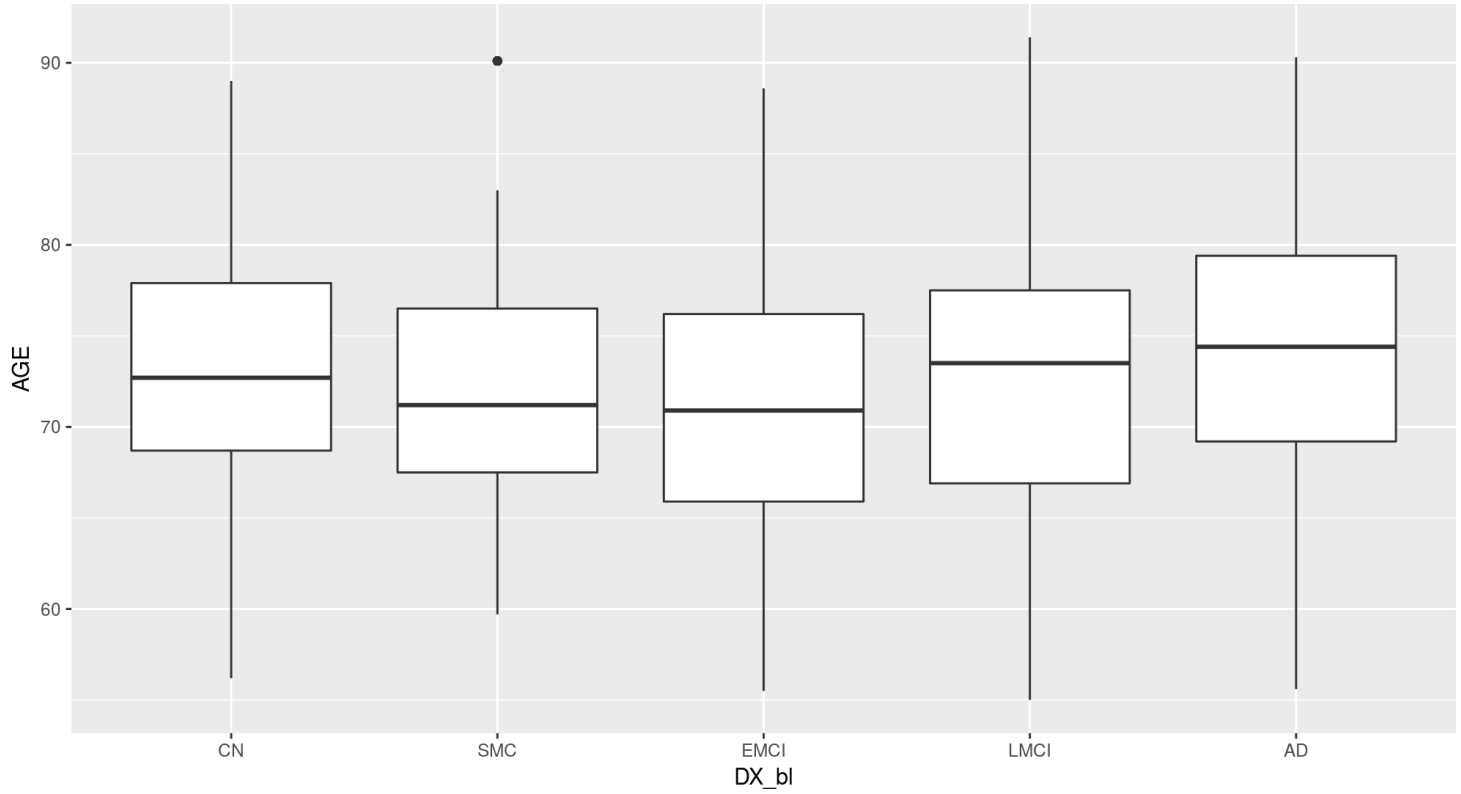
Make the dataframe long rather than wide

```
adniLong <- adni %>%
  select(-X) %>%
    gather(structure, volume, L_CA1:R_Mam)
adniLong %>%
  select(PTID, DX_bl, structure, volume) %>% sample_n(5)
```

```
##            PTID DX_bl   structure    volume
## 8863 073_S_4443  EMCI    R_CA2CA3 188.39999
## 7152 013_S_4395  LMCI R_subiculum 255.60001
## 3466 137_S_4536  EMCI   L_stratum 446.39997
## 6336 002_S_4270    CN       R_CA1 642.00003
## 4577 057_S_5295   SMC      L_Fimb  86.64124
```

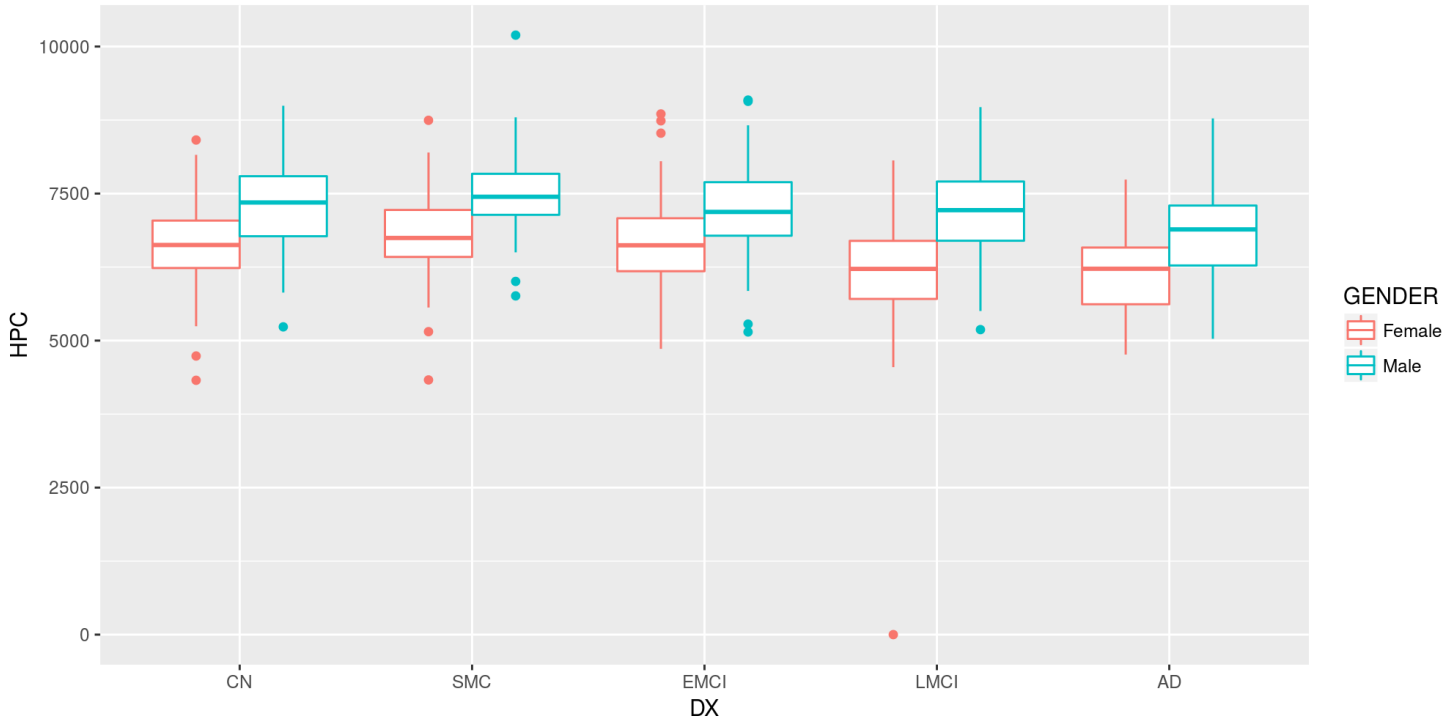
# A quick look at the data

```
library(ggplot2)
ggplot(adniLong) + aes(x=DX_bl, y=AGE) + geom_boxplot()
```



# A quick look at hippocampal volume

```
adniLong %>% group_by(PTID) %>%
  summarize(DX=DX_bl[1], GENDER=PTGENDER[1], HPC=sum(volume)) %>%
  ggplot() + aes(DX, HPC, colour=GENDER) + geom_boxplot()
```



# Classic Statistics

1. Decide on model
2. Apply model to every ROI/voxel separately
3. Widen confidence intervals to account for multiple comparisons

# map for looping over structures

```
library(broom)
adniLong %>%
  split(.$structure) %>%
  map(~lm(volume ~ AGE+PTGENDER+DX_bl, .)) %>%
  map_dfr(tidy, .id='roi') %>% head(14)
```

```
##      roi          term    estimate  std.error  statistic      p.value
## 1  L_Alv  (Intercept) 196.5742731 29.2852653  6.7123952 3.973773e-11
## 2  L_Alv           AGE   2.5174188  0.3956152  6.3633019 3.580089e-10
## 3  L_Alv PTGENDERMale  65.3630633  5.6039202 11.6638106 7.719764e-29
## 4  L_Alv     DX_blSMC  16.4673321  9.2922494  1.7721578 7.680579e-02
## 5  L_Alv    DX_blEMCI  16.8111358  8.1134240  2.0720149 3.863178e-02
## 6  L_Alv    DX_blLMCI  27.8697000  8.3720730  3.3288888 9.178674e-04
## 7  L_Alv      DX_blAD  44.5716230  8.5053623  5.2404144 2.126327e-07
## 8  L_CA1  (Intercept) 706.9671959 41.3804715 17.0845612 6.521867e-55
## 9  L_CA1           AGE   0.2790874  0.5590095  0.4992534 6.177587e-01
## 10 L_CA1 PTGENDERMale  87.9815655  7.9184141 11.1110083 1.619634e-26
## 11 L_CA1     DX_blSMC  36.2846228 13.1300726  2.7634747 5.870110e-03
## 12 L_CA1    DX_blEMCI  -1.3742345 11.4643766 -0.1198700 9.046207e-01
## 13 L_CA1    DX_blLMCI -29.5212053 11.8298511 -2.4954841 1.280902e-02
## 14 L_CA1      DX_blAD -49.0521356 12.0181907 -4.0814909 4.993610e-05
```

# better overview

```
rTable <- adniLong %>%
  split(.$structure) %>%
  map(~lm(volume ~ AGE+PTGENDER+DX_bl, .)) %>%
  map_dfr(tidy, .id='roi') %>%
  filter(startsWith(term, "DX")) %>%
  select(roi, term, statistic) %>%
  spread(term, statistic)
```

# better overview

```
rTable
```

```
##            roi    DX_blAD  DX_blEMCI  DX_blLMCI   DX_blSMC
## 1        L_Alv  5.2404144  2.0720149  3.3288888  1.7721578
## 2        L_CA1 -4.0814909 -0.1198700 -2.4954841  2.7634747
## 3     L_CA2CA3 -1.1964159 -0.2493317 -0.9781210  1.2829814
## 4      L_CA4DG -7.0736847 -1.0666621 -4.5515264  1.8258561
## 5       L_Fimb -5.8276864 -1.3677552 -5.1790266  1.0673108
## 6     L_Fornix -0.1788469  1.1224113 -1.1998364  1.6453122
## 7        L_Mam -3.0547427  0.4167804 -0.9110615  1.0460497
## 8    L_stratum -7.5668406 -0.8851018 -4.2477610  2.0285441
## 9  L_subiculum -8.4089152 -1.3472510 -4.6988241  1.4397638
## 10       R_Alv  2.2203457  1.3354607  1.9552464  1.0756636
## 11       R_CA1 -2.3190902  0.2005679 -1.8465551  2.0229622
## 12    R_CA2CA3 -5.0678893 -1.2333109 -2.5235391  0.1698605
## 13     R_CA4DG -4.2709485 -0.4105204 -3.5381430  0.7337568
## 14      R_Fimb -6.0683126 -1.5310185 -4.3249336 -0.1997899
## 15    R_Fornix -1.1565583  0.3071713 -2.3384244  1.1363678
## 16       R_Mam -1.9359947  1.2241687 -0.8748597  1.2207012
## 17   R_stratum -8.1615304 -1.7229391 -4.4048893  0.9723020
## 18 R_subiculum -7.7494464 -1.0529968 -4.5346801  1.7301202
```

# better overview

```
DT::datatable(rTable %>% remove_rownames() %>%
  column_to_rownames("roi") %>% round(2), options=list(pageLength=7))
```

# Frequentist Null Hypothesis Testing

- To form conclusions in frequentism we typically lean on null hypothesis testing.
- Null hypotheses are parameter values for your model you'd like to disprove
- If your statistics (and more extreme statistics) would be very unlikely given your null model you reject the null hypothesis, and conclude that the null hypothesis is not correct.
- Choosing a threshold for this probability (e.g. 0.05) and rejecting when your p-value is below the threshold gives you a fixed probability of making a "Type I" error, which conveniently is equal to your threshold.
- So if we reject all p-values when they are below 0.05 we have a 5% chance of rejecting when the null model is in fact true.
- If this is confusing, you're not alone, this is very hard to wrap your mind around.

# Zoom in on a single structure

```
adniLong %>%
  split(.$structure) %>%
  map(~lm(volume ~ AGE+PTGENDER+DX_bl, .)) %>%
  first %>%
  summary
```

```
##
## Call:
## lm(formula = volume ~ AGE + PTGENDER + DX_bl, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -398.90  -49.05   -7.69   45.79  280.80
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  196.5743    29.2853   6.712 3.97e-11 ***
## AGE            2.5174     0.3956   6.363 3.58e-10 ***
## PTGENDERMale  65.3631     5.6039  11.664  < 2e-16 ***
## DX_blSMC      16.4673     9.2922   1.772 0.076806 .
## DX_blEMCI     16.8111     8.1134   2.072 0.038632 *
## DX_blLMCI     27.8697     8.3721   3.329 0.000918 ***
## DX_blAD       44.5716     8.5054   5.240 2.13e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 73.1 on 696 degrees of freedom
```

# Interpreting these results

- After fitting our models we're left with:

1. coefficient estimates
2. t-statistics
3. p-values

- We know if our model assumptions are satisfied our t-statistics have a known distribution.
- From this distribution we can figure out the probability of t-statistics as large or larger than the one we observed (p-value)

# Dealing with Many Tests

- If you're testing a lot of hypotheses, a 5% chance of making a mistake adds up
- After 14 tests you have a better than a 50/50 chance of having made at least one mistake
- How do we control for this?
- Two main approaches Family-Wise Error Rate (FWER) control and False-Discovery Rate (FDR) control.

# FWER

- In family-wise error rate control, we try to limit the chance we will at least one type I error.
- Quite conservative, so in neuroimaging we tend to use False Discovery Rate control.

# FDR

- Instead of trying to control our chances of making at least one mistake, let's try to control the fraction of mistakes we make.
- To do this we employ the Benjamini-Hochberg procedure.
- The Benjamini-Hochberg procedure turns our p-values in q-values. Rejecting all q-values below some threshold controls the expected number of mistakes.
- For example if we reject all hypotheses with $q < 0.05$, we expect about 5% of our results to be false discoveries (type I errors).
- If we have 100's or more tests we can accept a few mistakes in the interest of finding the important results.

# multiple comparisons - omnibus FDR

```
pTable <- adniLong %>%
  split(.$structure) %>%
  map(~lm(volume ~ AGE+PTGENDER+DX_bl, .)) %>%
  map_dfr(tidy, .id='roi') %>%
  filter(startsWith(term, "DX")) %>%
  select(roi, term, p.value) %>%
  spread(term, p.value) %>%
  remove_rownames() %>%
  column_to_rownames("roi") %>%
  as.matrix()
qTable <- pTable
qTable[,] <- p.adjust(pTable, 'fdr')
```

# multiple comparisons - omnibus FDR

```
qTable
```

```
##                  DX_blAD  DX_blEMCI    DX_blLMCI   DX_blSMC
## L_Alv       1.913694e-06 0.09933886 3.304323e-03 0.15800049
## L_CA1       1.997444e-04 0.90462074 3.842707e-02 0.01921127
## L_CA2CA3    3.408123e-01 0.86311637 4.042228e-01 0.32715109
## L_CA4DG     5.305411e-11 0.37376998 3.761830e-05 0.14463584
## L_Fimb      8.840563e-08 0.30175056 2.338860e-06 0.37376998
## L_Fornix    8.773547e-01 0.36287262 3.408123e-01 0.19015000
## L_Mam       8.017881e-03 0.75494833 4.350941e-01 0.37376998
## L_stratum   2.186873e-12 0.44355718 1.038839e-04 0.10430127
## L_subiculum 1.682195e-14 0.30501574 2.064548e-05 0.27069092
## R_Alv       7.124433e-02 0.30501574 1.183444e-01 0.37376998
## R_CA1       5.726437e-02 0.87735472 1.423331e-01 0.10430127
## R_CA2CA3    3.715510e-06 0.34081228 3.706287e-02 0.87735472
## R_CA4DG     9.976500e-05 0.75494833 1.628422e-03 0.52953604
## R_Fimb      2.547278e-08 0.23302004 8.394683e-05 0.87735472
## R_Fornix    3.569035e-01 0.82778710 5.658467e-02 0.36168535
## R_Mam       1.198641e-01 0.34081228 4.435572e-01 0.34081228
## R_stratum   5.578523e-14 0.16607426 6.300383e-05 0.40422277
## R_subiculum 7.859835e-13 0.37376998 3.761830e-05 0.16607426
```

# Hippocampal anatomical hierarchy

Setting up a simple hierarchy, dividing the hippocampus in grey matter and tracts.

1. Hippocampal Formation
    1. Grey Matter
        1. CA1
        2. CA2/CA3
        3. CA4/DG
        4. subiculum
        5. stratum
        6. Mammilary bodies
    2. White Matter
        1. Alveus
        2. Fimbria
        3. Fornix

# Hippocampal anatomical hierarchy

```
library(data.tree)
hpc <- Node$new("HPC")
gm <- hpc$AddChild("GM")
ca1 <- gm$AddChild("CA1")
lca1 <- ca1$AddChild("left CA1")
lca1$volumes <- adni$L_CA1
rca1 <- ca1$AddChild("right CA1")
rca1$volumes <- adni$R_CA1
ca23 <- gm$AddChild("CA2CA3")
lca23 <- ca23$AddChild("left CA2CA3")
lca23$volumes <- adni$L_CA2CA3
rca23 <- ca23$AddChild("right CA2CA3")
rca23$volumes <- adni$R_CA2CA3
ca4 <- gm$AddChild("CA4DG")
lca4 <- ca4$AddChild("left CA4DG")
lca4$volumes <- adni$L_CA4DG
rca4 <- ca4$AddChild("right CA4DG")
rca4$volumes <- adni$R_CA4DG
subiculum <- gm$AddChild("subiculum")
lsubiculum <- subiculum$AddChild("left subiculum")
lsubiculum$volumes <- adni$L_subiculum
rsubiculum <- subiculum$AddChild("right subiculum")
rsubiculum$volumes <- adni$R_subiculum
stratum <- gm$AddChild("stratum")
lstratum <- stratum$AddChild("left stratum")
lstratum$volumes <- adni$L_stratum
rstratum <- stratum$AddChild("right stratum")
```

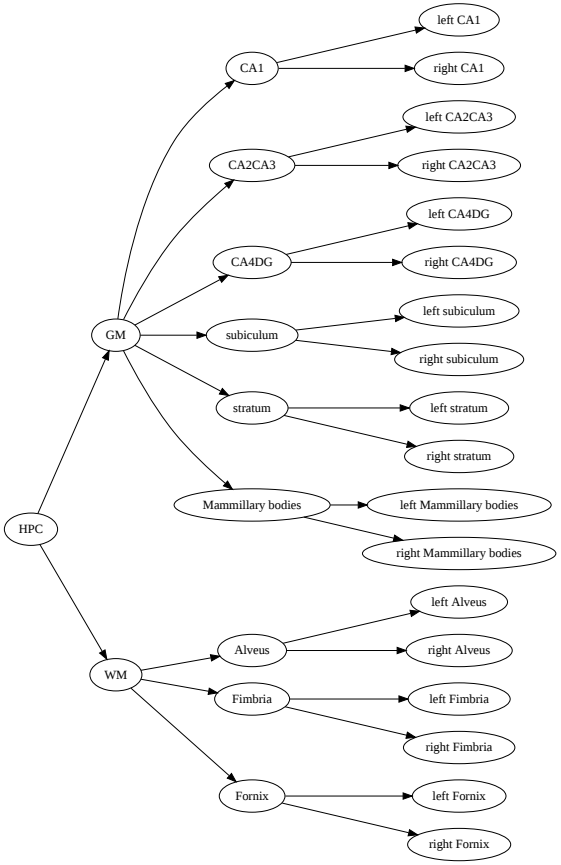
# Hippocampal anatomical hierarchy

```
hpc
```

```
##                                levelName
## 1  HPC
## 2   ¦--GM
## 3   ¦    ¦--CA1
## 4   ¦    ¦   ¦--left CA1
## 5   ¦    ¦   °--right CA1
## 6   ¦    ¦--CA2CA3
## 7   ¦    ¦   ¦--left CA2CA3
## 8   ¦    ¦   °--right CA2CA3
## 9   ¦    ¦--CA4DG
## 10  ¦    ¦   ¦--left CA4DG
## 11  ¦    ¦   °--right CA4DG
## 12  ¦    ¦--subiculum
## 13  ¦    ¦   ¦--left subiculum
## 14  ¦    ¦   °--right subiculum
## 15  ¦    ¦--stratum
## 16  ¦    ¦   ¦--left stratum
## 17  ¦    ¦   °--right stratum
## 18  ¦    °--Mammillary bodies
## 19  ¦        ¦--left Mammillary bodies
## 20  ¦        °--right Mammillary bodies
## 21  °--WM
## 22       ¦--Alveus
## 23       ¦   ¦--left Alveus
## 24       ¦   °--right Alveus
```

# Hippocampal anatomical hierarchy

```
SetGraphStyle(hpc, rankdir="LR")
plot(hpc)
```

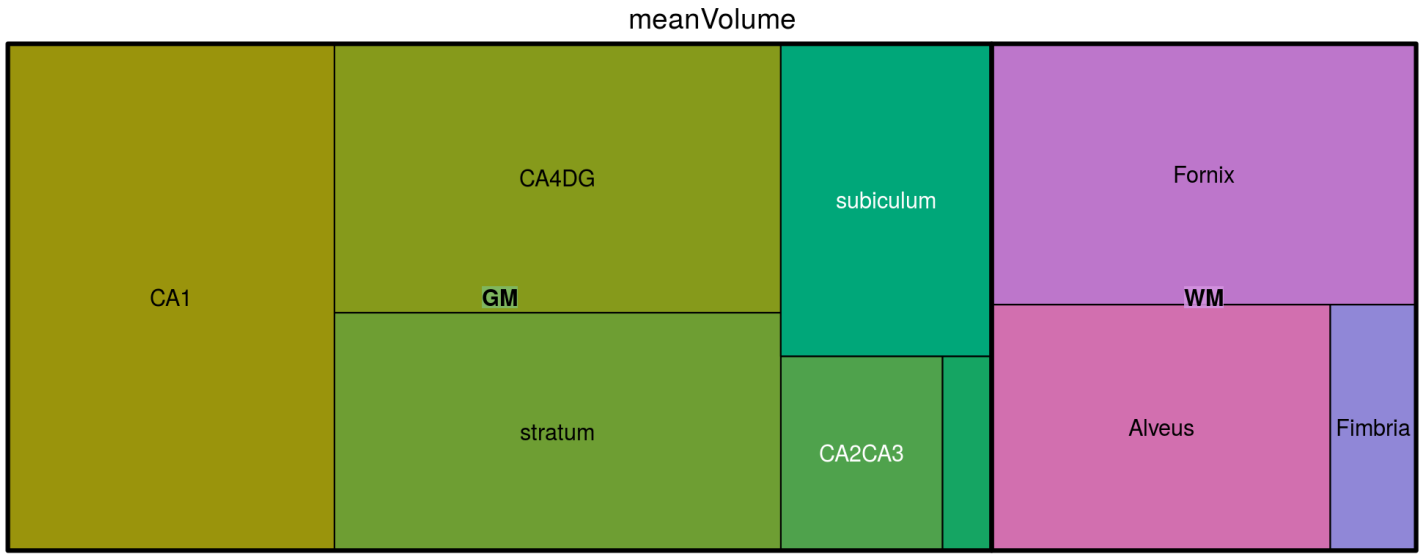


# Aggregate up the tree

```
hpc$Do(function(x){
  x$volumes <- Aggregate(x, "volumes", rowSums)
}, traversal="post-order", filterFun=isNotLeaf)

hpc$Do(function(x) {
  x$meanVolume <- mean(x$volumes)
})
```

# Tree graph

```
library(treemap)
ToDataFrameTable(hpc, "pathString", "meanVolume", "name") %>%
  mutate(path=strsplit(pathString, "/"),
         struct=map_chr(path, ~ .x[3]),
         tissue=map_chr(path, ~ .x[2])) %>%
  select(struct, tissue, name, meanVolume) %>%
  treemap(index=c("tissue", "struct"), vSize="meanVolume")
```



# Statistics on the tree

```
hpc$Do(function(x){
  adni$volumes <- x$volumes
  x$stats <-
    lm(volumes ~ AGE+PTGENDER+DX_bl, adni) %>%
    tidy() %>%
    filter(startsWith(term, "DX")) %>%
    select(term, estimate, statistic, p.value)
})
```

# Statistics on the tree

```
print(hpc, AD=function(x)
  x$stats %>% filter(term=="DX_blAD") %>% select(statistic) )
```

```
##                                levelName         AD
## 1  HPC                                    -4.5728017
## 2   ¦--GM                                 -6.4256670
## 3   ¦   ¦--CA1                            -3.2923199
## 4   ¦   ¦   ¦--left CA1                   -4.0814909
## 5   ¦   ¦   °--right CA1                  -2.3190902
## 6   ¦   ¦--CA2CA3                         -3.5392995
## 7   ¦   ¦   ¦--left CA2CA3                -1.1964159
## 8   ¦   ¦   °--right CA2CA3               -5.0678893
## 9   ¦   ¦--CA4DG                          -5.9874539
## 10  ¦   ¦   ¦--left CA4DG                 -7.0736847
## 11  ¦   ¦   °--right CA4DG                -4.2709485
## 12  ¦   ¦--subiculum                      -8.6168787
## 13  ¦   ¦   ¦--left subiculum             -8.4089152
## 14  ¦   ¦   °--right subiculum            -7.7494464
## 15  ¦   ¦--stratum                        -8.2297918
## 16  ¦   ¦   ¦--left stratum               -7.5668406
## 17  ¦   ¦   °--right stratum              -8.1615304
## 18  ¦   °--Mammillary bodies              -2.6843408
## 19  ¦       ¦--left Mammillary bodies     -3.0547427
## 20  ¦       °--right Mammillary bodies    -1.9359947
## 21  °--WM                                  0.7325428
## 22      ¦--Alveus                          4.1207504
```

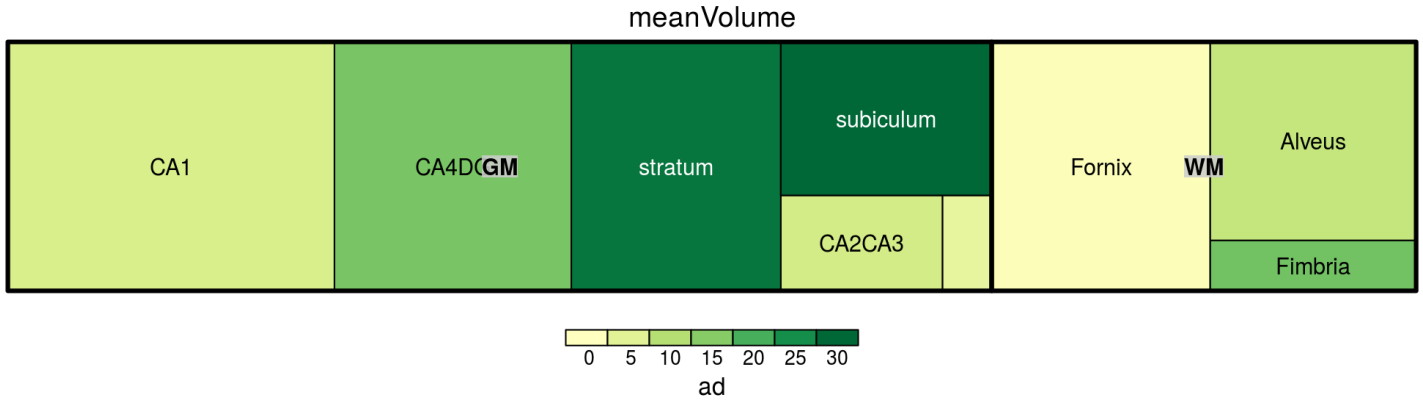
# Statistics on the tree

```
print(hpc, AD=function(x)
  x$stats %>% filter(term=="DX_blAD") %>%
    select(statistic),
  filterFun=isNotLeaf )
```

```
##                     levelName          AD
## 1  HPC                         -4.5728017
## 2    ¦--GM                     -6.4256670
## 3    ¦   ¦--CA1                -3.2923199
## 4    ¦   ¦--CA2CA3             -3.5392995
## 5    ¦   ¦--CA4DG              -5.9874539
## 6    ¦   ¦--subiculum          -8.6168787
## 7    ¦   ¦--stratum            -8.2297918
## 8    ¦   °--Mammillary bodies -2.6843408
## 9    °--WM                      0.7325428
## 10       ¦--Alveus              4.1207504
## 11       ¦--Fimbria            -6.5869583
## 12       °--Fornix             -0.6995234
```

# Statistics on the tree

```
hpc$Do(function(x){
  x$ad <- -log10(x$stats %>% filter(term=="DX_blAD") %>%
                   select(p.value))
})

ToDataFrameTable(hpc, "pathString", "meanVolume", "name", "ad") %>%
  mutate(path=strsplit(pathString, "/"),
         struct=map_chr(path, ~ .x[3]),
         tissue=map_chr(path, ~ .x[2])) %>%
  select(struct, tissue, name, meanVolume, ad) %>%
  treemap(index=c("tissue", "struct"), vSize="meanVolume",
    vColor="ad", type="value")
```



# And now for Bayesianism!

# Why Bayesian Statistics?

Have you ever...

1. Been confused about what a p-value means?
2. Been frustrated that a difference in significance doesn't mean a significant difference?
3. Known some values for a parameter are impossible but been unable to use that to your advantage?
4. Wanted to ask more interesting questions than whether or not a parameter is or isn't zero?
5. Wanted to use information from the literature to improve your estimates?

# Why Bayesian Statistics?

Have you ever...

1. Been confused about what a p-value means?
2. Been frustrated that a difference in significance doesn't mean a significant difference?
3. Known some values for a parameter are impossible but been unable to use that to your advantage?
4. Wanted to ask more interesting questions than whether or not a parameter is or isn't zero?
5. Wanted to use information from the literature to improve your estimates?

Then Bayesian statistics might be right for you!

# How you ask?

1. **De-emphasize binary descisions.** Bayesians avoid null hypothesis tests, instead focusing on estimating their parameters of interest, and reporting their uncertainty.
2. **Posterior Distributions** Bayesian analyses produce a distribution of possible parameter values (the posterior), that can be used to ask many interesting questions about values. E.g. what is the probability the effect in the hippocampus is larger than the effect in the anterior cingulate cortex.
3. **Prior Information** Bayesian analyses can use prior information. Bayesian analysis requires an *a priori* assessment of how likely certain parameters are. This can be vague (uninformative) or can precise (informative) and steer your analysis away from nonsensical results.

# Meet The Reverend

Reverend Thomas Bayes

# Bayes' Theorem

- Bayes noticed this useful property for the probabilities for two events "A" and "B"

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

- $P(A|B)$: The probability of A given that B happened
- $P(B|A)$: The probability of B given that A happened
- $P(A)$: The probability of A

- $P(B)$: the probability of B

- Bayes did this in the context of the binomial distribution

But who's that behind him!

# It's Pierre-Simon Laplace

# Bayesian Statistics

- Laplace generalized Bayes Theorem into it's modern form. While working on sex-ratios in French births.
- For light reading on the history of bayesianism consider reading the theory that would not die

# Bayes in brief

- Start with some parameters $\theta$
- Collect some data $D$
- And deduce the probability of different values of $\theta$ given that you observed $D$
- Key difference between Bayesianism and Frequentism is that view that $\theta$ has an associated probability distribution. In frequentism $\theta$ is an unknown constant.

# Different Probabilities

- Frequentists believe that probabilities represent the long-run proportion of events
- Under this model $P(\theta)$ doesn't make much sense.
- Ramsey and DeFinetti showed that probability can also represent degree of belief.
- Under this model $P(\theta)$ is an assesment of what you think the the parameter will be.
- For some of the philosophy underpinning bayesian reasoning consider reading Bayesian philosophy of science

# Bayes' Theorem Redux

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{\int P(D|\theta)P(\theta)d\theta}$$

**Posterior**: $P(\theta|D)$:

the probability of our parameters given our data

**Likelihood**: $P(D|\theta)$

The probability of our data given our parameters

**Prior**: $P(\theta)$

The probability of our parameters before we saw the data

**Normalizing Constant**: $\int P(D|\theta)P(\theta)d\theta$

The probability of the data averaged over all possible parameter sets

# Bayes' Theorem Redux

$$P(\theta|D) \propto P(D|\theta)P(\theta)$$

**Posterior**: $P(\theta|D)$:

the probability of our parameters given our data

**Likelihood**: $P(D|\theta)$

The probability of our data given our parameters

**Prior**: $P(\theta)$

The probability of our parameters before we saw the data

# Bayes' Theorem Redux

$$P(\theta|D) \propto P(\theta)P(D|\theta)$$

**Posterior**: $P(\theta|D)$:

the probability of our parameters given our data

**Prior**: $P(\theta)$

The probability of our parameters before we saw the data

**Likelihood**: $P(D|\theta)$

The probability of our data given our parameters

**Pardon the re-ordering**

# Posterior

$P(\theta|D)$

- The goal of bayesian statistics
- The posterior is probability distribution over parameters.
- Depends on the data we observed.
- Can be used to answer interesting questions. For example how likely is an effect between two biologically meaninful boundaries.

# Prior

$P(\theta)$

- This is what we knew before the experiment.
- The prior is also a probability distribution over parameters.
- Doesn't depend on the data we saw.
- Gives a probability for any value the parameters could take.

# Likelihood

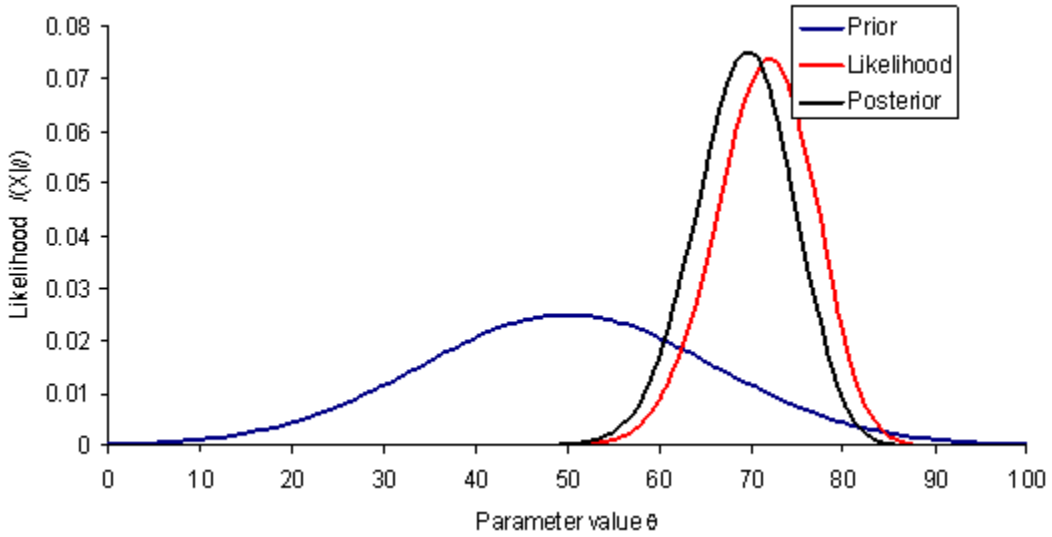
$P(D|\theta)$

- This is how probable our data is given a hypothetical parameter set
- The likelihood is a probability distribution over data (not parameters)
- Is still a function of parameters.

# In words

$$P(\theta|D) \propto P(\theta)P(D|\theta)$$

The probability of parameters given our data is proportional to how probable we thought they were before adjusted by how well they agree with the data we saw.



# A first example

- Let's revisit linear modelling but this time from a bayesian stand-point.

$$\mathrm{y} = X\beta + \epsilon$$

We'll make our probabilistic views explicit

$$\epsilon \sim \mathbb{N}(0, \sigma)$$

$\epsilon$ is normally distributed with some unknown variance $\sigma$

# Frequentist interpretation

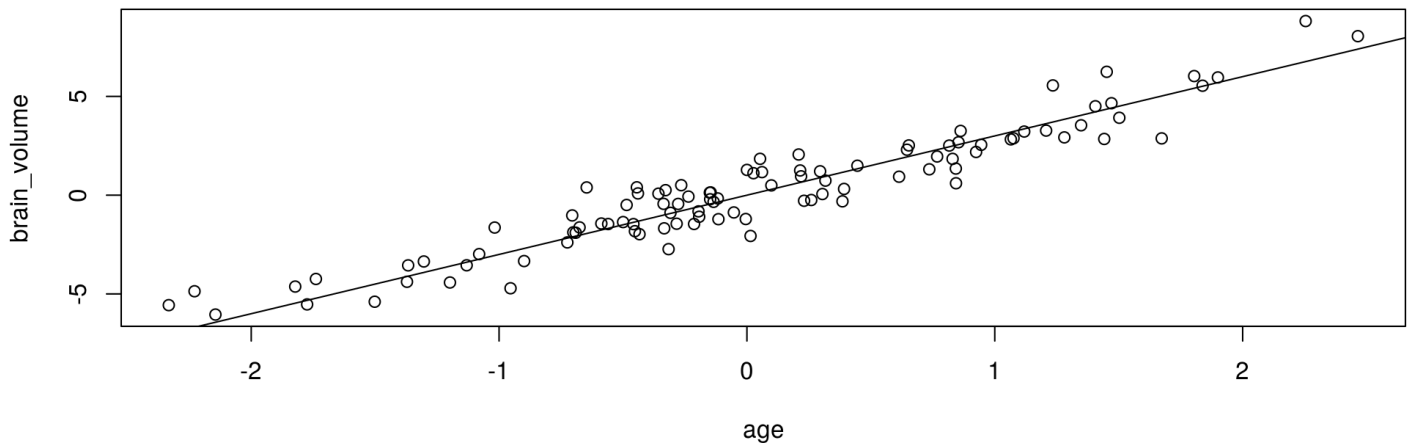
- In frequentism $\beta$ is some fixed value.
- We can estimate standard errors for $\beta$ and get p-values (likelihoods) that each component of $\beta$ is zero.

# Bayesian interpretation

- In bayesianism $\beta$ is a random variable that we're trying to learn about.
- In order to do this we have specify our prior belief about $\beta$
- If we say we know nothing about $\beta$, we get identical estimates to frequentism
- For our model we'll say $\beta \sim \mathbb{N}(0, 5)$

# Simulate some data

```
set.seed(20180613)
age <- rnorm(100)
brain_volume <- 3 * age + rnorm(100)
plot(age, brain_volume)
abline(0, 3)
```



# Fit a bayesian linear model

- For this we'll use the package `rstanarm`
-

```
suppressMessages(library(rstanarm))
ex <- data.frame(brain_volume = brain_volume, age = age)
bl <- stan_glm(brain_volume ~ age, data = ex,
                 prior = normal(0, 5))
```

```
##
## SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
##
## Gradient evaluation took 3.2e-05 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0.32 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

# How'd we do

```
bl
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      brain_volume ~ age
##  observations: 100
##  predictors:   2
## ------
##             Median MAD_SD
## (Intercept) 0.1    0.1
## age         2.8    0.1
## sigma       1.0    0.1
##
## Sample avg. posterior predictive distribution of y:
##          Median MAD_SD
## mean_PPD 0.2    0.1
##
## ------
## For info on the priors used see help('prior_summary.stanreg').
```

# How does `lm` do?

```
lmod <- lm(brain_volume ~ age, data = ex)
summary(lmod)
```

```
##
## Call:
## lm(formula = brain_volume ~ age, data = ex)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.18341 -0.64482  0.02391  0.53349  2.32316
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.07600    0.09525   0.798    0.427
## age          2.84502    0.09565  29.745   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9515 on 98 degrees of freedom
## Multiple R-squared:  0.9003,    Adjusted R-squared:  0.8993
## F-statistic: 884.8 on 1 and 98 DF,  p-value: < 2.2e-16
```

# Side-By-Side

```
coef(bl)
```

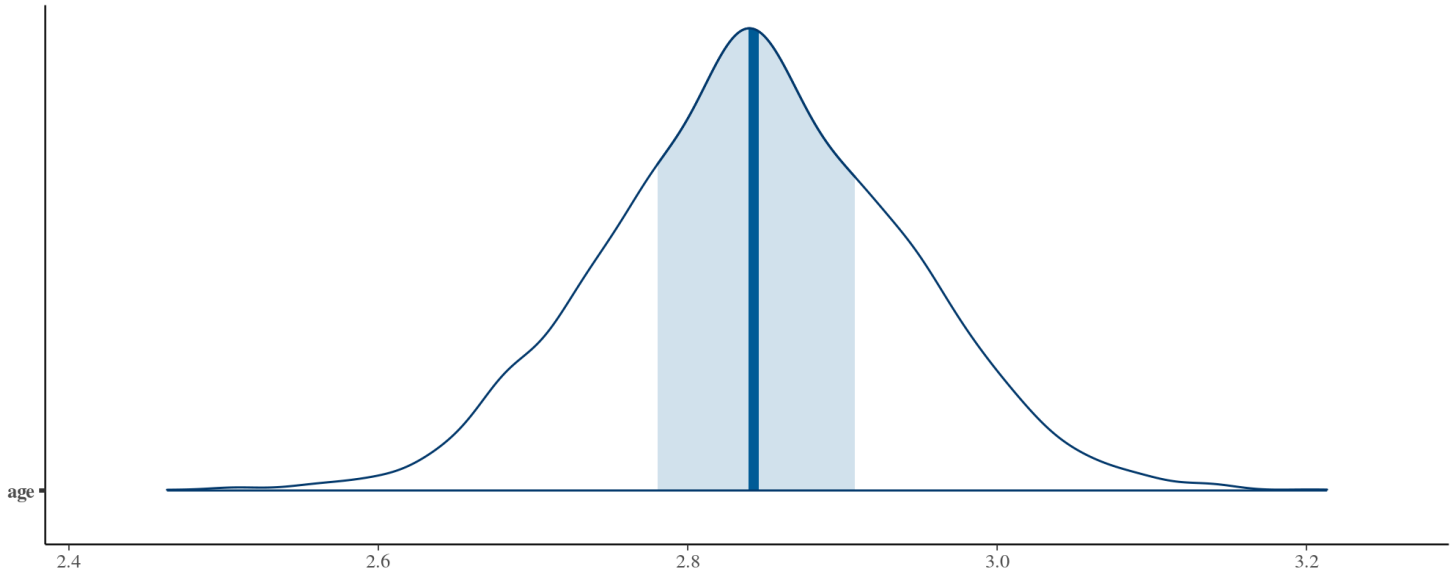
```
## (Intercept)         age
##   0.0778588   2.8426813
```

```
coef(lmod)
```

```
## (Intercept)         age
##   0.0760004   2.8450169
```

# Let's look at the posterior

```
suppressPackageStartupMessages(library(bayesplot))
mcmc_areas(as.matrix(bl), pars = "age")
```



# What's happening here?

- `rstanarm` is creating a posterior for us, but how?
- in most bayesian textbooks this is shown first analytically for simple models. *this is not what stan does*
- Stan *approximates* the posterior using samples
- Samples are generated with markov-chain monte carlo (MCMC)
- For more details on the technique see Michael Betancourt's A conceptual introduction to Hamiltonian Monte Carlo

# The posterior revisited

```
bl_post <- as.matrix(bl)

str(bl_post[,"age"])
```

```
##  num [1:4000] 2.91 2.93 2.82 2.87 2.93 ...
```

```
summary(bl_post[,"age"])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.463   2.780   2.843   2.844   2.908   3.213
```

# With real data now

- Let's look at the effect of diagnosis on CA1 volume

```
scale_vec <- function(x) (x - mean(x)) / sd(x)
ca1_ex <-
  adni %>%
  mutate(volume = scale_vec(L_CA1 + R_CA1))
```

- Scaling here helps improve model fitting, an allows you to interpret parameters in terms of whole sample standard deviations.
- For details on why this is a good idea consider checking out Gelman and Hill's Data Analysis Using Regression and Multilevel/Hierarchical Models

# Model

$$\text{volume} = \beta_D D + \epsilon$$

- We're not going to estimate an intercept for this model, fitting instead the mean volume for each diagnosis
- in a R formula it will look like

```
volume ~ -1 + DX_bl
```

- The -1 removes the intercept

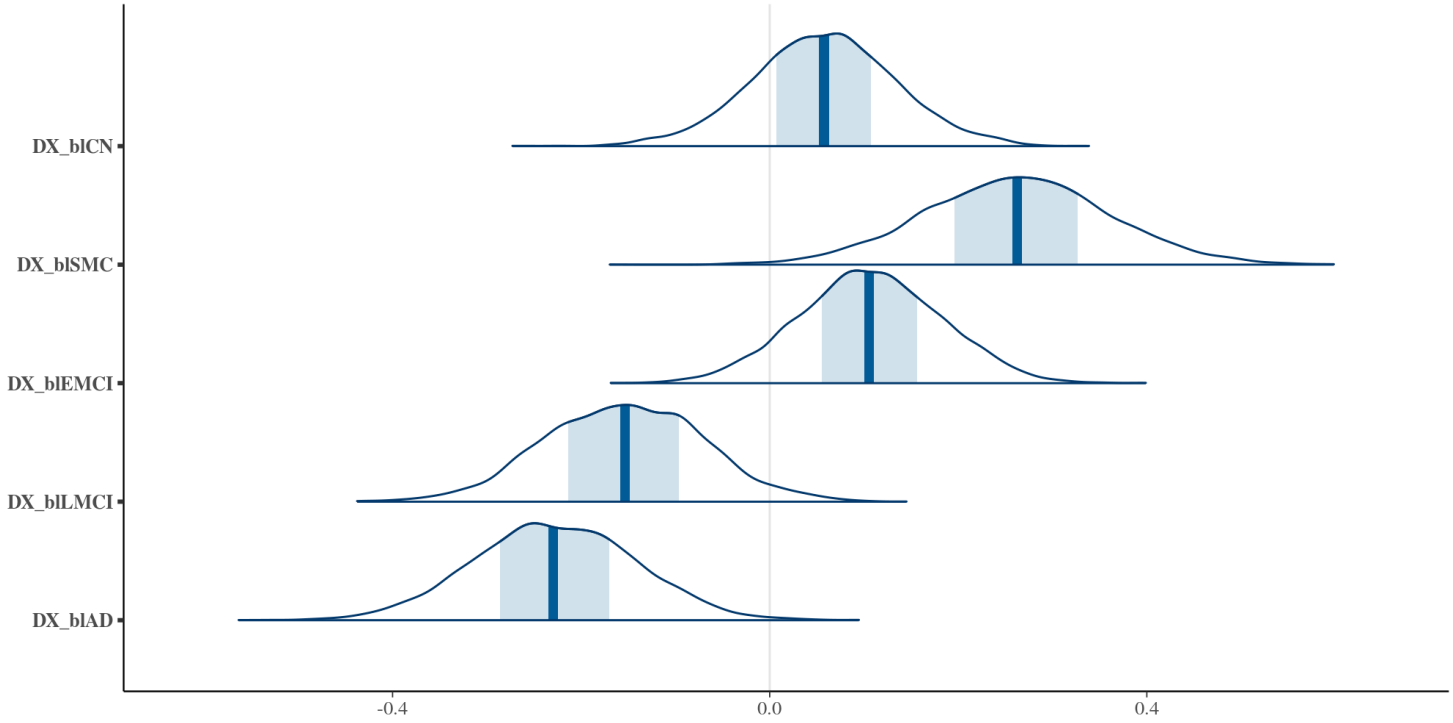
# Fit

```
invisible(
  capture.output({
    ca1_mod <- stan_glm(volume ~ -1 + DX_bl, data = ca1_ex)
  }))

ca1_mod
```

```
## stan_glm
##  family:       gaussian [identity]
##  formula:      volume ~ -1 + DX_bl
##  observations: 703
##  predictors:   5
## ------
##           Median MAD_SD
## DX_blCN    0.1    0.1
## DX_blSMC   0.3    0.1
## DX_blEMCI  0.1    0.1
## DX_blLMCI -0.2    0.1
## DX_blAD   -0.2    0.1
## sigma      1.0    0.0
##
## Sample avg. posterior predictive distribution of y:
##          Median MAD_SD
## mean_PPD 0.0    0.1
##
## ------
## For info on the priors used see help('prior_summary.stanreg')
```

So we see that the effect of each diagnosis has on CA1 volume. Let's visualize the posterior for these effects

```
mcmc_areas(as.matrix(ca1_mod), regex_pars = "DX.*")
```



```
table(ca1_ex$DX_bl)
```

```
##
##   CN  SMC EMCI LMCI   AD
##  167   99  163  141  133
```

So we can see some signs that the posterior width is driven in part by sample size, nice.

# Posterior Magic

First let's extract the posterior

```
ca1_post <- as.matrix(ca1_mod)
```

What do we think the probability is that AD patients have smaller CA1s than controls? Easy to answer with the posterior!

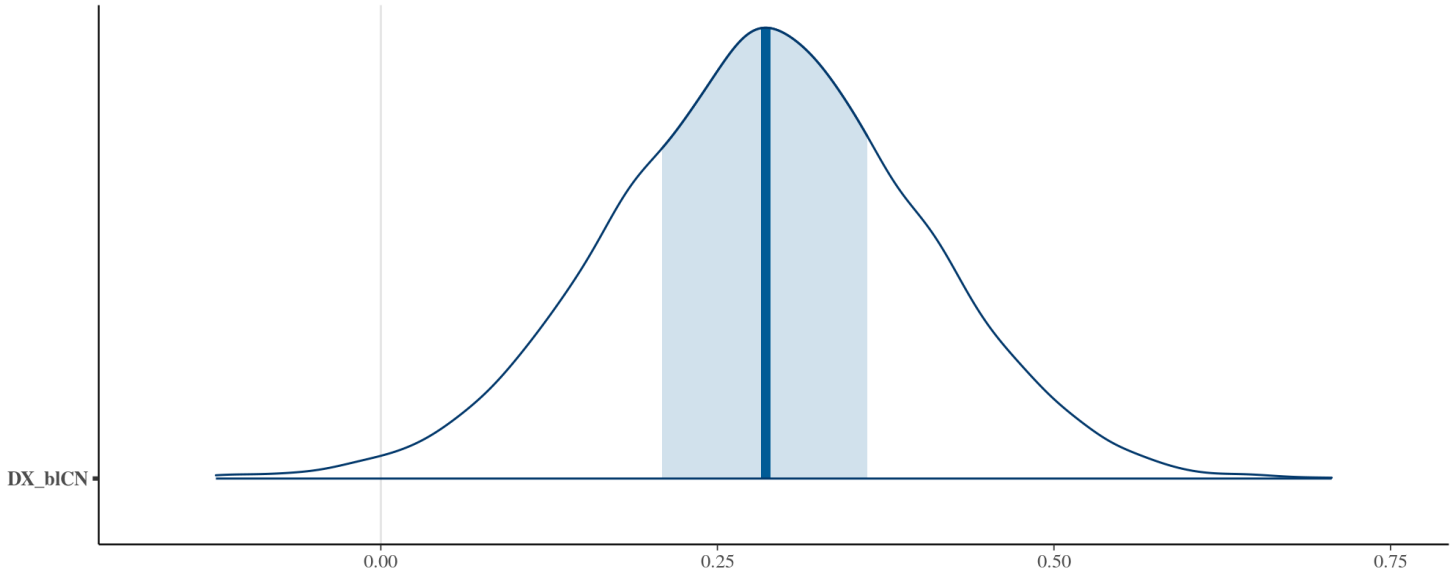
```
mean(ca1_post[,"DX_blCN"] > ca1_post[,"DX_blAD"])
```

```
## [1] 0.99125
```

So we're pretty sure about this.

# But how big is the difference?

```
mcmc_areas(ca1_post[,"DX_blCN", drop = FALSE] - ca1_post[,"DX_blAD"])
```



# Looking at many structures

# Data preparation

```
hpc$Do(function(x){
  x$normVolumes <- scale(x$volumes)
})

hpcSym <- Clone(hpc)
Prune(hpcSym, isNotLeaf)
```

```
## [1] 18
```

```
data <- hpcSym$Get("normVolumes", filterFun=isLeaf)
colnames(data) <- hpcSym$Get("name", filterFun = isLeaf)
adniLongNorm <- adni %>%
  mutate(AGE=AGE/10) %>%
  select(PTID:DX_bl) %>%
  cbind(data) %>%
  # sample_n(100) %>% # to make it not last an ice-age
  gather(structure, volume, CA1:Fornix)
```

# The model

```
flathierarchy <- stan_lmer(
  volume ~ AGE+PTGENDER+DX_bl + (1|PTID) +
           (AGE+PTGENDER+DX_bl|structure),
  adniLongNorm , chains=3, cores=3, adapt_delta = 0.99)
```

The model takes a while to run, unless you specified sample_n to a relatively small number. So just load the output from an earlier, saved run instead.

```
flathierarchy <- readRDS("flathierarchy.Rds")
```

# The model

```
flathierarchy <- stan_lmer(
  volume ~ AGE+PTGENDER+DX_bl + (1|PTID) +
           (AGE+PTGENDER+DX_bl|structure),
  adniLongNorm , chains=3, cores=3, adapt_delta = 0.99)
```

What is this model doing?

- volume ~ AGE + PTGENDER + DX_bl -> the same model we saw before
- (1 | PTID) -> allow for variation across individuals; i.e. those with a larger CA1 are also likely to have a larger DG
- (AGE+PTGENDER+DX_bl | structure) -> repeat of the model, but now grouped by structure.

# The model

```
flathierarchy <- stan_lmer(
  volume ~ AGE+PTGENDER+DX_bl + (1|PTID) +
           (AGE+PTGENDER+DX_bl|structure),
  adniLongNorm , chains=3, cores=3, adapt_delta = 0.99)
```

What is this model doing?

- volume ~ AGE + PTGENDER + DX_bl -> the same model we saw before
- (1 | PTID) -> allow for variation across individuals; i.e. those with a larger CA1 are also likely to have a larger DG
- (AGE+PTGENDER+DX_bl | structure) -> repeat of the model, but now grouped by structure.

Huh?

# The model

```
flathierarchy <- stan_lmer(
  volume ~ AGE+PTGENDER+DX_bl + (1|PTID) +
           (AGE+PTGENDER+DX_bl|structure),
  adniLongNorm , chains=3, cores=3, adapt_delta = 0.99)
```

What is this model doing?

- volume ~ AGE + PTGENDER + DX_bl -> the same model we saw before
- (1 | PTID) -> allow for variation across individuals; i.e. those with a larger CA1 are also likely to have a larger DG
- (AGE+PTGENDER+DX_bl | structure) -> repeat of the model, but now grouped by structure.

Huh?

- the main effects (first part) are across all structures
- for each structure, the deviation from the main effect is estimated

# The magic of pooling

How do we work with multiple brain structures at the same time? By pooling. Here's what that could mean:

# The magic of pooling

How do we work with multiple brain structures at the same time? By pooling. Here's what that could mean:

**Complete pooling:** we ignore that we have multiple structures, and sum them. I.e. a single model across overall hippocampal volume.

# The magic of pooling

How do we work with multiple brain structures at the same time? By pooling. Here's what that could mean:

**Complete pooling:** we ignore that we have multiple structures, and sum them. I.e. a single model across overall hippocampal volume.

**No pooling:** we compute a separate model for each structure. What happens in CA1 has no bearing on what happens in CA2/CA3. This is what we do with our massively univariate models. You need to control for multiple comparisons by widening your confidence intervals.

# The magic of pooling

How do we work with multiple brain structures at the same time? By pooling. Here's what that could mean:

**Complete pooling:** we ignore that we have multiple structures, and sum them. I.e. a single model across overall hippocampal volume.

**No pooling:** we compute a separate model for each structure. What happens in CA1 has no bearing on what happens in CA2/CA3. This is what we do with our massively univariate models. You need to control for multiple comparisons by widening your confidence intervals.

**Partial pooling:** what happens in CA1 will *somewhat* influence what happens in CA2/CA3, in that the model will share information across structures. These are our hierarchical Bayesian models.

# The magic of pooling

How do we work with multiple brain structures at the same time? By pooling. Here's what that could mean:

**Complete pooling:** we ignore that we have multiple structures, and sum them. I.e. a single model across overall hippocampal volume.

**No pooling:** we compute a separate model for each structure. What happens in CA1 has no bearing on what happens in CA2/CA3. This is what we do with our massively univariate models. You need to control for multiple comparisons by widening your confidence intervals.

**Partial pooling:** what happens in CA1 will *somewhat* influence what happens in CA2/CA3, in that the model will share information across structures. These are our hierarchical Bayesian models.

Partial pooling, once more: the prior for each structure is the pooled estimated of all structures. The stronger the data for each structure (i.e. the likelihood), the less the estimate will shrink towards the pooled estimates. And the pooled estimates are themselves determined from the data.

# The magic of pooling

How do we work with multiple brain structures at the same time? By pooling. Here's what that could mean:

**Complete pooling:** we ignore that we have multiple structures, and sum them. I.e. a single model across overall hippocampal volume.

**No pooling:** we compute a separate model for each structure. What happens in CA1 has no bearing on what happens in CA2/CA3. This is what we do with our massively univariate models. You need to control for multiple comparisons by widening your confidence intervals.

**Partial pooling:** what happens in CA1 will *somewhat* influence what happens in CA2/CA3, in that the model will share information across structures. These are our hierarchical Bayesian models.

Partial pooling, once more: the prior for each structure is the pooled estimated of all structures. The stronger the data for each structure (i.e. the likelihood), the less the estimate will shrink towards the pooled estimates. And the pooled estimates are themselves determined from the data.

And since all structures share information, you need not control for multiple comparisons. Shrinkage replaces widening of confidence intervals.

# Covariance priors

So if the prior for each structure is estimated from the data, we don't need a prior? Not quite - we have to decide how pooling happens.

# Covariance priors

So if the prior for each structure is estimated from the data, we don't need a prior? Not quite - we have to decide how pooling happens.

This is our model from before:

$y = \alpha + X\beta + Zb + \epsilon$

- where $X$ is the matrix of predictors for the main effect, AGE+PTGENDER+DX_bl in our model.
- and where $Z$ is the matrix that encodes deviation in the predictors across groups, or structures in our case, which we specified as (AGE+PTGENDER+DX_bl|structure) and (1|PTID).

# Covariance priors

So if the prior for each structure is estimated from the data, we don't need a prior? Not quite - we have to decide how pooling happens.

This is our model from before:

$y = \alpha + X\beta + Zb + \epsilon$

- where $X$ is the matrix of predictors for the main effect, AGE+PTGENDER+DX_bl in our model.
- and where $Z$ is the matrix that encodes deviation in the predictors across groups, or structures in our case, which we specified as (AGE+PTGENDER+DX_bl|structure) and (1|PTID).

The likelihood is thus

$y \sim \mathcal{N}(\alpha + X\beta + Zb, \sigma^2 I)$

- intercept and coefficients common across structures
- deviations in intercepts and coefficients that vary across structures

# Covariance priors

Back to the likelihood:

$y \sim \mathcal{N}(\alpha + X\beta + Zb, \sigma^2 I)$

- intercept and coefficients common across structures
- deviations in intercepts and coefficients that vary across structures

# Covariance priors

Back to the likelihood:

$y \sim \mathcal{N}(\alpha + X\beta + Zb, \sigma^2 I)$

- intercept and coefficients common across structures
- deviations in intercepts and coefficients that vary across structures

We now need priors on parameters. We maintain the simple assumption that $b \sim \mathcal{N}(0, \Sigma)$ - i.e. the deviation for each structure is centred at zero with some variance.

# Covariance priors

Back to the likelihood:

$y \sim \mathcal{N}(\alpha + X\beta + Zb, \sigma^2 I)$

- intercept and coefficients common across structures
- deviations in intercepts and coefficients that vary across structures

We now need priors on parameters. We maintain the simple assumption that $b \sim \mathcal{N}(0, \Sigma)$ - i.e. the deviation for each structure is centred at zero with some variance.

But now we need to state our prior beliefs about $\Sigma$ in addition to $\alpha$, $\beta$, and $\sigma$.

# Covariance priors

Back to the likelihood:

$y \sim \mathcal{N}(\alpha + X\beta + Zb, \sigma^2 I)$

- intercept and coefficients common across structures
- deviations in intercepts and coefficients that vary across structures

We now need priors on parameters. We maintain the simple assumption that $b \sim \mathcal{N}(0, \Sigma)$ - i.e. the deviation for each structure is centred at zero with some variance.

But now we need to state our prior beliefs about $\Sigma$ in addition to $\alpha$, $\beta$, and $\sigma$.

Our prior for varying slopes and intercepts is a "zero-mean random vector following a multivariate Gaussian distribution with an unknown covariance matrix to be estimated"

# Covariance priors

Back to the likelihood:

$y \sim \mathcal{N}(\alpha + X\beta + Zb, \sigma^2 I)$

- intercept and coefficients common across structures
- deviations in intercepts and coefficients that vary across structures

We now need priors on parameters. We maintain the simple assumption that $b \sim \mathcal{N}(0, \Sigma)$ - i.e. the deviation for each structure is centred at zero with some variance.

But now we need to state our prior beliefs about $\Sigma$ in addition to $\alpha$, $\beta$, and $\sigma$.

Our prior for varying slopes and intercepts is a "zero-mean random vector following a multivariate Gaussian distribution with an unknown covariance matrix to be estimated"

More detail here: [http://mc-stan.org/rstanarm/articles/glmer.html]

# Taking a peek

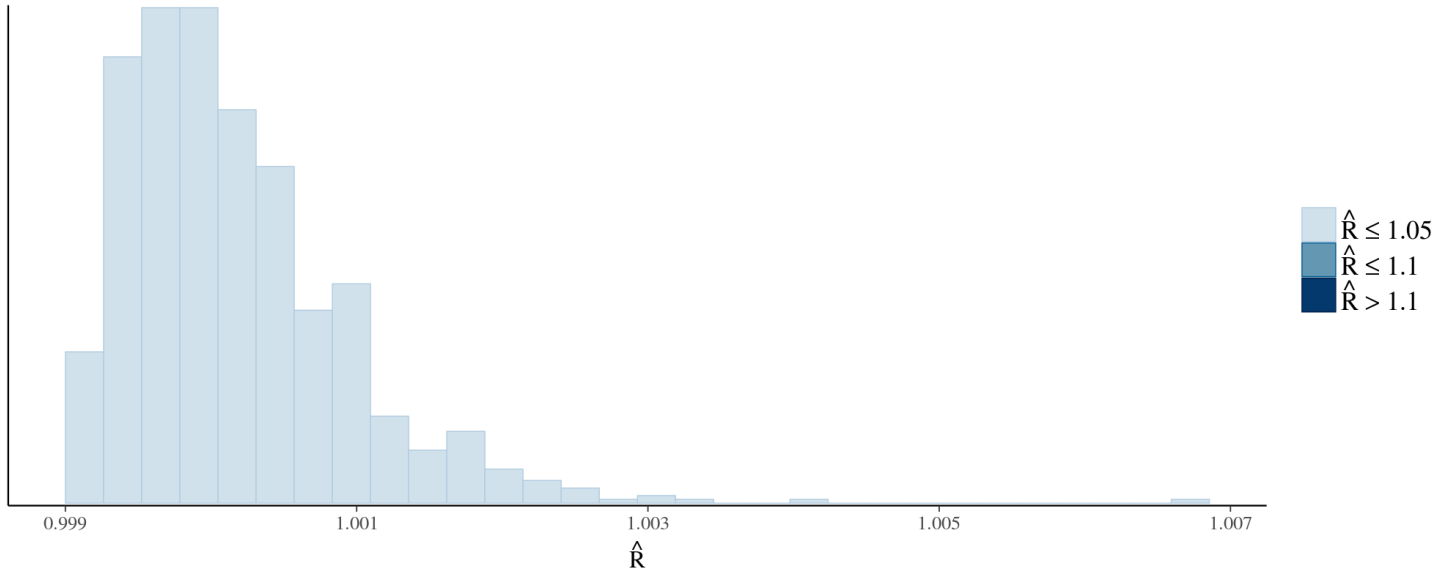
```
summary(flathierarchy, digits=3)
```

```
##
## Model Info:
##
##  function:     stan_lmer
##  family:       gaussian [identity]
##  formula:      volume ~ AGE + PTGENDER + DX_bl + (1 | PTID) + (AGE + PTGENDER +
##        DX_bl | structure)
##  algorithm:    sampling
##  priors:       see help('prior_summary')
##  sample:       3000 (posterior sample size)
##  observations: 6327
##  groups:       PTID (703), structure (9)
##
## Estimates:
##                                                mean      sd        2.5%
## (Intercept)                                     0.589     0.416    -0.219
## AGE                                            -0.110     0.056    -0.223
## PTGENDERMale                                    0.631     0.074     0.488
## DX_blSMC                                        0.152     0.099    -0.042
## DX_blEMCI                                      -0.024     0.091    -0.207
## DX_blLMCI                                      -0.270     0.110    -0.488
## DX_blAD                                        -0.414     0.141    -0.702
## b[(Intercept) PTID:002_S_4213]                  0.355     0.198    -0.023
## b[(Intercept) PTID:002_S_4219]                  0.081     0.205    -0.332
## b[(Intercept) PTID:002_S_4225]                  0.303     0.206    -0.203
```

# Quick diagnostics: Rhat

```
mcmc_rhat_hist(rhat(flathierarchy))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

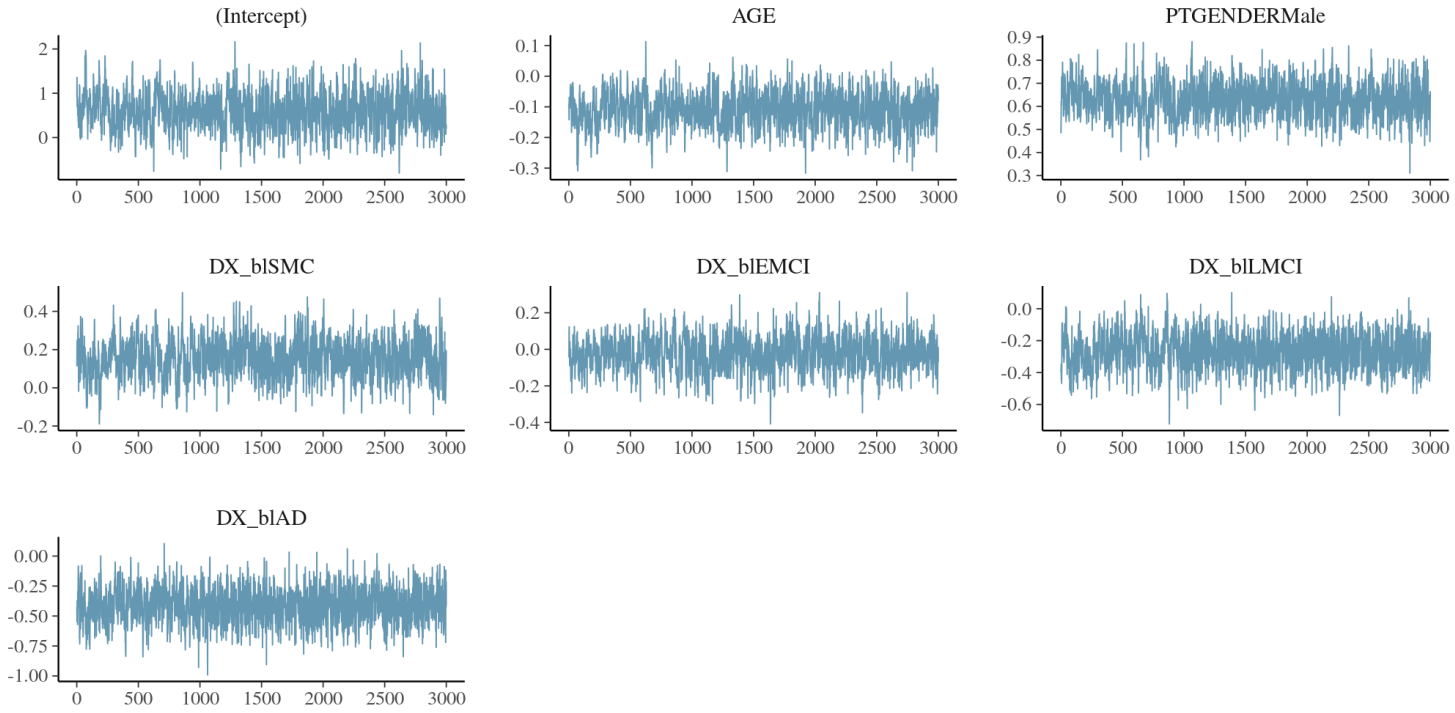


# More diagnostics: trace plots

We'll need data in matrix form for this and many future ops

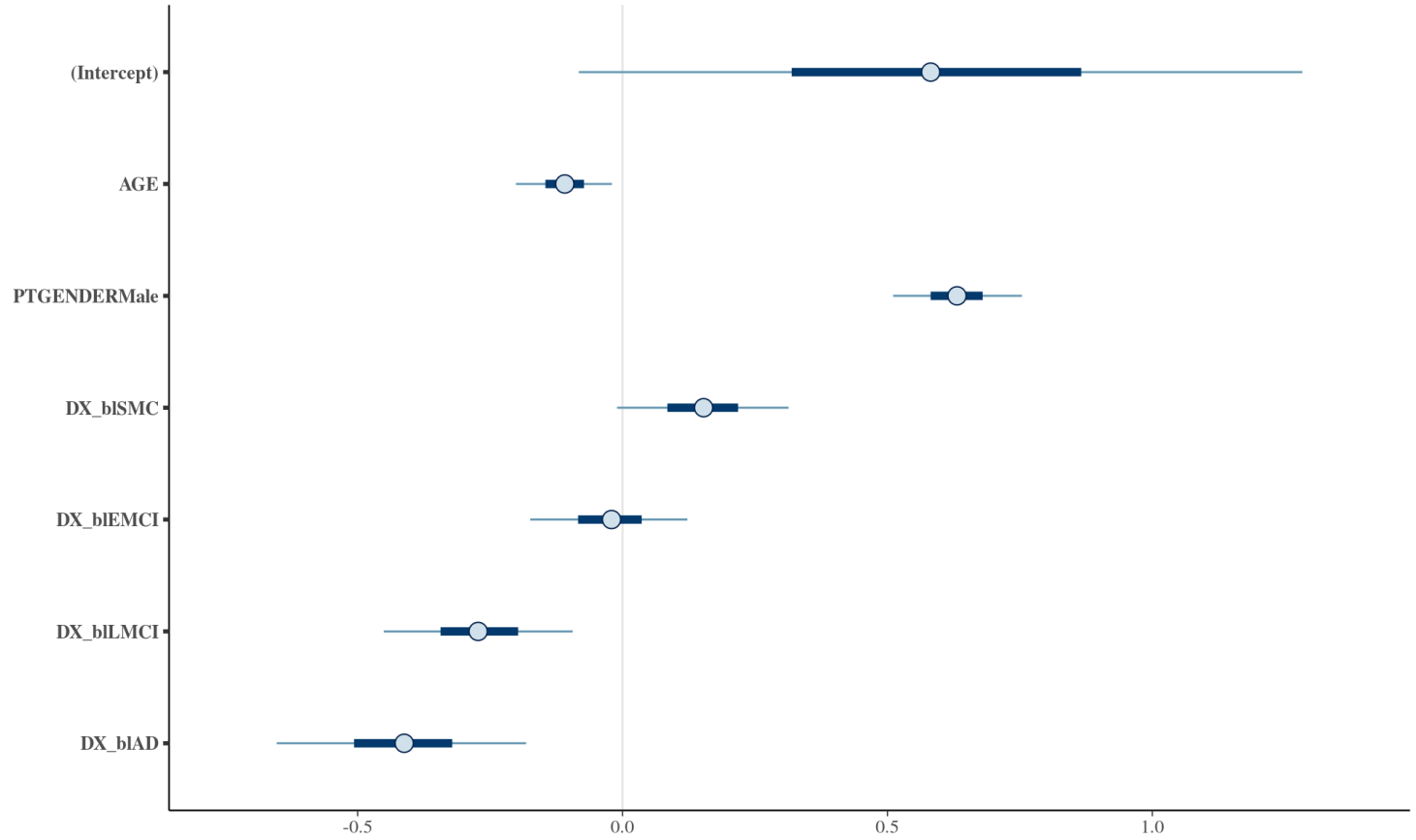
```
flathierarchym <- as.matrix(flathierarchy)
```

```
mcmc_trace(flathierarchym[,1:7])
```



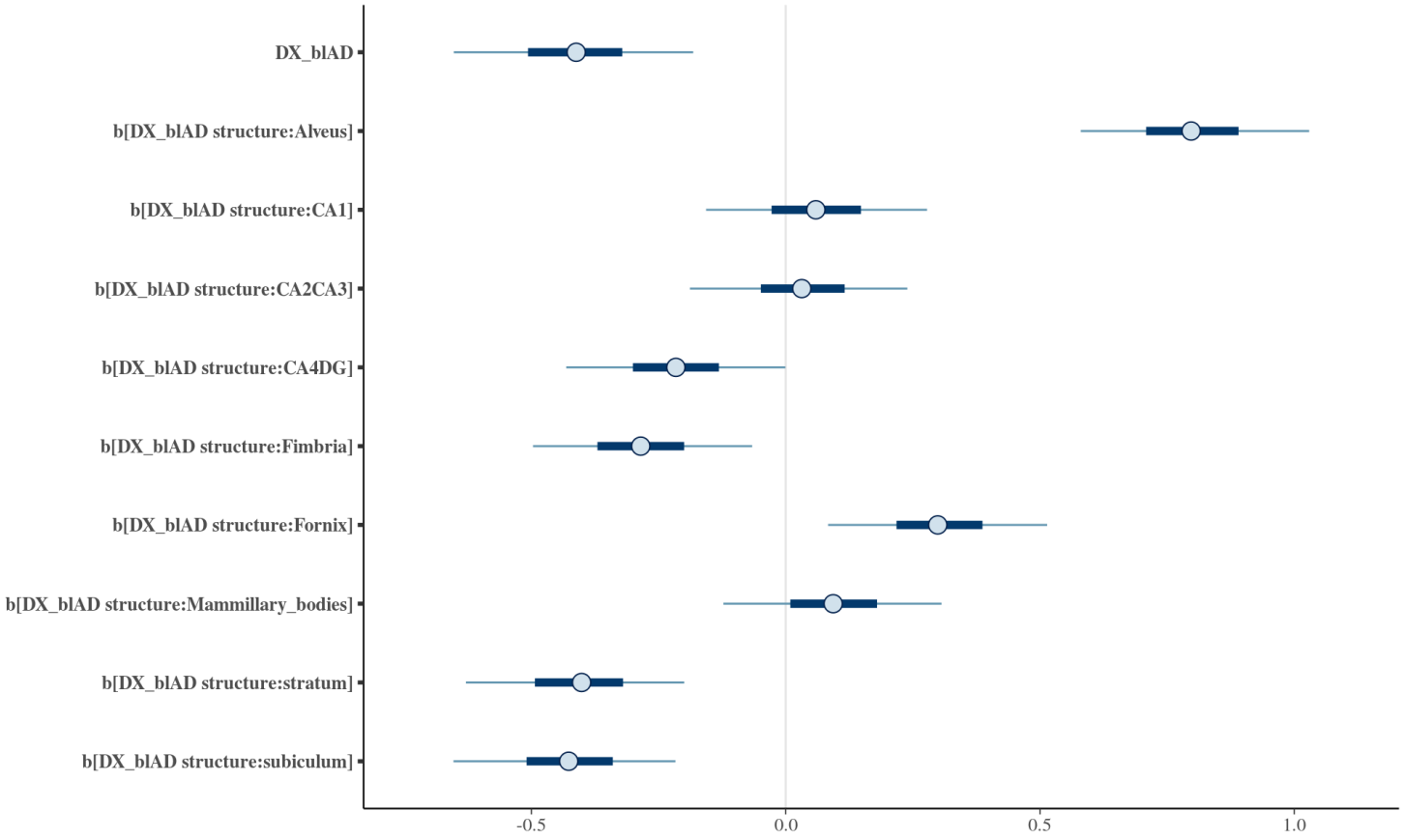
# The main effects

```
mcmc_intervals(flathierarchym[,1:7])
```



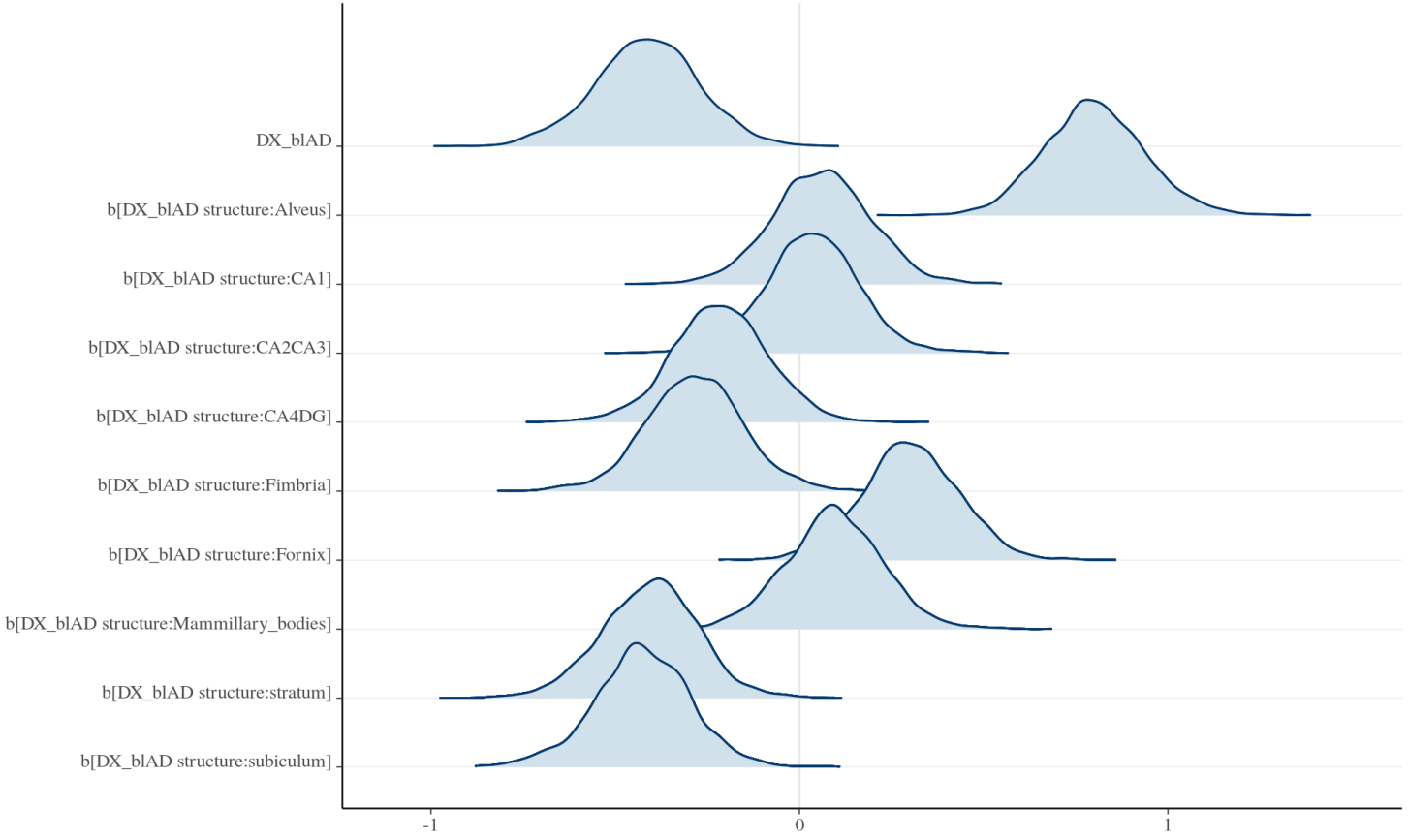
# Effect across structures

```
mcmc_intervals(flathierarchym,
  pars="DX_blAD", regex_pars = 'DX_blAD structure')
```



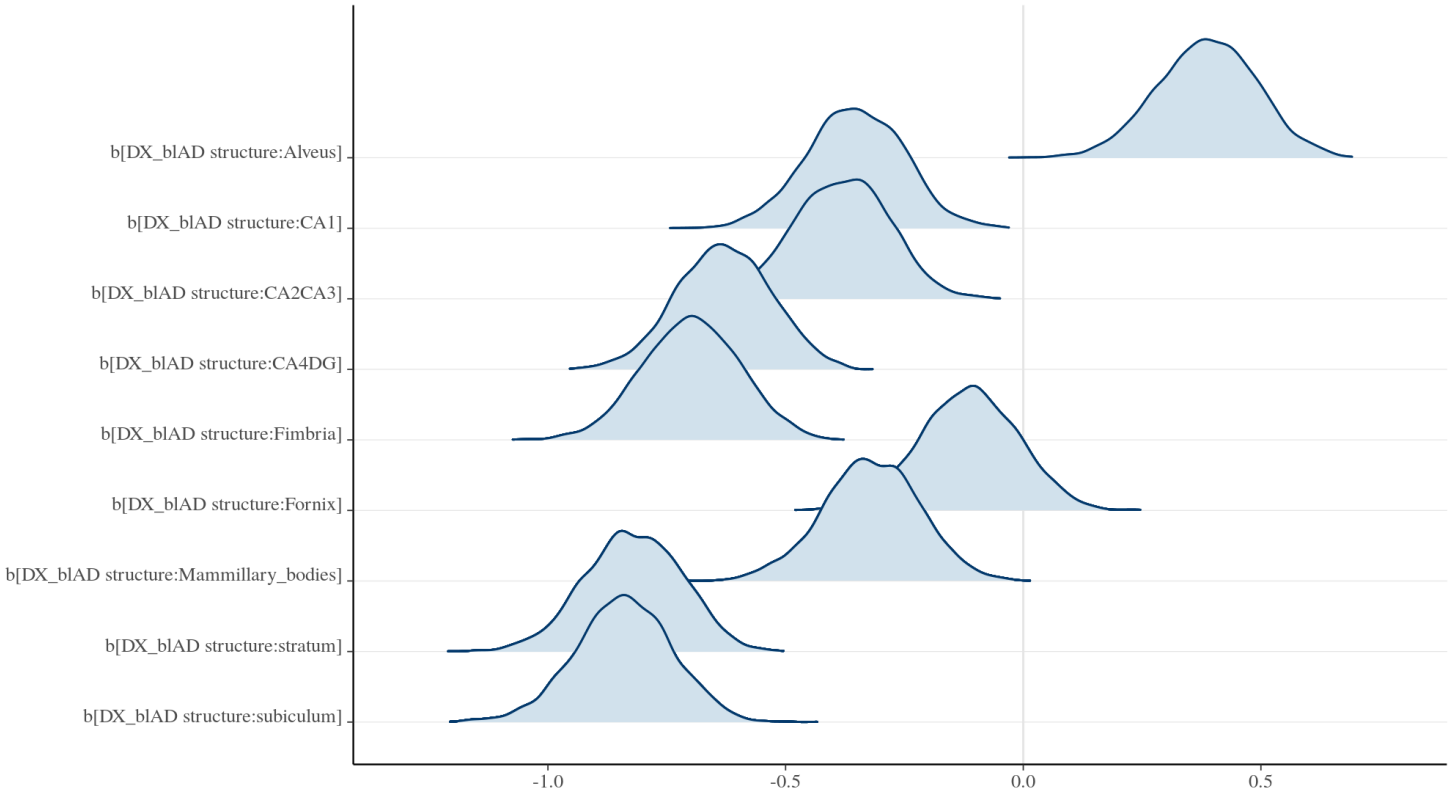
# Effect across structures

```
mcmc_areas_ridges(flathierarchym,
  pars="DX_blAD", regex_pars = 'DX_blAD structure')
```



# Summed effects

```
vars <- colnames(flathierarchym)
ADsamples <- flathierarchym[,grep('DX_blAD structure', vars)]
mcmc_areas_ridges(ADsamples + flathierarchym[,"DX_blAD"])
```



# Probability calculations

What is the probability that CA1 was more affected than CA2/CA3 in AD?

```
mean(flathierarchym[,'b[DX_blAD structure:CA1]'] <
       flathierarchym[,'b[DX_blAD structure:CA2CA3]'])
```

```
## [1] 0.3806667
```

Not very likely

```
quantile(flathierarchym[,'b[DX_blAD structure:CA1]'] -
       flathierarchym[,'b[DX_blAD structure:CA2CA3]'])
```

```
##          0%         25%         50%         75%        100%
## -0.27756997 -0.03709035  0.02793064  0.09358587  0.37136809
```

# Probability calculations

What is the probability that the subiculum was more affected than CA1 in AD?

```
mean(flathierarchym[,'b[DX_blAD structure:subiculum]'] <
       flathierarchym[,'b[DX_blAD structure:CA1]'])
```

```
## [1] 1
```

Almost certainly

```
quantile(flathierarchym[,'b[DX_blAD structure:subiculum]'] -
       flathierarchym[,'b[DX_blAD structure:CA1]'])
```

```
##          0%         25%         50%         75%        100%
## -0.8460248 -0.5503639 -0.4879033 -0.4216953 -0.1389502
```

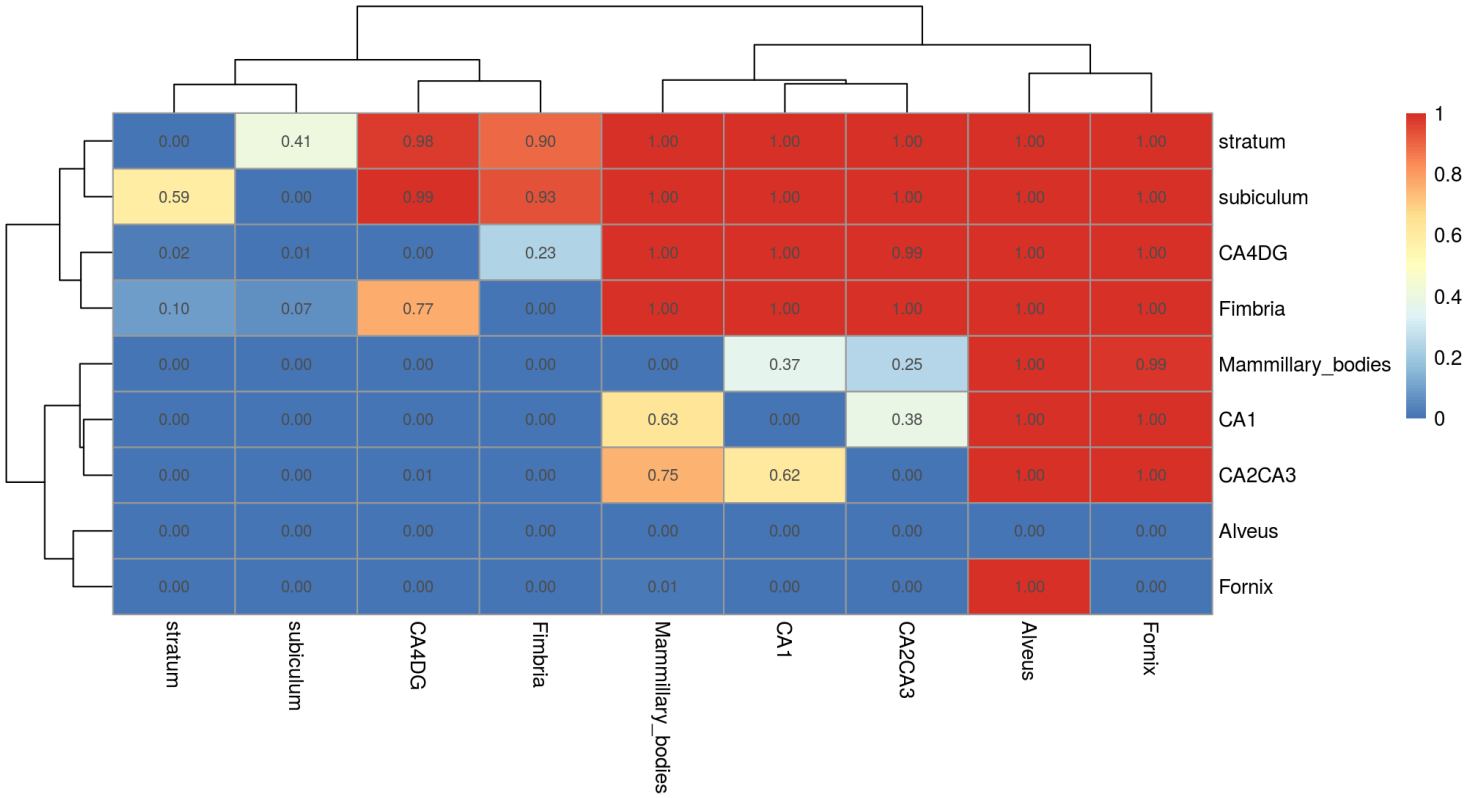
# All pairs of probabilities

```
ADvars <- grep("DX_blAD structure", vars)
probcomps <- matrix(nrow=length(ADvars), ncol=length(ADvars))
for (i in 1:length(ADvars)) {
  for (j in 1:length(ADvars)) {
    probcomps[i,j] <- mean(flathierarchym[,ADvars[i]] <
                           flathierarchym[,ADvars[j]])
  }
}
```

```
pnames <- sub('.+structure:(.+)\\]', '\\1',
              colnames(flathierarchym[,ADvars]))
rownames(probcomps) <- pnames
colnames(probcomps) <- pnames
```

# All pairs of probabilities

```
library(pheatmap)
pheatmap(probcomps, display_numbers=T)
```



# Estimating shrinkage

First, compute the least squares estimates

```
lmEstimates <- adniLongNorm %>%
  split(.$structure) %>%
  map(~lm(volume ~ AGE+PTGENDER+DX_bl, .)) %>%
  map_dfr(tidy, .id='roi') %>%
  filter(term == "DX_blAD") %>%
  select(roi, estimate)
```

# Estimating shrinkage

Next, compute the medians from the Hierarchical Bayesian model

```
sweptADsamples <- flathierarchym[,ADvars] + flathierarchym[,"DX_blAD"]
lmAndBayes <- sweptADsamples %>%
  as_data_frame %>%
  gather(term, value) %>%
  group_by(term) %>%
  summarize(median=median(value)) %>%
  mutate(term=sub('.+structure:(.+)\\]', '\\1', term)) %>%
  mutate(term=sub('_', ' ', term)) %>%
  inner_join(lmEstimates, by=c("term" = "roi"))
```

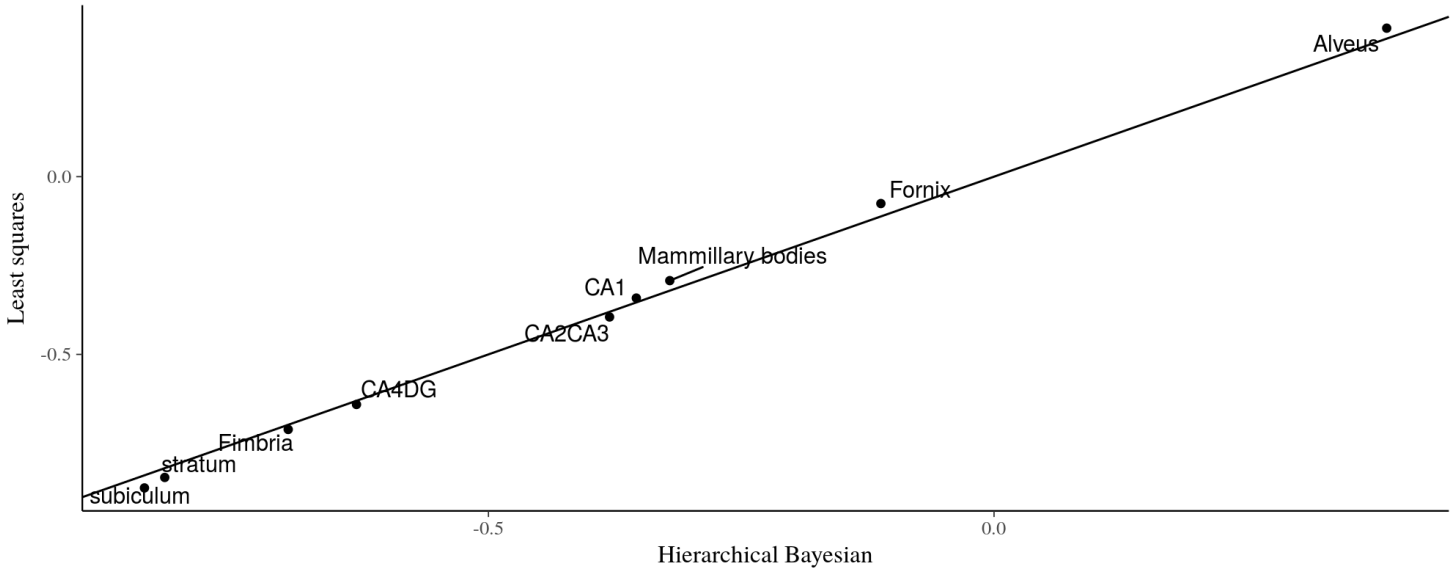
# Estimating Shrinkage

```
lmAndBayes %>% mutate(diff=abs(median - estimate))
```

```
## # A tibble: 9 x 4
##   term              median estimate   diff
##   <chr>              <dbl>    <dbl>  <dbl>
## 1 Alveus             0.388   0.418  0.0294
## 2 CA1               -0.354  -0.342  0.0122
## 3 CA2CA3            -0.380  -0.395  0.0146
## 4 CA4DG             -0.630  -0.641  0.0104
## 5 Fimbria           -0.698  -0.711  0.0132
## 6 Fornix            -0.112  -0.0757 0.0362
## 7 Mammillary bodies -0.321  -0.293  0.0282
## 8 stratum           -0.820  -0.846  0.0261
## 9 subiculum         -0.840  -0.876  0.0355
```

# Estimating Shrinkage

```
library(ggrepel)
ggplot(lmAndBayes) +
  aes(median, estimate) +
  geom_point() +
  geom_abline(aes(intercept=0, slope=1)) +
  geom_text_repel(aes(label=term)) +
  xlab("Hierarchical Bayesian") +
  ylab("Least squares")
```



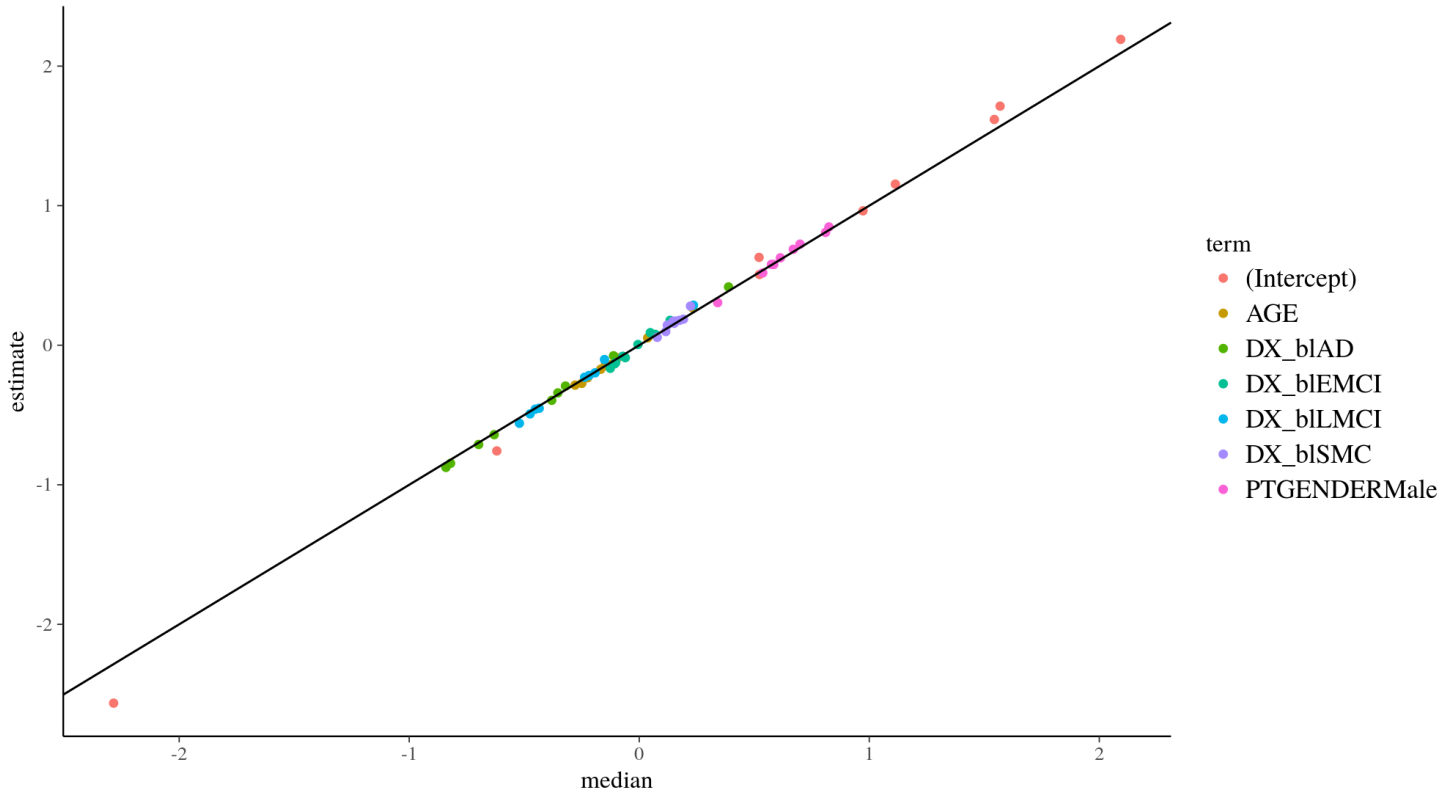
# Estimating shrinkage across all terms

```
structvars <- grep('b.+structure', vars, value = T)
mainterms <- sub('b\\[(.+) str.+', '\\1', structvars)

structswmain <- flathierarchym[,structvars] + flathierarchym[,mainterms]

bOut <- structswmain %>%
  as.data.frame %>%
  gather(mterm, value) %>%
  group_by(mterm) %>%
  summarize(median=median(value)) %>%
  mutate(roi = sub('.+structure:(.+)\\]', '\\1', mterm),
         roi = sub('_', ' ', roi),
         term = sub('b\\[(.+) str.+', '\\1', mterm)) %>%
  select(median, roi, term)


lOut <- adniLongNorm %>%
  split(.$structure) %>%
  map(~lm(volume ~ AGE+PTGENDER+DX_bl, .)) %>%
  map_dfr(tidy, .id='roi') %>%
  select(estimate, roi, term)

blOut <- inner_join(bOut, lOut)
```

```
## Joining, by = c("roi", "term")
```

# Estimating shrinkage across all terms

```
ggplot(blOut) +
  aes(median, estimate, colour=term) +
  geom_point() +
  geom_abline(aes(intercept=0, slope=1))
```



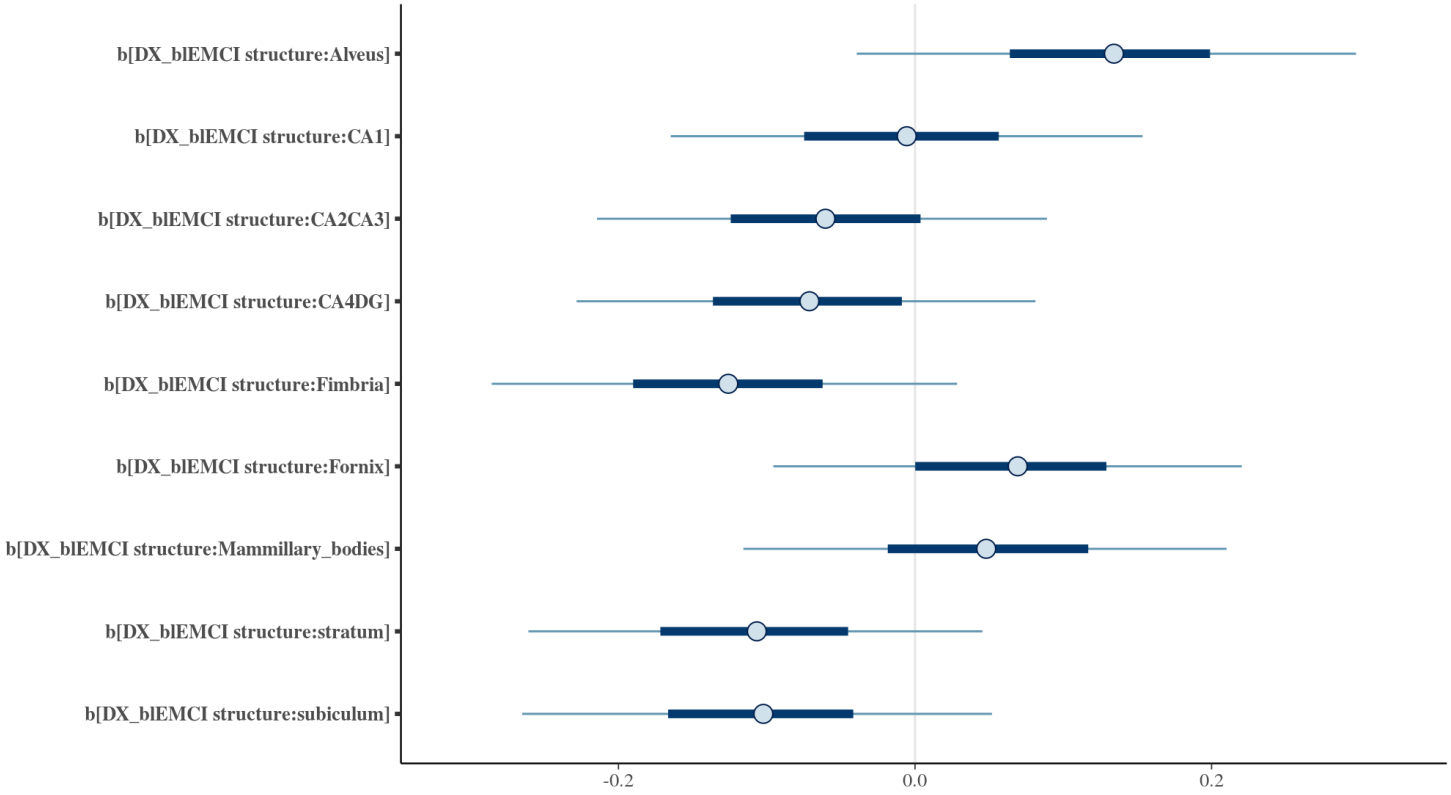
# Estimating shrinkage across all terms

```
ggplot(blOut) +
  aes(median, estimate, colour=roi) +
  geom_point() +
  geom_abline(aes(intercept=0, slope=1)) +
  xlab("Hierarchical bayes") +
  ylab("Least squares") +
  facet_wrap(~term, scales="free")
```

# What about the MCI groups?

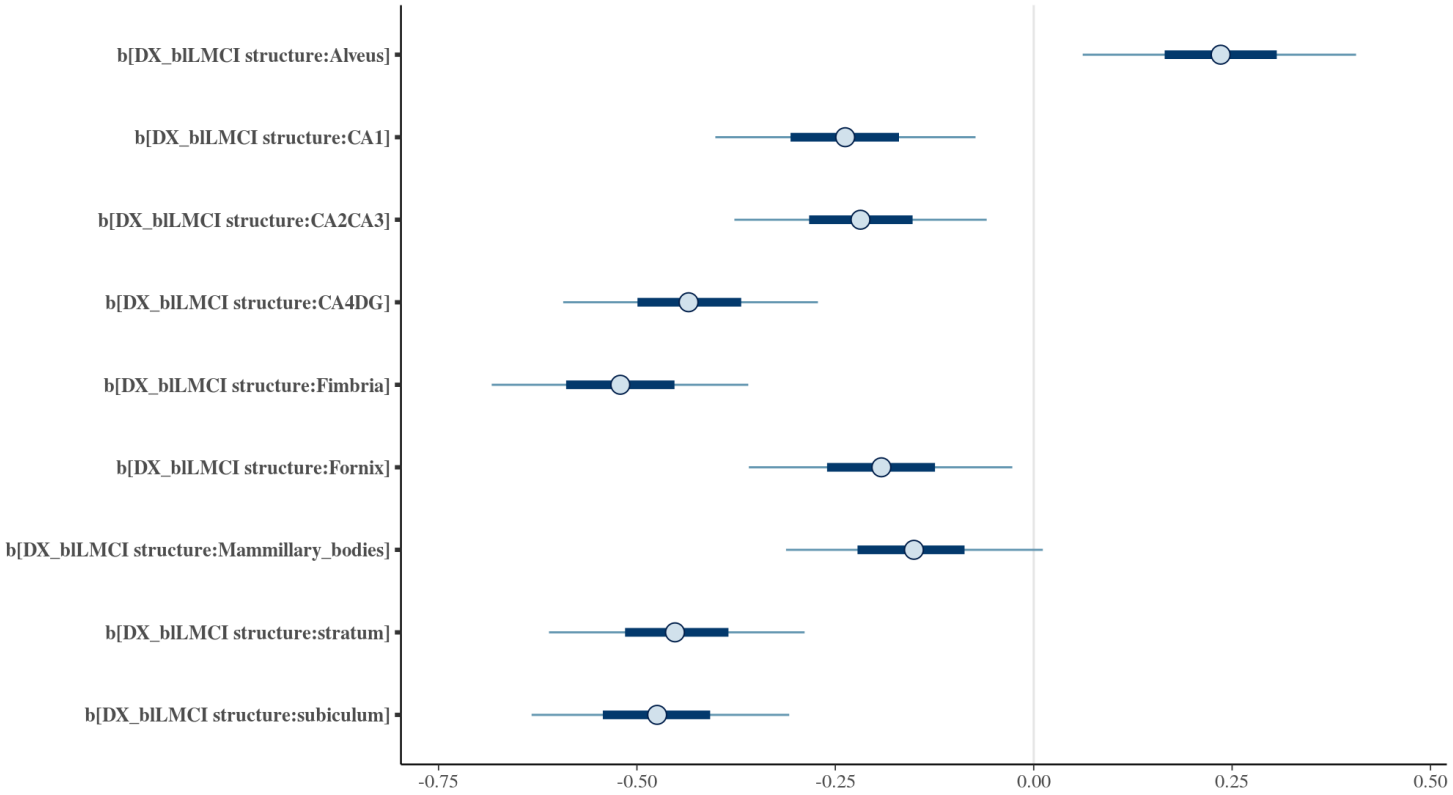
Remember, nothing survived FDR comparisons ...

```
mcmc_intervals(structswmain, regex_pars="EMCI")
```



# And LMCI

```
mcmc_intervals(structswmain, regex_pars="LMCI")
```



# A reminder about LMCI FDR

```
hpcSym$Do(function(x){
  adni$volume <- x$normVolumes
  x$stats <- tidy(lm(volume ~ AGE+PTGENDER+DX_bl, adni))
  x$LMCI <- x$stats %>% filter(term=="DX_blLMCI") %>% select(p.value)
})
round(p.adjust(
  hpcSym$Get("LMCI", filterFun = isLeaf), 'fdr'),
  3)
```

```
##               CA1             CA2CA3              CA4DG          subiculum
##             0.038              0.063              0.000              0.000
##           stratum Mammillary bodies             Alveus             Fimbria
##             0.000              0.340              0.008              0.000
##            Fornix
##             0.072
```

# Comparing probabilities

```
LMCIprobs <- structswmain %>%
  as.data.frame %>%
  gather(mterm, value) %>%
  group_by(mterm) %>%
  summarize(median=median(value),
            prob=case_when(
              median > 0 ~ 1 - mean(value>0),
              median < 0 ~ 1 - mean(value<0)
            )) %>%
  mutate(roi = sub('.+structure:(.+)\\]', '\\1', mterm),
         roi = sub('_', ' ', roi),
         term = sub('b\\[(.+) str.+', '\\1', mterm)) %>%
  filter(term=="DX_blLMCI") %>%
  select(roi, median, prob)

LMCIq <- hpcSym$Get("LMCI", filterFun = isLeaf) %>%
  as.data.frame() %>%
  gather(roi, p) %>%
  mutate(q=p.adjust(p, 'fdr'),
         roi=sub('\\.', ' ', roi))
```

# Comparing probabilities

```
inner_join(LMCIprobs, LMCIq) %>% mutate_at(vars(median:q), round, 3)
```

```
## Joining, by = "roi"
```

```
## # A tibble: 9 x 5
##   roi                median    prob       p       q
##   <chr>               <dbl>   <dbl>   <dbl>   <dbl>
## 1 Alveus              0.236 0.0110  0.00400 0.00800
## 2 CA1                -0.238 0.00900 0.0250  0.0380
## 3 CA2CA3             -0.218 0.0150  0.0490  0.0630
## 4 CA4DG              -0.435 0       0       0
## 5 Fimbria            -0.521 0       0       0
## 6 Fornix             -0.192 0.0280  0.0640  0.0720
## 7 Mammillary bodies -0.151 0.0630  0.340   0.340
## 8 stratum            -0.452 0       0       0
## 9 subiculum          -0.475 0       0       0
```

# A more complex hierarchy

Create an ancestor variable from the tree, in this case corresponding to WM or GM

```
hpc$Do(function(x){
  x$normVolumes <- scale(x$volumes)
})
hpc$Do(function(x){
  x$tissue <- x$parent$name
}, filterFun = isLeaf)

hpcSym <- Clone(hpc)
Prune(hpcSym, isNotLeaf)
```

```
## [1] 18
```

```
hpcSym$Do(function(x){
  x$tissue <- x$parent$name
}, filterFun = isLeaf)

data <- hpcSym$Get("normVolumes", filterFun=isLeaf)
colnames(data) <- hpcSym$Get("name", filterFun = isLeaf)

tissueTable <- ToDataFrameTable(hpcSym, "name", "tissue")
```

# A more complex hierarchy

```
adniLongNorm <- adni %>% mutate(AGE=AGE/10) %>%
  select(PTID:DX_bl) %>%
  cbind(data) %>%
  #sample_n(50) %>%
  gather(structure, volume, CA1:Fornix) %>%
  left_join(tissueTable, by=c("structure" = "name"))

adniLongNorm %>% sample_n(5)
```

```
##            PTID VISCODE ImageUID ORIGPROT COLPROT  AGE APOE4 PTGENDER
## 2815 002_S_4225      bl   258686    ADNI2   ADNI2 6.99     1     Male
## 475  099_S_4157      bl   254781    ADNI2   ADNI2 8.11     0   Female
## 4187 141_S_4438      bl   277669    ADNI2   ADNI2 7.68     0     Male
## 2451 052_S_4959      bl   394766    ADNI2   ADNI2 7.75     1     Male
## 5292 067_S_5205      bl   377885    ADNI2   ADNI2 5.93     0   Female
##      DX_bl         structure     volume tissue
## 2815    CN           stratum  0.8452908     GM
## 475   EMCI               CA1 -0.2452777     GM
## 4187  EMCI Mammillary bodies -0.9427007     GM
## 2451    AD         subiculum  1.5569059     GM
## 5292    AD           Fimbria  0.4295960     WM
```

# A more complex hierarchy

Allow for structure within tissue

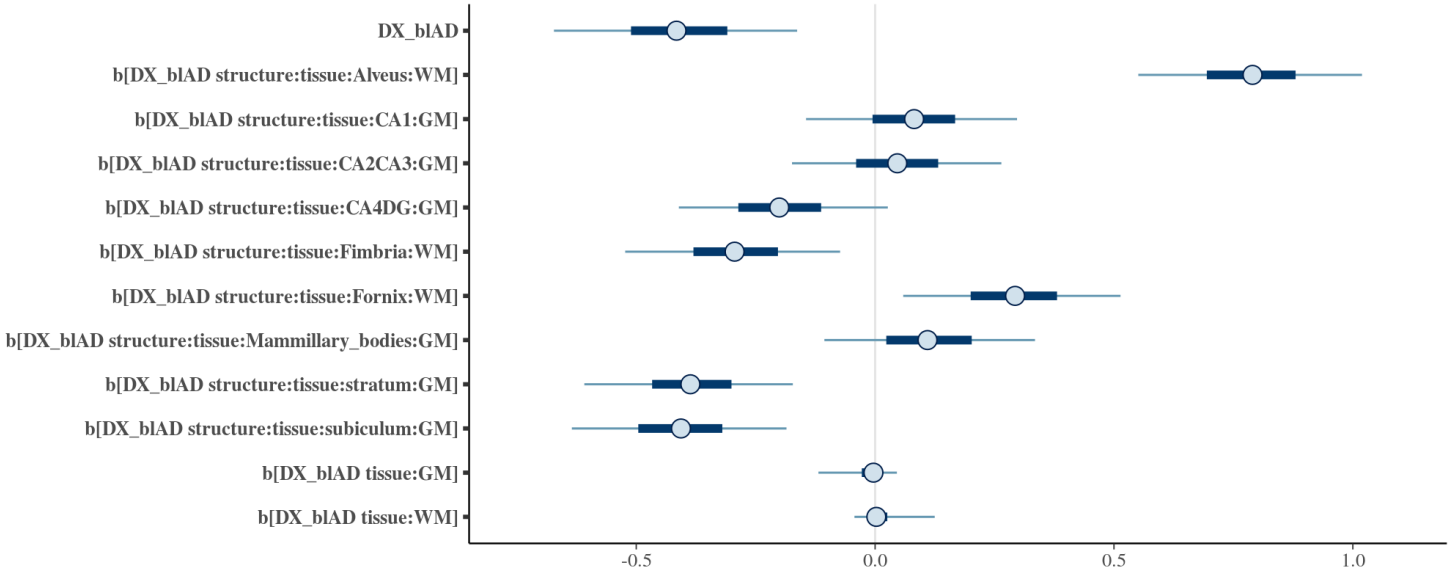
```
# this takes a long time unless a small subset of the data is used
twolevelhierarchy <-
  stan_lmer(volume ~ AGE+PTGENDER+DX_bl + (1|PTID) +
              (AGE+PTGENDER+DX_bl|tissue/structure),
            adniLongNorm , chains=3, cores=3, adapt_delta = 0.99)
```

```
# loads the precomputed output from the above command
twolevelhierarchy <- readRDS("twolevelhierarchy.Rds")
twolevelhierarchym <- as.matrix(twolevelhierarchy)
```

# A look at the new encoding

Remember that you are encoding the difference from the level one up

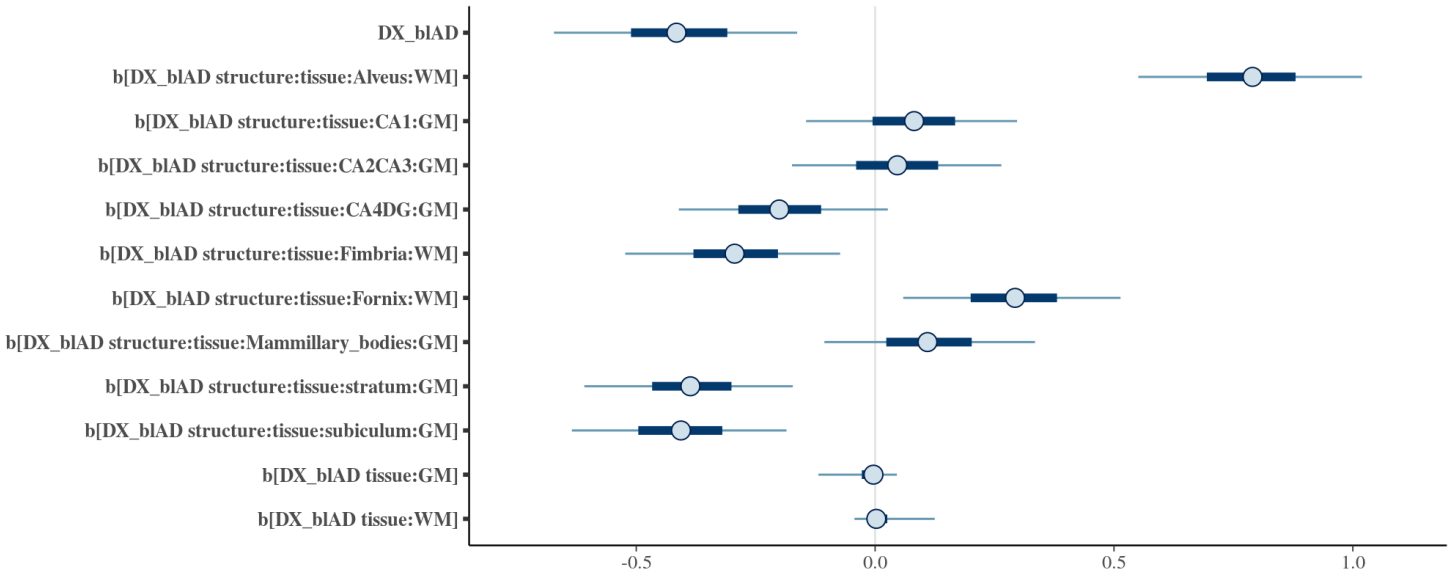
```
mcmc_intervals(twolevelhierarchym, regex_pars="b\\[DX_blAD", pars="DX_blAD")
```



# A look at the new encoding

Remember that you are encoding the difference from the level one up

```
mcmc_intervals(twolevelhierarchym, regex_pars="b\\[DX_blAD", pars="DX_blAD")
```

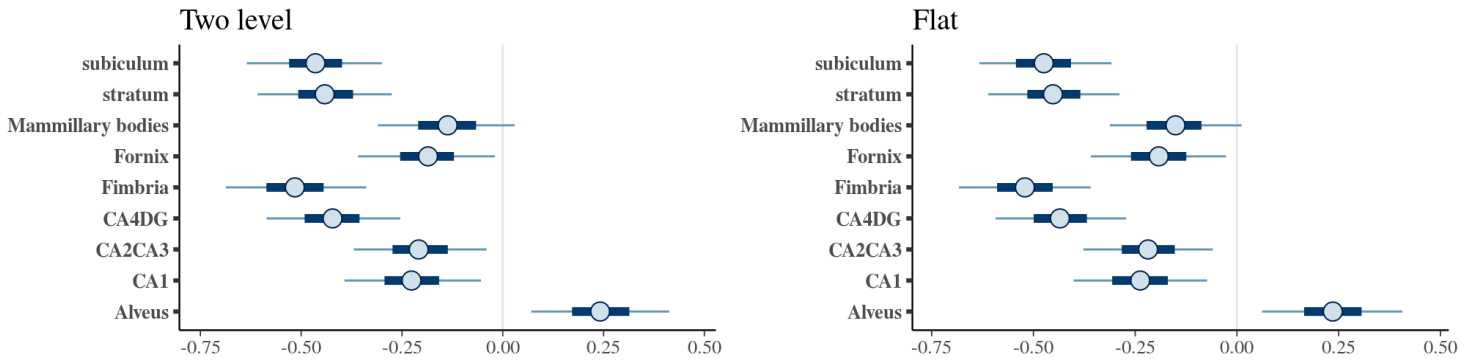



In this case, the extra hierarchy level has little effect

# A look at plots - LMCI as before

```
suppressMessages(library(gridExtra))
vars <- colnames(twolevelhierarchym)
structvars <- grep('b.+structure', vars, value = T)

correspondingtissue <- sub('b\\[(.+) structure:tissue:.+:(.+)\\]', 'b[\\1 tissue:\\2]', structvars)
correspondingmain <- sub('b\\[(.+) structure:tissue:.+:(.+)\\]', '\\1', structvars)

twolevelstructs <- twolevelhierarchym[,structvars] +
  twolevelhierarchym[,correspondingtissue] +
  twolevelhierarchym[,correspondingmain]

ylabels <- tissueTable %>% arrange(name) %>% select(name)

grid.arrange(
  mcmc_intervals(twolevelstructs, regex_pars = "DX_blLMCI") +
    ggtitle("Two level") + scale_y_discrete(labels=ylabels$name),
  mcmc_intervals(structswmain, regex_pars = "DX_blLMCI") +
    ggtitle("Flat") + scale_y_discrete(labels=ylabels$name),
  ncol=2)
```

```
## Scale for 'y' is already present. Adding another scale for 'y', which
## will replace the existing scale.
## Scale for 'y' is already present. Adding another scale for 'y', which
## will replace the existing scale.
```



# Installing RMINC

- RMINC provides many convenience functions for working with hierarchically structured volume data.
- The lead developers are responsive on github and willing to offer help if needed.
- Full install guide can be found at https://github.com/Mouse-Imaging-Centre/RMINC/blob/master/INSTALL

# Linux install:

1. Get the minc-toolkit-v2 (https://bic-mni.github.io/)
2. Source the configure script `source /opt/minc/1.9.16/minc-toolkit-config.sh` or where-ever your minc-toolkit version was installed
3. Set your `MINC_PATH` environment variable `export MINC_PATH=/opt/minc/1.9.16/`
4. Open an R session
5. Install `devtools` if you have not already `install.packages("devtools")`
6. Install `RMINC` with

```
devtools::install_github("Mouse-Imaging-Centre/RMINC",
  dependencies = c("Depends", "Imports", "LinkingTo",
                   "Suggests","Enhances"))

#Mac Install

Mac install is very similar, however fortran libraries are necessary, and so you
package manager [brew](https://brew.sh/). From there you will need to run `brew i

Then find a directory that looks like:
```

```
/usr/local/Cellar/gcc/7.*.*/lib/gcc/7
```