Supplementary Materials for
Gut Microbiota in Parkinson’s Disease: Temporal stability and relations to disease progression

Velma T. E. Aho; Pedro A. B. Pereira; Sari Voutilainen; Lars Paulin; Eero Pekkonen; Petri Auvinen; Filip Scheperjans

Table of Contents

Supplementary results

Dietary data analyses

The subjects’ diet was measured using a Food Frequency Questionnaire (FFQ). PD patients had an overall higher energy intake than control subjects (p = 0.005 for intake in kcal). Considering specific food items (171 variables, including all items from the FFQ and some grouped ones, such as “all dairy products” or “all fruits and vegetables”) and nutrients (46 nutrients calculated based on FFQ answers), none differed significantly between the PD and control groups after multiple comparison correction. The lowest adjusted *p*-values for continuous diet variables were for magnesium and niacin intake, with adjusted p = 0.0718 for both. For categorical versions of the same variables, split by quintiles, the variables that had the lowest adjusted *p*-values were cabbage-containing dishes and pasta dishes, with adjusted p = 0.0984 for both; these do not seem likely to be particularly meaningful. Overall, this suggests that there are no major dietary differences between groups that need to be considered as confounders. Contrasting stable and progressed patients suggested that they also have similar diets, with no statistically significant differences in energy intake (p = 0.5 for intake in kcal) or any food items or nutrients; no variables had adjusted *p*-values < 0.1.

Even though there were no differences in specific dietary items, there could be differing dietary patterns between groups. To look for these, we performed a Principal Component Analysis (PCA) with a list of 31 non-overlapping food items (Table S1). This analysis did not reveal any strong patterns. The first principal component (PC1) explained 7.6% of the variation and seemed to correspond roughly to a healthy/unhealthy diet, with positive loadings for items such as candy, butter, potatoes, bread, and confectionery, and negative loadings for nuts, muesli, fish, cottage cheese and berries (Table S1). The other PCs were not as easy to interpret; for example, for PC2, the main contributing variables included margarine, cottage cheese, fruits and confectionery (positive loadings) and wine, shellfish, fish and beer (negative loadings). Visually, PC1 and PC3 seemed to best separate controls from PD patients, suggesting that patients tend to be on the unhealthy side of PC1 (Figure 3). We kept PC1 as a potential confounder to be further assessed in microbial diversity analyses.

Table S1. Component loadings from a Principal Component Analysis of Food Frequency Questionnaire data.

	PC1	PC2	PC3	PC4	PC5
Candy	0.326	-0.065	0.058	-0.290	-0.058
Butters	0.314	0.041	-0.163	0.304	0.016
Potatoes	0.313	0.084	-0.022	0.221	0.069
Bread	0.240	0.156	0.226	0.112	0.086
Confectionery	0.236	0.207	-0.117	0.068	-0.249
Meat	0.216	-0.165	-0.257	-0.127	0.126
Spirits	0.201	-0.164	0.016	-0.023	0.013
Sugar-sweetened drinks	0.135	-0.028	-0.257	-0.164	-0.237
Coffee	0.098	0.052	0.159	-0.044	0.240
Cheese	0.095	0.130	0.252	-0.325	0.186
Artificially sweetened drinks	0.060	-0.125	-0.287	0.262	-0.041
Beer	0.052	-0.264	0.147	-0.125	0.034
Margarine	0.048	0.269	0.179	-0.226	0.042
Long drink	0.028	-0.157	-0.002	-0.069	-0.183
Wine	-0.014	-0.397	0.206	-0.048	-0.020
Eggs	-0.060	-0.142	-0.132	-0.136	0.192
Processed cheese	-0.067	0.120	-0.356	-0.021	0.193
Shellfish	-0.077	-0.381	-0.174	0.088	0.123
Vegetables	-0.090	-0.026	0.088	0.445	0.259
Fruits	-0.110	0.209	0.092	0.239	0.326
Milk and ice cream	-0.119	0.189	-0.113	0.112	-0.136
Quark	-0.150	0.022	-0.203	-0.260	0.229
Fruit juice	-0.164	-0.138	-0.075	0.020	0.068
Oil	-0.180	-0.100	0.139	0.140	-0.107
Cereal	-0.182	0.125	-0.059	-0.175	-0.260
Porridge	-0.186	0.194	-0.118	0.052	-0.193
Berries	-0.196	0.077	-0.099	-0.122	0.224
Cottage cheese	-0.204	0.212	-0.261	-0.163	0.228
Fish	-0.207	-0.303	-0.084	0.017	-0.091
Muesli	-0.240	0.126	0.094	0.068	-0.391
Nuts	-0.268	-0.069	0.352	0.053	0.031

Additional diversity analyses

We used linear regression to explore the relationships between PD, confounders and alpha diversity, focusing on the variables that had a significant uncorrected *p*-value for all three indices: BMI, history of ENT surgery, and CCB medication. We found an interaction between BMI and PD in relation to the Shannon and inverse Simpson indices. The models with this interaction were a better fit and explained more variation in the data than those without it (Table S2A). A similar interaction did not improve the models for CCB medication or history of ENT surgery. Observed richness was inversely correlated with BMI regardless of PD status. We also looked for associations between alpha diversity and the Victoria Bowel Performance Scale (BPS) for stool consistency. Linear regression of average BPS score, PD status, and alpha diversity indices suggested an interaction between PD and BPS in for observed richness, but not for the Shannon and inverse Simpson indices (Table S2B).

Table S2. Linear regression and interactions of alpha diversity indices, PD status and confounders.

A. Body Mass Index (BMI; using data from both timepoints)

Model	Variable	Adjusted R- squared	Estimate	*p*-value
Observed richness ~ PD + BMI	model	0.033		***0.006***
	Intercept		528.93	< ***0.001***
	PD		4.07	0.738
	BMI		-4.88	***0.001***
Observed richness ~ PD * BMI	model	0.034		***0.010***
	Intercept		582.09	< ***0.001***
	PD		-84.87	0.309
	BMI		-6.89	***0.004***
	PD : BMI		3.33	0.281
Shannon ~ PD + BMI	model	0.018		***0.039***
	Intercept		4.16	< ***0.001***
	PD		-0.02	0.720
	BMI		-0.01	***0.013***

Shannon ~ PD * BMI	model	0.036		***0.007***
	Intercept		4.58	***< 0.001***
	PD		-0.71	***0.017***
	BMI		-0.03	***0.001***
	PD : BMI		0.03	***0.018***
Inverse Simpson ~ PD + BMI	model	0.009		0.115
Inverse Simpson ~ PD * BMI	model	0.024		***0.030***
	Intercept		35.97	***< 0.001***
	PD		-14.84	***0.029***
	BMI		-0.57	***0.004***
	PD : BMI		0.54	***0.032***

B. Victoria Bowel Performance Scale (BPS; follow-up only)

Model	**Variable**	**Adjusted R-squared**	**Estimate**	***p*-value**
Observed richness ~ PD + BPS	model	0.028		0.061
	Intercept		413.42	***< 0.001***
	PD		-11.71	0.492
	BPS		-17.43	***0.019***
Observed richness ~ PD * BPS	model	0.083		***0.003***
	Intercept		414.48	***< 0.001***
	PD		-6.35	0.703
	BPS		-48.01	***< 0.001***
	PD : BPS		44.56	***0.004***
Shannon ~ PD + BPS	model	-0.011		0.754
Shannon ~ PD * BPS	model	0.007		0.284
Inverse Simpson ~ PD + BPS	model	-0.015		0.913
Inverse Simpson ~ PD * BPS	model	-0.013		0.723

Statistically significant *p*-values are marked in bold italic font.

Regarding diet and microbiota data, we first looked for associations between diet variables and alpha diversity, using microbiota data from the follow-up timepoint, which is when the FFQ data was collected. We tested the first five PCs from the diet PCA, all nutrients, the set of 31 food items used for the PCA, and the use of probiotics. None had a significant effect on any of the three alpha diversity indices (adjusted $p > 0.2$ for all variables).

Beta diversity comparisons for categorical dietary variables at follow-up, run with adonis on OTU level data first with only control subjects, then with all subjects correcting for PD status, resulted in two variables that were significant for both comparisons: niacin intake ($p < 0.02$ for both) and PC1 ($p < 0.04$ for both), indicating that overall, diet variables are not strong confounders in this data (Table S3), but that PC1 perhaps captures some dietary variation that is relevant for the microbial community composition, supporting its use in further comparisons.

Table S3. Adonis: Dietary confounders, OTU-level data

Food item	***p*-value, controls only**	***p*-value, all subjects, corrected for PD status**
Niacin	***0.017***	***0.012***
Diet PC1	***0.038***	***0.026***
Protein	***0.036***	0.145
Linoleic acid	0.645	***0.039***
Sterols	0.098	***0.001***
Lactose	0.161	***0.047***
Vitamin E	0.291	***0.042***
Bread (all types)	***0.026***	0.141
Porridge	0.410	***0.047***
Butter / butter-type products	***0.049***	0.085
Fresh vegetables	***0.038***	0.134
Fruit juice	***0.004***	0.273
Candy	0.501	***0.047***
Sugar-sweetened drinks	***0.015***	0.095

Parkinson's disease phenotypes and microbiota

We ran additional comparisons to contrast the microbiota of PD patients with a tremor dominant (TD) phenotype to those representing the postural instability and gait difficulty (PIGD) phenotype. For these analyses, we used the same subset of 56 PD patients that we had selected for the progression comparisons. There was no difference in alpha diversity ($p > 0.3$ for all three indices at both timepoints) or beta diversity ($p > 0.2$ at both timepoints) between the TD and PIGD patients.

In differential abundance analyses, run separately at each timepoint, ANCOM detected only two OTUs at baseline and no taxa at follow-up. DESeq2 (uncorrected for confounders, run separately for each timepoint) detected longer lists of taxa; baseline: 14 OTUs, 5 genera and 2 families, follow-up: 15 OTUs, 2 genera and 1 family (Table S8B). Neither of the two ANCOM-detected OTUs were significant according to DESeq2. Out of the DESeq2 results, two OTUs (one representing the genus *Asteroleplasma*, the other an unclassified *Lachnospiraceae* OTU), the genus *Asteroleplasma* and the family *Anaeroplasmataceae* were significant at both timepoints. The difference in *Enterobacteriaceae* at baseline, which was previously detected with the metastats method in our pilot study (Scheperjans *et al.*, 2015), was significant with DESeq2 (adjusted $p = 0.048$), but there was no significant difference for this family at follow-up.

Supplementary tables for differential abundance analyses

Table S4. Differential abundance comparison results for PD status.

A. ANCOM (list of all differentially abundant taxa)

Level	Timepoint	Taxon
Family	baseline	Bifidobacteriaceae
Family	baseline	Prevotellaceae
Family	followup	Bifidobacteriaceae
Family	followup	Prevotellaceae
Family	followup	Puniceicoccaceae
Genus	baseline	Bifidobacterium
Genus	followup	Bifidobacterium
Genus	followup	Roseburia
Genus	followup	Prevotella
OTU	baseline	Otu0030 (Alistipes)
OTU	baseline	Otu0104 (Clostridium IV)
OTU	baseline	Otu0007 (Bifidobacterium)
OTU	baseline	Otu0377 (Ruminococcaceae unclassified)
OTU	baseline	Otu0059 (Ruminococcaceae unclassified)
OTU	baseline	Otu0217 (Intestinimonas)
OTU	followup	Otu0104 (Clostridium IV)
OTU	followup	Otu0109 (Ruminococcus)
OTU	followup	Otu0105 (Oscillibacter)
OTU	followup	Otu0051 (Clostridium sensu stricto)
OTU	followup	Otu0078 (Firmicutes unclassified)
OTU	followup	Otu0131 (Bacteroides)
OTU	followup	Otu0007 (Bifidobacterium)
OTU	followup	Otu0129 (Clostridium XlVa)

B. Random forests (list of taxa with $p < 0.05$ for mean decrease in Gini)

Level	Timepoint	Taxon	Mean decrease in Gini	*p*-value for mean decrease in Gini
Family	baseline	Lachnospiraceae	4.2204	0.0198
Family	baseline	Prevotellaceae	3.4039	0.0099
Family	baseline	Puniceicoccaceae	2.7669	0.0099
Family	baseline	Rikenellaceae	4.0141	0.0297
Family	followup	Bifidobacteriaceae	3.8736	0.0396
Family	followup	Lactobacillaceae	2.7425	0.0396
Family	followup	Puniceicoccaceae	2.0532	0.0099
Genus	baseline	Alistipes	2.2726	0.0099
Genus	baseline	Blautia	1.5800	0.0396
Genus	baseline	Fusicatenibacter	2.0744	0.0099
Genus	baseline	Prevotella	1.6880	0.0495
Genus	baseline	Roseburia	2.2940	0.0198
Genus	followup	Bifidobacterium	2.0608	0.0396

Genus	followup	Butyricicoccus	2.5655	0.0099
Genus	followup	Clostridium_XlVa	2.4202	0.0198
Genus	followup	Faecalicoccus	1.3253	0.0099
Genus	followup	Granulicatella	0.5614	0.0495
Genus	followup	Lachnospira	2.0733	0.0495
Genus	followup	Lactobacillus	1.3817	0.0297
Genus	followup	Prevotella	1.4953	0.0297
Genus	followup	Roseburia	2.2810	0.0099
OTU	baseline	Otu0003 (Roseburia)	0.6162	0.0198
OTU	baseline	Otu0008 (Fusicatenibacter)	0.7087	0.0297
OTU	baseline	Otu0024 (Blautia)	0.6429	0.0198
OTU	baseline	Otu0030 (Alistipes)	0.8226	0.0099
OTU	baseline	Otu0059 (Ruminococcaceae unclassified)	0.5939	0.0198
OTU	baseline	Otu0062 (Blautia)	0.4059	0.0297
OTU	baseline	Otu0067 (Oscillibacter)	0.5226	0.0198
OTU	baseline	Otu0073 (Ruminococcaceae unclassified)	0.4875	0.0297
OTU	baseline	Otu0074 (Clostridium XlVa)	0.6206	0.0099
OTU	baseline	Otu0098 (Bacteroides)	0.4360	0.0297
OTU	baseline	Otu0104 (Clostridium IV)	0.6063	0.0099
OTU	baseline	Otu0107 (Clostridium XlVa)	0.2639	0.0396
OTU	baseline	Otu0110 (Ruminococcus)	0.6342	0.0198
OTU	baseline	Otu0134 (Clostridiales unclassified)	0.2965	0.0198
OTU	baseline	Otu0154 (Ruminococcaceae unclassified)	0.3465	0.0495
OTU	baseline	Otu0164 (Clostridium XVIII)	0.5097	0.0198
OTU	baseline	Otu0171 (Bacteria unclassified)	0.2629	0.0396
OTU	baseline	Otu0202 (Lachnospiraceae unclassified)	0.2384	0.0198
OTU	baseline	Otu0217 (Intestinimonas)	0.5363	0.0099
OTU	baseline	Otu0234 (Rhodospirillaceae unclassified)	0.2094	0.0495
OTU	baseline	Otu0249 (Rhodospirillaceae unclassified)	0.2297	0.0495
OTU	baseline	Otu0318 (Clostridium IV)	0.3806	0.0198
OTU	baseline	Otu0377 (Ruminococcaceae unclassified)	0.5025	0.0099
OTU	baseline	Otu0381 (Clostridiales unclassified)	0.3233	0.0396
OTU	baseline	Otu0412 (Ruminococcaceae unclassified)	0.3298	0.0396
OTU	baseline	Otu0429 (Bacteria unclassified)	0.1397	0.0396
OTU	baseline	Otu0437 (Clostridium IV)	0.3311	0.0297
OTU	baseline	Otu0444 (Lachnospiraceae unclassified)	0.3098	0.0396
OTU	baseline	Otu0455 (Ruminococcaceae unclassified)	0.4739	0.0099
OTU	baseline	Otu0513 (Anaerotruncus)	0.3573	0.0297
OTU	baseline	Otu0575 (Clostridiales unclassified)	0.6427	0.0099
OTU	baseline	Otu0586 (Ruminococcaceae unclassified)	0.2649	0.0198
OTU	baseline	Otu0625 (Clostridiales unclassified)	0.4693	0.0099
OTU	followup	Otu0003 (Roseburia)	0.5975	0.0198
OTU	followup	Otu0024 (Blautia)	0.6578	0.0198
OTU	followup	Otu0036 (Roseburia)	0.4256	0.0297
OTU	followup	Otu0052 (Ruminococcaceae unclassified)	0.3710	0.0198
OTU	followup	Otu0070 (Lachnospiraceae unclassified)	0.3858	0.0495
OTU	followup	Otu0073 (Ruminococcaceae unclassified)	0.7037	0.0099
OTU	followup	Otu0074 (Clostridium XlVa)	0.9225	0.0099
OTU	followup	Otu0083 (Butyricicoccus)	0.4362	0.0396
OTU	followup	Otu0104 (Clostridium IV)	0.9145	0.0099
OTU	followup	Otu0105 (Oscillibacter)	0.4399	0.0099
OTU	followup	Otu0109 (Ruminococcus)	0.9673	0.0099
OTU	followup	Otu0129 (Clostridium XlVa)	0.3162	0.0396
OTU	followup	Otu0131 (Bacteroides)	0.3433	0.0198
OTU	followup	Otu0167 (Oscillibacter)	0.5682	0.0396
OTU	followup	Otu0177 (Desulfovibrio)	0.2197	0.0297
OTU	followup	Otu0207 (Bacteria unclassified)	0.3231	0.0396
OTU	followup	Otu0288 (Clostridium XlVa)	0.2372	0.0297
OTU	followup	Otu0363 (Lactobacillus)	0.1771	0.0495
OTU	followup	Otu0365 (Clostridiales unclassified)	0.2979	0.0396
OTU	followup	Otu0377 (Ruminococcaceae unclassified)	0.2592	0.0495
OTU	followup	Otu0379 (Alistipes)	0.2880	0.0198
OTU	followup	Otu0411 (Puniceicoccaceae unclassified)	0.3351	0.0198
OTU	followup	Otu0433 (Clostridiales unclassified)	0.3092	0.0495
OTU	followup	Otu0464 (Lactobacillus)	0.7432	0.0099
OTU	followup	Otu0468 (Faecalibacterium)	0.2439	0.0495
OTU	followup	Otu0469 (Lachnospiraceae unclassified)	0.2551	0.0297
OTU	followup	Otu0524 (Clostridiales unclassified)	0.5059	0.0099
OTU	followup	Otu0527 (Erysipelotrichaceae unclassified)	0.2762	0.0396
OTU	followup	Otu0631 (Clostridiales unclassified)	0.2584	0.0198

C. DESeq2, model: Rome III score + BMI + PD : subject + timepoint * PD, leave-one-out loop run 62 times (list of taxa with mean adjusted *p* < 0.05)

Level	Contrast	Taxon	Mean log2 fold change	SD for log2 fold change	Mean adjusted *p*-value	SD for adjusted *p*-value
Family	PD vs C, baseline	Porphyromonadaceae	-2.9176	0.1300	0.0031	0.0006
Family	PD vs C, baseline	Prevotellaceae	-7.4107	0.2141	0.0018	0.0006
Family	PD vs C, baseline	Veillonellaceae	-6.5409	0.9537	0.0164	0.1259
Family	PD vs C, followup	Porphyromonadaceae	-2.7642	0.1283	0.0058	0.0011
Family	PD vs C, followup	Prevotellaceae	-7.4819	0.2080	0.0015	0.0006
Family	PD vs C, followup	Veillonellaceae	-7.0958	0.9588	0.0162	0.1268
Family	Rome III 9-15 sum score	Bifidobacteriaceae	0.1881	0.0040	0.0002	0.0002
Family	BMI	Clostridiaceae_1	0.7520	0.0206	0.0000	0.0000
Genus	PD vs C, baseline	Clostridium_XlVa	5.3209	0.1209	0.0000	0.0000
Genus	PD vs C, baseline	Clostridium_XVIII	-7.2388	1.1429	0.0428	0.1221
Genus	PD vs C, baseline	Dialister	-11.3419	1.4701	0.0161	0.1267
Genus	PD vs C, baseline	Prevotella	-6.1119	0.1148	0.0417	0.0085
Genus	PD vs C, baseline	Romboutsia	6.5350	0.4349	0.0294	0.0064
Genus	PD vs C, followup	Clostridium_IV	2.7394	0.0838	0.0489	0.0126
Genus	PD vs C, followup	Clostridium_XlVa	5.0995	0.1180	0.0000	0.0000
Genus	PD vs C, followup	Dialister	-11.0257	1.4695	0.0161	0.1265
Genus	PD vs C, followup	Prevotella	-6.2879	0.1085	0.0432	0.0044
Genus	PD vs C, followup	Romboutsia	6.3534	0.4442	0.0431	0.0064
Genus	Rome III 9-15 sum score	Bifidobacterium	0.1903	0.0050	0.0006	0.0011
Genus	BMI	Clostridium_sensu_stricto	0.7635	0.0198	0.0000	0.0000
Genus	BMI	Coprococcus	-0.3734	0.0126	0.0051	0.0046
OTU	PD vs C, baseline	Otu0062 (Blautia)	-4.3308	0.1274	0.0485	0.0076
OTU	PD vs C, baseline	Otu0264 (Butyricimonas)	-9.3180	0.0584	0.0470	0.0064
OTU	BMI	Otu0051 (Clostridium sensu stricto)	0.6295	0.0186	0.0000	0.0001
OTU	BMI	Otu0582 (Actinomyces)	-0.2213	0.0164	0.0192	0.1256

Table S5. Model significances for random forest comparisons, PD patients/control subjects

Level	Timepoint	Actual Out of Box error	Out of Box error for randomly permuted data	*p*-value for model significance
OTU	baseline	0.375	0.516	0.001
OTU	follow-up	0.352	0.516	< 0.001
Genus	baseline	0.430	0.508	0.045
Genus	follow-up	0.328	0.516	< 0.001
Family	baseline	0.391	0.516	0.005
Family	follow-up	0.430	0.516	0.046

Table S6. Differential abundance comparison results for disease progression (within the PD patient group)

A. ANCOM (list of all differentially abundant taxa)

Level	Timepoint	Taxon
OTU	followup	Otu0148 (Bifidobacterium)
OTU	followup	Otu0327 (Lachnospiraceae unclassified)

B. Random forests (list of taxa with $p < 0.05$ for mean decrease in Gini)

Level	Timepoint	Taxon	Mean decrease in Gini	*p*-value for mean decrease in Gini
Family	baseline	Eubacteriaceae	0.9975	0.0297
Family	baseline	Streptococcaceae	1.7012	0.0099
Family	baseline	Synergistaceae	0.8288	0.0495
Family	followup	Actinomycetaceae	1.3350	0.0297
Family	followup	Anaeroplasmataceae	0.9584	0.0099
Genus	baseline	Anaerotruncus	0.7630	0.0396
Genus	baseline	Cloacibacillus	0.3297	0.0396
Genus	baseline	Eubacterium	0.6376	0.0099
Genus	baseline	Intestinimonas	0.7467	0.0297
Genus	followup	Actinomyces	0.6551	0.0396
Genus	followup	Asteroleplasma	0.2929	0.0495
Genus	followup	Gordonibacter	0.2688	0.0198
Genus	followup	Rothia	0.3762	0.0297

Genus	followup	Solobacterium	0.3533	0.0099
OTU	baseline	Otu0002 (Ruminococcaceae unclassified)	0.2327	0.0396
OTU	baseline	Otu0038 (Clostridiales unclassified)	0.2875	0.0099
OTU	baseline	Otu0049 (Ruminococcaceae unclassified)	0.2246	0.0396
OTU	baseline	Otu0103 (Ruminococcaceae unclassified)	0.1077	0.0198
OTU	baseline	Otu0111 (Streptococcus)	0.2119	0.0495
OTU	baseline	Otu0118 (Ruminococcaceae unclassified)	0.4181	0.0099
OTU	baseline	Otu0128 (Ruminococcus)	0.2082	0.0396
OTU	baseline	Otu0144 (Ruminococcus2)	0.2074	0.0297
OTU	baseline	Otu0149 (Ruminococcaceae unclassified)	0.1681	0.0198
OTU	baseline	Otu0166 (Ruminococcaceae unclassified)	0.2732	0.0198
OTU	baseline	Otu0171 (Bacteria unclassified)	0.2565	0.0198
OTU	baseline	Otu0191 (Rhodospirillales unclassified)	0.3109	0.0198
OTU	baseline	Otu0257 (Streptococcus)	0.3098	0.0198
OTU	baseline	Otu0318 (Clostridium IV)	0.1754	0.0297
OTU	baseline	Otu0327 (Lachnospiraceae unclassified)	0.1638	0.0495
OTU	baseline	Otu0343 (Mollicutes unclassified)	0.0818	0.0495
OTU	baseline	Otu0365 (Clostridiales unclassified)	0.2320	0.0297
OTU	baseline	Otu0377 (Ruminococcaceae unclassified)	0.3272	0.0099
OTU	baseline	Otu0443 (Clostridiales unclassified)	0.2697	0.0099
OTU	baseline	Otu0497 (Clostridiales unclassified)	0.1116	0.0396
OTU	baseline	Otu0514 (Clostridiales unclassified)	0.5542	0.0099
OTU	baseline	Otu0534 (Ruminococcaceae unclassified)	0.1268	0.0396
OTU	followup	Otu0049 (Ruminococcaceae unclassified)	0.1822	0.0297
OTU	followup	Otu0084 (Clostridium IV)	0.4822	0.0099
OTU	followup	Otu0115 (Lachnospiraceae unclassified)	0.5202	0.0099
OTU	followup	Otu0118 (Ruminococcaceae unclassified)	0.2262	0.0396
OTU	followup	Otu0148 (Bifidobacterium)	0.2823	0.0099
OTU	followup	Otu0166 (Ruminococcaceae unclassified)	0.1455	0.0396
OTU	followup	Otu0168 (Lachnospiraceae unclassified)	0.1419	0.0396
OTU	followup	Otu0208 (Anaerotruncus)	0.1374	0.0495
OTU	followup	Otu0241 (Clostridiales unclassified)	0.1739	0.0396
OTU	followup	Otu0307 (Firmicutes unclassified)	0.4191	0.0099
OTU	followup	Otu0327 (Lachnospiraceae unclassified)	0.4636	0.0099
OTU	followup	Otu0334 (Ruminococcus)	0.1260	0.0198
OTU	followup	Otu0350 (Deltaproteobacteria unclassified)	0.2977	0.0297
OTU	followup	Otu0413 (Ruminococcaceae unclassified)	0.4714	0.0099
OTU	followup	Otu0513 (Anaerotruncus)	0.1884	0.0297

C. DESeq2, model: COMT inhibitor use + Progression, run separately for baseline and follow-up (list of taxa with adjusted p < 0.05)

Level	Timepoint	Taxon	Variable	Log2 fold change	SE for log2 fold change	*p*-value	Adjusted *p*-value
Family	baseline	Streptococcaceae	Progression	2.2304	0.5388	0.0000	0.0013
Family	baseline	Enterococcaceae	COMT inhibitor	14.6724	4.6902	0.0018	0.0351
Family	baseline	Peptostreptococcaceae	COMT inhibitor	-2.9684	0.9556	0.0019	0.0351
Family	followup	Anaeroplasmataceae	Progression	7.5962	1.3797	0.0000	0.0000
Family	followup	Oxalobacteraceae	Progression	-2.7586	0.9027	0.0022	0.0277
Family	followup	Prevotellaceae	Progression	-4.8780	1.0187	0.0000	0.0000
Family	followup	Verrucomicrobiaceae	Progression	-2.2722	0.7763	0.0034	0.0316
Family	followup	Lachnospiraceae	COMT inhibitor	-1.5118	0.3315	0.0000	0.0001
Family	followup	Lactobacillaceae	COMT inhibitor	6.2970	0.9224	0.0000	0.0000
Family	followup	Ruminococcaceae	COMT inhibitor	-1.0925	0.2406	0.0000	0.0001
Genus	baseline	Bacteroides	Progression	-1.4539	0.3999	0.0003	0.0062
Genus	baseline	Butyrivibrio	Progression	-6.9646	1.5433	0.0000	0.0003
Genus	baseline	Prevotella	Progression	-5.4929	0.9468	0.0000	0.0000
Genus	baseline	Streptococcus	Progression	2.3720	0.5761	0.0000	0.0011
Genus	baseline	Acidaminococcus	COMT inhibitor	7.0703	2.2314	0.0015	0.0345
Genus	baseline	Eisenbergiella	COMT inhibitor	-4.7103	1.3328	0.0004	0.0184
Genus	baseline	Enterococcus	COMT inhibitor	15.3872	4.6896	0.0010	0.0310
Genus	baseline	Megasphaera	COMT inhibitor	12.2362	2.1570	0.0000	0.0000
Genus	followup	Acidaminococcus	Progression	-8.2094	2.1291	0.0001	0.0026
Genus	followup	Akkermansia	Progression	-2.4095	0.7950	0.0024	0.0314
Genus	followup	Asteroleplasma	Progression	8.2036	1.5868	0.0000	0.0000
Genus	followup	Coprobacter	Progression	-5.0996	1.2890	0.0001	0.0023
Genus	followup	Desulfovibrio	Progression	-4.6355	1.3998	0.0009	0.0139
Genus	followup	Eisenbergiella	Progression	2.6567	0.7341	0.0003	0.0053
Genus	followup	Prevotella	Progression	-5.8420	1.0593	0.0000	0.0000
Genus	followup	Anaerostipes	COMT inhibitor	-4.4211	0.7090	0.0000	0.0000
Genus	followup	Bifidobacterium	COMT inhibitor	1.9306	0.6254	0.0020	0.0124
Genus	followup	Blautia	COMT inhibitor	-2.1822	0.4312	0.0000	0.0000
Genus	followup	Clostridium_XlVb	COMT inhibitor	1.8533	0.6484	0.0043	0.0229
Genus	followup	Faecalibacterium	COMT inhibitor	-1.6108	0.4876	0.0010	0.0068

Genus	followup	Fusicatenibacter	COMT inhibitor	-2.3327	0.5848	0.0001	0.0007
Genus	followup	Lactobacillus	COMT inhibitor	6.3846	0.9494	0.0000	0.0000
Genus	followup	Veillonella	COMT inhibitor	3.0341	0.9146	0.0009	0.0068
OTU	baseline	Otu0016	Progression	2.8073	0.8665	0.0012	0.0446
OTU	baseline	Otu0042	Progression	-5.8168	1.2271	0.0000	0.0003
OTU	baseline	Otu0047	Progression	-6.6654	1.5384	0.0000	0.0012
OTU	baseline	Otu0055	Progression	-7.3650	1.7079	0.0000	0.0012
OTU	baseline	Otu0085	Progression	-11.3475	1.6944	0.0000	0.0000
OTU	baseline	Otu0093	Progression	-6.8446	1.7265	0.0001	0.0039
OTU	baseline	Otu0111	Progression	2.4497	0.6455	0.0001	0.0069
OTU	baseline	Otu0115	Progression	-3.7606	1.0709	0.0004	0.0185
OTU	baseline	Otu0222	Progression	-8.6971	2.0797	0.0000	0.0018
OTU	baseline	Otu0268	Progression	-24.0220	2.3727	0.0000	0.0000
OTU	baseline	Otu0011	COMT inhibitor	-5.2452	1.3947	0.0002	0.0043
OTU	baseline	Otu0019	COMT inhibitor	4.7897	1.4594	0.0010	0.0176
OTU	baseline	Otu0024	COMT inhibitor	-2.6349	0.6991	0.0002	0.0043
OTU	baseline	Otu0039	COMT inhibitor	-6.9875	1.9935	0.0005	0.0102
OTU	baseline	Otu0059	COMT inhibitor	2.4123	0.8086	0.0029	0.0378
OTU	baseline	Otu0069	COMT inhibitor	-10.1463	2.3166	0.0000	0.0005
OTU	baseline	Otu0085	COMT inhibitor	12.7473	2.3013	0.0000	0.0000
OTU	baseline	Otu0099	COMT inhibitor	8.0118	1.6987	0.0000	0.0001
OTU	baseline	Otu0101	COMT inhibitor	-8.3667	2.6589	0.0017	0.0257
OTU	baseline	Otu0104	COMT inhibitor	-7.3597	2.1517	0.0006	0.0127
OTU	baseline	Otu0110	COMT inhibitor	-9.1313	2.2677	0.0001	0.0020
OTU	baseline	Otu0143	COMT inhibitor	-4.3970	1.3323	0.0010	0.0173
OTU	baseline	Otu0155	COMT inhibitor	-23.0171	2.2671	0.0000	0.0000
OTU	baseline	Otu0176	COMT inhibitor	-22.2416	3.3225	0.0000	0.0000
OTU	baseline	Otu0186	COMT inhibitor	-6.6298	1.4939	0.0000	0.0005
OTU	baseline	Otu0187	COMT inhibitor	-23.7646	2.6369	0.0000	0.0000
OTU	baseline	Otu0198	COMT inhibitor	8.1260	2.6027	0.0018	0.0268
OTU	baseline	Otu0202	COMT inhibitor	5.7241	1.1807	0.0000	0.0001
OTU	baseline	Otu0241	COMT inhibitor	-3.3951	1.0180	0.0009	0.0161
OTU	baseline	Otu0242	COMT inhibitor	13.5512	3.0746	0.0000	0.0005
OTU	baseline	Otu0247	COMT inhibitor	-7.3819	2.4385	0.0025	0.0345
OTU	baseline	Otu0294	COMT inhibitor	2.2845	0.5957	0.0001	0.0041
OTU	baseline	Otu0365	COMT inhibitor	-5.5055	1.6124	0.0006	0.0127
OTU	baseline	Otu0373	COMT inhibitor	7.0607	2.1663	0.0011	0.0182
OTU	baseline	Otu0376	COMT inhibitor	2.5103	0.6663	0.0002	0.0043
OTU	baseline	Otu0495	COMT inhibitor	-5.0187	1.3499	0.0002	0.0048
OTU	baseline	Otu0501	COMT inhibitor	12.3303	4.2170	0.0035	0.0442
OTU	baseline	Otu0631	COMT inhibitor	-5.3586	1.7731	0.0025	0.0345
OTU	followup	Otu0013	Progression	-2.4057	0.7950	0.0025	0.0319
OTU	followup	Otu0019	Progression	-5.9334	1.2445	0.0000	0.0001
OTU	followup	Otu0042	Progression	-6.8500	1.2983	0.0000	0.0000
OTU	followup	Otu0063	Progression	7.8017	1.8072	0.0000	0.0006
OTU	followup	Otu0064	Progression	-1.9876	0.6878	0.0039	0.0477
OTU	followup	Otu0068	Progression	-6.2879	1.5805	0.0001	0.0017
OTU	followup	Otu0082	Progression	8.6731	1.6667	0.0000	0.0000
OTU	followup	Otu0084	Progression	2.1085	0.4881	0.0000	0.0006
OTU	followup	Otu0101	Progression	-7.5085	2.1010	0.0004	0.0076
OTU	followup	Otu0110	Progression	-4.8578	1.5237	0.0014	0.0200
OTU	followup	Otu0115	Progression	3.5482	1.0654	0.0009	0.0143
OTU	followup	Otu0119	Progression	5.8315	1.7543	0.0009	0.0143
OTU	followup	Otu0126	Progression	-3.9484	1.2698	0.0019	0.0251
OTU	followup	Otu0143	Progression	2.7370	0.7704	0.0004	0.0077
OTU	followup	Otu0222	Progression	-23.9397	2.5737	0.0000	0.0000
OTU	followup	Otu0233	Progression	-9.9412	2.3033	0.0000	0.0006
OTU	followup	Otu0234	Progression	-9.8729	2.3076	0.0000	0.0006
OTU	followup	Otu0241	Progression	2.5117	0.7870	0.0014	0.0200
OTU	followup	Otu0242	Progression	10.2636	2.4860	0.0000	0.0011
OTU	followup	Otu0268	Progression	-7.3194	2.1005	0.0005	0.0093
OTU	followup	Otu0276	Progression	-9.9660	2.4845	0.0001	0.0016
OTU	followup	Otu0290	Progression	-9.6014	1.8404	0.0000	0.0000
OTU	followup	Otu0327	Progression	5.9307	1.8058	0.0010	0.0157
OTU	followup	Otu0352	Progression	-4.8888	1.2836	0.0001	0.0032
OTU	followup	Otu0372	Progression	-8.2235	1.8753	0.0000	0.0006
OTU	followup	Otu0380	Progression	4.7922	1.4416	0.0009	0.0143
OTU	followup	Otu0008	COMT inhibitor	-2.1117	0.5889	0.0003	0.0055
OTU	followup	Otu0009	COMT inhibitor	2.4230	0.6906	0.0005	0.0067
OTU	followup	Otu0024	COMT inhibitor	-3.2557	0.5675	0.0000	0.0000
OTU	followup	Otu0032	COMT inhibitor	-2.4835	0.7887	0.0016	0.0200
OTU	followup	Otu0044	COMT inhibitor	-3.3051	0.5821	0.0000	0.0000
OTU	followup	Otu0050	COMT inhibitor	-2.2623	0.7296	0.0019	0.0212

OTU	followup	Otu0059	COMT inhibitor	3.0547	0.6541	0.0000	0.0001
OTU	followup	Otu0062	COMT inhibitor	-1.7837	0.5845	0.0023	0.0242
OTU	followup	Otu0063	COMT inhibitor	-5.7782	2.0539	0.0049	0.0429
OTU	followup	Otu0079	COMT inhibitor	-2.8382	0.9852	0.0040	0.0370
OTU	followup	Otu0085	COMT inhibitor	8.3285	2.2797	0.0003	0.0046
OTU	followup	Otu0091	COMT inhibitor	-6.8719	2.3628	0.0036	0.0348
OTU	followup	Otu0102	COMT inhibitor	-3.3131	0.6084	0.0000	0.0000
OTU	followup	Otu0108	COMT inhibitor	3.4233	0.9921	0.0006	0.0080
OTU	followup	Otu0120	COMT inhibitor	7.9010	1.7697	0.0000	0.0003
OTU	followup	Otu0134	COMT inhibitor	-8.8668	2.1189	0.0000	0.0009
OTU	followup	Otu0144	COMT inhibitor	-4.8371	0.9568	0.0000	0.0000
OTU	followup	Otu0180	COMT inhibitor	3.5899	1.1134	0.0013	0.0168
OTU	followup	Otu0193	COMT inhibitor	-3.0498	0.8514	0.0003	0.0055
OTU	followup	Otu0198	COMT inhibitor	-8.6485	2.8491	0.0024	0.0242
OTU	followup	Otu0204	COMT inhibitor	4.7010	0.9863	0.0000	0.0001
OTU	followup	Otu0211	COMT inhibitor	10.5372	2.7682	0.0001	0.0031
OTU	followup	Otu0220	COMT inhibitor	-4.6801	1.6469	0.0045	0.0408
OTU	followup	Otu0222	COMT inhibitor	-22.4018	2.8757	0.0000	0.0000
OTU	followup	Otu0230	COMT inhibitor	-7.1293	2.3432	0.0023	0.0242
OTU	followup	Otu0233	COMT inhibitor	7.8927	2.5283	0.0018	0.0203
OTU	followup	Otu0234	COMT inhibitor	9.6217	2.5323	0.0001	0.0031
OTU	followup	Otu0235	COMT inhibitor	9.2444	2.4799	0.0002	0.0038
OTU	followup	Otu0242	COMT inhibitor	10.5127	2.7636	0.0001	0.0031
OTU	followup	Otu0267	COMT inhibitor	8.0190	1.5866	0.0000	0.0000
OTU	followup	Otu0272	COMT inhibitor	1.9965	0.5419	0.0002	0.0043
OTU	followup	Otu0276	COMT inhibitor	10.4602	2.7321	0.0001	0.0031
OTU	followup	Otu0282	COMT inhibitor	-2.8771	0.9151	0.0017	0.0200
OTU	followup	Otu0294	COMT inhibitor	1.7015	0.5088	0.0008	0.0114
OTU	followup	Otu0316	COMT inhibitor	5.4754	1.0619	0.0000	0.0000
OTU	followup	Otu0320	COMT inhibitor	2.7021	0.7130	0.0002	0.0031
OTU	followup	Otu0325	COMT inhibitor	-23.1865	2.6674	0.0000	0.0000
OTU	followup	Otu0342	COMT inhibitor	3.3322	1.1290	0.0032	0.0311
OTU	followup	Otu0363	COMT inhibitor	4.8029	1.5351	0.0018	0.0203
OTU	followup	Otu0372	COMT inhibitor	5.7476	2.0451	0.0049	0.0429
OTU	followup	Otu0380	COMT inhibitor	5.0476	1.5999	0.0016	0.0200
OTU	followup	Otu0464	COMT inhibitor	6.7036	1.6392	0.0000	0.0012
OTU	followup	Otu0634	COMT inhibitor	4.9226	1.4008	0.0004	0.0067

Table S7. Model significances for random forest comparisons, progressed/stable PD patients

Level	Timepoint	Actual Out of Box error	Out of Box error for randomly permuted data	*p*-value for model significance
Family	baseline	0.286	0.286	0.232
Family	follow-up	0.321	0.286	0.798
Genus	baseline	0.304	0.286	0.660
Genus	follow-up	0.321	0.286	0.910
OTU	baseline	0.304	0.286	0.728
OTU	follow-up	0.268	0.286	0.044

Table S8. Differential abundance comparison results for disease phenotype (TD vs PIGD; within the PD patient group)

A. ANCOM (list of all differentially abundant taxa)

Level	Timepoint	Taxon
OTU	baseline	Otu0048
OTU	baseline	Otu0170

B. DESeq2, run separately for baseline and follow-up (list of taxa with adjusted *p* < 0.05)

Level	Timepoint	Taxon	Log2 fold change	SE for log2 fold change	*p*-value	Adjusted *p*-value
Family	baseline	Anaeroplasmataceae	-5.4442	1.4962	0.0003	0.0099
Family	baseline	Enterobacteriaceae	-2.2588	0.7086	0.0014	0.0258
Family	followup	Anaeroplasmataceae	-6.3492	1.5701	0.0001	0.0019
Genus	baseline	Asteroleplasma	-5.2317	1.6364	0.0014	0.0242
Genus	baseline	Butyrivibrio	4.5432	1.0290	0.0000	0.0009

Genus	baseline	Clostridium_sensu_stricto	-2.2195	0.6302	0.0004	0.0129
Genus	baseline	Eisenbergiella	-2.8297	0.8628	0.0010	0.0226
Genus	baseline	Gemmiger	1.5637	0.4452	0.0004	0.0129
Genus	followup	Asteroleplasma	-6.6690	1.7843	0.0002	0.0082
Genus	followup	Lactococcus	-3.1747	0.8414	0.0002	0.0082
OTU	baseline	Otu0012 (Bacteroides)	1.8238	0.5327	0.0006	0.0246
OTU	baseline	Otu0019 (Prevotella)	-3.0993	0.8683	0.0004	0.0160
OTU	baseline	Otu0031 (Escherichia/Shigella)	-2.6344	0.7969	0.0009	0.0273
OTU	baseline	Otu0047 (Butyrivibrio)	4.5339	1.0628	0.0000	0.0024
OTU	baseline	Otu0063 (Acidaminococcaceae unclassified)	6.4045	1.7359	0.0002	0.0134
OTU	baseline	Otu0082 (Asteroleplasma)	-5.5362	1.6691	0.0009	0.0273
OTU	baseline	Otu0093 (Clostridiales unclassified)	6.3962	1.2701	0.0000	0.0001
OTU	baseline	Otu0115 (Lachnospiraceae unclassified)	5.4087	0.9346	0.0000	0.0000
OTU	baseline	Otu0178 (Bacteria unclassified)	7.0462	1.8853	0.0002	0.0133
OTU	baseline	Otu0197 (Deltaproteobacteria unclassified)	-3.6637	1.1336	0.0012	0.0314
OTU	baseline	Otu0242 (Bifidobacterium)	-8.4058	2.5419	0.0009	0.0273
OTU	baseline	Otu0284 (Clostridiales unclassified)	6.3742	1.6772	0.0001	0.0129
OTU	baseline	Otu0360 (Lachnospiraceae unclassified)	-1.4431	0.4021	0.0003	0.0160
OTU	baseline	Otu0391 (Lachnospiraceae unclassified)	4.5159	1.3712	0.0010	0.0273
OTU	followup	Otu0013 (Akkermansia)	2.2667	0.6439	0.0004	0.0154
OTU	followup	Otu0053 (Clostridium XlVa)	-1.7338	0.5510	0.0017	0.0394
OTU	followup	Otu0069 (Ruminococcaceae unclassified)	-4.4629	1.3742	0.0012	0.0379
OTU	followup	Otu0082 (Asteroleplasma)	-7.0719	1.8718	0.0002	0.0090
OTU	followup	Otu0085 (Porphyromonadaceae unclassified)	-5.3911	1.6922	0.0014	0.0380
OTU	followup	Otu0091 (Barnesiella)	-9.6673	1.8132	0.0000	0.0000
OTU	followup	Otu0112 (Ruminococcaceae unclassified)	5.5904	1.5038	0.0002	0.0090
OTU	followup	Otu0117 (Ruminococcaceae unclassified)	9.6231	1.6616	0.0000	0.0000
OTU	followup	Otu0174 (Clostridiales unclassified)	-3.7366	0.9992	0.0002	0.0090
OTU	followup	Otu0185 (Lachnospiraceae unclassified)	-2.4229	0.6878	0.0004	0.0154
OTU	followup	Otu0219 (Lachnospiraceae unclassified)	-7.3982	1.8968	0.0001	0.0069
OTU	followup	Otu0230 (Clostridia unclassified)	-10.1819	1.6834	0.0000	0.0000
OTU	followup	Otu0238 (Ruminococcaceae unclassified)	-6.9644	2.1916	0.0015	0.0380
OTU	followup	Otu0360 (Lachnospiraceae unclassified)	-2.0363	0.4695	0.0000	0.0013
OTU	followup	Otu0380 (Ruminococcaceae unclassified)	-4.4091	1.3826	0.0014	0.0380

R code for 'Gut microbiota in Parkinson's disease: Temporal stability and relations to disease progression'

Velma T. E. Aho

29 April 2019

R preamble

Load required packages and additional scripts:

```
library("knitr")
library("kableExtra")
library("phyloseq")
library("ggplot2")
library("grid")
library("gridExtra")
library("reshape2")
library("vegan")
library("devtools")
source("Inputs/relabund_barcharts_v4.5.R")
source("Inputs/taxaLevelCollapser_v5.R")
```

```r
# Figure widths:
library("measurements")
halfpage <- conv_unit(85, "mm", "inch")
fullpage <- conv_unit(170, "mm", "inch")
maxhi <- conv_unit(225, "mm", "inch")
```

Preliminary microbiota analyses: experimental controls and outlier samples

Import the full OTU data, including experimental controls (blanks). There were three types of these: extraction blanks (blanks included during the DNA extraction protocol without template DNA), PCR blanks (blanks included during PCR without template DNA), and lambda blanks (lambda phage DNA introduced to see if it could function as a carrier for potential contaminant bacterial DNA).

```r
# Metadata
basicmeta <- read.csv("Inputs/basicmeta.csv", row.names = 1)
basicmeta$Group <- factor(basicmeta$Group,
                        levels = c("extract_blank", "lambda_blank", "pcr_blank", "control",
                            "Parkinson"))

# 16s data
pd16s <- t(as.matrix(read.table("Inputs/pdfu_otutable.txt")))

# Taxonomy table
pdtax <- as.matrix(read.csv("Inputs/pdfu_taxtable.csv", sep = "\t", row.names = 1))

# Make a phyloseq object
pdfu_prelim <- phyloseq(otu_table(pd16s, taxa_are_rows = TRUE), tax_table(pdtax),
    sample_data(basicmeta))
```

Number of sequence reads

Plot summary data of read counts by group:

```r
read_sums <- data.frame(Reads = sample_sums(pdfu_prelim),
           Group = basicmeta$Group)

read_summary <- data.frame(levels(read_sums$Group),
                         aggregate(read_sums$Reads,
                                   by = list(read_sums$Group),
                                   FUN = function(x) cbind(mean(x), median(x), sd(x), min(x),
                                       max(x)))$x)
colnames(read_summary)<-c("Group", "mean", "median", "sd", "min", "max")

kable(read_summary, digits = 2)
```

Group	mean	median	sd	min	max
extract_blank	609.00	168.0	891.37	154	1946
lambda_blank	3276.33	114.0	14241.22	23	65423
pcr_blank	400.22	109.0	636.79	32	1875
control	72365.21	72908.5	34969.30	600	161050
Parkinson	73350.79	72032.0	36509.40	2201	262621

Most of the experimental control samples have very few reads. The following five have over 1000:

```r
kable(subset(read_sums, Group != "Parkinson" & Group != "control" & Reads > 1000))
```

	Reads	Group
Blank1	1946	extract_blank
Lambda20	65423	lambda_blank

	Reads	Group
Lambda515	1011	lambda_blank
PCRblank2	1875	pcr_blank
PCRblank5	1041	pcr_blank

There is also one patient sample with less than 1000 reads:

```
subset(read_sums, (Group == "Parkinson" | Group == "control") & Reads < 1000)
```

```
##        Reads   Group
## C00440   600 control
```

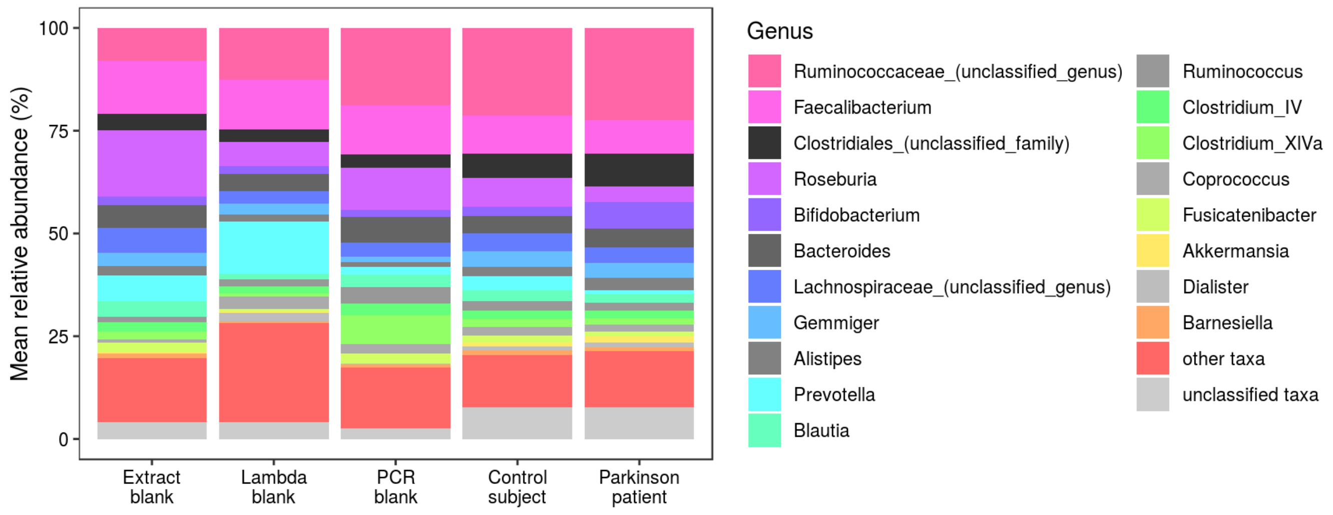
Trim all samples with under 1000 reads (and the paired sample of the low-read patient sample):

```
pdfu_prelim <- subset_samples(pdfu_prelim, sample_sums(pdfu_prelim)>1000 &
                                ((Type == "sample" & Subject != "C0044") | Type != "sample"))
```

Experimental controls

Quick relative abundance bar charts to check what the samples look like:

```
# genus level phyloseq object:
pdfu_prelim_gen <- collapseTaxLevel(pdfu_prelim, level = "Genus", maxUnclassifiedLevel = "Family")

relAbundChart(pdfu_prelim_gen, taxaCount = 20, byVariable = "Group") +
  theme_bw() +
  scale_x_discrete(labels = c("Extract\nblank", "Lambda\nblank", "PCR\nblank", "Control\nsubject",
      "Parkinson\npatient")) +
  theme(panel.grid = element_blank(),
        axis.text = element_text(color = "black"))
```

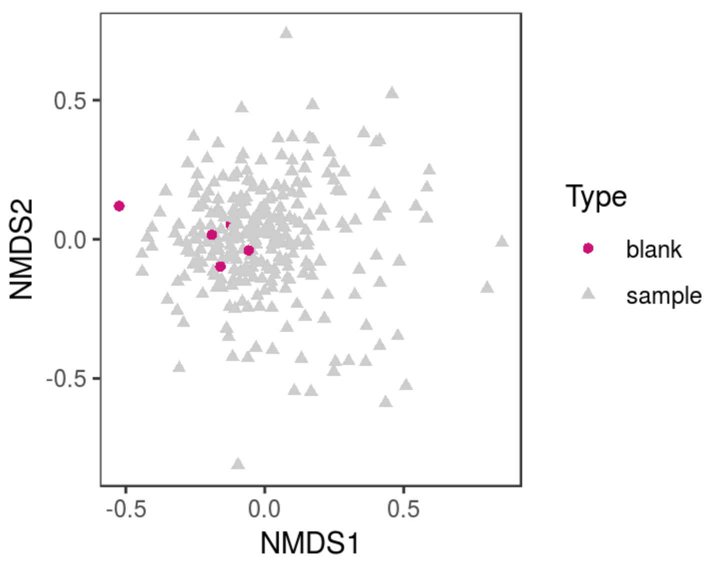


The remaining experimental control samples look very similar to the actual stool samples.

NMDS ordination (starting with the genus level) to further check where these samples fall in comparison to the actual samples:

```
pdfu_prelim_gen_R <- rarefy_even_depth(pdfu_prelim_gen, rngseed = 101125)

pdfu_prelim_gen_ord <- ordinate(pdfu_prelim_gen_R, "NMDS", "bray", try = 100)
```

```
plot_ordination(pdfu_prelim_gen_R, pdfu_prelim_gen_ord, type = "samples", color = "Type", shape =
    "Type") +
  theme_bw() +
  coord_fixed() +
  scale_color_manual(values = c("deeppink3", "gray80")) +
  theme(panel.grid = element_blank())
```



The ordination shows that the five experimental control samples with more than 1000 sequence reads have communities very similar to the stool samples. These samples represented a small subset of all experimental control samples.

```
kable(cbind(All_samples = table(basicmeta$Group), Samples_over_1000 =
    table(sample_data(pdfu_prelim)$Group)), col.names = c("All samples", "Samples with over 1000
    reads"))
```

	All samples	Samples with over 1000 reads
extract_blank	4	1
lambda_blank	21	2
pcr_blank	9	2
control	130	128
Parkinson	136	136

We concluded that the microbes detected in these blanks result from cross-contamination in the laboratory, and are unlikely to suggest an issue with contamination, since the bacteria detected in them represent taxa commonly detected in stool. Most of the experimental control samples have very few sequence reads (less than 500).

Suspicious patient samples

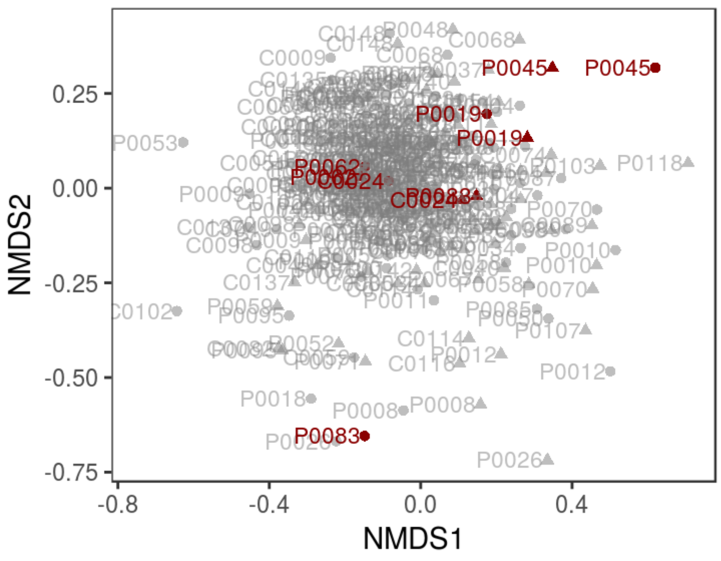
Remove the blanks, and continue with preliminary analysis of actual patient samples:

```
pdfu_prelim_gen_trim <- subset_samples(pdfu_prelim_gen, Type != "blank")

pdfu_prelim_gen_trim_R <- rarefy_even_depth(pdfu_prelim_gen_trim, rngseed = 124659)

pdfu_prelim_gen_trim_ord <- ordinate(pdfu_prelim_gen_trim_R, "NMDS", "bray", try = 400)
```

A small set of samples had been flagged suspicious by the clinician for clinical or sampling reasons. Make a plot with these subjects' both samples colored:

```
suspicious_ord <- plot_ordination(pdfu_prelim_gen_trim_R, pdfu_prelim_gen_trim_ord,
                       type = "samples", justDF = TRUE)

# Reorder to get the "suspicious" points on top:
suspicious_ord <- suspicious_ord[order(suspicious_ord$Suspicious), ]

ggplot(suspicious_ord, aes(x = NMDS1, y = NMDS2, color = Suspicious, alpha = Suspicious, shape =
    Timepoint)) +
  geom_point() +
  theme_bw() +
  coord_fixed() +
  scale_color_manual(values = c("gray50", "darkred")) +
```

```r
  scale_alpha_manual(values = c(0.5, 1)) +
  geom_text(aes(label = Subject, x = NMDS1, y = NMDS2), size = 3, nudge_x = -0.1) +
  theme(legend.position = "none", panel.grid = element_blank())
```



Based on this plots & clinical notes:

- C24 is OK (notes: follow-up sample tube leaked fluid during transport/storage, but the microbial communities look fairly similar in the two samples and overall not different from other samples in the plot)

- P19 is OK (notes: follow-up sample tube leaked fluid during transport/storage, but the microbial communities look fairly similar in the two samples and overall not different from other samples in the plot)

- P62 is OK (subject was excluded from pilot because of nose polyps; these are not likely to affect gut microbiota notably, and the microbial communities look similar to other samples in the plot)

- P45 is outside the main cluster, and has notes about sampling problems (patient placed samples in trash can and they were recovered some time later). Exclude.

- P83 has a large difference between samples, and has notes that mention multiple antibiotic regimens between time points. Exclude.

- Additionally, P26 has nothing special in notes, but is a PD patient and very different from all other samples. Exclude as an outlier.

One could argue for there being other samples that are as outlierish as these, but the two other samples aside from P26 have additional reasons for exclusion, and P26 seems very distinct from all other samples, so excluding it seems like the safest choice to avoid outlier effects between the PD and control groups.

To summarize:
Keep: C24, P19, P62
Exclude: P26, P45, P83

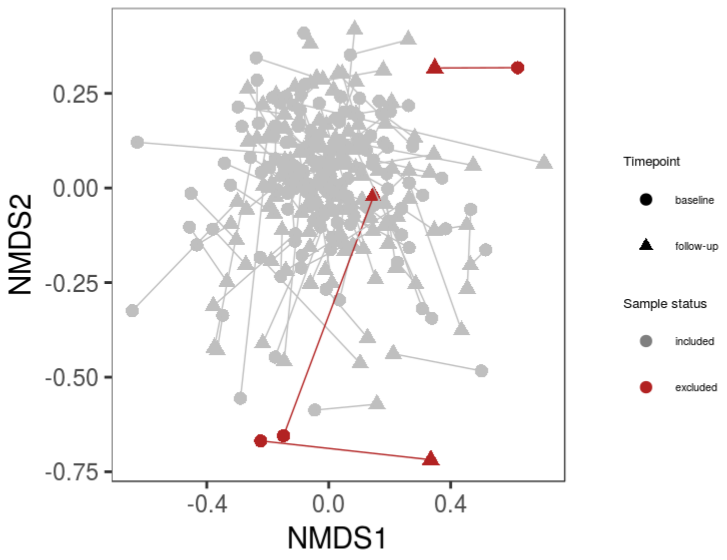
Redraw the plot with these samples highlighted:

```r
# Adjust the "Suspicious" variable:
keep <- paste(rep(c("C0024", "P0019", "P0062", "P0118"), each = 2), c("O", "N"), sep = "")
drop <- paste(rep(c("P0083", "P0026", "P0045"), each = 2), c("O", "N"), sep = "")

sample_data(pdfu_prelim_gen_trim_R)[keep, "Suspicious"] <- 1
sample_data(pdfu_prelim_gen_trim_R)[drop, "Suspicious"] <- "check"
levels(sample_data(pdfu_prelim_gen_trim_R)$Suspicious) <- c("included", "excluded")

# Plot again:
ordGenSuspicious <- plot_ordination(pdfu_prelim_gen_trim_R, pdfu_prelim_gen_trim_ord, type =
    "samples", color = "Suspicious", shape = "Timepoint") +
  theme_bw() +
  geom_point(size = 2) +
  scale_color_manual(values = c("gray75", "firebrick")) +
  geom_polygon(aes(group = Subject), size = 0.25) +
  theme(legend.position  =  "none",
        panel.grid = element_blank(),
        panel.border = element_rect(fill = NA, size = 0.25))

# Make a nice legend:

g_legend <- function(a.gplot){
  tmp <- ggplot_gtable(ggplot_build(a.gplot))
  leg <- which(sapply(tmp$grobs, function(x) x$name) == "guide-box")
  legend <- tmp$grobs[[leg]]
  legend
}

ordLeg <- g_legend(plot_ordination(pdfu_prelim_gen_trim_R, pdfu_prelim_gen_trim_ord, type =
    "samples", color = "Suspicious", shape = "Timepoint") +
    theme_bw(base_size = 5) +
    scale_color_manual(values = c("gray50", "firebrick"), name = "Sample status") +
    scale_shape_manual(values = c(19, 17), labels = c("baseline", "follow-up")))
```

Figure 1

Plot of samples excluded based on preliminary microbiota comparisons:

```
grid.arrange(ordGenSuspicious, ordLeg, nrow = 1, widths = c(8, 2.5))
```



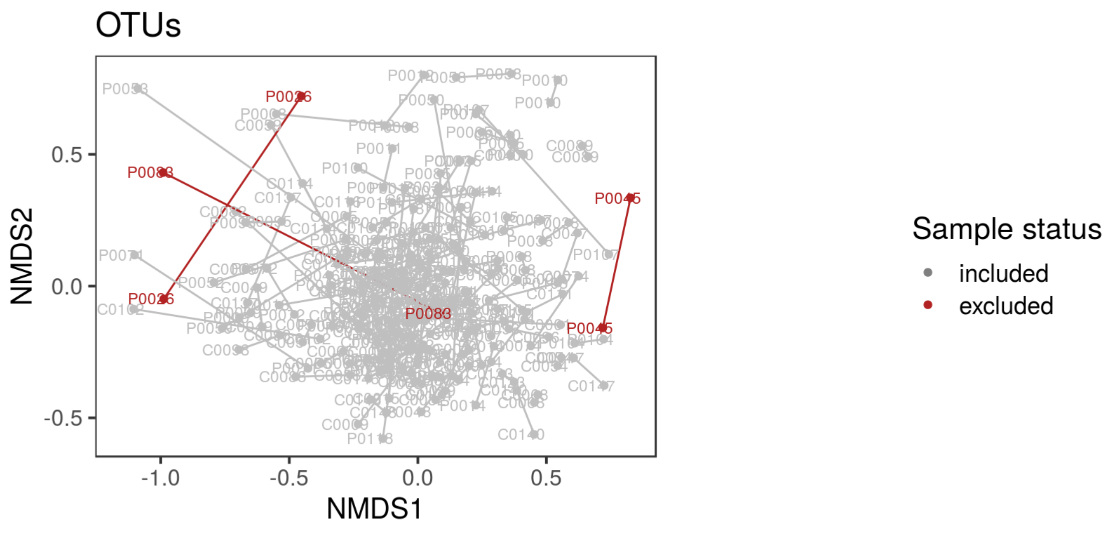
```
# Export to pdf
# (readjusting some graphical parameters)

ordGenSuspicious <- plot_ordination(pdfu_prelim_gen_trim_R, pdfu_prelim_gen_trim_ord, type =
    "samples", color = "Suspicious", shape = "Timepoint") +
  theme_bw(base_size = 8) +
  geom_point(size = 0.5) +
  coord_fixed() +
  scale_color_manual(values = c("gray75", "firebrick")) +
  geom_polygon(aes(group = Subject), size = 0.25) +
  theme(legend.position  =  "none",
        panel.grid = element_blank(),
        panel.border = element_rect(fill = NA, size = 0.25))

ordLeg <- g_legend(plot_ordination(pdfu_prelim_gen_trim_R, pdfu_prelim_gen_trim_ord, type =
    "samples", color = "Suspicious", shape = "Timepoint") +
    theme_bw(base_size = 8) +
    scale_color_manual(values = c("gray50", "firebrick"), name = "Sample status") +
    scale_shape_manual(values = c(19, 17), labels = c("baseline", "follow-up")))

fig1 <- arrangeGrob(ordGenSuspicious, ordLeg, nrow = 1, widths = c(8, 2.5))

ggsave(fig1, filename = "Outputs/figure1.pdf", device = cairo_pdf,
       width = 3.35, height = 2, units = "in")
```

Finally, check the same on OTU and family levels to see if these choices make sense:

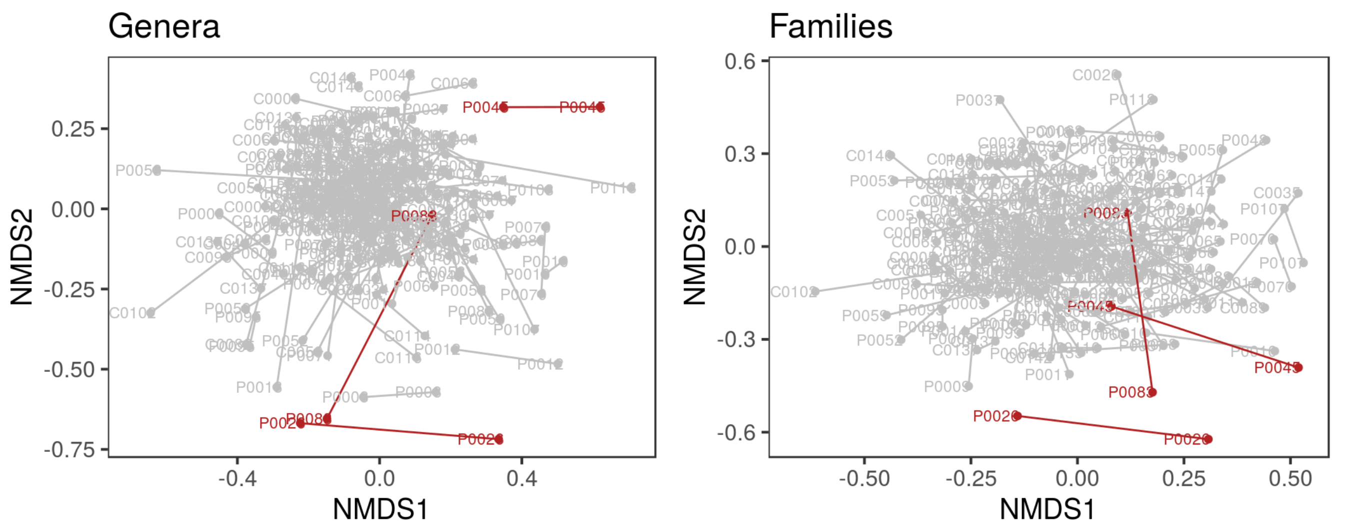
```
# Genus level plot with labels

ordGenSuspicious2 <- plot_ordination(pdfu_prelim_gen_trim_R, pdfu_prelim_gen_trim_ord, type =
    "samples", color = "Suspicious") +
  theme_bw(base_size = 15) +
  scale_color_manual(values = c("gray75", "firebrick")) +
  geom_polygon(aes(group = Subject)) +
  geom_text(aes(label = Subject, x = NMDS1, y = NMDS2), size = 3, nudge_x = -0.05) +
  theme(legend.position  =  "none", panel.grid = element_blank())

# OTU level plot
pdfu_prelim_trim_R <- rarefy_even_depth(subset_samples(pdfu_prelim, Type == "sample"), rngseed =
    459116)

sample_data(pdfu_prelim_trim_R)$Suspicious <- sample_data(pdfu_prelim_gen_trim_R)$Suspicious

pdfu_prelim_trim_ord <- ordinate(pdfu_prelim_trim_R,"NMDS", "bray", try = 400)

ordOTUSuspicious <- plot_ordination(pdfu_prelim_trim_R, pdfu_prelim_trim_ord, type = "samples",
    color = "Suspicious") +
  theme_bw(base_size = 15) +
  scale_color_manual(values = c("gray75", "firebrick")) +
  geom_polygon(aes(group = Subject)) +
  geom_text(aes(label = Subject, x = NMDS1, y = NMDS2), size = 3, nudge_x = -0.05) +
  theme(legend.position = "none", panel.grid = element_blank())

# Family level plot
pdfu_prelim_fam_trim_R <- rarefy_even_depth(collapseTaxLevel(subset_samples(pdfu_prelim, Type ==
    "sample"), level = "Family", maxUnclassifiedLevel = "Order"), rngseed = 110199)

sample_data(pdfu_prelim_fam_trim_R)$Suspicious <- sample_data(pdfu_prelim_gen_trim_R)$Suspicious

pdfu_prelim_fam_trim_ord <- ordinate(pdfu_prelim_fam_trim_R, "NMDS", "bray", try = 400)

ordFamSuspicious <- plot_ordination(pdfu_prelim_fam_trim_R, pdfu_prelim_fam_trim_ord, type =
    "samples", color = "Suspicious") +
  theme_bw(base_size = 15) +
  scale_color_manual(values = c("gray75", "firebrick")) +
  geom_text(aes(label = Subject, x = NMDS1, y = NMDS2), size = 3, nudge_x = -0.05) +
  geom_polygon(aes(group = Subject)) +
  theme(legend.position = "none",  panel.grid = element_blank())

ordLeg2 <- g_legend(plot_ordination(pdfu_prelim_gen_trim_R, pdfu_prelim_gen_trim_ord, type =
    "samples",
 color = "Suspicious") +
    theme_bw(base_size = 16) +
    scale_color_manual(values = c("gray50", "firebrick"), name = "Sample status"))
```

Draw all three plots:

```
grid.arrange(ordOTUSuspicious + ggtitle("OTUs"),
             ordLeg2,
             nrow = 1)
```



```
grid.arrange(ordGenSuspicious2 + ggtitle("Genera"),
             ordFamSuspicious + ggtitle("Families"),
             nrow = 1)
```



As noted before, two of the excluded subject had additional sampling-related issues that support excluding them. P26, who doesn't, seems consistently outlierish on all three levels. All excluded subjects are Parkinson's patients, and removing outlierish points from the patient group is a conservative choice that it will, if anything, lessen the differences between the patient and control groups.

Clinical data comparisons

Data import

Import the full clinical metadata:

```
pdclin <- read.table("Inputs/pdclin_final.txt", sep = "\t")
```

Make sure the numerous yes/no variables coded as 1/0 are factors:

```
varTypeFix <- function(df){

  for(i in 1:ncol(df)){

    var <- df[,i]
    varvals <- sort(unique(na.omit(var)))
    if(length(varvals) == 2){
      if(identical(varvals, as.integer(c(0, 1))) | identical(varvals, c("0", "1")) | is.factor(var))
          {
        var <- factor(var)
      }
    } else if(is.factor(var)){
      var <- factor(var)
    }
    df[,i] <- var
  }

  return(df)
}

pdclin <- varTypeFix(pdclin)

# two additional variables need to be fixed manually:
pdclin$tobacco_when_last <- factor(pdclin$tobacco_when_last)
pdclin$rasagiline_mg <- as.numeric(as.character(pdclin$rasagiline_mg))
```

Create subsets of metadata for the two timepoints:

```
metaFU <- subset(pdclin, Timepoint == "followup")
metaBL <- subset(pdclin, Timepoint == "baseline")
```

Basic statistics for PD vs control

Because the output of the base-R summary function isn't the most handy to deal with, the metadata comparisons were done with a (somewhat clunky) custom function:

```
metaSummaryGeneric <- function(dfin, varList, groupVar, pairing = FALSE, normtest = FALSE){

    res <- as.data.frame(matrix(nrow = length(varList), ncol = 15))
    rownames(res)<-varList
    colnames(res)<-c("percent_level", paste(
      rep(c(levels(dfin[[groupVar]])[1], levels(dfin[[groupVar]])[2]), each = 6),
      rep(c("%", "Q1", "median", "mean", "Q3", "SD"), 2), sep = "_"), "pVal", "test")

    for(i in 1:length(varList)){

      df <- dfin

      if(pairing == TRUE){
      df <- subset(df, !(Subject %in% df[which(is.na(df[[varList[i]]])), "Subject"]))
      }

        var <- df[[varList[i]]]

        if(table(var)[1] == length(var) | sum(table(var)) == 0){
            res[i, 15] <- "none"
            next
        }

        varN <- subset(df, df[[groupVar]] == levels(df[[groupVar]])[1])[[varList[i]]]
        varP <- subset(df, df[[groupVar]] == levels(df[[groupVar]])[2])[[varList[i]]]

        if(is.factor(var)){
            if(sum(is.na(varN))<length(varN) &
                    length(levels(var)) == 2){
                res[i, 1] <- names(table(var)[2])
                res[i, 2] <- table(varN)[2]/length(varN)*100
                res[i, 8] <- table(varP)[2]/length(varP)*100
                res[i, 14] <- fisher.test(table(df[, c(groupVar, varList[i])]))$p
                res[i, 15] <- "fisher"
                }
        } else if(sum(is.na(varN))<length(varN)){
            res[i, 3:6] <- summary(varN)[2:5]
            res[i, 7] <- sd(varN, na.rm = TRUE)
            res[i, 9:12] <- summary(varP)[2:5]
            res[i, 13] <- sd(varP, na.rm = TRUE)
            if(normtest == TRUE & shapiro.test(var)$p.value>0.05){
                res[i, 14] <- t.test(var~df[[groupVar]])$p.value
                res[i, 15]<-"t"
            } else if(pairing == TRUE){
              res[i, 14] <- wilcox.test(var~df[[groupVar]], paired = TRUE)$p.value
              res[i, 15] <- "wilcox"
            } else {
              res[i, 14] <- wilcox.test(var~df[[groupVar]])$p.value
              res[i, 15] <- "wilcox"
            }
        }
    }
    return(res)
}
```

The clinical comparisons at each timepoint were run with this function, with the results for age, sex, BMI and all those variables that had $p < 0.1$ at either timepoint collected into the final table:

```
# Baseline
metaCompBL <- metaSummaryGeneric(metaBL, colnames(metaBL)[2:ncol(metaBL)], "Parkinson", normtest =
    TRUE)
metaCompDiffBL <- subset(metaCompBL, pVal<0.1)
metaCompDiffBL <- rbind(metaCompDiffBL, metaCompBL[c("gender", "age_at_stool_collection", "BMI"),])

## Follow-up
metaCompFU <- metaSummaryGeneric(metaFU, colnames(metaFU)[2:ncol(metaFU)], "Parkinson", normtest =
    TRUE)
metaCompDiffFU <- subset(metaCompFU, pVal<0.1)
metaCompDiffFU <- rbind(metaCompDiffFU, metaCompFU[c("gender", "BMI"),])

# Add variables that are only significant at the other timepoint to the results
metaCompDiffBL <- rbind(metaCompDiffBL, metaCompBL[rownames(metaCompBL) %in%
            setdiff(rownames(metaCompDiffFU), rownames(metaCompDiffBL)),])
metaCompDiffFU <- rbind(metaCompDiffFU, metaCompFU[rownames(metaCompFU) %in%
            setdiff(rownames(metaCompDiffBL), rownames(metaCompDiffFU)),])

# Combine the results
metaCompDiffBL$Timepoint <- "baseline"
metaCompDiffFU$Timepoint <- "followup"
metaCompDiffBL$Variable <- rownames(metaCompDiffBL)
metaCompDiffFU$Variable <- rownames(metaCompDiffFU)
metaCompDiff <- rbind(metaCompDiffBL, metaCompDiffFU)

# Remove variables missing entirely from baseline
metaCompDiff <- subset(metaCompDiff, !(Variable %in% subset(metaCompDiff, test == "none")$Variable))

# Additional data rearrangement and summarization
metaCompDiffExport <- metaCompDiff[order(metaCompDiff$Variable),]

# Concatenation function to combine data from various columns
summaryConcat <- function(df, vargroup){
    percvar <- paste(vargroup, "%", sep = "_")
    meanvar <- paste(vargroup, "mean", sep = "_")
    sdvar <- paste(vargroup, "SD", sep = "_")
    q1var <- paste(vargroup, "Q1", sep = "_")
    q3var <- paste(vargroup, "Q3", sep = "_")
    medianvar <- paste(vargroup, "median", sep = "_")

    pmmvar <- round(df[, percvar], 2)
    pmmvar[df$test == "t"] <- paste(
        round(df[df$test == "t",meanvar],2), " ± ",
        round(df[df$test == "t",sdvar],2), sep = "")
    pmmvar[df$test == "wilcox"] <- paste(
        round(df[df$test == "wilcox",medianvar],2), " [",
        round(df[df$test == "wilcox",q1var],2), "-",
        round(df[df$test == "wilcox",q3var],2), "]", sep="")

    return(pmmvar)
}

metaCompDiffExport$control_perc_meanSD_medianIQR <- summaryConcat(metaCompDiffExport, "control")
metaCompDiffExport$Parkinson_perc_meanSD_medianIQR <- summaryConcat(metaCompDiffExport, "Parkinson")

# BMI tests as normaly distributed at one time point and not the other.
# separately rerun and rearrange to use non-parametric for both timepoints
fuBMI <- which(metaCompDiffExport$Variable == "BMI" & metaCompDiffExport$Timepoint == "followup")

metaCompDiffExport[fuBMI,"pVal"] <- wilcox.test(metaFU$BMI~metaFU$Parkinson)$p.value
metaCompDiffExport[fuBMI,"test"] <- "wilcox"
metaCompDiffExport[fuBMI, "control_perc_meanSD_medianIQR"] <- paste(
        round(metaCompDiffExport[fuBMI,"control_median"], 2), " [",
        round(metaCompDiffExport[fuBMI,"control_Q1"], 2), "-",
        round(metaCompDiffExport[fuBMI,"control_Q3"], 2), "]", sep="")
metaCompDiffExport[fuBMI, "Parkinson_perc_meanSD_medianIQR"] <- paste(
        round(metaCompDiffExport[fuBMI,"Parkinson_median"], 2), " [",
        round(metaCompDiffExport[fuBMI,"Parkinson_Q1"], 2), "-",
        round(metaCompDiffExport[fuBMI,"Parkinson_Q3"], 2), "]", sep="")

# Reorder columns
metaCompDiffExport <- metaCompDiffExport[, c("Variable", "Timepoint", "percent_level",
    "control_perc_meanSD_medianIQR", "Parkinson_perc_meanSD_medianIQR", "pVal", "test")]

# leave out the mg-amount variables of drugs:
metaCompDiffExport <- metaCompDiffExport[-grep("mg",rownames(metaCompDiffExport)),]

# Save the list of differing variables for further use,
diffVarsCvsPD <- unique(metaCompDiffExport$Variable)
diffVarsCvsPD
```

```
##  [1] "age_at_stool_collection"
##  [2] "BMI"
##  [3] "GDS_15"
##  [4] "gender"
##  [5] "history_TIA_ischemic_stroke"
##  [6] "LED"
##  [7] "meds_ACEI_ARB"
##  [8] "meds_anticholinergic"
##  [9] "meds_ca_antagonist"
## [10] "meds_COMT_inhibitor"
## [11] "meds_dopa"
## [12] "meds_dopamine_agonist"
## [13] "meds_MAO_inhibitor"
## [14] "meds_statin"
## [15] "meds_Warfarin"
## [16] "MMSE_total"
## [17] "NMSQuest_total"
## [18] "NMSS_total"
## [19] "RBDSQ"
## [20] "RLS"
## [21] "Rome_III_constip_defec_sumscore_9.15"
## [22] "Rome_III_IBS_criteria_fulfilled"
## [23] "SCS_PD_total"
## [24] "SDQ_total"
## [25] "sniffinsticks"
## [26] "Wexner_total"
```

Table 3

The results of these comparisons were exported (only first ten rows shown here):

```
rownames(metaCompDiffExport) <- NULL

kable(head(metaCompDiffExport, 10), digits = 3, col.names = c("Variable", "Timepoint", "percent
    level", "control % / meanSD / medianIQR", "Parkinson % / meanSD / medianIQR", "pVal", "test"))
```

Variable	Timepoint	percent level	control % / meanSD / medianIQR	Parkinson % / meanSD / medianIQR	pVal	test
age_at_stool_collection	baseline	NA	64.45 ± 6.9	65.2 ± 5.52	0.499	t
age_at_stool_collection	followup	NA	66.53 ± 6.89	67.33 ± 5.51	0.471	t
BMI	baseline	NA	26.23 [24.1–28.05]	26.51 [24.25–29.36]	0.319	wilcox
BMI	followup	NA	26.94 [24.32–28.64]	27.24 [23.95–30.08]	0.572	wilcox
GDS_15	baseline	NA	1 [0–1]	2 [1–4]	0.000	wilcox
GDS_15	followup	NA	0 [0–1]	3 [2–6]	0.000	wilcox
gender	baseline	M	50	51.56	1.000	fisher
gender	followup	M	50	51.56	1.000	fisher
history_TIA_ischemic_stroke	baseline	1	37.5	6.25	0.000	fisher
history_TIA_ischemic_stroke	followup	1	37.5	7.81	0.000	fisher

```
## Export
write.csv(metaCompDiffExport, file = "Outputs/table_3.csv")
```

Measuring disease progression in PD patients

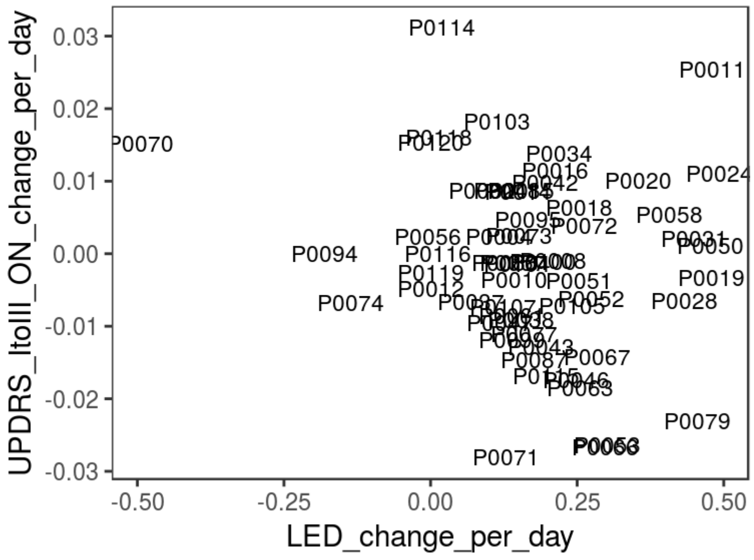
Calculate UPDRS I to III change and create a separate table of PD-only data for progression comparisons; exclude patients on Deep Brain Stimulation (DBS == 1), those with missing values in LED or UPDRS I to III change, and those who are left unpaired after these exclusions:

```
pdclin$UPDRS_ItoIII_ON_change <- c(
  rep(NA, sum(pdclin$Parkinson == "control" & pdclin$Timepoint == "baseline")),
    rep(0, sum(pdclin$Parkinson == "Parkinson" & pdclin$Timepoint == "baseline")),
    rep(NA, sum(pdclin$Parkinson == "control" & pdclin$Timepoint == "followup")),
    subset(pdclin, Timepoint == "followup" & Parkinson == "Parkinson")[["UPDRS_ItoIII_ON"]]-
        subset(pdclin, Timepoint == "baseline" & Parkinson == "Parkinson")[["UPDRS_ItoIII_ON"]])

prog_change <- subset(pdclin, Parkinson == "Parkinson" & DBS != 1)
prog_change <- subset(prog_change, !is.na(prog_change$LED) &
    !is.na(prog_change$UPDRS_ItoIII_ON_change))
prog_change <- prog_change[!(prog_change$Subject %in% names(which(table(prog_change$Subject) ==
    1))), ]
```

Calculate “change per day” variables for LED and UPDRS I to III and plot the values:

```
fu_bl_change <- function(df, var){
    newvar <- rep(NA, nrow(subset(df, Timepoint == "baseline")))
    newvar <- c(newvar, (df[df$Timepoint == "followup", var]-
        df[df$Timepoint == "baseline", var])/
            df[df$Timepoint == "followup", "days_between_appointments"])
    return(newvar)
}

prog_change$LED_change_per_day <- fu_bl_change(prog_change, "LED")
prog_change$UPDRS_ItoIII_ON_change_per_day <- fu_bl_change(prog_change, "UPDRS_ItoIII_ON")
```

```
ggplot(subset(prog_change, Timepoint == "followup"),
       aes(x = LED_change_per_day, y = UPDRS_ItoIII_ON_change_per_day, label = Subject)) +
  geom_text(size = 3) +
  theme_bw() +
  theme(panel.grid = element_blank())
```



One subject (P0070) has a very large change in LED score; this was due to a side-effect related adjustment in medication. Exclude this subject from further progression comparisons:

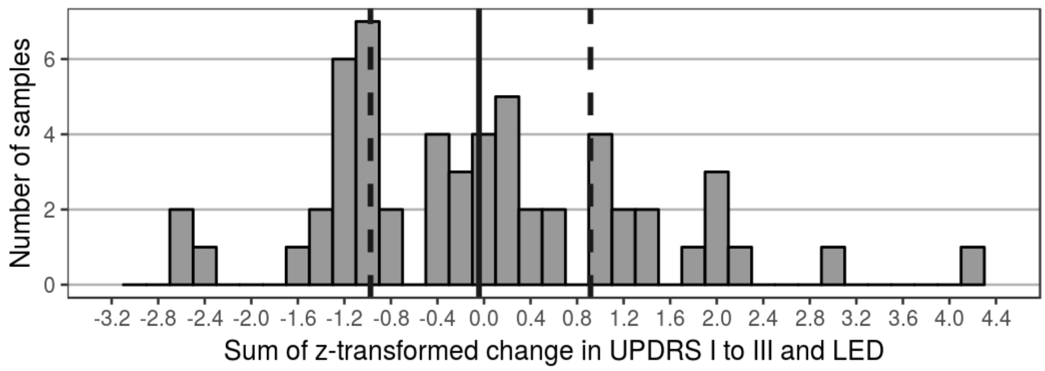
```
prog_change <- subset(prog_change, Subject != "P0070")
```

Z-transform the “change per day” variables, subset to follow-up only, and create a combined sum variable:

```
prog_change$LED_change_z <- scale(prog_change$LED_change_per_day)[, 1]
prog_change$UPDRS_ItoIII_ON_change_z <- scale(prog_change$UPDRS_ItoIII_ON_change_per_day)[, 1]

prog_change_fu <- subset(prog_change, Timepoint == "followup")
prog_change_fu$UPDRS_LED_change_sum <- rowSums(prog_change_fu[, c("LED_change_z",
    "UPDRS_ItoIII_ON_change_z")])

summary(prog_change_fu$UPDRS_LED_change_sum)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.59786 -0.97567 -0.04293  0.00000  0.91515  4.11067
```

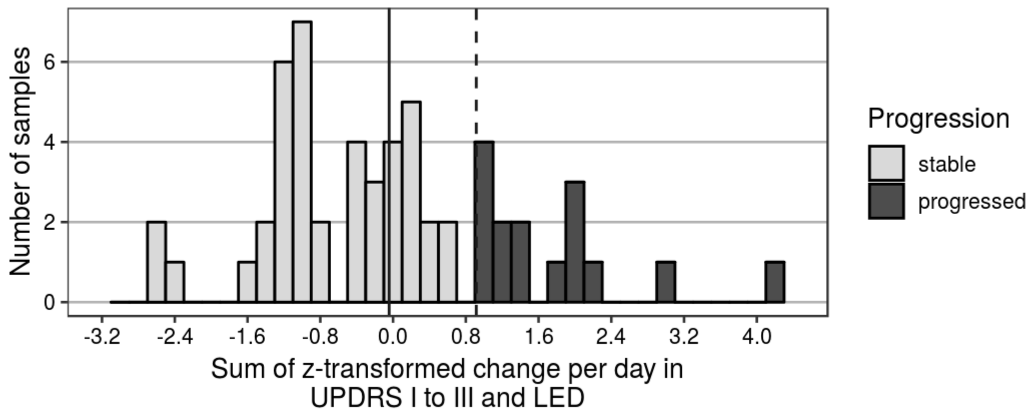
```
# Plot that shows the median and 1st and 3rd quartiles:
ggplot(prog_change_fu, aes(UPDRS_LED_change_sum)) +
  geom_histogram(color = "black", binwidth = 0.2, fill = "gray60") +
  theme_bw(base_size = 10) +
  scale_x_continuous(breaks = c(seq(-3.2, 4.5, 0.4)), limits = c(-3.2, 4.4)) +
  geom_vline(xintercept = quantile(prog_change_fu$UPDRS_LED_change_sum, probs = 0.25), lty = 2,
      color = "gray10", size = 1) +
  geom_vline(xintercept = median(prog_change_fu$UPDRS_LED_change_sum), color = "gray10", size = 1) +
  geom_vline(xintercept = quantile(prog_change_fu$UPDRS_LED_change_sum, probs = 0.75),
             lty = 2, color = "gray10", size = 1) +
  xlab("Sum of z-transformed change in UPDRS I to III and LED") +
  scale_y_continuous(breaks = seq(0, 10, 2)) +
  ylab("Number of samples") +
  theme(panel.grid.major.x = element_blank(), panel.grid.minor.x = element_blank(),
        panel.grid.major.y = element_line(color = "gray70"), panel.grid.minor.y = element_blank())
```



To contrast the subjects with the largest change in UPDRS and LED sum, we decided to use the 3rd quartile as a cutoff, and created a progression variable based on this. A histogram colored by the progression variable:

Figure 2

```r
# Progression variable:
prog_change_fu$ProgCat <- cut(prog_change_fu$UPDRS_LED_change_sum,
                              breaks = c(min(prog_change_fu$UPDRS_LED_change_sum) - 0.1, 0.8,
                                         max(prog_change_fu$UPDRS_LED_change_sum)))
levels(prog_change_fu$ProgCat) <- c("stable", "progressed")

# Redraw the plot using this new variable:
fig2 <- ggplot(prog_change_fu, aes(UPDRS_LED_change_sum, fill = ProgCat)) +
    geom_histogram(color = "black", binwidth = 0.2) +
    theme_bw(base_size = 10) +
    scale_x_continuous(breaks = c(seq(-3.2, 4.5, 0.8)), limits = c(-3.2, 4.4)) +
    scale_fill_manual(values = c("gray85", "gray30"), name = "Progression") +
    geom_vline(xintercept = median(prog_change_fu$UPDRS_LED_change_sum), color = "gray10", size =
        0.5) +
    geom_vline(xintercept = quantile(prog_change_fu$UPDRS_LED_change_sum, probs = 0.75),
               lty = 2, color = "gray10", size = 0.5) +
    xlab("Sum of z-transformed change per day in\nUPDRS I to III and LED") +
    scale_y_continuous(breaks = seq(0, 10, 2)) +
  ylab("Number of samples") +
    theme(panel.grid.major.x = element_blank(),
          panel.grid.major.y = element_line(color = "gray70"),
          panel.grid.minor = element_blank(),
          axis.text = element_text(color = "black"),
          legend.key.height = unit(5, "mm"),
          legend.key.width = unit(5, "mm"))
fig2
```



```r
## Export to pdf
ggsave(fig2, filename = "Outputs/figure2.pdf", device = cairo_pdf,
       width = fullpage, height = 3, units = "in")
```

Add this progression variable to the data with both time points (the same value at baseline and follow-up, so at baseline, this variable classifies subjects depending on whether they will or won't progress between timepoints):

```
prog_change$ProgCat <- rep(prog_change_fu$ProgCat, 2)
prog_change$ProgNum <- rep(prog_change_fu$UPDRS_LED_change_sum, 2)
```

Comparisons between the disease progression categories

Difference between timepoints

Test whether there is a difference between timepoints for each variable when only looking at the stable or the progressed patients ("Does this variable differ between baseline and follow-up in stable patients? What about progressed patients?"). Again, run with a (rather cumbersome) custom summary function.

Select variables for comparisons (those that don't have too many NA values and are not "change between timepoints" variables or the progression variable that are being contrasted).

```
# select variables for comparisons
tpCompVars <- colnames(prog_change)[-c(1, grep("Timepoint", colnames(prog_change)),
        grep("change", colnames(prog_change)), grep("Prog", colnames(prog_change)))]
tpCompVars <- tpCompVars[which(colSums(is.na(prog_change[,tpCompVars])) < 50)]
tpCompVars <- tpCompVars[-grep("Subject", tpCompVars)]

# Add a sum variable of all levodopa medications:
prog_change$levodopa_mg <- rowSums(prog_change[, grep("levodopa.*mg", colnames(prog_change))])
tpCompVars <- c(tpCompVars, "levodopa_mg")

# Run the comparisons separately for each group:

# Stable
metapd_timepoints_summary_noprog <- metaSummaryGeneric(subset(prog_change, ProgCat == "stable"),
    tpCompVars, "Timepoint", pairing = TRUE)

# Progressed
metapd_timepoints_summary_prog <- metaSummaryGeneric(subset(prog_change, ProgCat == "progressed"),
    tpCompVars, "Timepoint", pairing = TRUE)
```

Perform some further rearrangement steps to get the table to the final form:

```
summaryConcat <- function(df, vargroup){
  percvar <- paste(vargroup, "%", sep = "_")
  meanvar <- paste(vargroup, "mean", sep = "_")
  sdvar <- paste(vargroup, "SD", sep = "_")
  q1var <- paste(vargroup, "Q1", sep = "_")
  q3var <- paste(vargroup, "Q3", sep = "_")
  medianvar <- paste(vargroup, "median", sep = "_")

  pmmvar <- round(df[,percvar],2)
  pmmvar[df$test == "t"]<-paste(
    round(df[df$test == "t",meanvar],2), " ± ",
    round(df[df$test == "t",sdvar],2),
    sep="")
  pmmvar[df$test == "wilcox"]<-paste(
    round(df[df$test == "wilcox",medianvar],2), " [",
    round(df[df$test == "wilcox",q1var],2), "-",
    round(df[df$test == "wilcox",q3var],2), "]", sep = "")

  return(pmmvar)
}

metapd_timepoints_summary_noprog <-
    metapd_timepoints_summary_noprog[!is.na(metapd_timepoints_summary_noprog$test),]
metapd_timepoints_summary_noprog$bl_perc_meanSD_medianIQR <-
    summaryConcat(metapd_timepoints_summary_noprog, "baseline")
```

```
metapd_timepoints_summary_noprog$fu_perc_meanSD_medianIQR <-
    summaryConcat(metapd_timepoints_summary_noprog, "followup")
metapd_timepoints_summary_noprog <- metapd_timepoints_summary_noprog[, c(
  "bl_perc_meanSD_medianIQR", "fu_perc_meanSD_medianIQR", "pVal", "test")]
colnames(metapd_timepoints_summary_noprog) <- paste("stable",
    colnames(metapd_timepoints_summary_noprog), sep = "_")

metapd_timepoints_summary_prog <-
    metapd_timepoints_summary_prog[!is.na(metapd_timepoints_summary_prog$test),]
metapd_timepoints_summary_prog$bl_perc_meanSD_medianIQR <-
    summaryConcat(metapd_timepoints_summary_prog, "baseline")
metapd_timepoints_summary_prog$fu_perc_meanSD_medianIQR <-
    summaryConcat(metapd_timepoints_summary_prog, "followup")
metapd_timepoints_summary_prog <- metapd_timepoints_summary_prog[,c(
  "bl_perc_meanSD_medianIQR", "fu_perc_meanSD_medianIQR", "pVal", "test")]
colnames(metapd_timepoints_summary_prog) <- paste("progressed",
    colnames(metapd_timepoints_summary_prog), sep = "_")

metapd_tp_prog_summary <- cbind(metapd_timepoints_summary_noprog, metapd_timepoints_summary_prog)
metapd_tp_prog_summary$Variable <- rownames(metapd_tp_prog_summary)
```

Difference between progression categories

Test whether there is a difference between the stable and progressed patients at each timepoint ("Do the stable and progressed patients differ at baseline? What about follow-up?"), also with a customized summary function.

Run the comparisons and add these to the final output table from the between-timepoint comparisons in the previous subsection:

```
prog_conf_comp_bl <- metaSummaryGeneric(subset(prog_change, Timepoint == "baseline"),
    metapd_tp_prog_summary$Variable, "ProgCat")
colnames(prog_conf_comp_bl) <- paste("prog_baseline", colnames(prog_conf_comp_bl), sep = "_")

prog_conf_comp_fu <- metaSummaryGeneric(subset(prog_change, Timepoint == "followup"),
    metapd_tp_prog_summary$Variable, "ProgCat")
colnames(prog_conf_comp_fu) <- paste("prog_followup", colnames(prog_conf_comp_fu), sep = "_")
```

Combine the two sets of comparisons

Table 4

Collect the combined results for differences in progressed and stable patients, between groups and between timepoints. This table was exported for the manuscript (only first 10 rows shown here).

```
# Combine results with the between-timepoint comparisons:
metapd_tp_prog_summary <- cbind(metapd_tp_prog_summary,
    prog_conf_comp_bl[, c("prog_baseline_pVal", "prog_baseline_test")],
    prog_conf_comp_fu[, c("prog_followup_pVal", "prog_followup_test")])

# Table of results
kable(metapd_tp_prog_summary[1:10, 1:4], col.names = c("stable baseline % / meanSD /medianIQR",
    "stable followup % / meanSD /medianIQR", "stable timepoint diff pVal", "stable timepoint diff
    test"))
```

	stable baseline % / meanSD /medianIQR	stable followup % / meanSD /medianIQR	stable timepoint diff pVal	stable timepoint diff test
gender	53.66	53.66	1.0000000	fisher
BMI	26.25 [24.15–29.8]	27.28 [24.27–30.21]	0.2247968	wilcox
age_at_stool_collection	65 [61–69]	67 [63–71]	0.0000000	wilcox
sniffinsticks	7 [6–9]	7 [5–10]	0.8867549	wilcox
age_pd_diagnosed	60 [58–64]	60 [58–64]	NaN	wilcox
age_motor_-symptoms_onset	59 [56–63]	59 [56–63]	NaN	wilcox

	stable baseline % / meanSD /medianIQR	stable followup % / meanSD /medianIQR	stable timepoint diff pVal	stable timepoint diff test
age_NMS_onset	58 [55–60.5]	58 [55–60.5]	NaN	wilcox
PD_relative	23.08	25.64	1.0000000	fisher
history_-lactoseintolerance	9.76	9.76	1.0000000	fisher
history_diverticulosis	2.44	2.44	1.0000000	fisher

```r
kable(metapd_tp_prog_summary[1:10, 5:8], col.names = c("progressed baseline % / meanSD /medianIQR",
    "progressed followup % / meanSD /medianIQR", "progressed timepoint diff pVal", "progressed
    timepoint diff test"))
```

	progressed baseline % / meanSD /medianIQR	progressed followup % / meanSD /medianIQR	progressed timepoint diff pVal	progressed timepoint diff test
gender	46.67	46.67	1.0000000	fisher
BMI	26.53 [25.28–28.24]	26.9 [23.89–29.89]	0.8903809	wilcox
age_at_stool_collection	65 [62.5–66.5]	67 [65–69]	0.0003229	wilcox
sniffinsticks	7 [7–9.75]	7.5 [6–10]	0.8868323	wilcox
age_pd_diagnosed	60 [57–65]	60 [57–65]	NaN	wilcox
age_motor_-symptoms_onset	59 [56.5–63.5]	59 [56.5–63.5]	NaN	wilcox
age_NMS_onset	61.5 [58.12–64]	61.5 [58.12–64]	NaN	wilcox
PD_relative	13.33	13.33	1.0000000	fisher
history_-lactoseintolerance	13.33	13.33	1.0000000	fisher
history_diverticulosis	NA	NA	NA	none

```r
kable(metapd_tp_prog_summary[1:10, 9:ncol(metapd_tp_prog_summary)], col.names = c("Variable", "prog
    vs stable baseline pVal", "prog vs stable baseline test", "prog vs stable followup pVal", "prog
    vs stable followup test"))
```

	Variable	prog vs stable baseline pVal	prog vs stable baseline test	prog vs stable followup pVal	prog vs stable followup test
gender	gender	0.7654898	fisher	0.7654898	fisher
BMI	BMI	0.9230736	wilcox	0.6080844	wilcox
age_at_stool_-collection	age_at_stool_-collection	0.8023173	wilcox	0.9040648	wilcox
sniffinsticks	sniffinsticks	0.4335528	wilcox	0.3749284	wilcox
age_pd_diagnosed	age_pd_diagnosed	0.9629168	wilcox	0.9629168	wilcox
age_motor_-symptoms_onset	age_motor_-symptoms_onset	0.8456205	wilcox	0.8456205	wilcox
age_NMS_onset	age_NMS_onset	0.2334229	wilcox	0.2334229	wilcox
PD_relative	PD_relative	0.7075588	fisher	0.4809485	fisher
history_-lactoseintolerance	history_-lactoseintolerance	0.6537878	fisher	0.6537878	fisher
history_-diverticulosis	history_-diverticulosis	1.0000000	fisher	1.0000000	fisher

```r
## Export
write.csv(metapd_tp_prog_summary, "Outputs/table_4.csv")
```

List of variables differing between groups for further use downstream:

```r
prog_sigvars <- unique(c(subset(metapd_tp_prog_summary, prog_baseline_pVal < 0.1)$Variable,
    subset(metapd_tp_prog_summary, prog_followup_pVal < 0.1)$Variable))
prog_sigvars
```

```
##  [1] "meds_ASA"                "meds_statin"
##  [3] "ropinirole_mg"           "meds_COMT_inhibitor"
##  [5] "meds_dopamine_agonist"   "pigd_score_jankovic_ON"
##  [7] "UPDRS_III_total_ON"      "levodopa_entacapone_mg"
##  [9] "entacapone_mg"           "pramipexole_mg"
## [11] "UPDRS_V_ON"              "UPDRS_II_total"
## [13] "UPDRS_ItoIII_ON"
```

Final clinical data table

Add the progression variable to the main clinical data table and make a list of samples used in the progression analyses. This will be the metadata used in the final analyses.

```r
pdclin[rownames(prog_change),"ProgCat"]<-prog_change$ProgCat
pdclin[rownames(prog_change),"ProgNum"]<-prog_change$ProgNum

prog_samples <- rownames(pdclin)[!is.na(pdclin$ProgCat)]
```

Dietary data analyses (FFQ data)

Food Frequency Questionnaire data was only collected at the follow-up timepoint, so the following analyses will all concern that time point. (The variable names for this data are in Finnish, and some variables are local brands; translations are provided as necessary.)

Import data:

```r
pdffq <- read.table("Inputs/pdfu_ffq.txt")
```

Data setup and energy correction

For nutrients:

```r
# List nutrients that are already given as percent of energy
nutrVarsE <- grep("_E", colnames(pdffq), value = TRUE)

# Convert non-energy nutrients to units per 1000 kcal
nutrVarsNE <- colnames(pdffq)[31:63]
nutrVarsNEdf <- pdffq[, nutrVarsNE] / pdffq$Energ_kc * 1000
colnames(nutrVarsNEdf) <- paste(colnames(nutrVarsNEdf), "per1kkc", sep = "_")
nutrVarsNE <- colnames(nutrVarsNEdf)

# A new data frame with only the energy adjusted variables:
pdffqEA <- cbind(pdffq$Parkinson, pdffq[, nutrVarsE], nutrVarsNEdf)
colnames(pdffqEA)[1]<-"Parkinson"

nutrVars <- c(nutrVarsE, nutrVarsNE)
```

For specific food items:

```r
foodVars <- colnames(pdffq[64:234])

foodNorm <- pdffq[, foodVars] / pdffq$Energ_kc * 1000
colnames(foodNorm) <- paste(foodVars, "per1kkc", sep = "_")
foodVars <- colnames(foodNorm)
```

Combine the two into a new table of energy-adjusted variables:

```r
pdffqEA <- cbind(pdffqEA, foodNorm)
```

Make a version of this table where the variables are categorical instead of continuous, splitting them into quintiles:

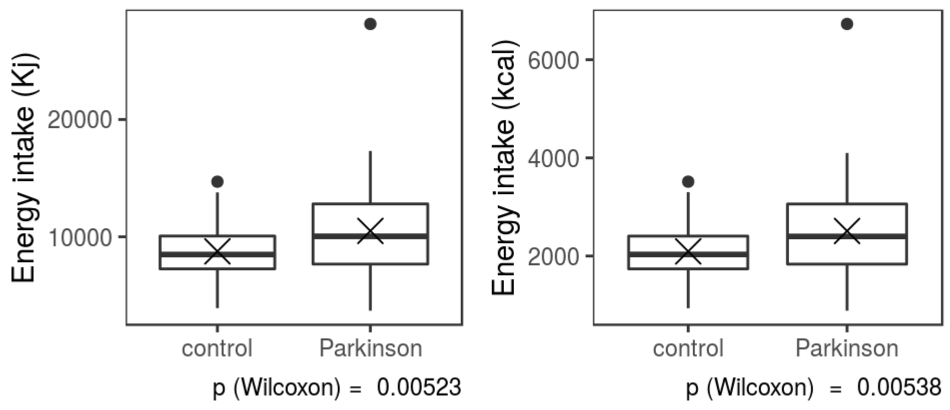
```r
pdffqCat <- pdffqEA

for(i in 2:ncol(pdffqCat)){
    pdffqCat[, i] <- cut(pdffqCat[, i], breaks = unique(quantile(pdffqCat[, i], probs = c(0, 0.2,
        0.4, 0.6, 0.8, 1))), include.lowest = TRUE)
}

# some variables only result in only one category; drop these as uninformative:
singleCatVars <- which(lengths(sapply(pdffqCat, table)) == 1)
pdffqCat <- pdffqCat[, -singleCatVars]
```

Diet variables and PD vs control

Overall energy intake

Plot and test if there are differences in overall energy intake between the groups:

```
ffqPlotEkj <- ggplot(pdffq, aes(x = Parkinson, y = Energ_Kj)) +
    geom_boxplot() +
  theme_bw() +
    stat_summary(fun.y = mean, geom = "point", shape = 4, size = 4) +
    ylab("Energy intake (Kj)") + xlab(NULL) +
    theme(panel.grid = element_blank()) +
    labs(caption = paste("p (Wilcoxon) = ",
                         round(wilcox.test(pdffq$Energ_Kj ~ pdffq$Parkinson)$p.value,
                               digits = 5)))

ffqPlotEkc <- ggplot(pdffq, aes(x = Parkinson, y = Energ_kc)) +
    geom_boxplot() +
  theme_bw() +
    stat_summary(fun.y = mean, geom = "point", shape = 4, size = 4) +
    ylab("Energy intake (kcal)") + xlab(NULL) +
    theme(panel.grid = element_blank()) +
    labs(caption = paste("p (Wilcoxon)  = ",
                         round(wilcox.test(pdffq$Energ_kc ~ pdffq$Parkinson)$p.value,
                               digits = 5)))

grid.arrange(ffqPlotEkj, ffqPlotEkc, nrow = 1)
```

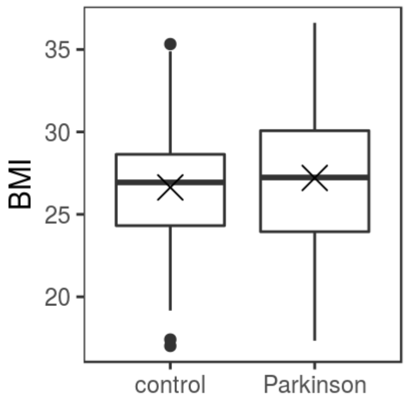


Energy intake is higher in PD patients than controls, and this difference is statistically significant.

BMI

It was already shown in the earlier metadata comparisons above that there is no difference in BMI between the PD and control groups, but here's a plot to illustrate this.

```
ggplot(subset(pdclin, Timepoint == "followup"), aes(x = Parkinson, y = BMI)) +
    geom_boxplot() +
  theme_bw() +
    stat_summary(fun.y = mean, geom = "point", shape = 4, size = 4) +
    ylab("BMI") +
  xlab(NULL) +
    theme(panel.grid = element_blank())
```



Continuous diet variables

Compare the consumption of the energy-adjusted dietary items (as continuous variables) between PD and control.

```
# Make an empty data frame
numDiffs <- as.data.frame(matrix(ncol = 2, nrow = (ncol(pdffqEA)-1)))
colnames(numDiffs) <- c("Var", "WilcoxP")

# Test for differences
for(i in 2:ncol(pdffqEA)){
    numDiffs[i-1, 1] <- colnames(pdffqEA)[i]
    numDiffs[i-1, 2] <- wilcox.test(pdffqEA[, i] ~ pdffqEA$Parkinson)$p.value
}

# Adjust for multiple comparisons
numDiffs[, "pAdj"] <- p.adjust(numDiffs[, "WilcoxP"], method = "fdr")

# Table of variables closest to significant
kable_styling(kable(subset(numDiffs, pAdj < 0.1), digits = 4))
```

	Var	WilcoxP	pAdj
30	Magnes_per1kkc	7e-04	0.0718
36	Niasiini_per1kkc	6e-04	0.0718

None of the diet variables have a statistically significant multiple comparison corrected difference in consumption between the PD and control groups. Only two have $p < 0.1$. These are magnesium ("Magnes") and niacin ("Niasiini") intake.

Categorical diet variables

Compare the consumption of the various dietary items in categorical form between PD and control.

```
# Make an empty data frame
catDiffs <- as.data.frame(matrix(ncol = 2, nrow = (ncol(pdffqCat)-1)))
colnames(catDiffs) <- c("Var", "FisherP")

# Test for differences
for(i in 2:ncol(pdffqCat)){
    catDiffs[i-1, 1] <- colnames(pdffqCat)[i]
    catDiffs[i-1, 2] <- fisher.test(x = pdffqCat[, i],
        y = pdffqCat$Parkinson)$p.value
}

# Adjust for multiple comparisons
catDiffs[, "pAdj"] <- p.adjust(catDiffs[, "FisherP"], method = "fdr")

# Table
kable_styling(kable(subset(catDiffs, pAdj < 0.1), digits = 4))
```

	Var	FisherP	pAdj
143	Kaaliruo__per1kkc	9e-04	0.0984
157	Makaryht__per1kkc	1e-03	0.0984

Similarly to the continuous variables, none of the categorical diet variables have a statistically significant multiple comparison corrected difference in consumption between the PD and control groups. Only two have $p < 0.1$, and these are cabbage-containing dishes ("Kaaliruo") and pasta dishes ("Makaryht"). These do not sound like variables that are particularly interesting to explore further.

Diet variables and PD progression

Test the progression categories separately to see if there are differences in their diets.

Data setup

```
## Set up progression-specific diet data
progmeta <- subset(pdclin, rownames(pdclin) %in% prog_samples & Timepoint=="followup")

progffqCat <- subset(pdffqCat, rownames(pdffqCat) %in% progmeta$Subject)
progffqCat <- cbind(progffqCat, subset(pdffq, rownames(pdffqCat) %in% progmeta$Subject)[,
    c("Energ_Kj", "Energ_kc")])
progffqCat$Progression <- progmeta[progmeta$Subject %in% rownames(progffqCat), "ProgCat"]
progffqCat$ProgressionNum <- progmeta[progmeta$Subject %in% rownames(progffqCat),
    "UPDRS_LED_change_sum"]
progffqCat$Progression <- factor(progffqCat$Progression)
```

Overall energy intake

```
wilcox.test(progffqCat$Energ_kc ~ progffqCat$Progression)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  progffqCat$Energ_kc by progffqCat$Progression
## W = 271, p-value = 0.5054
## alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(progffqCat$Energ_Kj ~ progffqCat$Progression)
```

```
##
##  Wilcoxon rank sum test
##
## data:  progffqCat$Energ_Kj by progffqCat$Progression
## W = 271, p-value = 0.5094
## alternative hypothesis: true location shift is not equal to 0
```

There's no difference in energy consumption between the two disease progression groups.

Continuous diet variables

Compare the consumption of the various continuous, energy-adjusted dietary items between progressed and stable PD patients.

```
# Subset the continuous table for progression
progffqEA <- subset(pdffqEA, rownames(pdffqEA) %in% rownames(progffqCat))
progffqEA$Progression <- progffqCat$Progression

# Make an empty data frame
numDiffsProg <- as.data.frame(matrix(ncol = 2, nrow = (ncol(progffqEA)-2)))
colnames(numDiffsProg) <- c("Var", "WilcoxP")
```

```
# Compare
for(i in 2:(ncol(progffqEA)-1)){
    numDiffsProg[i-1, 1] <- colnames(progffqEA)[i]
    numDiffsProg[i-1, 2] <- wilcox.test(progffqEA[,i]~progffqEA$Progression)$p.value
}

# Adjust for multiple comparisons and check results
numDiffsProg[, "pAdj"]<-p.adjust(numDiffsProg[, "WilcoxP"], method = "fdr")
subset(numDiffsProg, pAdj < 0.1)
```

```
## [1] Var     WilcoxP pAdj
## <0 rows> (or 0-length row.names)
```

There are no variables that have a multiple comparison corrected p-value < 0.1, let alone below the 0.05 cutoff.

Categorical diet variables

Compare the consumption of the various dietary items in categorical form between the two progression groups.

```
# Make an empty data frame
catDiffsProg <- as.data.frame(matrix(ncol = 2, nrow = (ncol(progffqCat)-5)))
colnames(catDiffsProg) <- c("Var", "FisherP")

# Compare
for(i in 2:197){
    catDiffsProg[i-1, 1] <- colnames(progffqCat)[i]
    catDiffsProg[i-1, 2] <- fisher.test(x = progffqCat[, i],
        y = progffqCat$Progression)$p.value
}

# Adjust for multiple comparisons and check results
catDiffsProg[, "pAdj"] <- p.adjust(catDiffsProg[, "FisherP"], method = "fdr")
subset(catDiffsProg, pAdj < 0.1)
```

```
## [1] Var     FisherP pAdj
## <0 rows> (or 0-length row.names)
```

Similarly to the continuous variable comparisons, there are no categorical diet variables that differ significantly between the stable and progressed PD patients.

Dietary patterns with Principal Component Analysis

Perform a PCA to look for overall dietary pattern differences between subjects, using a hand-picked set of non-overlapping food item variables.

```
# Import list variables:
ffqpcaVars <- scan("Inputs/ffq_var_select.txt", what = "character")
ffqpcaVars <- paste(ffqpcaVars, "per1kkc", sep="_")

# Drop variables that ended up with only one level in the categorization
ffqpcaVars <- ffqpcaVars[-which(ffqpcaVars %in% names(singleCatVars))]

## Calculate the PCA with these variables & rename the columns in English:
pcadata <- scale(pdffqEA[, ffqpcaVars])
pcadata <- pcadata[1:nrow(pcadata),]
colnames(pcadata) <- scan("Inputs/ffq_englvars.txt", what="character")
ffqPCAf <- prcomp(pcadata)

# Plot the result
library("factoextra")

fviz_screeplot(ffqPCAf, ncp = 15) +
  theme_bw() +
  geom_hline(yintercept = 5, linetype = 2, color = "red")
```

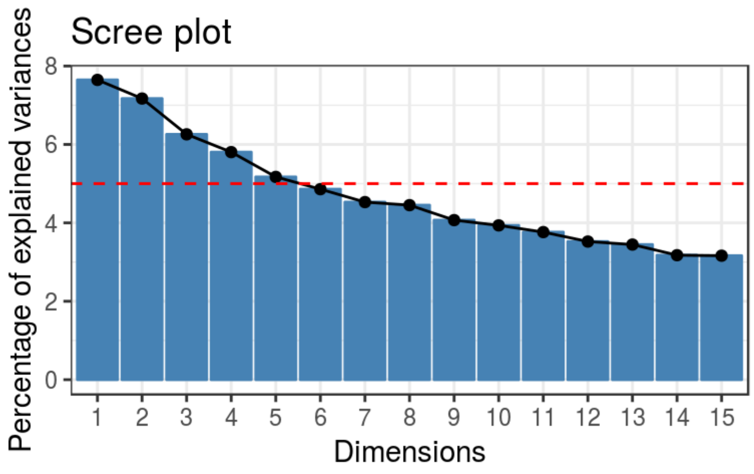


Table S1

The component loadings of the first five Principal Components (PCs) were exported as a supplementary table (only first 10 rows shown here).

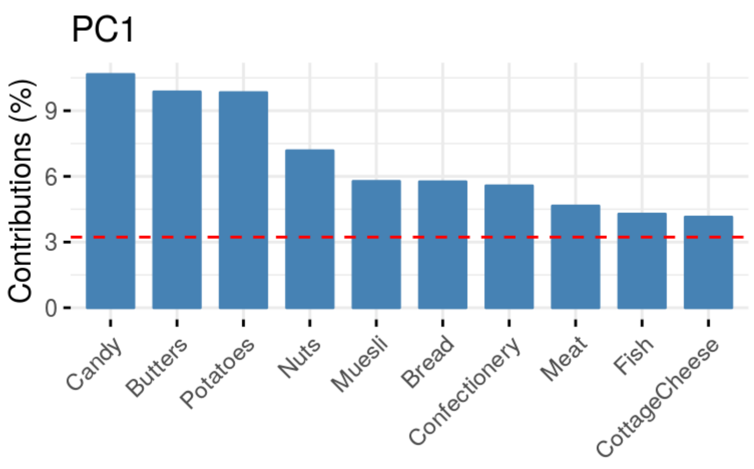
```r
kable(ffqPCAf$rotation[1:10, 1:5], digits = 3)
```

	PC1	PC2	PC3	PC4	PC5
Milk_IceCream	-0.119	0.189	-0.113	0.112	-0.136
Cheese	0.095	0.130	0.252	-0.325	0.186
ProcessedCheese	-0.067	0.120	-0.356	-0.021	0.193
CottageCheese	-0.204	0.212	-0.261	-0.163	0.228
Quark	-0.150	0.022	-0.203	-0.260	0.229
Bread	0.240	0.156	0.226	0.112	0.086
Porridge	-0.186	0.194	-0.118	0.052	-0.193
Cereal	-0.182	0.125	-0.059	-0.175	-0.260
Muesli	-0.240	0.126	0.094	0.068	-0.391
Confectionery	0.236	0.207	-0.117	0.068	-0.249

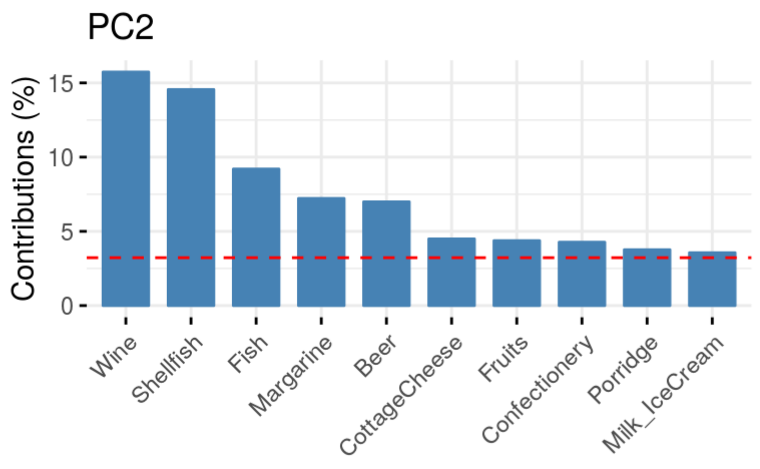
```r
# Export
write.csv(ffqPCAf$rotation[,1:5], "Outputs/table_s1.csv")
```

Overall, there don't seem to be very strong patterns in the diet data, since even the best principal component explains less than 8% of the variation. Plot the top 10 contributing variables for the first three PCs:

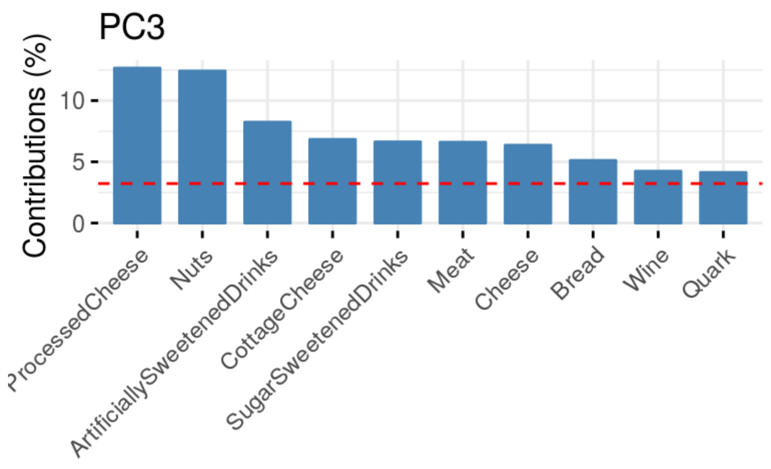
```r
fviz_contrib(ffqPCAf, choice = "var", axes = 1, top = 10) + ggtitle("PC1")
```



```r
fviz_contrib(ffqPCAf, choice = "var", axes = 2, top = 10) + ggtitle("PC2")
```



```
fviz_contrib(ffqPCAf, choice = "var", axes = 3, top = 10) + ggtitle("PC3")
```

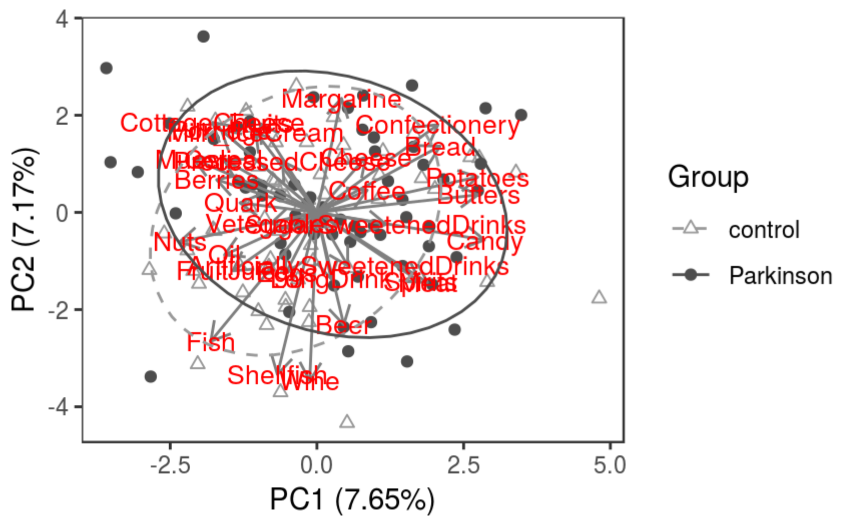


Save the first five PCs for further use:

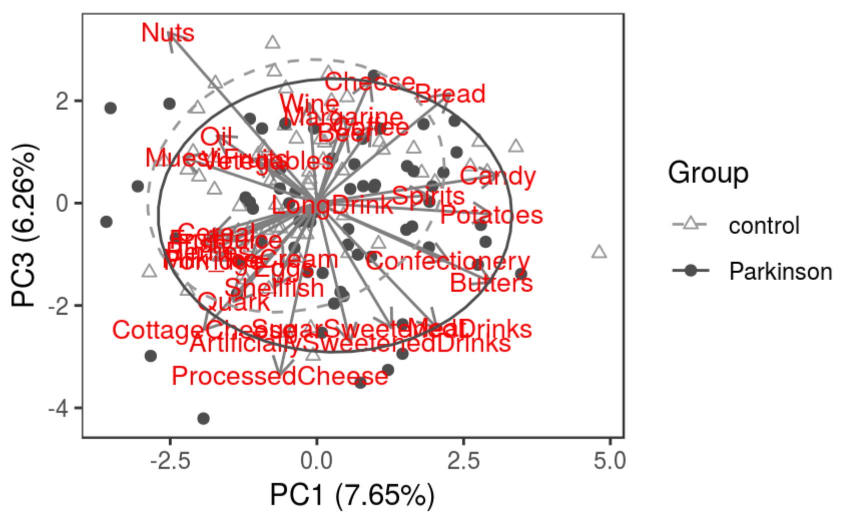
```
pdffqCat$FoodPC1<-ffqPCAf$x[,"PC1"]
pdffqCat$FoodPC2<-ffqPCAf$x[,"PC2"]
pdffqCat$FoodPC3<-ffqPCAf$x[,"PC3"]
pdffqCat$FoodPC4<-ffqPCAf$x[,"PC4"]
pdffqCat$FoodPC5<-ffqPCAf$x[,"PC5"]
```

Further visualizations of the results:

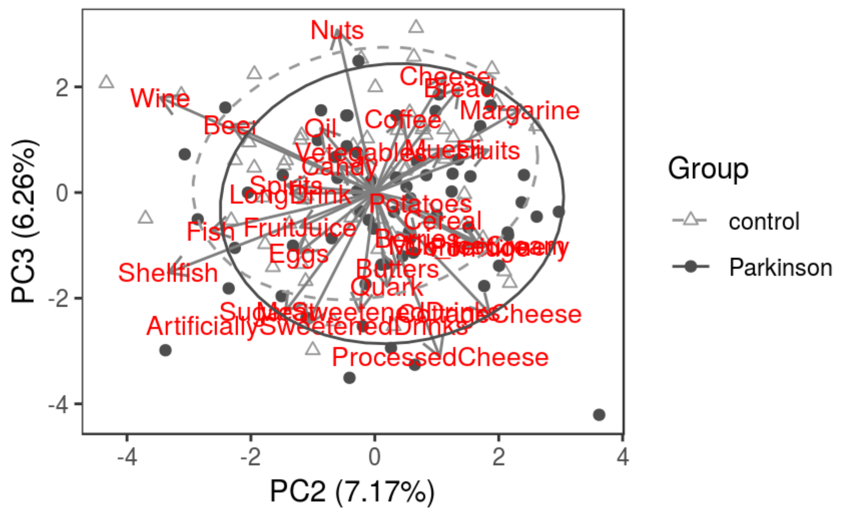
```
library("ggfortify")

pcaPlotPD <- function(pcdat, df, x, y){
  autoplot(pcdat, data = df, x = x, y = y, colour = "Parkinson", shape = "Parkinson", loadings =
      TRUE, loadings.colour = "gray50", loadings.label = TRUE, loadings.label.size = 3.5, scale = 0)
      +
    theme_bw() +
    stat_ellipse(aes(group = Parkinson, color = Parkinson, lty = Parkinson), level = 0.9) +
    scale_color_manual(values = c("gray60", "gray30"), name = "Group") +
    scale_shape_manual(values = c(2, 19), name = "Group") +
    scale_linetype_manual(values = c(2, 1), name = "Group") +
    theme(panel.grid = element_blank())
}

pcaPlotPD(ffqPCAf, pdffqEA, 1, 2)
```



```
pcaPlotPD(ffqPCAf, pdffqEA, 1, 3)
```



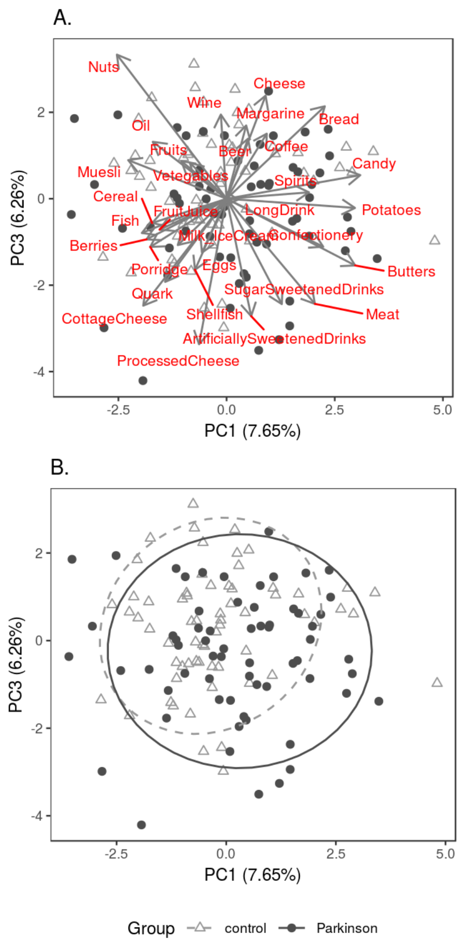
```
pcaPlotPD(ffqPCAf, pdffqEA, 2, 3)
```



Out of this biplot, the one with the 1st and 3rd PCs looks like it separates the samples best:

Figure 3

```
fig3 <- arrangeGrob(
  autoplot(ffqPCAf, data = pdffqEA, x = 1, y = 3,
           colour = "Parkinson", shape = "Parkinson",
           loadings = TRUE, loadings.colour = "gray50",
           loadings.label = TRUE, loadings.label.size = 2.75,
           scale = 0, loadings.label.repel = TRUE) +
    ggtitle("A.") +
    theme_bw(base_size = 9) +
    coord_fixed() +
    scale_color_manual(values = c("gray60", "gray30"), name = "Group") +
    scale_shape_manual(values = c(2, 19), name = "Group") +
    theme(panel.grid = element_blank(),
          legend.position = "none"),
  autoplot(ffqPCAf, data = pdffqEA, x = 1, y = 3,
           colour = "Parkinson", shape = "Parkinson",
           loadings = FALSE, loadings.colour = "gray50",
           loadings.label = FALSE, loadings.label.size = 2.75, scale = 0) +
    ggtitle("B.") +
    theme_bw(base_size = 9) +
    coord_fixed() +
    stat_ellipse(aes(group = Parkinson, color = Parkinson, lty = Parkinson), level = 0.9) +
    scale_color_manual(values = c("gray60", "gray30"), name = "Group") +
    scale_shape_manual(values = c(2, 19), name = "Group") +
    scale_linetype_manual(values = c(2, 1), name = "Group") +
    theme(panel.grid = element_blank(),
          legend.position = "bottom"),
  ncol = 1, heights=c(0.88, 1))

grid.arrange(fig3)
```



```
## Export to pdf
ggsave(fig3, filename = "Outputs/figure3.pdf", device = cairo_pdf,
       width = halfpage, height = 6.5, units = "in")
```

Based on these, PC1 will be kept for further downstream comparisons.

Microbiota data: setup and basics

Metadata and collinearity / confounders

Test what variables are correlated with Parkinson's status to decide which to include as potential confounders in PD vs control comparisons, and which are better left for the PD-only comparisons.

```
## Screen potential confounders for comparisons -----

## Add diet PC1 from the dietary analysis to the metadata table
# (using values from follow-up based diet results for both time points;
# assuming that the subjects' diet hasn't overall changed too much over two years)

# Make sure subjects match
identical(rep(rownames(pdffqCat), 2), as.character(pdclin$Subject))
```

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

```
# Add to data frame
pdclin$FoodPC1 <- rep(pdffqCat$FoodPC1, 2)

# Add diet PC to the list of variables differing between PD and C for testing
adVars <- c(diffVarsCvsPD, "FoodPC1")

# Calculate Pearson correlations for these variables
# (all are either numerical or 0/1)
adVarsCor <- cor(as.data.frame(lapply(pdclin[, c("Parkinson", adVars)], as.numeric)), use =
    "pairwise.complete.obs")

# For PD-only comparisons:
adVarsPD <- c(names(which(abs(adVarsCor[, "Parkinson"]) > 0.45)), "meds_COMT_inhibitor")
# COMT inhibitors added to this list, since only PD patients use them.
adVarsPD <- adVarsPD[-grep("Parkinson", adVarsPD)] # drop the Parkinson variable

# For PD vs C comparisons
adVarsPDvsC <- names(which(abs(adVarsCor[, "Parkinson"]) <= 0.45))
adVarsPDvsC <- adVarsPDvsC[-grep("meds_COMT_inhibitor", adVarsPDvsC)] # leave COMT out

# Add the usage choices to the table
adVarsCorSummary <- data.frame(Variable = rownames(adVarsCor), PDCor = adVarsCor[, "Parkinson"])
adVarsCorSummary$UseForPDvsCAdonis <- as.numeric(adVarsCorSummary$Variable %in% adVarsPDvsC)
adVarsCorSummary$UseForProgAdonis <- as.numeric(adVarsCorSummary$Variable %in% adVarsPD)
adVarsCorSummary <- adVarsCorSummary[-grep("Parkinson", adVarsCorSummary$Variable),]
adVarsCorSummary <- adVarsCorSummary[order(adVarsCorSummary$UseForProgAdonis,
    adVarsCorSummary$PDCor),]
rownames(adVarsCorSummary) <- NULL
```

Table 2

Export the list of clinical confounders.

```
kable(adVarsCorSummary, digits = 3)
```

Variable	PDCor	UseForPDvsCAdonis	UseForProgAdonis
history_TIA_ischemic_stroke	-0.364	1	0
meds_statin	-0.332	1	0
meds_Warfarin	-0.205	1	0
meds_ca_antagonist	-0.165	1	0
meds_ACEI_ARB	-0.160	1	0
MMSE_total	-0.159	1	0
gender	0.016	1	0
age_at_stool_collection	0.061	1	0
BMI	0.094	1	0
FoodPC1	0.145	1	0
meds_anticholinergic	0.161	1	0
Rome_III_IBS_criteria_fulfilled	0.280	1	0

Variable	PDCor	UseForPDvsCAdonis	UseForProgAdonis
Wexner_total	0.418	1	0
RLS	0.419	1	0
Rome_III_constip_defec_sumscore_9.15	0.432	1	0
sniffinsticks	-0.769	0	1
meds_COMT_inhibitor	0.307	0	1
GDS_15	0.491	0	1
SCS_PD_total	0.520	0	1
SDQ_total	0.529	0	1
RBDSQ	0.580	0	1
NMSS_total	0.607	0	1
meds_dopa	0.662	0	1
NMSQuest_total	0.714	0	1
meds_MAO_inhibitor	0.736	0	1
LED	0.753	0	1
meds_dopamine_agonist	0.807	0	1

```r
# Export
write.csv(adVarsCorSummary, "Outputs/table_2.csv")
```

Data setup

Full data

Set up the final microbiota data objects for the rest of the analyses:

```r
## Adding a sequencing run variable
run_batch <- read.csv("Inputs/run_variable.csv")

# Verify that the names match the clinical data table; add to clinical data
if(identical(as.character(run_batch$Sample), rownames(pdclin))){
  pdclin$Run <- run_batch$Run
} else {
  print("Names don't match!")
}

## Match the 16S table to the clinical metadata
pd16s <- pd16s[,rownames(pdclin)]

## Create the final phyloseq object
newPD <- phyloseq(otu_table(pd16s, taxa_are_rows = TRUE), tax_table(pdtax), sample_data(pdclin))

# Delete OTUs with 0 reads in the remaining samples
newPD <- subset_taxa(newPD, taxa_sums(newPD) > 0)

# Add a combined timepoint + PD variable:
sample_data(newPD)$PD_TP <- factor(paste(sample_data(newPD)$Parkinson, sample_data(newPD)$Timepoint,
    sep = "_"), levels = c("control_baseline", "Parkinson_baseline", "control_followup",
    "Parkinson_followup"))

## Make genus and family level objects
newPDgen <- collapseTaxLevel(newPD, level = "Genus", fixUnclassifieds = FALSE)
newPDfam <- collapseTaxLevel(newPD, level = "Family", fixUnclassifieds = FALSE)
```

Progression

A separate phyloseq object for PD-only progression comparisons:

```r
progmeta <- subset(pdclin, rownames(pdclin) %in% prog_samples)

progPhy <- phyloseq(otu_table(pd16s[, colnames(pd16s) %in% rownames(progmeta)], taxa_are_rows =
    TRUE), tax_table(pdtax), sample_data(progmeta))

# combined timepoint + progression variable:
```

```r
sample_data(progPhy)$ProgTP <- factor(paste(sample_data(progPhy)$ProgCat,
    sample_data(progPhy)$Timepoint, sep = "_"),
    levels = c("stable_baseline", "progressed_baseline", "stable_followup", "progressed_followup"))

## Make genus and family level objects
progPhyGen <- collapseTaxLevel(progPhy, level = "Genus", fixUnclassifieds = FALSE)
progPhyFam <- collapseTaxLevel(progPhy, level = "Family", fixUnclassifieds = FALSE)
```

PD Phenotype

A separate phyloseq object for PD-only comparisons of PD phenotypes (TD vs PIGD); dropping patients with a mixed (MX) phenotype:

```r
# Rename the target variable to something less wordy:
colnames(sample_data(progPhy))[which(colnames(sample_data(progPhy)) ==
    "td_pigd_subtype_cutoff_1.5_1.0_jankovic_ON")] <- "JankovicClass"

# Drop MX subjects:
progPhyPhe <- subset_samples(progPhy, JankovicClass != "MX")
progPhyPheGen <- collapseTaxLevel(progPhyPhe, level = "Genus", fixUnclassifieds = FALSE)
progPhyPheFam <- collapseTaxLevel(progPhyPhe, level = "Family", fixUnclassifieds = FALSE)

# Check how many samples are left:
kable_styling(kable(table(sample_data(progPhyPhe)$JankovicClass,
    sample_data(progPhyPhe)$Timepoint)), full_width = FALSE)
```

	baseline	followup
PIGD	28	35
TD	21	20

Diet

A phyloseq object with metadata that includes all nutrients, probiotic use, the list of non-overlapping food items used for the PCA, and the first five PCs from the diet PCA, for running comparisons of diet vs microbiota:

```r
# List of variables:
ffqVars <- c(colnames(pdffqCat[2:47]), ffqpcaVars, "Probio_per1kkc", "FoodPC1", paste("FoodPC", 2:5,
    sep = ""))

# Put together metadata
ffqmeta <- pdffqCat[, ffqVars]
rownames(ffqmeta) <- paste(rownames(ffqmeta), "N", sep = "")
ffqmeta <- cbind(pdclin[rownames(ffqmeta),], ffqmeta[,-grep("FoodPC1", colnames(ffqmeta))])
# (PC1 was already added to the clinical metadata earlier; left out to avoid duplicating it)

# Create phyloseq object
ffqPhy <- phyloseq(otu_table(pd16s[, colnames(pd16s) %in% rownames(ffqmeta)], taxa_are_rows = TRUE),
    tax_table(pdtax), sample_data(ffqmeta))
```

Basic statistics and plots of data

Number of sequence reads in final data set:

```r
# Total number of reads:
sum(sample_sums(newPD))
```

```
## [1] 18867278
```

```r
# Summary statistics
summary(sample_sums(newPD))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2201   50032   73078   73700   98685  262621
```

Numbers of taxa on different taxonomic levels:

```
print(paste("OTUs: ", ntaxa(newPD), ", genera: ", ntaxa(subset_taxa(newPDgen, Genus !=
    "unclassified")), ", families: ", ntaxa(subset_taxa(newPDfam, Family != "unclassified")),
    sep=""))
```

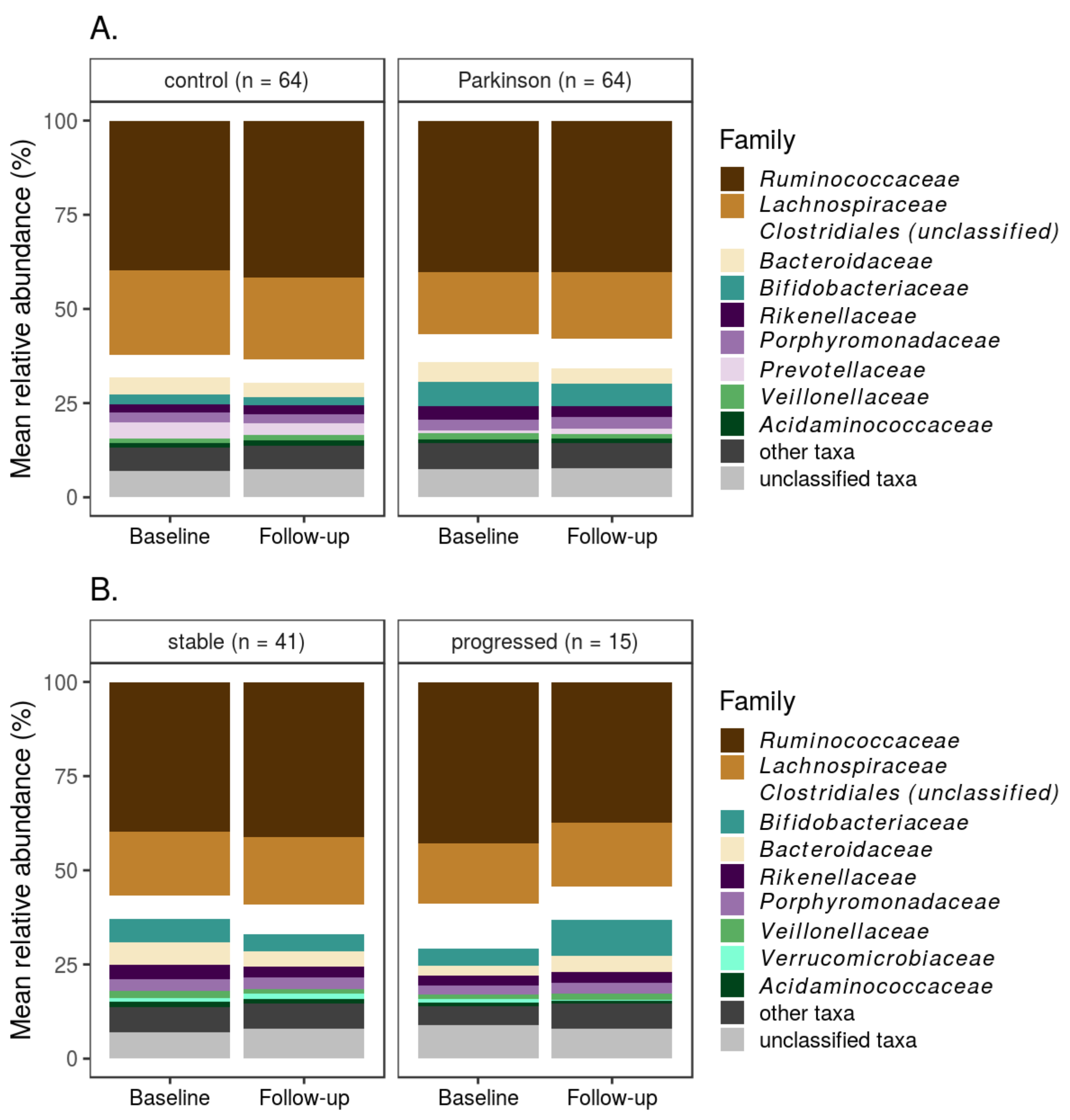
```
## [1] "OTUs: 2836, genera: 198, families: 77"
```

Bar charts of the most common bacterial families in the data:

Figure 4

```
# Set up data for PD vs control plot
bcdataF <- relAbundChart(collapseTaxLevel(newPD, level = "Family", maxUnclassifiedLevel = "Order"),
    byVariable = "PD_TP", taxaCount = 11, table = TRUE)
bcdataF$Taxon <- factor(gsub("_", " ", rownames(bcdataF)))
bcdataF <- melt(bcdataF)
bcdataF$Taxon <- factor(bcdataF$Taxon, levels = unique(bcdataF$Taxon))
bcdataF$PD <- factor(gsub("_.*", "", bcdataF$variable), levels = c("control", "Parkinson"))
bcdataF$TP <- factor(gsub(".*_", "", bcdataF$variable), levels = c("baseline", "followup"))
levels(bcdataF$TP) <- c("Baseline", "Follow-up")

bccolsF <- c(brewer.pal(name = "BrBG", n = 10)[c(1, 3, 4, 5, 8)], brewer.pal(name = "PRGn", n =
    10)[c(1, 3, 5, 8, 10)], "gray25", "gray75")
names(bccolsF) <- unique(bcdataF$Taxon)

# Set up data for progression plot
bcdataProgFam <- relAbundChart(collapseTaxLevel(progPhy, level = "Family", maxUnclassifiedLevel =
    "Order"), byVariable = "ProgTP", taxaCount = 11, table = TRUE)
bcdataProgFam$Taxon <- factor(gsub("_", " ", rownames(bcdataProgFam)))
bcdataProgFam <- melt(bcdataProgFam)
bcdataProgFam$Taxon <- factor(bcdataProgFam$Taxon, levels = unique(bcdataProgFam$Taxon))
bcdataProgFam$Prog <- factor(gsub("_.*", "", bcdataProgFam$variable), levels = c("stable",
    "progressed"))
levels(bcdataProgFam$Prog) <- c("stable", "progressed")
bcdataProgFam$TP <- factor(gsub(".*_", "", bcdataProgFam$variable), levels = c("baseline",
    "followup"))

progFamCols <- as.character(unique(bcdataProgFam$Taxon))
bccolsProgF <- bccolsF[match(progFamCols, names(bccolsF))]
# one there's one family that's not in the other color list, add:
names(bccolsProgF)[is.na(bccolsProgF)] <- progFamCols[!(progFamCols %in% names(bccolsF))]
bccolsProgF[is.na(bccolsProgF)] <- c("aquamarine")

## Plot the two together

# Facet labels with numbers of samples
labelsPDC <- c(control = paste("control (n = ",
                               nrow(subset(pdclin, Parkinson == "control"))/2, ")", sep = ""),
               Parkinson = paste("Parkinson (n = ",
                                 nrow(subset(pdclin, Parkinson == "Parkinson"))/2, ")", sep = ""))
labelsProg <- c(stable = paste("stable (n = ",
                               nrow(subset(progmeta, ProgCat == "stable"))/2, ")", sep = ""),
                progressed = paste("progressed (n = ",
                                   nrow(subset(progmeta, ProgCat == "progressed"))/2, ")", sep = ""))

# Shorten one unclassified label
names(bccolsF)[3] <- gsub(" \\(unclassified.*", " (unclassified)", names(bccolsF)[3])
names(bccolsProgF)[3] <- gsub(" \\(unclassified.*", " (unclassified)", names(bccolsProgF)[3])
```

```r
# Italics for families in legends
taxlabelsPDC <- parse(text = c(paste('italic(`', names(bccolsF)[1:10],'`)', sep = ""),
                               paste('plain(`', names(bccolsF)[11:12],'`)', sep="")))
taxlabelsProg <- parse(text = c(paste('italic(`', names(bccolsProgF)[1:10],'`)', sep = ""),
                                paste('plain(`', names(bccolsProgF)[11:12],'`)', sep="")))

# Make the plot
fig4 <- arrangeGrob(
  ggplot(bcdataF, aes(x = TP, y = value, fill = Taxon)) +
    geom_bar(stat = "identity", position = "stack") +
    xlab(NULL) +
    ylab("Mean relative abundance (%)") +
    scale_fill_manual(values = bccolsF, labels = taxlabelsPDC, name = "Family") +
    guides(fill = guide_legend(ncol = 1)) +
    ggtitle("A.") +
    theme_bw(base_size = 11) +
    facet_grid(~ PD, labeller = labeller(PD = labelsPDC)) +
    scale_x_discrete(labels = c("Baseline", "Follow-up")) +
    theme(axis.text.x = element_text(color = "black"),
          strip.background = element_rect(fill = "white"),
          panel.grid = element_blank(),
          legend.text.align = 0,
          legend.key.height = unit(4, "mm"),
          legend.key.width = unit(4, "mm"),
          legend.text = element_text(size = 9),
          legend.margin = ggplot2::margin(r = 7, unit = "mm")),
  ggplot(bcdataProgFam, aes(x = TP, y = value, fill = Taxon)) +
    geom_bar(stat = "identity", position = "stack") +
    facet_grid(~ Prog, labeller = labeller(Prog = labelsProg)) +
    xlab(NULL) +
    ylab("Mean relative abundance (%)") +
    scale_fill_manual(values = bccolsProgF, labels = taxlabelsProg, name = "Family") +
    guides(fill = guide_legend(ncol = 1)) +
    ggtitle("B.") +
    theme_bw(base_size = 11) +
    scale_x_discrete(labels = c("Baseline", "Follow-up")) +
    theme(axis.text.x = element_text(color = "black"),
          strip.background = element_rect(fill = "white"),
          panel.grid = element_blank(),
          legend.text.align = 0,
          legend.key.height = unit(4, "mm"),
          legend.key.width = unit(4, "mm"),
          legend.text = element_text(size = 9),
          legend.margin = ggplot2::margin(r = 7, unit = "mm")),
  ncol = 1)

grid.arrange(fig4)
```



```r
## Export to pdf
ggsave(fig4, filename = "Outputs/figure4.pdf", device = cairo_pdf,
       width = fullpage, height = maxhi*0.75, units = "in")
```

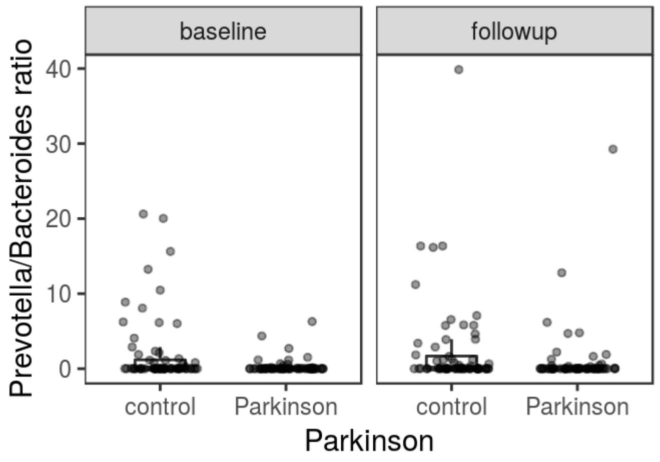
Basic comparisons for commonly used ratios for specific bacterial phyla and genera:

```r
# Make a new data frame for this
ptbdf <- as.data.frame(matrix(nrow = nrow(sample_data(newPDgen)), ncol = 0))

# Collapse microbiota data to phylum level
newPDphy <- collapseTaxLevel(newPD, level = "Phylum", fixUnclassifieds = FALSE)

# clinical data
rownames(ptbdf) <- rownames(sample_data(newPD))
ptbdf$Parkinson <- sample_data(newPD)$Parkinson
ptbdf$Timepoint <- sample_data(newPD)$Timepoint
ptbdf$Progression <- sample_data(newPD)$ProgCat

# Firmicutes/Bacteroides ratio
ptbdf$FtoB <- as.vector(otu_table(newPDphy)["Firmicutes"]) /
    as.vector(otu_table(newPDphy)["Bacteroidetes"])

# Prevotella/Bacteroides ratio
```

```r
ptbdf$PtoB <- as.vector(otu_table(newPDgen)["Prevotella"]) /
    as.vector(otu_table(newPDgen)["Bacteroides"])

# Separate subset for PD progression
ptbdfProg <- subset(ptbdf, rownames(ptbdf) %in% prog_samples)

## Test for statistically significant differences (timepoints separated)

# PD vs control
ptb_sigs <- data.frame(
  Baseline = c(
    wilcox.test(data = subset(ptbdf, Timepoint == "baseline"), FtoB ~ Parkinson)$p.value,
    wilcox.test(data = subset(ptbdf, Timepoint == "baseline"), PtoB ~ Parkinson)$p.value),
  Followup = c(
    wilcox.test(data = subset(ptbdf, Timepoint == "followup"), FtoB ~ Parkinson)$p.value,
    wilcox.test(data = subset(ptbdf, Timepoint == "followup"), PtoB ~ Parkinson)$p.value),
  row.names = c("Firmicutes_to_Bacteroidetes", "Prevotella_to_Bacteroides"))

kable_styling(kable(ptb_sigs, digits = 3), full_width = FALSE)
```

	Baseline	Followup
Firmicutes_to_Bacteroidetes	0.832	0.670
Prevotella_to_Bacteroides	0.052	0.011

```r
# Plot the Prevotella/Bacteroidetes ratio
ggplot(ptbdf, aes(x = Parkinson, y = PtoB)) +
  geom_boxplot(width = 0.4, outlier.shape = NA) +
  geom_jitter(width = 0.3, alpha = 0.4, size = 1) +
  facet_grid(~Timepoint) +
  theme_bw() +
  ylab("Prevotella/Bacteroides ratio") +
  theme(panel.grid = element_blank())
```



The *Firmicutes/Bacteroidetes* ratio does not differ between groups, but *Prevotella/Bacteroides* is significantly different at follow-up, and borderline ($0.1 > p > 0.05$) at baseline.

What about the PD progression categories?

```r
ptb_prog_sigs <- data.frame(
  Baseline = c(
    wilcox.test(data = subset(ptbdfProg, Timepoint == "baseline"), FtoB ~ Progression)$p.value,
    wilcox.test(data = subset(ptbdfProg, Timepoint == "baseline"), PtoB ~ Progression)$p.value),
  Followup = c(
    wilcox.test(data = subset(ptbdfProg, Timepoint == "followup"), FtoB ~ Progression)$p.value,
    wilcox.test(data = subset(ptbdfProg, Timepoint == "followup"), PtoB ~ Progression)$p.value),
  row.names = c("Firmicutes_to_Bacteroidetes", "Prevotella_to_Bacteroides"))
```

```
kable_styling(kable(ptb_prog_sigs, digits = 3), full_width = FALSE)
```

	Baseline	Followup
Firmicutes_to_Bacteroidetes	0.012	0.840
Prevotella_to_Bacteroides	0.511	0.107

The *Firmicutes/Bacteroidetes* ratio differs significantly at baseline, but not at follow-up; there is no difference in the *Prevotella/Bacteroides* ratio at either timepoint.

Enterotypes

Enterotyping was run separately with the online (tool)[http://enterotypes.org/] using microbiota data summarized to genus level. Comparisons for this data:

```
# Import enterotyping results
pdET <- read.table("Inputs/enterotyping_results.txt", header = TRUE, sep = ",")
# Rename enterotypes with full names of dominant taxon:
levels(pdET$ET) <- c("Bacteroides", "Firmicutes", "Prevotella")

# Add metadata variables
pdET$Subject <- factor(substr(pdET$Sample, start = 1, stop = 5))
pdET$Parkinson <- factor(substr(pdET$Sample, start = 1, stop = 1))
levels(pdET$Parkinson) <- c("control", "Parkinson")
pdET$Timepoint <- factor(substr(pdET$Sample, start = 6, stop = 6), levels=c("O", "N"))
levels(pdET$Timepoint) <- c("baseline", "follow-up")

# Summary table of frequencies
pdETsummaries <- as.data.frame(table(pdET$Parkinson, pdET$ET, pdET$Timepoint))
colnames(pdETsummaries) <- c("Parkinson", "ET", "Timepoint", "Freq")
levels(pdETsummaries$Timepoint) <- c("baseline", "follow-up")

# Table and summary with PD split by progression category
pdETprog <- data.frame(pdET, ProgCat = pdclin[as.character(pdET$Sample), "ProgCat"])
pdETprog <- subset(pdETprog, !is.na(pdETprog$ProgCat))
pdETprog <- rbind(pdETprog[, c("Sample", "ET", "ProgCat", "Timepoint")], data.frame(subset(pdET,
    Parkinson=="control")[, c("Sample", "ET", "Timepoint")], ProgCat="control"))
pdETprog$ProgCat <- factor(pdETprog$ProgCat, levels = c("control", "stable", "progressed"))

pdETsummariesProg <- as.data.frame(table(pdETprog$ProgCat, pdETprog$ET, pdETprog$Timepoint))
colnames(pdETsummariesProg) <- c("ProgCat", "ET", "Timepoint", "Freq")

# Statistical significances for the differences in distributions

set.seed(45877298)

# Table for collecting results
chisqETres <- as.data.frame(matrix(ncol = 3, nrow = 4))
colnames(chisqETres) <- c("Comparison", "p_baseline", "p_followup")
chisqETres$Comparison <- c("PD vs control", "prog vs stable", "prog vs control", "stable vs control")

# PD vs C
chisqETres[1, 2] <- chisq.test(as.matrix(table(subset(pdET, Timepoint == "baseline")[,
    c("Parkinson", "ET")])))$p.value
chisqETres[1, 3] <- chisq.test(as.matrix(table(subset(pdET, Timepoint == "follow-up")[,
    c("Parkinson", "ET")])))$p.value

# Progressed vs stable
chisqETres[2, 2] <- chisq.test(as.matrix(table(subset(pdETprog, Timepoint == "baseline" & ProgCat !=
    "control")[, c("ProgCat", "ET")]))[2:3, ], simulate.p.value = TRUE)$p.value
chisqETres[2, 3] <- chisq.test(as.matrix(table(subset(pdETprog, Timepoint == "follow-up" & ProgCat
    != "control")[, c("ProgCat", "ET")]))[2:3, ], simulate.p.value = TRUE)$p.value
```

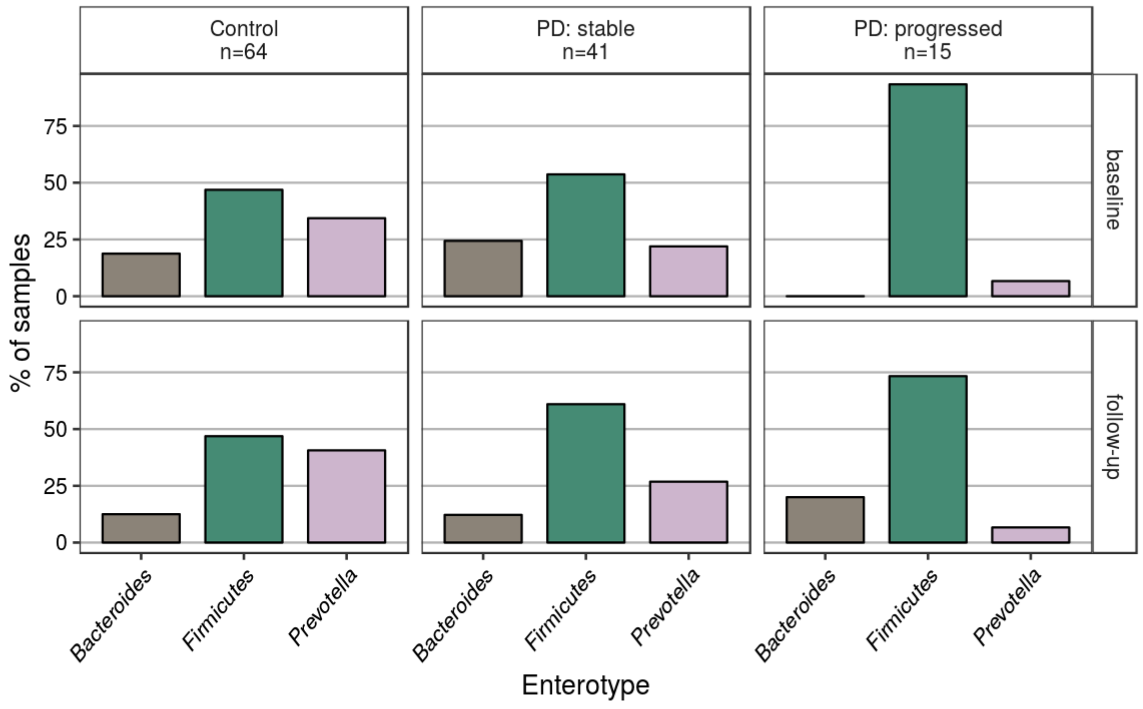
```r
## Progressed vs controls
chisqETres[3, 2] <- chisq.test(as.matrix(table(subset(pdETprog, Timepoint == "baseline" & ProgCat !=
    "stable")[, c("ProgCat", "ET")]))[c(1,3),], simulate.p.value = TRUE)$p.value
chisqETres[3, 3] <- chisq.test(as.matrix(table(subset(pdETprog, Timepoint == "follow-up" & ProgCat
    != "stable")[, c("ProgCat", "ET")]))[c(1,3),], simulate.p.value = TRUE)$p.value

# Stable vs controls
chisqETres[4, 2] <- chisq.test(as.matrix(table(subset(pdETprog, Timepoint == "baseline" & ProgCat !=
    "progressed")[, c("ProgCat", "ET")]))[1:2, ], simulate.p.value = TRUE)$p.value
chisqETres[4, 3] <- chisq.test(as.matrix(table(subset(pdETprog, Timepoint == "follow-up" & ProgCat
    != "progressed")[, c("ProgCat", "ET")]))[1:2, ], simulate.p.value = TRUE)$p.value

kable_styling(kable(chisqETres, digits = 4), full_width = FALSE)
```

Comparison	p_baseline	p_followup
PD vs control	0.0441	0.0250
prog vs stable	0.0160	0.2909
prog vs control	0.0005	0.0430
stable vs control	0.3893	0.3053

Figure 5

A plot of the enterotype distributions:

```r
# Reorganizing the tables

## Control + progression into percentages
pdETsummariesProgPerc <- dcast(pdETsummariesProg, ProgCat + Timepoint ~ ET, value.var = "Freq")
pdETsummariesProgPerc[, c("Bacteroides", "Firmicutes", "Prevotella")] <-
    prop.table(as.matrix(pdETsummariesProgPerc[, c("Bacteroides", "Firmicutes", "Prevotella")]), 1)
    * 100
pdETsummariesProgPerc <- melt(pdETsummariesProgPerc)

levels(pdETsummariesProgPerc$ProgCat) <- c(paste("Control\nn=", nrow(subset(pdETprog, ProgCat ==
    "control")) / 2, sep = ""), paste("PD: stable\nn=", nrow(subset(pdETprog, ProgCat == "stable"))
    / 2, sep = ""), paste("PD: progressed\nn=", nrow(subset(pdETprog, ProgCat == "progressed")) / 2,
    sep = ""))

# Plot

fig5 <- ggplot(pdETsummariesProgPerc, aes(x = variable, y = value, fill = variable)) +
  geom_bar(stat = "identity", position = position_dodge(), color = "black", width = 0.75, size =
      0.4) +
  theme_bw(base_size = 10) +
  ylab("% of samples") +
  xlab("Enterotype") +
  facet_grid(Timepoint ~ ProgCat) +
  scale_fill_manual(values = c("antiquewhite4", "aquamarine4", "thistle3"), name = "Enterotype") +
  theme(panel.grid.major.x = element_blank(),
        panel.grid.minor.x = element_blank(),
        panel.grid.major.y = element_line(color = "gray70", size = 0.4),
        panel.grid.minor.y = element_blank(),
        strip.background = element_rect(fill = "white"),
        legend.position = "none",
        axis.text.x  =  element_text(color = "black", face = "italic", angle = 50, hjust = 1, vjust
            = 1),
        axis.text.y  =  element_text(color = "black"))
fig5
```



```
## Export to pdf
ggsave(fig5, filename = "Outputs/figure5.pdf", device = cairo_pdf,
       width = halfpage, height = 2.75, units = "in")
```

Alpha diversity analyses

PD vs control

Correlations with confounders

Calculate Pearson's correlations for all variables and three alpha diversity measures:

```
# Data frame with alpha diversity data
pdDivs <- cbind(pdclin, estimate_richness(newPD, measures = c("Observed", "Shannon", "InvSimpson")))
pdDivs$Parkinson <- factor(pdDivs$Parkinson, labels = c("control", "Parkinson"))

# Calculating correlations for all variables (except those with lots of missing values)

# List of variables to use
divVars <- colnames(pdDivs)[which(colSums(is.na(pdDivs))<50)]
divVars <- divVars[-which(divVars %in% c("Observed", "Shannon", "InvSimpson"))]

# Run correlations
divCors <- t(sapply(divVars, function(x) c(
  unname(unlist(cor.test(y = as.numeric(pdDivs[,x]), x = pdDivs$Observed)[c("estimate",
      "p.value")])),
  unname(unlist(cor.test(y = as.numeric(pdDivs[,x]), x = pdDivs$Shannon)[c("estimate", "p.value")])),
  unname(unlist(cor.test(y = as.numeric(pdDivs[,x]), x = pdDivs$InvSimpson)[c("estimate",
      "p.value")])))))
colnames(divCors) <- c("ObsCor", "ObsP", "ShanCor", "ShanP", "InvSCor", "InvSP")
divCors <- data.frame(Var = divVars, divCors)

# Multiple comparison corrections
divCors$ObsPAdj <- p.adjust(divCors$ObsP, method  =  "fdr")
divCors$ShanPAdj <- p.adjust(divCors$ShanP, method  =  "fdr")
divCors$InvSPAdj <- p.adjust(divCors$InvSP, method  =  "fdr")
```

Export the results for variables that are significant for at least one comparison for the manuscript:

Table 5

```r
divCorExp <- rbind(divCors[c("Parkinson", "Timepoint"),], subset(divCors, rowSums(divCors[, c(3, 5,
    7)] < 0.05) > 0))

kable_styling(kable(divCorExp, digits = 3), font_size = 10)
```

	Var	ObsCor	ObsP	ShanCor	ShanP	InvSCor	InvSP	ObsPAdj	ShanPAdj	InvSPAdj
Parkinson	Parkinson	0.005	0.942	-0.023	0.718	-0.026	0.678	0.990	0.908	0.915
Timepoint	Timepoint	0.104	0.098	0.042	0.506	0.055	0.382	0.496	0.832	0.873
BMI	BMI	-0.200	0.001	-0.159	0.012	-0.129	0.041	0.057	0.190	0.417
history_lactoseintolerance	history_lactoseintolerance	0.129	0.039	0.116	0.063	0.108	0.086	0.280	0.517	0.579
history_colon_irritabile	history_colon_irritabile	0.149	0.017	0.116	0.064	0.131	0.036	0.232	0.517	0.417
history_hypothyreosis	history_hypothyreosis	0.161	0.010	0.078	0.212	0.031	0.618	0.199	0.832	0.894
history_hyperthyreosis	history_hyperthyreosis	-0.034	0.591	-0.143	0.022	-0.110	0.078	0.970	0.299	0.576
history_appendectomy	history_appendectomy	-0.054	0.392	-0.133	0.034	-0.110	0.078	0.970	0.388	0.576
history_hernia_repair	history_hernia_repair	0.013	0.835	0.101	0.108	0.135	0.031	0.990	0.729	0.417
history_CAD	history_CAD	0.105	0.096	0.168	0.007	0.169	0.007	0.496	0.153	0.181
history_ENT_surgery	history_ENT_surgery	0.169	0.007	0.189	0.002	0.163	0.009	0.184	0.096	0.181
tobacco_100_in_life	tobacco_100_in_life	0.092	0.140	0.167	0.008	0.140	0.025	0.630	0.153	0.403
Wexner_total	Wexner_total	0.151	0.015	0.036	0.569	0.020	0.744	0.232	0.832	0.972
Rome_III_constip_defec_sumscore_9.15	Rome_III_constip_defec_sumscore_9.15	0.212	0.001	0.031	0.616	0.010	0.872	0.052	0.832	0.985
Rome_III_IBS_criteria_fulfilled	Rome_III_IBS_criteria_fulfilled	0.128	0.040	0.033	0.597	-0.007	0.913	0.280	0.832	0.985
meds_ca_antagonist	meds_ca_antagonist	0.136	0.030	0.216	0.001	0.217	0.000	0.266	0.042	0.039
meds_thyroxine	meds_thyroxine	0.143	0.022	0.051	0.412	0.010	0.874	0.258	0.832	0.985
selegeline_mg	selegeline_mg	-0.043	0.494	-0.124	0.048	-0.163	0.009	0.970	0.487	0.181
pramipexole_mg	pramipexole_mg	-0.139	0.026	-0.028	0.657	-0.001	0.985	0.265	0.845	0.985
FoodPC1	FoodPC1	-0.127	0.042	-0.087	0.164	-0.055	0.379	0.280	0.832	0.873

```r
## Export
write.csv(divCorExp, "Outputs/table_5.csv")
```

A handful of variables have a significant p-value for all three indices (although only one, Ca channel antagonist use, also has significant multiple comparison corrected p for any of them).

The Parkinson status variable was not correlated with any of the diversity indices in a statistically significant manner. Contrast the Parkinson status and timepoint variables separately anyway:

Differences between timepoints and PD status

```r
## Only timepoints / only PD status, without subsetting
adiv_overall <- data.frame(
  # Differences between timepoints
  Timepoint = c(
    wilcox.test(data = pdDivs, Observed ~ Timepoint)$p.value,
    wilcox.test(data = pdDivs, Shannon ~ Timepoint)$p.value,
    wilcox.test(data = pdDivs, InvSimpson ~ Timepoint)$p.value),
  # Differences between PD and controls
  PD_vs_C = c(
    wilcox.test(data = pdDivs, Observed ~ Parkinson)$p.value,
    wilcox.test(data = pdDivs, Shannon ~ Parkinson)$p.value,
    wilcox.test(data = pdDivs, InvSimpson ~ Parkinson)$p.value),
  row.names = c("Observed", "Shannon", "Inverse Simpson"))

kable_styling(kable(adiv_overall, digits = 3), full_width = FALSE)
```

	Timepoint	PD_vs_C
Observed	0.123	0.930
Shannon	0.515	0.690
Inverse Simpson	0.273	0.635

```r
## PD vs control with the timepoints separated

# Generic function for timepoint-separated contrasting of all three indices:
adivTab <- function(df, var){
  bl_df <- subset(df, Timepoint == "baseline")
  fu_df <- subset(df, Timepoint == "followup")

  adiv_res <- data.frame(
  Baseline = c(
    wilcox.test(data = bl_df, formula(paste("Observed ~", var)))$p.value,
    wilcox.test(data = bl_df, formula(paste("Shannon ~", var)))$p.value,
    wilcox.test(data = bl_df, formula(paste("InvSimpson ~", var)))$p.value),
  Followup = c(
    wilcox.test(data = fu_df, formula(paste("Observed ~", var)))$p.value,
    wilcox.test(data = fu_df, formula(paste("Shannon ~", var)))$p.value,
    wilcox.test(data = fu_df, formula(paste("InvSimpson ~", var)))$p.value),
  row.names = c("Observed", "Shannon", "Inverse Simpson"))

  return(adiv_res)
}

adiv_pdc_tpsep <- adivTab(pdDivs, "Parkinson")

kable_styling(kable(adiv_pdc_tpsep, digits = 3), full_width = FALSE)
```

	Baseline	Followup
Observed	0.830	0.909
Shannon	0.922	0.677
Inverse Simpson	0.926	0.629

```r
## Difference in timepoint within group

adiv_tp_pdcsep <- data.frame(
# PD patients only
  PD_patients = c(
    wilcox.test(data = subset(pdDivs, Parkinson == "Parkinson"), Observed ~ Timepoint)$p.value,
    wilcox.test(data = subset(pdDivs, Parkinson == "Parkinson"), Shannon ~ Timepoint)$p.value,
    wilcox.test(data = subset(pdDivs, Parkinson == "Parkinson"), InvSimpson ~ Timepoint)$p.value),
# Control subjects only
  Controls = c(
    wilcox.test(data = subset(pdDivs, Parkinson == "control"), Observed ~ Timepoint)$p.value,
    wilcox.test(data = subset(pdDivs, Parkinson == "control"), Shannon ~ Timepoint)$p.value,
    wilcox.test(data = subset(pdDivs, Parkinson == "control"), InvSimpson ~ Timepoint)$p.value),
  row.names = c("Observed", "Shannon", "Inverse Simpson"))

kable_styling(kable(adiv_tp_pdcsep, digits = 3), full_width = FALSE)
```

	PD_patients	Controls
Observed	0.337	0.228
Shannon	0.802	0.522
Inverse Simpson	0.515	0.473

None of these comparisons suggest any differences in alpha diversity between timepoints or between the PD and control groups.

PD vs control and confounders

BMI

Some further exploration of the confounders that were significant:

```
# PD status and the three variables that had p < 0.05 for all three indices,
# looking for possible interactions

summary(lm(Observed ~ Parkinson * BMI, pdDivs)) # nothing for PD
```

```
##
## Call:
## lm(formula = Observed ~ Parkinson * BMI, data = pdDivs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -243.070  -57.991   -0.824   66.579  224.806
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              582.092     63.857   9.116  < 2e-16 ***
## ParkinsonParkinson       -84.872     83.204  -1.020  0.30870
## BMI                       -6.891      2.394  -2.878  0.00435 **
## ParkinsonParkinson:BMI     3.328      3.080   1.081  0.28097
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 95.85 on 247 degrees of freedom
##   (5 observations deleted due to missingness)
## Multiple R-squared:  0.04511,    Adjusted R-squared:  0.03351
## F-statistic: 3.889 on 3 and 247 DF,  p-value: 0.009641
```

```
summary(lm(InvSimpson ~ Parkinson * BMI, pdDivs)) # an interaction?
```

```
##
## Call:
## lm(formula = InvSimpson ~ Parkinson * BMI, data = pdDivs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.169  -5.568  -1.374   5.303  28.526
##
## Coefficients:
```

```
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              35.9667     5.1870   6.934 3.56e-11 ***
## ParkinsonParkinson     -14.8442     6.7585  -2.196  0.02899 *
## BMI                      -0.5731     0.1945  -2.947  0.00352 **
## ParkinsonParkinson:BMI   0.5402     0.2502   2.159  0.03178 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.786 on 247 degrees of freedom
##   (5 observations deleted due to missingness)
## Multiple R-squared:  0.0355, Adjusted R-squared:  0.02378
## F-statistic:  3.03 on 3 and 247 DF,  p-value: 0.03003
```

```
summary(lm(Shannon ~ Parkinson * BMI, pdDivs)) # also an interaction?
```

```
##
## Call:
## lm(formula = Shannon ~ Parkinson * BMI, data = pdDivs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.05241 -0.23257  0.00035  0.27141  0.71003
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             4.579313   0.227206  20.155  < 2e-16 ***
## ParkinsonParkinson     -0.712663   0.296043  -2.407 0.016806 *
## BMI                    -0.029233   0.008519  -3.431 0.000704 ***
## ParkinsonParkinson:BMI  0.026078   0.010957   2.380 0.018074 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.341 on 247 degrees of freedom
##   (5 observations deleted due to missingness)
## Multiple R-squared:  0.0476, Adjusted R-squared:  0.03603
## F-statistic: 4.115 on 3 and 247 DF,  p-value: 0.007148
```

```
summary(lm(Observed ~ Parkinson * history_ENT_surgery, pdDivs)) # no
```

```
##
## Call:
## lm(formula = Observed ~ Parkinson * history_ENT_surgery, data = pdDivs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -230.971  -62.217   -8.074   68.166  257.577
##
## Coefficients:
##                                         Estimate Std. Error t value
## (Intercept)                              387.149      9.679  39.998
## ParkinsonParkinson                        10.822     13.655   0.793
## history_ENT_surgery1                      63.851     21.075   3.030
## ParkinsonParkinson:history_ENT_surgery1  -46.399     30.014  -1.546
##                                         Pr(>|t|)
## (Intercept)                               <2e-16 ***
## ParkinsonParkinson                        0.4288
## history_ENT_surgery1                      0.0027 **
## ParkinsonParkinson:history_ENT_surgery1   0.1234
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 97.27 on 252 degrees of freedom
## Multiple R-squared:  0.03762,    Adjusted R-squared:  0.02617
## F-statistic: 3.284 on 3 and 252 DF,  p-value: 0.02146
```

```r
summary(lm(InvSimpson ~ Parkinson * history_ENT_surgery, pdDivs)) # no
```

```
##
## Call:
## lm(formula = InvSimpson ~ Parkinson * history_ENT_surgery, data = pdDivs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.213  -5.640  -1.635   4.943  31.232
##
## Coefficients:
##                                         Estimate Std. Error t value
## (Intercept)                             20.03663    0.77823  25.746
## ParkinsonParkinson                      -0.36549    1.09788  -0.333
## history_ENT_surgery1                     3.21030    1.69446   1.895
## ParkinsonParkinson:history_ENT_surgery1 -0.09712    2.41321  -0.040
##                                         Pr(>|t|)
## (Intercept)                               <2e-16 ***
## ParkinsonParkinson                        0.7395
## history_ENT_surgery1                      0.0593 .
## ParkinsonParkinson:history_ENT_surgery1   0.9679
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.821 on 252 degrees of freedom
## Multiple R-squared:  0.02721,    Adjusted R-squared:  0.01563
## F-statistic: 2.349 on 3 and 252 DF,  p-value: 0.07301
```

```r
summary(lm(Shannon ~ Parkinson * history_ENT_surgery, pdDivs)) # no
```

```
##
## Call:
## lm(formula = Shannon ~ Parkinson * history_ENT_surgery, data = pdDivs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.02060 -0.23367 -0.00376  0.25992  0.85778
##
## Coefficients:
##                                          Estimate Std. Error t value
## (Intercept)                              3.757001   0.034427 109.129
## ParkinsonParkinson                       0.006612   0.048568   0.136
## history_ENT_surgery1                     0.213972   0.074959   2.855
## ParkinsonParkinson:history_ENT_surgery1 -0.102616   0.106755  -0.961
##                                         Pr(>|t|)
## (Intercept)                              < 2e-16 ***
## ParkinsonParkinson                       0.89181
## history_ENT_surgery1                     0.00467 **
## ParkinsonParkinson:history_ENT_surgery1  0.33736
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.346 on 252 degrees of freedom
## Multiple R-squared:  0.03974,    Adjusted R-squared:  0.02831
## F-statistic: 3.477 on 3 and 252 DF,  p-value: 0.01663
```

```r
summary(lm(Observed ~ Parkinson * meds_ca_antagonist, pdDivs)) # neither variable significant on
    their own
```

```
##
## Call:
## lm(formula = Observed ~ Parkinson * meds_ca_antagonist, data = pdDivs)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -251.348  -65.829   -4.403   64.990  240.597
##
## Coefficients:
##                                      Estimate Std. Error t value
## (Intercept)                           397.829      9.499  41.881
## ParkinsonParkinson                     -3.425     13.033  -0.263
## meds_ca_antagonist1                    15.519     22.409   0.693
## ParkinsonParkinson:meds_ca_antagonist1   85.633     40.429   2.118
##                                      Pr(>|t|)
## (Intercept)                            <2e-16 ***
## ParkinsonParkinson                     0.7929
## meds_ca_antagonist1                    0.4892
## ParkinsonParkinson:meds_ca_antagonist1   0.0351 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 97.34 on 252 degrees of freedom
## Multiple R-squared:  0.03641,    Adjusted R-squared:  0.02494
## F-statistic: 3.174 on 3 and 252 DF,  p-value: 0.02482
```

```r
summary(lm(InvSimpson ~ Parkinson * meds_ca_antagonist, pdDivs)) # no
```

```
##
## Call:
## lm(formula = InvSimpson ~ Parkinson * meds_ca_antagonist, data = pdDivs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.662  -5.638  -1.520   5.021  27.123
##
## Coefficients:
##                                      Estimate Std. Error t value
## (Intercept)                           19.9620     0.7539  26.478
## ParkinsonParkinson                    -0.1829     1.0344  -0.177
## meds_ca_antagonist1                    4.1838     1.7785   2.352
## ParkinsonParkinson:meds_ca_antagonist1   3.2740     3.2087   1.020
##                                      Pr(>|t|)
## (Intercept)                            <2e-16 ***
## ParkinsonParkinson                     0.8598
## meds_ca_antagonist1                    0.0194 *
## ParkinsonParkinson:meds_ca_antagonist1   0.3085
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.725 on 252 degrees of freedom
## Multiple R-squared:  0.05089,    Adjusted R-squared:  0.03959
## F-statistic: 4.504 on 3 and 252 DF,  p-value: 0.004247
```

```r
summary(lm(Shannon ~ Parkinson * meds_ca_antagonist, pdDivs)) # 0.1 > p > 0.05, no significance for
    PD
```

```
##
## Call:
## lm(formula = Shannon ~ Parkinson * meds_ca_antagonist, data = pdDivs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.03760 -0.22850  0.01048  0.26130  0.68915
##
## Coefficients:
##                                      Estimate Std. Error t value
## (Intercept)                            3.77400    0.03345 112.815
## ParkinsonParkinson                    -0.01575    0.04590  -0.343
## meds_ca_antagonist1                    0.15657    0.07892   1.984
## ParkinsonParkinson:meds_ca_antagonist1  0.24135    0.14238   1.695
##                                       Pr(>|t|)
## (Intercept)                             <2e-16 ***
## ParkinsonParkinson                      0.7318
## meds_ca_antagonist1                     0.0483 *
## ParkinsonParkinson:meds_ca_antagonist1   0.0913 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3428 on 252 degrees of freedom
## Multiple R-squared:  0.05741,    Adjusted R-squared:  0.04619
## F-statistic: 5.116 on 3 and 252 DF,  p-value: 0.001878
```

The potential BMI * PD interaction seems worth exploring in more detail. (Here done without taking into account the timepoint for the samples, so technically this is pseudoreplicated data.)

```r
# Make models with and without interaction for each diversity index
lmObsM1 <- summary(lm(Observed ~ Parkinson + BMI, pdDivs))
lmObsM2 <- summary(lm(Observed ~ Parkinson * BMI, pdDivs))
lmShanM1 <- summary(lm(Shannon ~ Parkinson + BMI, pdDivs))
lmShanM2 <- summary(lm(Shannon ~ Parkinson * BMI, pdDivs))
lmInvSM1 <- summary(lm(InvSimpson ~ Parkinson + BMI, pdDivs))
lmInvSM2 <- summary(lm(InvSimpson ~ Parkinson * BMI, pdDivs))

modlist <- list(lmObsM1, lmShanM1, lmInvSM1, lmObsM2, lmShanM2, lmInvSM2)

# Make a table with values extracted from these
bmiPDlm <- as.data.frame(matrix(ncol = 5, nrow = (3 * 4 + 3 * 5)))
colnames(bmiPDlm) <- c("Model", "Variable", "adjRsquared", "Estimate", "pval")
bmiPDlm$Model <- c(rep(c("Observed richness ~ PD + BMI", "Shannon ~ PD + BMI", "InvS ~ PD + BMI"),
    each = 4),
                 rep(c("Observed richness ~ PD * BMI", "Shannon ~ PD * BMI", "InvS ~ PD * BMI"),
                     each = 5))
bmiPDlm$Model <- factor(bmiPDlm$Model, levels=c("Observed richness ~ PD + BMI", "Observed richness ~
    PD * BMI",
                               "Shannon ~ PD + BMI", "Shannon ~ PD * BMI",
                               "InvS ~ PD + BMI", "InvS ~ PD * BMI"))
bmiPDlm$Variable <- c(rep(c("model", "Intercept", "PD", "BMI"), 3),
                    rep(c("model", "Intercept", "PD", "BMI", "PD:BMI"), 3))

# R-squared
bmiPDlm[bmiPDlm$Variable == "model", "adjRsquared"] <- unlist(sapply(modlist, function(x)
    x["adj.r.squared"]))

# P-value for full model
bmiPDlm[bmiPDlm$Variable == "model", "pval"] <- sapply(modlist, function(x)
    pf(unlist(x["fstatistic"])[1], unlist(x["fstatistic"])[2], unlist(x["fstatistic"])[3], lower =
    FALSE))

# P-values
bmiPDlm[bmiPDlm$Variable!="model", "pval"] <- c(
  as.vector(unlist(sapply(modlist[1:3], function(x) x$coefficients[1:3, "Pr(>|t|)"]))),
  as.vector(unlist(sapply(modlist[4:6], function(x) x$coefficients[1:4, "Pr(>|t|)"]))))

# Estimates
bmiPDlm[bmiPDlm$Variable!="model", "Estimate"] <- c(
  as.vector(unlist(sapply(modlist[1:3], function(x) x$coefficients[1:3, "Estimate"]))),
  as.vector(unlist(sapply(modlist[4:6], function(x) x$coefficients[1:4, "Estimate"]))))

# Order by model levels
```

```
bmiPDlm <- bmiPDlm[order(bmiPDlm$Model),]
rownames(bmiPDlm) <- NULL
```

Table S2A

Export the results:

```
kable_styling(kable(bmiPDlm, digits = 3), font_size = 10, full_width = FALSE)
```

Model	Variable	adjRsquared	Estimate	pval
Observed richness ~ PD + BMI	model	0.033	NA	0.006
Observed richness ~ PD + BMI	Intercept	NA	528.932	0.000
Observed richness ~ PD + BMI	PD	NA	4.066	0.738
Observed richness ~ PD + BMI	BMI	NA	-4.879	0.001
Observed richness ~ PD * BMI	model	0.034	NA	0.010
Observed richness ~ PD * BMI	Intercept	NA	582.092	0.000
Observed richness ~ PD * BMI	PD	NA	-84.872	0.309
Observed richness ~ PD * BMI	BMI	NA	-6.891	0.004
Observed richness ~ PD * BMI	PD:BMI	NA	3.328	0.281
Shannon ~ PD + BMI	model	0.018	NA	0.039
Shannon ~ PD + BMI	Intercept	NA	4.163	0.000
Shannon ~ PD + BMI	PD	NA	-0.016	0.720
Shannon ~ PD + BMI	BMI	NA	-0.013	0.013
Shannon ~ PD * BMI	model	0.036	NA	0.007
Shannon ~ PD * BMI	Intercept	NA	4.579	0.000
Shannon ~ PD * BMI	PD	NA	-0.713	0.017
Shannon ~ PD * BMI	BMI	NA	-0.029	0.001
Shannon ~ PD * BMI	PD:BMI	NA	0.026	0.018
InvS ~ PD + BMI	model	0.009	NA	0.115
InvS ~ PD + BMI	Intercept	NA	27.337	0.000
InvS ~ PD + BMI	PD	NA	-0.406	0.683
InvS ~ PD + BMI	BMI	NA	-0.247	0.046
InvS ~ PD * BMI	model	0.024	NA	0.030
InvS ~ PD * BMI	Intercept	NA	35.967	0.000
InvS ~ PD * BMI	PD	NA	-14.844	0.029
InvS ~ PD * BMI	BMI	NA	-0.573	0.004
InvS ~ PD * BMI	PD:BMI	NA	0.540	0.032

```
## Export
write.csv(bmiPDlm, "Outputs/table_s2a.csv")
```

Plot for the interaction for Shannon & inverse Simpson:

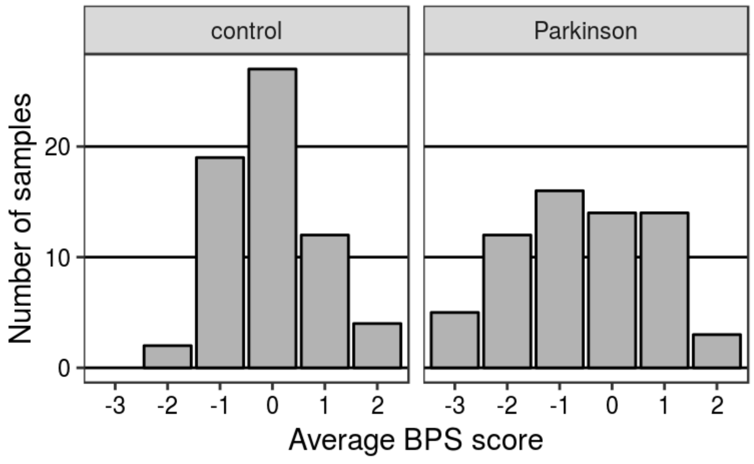
```
# Make models with timepoint included
shanM2mod <- lm(Shannon ~ Timepoint + Parkinson * BMI, pdDivs)
invsM2mod <- lm(InvSimpson ~ Timepoint + Parkinson * BMI, pdDivs)

# Make predicted data based on these models and plot

# Shannon

shannonPred <- data.frame(Shannon = predict(shanM2mod, pdDivs, se.fit = TRUE),
                          Parkinson = pdDivs$Parkinson,
                          BMI = pdDivs$BMI,
                          Timepoint = pdDivs$Timepoint)
colnames(shannonPred)[1] <- "Shannon"
shannonPred$ymin <- shannonPred$Shannon - shannonPred$Shannon.se.fit
shannonPred$ymax <- shannonPred$Shannon + shannonPred$Shannon.se.fit

bmiAlphaShan <- ggplot(pdDivs, aes(x = BMI, y = Shannon, color = Timepoint, shape = Timepoint)) +
 ggtitle("A.") +
 geom_point() +
 facet_grid(~Parkinson) +
 theme_bw() +
 scale_color_manual(values = c("gray20", "seagreen")) +
 geom_ribbon(data = shannonPred, aes(ymin = ymin, ymax = ymax), fill = "grey70", alpha = 0.5, col =
     NA) +
 geom_line(data = shannonPred, aes(x = BMI, y = Shannon, color = Timepoint)) +
 ylab("Shannon") +
 theme(panel.grid = element_blank(), legend.position = "bottom")
```

```r
# Inverse Simpson

invsPred <- data.frame(InvSimpson = predict(invsM2mod, pdDivs, se.fit = TRUE),
                       Parkinson = pdDivs$Parkinson,
                       BMI = pdDivs$BMI,
                       Timepoint = pdDivs$Timepoint)
colnames(invsPred)[1] <- "InvSimpson"
invsPred$ymin <- invsPred$InvSimpson - invsPred$InvSimpson.se.fit
invsPred$ymax <- invsPred$InvSimpson + invsPred$InvSimpson.se.fit

bmiAlphaInvS <- ggplot(pdDivs, aes(x = BMI, y = InvSimpson, color = Timepoint, shape = Timepoint)) +
 ggtitle("B.") +
 geom_point() +
 facet_grid(~Parkinson) +
 theme_bw() +
 scale_color_manual(values = c("gray20", "seagreen")) +
 geom_ribbon(data = invsPred, aes(ymin = ymin, ymax = ymax),
       fill = "grey70", alpha = 0.5, col = NA) +
 geom_line(data = invsPred, aes(x = BMI, y = InvSimpson, color = Timepoint)) +
 ylab("Inverse Simpson") +
 theme(panel.grid = element_blank(), legend.position = "bottom")
```

Victoria Bowel Performance Scale (BPS)

This additional variable was explored due to the literature reports of stool consistency being associated with alpha diversity, and was only available at follow-up.

```r
# Make a subset of follow-up data for these analyses
newPD_FU <- subset_samples(newPD, Timepoint == "followup")

# Import BPS data
bps_df <- read.csv("Inputs/bps_data.csv", sep = "\t", row.names = 1)
rownames(bps_df) <- paste(rownames(bps_df), "N", sep="")
bps_df <- bps_df[rownames(sample_data(newPD_FU)),]
bps_df$Parkinson <- sample_data(newPD_FU)$Parkinson

# Make a categorical BPS variable

bps_df$BPSCat <- cut(bps_df$Stool_diary_D1_characteristic_average, breaks = c(-3.1, -2.5, -1.5,
    -0.5, 0.5, 1.5, 2.5, 3))
levels(bps_df$BPSCat) <- c(-3, -2, -1, 0, 1, 2, 3)

# What does this data look like?
ggplot(bps_df, aes(x=BPSCat)) +
  geom_bar(color="black", fill="gray70") +
  xlab("Average BPS score") +
  ylab("Number of samples") +
  theme_bw() +
  facet_grid(~Parkinson) +
  theme(panel.grid.minor = element_blank(),
        panel.grid.major.x = element_blank(),
        panel.grid.major.y = element_line(color="black"),
        axis.text = element_text(color="black"))
```



As could be expected, the PD patients have a wider spread of values here. Contrast alpha diversity and BPS, and collect the results into a table.

```
# Make a subset of the phyloseq object for follow-up only
newPD_FU <- phyloseq(otu_table(newPD_FU, taxa_are_rows=TRUE), tax_table(newPD_FU),
    sample_data(bps_df))

# Calculate diversity indices
bps_divs <- cbind(bps_df, estimate_richness(newPD_FU, measures = c("Observed", "Shannon",
    "InvSimpson")))

# Make models with and without interactions

lmObs_BPS1 <- summary(lm(Observed ~ Parkinson + Stool_diary_D1_characteristic_average, bps_divs))
lmObs_BPS2 <- summary(lm(Observed ~ Parkinson * Stool_diary_D1_characteristic_average, bps_divs))
lmShan_BPS1 <- summary(lm(Shannon ~ Parkinson + Stool_diary_D1_characteristic_average, bps_divs))
lmShan_BPS2 <- summary(lm(Shannon ~ Parkinson * Stool_diary_D1_characteristic_average, bps_divs))
lmInvS_BPS1 <- summary(lm(InvSimpson ~ Parkinson + Stool_diary_D1_characteristic_average, bps_divs))
lmInvS_BPS2 <- summary(lm(InvSimpson ~ Parkinson * Stool_diary_D1_characteristic_average, bps_divs))

modlist_BPS <- list(lmObs_BPS1, lmShan_BPS1, lmInvS_BPS1, lmObs_BPS2, lmShan_BPS2, lmInvS_BPS2)

# Collect values into a table, similarly to the BMI analyses

bpsPDlm <- as.data.frame(matrix(ncol = 5, nrow = (3 * 4 + 3 * 5)))
colnames(bpsPDlm) <- c("Model", "Variable", "adjRsquared", "Estimate", "pval")
bpsPDlm$Model <- c(rep(c("Observed richness ~ PD + BPS", "Shannon ~ PD + BPS", "InvS ~ PD + BPS"),
    each = 4),
                 rep(c("Observed richness ~ PD * BPS", "Shannon ~ PD * BPS", "InvS ~ PD * BPS"),
                     each = 5))
bpsPDlm$Variable <- c(rep(c("model", "Intercept", "PD", "BPS"), 3),
                    rep(c("model", "Intercept", "PD", "BPS", "PD:BPS"), 3))
bpsPDlm$Model <- factor(bpsPDlm$Model, levels = c("Observed richness ~ PD + BPS", "Observed richness
    ~ PD * BPS",
                              "Shannon ~ PD + BPS", "Shannon ~ PD * BPS", "InvS ~ PD + BPS", "InvS
                                  ~ PD * BPS"))

# R-squared
bpsPDlm[bpsPDlm$Variable == "model", "adjRsquared"] <- unlist(sapply(modlist_BPS, function(x)
    x["adj.r.squared"]))

# P-value for full model
bpsPDlm[bpsPDlm$Variable == "model", "pval"] <- sapply(modlist_BPS, function(x)
    pf(unlist(x["fstatistic"])[1], unlist(x["fstatistic"])[2], unlist(x["fstatistic"])[3], lower =
    FALSE))

# P-values for variables
```

```
bpsPDlm[bpsPDlm$Variable!="model", "pval"] <- c(
  as.vector(unlist(sapply(modlist_BPS[1:3], function(x) x$coefficients[1:3, "Pr(>|t|)"]))),
  as.vector(unlist(sapply(modlist_BPS[4:6], function(x) x$coefficients[1:4, "Pr(>|t|)"]))))

# Estimates
bpsPDlm[bpsPDlm$Variable!="model", "Estimate"] <- c(
  as.vector(unlist(sapply(modlist_BPS[1:3], function(x) x$coefficients[1:3, "Estimate"]))),
  as.vector(unlist(sapply(modlist_BPS[4:6], function(x) x$coefficients[1:4, "Estimate"]))))

# Order by model levels
bpsPDlm <- bpsPDlm[order(bpsPDlm$Model),]
rownames(bpsPDlm) <- NULL
```

Table S2B

Export the results:

```
kable_styling(kable(bpsPDlm, digits = 3), font_size = 10, full_width = FALSE)
```

Model	Variable	adjRsquared	Estimate	pval
Observed richness ~ PD + BPS	model	0.028	NA	0.061
Observed richness ~ PD + BPS	Intercept	NA	413.419	0.000
Observed richness ~ PD + BPS	PD	NA	-11.715	0.492
Observed richness ~ PD + BPS	BPS	NA	-17.430	0.019
Observed richness ~ PD * BPS	model	0.083	NA	0.003
Observed richness ~ PD * BPS	Intercept	NA	414.484	0.000
Observed richness ~ PD * BPS	PD	NA	-6.349	0.703
Observed richness ~ PD * BPS	BPS	NA	-48.008	0.000
Observed richness ~ PD * BPS	PD:BPS	NA	44.563	0.004
Shannon ~ PD + BPS	model	-0.011	NA	0.754
Shannon ~ PD + BPS	Intercept	NA	3.822	0.000
Shannon ~ PD + BPS	PD	NA	-0.034	0.588
Shannon ~ PD + BPS	BPS	NA	-0.017	0.534
Shannon ~ PD * BPS	model	0.007	NA	0.284
Shannon ~ PD * BPS	Intercept	NA	3.825	0.000
Shannon ~ PD * BPS	PD	NA	-0.021	0.731
Shannon ~ PD * BPS	BPS	NA	-0.087	0.067
Shannon ~ PD * BPS	PD:BPS	NA	0.103	0.073
InvS ~ PD + BPS	model	-0.015	NA	0.913
InvS ~ PD + BPS	Intercept	NA	21.075	0.000
InvS ~ PD + BPS	PD	NA	-0.368	0.794
InvS ~ PD + BPS	BPS	NA	-0.233	0.701
InvS ~ PD * BPS	model	-0.013	NA	0.723
InvS ~ PD * BPS	Intercept	NA	21.108	0.000
InvS ~ PD * BPS	PD	NA	-0.200	0.888
InvS ~ PD * BPS	BPS	NA	-1.189	0.273
InvS ~ PD * BPS	PD:BPS	NA	1.394	0.287

```
## Export
write.csv(bpsPDlm, "Outputs/table_s2b.csv")
```

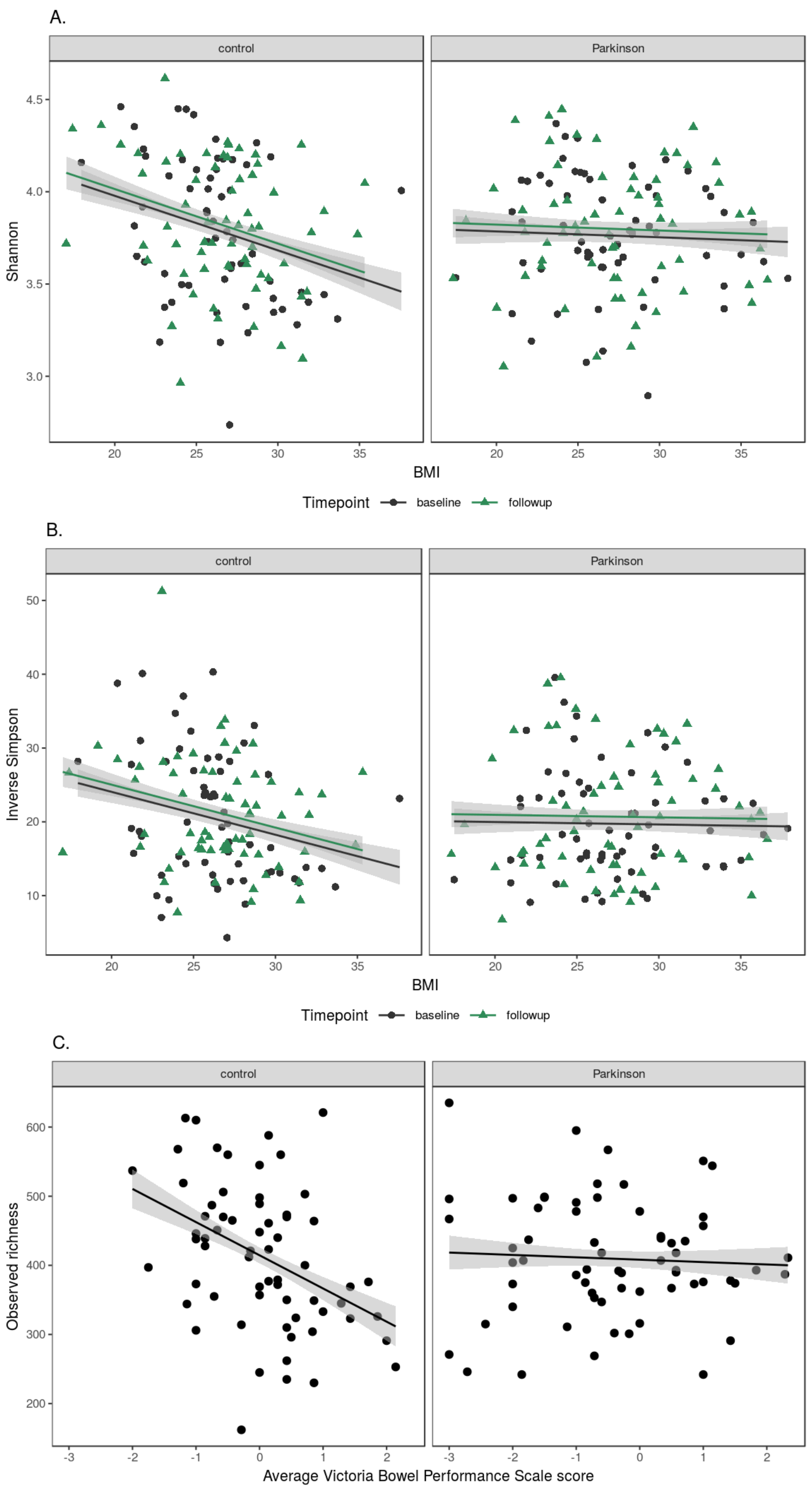
There seems to be an interaction for PD and BPS for observed richness, but not for the two indices that include evenness. Make a plot for observed richness:

```
# Table
bpsObsM2 <- lm(Observed ~ Parkinson * Stool_diary_D1_characteristic_average, bps_divs)

bpsObsPred <- data.frame(Observed = predict(bpsObsM2, bps_divs, se.fit = TRUE),
                         Parkinson = bps_divs$Parkinson,
                         Stool_diary_D1_characteristic_average =
                             bps_divs$Stool_diary_D1_characteristic_average)

bpsLMplot <- ggplot(bps_divs, aes(x=Stool_diary_D1_characteristic_average, y=Observed)) +
    ggtitle("C.") +
    facet_grid(~Parkinson) +
    geom_point() +
    theme_bw() +
    geom_ribbon(aes(ymin = bpsObsPred$Observed.fit - bpsObsPred$Observed.se.fit,
                    ymax = bpsObsPred$Observed.fit + bpsObsPred$Observed.se.fit),
               fill = "grey70", alpha=0.5) +
    geom_line(data = bpsObsPred, aes(x = Stool_diary_D1_characteristic_average, y = Observed.fit)) +
    ylab("Observed richness") +
    xlab("Average Victoria Bowel Performance Scale score") +
    theme(panel.grid=element_blank())
```

Plot the BMI and BPS models together

Figure 6

```
# Plot the three interaction + alpha diversity plots together:
fig6 <- arrangeGrob(bmiAlphaShan +
               theme_bw(base_size = 8) +
               theme(legend.position = "bottom", panel.grid = element_blank(),
                     legend.box.margin = ggplot2::margin(c(-10,0,-10,0))),
             bmiAlphaInvS +
               theme_bw(base_size = 8) +
               theme(legend.position = "bottom", panel.grid = element_blank(),
                     legend.box.margin = ggplot2::margin(c(-10,0,-10,0))),
             bpsLMplot +
               theme_bw(base_size = 8) +
               theme(legend.position = "bottom", panel.grid = element_blank()),
               ncol=1, heights=c(1.1,1.1,1))

grid.arrange(fig6)
```



```r
## Export to pdf
ggsave(fig6, filename = "Outputs/figure6.pdf", device = cairo_pdf,
       height = maxhi*0.7, width = halfpage, units = "in")
```

Progression

```r
# Alpha diversity data for PD/progression only
```

```
pdDivsProg <- cbind(progmeta, estimate_richness(progPhy, measures=c("Observed",
    "Shannon", "InvSimpson")))

progDivRes <- adivTab(pdDivsProg, "ProgCat")

kable_styling(kable(progDivRes, digits = 3), full_width = FALSE)
```

	Baseline	Followup
Observed	0.482	0.215
Shannon	0.441	0.647
Inverse Simpson	0.232	0.840

No differences in alpha diversity when contrasting stable and progressed PD patients at either timepoint.

PD phenotypes (TD vs PIGD)

Simple contrasts of alpha diversity between the TD and PIGD groups, separately for each timepoint:

```
# Calculate diversity index values
pheRich <- estimate_richness(progPhyPhe, measures = c("Observed", "Shannon", "InvSimpson"))
pheRich <- cbind(pheRich, sample_data(progPhyPhe)[, c("JankovicClass", "Timepoint")])

pheDivRes <- adivTab(pheRich, "JankovicClass")

kable_styling(kable(pheDivRes, digits = 3), full_width = FALSE)
```

	Baseline	Followup
Observed	0.572	0.368
Shannon	0.849	0.493
Inverse Simpson	0.568	0.762

No significant differences in alpha diversity between the phenotypes at either timepoint with any of the indices.

Diet

Test all the variables of interest against three alpha diversity indices:

```
ffqDivs <- cbind(ffqmeta, estimate_richness(ffqPhy, measures = c("Observed", "Shannon",
    "InvSimpson")))

ffqDivsDiff <- as.data.frame(t(sapply(ffqVars, function (x)
  c(kruskal.test(x=ffqDivs$Observed, g=ffqDivs[[x]])$p.value,
    kruskal.test(x=ffqDivs$Shannon, g=ffqDivs[[x]])$p.value,
    kruskal.test(x=ffqDivs$InvSimpson, g=ffqDivs[[x]])$p.value))))

colnames(ffqDivsDiff) <- c("pObserved", "pShannon", "pInvSimpson")

# Is anything significant when unadjusted?
kable_styling(kable(subset(ffqDivsDiff, pObserved < 0.05 | pShannon < 0.05 | pInvSimpson < 0.05),
    digits = 3), full_width = FALSE)
```

	pObserved	pShannon	pInvSimpson
Kuitul_g_per1kkc	0.410	0.042	0.180
Jodi_per1kkc	0.481	0.021	0.017
Karoteno_per1kkc	0.187	0.015	0.091
D_vitam_per1kkc	0.205	0.048	0.172
Puuro_per1kkc	0.325	0.066	0.027
Hedmeyh_per1kkc	0.037	0.025	0.160
Olutyht_per1kkc	0.003	0.055	0.101

```
# Adjust for multiple comparisons
ffqDivsDiff$padjObserved <- p.adjust(ffqDivsDiff$pObserved, method = "fdr")
```

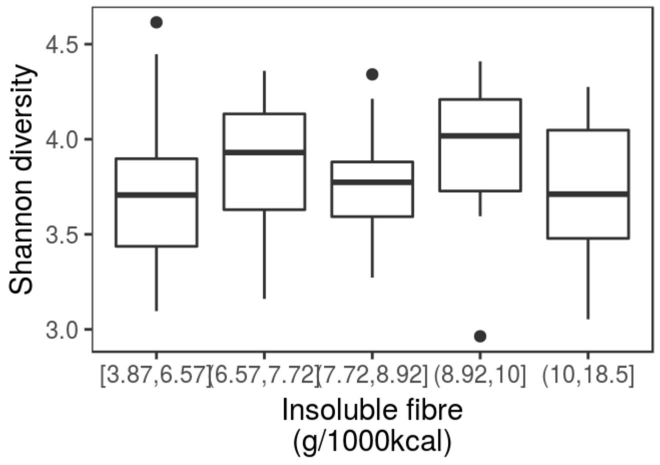
```
ffqDivsDiff$padjShannon <- p.adjust(ffqDivsDiff$pShannon, method = "fdr")
ffqDivsDiff$padjInvSimpson <- p.adjust(ffqDivsDiff$pInvSimpson, method = "fdr")

# Any significant adjusted p-values (or even anywhere close)?
nrow(subset(ffqDivsDiff, padjObserved < 0.1 | padjShannon < 0.1 | padjInvSimpson < 0.1))
```

```
## [1] 0
```

```
# no, nothing is
```

One potentially interesting variable that had a significant uncorrected *p*-value for Shannon diversity was fiber. Plot this separately:

```
ggplot(ffqDivs, aes(x=Kuitul_g_per1kkc, y=Shannon)) +
  geom_boxplot() +
  theme_bw() +
  ylab("Shannon diversity") +
  xlab("Insoluble fibre\n(g/1000kcal)") +
  theme(panel.grid=element_blank())
```



This doesn't look particularly clear; higher fiber consumption definitely doesn't seem to be linked with higher diversity.

Beta diversity analyses

PD vs control

Data setup

Set up data for the comparison, including dropping some subjects that had NA valuers for a few potential confounders:

```
# List of confounders from earlier metadata comparisons:
adVarsPDvsC
```

```
##  [1] "age_at_stool_collection"
##  [2] "BMI"
##  [3] "gender"
##  [4] "history_TIA_ischemic_stroke"
##  [5] "meds_ACEI_ARB"
##  [6] "meds_anticholinergic"
##  [7] "meds_ca_antagonist"
##  [8] "meds_statin"
##  [9] "meds_Warfarin"
## [10] "MMSE_total"
## [11] "RLS"
## [12] "Rome_III_constip_defec_sumscore_9.15"
## [13] "Rome_III_IBS_criteria_fulfilled"
## [14] "Wexner_total"
## [15] "FoodPC1"
```

```
# Missing values check:
sort(colSums(is.na(sample_data(newPD)[,adVarsPDvsC])), decreasing = TRUE)
```

```
##                              BMI          history_TIA_ischemic_stroke
##                                5                                    2
##           age_at_stool_collection                               gender
##                                0                                    0
##                     meds_ACEI_ARB                meds_anticholinergic
##                                0                                    0
##                meds_ca_antagonist                          meds_statin
##                                0                                    0
##                     meds_Warfarin                           MMSE_total
##                                0                                    0
##                               RLS Rome_III_constip_defec_sumscore_9.15
##                                0                                    0
##      Rome_III_IBS_criteria_fulfilled                        Wexner_total
##                                0                                    0
##                           FoodPC1
##                                0
```

```
# Drop the subjects with NA values from the metadata:
pdmetaAD <- subset(pdclin, !is.na(pdclin$history_TIA_ischemic_stroke) & !is.na(pdclin$BMI))

# OTU level phyloseq object
newPD_R <- rarefy_even_depth(subset_samples(newPD,
    !is.na(sample_data(newPD)$history_TIA_ischemic_stroke) & !is.na(sample_data(newPD)$BMI)),
    rngseed = 521483)

# Genus level phyloseq object
newPDgen_R <- rarefy_even_depth(subset_samples(newPDgen,
    !is.na(sample_data(newPDgen)$history_TIA_ischemic_stroke) & !is.na(sample_data(newPDgen)$BMI)),
    rngseed = 113210)

# Family level phyloseq object
newPDfam_R <- rarefy_even_depth(subset_samples(newPDfam,
    !is.na(sample_data(newPDfam)$history_TIA_ischemic_stroke) & !is.na(sample_data(newPDfam)$BMI)),
    rngseed = 464519)

# Calculate distance matrices (with the default method, Bray-Curtis)
pddisOTU <- vegdist(t(as.data.frame(as.matrix(otu_table(newPD_R)))))
pddisGen <- vegdist(t(as.data.frame(as.matrix(otu_table(newPDgen_R)))))
pddisFam <- vegdist(t(as.data.frame(as.matrix(otu_table(newPDfam_R)))))
```

ADONIS

Calculate significance for adonis models that just have timepoint and PD status without any confounders:

```
# OTU level
adTPPD_noconf_OTUs <- adonis2(pddisOTU ~ sample_data(newPD_R)[["Timepoint"]] +
            sample_data(newPD_R)[["Parkinson"]], perm = 9999, by = "margin")
adTPPD_noconf_OTUs
```

```
## Permutation test for adonis under reduced model
## Marginal effects of terms
## Permutation: free
## Number of permutations: 9999
##
## adonis2(formula = pddisOTU ~ sample_data(newPD_R)[["Timepoint"]] +
    sample_data(newPD_R)[["Parkinson"]], permutations = 9999, by = "margin")
##                                        Df SumOfSqs      R2      F Pr(>F)
## sample_data(newPD_R)[["Timepoint"]]    1    0.174 0.00292 0.7314 0.8984
## sample_data(newPD_R)[["Parkinson"]]    1    0.907 0.01522 3.8146 0.0001 ***
## Residual                             246   58.522 0.98185
## Total                               248   59.604 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Genus level
adTPPD_noconf_Gen <- adonis2(pddisGen ~ sample_data(newPDgen_R)[["Timepoint"]] +
            sample_data(newPDgen_R)[["Parkinson"]], perm = 9999, by = "margin")
adTPPD_noconf_Gen
```

```
## Permutation test for adonis under reduced model
## Marginal effects of terms
## Permutation: free
## Number of permutations: 9999
##
## adonis2(formula = pddisGen ~ sample_data(newPDgen_R)[["Timepoint"]] +
    sample_data(newPDgen_R)[["Parkinson"]], permutations = 9999, by = "margin")
##                                         Df SumOfSqs      R2      F Pr(>F)
## sample_data(newPDgen_R)[["Timepoint"]]   1   0.0754 0.00330 0.8375 0.5441
## sample_data(newPDgen_R)[["Parkinson"]]   1   0.5955 0.02611 6.6174 0.0001
## Residual                               246  22.1360 0.97057
## Total                                  248  22.8072 1.00000
##
## sample_data(newPDgen_R)[["Timepoint"]]
## sample_data(newPDgen_R)[["Parkinson"]] ***
## Residual
## Total
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Family level
adTPPD_noconf_Fam <- adonis2(pddisFam ~ sample_data(newPDfam_R)[["Timepoint"]] +
            sample_data(newPDfam_R)[["Parkinson"]], perm = 9999, by = "margin")
adTPPD_noconf_Fam
```

```
## Permutation test for adonis under reduced model
## Marginal effects of terms
## Permutation: free
## Number of permutations: 9999
##
## adonis2(formula = pddisFam ~ sample_data(newPDfam_R)[["Timepoint"]] +
    sample_data(newPDfam_R)[["Parkinson"]], permutations = 9999, by = "margin")
##                                         Df SumOfSqs      R2      F Pr(>F)
## sample_data(newPDfam_R)[["Timepoint"]]   1   0.0433 0.00281 0.7156 0.6472
## sample_data(newPDfam_R)[["Parkinson"]]   1   0.4691 0.03049 7.7593 0.0001
## Residual                               246  14.8731 0.96667
## Total                                  248  15.3859 1.00000
##
## sample_data(newPDfam_R)[["Timepoint"]]
## sample_data(newPDfam_R)[["Parkinson"]] ***
## Residual
## Total
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Check for subsets (PD only / C only) to see if the timepoints differ:

```
# PD patients only:
pddisPD <- vegdist(t(as.data.frame(as.matrix(otu_table(subset_samples(newPDgen_R,
    sample_data(newPDgen_R)$Parkinson == "Parkinson"))))))
adonisTP_P <- adonis2(pddisPD ~ Timepoint, subset(pdmetaAD, Parkinson == "Parkinson"), perm = 9999,
    by = "margin")
adonisTP_P
```

```
## Permutation test for adonis under NA model
## Marginal effects of terms
## Permutation: free
## Number of permutations: 9999
##
## adonis2(formula = pddisPD ~ Timepoint, data = subset(pdmetaAD, Parkinson == "Parkinson"),
    permutations = 9999, by = "margin")
##            Df SumOfSqs      R2      F Pr(>F)
## Timepoint   1   0.0518 0.00489 0.5993 0.8094
## Residual  122  10.5498 0.99511
## Total     123  10.6016 1.00000
```

```
# Controls only:
pddisC <- vegdist(t(as.data.frame(as.matrix(otu_table(subset_samples(newPDgen_R,
    sample_data(newPDgen_R)$Parkinson == "control"))))))
adonisTP_C <- adonis2(pddisC ~ Timepoint, subset(pdmetaAD, Parkinson == "control"), perm = 9999, by
    = "margin")
adonisTP_C
```

```
## Permutation test for adonis under NA model
## Marginal effects of terms
## Permutation: free
## Number of permutations: 9999
##
## adonis2(formula = pddisC ~ Timepoint, data = subset(pdmetaAD, Parkinson == "control"),
    permutations = 9999, by = "margin")
##            Df SumOfSqs      R2      F Pr(>F)
## Timepoint   1   0.0595 0.00513 0.6339 0.7525
## Residual  123  11.5503 0.99487
## Total     124  11.6098 1.00000
```

There is no difference between timepoints, looking at either the control subjects or the PD patients.

Next, run an adonis loop with all potential confounding variables one by one (while correcting for timepoint and PD status):

```
# Initialize data frame for collecting results:
pdADONIS <- data.frame(row.names = adVarsPDvsC)

set.seed(27543298)

# Function for grabbing the p-value from comparisons
adloop <- function(dist, meta, varlist){
    adres <- sapply(varlist, function (x) adonis2(dist ~ ., meta[, c("Timepoint", "Parkinson", x)],
        by = "margin", permutations = 999)["Pr(>F)"][3,])
    return(adres)
}

# Run comparisons
pdADONIS$p_OTU <- adloop(pddisOTU, pdmetaAD, adVarsPDvsC)
pdADONIS$p_Gen <- adloop(pddisGen, pdmetaAD, adVarsPDvsC)
pdADONIS$p_Fam <- adloop(pddisFam, pdmetaAD, adVarsPDvsC)

# Order by the number of significant results
pdADONIS$n_sigs <- rowSums(pdADONIS[, 1:3] < 0.05)
```

Export these results:

Table 6B

```
kable_styling(kable(pdADONIS[order(pdADONIS$n_sigs, decreasing = TRUE),]))
```

	p_OTU	p_Gen	p_Fam	n_sigs
BMI	0.001	0.002	0.003	3
meds_ACEI_ARB	0.032	0.046	0.018	3
Rome_III_constip_-defec_sumscore_9.15	0.001	0.001	0.001	3
Wexner_total	0.001	0.002	0.004	3
meds_ca_antagonist	0.017	0.031	0.183	2
RLS	0.034	0.085	0.043	2
FoodPC1	0.003	0.009	0.064	2
history_TIA_-ischemic_stroke	0.007	0.103	0.186	1
age_at_stool_collection	0.153	0.170	0.266	0
gender	0.053	0.106	0.065	0
meds_anticholinergic	0.287	0.828	0.882	0
meds_statin	0.068	0.370	0.453	0
meds_Warfarin	0.564	0.610	0.339	0
MMSE_total	0.730	0.561	0.310	0
Rome_III_IBS_-criteria_fulfilled	0.229	0.191	0.276	0

```r
## Export
write.csv(pdADONIS, "Outputs/table_6b.csv")
```

The variables that were significant in these single-confounder models for at least one level were kept for further comparisons.

```r
# Make a new set out of the significant variables from the above table
adVarsSelect <- row.names(pdADONIS[pdADONIS[,"n_sigs"] > 0, ])
adVarsSelect <- c("Timepoint", "Parkinson", adVarsSelect)

# Since the Rome and Wexner scores are highly collinear, only one of them will be
# included in further comparisons; we chose the Rome III score, which is better validated.

adVarsSelect <- adVarsSelect[-grep("Wexner", adVarsSelect)]

# Adonis with all the variables on this list for each of the three levels
adAllSigsOTU <- adonis2(pddisOTU ~ ., data = pdmetaAD[, adVarsSelect], perm = 9999, by = "margin")
adAllSigsGen <- adonis2(pddisGen ~ ., data = pdmetaAD[, adVarsSelect], perm = 9999, by = "margin")
adAllSigsFam <- adonis2(pddisFam ~ ., data = pdmetaAD[, adVarsSelect], perm = 9999, by = "margin")

pdADONIS2 <- data.frame(row.names = adVarsSelect)

pdADONIS2[adVarsSelect, "p_OTU_full"] <- adAllSigsOTU[adVarsSelect, 5]
pdADONIS2[adVarsSelect, "p_Gen_full"] <- adAllSigsGen[adVarsSelect, 5]
pdADONIS2[adVarsSelect, "p_Fam_full"] <- adAllSigsFam[adVarsSelect, 5]

pdADONIS2$n_sigs <- rowSums(pdADONIS2 < 0.05)
```

Envfit

Alternative test for beta diversity associations: envfit.

```r
# Select the variables that were significant in the above ADONIS models:
pdcADsigs <- rownames(pdADONIS2)[which(pdADONIS2$n_sigs > 0)]

# Calculate NMDS ordinations
pdmdsOTU <- metaMDS(t(as.data.frame(as.matrix(otu_table(newPD_R)))), try = 500)
pdmdsGen <- metaMDS(t(as.data.frame(as.matrix(otu_table(newPDgen_R)))), try = 500)
pdmdsFam <- metaMDS(t(as.data.frame(as.matrix(otu_table(newPDfam_R)))), try = 500)
```

```r
# Run envfit:

# OTU level data
efOTU <- envfit(pdmdsOTU, pdmetaAD[, c("Timepoint", pdcADsigs)], permutations = 9999, na.rm = TRUE)
```

```r
# Genus level data
efGen <- envfit(pdmdsGen, pdmetaAD[, c("Timepoint", pdcADsigs)], permutations = 9999, na.rm = TRUE)

# Family level data
efFam <- envfit(pdmdsFam, pdmetaAD[, c("Timepoint", pdcADsigs)], permutations = 9999, na.rm = TRUE)
```

Combined results

Collect the results of the above adonis and envfit comparisons and export:

Table 6AC

```r
# Make empty dataframe
pdcBdivs <- as.data.frame(matrix(ncol = 3 + length(adVarsSelect), nrow = 9))
colnames(pdcBdivs) <- c("Model", "Test", "Level", adVarsSelect)
pdcBdivs$Model <- c(rep("TP+PD", 3), rep("TP+PD+confounders", 6))
pdcBdivs$Test <- c(rep("adonis", 6), rep("envfit", 3))
pdcBdivs$Level <- rep(c("OTU", "Genus", "Family"), 3)

# ADONIS results without confounders
pdcBdivs[1, 4:5] <- adTPPD_noconf_OTUs[1:2, 5]
pdcBdivs[2, 4:5] <- adTPPD_noconf_Gen[1:2, 5]
pdcBdivs[3, 4:5] <- adTPPD_noconf_Fam[1:2, 5]

pdcBdivsVars <- rownames(adAllSigsOTU)[-grep("Residual", rownames(adAllSigsOTU))]

# ADONIS results with confounders
pdcBdivs[4, adVarsSelect] <- adAllSigsOTU[adVarsSelect, 5]
pdcBdivs[5, adVarsSelect] <- adAllSigsGen[adVarsSelect, 5]
pdcBdivs[6, adVarsSelect] <- adAllSigsFam[adVarsSelect, 5]

# Envfit results with confounders

# Factors
pdcBdivs[7, names(efOTU$factors$pval)] <- efOTU$factors$pval
pdcBdivs[8, names(efGen$factors$pval)] <- efGen$factors$pval
pdcBdivs[9, names(efFam$factors$pval)] <- efFam$factors$pval

# Numeric variables
pdcBdivs[7, rownames(efOTU$vectors$arrows)] <- efOTU$vectors$pvals
pdcBdivs[8, rownames(efGen$vectors$arrows)] <- efGen$vectors$pvals
pdcBdivs[9, rownames(efFam$vectors$arrows)] <- efFam$vectors$pvals

# Reorder
pdcBdivs <- pdcBdivs[, order(colSums(!is.na(pdcBdivs)), decreasing = TRUE)]
```

Table shown split into two parts to fit the page:

```r
kable(pdcBdivs[, 1:8], digits = 4)
```

Model	Test	Level	Timepoint	Parkinson	BMI	history_TIA_ischemic_stroke	meds_ACEI_ARB
TP+PD	adonis	OTU	0.8984	0.0001	NA	NA	NA
TP+PD	adonis	Genus	0.5441	0.0001	NA	NA	NA
TP+PD	adonis	Family	0.6472	0.0001	NA	NA	NA
TP+PD+confounders	adonis	OTU	0.8879	0.0062	0.0015	0.0089	0.0199
TP+PD+confounders	adonis	Genus	0.5103	0.0157	0.0017	0.0846	0.1678
TP+PD+confounders	adonis	Family	0.6253	0.0170	0.0246	0.3485	0.0441
TP+PD+confounders	envfit	OTU	0.9948	0.0006	0.0104	0.9476	0.1882
TP+PD+confounders	envfit	Genus	0.0828	0.0007	0.3718	0.6603	0.0974
TP+PD+confounders	envfit	Family	0.1287	0.0009	0.6067	0.9814	0.9532

```r
kable(pdcBdivs[, c(1:3, 9:12)], digits = 3)
```

Model	Test	Level	meds_ca_antagonist	Rome_III_constip_defec_sumscore_9.15	FoodPC1	RLS
TP+PD	adonis	OTU	NA	NA	NA	NA
TP+PD	adonis	Genus	NA	NA	NA	NA
TP+PD	adonis	Family	NA	NA	NA	NA
TP+PD+confounders	adonis	OTU	0.003	0.000	0.013	0.086
TP+PD+confounders	adonis	Genus	0.006	0.000	0.096	0.110
TP+PD+confounders	adonis	Family	0.121	0.001	0.226	0.133
TP+PD+confounders	envfit	OTU	0.073	0.000	0.063	NA
TP+PD+confounders	envfit	Genus	0.162	0.000	0.560	NA
TP+PD+confounders	envfit	Family	0.044	0.000	0.303	NA

```r
## Export
write.csv(pdcBdivs, "Outputs/table_6ac.csv")
```

Ordination plots of beta diversity

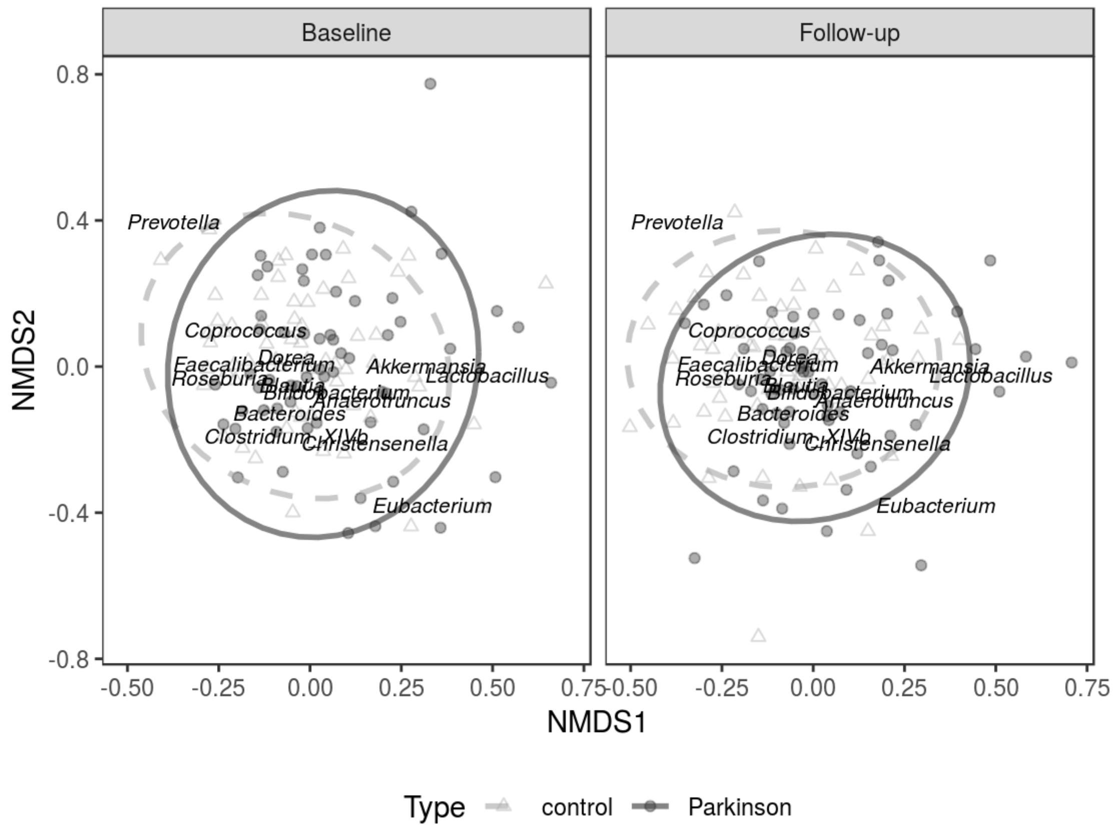
A plot showing the centroids for genera of interest:

Figure 7

```r
# Make data frame out of NMDS sample data
pdmdsGendf <- data.frame(NMDS1 = pdmdsGen$points[,1], NMDS2 = pdmdsGen$points[,2], pdmetaAD)

# Collect taxon data from the NMDS into a data frame
nmdsSpecAll <- data.frame(NMDS1 = pdmdsGen$species[, 1], NMDS2 = pdmdsGen$species[, 2])

# Select genera of interest (hand-picked based on previous literature)
litSigs <- read.csv("Inputs/sigtaxa_from_literature.csv", sep = "\t")
nmdsSpecLit <- gsub(" ", "_", subset(litSigs, Level == "genus")$Taxon.mentioned)

# Are all these genera available in our data?
nmdsSpecLit[!(nmdsSpecLit %in% rownames(nmdsSpecAll))]
```

```
## [1] "Oscillospira"     "Catabacter"       "Clostridium_XIVb"
```

```r
# A few are not, although the Clostridium taxon is missing because of a typo in the taxonomy;
# fix for this:
rownames(nmdsSpecAll) <- gsub("XlVb", "XIVb", rownames(nmdsSpecAll))

# Select these taxa from the data
nmdsSpec <- nmdsSpecAll[rownames(nmdsSpecAll) %in% nmdsSpecLit,]
nmdsSpec$Genus <- rownames(nmdsSpec)

# Draw plot
pdMDSplotSpec <- ggplot(pdmdsGendf) +
  geom_point(mapping = aes(x = NMDS1, y = NMDS2, col = Parkinson, shape = Parkinson), alpha = 0.4) +
  coord_fixed() +
  theme_bw() +
  facet_grid(~Timepoint, labeller = labeller(Timepoint = c(baseline = "Baseline", followup =
      "Follow-up"))) +
  scale_color_manual(values = c("gray65", "gray20"), name = "Type") +
  scale_shape_manual(name = "Type", values = c(2, 19)) +
  scale_linetype_manual(values = c("dashed", "solid"), name = "Type") +
  stat_ellipse(aes(x = NMDS1, y = NMDS2, colour = Parkinson, lty = Parkinson), level = 0.95, lwd =
      1, alpha = 0.6) +
  geom_text(data = nmdsSpec, aes(x = NMDS1, y = NMDS2, label = Genus), size = 2.75, fontface =
      "italic") +
  theme(legend.position = "bottom", panel.grid.minor = element_blank(), panel.grid.major =
      element_blank())
pdMDSplotSpec
```



```
## Export to pdf
ggsave(pdMDSplotSpec, filename = "Outputs/figure7.pdf", device = cairo_pdf,
       width = fullpage, height = 4.75, units = "in")
```

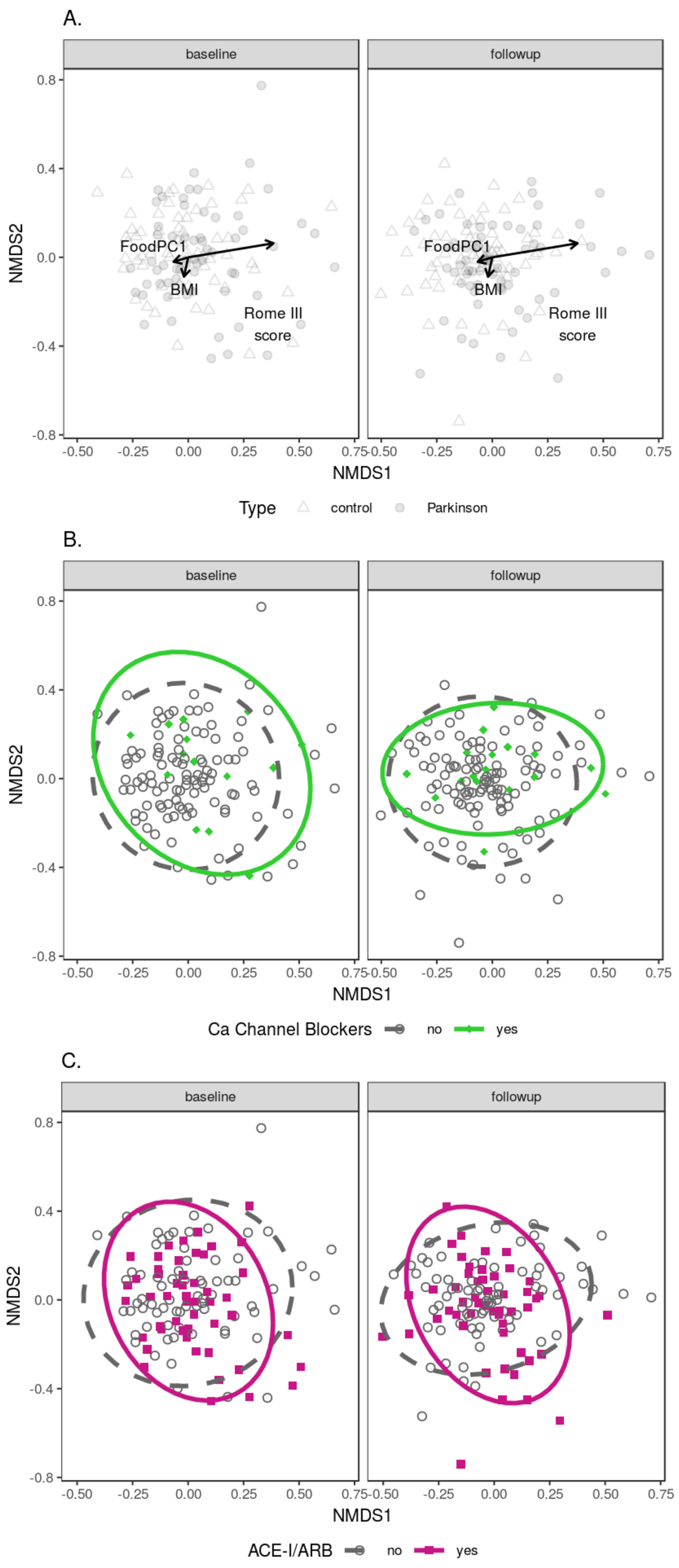
Plots for the main confounding variables:

Figure 8

```
# Pick information for the numeric variables' vectors from envfit:
efVec <- as.data.frame(scores(efGen, display = "vectors"))
efVec <- cbind(efVec, Variable = rownames(efVec))

# Hand-adjusted positions for variable labels:
efVec$TextX <- efVec$NMDS1
efVec$TextY <- efVec$NMDS2-0.055
efVec$Labels <- as.character(efVec$Variable)
efVec$Labels[efVec$Labels == "Rome_III_constip_defec_sumscore_9.15"] <- "Rome III\nscore"
efVec["FoodPC1", "TextY"] <- 0.05
efVec["FoodPC1", "TextX"] <- (-0.155)
efVec["Rome_III_constip_defec_sumscore_9.15", "TextY"] <- (-0.3)

# Plot for numeric confounders
pdMDSplotEVec <- ggplot(pdmdsGendf) +
  geom_point(mapping = aes(x = NMDS1, y = NMDS2,
                           shape = Parkinson),
             size = 1.5, alpha = 0.1) +
    coord_fixed() +
  ggtitle("A.") +
  theme_bw(base_size = 8) +
    scale_shape_manual(values = c(2, 19), name = "Type") +
    geom_segment(data = efVec,
          aes(x = 0, xend = NMDS1, y = 0, yend = NMDS2), lwd = 0.5,
          arrow = arrow(length = unit(0.15, "cm")), colour = "black") +
    geom_text(data = efVec, aes(x = TextX, y = TextY, label = Labels),
        size = 2.5) +
```

```r
    theme(legend.position = "bottom",
          panel.grid.minor = element_blank(),
          panel.grid.major = element_blank(),
          legend.box.margin = ggplot2::margin(c(-10,0,-10,0)))

# Plot for calcium channel blocker medication
pdMDSplotECA <- ggplot(pdmdsGendf) +
  geom_point(mapping = aes(x = NMDS1, y = NMDS2,
                           col = meds_ca_antagonist, shape = meds_ca_antagonist),
             size = 1.5) +
    coord_fixed() +
  ggtitle("B.") +
  theme_bw(base_size = 8) +
    scale_color_manual(values = c("gray40", "limegreen"),
                       name = "Ca Channel Blockers",
                       labels = c("no", "yes")) +
    scale_shape_manual(values = c(1,18),
                       name = "Ca Channel Blockers",
                       labels = c("no", "yes")) +
    scale_linetype_manual(values = c(2,1),
                          name = "Ca Channel Blockers",
                          labels = c("no", "yes")) +
    stat_ellipse(aes(x = NMDS1, y = NMDS2,
                     lty = meds_ca_antagonist, color = meds_ca_antagonist),
                 level = 0.95, lwd = 1) +
    theme(legend.position = "bottom",
          panel.grid.minor = element_blank(),
          panel.grid.major = element_blank(),
          legend.box.margin = ggplot2::margin(c(-10, 0, -10, 0)))

# Plot for ACE-I/ARB medication
pdMDSplotEACE <- ggplot(pdmdsGendf) +
  geom_point(mapping = aes(x = NMDS1, y = NMDS2,
                           col = meds_ACEI_ARB, shape = meds_ACEI_ARB),
             size = 1.5) +
    coord_fixed() +
  ggtitle("C.") +
  theme_bw(base_size = 8) +
    scale_color_manual(values = c("gray40", "mediumvioletred"),
                       name = "ACE-I/ARB",
                       labels = c("no", "yes")) +
    scale_shape_manual(values = c(1,15),
                       name = "ACE-I/ARB",
                       labels = c("no", "yes")) +
    scale_linetype_manual(values = c(2,1),
                          name = "ACE-I/ARB",
                          labels = c("no", "yes")) +
    stat_ellipse(aes(x = NMDS1, y = NMDS2,
                     lty = meds_ACEI_ARB, color = meds_ACEI_ARB),
        level = 0.95, lwd = 1) +
    theme(legend.position = "bottom",
          panel.grid.minor = element_blank(),
          panel.grid.major = element_blank(),
          legend.box.margin = ggplot2::margin(c(-10, 0, -10, 0)))

# Plot all three together:
fig8 <- arrangeGrob(
    pdMDSplotEVec + facet_grid(~Timepoint),
    pdMDSplotECA + facet_grid(~Timepoint),
    pdMDSplotEACE + facet_grid(~Timepoint),
    ncol = 1)
```

```
grid.arrange(fig8)
```



```
## Export to pdf
ggsave(fig8, filename = "Outputs/figure8.pdf", device = cairo_pdf,
       height = maxhi*0.9, width = halfpage, units = "in")
```

Diet

```
## Only controls

# Make a subsetted, subsampled phyloseq object and calculate distance matrix
ffqPhyC <- subset_samples(ffqPhy, Parkinson == "control")
ffqPhyCR <- rarefy_even_depth(ffqPhyC, rngseed = 859410)
ffqDistC <- vegdist(t(as.data.frame(as.matrix(otu_table(ffqPhyCR)))))

ffqAdC <- sapply(ffqVars, function(x) adonis(ffqDistC ~ sample_data(ffqPhyCR)[[x]], perm =
    999)$aov.tab[1,"Pr(>F)"])

# Full data, correcting for PD status, OTU level

# Make a subsampled phyloseq object and calculate distance matrix
ffqPhyR <- rarefy_even_depth(ffqPhy, rngseed = 111213)
ffqOTUDist <- vegdist(t(as.data.frame(as.matrix(otu_table(ffqPhyR)))))

ffqAd <- sapply(ffqVars, function(x) adonis2(ffqOTUDist ~ ., as(sample_data(ffqPhyR)[,
    c("Parkinson", x)], "data.frame"), perm = 999, by = "margin")[2, "Pr(>F)"])
```

Collect the results into a table:

Table S3

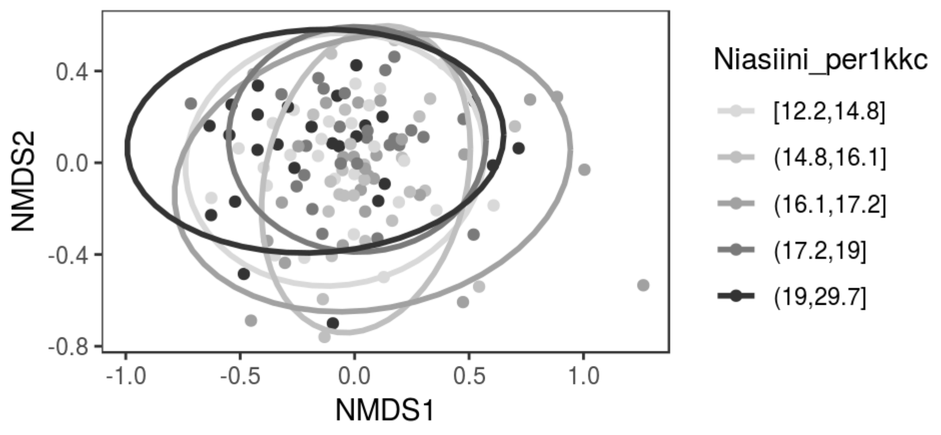
```
ffqAdFull <- as.data.frame(cbind(ffqAdC, ffqAd))
colnames(ffqAdFull) <- c("OnlyC", "PD_C_OTU")
ffqAdFull <- ffqAdFull[order(rowSums(ffqAdFull < 0.05), decreasing = TRUE),]

kable(subset(ffqAdFull, OnlyC < 0.05 | PD_C_OTU < 0.05))
```

	OnlyC	PD_C_OTU
Niasiini_per1kkc	0.017	0.012
FoodPC1	0.038	0.026
Prot_E	0.036	0.145
Lino_E	0.645	0.039
Sterol_per1kkc	0.098	0.001
Lakto_g_per1kkc	0.161	0.047
E_vitam_per1kkc	0.291	0.042
Leipayht_per1kkc	0.026	0.141
Puuro_per1kkc	0.410	0.047
Voiriini_per1kkc	0.049	0.085
Vihantuo_per1kkc	0.038	0.134
Hedmeyh_per1kkc	0.004	0.273
Karkit_per1kkc	0.501	0.047
Virvsok_per1kkc	0.015	0.095

```
## Export
write.table(ffqAdFull, "Outputs/table_s3.csv")
```

Plot for niacin intake (“Niasiini_per1kkc”) which was the only variable in addition to FoodPC1 that was significant in both comparisons:

```
ffq_phy_ord <- ordinate(ffqPhyR, "NMDS", "bray", try = 400)

plot_ordination(ffqPhyR, ffq_phy_ord, color = "Niasiini_per1kkc") +
  theme_bw() +
  coord_fixed() +
  stat_ellipse(level = 0.95, lwd = 1) +
  scale_color_manual(values = gray.colors(n = 5, start = 0.85, end = 0.2)) +
  theme(panel.grid = element_blank())
```



Not the clearest picture (what with five categories in the plot), but there could potentially be something here.

Progression

Data setup

Set up PD-only data for progression beta diversity analyses

```
# Variables of interest:
# combine the ones that differed between progression groups in the initial metadata comparisons
# and those that were correlated with PD status:
adVarsPD <- union(adVarsPD, prog_sigvars)

# Drop some variables, including the ones used in progression categorization
adVarsPD <- adVarsPD[-c(grep("LED", adVarsPD), grep("UPDRS", adVarsPD))]
adVarsPD
```

```
##  [1] "GDS_15"                 "meds_dopa"
##  [3] "meds_dopamine_agonist"  "meds_MAO_inhibitor"
##  [5] "NMSQuest_total"         "NMSS_total"
##  [7] "RBDSQ"                  "SCS_PD_total"
##  [9] "SDQ_total"              "sniffinsticks"
## [11] "meds_COMT_inhibitor"    "meds_ASA"
## [13] "meds_statin"            "ropinirole_mg"
## [15] "pigd_score_jankovic_ON" "levodopa_entacapone_mg"
## [17] "entacapone_mg"          "pramipexole_mg"
```

```
# Subset data and drop three subjects who have missing values for Sniffinsticks
sort(colSums(is.na(sample_data(progPhy)[, adVarsPD])), decreasing = TRUE)
```

```
##          sniffinsticks                 GDS_15              meds_dopa
##                      3                      0                      0
##  meds_dopamine_agonist     meds_MAO_inhibitor         NMSQuest_total
##                      0                      0                      0
##             NMSS_total                  RBDSQ           SCS_PD_total
##                      0                      0                      0
##              SDQ_total    meds_COMT_inhibitor               meds_ASA
##                      0                      0                      0
##            meds_statin          ropinirole_mg pigd_score_jankovic_ON
##                      0                      0                      0
## levodopa_entacapone_mg          entacapone_mg         pramipexole_mg
##                      0                      0                      0
```

```
# Metadata
progmetaAD <- subset(progmeta, !is.na(progmeta$sniffinsticks))
```

```
# OTU level
progPhy_R <- rarefy_even_depth(subset_samples(progPhy, !is.na(sample_data(progPhy)$sniffinsticks)),
    rngseed = 232157)

# Genus level
progPhyGen_R <- rarefy_even_depth(subset_samples(progPhyGen,
    !is.na(sample_data(progPhyGen)$sniffinsticks)), rngseed = 861259)

# Family level
progPhyFam_R <- rarefy_even_depth(subset_samples(progPhyFam,
    !is.na(sample_data(progPhyFam)$sniffinsticks)), rngseed = 948329)

# Calculate distance matrices
progdisOTU <- vegdist(t(as.data.frame(as.matrix(otu_table(progPhy_R)))))
progdisGen <- vegdist(t(as.data.frame(as.matrix(otu_table(progPhyGen_R)))))
progdisFam <- vegdist(t(as.data.frame(as.matrix(otu_table(progPhyFam_R)))))
```

ADONIS

Adonis for progression without confounders, timepoint in model:

```
adCatOTUs <- adonis2(progdisOTU ~ ., progmetaAD[, c("Timepoint", "ProgCat")], perm = 9999, by =
    "margin")
adCatGen <- adonis2(progdisGen ~ ., progmetaAD[, c("Timepoint", "ProgCat")], perm = 9999, by =
    "margin")
adCatFam <- adonis2(progdisFam ~ ., progmetaAD[, c("Timepoint", "ProgCat")], perm = 9999, by =
    "margin")
```

Comparisons of potential confounders in single-variable models:

```
pdADONISprog <- data.frame(row.names = adVarsPD)

# Function for grabbing the p-value from comparisons
adloopProg <- function(dist, meta, varlist){
    adres <- sapply(varlist, function (x) adonis2(dist ~ ., meta[, c("Timepoint", "ProgCat", x)], by
        = "margin")["Pr(>F)"][3,])
    return(adres)
}

set.seed(13524508)

pdADONISprog$p_OTU <- adloopProg(progdisOTU, progmetaAD, adVarsPD)
pdADONISprog$p_Gen <- adloopProg(progdisGen, progmetaAD, adVarsPD)
pdADONISprog$p_Fam <- adloopProg(progdisFam, progmetaAD, adVarsPD)

pdADONISprog[, "n_sigs"] <- rowSums(pdADONISprog[, 1:3] < 0.05)
pdADONISprog <- pdADONISprog[order(pdADONISprog$n_sigs, decreasing = TRUE),]
```

Table 7B

Export the single-variable results:

```
kable(pdADONISprog)
```

	p_OTU	p_Gen	p_Fam	n_sigs
SDQ_total	0.012	0.007	0.010	3
meds_COMT_inhibitor	0.001	0.001	0.001	3
levodopa_entacapone_-mg	0.001	0.001	0.001	3
entacapone_mg	0.001	0.001	0.001	3
NMSQuest_total	0.003	0.018	0.137	2
SCS_PD_total	0.001	0.005	0.166	2
meds_statin	0.010	0.038	0.065	2
GDS_15	0.016	0.395	0.308	1

	p_OTU	p_Gen	p_Fam	n_sigs
meds_dopa	0.025	0.061	0.088	1
NMSS_total	0.018	0.070	0.306	1
RBDSQ	0.007	0.131	0.079	1
ropinirole_mg	0.365	0.045	0.223	1
meds_dopamine_-agonist	0.050	0.365	0.074	0
meds_MAO_inhibitor	0.545	0.826	0.442	0
sniffinsticks	0.422	0.342	0.093	0
meds_ASA	0.430	0.824	0.943	0
pigd_score_jankovic_-ON	0.057	0.176	0.484	0
pramipexole_mg	0.075	0.060	0.059	0

```r
## Export
write.csv(pdADONISprog, "Outputs/table_7b.csv")
```

COMT inhibitors are significant on all three levels as three differently measured variables (meds_COMT_inhibitor (yes/no), levodopa_entacapone_mg (numeric, combined medication variable), and entacapone_mg (numeric variable for amount of this COMT inhibitor). SDQ score and statins are also significant on all levels.

Combined model with the significant confounders from the above comparisons:

```r
#  Trim set of confounders:
adVarsPDsel <- rownames(pdADONISprog[which(pdADONISprog$n_sigs > 0),])
adVarsPDsel <- adVarsPDsel[-grep("entacapone", adVarsPDsel)] # leave out the "mg" variables for COMT

## Run full models, with progression and confounders
adprog_cat_OTU <- adonis2(progdisOTU ~ ., progmetaAD[, c("Timepoint", "ProgCat", adVarsPDsel)], perm
    = 9999, by = "margin")
adprog_cat_Gen <- adonis2(progdisGen ~ ., progmetaAD[, c("Timepoint", "ProgCat", adVarsPDsel)], perm
    = 9999, by = "margin")
adprog_cat_Fam <- adonis2(progdisFam ~ ., progmetaAD[, c("Timepoint", "ProgCat", adVarsPDsel)], perm
    = 9999, by = "margin")
```

Collect the results of the two different adonis models:

```r
# Make data frame
progBdivs <- as.data.frame(matrix(ncol = 5 + length(adVarsPDsel), nrow = 9))
colnames(progBdivs) <- c("Test", "Model", "Level", "Timepoint", "ProgCat", adVarsPDsel)
progBdivs$Test <- c(rep("adonis2", 6), rep("envfit", 3))
progBdivs$Model <- c(rep("TP+Prog", 3), rep("TP+Prog+confounders", 6))
progBdivs$Level <- rep(c("OTU", "Genus", "Family"), 3)

# Progression without confounders
progBdivs[1, 4:5] <- adCatOTUs[1:2, 5]
progBdivs[2, 4:5] <- adCatGen[1:2, 5]
progBdivs[3, 4:5] <- adCatFam[1:2, 5]

# Progression with confounders
progBdivs[4, 4:ncol(progBdivs)] <- adprog_cat_OTU[colnames(progBdivs)[4:ncol(progBdivs)], 5]
progBdivs[5, 4:ncol(progBdivs)] <- adprog_cat_Gen[colnames(progBdivs)[4:ncol(progBdivs)], 5]
progBdivs[6, 4:ncol(progBdivs)] <- adprog_cat_Fam[colnames(progBdivs)[4:ncol(progBdivs)], 5]
```

Envfit

```r
# Drop a few variables that were not significant in the adonis comparisons above
progEnvVars <- names(which(colSums(progBdivs[4:6, 5:ncol(progBdivs)] < 0.05) > 0))

# Use the select confounders as environmental variables in vegan metaMDS

efOTUprog <- envfit(metaMDS(t(as.data.frame(as.matrix(otu_table(progPhy_R)))), try = 500),
    progmetaAD[, c("Timepoint", "ProgCat", progEnvVars)], permu = 9999, na.rm = TRUE)

efGenprog <- envfit(metaMDS(t(as.data.frame(as.matrix(otu_table(progPhyGen_R)))), try = 500),
    progmetaAD[, c("Timepoint", "ProgCat", progEnvVars)], permu = 9999, na.rm = TRUE)
```

```
efFamprog <- envfit(metaMDS(t(as.data.frame(as.matrix(otu_table(progPhyFam_R)))), try = 500),
    progmetaAD[, c("Timepoint", "ProgCat", progEnvVars)], permu = 9999, na.rm = TRUE)
```

Combined results

Collect the results of the adonis and envfit comparisons and export:

Table 7AC

```
## Add results from envfit to previous data frame
progefobjs <- list(efOTUprog, efGenprog, efFamprog)

# Vectors
progBdivs[7:9, gsub("pvals.", "", names(unlist(progefobjs[[1]][["vectors"]]["pvals"])))] <-
    unlist(sapply(progefobjs, function(x) x[["vectors"]]["pvals"]))

# Factors
progBdivs[7:9, gsub("pvals.", "", names(unlist(progefobjs[[1]][["factors"]]["pvals"])))] <-
    unlist(sapply(progefobjs, function(x) x[["factors"]]["pvals"]))
```

Table shown split into two parts to fit the page:

```
kable(progBdivs[, 1:9], digits = 4)
```

Test	Model	Level	Timepoint	ProgCat	SDQ_total	meds_COMT_inhibitor	NMSQuest_total	SCS_PD_total
adonis2	TP+Prog	OTU	1.0000	0.2451	NA	NA	NA	NA
adonis2	TP+Prog	Genus	0.9448	0.3443	NA	NA	NA	NA
adonis2	TP+Prog	Family	0.8504	0.1422	NA	NA	NA	NA
adonis2	TP+Prog+confounders	OTU	0.8647	0.3956	0.1143	0.0005	0.0357	0.0018
adonis2	TP+Prog+confounders	Genus	0.5149	0.3344	0.1155	0.0011	0.0265	0.0353
adonis2	TP+Prog+confounders	Family	0.5823	0.2040	0.0670	0.0005	0.3213	0.2705
envfit	TP+Prog+confounders	OTU	0.7974	0.3302	NA	0.0001	0.2269	0.0312
envfit	TP+Prog+confounders	Genus	0.3717	0.5918	NA	0.2945	0.0237	0.4271
envfit	TP+Prog+confounders	Family	0.0738	0.8894	NA	0.8875	0.1016	0.4790

```
kable(progBdivs[, c(1:3, 10:15)], digits = 4)
```

Test	Model	Level	meds_statin	GDS_15	meds_dopa	NMSS_total	RBDSQ	ropinirole_mg
adonis2	TP+Prog	OTU	NA	NA	NA	NA	NA	NA
adonis2	TP+Prog	Genus	NA	NA	NA	NA	NA	NA
adonis2	TP+Prog	Family	NA	NA	NA	NA	NA	NA
adonis2	TP+Prog+confounders	OTU	0.0416	0.1269	0.4210	0.4467	0.0624	0.4110
adonis2	TP+Prog+confounders	Genus	0.2135	0.8041	0.3441	0.2155	0.5134	0.0976
adonis2	TP+Prog+confounders	Family	0.4417	0.6596	0.2083	0.4982	0.3120	0.4395
envfit	TP+Prog+confounders	OTU	0.4745	NA	NA	NA	NA	NA
envfit	TP+Prog+confounders	Genus	0.0003	NA	NA	NA	NA	NA
envfit	TP+Prog+confounders	Family	0.0054	NA	NA	NA	NA	NA

```
# col.names = c("Test", "Model", "Level", "Timepoint", "ProgCat", "SDQ", "COMT", "NMSQuest",
    "SCS-PD", "statins", "GDS15", "L-dopa", "NMSS", "RBDSQ", "ropinirole (mg)"

## Export
write.csv(progBdivs, "Outputs/table_7ac.csv")
```

PD phenotypes (TD vs PIGD)

Test without confounders using adonis, both with timepoint included in the model, or separately for each timepoint, starting with OTU level only.

```
# Subsample data
progPhyPheR <- rarefy_even_depth(progPhyPhe, rngseed = 987651)

# Calculate distance matrix
```

```
progPhyPheDist <- vegdist(t(as.data.frame(as.matrix(otu_table(progPhyPheR)))), method = "bray")

# Run model with all data, timepoint in model
adonis2(progPhyPheDist ~ sample_data(progPhyPheR)[["JankovicClass"]] +
    sample_data(progPhyPheR)[["Timepoint"]], perm = 9999)
```

```
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 9999
##
## adonis2(formula = progPhyPheDist ~ sample_data(progPhyPheR)[["JankovicClass"]] +
    sample_data(progPhyPheR)[["Timepoint"]], permutations = 9999)
##                                             Df SumOfSqs      R2      F
## sample_data(progPhyPheR)[["JankovicClass"]]   1   0.2254 0.00888 0.9101
## sample_data(progPhyPheR)[["Timepoint"]]       1   0.1405 0.00553 0.5671
## Residual                                    101  25.0205 0.98559
## Total                                       103  25.3864 1.00000
##                                             Pr(>F)
## sample_data(progPhyPheR)[["JankovicClass"]] 0.5989
## sample_data(progPhyPheR)[["Timepoint"]]     0.9913
## Residual
## Total
```

```
# Model with only baseline data
progPhyPheDistBL <- vegdist(t(as.data.frame(as.matrix(otu_table(subset_samples(progPhyPheR,
    Timepoint == "baseline"))))), method = "bray")
pheMetaBL <- as(sample_data(subset_samples(progPhyPheR, Timepoint == "baseline")), "data.frame")
adonis(progPhyPheDistBL ~ pheMetaBL[["JankovicClass"]], perm = 9999)
```

```
##
## Call:
## adonis(formula = progPhyPheDistBL ~ pheMetaBL[["JankovicClass"]],      permutations = 9999)
##
## Permutation: free
## Number of permutations: 9999
##
## Terms added sequentially (first to last)
##
##                              Df SumsOfSqs MeanSqs F.Model     R2 Pr(>F)
## pheMetaBL[["JankovicClass"]]  1    0.2818 0.28179  1.1212 0.0233 0.2621
## Residuals                    47   11.8121 0.25132         0.9767
## Total                        48   12.0939                 1.0000
```

```
# Model with only follow-up data
progPhyPheDistFU <- vegdist(t(as.data.frame(as.matrix(otu_table(subset_samples(progPhyPheR,
    Timepoint == "followup"))))), method = "bray")
pheMetaFU <- as(sample_data(subset_samples(progPhyPheR, Timepoint=="followup")), "data.frame")
adonis(progPhyPheDistFU ~ pheMetaFU[["JankovicClass"]], perm = 9999)
```

```
##
## Call:
## adonis(formula = progPhyPheDistFU ~ pheMetaFU[["JankovicClass"]],      permutations = 9999)
##
## Permutation: free
## Number of permutations: 9999
##
## Terms added sequentially (first to last)
##
##                              Df SumsOfSqs MeanSqs F.Model      R2 Pr(>F)
## pheMetaFU[["JankovicClass"]]  1    0.1968 0.19677 0.80491 0.01496 0.7905
## Residuals                    53   12.9564 0.24446         0.98504
## Total                        54   13.1532                 1.00000
```

The results offer no support for a difference in beta diversity between the two PD phenotypes on the OTU level, and since this is the level with the highest resolution, it seems unlikely that there would be differences in summarized data on different taxonomic levels, either.

Differential abundance

PD vs control

Data filtering

Trim to taxa that are present in around 1/10 samples and drop the "unclassified" bin (which is basically just a garbage bin of taxa that were unclassified at the selected taxonomic level).

```
round((nrow(pdclin) / 10))
```

```
## [1] 26
```

```
trim_co1 <- round((nrow(pdclin) / 10))

newPDtrim <- filter_taxa(newPD, function(x) sum(x > 1) > trim_co1 & sum(x) > 999, prune = TRUE)

newPDtrimGen <- filter_taxa(collapseTaxLevel(newPD, level = "Genus", fixUnclassifieds = FALSE),
    function(x) sum(x > 1) > trim_co1, prune = TRUE)
newPDtrimGen <- subset_taxa(newPDtrimGen, Genus != "unclassified")

newPDtrimFam <- filter_taxa(collapseTaxLevel(newPD, level = "Family", fixUnclassifieds = FALSE),
    function(x) sum(x > 1) > trim_co1, prune = TRUE)
newPDtrimFam <- subset_taxa(newPDtrimFam, Family != "unclassified")
```

Regarding confounder selection, based on the diversity comparisons, we decided to use BMI and the Rome III score, which were the most consistently significant in various different models and subsets of data.

ANCOM

Comparisons with ANCOM were run separately for each timepoint and corrected for Rome III score and BMI. The version of the program used was an early version of ANCOM 2.

```
source("Inputs/ANCOM_updated_code.R")
library("exactRankTests")
library("nlme")
library("tidyr")

ancom2CvsPD <- function(phyloObj, adjp = 2){
  otu_data <- data.frame(Sample.ID = sample_data(phyloObj)$Subject,
                     as.data.frame(as.matrix(t(otu_table(phyloObj)))))
  meta_data <- data.frame(Sample.ID = sample_data(phyloObj)$Subject,
                        Parkinson = sample_data(phyloObj)$Parkinson,
                        Rome = sample_data(phyloObj)$Rome_III_constip_defec_sumscore_9.15,
                        BMI = sample_data(phyloObj)$BMI)
  res <- ANCOM.main(OTUdat = otu_data,
                  Vardat = meta_data,
                  adjusted = TRUE,
                  repeated = FALSE,
                  main.var = "Parkinson",
                  adj.formula = "Rome + BMI",
                  repeat.var = NULL,
                  longitudinal = FALSE,
                  random.formula = NULL,
                  multcorr = adjp,
                  sig = 0.05,
                  prev.cut = 0.9)
```

```r
  return(res)
}

getAnRes <- function(resdf){
  hits <- as.character(resdf$W.taxa[resdf$W.taxa$detected_0.6 == TRUE, "otu.names"])
  return(hits)
}

## PD vs control

# OTUs
# NB: the OTU-level ANCOM comparisons take a while (but less than an hour)

# Baseline
ancomOTUsBL <- ancom2CvsPD(subset_samples(newPDtrim, Timepoint == "baseline"))
# Followup
ancomOTUsFU <- ancom2CvsPD(subset_samples(newPDtrim, Timepoint == "followup"))

# Genera

# Baseline
ancomGenBL <- ancom2CvsPD(subset_samples(newPDtrimGen, Timepoint == "baseline"))
# Followup
ancomGenFU <- ancom2CvsPD(subset_samples(newPDtrimGen, Timepoint == "followup"))

# Families

# Baseline
ancomFamBL <- ancom2CvsPD(subset_samples(newPDtrimFam, Timepoint == "baseline"))
# Followup
ancomFamFU <- ancom2CvsPD(subset_samples(newPDtrimFam, Timepoint == "followup"))
```

Save the results from ANCOM to a table:

Table S4A

```r
# Collect results
ancomRes <- data.frame(Level = c(rep("OTU", 2), rep("Genus", 2), rep("Family",2)),
                       Timepoint = rep(c("baseline", "followup"), 3))
ancomRes$List <- c(paste(getAnRes(ancomOTUsBL), collapse = ","),
                   paste(getAnRes(ancomOTUsFU), collapse = ","),
                   paste(getAnRes(ancomGenBL), collapse = ","),
                   paste(getAnRes(ancomGenFU), collapse = ","),
                   paste(getAnRes(ancomFamBL), collapse = ","),
                   paste(getAnRes(ancomFamFU), collapse = ","))
ancomRes <- separate_rows(ancomRes, List, sep = ",")
colnames(ancomRes) <- c("Level", "Timepoint", "Taxon")

# Add genus classifications to OTU level results for exporting
ancomResExport <- ancomRes
ancomResExport$Taxon <- as.character(ancomResExport$Taxon)
ancomResExport[ancomResExport$Level == "OTU", "Taxon"] <- paste(ancomResExport[ancomResExport$Level
    == "OTU", "Taxon"], " (", gsub("_", " ", tax_table(newPD)[ancomResExport[ancomResExport$Level ==
    "OTU", "Taxon"],"Genus"]), ")", sep = "")
ancomResExport <- ancomResExport[order(ancomResExport$Level, ancomResExport$Timepoint), ]
rownames(ancomResExport) <- NULL
```

```r
kable_styling(kable(ancomResExport), font_size = 10, full_width = FALSE)
```

Level	Timepoint	Taxon
Family	baseline	Bifidobacteriaceae
Family	baseline	Prevotellaceae
Family	followup	Bifidobacteriaceae

Level	Timepoint	Taxon
Family	followup	Prevotellaceae
Family	followup	Puniceicoccaceae
Genus	baseline	Bifidobacterium
Genus	followup	Bifidobacterium
Genus	followup	Roseburia
Genus	followup	Prevotella
OTU	baseline	Otu0030 (Alistipes)
OTU	baseline	Otu0104 (Clostridium IV)
OTU	baseline	Otu0007 (Bifidobacterium)
OTU	baseline	Otu0377 (Ruminococcaceae unclassified)
OTU	baseline	Otu0059 (Ruminococcaceae unclassified)
OTU	baseline	Otu0217 (Intestinimonas)
OTU	followup	Otu0104 (Clostridium IV)
OTU	followup	Otu0109 (Ruminococcus)
OTU	followup	Otu0105 (Oscillibacter)
OTU	followup	Otu0051 (Clostridium sensu stricto)
OTU	followup	Otu0078 (Firmicutes unclassified)
OTU	followup	Otu0131 (Bacteroides)
OTU	followup	Otu0007 (Bifidobacterium)
OTU	followup	Otu0129 (Clostridium XlVa)

```r
## Export:
write.csv(ancomResExport, "Outputs/table_s4a.csv")

# Rearrange for downstream comparisons
ancomRes <- ancomRes[, c("Taxon", "Level", "Timepoint")]
ancomRes$pval <- NA
```

Random forests

```r
library("randomForest")
library("rfUtilities")
library("plyr")
library("rfPermute")
library("e1071")

# Function for random forest classifier
rfCvsPD <- function(phyloObj, rngseed, treen){
    rf.data <- data.frame(t(otu_table(phyloObj)), Parkinson = sample_data(phyloObj)[["Parkinson"]])
    set.seed(rngseed)
    rfres <- rfPermute(Parkinson~., data = rf.data, ntree = treen, num.cores = 4)
    return(rfres)
}

# Function for getting taxon significances
getRFsigs <- function(rfres, lvl){
    rfsig <- as.data.frame(rp.importance(rfres)[, c("MeanDecreaseGini", "MeanDecreaseGini.pval")])
    rfsig$Level <- lvl
    rfsig$Taxon <- rownames(rfsig)
    return(rfsig)
}

# Function for getting model significance
getRFmodelSig <- function(rfres, phyloObj){
    modsig <- rf.significance(rfres, t(otu_table(phyloObj)), num.cores = 4)
    modsigout <- unlist(c(modsig[c("pValue", "test.OOB")], median(modsig$RandOOB)))
    return(modsigout)
}

# Run comparisons

# OTUs

rfOtusBL <- rfCvsPD(subset_samples(newPDtrim, Timepoint == "baseline"), 624443, 500)
rfSigOtusBL <- getRFsigs(rfOtusBL, "OTU")
```

```r
rfOtusFU <- rfCvsPD(subset_samples(newPDtrim, Timepoint == "followup"), 244134, 500)
rfSigOtusFU <- getRFsigs(rfOtusFU, "OTU")

# Genera

rfGenBL <- rfCvsPD(subset_samples(newPDtrimGen, Timepoint == "baseline"), 927991, 500)
rfSigGenBL <- getRFsigs(rfGenBL, "Genus")

rfGenFU <- rfCvsPD(subset_samples(newPDtrimGen, Timepoint == "followup"), 199877, 500)
rfSigGenFU <- getRFsigs(rfGenFU, "Genus")

# Families

rfFamBL <- rfCvsPD(subset_samples(newPDtrimFam, Timepoint == "baseline"), 611582, 500)
rfSigFamBL <- getRFsigs(rfFamBL, "Family")

rfFamFU <- rfCvsPD(subset_samples(newPDtrimFam, Timepoint == "followup"), 882945, 500)
rfSigFamFU <- getRFsigs(rfFamFU, "Family")
```

Table S4B

Save the results of random forests to a table (here showing only 20 first taxa):

```r
# Collect the results
rfRes <- rbind(
  data.frame(rbind(rfSigOtusBL, rfSigGenBL, rfSigFamBL), Timepoint = "baseline"),
  data.frame(rbind(rfSigOtusFU, rfSigGenFU, rfSigFamFU), Timepoint = "followup"))
rfRes$Timepoint <- factor(rfRes$Timepoint, levels = c("baseline", "followup"))
rfRes$Level <- factor(rfRes$Level)

# Trim to significant only and rearrange for final table
rfResSigs <- subset(rfRes, MeanDecreaseGini.pval < 0.05)
rfResSigs <- rfResSigs[,c("Level", "Timepoint", "Taxon", "MeanDecreaseGini",
    "MeanDecreaseGini.pval")]
rfResSigs <- rfResSigs[order(rfResSigs$Level, rfResSigs$Timepoint, rfResSigs$Taxon),]

# Add genus information to OTU level results
rfResSigs[rfResSigs$Level == "OTU", "Taxon"] <- paste(rfResSigs[rfResSigs$Level == "OTU", "Taxon"],
    " (", gsub("_", " ", tax_table(newPD)[rfResSigs[rfResSigs$Level == "OTU", "Taxon"], "Genus"]),
    ")", sep="")
rownames(rfResSigs) <- NULL
```

```r
kable_styling(kable(head(rfResSigs, 20), digits = 4), font_size = 10, full_width = FALSE)
```

Level	Timepoint	Taxon	MeanDecreaseGini	MeanDecreaseGini.pval
Family	baseline	Lachnospiraceae	4.2204	0.0198
Family	baseline	Prevotellaceae	3.4039	0.0099
Family	baseline	Puniceicoccaceae	2.7669	0.0099
Family	baseline	Rikenellaceae	4.0141	0.0297
Family	followup	Bifidobacteriaceae	3.8736	0.0396
Family	followup	Lactobacillaceae	2.7425	0.0396
Family	followup	Puniceicoccaceae	2.0532	0.0099
Genus	baseline	Alistipes	2.2726	0.0099
Genus	baseline	Blautia	1.5800	0.0396
Genus	baseline	Fusicatenibacter	2.0744	0.0099
Genus	baseline	Prevotella	1.6880	0.0495
Genus	baseline	Roseburia	2.2940	0.0198
Genus	followup	Bifidobacterium	2.0608	0.0396
Genus	followup	Butyricicoccus	2.5655	0.0099
Genus	followup	Clostridium_XlVa	2.4202	0.0198
Genus	followup	Faecalicoccus	1.3253	0.0099
Genus	followup	Granulicatella	0.5614	0.0495
Genus	followup	Lachnospira	2.0733	0.0495
Genus	followup	Lactobacillus	1.3817	0.0297
Genus	followup	Prevotella	1.4953	0.0297

```r
# Export
write.csv(rfResSigs, "Outputs/table_s4b.csv")
```

```
# Trim results for downstream comparisons
rfRes <- rfRes[, c("Taxon", "Level", "Timepoint", "MeanDecreaseGini.pval")]
colnames(rfRes) <- c("Taxon", "Level", "Timepoint", "pval")
rownames(rfRes) <- NULL
```

Test for the classifier significance:

```
# NB: this is quite slow.
rfModSigs <- rbind(
    getRFmodelSig(rfOtusBL, subset_samples(newPDtrim, Timepoint == "baseline")),
    getRFmodelSig(rfOtusFU, subset_samples(newPDtrim, Timepoint == "followup")),
    getRFmodelSig(rfGenBL, subset_samples(newPDtrimGen, Timepoint == "baseline")),
    getRFmodelSig(rfGenFU, subset_samples(newPDtrimGen, Timepoint == "followup")),
    getRFmodelSig(rfFamBL, subset_samples(newPDtrimFam, Timepoint == "baseline")),
    getRFmodelSig(rfFamFU, subset_samples(newPDtrimFam, Timepoint == "followup"))
    )
```

Fill in the rest of the details to the table:

Table S5

```
colnames(rfModSigs) <- c("pValue", "realOOB", "permOOB")
rfModSigs <- as.data.frame(rfModSigs)
rfModSigs$Level <- factor(rep(c("OTU", "Genus", "Family"), each = 2))
rfModSigs$Timepoint <- factor(rep(c("baseline", "followup"), 3), levels = c("baseline", "followup"))

rfModSigs <- rfModSigs[, c("Level", "Timepoint", "realOOB", "permOOB", "pValue")]

kable_styling(kable(rfModSigs, digits = 3), full_width = FALSE)
```

Level	Timepoint	realOOB	permOOB	pValue
OTU	baseline	0.375	0.516	0.001
OTU	followup	0.352	0.516	0.000
Genus	baseline	0.430	0.508	0.045
Genus	followup	0.328	0.516	0.000
Family	baseline	0.391	0.516	0.005
Family	followup	0.430	0.516	0.046

```
## Export
write.csv(rfModSigs, "Outputs/table_s5.csv")
```

DESeq2

To use all of the data from both timepoints and to include the subject identity information in the model, we decided to run DESeq2 with the following model: Rome_III_constip_defec_sumscore_9.15 + BMI + Parkinson:DummySubject + Timepoint*Parkinson. However, some subjects had missing values for BMI, and dropping them leads to an unbalanced design where there is one PD patient more than there are controls. To solve this issue (and also producing more robust results as a by-product), we decided to run the comparisons with a leave-one-out approach, dropping each PD patient in turn.

```
# Drop cases with missing BMI values:
missingBMIsamples <- names(which(table(sample_data(subset_samples(newPDtrim, !is.na(BMI)))$Subject)
    < 2))
newPDtrimNoNA <- subset_samples(newPDtrim, !(Subject %in% c(missingBMIsamples)))

# Number of samples remaining in each group:
table(sample_data(newPDtrimNoNA)$Parkinson, sample_data(newPDtrimNoNA)$Timepoint)
```

```
##
##             baseline followup
##   control         61       61
##   Parkinson       62       62
```

```
library("DESeq2")

# Function for running DESeq2 with the selected model + collecting results:
dsPDTP_Conf <- function(phylo0){

  ds0 <- phyloseq_to_deseq2(phylo0, ~ Rome_III_constip_defec_sumscore_9.15 +
                              BMI + Parkinson:DummySubject + Timepoint*Parkinson)

  ds0 <- DESeq(ds0, fitType = "parametric", sfType = "poscounts")

  resFinal <- as.data.frame(matrix(nrow = 0, ncol = 0))

  resDF <- function(contrast = NA, name = NA, outputname){
    if(!is.na(name)){
    resDF <- results(ds0, cooksCutoff = FALSE, name = name)
    } else if (!is.na(contrast)){
    resDF <- results(ds0, cooksCutoff = FALSE, contrast = contrast)
    }
    resDF <- data.frame(cbind(as(resDF, "data.frame"), as(tax_table(phylo0)[rownames(resDF),],
        "matrix")), Contrast = outputname)
    return(resDF)
}

  # PD vs C at baseline
  resFinal <- rbind(resFinal, resDF(name = "Parkinson_Parkinson_vs_control", outputname = "PD vs C,
      baseline"))

  # PD vs C at followup
  resFinal <- rbind(resFinal, resDF(contrast = list(c("Parkinson_Parkinson_vs_control",
      "ParkinsonParkinson.Timepointfollowup")), outputname = "PD vs C, followup"))

  # Taxa that have change differently between timepoints for PD vs C
  resFinal <- rbind(resFinal, resDF(name = "ParkinsonParkinson.Timepointfollowup", outputname = "PD
      vs C, between timepoints"))

  # Confounders: Rome III score
  resFinal <- rbind(resFinal, resDF(name = "Rome_III_constip_defec_sumscore_9.15", outputname =
      "Rome_III_constip_defec_sumscore_9.15"))

  # Confounders: BMI
  resFinal <- rbind(resFinal, resDF(name = "BMI", outputname = "BMI"))

  return(resFinal)
}

# List of all PD subjects:
pdSubj <- as.character(sample_data(subset_samples(newPDtrimNoNA,
        Parkinson=="Parkinson" & Timepoint=="baseline"))$Subject)

# 62-round leave one out loop
# with DS2 console output saved into separate log file
## NB: since this runs DESeq2 62 * 3 times, it takes hours to finish!

otuRes <- list()
genRes <- list()
famRes <- list()

outlog <- file("ds2runlog.txt", open = "wt")
sink(outlog, type = "message")
for(i in pdSubj){
    print(paste("Dropped sample:", i))
    newPDtrim2 <- subset_samples(newPDtrimNoNA, Subject != i)
    sample_data(newPDtrim2)$DummySubject <- factor(rep(1:61, 4))

    newPDtrimGen2 <- collapseTaxLevel(newPDtrim2, level = "Genus", fixUnclassifieds = FALSE)
    newPDtrimGen2 <- subset_taxa(newPDtrimGen2, Genus != "unclassified")

    newPDtrimFam2 <- collapseTaxLevel(newPDtrim2, level = "Family", fixUnclassifieds = FALSE)
    newPDtrimFam2 <- subset_taxa(newPDtrimFam2, Family != "unclassified")

    otuRes[[i]] <- dsPDTP_Conf(newPDtrim2)
    genRes[[i]] <- dsPDTP_Conf(newPDtrimGen2)
    famRes[[i]] <- dsPDTP_Conf(newPDtrimFam2)
}
```

```
## [1] "Dropped sample: P0004"
## [1] "Dropped sample: P0005"
## [1] "Dropped sample: P0008"
## [1] "Dropped sample: P0009"
## [1] "Dropped sample: P0010"
## [1] "Dropped sample: P0011"
## [1] "Dropped sample: P0012"
## [1] "Dropped sample: P0014"
## [1] "Dropped sample: P0015"
## [1] "Dropped sample: P0016"
## [1] "Dropped sample: P0017"
## [1] "Dropped sample: P0018"
## [1] "Dropped sample: P0019"
## [1] "Dropped sample: P0020"
## [1] "Dropped sample: P0024"
## [1] "Dropped sample: P0028"
## [1] "Dropped sample: P0031"
## [1] "Dropped sample: P0034"
## [1] "Dropped sample: P0037"
## [1] "Dropped sample: P0038"
## [1] "Dropped sample: P0042"
## [1] "Dropped sample: P0043"
## [1] "Dropped sample: P0046"
## [1] "Dropped sample: P0047"
## [1] "Dropped sample: P0048"
## [1] "Dropped sample: P0050"
## [1] "Dropped sample: P0051"
## [1] "Dropped sample: P0052"
## [1] "Dropped sample: P0053"
## [1] "Dropped sample: P0056"
## [1] "Dropped sample: P0057"
## [1] "Dropped sample: P0058"
## [1] "Dropped sample: P0059"
## [1] "Dropped sample: P0060"
## [1] "Dropped sample: P0061"
## [1] "Dropped sample: P0063"
## [1] "Dropped sample: P0066"
## [1] "Dropped sample: P0067"
## [1] "Dropped sample: P0068"
## [1] "Dropped sample: P0070"
## [1] "Dropped sample: P0071"
## [1] "Dropped sample: P0072"
## [1] "Dropped sample: P0073"
## [1] "Dropped sample: P0074"
## [1] "Dropped sample: P0077"
## [1] "Dropped sample: P0079"
## [1] "Dropped sample: P0085"
## [1] "Dropped sample: P0087"
## [1] "Dropped sample: P0088"
## [1] "Dropped sample: P0094"
## [1] "Dropped sample: P0095"
## [1] "Dropped sample: P0099"
## [1] "Dropped sample: P0100"
## [1] "Dropped sample: P0103"
## [1] "Dropped sample: P0105"
## [1] "Dropped sample: P0107"
## [1] "Dropped sample: P0114"
## [1] "Dropped sample: P0115"
## [1] "Dropped sample: P0116"
## [1] "Dropped sample: P0118"
## [1] "Dropped sample: P0119"
## [1] "Dropped sample: P0120"
```

```
sink(file = NULL, type = "message")

dsLoopResults<-function(resList){
    resAverages<-resList[[1]][, 7:ncol(resList[[1]])]
    resAverages$meanLog2fc <- rowMeans(rbind(sapply(resList, '[[', "log2FoldChange")), na.rm = TRUE)
    resAverages$sdLog2fc <- rowSds(rbind(sapply(resList, '[[', "log2FoldChange")), na.rm = TRUE)
    resAverages$meanPadj <- rowMeans(rbind(sapply(resList, '[[', "padj")), na.rm = TRUE)
    resAverages$sdPadj <- rowSds(rbind(sapply(resList, '[[', "padj")), na.rm = TRUE)
    return(resAverages)
}

otuResAverages <- dsLoopResults(otuRes)
genResAverages <- dsLoopResults(genRes)
famResAverages <- dsLoopResults(famRes)

dsCvsPDConfOTU <- data.frame(otuResAverages, Level = "OTU")
dsCvsPDConfGen <- data.frame(genResAverages, Level = "Genus")
dsCvsPDConfFam <- data.frame(famResAverages, Genus = NA, Level = "Family")

dsCvsPDRes <- data.frame(rbind(data.frame(rbind(dsCvsPDConfOTU, dsCvsPDConfGen, dsCvsPDConfFam),
    Model = "Conf, dummySubject, loop")))

write.table(file = "ds2_results_full.txt", dsCvsPDRes)
```

Rearrange results for export & downstream comparisons:

Table S4C

```
# Overall trimming of the results table
dsResFull <- dsCvsPDRes[, c("Level", "Contrast", "Family", "Genus", "meanLog2fc", "sdLog2fc",
    "meanPadj", "sdPadj")]
dsResFull$Contrast <- factor(dsResFull$Contrast, levels = c("PD vs C, baseline", "PD vs C, between
    timepoints", "PD vs C, followup", "Rome_III_constip_defec_sumscore_9.15", "BMI"))

# Taxon as a single column
dsResFull[dsResFull$Level == "OTU", "Taxon"] <- substr(rownames(dsResFull[dsResFull$Level ==
    "OTU",]), start = 1, stop = 7)
dsResFull[dsResFull$Level == "Genus", "Taxon"] <- as.vector(dsResFull[dsResFull$Level == "Genus",
    "Genus"])
dsResFull[dsResFull$Level == "Family", "Taxon"] <- as.vector(dsResFull[dsResFull$Level == "Family",
    "Family"])
rownames(dsResFull) <- NULL

dsResFull <- dsResFull[, c("Level", "Contrast", "Taxon", "meanLog2fc", "sdLog2fc", "meanPadj",
    "sdPadj")]

# Trim results for exporting
dsResExport <- subset(dsResFull, meanPadj < 0.05)
dsResExport <- dsResExport[order(dsResExport$Level, dsResExport$Contrast, dsResExport$Taxon),]
```

```
# Add genus classification to the OTUs
dsResExport[dsResExport$Level == "OTU", "Taxon"] <- paste(dsResExport[dsResExport$Level == "OTU",
    "Taxon"], " (", gsub("_", " ", tax_table(newPD)[dsResExport[dsResExport$Level == "OTU",
    "Taxon"], "Genus"]), ")", sep ="")
rownames(dsResExport) <- NULL
```

```
kable_styling(kable(dsResExport, digits = 4), font_size = 10)
```

Level	Contrast	Taxon	meanLog2fc	sdLog2fc	meanPadj	sdPadj
OTU	PD vs C, baseline	Otu0062 (Blautia)	-4.3308	0.1274	0.0485	0.0076
OTU	PD vs C, baseline	Otu0264 (Butyricimonas)	-9.3180	0.0584	0.0470	0.0064
OTU	BMI	Otu0051 (Clostridium sensu stricto)	0.6295	0.0186	0.0000	0.0001
OTU	BMI	Otu0582 (Actinomyces)	-0.2213	0.0164	0.0192	0.1256
Genus	PD vs C, baseline	Clostridium_XlVa	5.3209	0.1209	0.0000	0.0000
Genus	PD vs C, baseline	Clostridium_XVIII	-7.2388	1.1429	0.0428	0.1221
Genus	PD vs C, baseline	Dialister	-11.3419	1.4701	0.0161	0.1267
Genus	PD vs C, baseline	Prevotella	-6.1119	0.1148	0.0417	0.0085
Genus	PD vs C, baseline	Romboutsia	6.5350	0.4349	0.0294	0.0064
Genus	PD vs C, followup	Clostridium_IV	2.7394	0.0838	0.0489	0.0126
Genus	PD vs C, followup	Clostridium_XlVa	5.0995	0.1180	0.0000	0.0000
Genus	PD vs C, followup	Dialister	-11.0257	1.4695	0.0161	0.1265
Genus	PD vs C, followup	Prevotella	-6.2879	0.1085	0.0432	0.0044
Genus	PD vs C, followup	Romboutsia	6.3534	0.4442	0.0431	0.0064
Genus	Rome_III_constip_defec_sumscore_9.15	Bifidobacterium	0.1903	0.0050	0.0006	0.0011
Genus	BMI	Clostridium_sensu_stricto	0.7635	0.0198	0.0000	0.0000
Genus	BMI	Coprococcus	-0.3734	0.0126	0.0051	0.0046
Family	PD vs C, baseline	Porphyromonadaceae	-2.9176	0.1300	0.0031	0.0006
Family	PD vs C, baseline	Prevotellaceae	-7.4107	0.2141	0.0018	0.0006
Family	PD vs C, baseline	Veillonellaceae	-6.5409	0.9537	0.0164	0.1259
Family	PD vs C, followup	Porphyromonadaceae	-2.7642	0.1283	0.0058	0.0011
Family	PD vs C, followup	Prevotellaceae	-7.4819	0.2080	0.0015	0.0006
Family	PD vs C, followup	Veillonellaceae	-7.0958	0.9588	0.0162	0.1268
Family	Rome_III_constip_defec_sumscore_9.15	Bifidobacteriaceae	0.1881	0.0040	0.0002	0.0002
Family	BMI	Clostridiaceae_1	0.7520	0.0206	0.0000	0.0000

```
## Export
write.csv(dsResExport, "Outputs/table_s4c.csv")

# Trim results further for downstream comparisons

dsRes <- subset(dsResFull, Contrast != "Rome_III_constip_defec_sumscore_9.15" & Contrast != "BMI" &
    Contrast != "PD vs C, between timepoints")[, c("meanPadj", "Taxon", "Contrast", "Level")]

# Timepoint
dsRes[grep("baseline", dsRes$Contrast), "Timepoint"] <- "baseline"
dsRes[grep("followup", dsRes$Contrast), "Timepoint"] <- "followup"
dsRes$Timepoint <- factor(dsRes$Timepoint)

# Final table
dsRes <- dsRes[, c("Taxon", "Level", "Timepoint", "meanPadj")]
colnames(dsRes) <- c("Taxon", "Level", "Timepoint", "pval")
```

Compare and contrast the three methods

A rather complicated and long-winded way of making a nice summarized table of taxa of interest for exporting:

```
resAll <- rbind(data.frame(ancomRes, Method = "ANCOM"),
              data.frame(dsRes, Method = "DESeq2"),
              data.frame(rfRes, Method = "RF"))

resAllOnlySigs <- subset(resAll, pval < 0.05 | is.na(pval))
# (the "is.na" part is for ANCOM, which doesn't provide p-values;
# all the taxa listed for ANCOM were flagged as differentially abundant)

## Collect lists of taxa that overlap across methods (significant at either timepoint)

# OTUs
# (none for DS2)
```

```r
sigsPerMethodOTUs <- dcast(subset(resAllOnlySigs, Level == "OTU"), Taxon ~ Method)
overlapSigOTUs <- as.character(sigsPerMethodOTUs[which(rowSums(sigsPerMethodOTUs[, 2:3] > 0) > 1),
    "Taxon"])

# Genera
sigsPerMethodGen <- dcast(subset(resAllOnlySigs, Level == "Genus"), Taxon ~ Method)
overlapSigGen <- as.character(sigsPerMethodGen[which(rowSums(sigsPerMethodGen[, 2:4] > 0) > 1),
    "Taxon"])

# Families
sigsPerMethodFam <- dcast(subset(resAllOnlySigs, Level == "Family"), Taxon ~ Method)
overlapSigFam <- as.character(sigsPerMethodFam[which(rowSums(sigsPerMethodFam[, 2:4] > 0) > 1),
    "Taxon"])

# Collect result tables in comparison to literature
# (using the "litSigs" list of taxa from the literature imported earlier)
# so that the resulting list includes taxa that were either
# 1) detected by multiple tools in the current analyses
# 2) reported in previous literature & have p < 0.1 for at least one tool in the current analyses

# OTUs
# Starting from the list of species and genera from literature:
litSigs <- read.csv("Inputs/sigtaxa_from_literature.csv", sep = "\t")
litGenSpec <- sort(unique(c(as.character(subset(litSigs, Level == "genus")$Taxon.mentioned), gsub("
    .*", "", as.character(subset(litSigs, Level == "species")$Taxon.mentioned)))))

resLitOTUs <- subset(resAll, Level == "OTU")
resLitOTUs$Genus <- tax_table(newPD)[as.character(resLitOTUs$Taxon), "Genus"]
resLitOTUs <- subset(resLitOTUs, Taxon %in% overlapSigOTUs | (Genus %in% litGenSpec & (pval < 0.1 |
    Method == "ANCOM")))

# Genera
resLitGen <- subset(resAll, Taxon %in% overlapSigGen | (Taxon %in% subset(litSigs, Level ==
    "genus")$Taxon.mentioned & (pval < 0.1 | Method =="ANCOM")))

# Families
resLitFam <- subset(resAll, Taxon %in% overlapSigFam | (Taxon %in% subset(litSigs, Level ==
    "family")$Taxon.mentioned & (pval < 0.1 | Method =="ANCOM")))

# Collect the lists of taxa to level-specific vectors:
sharedOtus <- sort(unique(resLitOTUs$Taxon))
sharedOtus <- paste(sharedOtus, tax_table(newPD)[sharedOtus, "Genus"])
sharedGen <- as.character(unique(resLitGen$Taxon))
sharedFam <- as.character(unique(resLitFam$Taxon))

# Make a new dataframe with the relative abundances of these taxa:
sharedSigAbunds <- data.frame(row.names = rownames(sample_data(newPD)))
sharedSigAbunds$Parkinson <- factor(sample_data(newPD)$Parkinson, labels = c("control", "Parkinson"))
sharedSigAbunds$Timepoint <- sample_data(newPD)$Timepoint

sharedSigAbunds <- cbind(sharedSigAbunds,
                        t(prop.table(otu_table(newPD), 2) * 100)[, gsub(" .*", "",  sharedOtus)],
                        t(prop.table(otu_table(newPDgen), 2) * 100)[, sharedGen],
                        t(prop.table(otu_table(newPDfam), 2) * 100)[, sharedFam])

# Function to summarize taxonomy results to a nice table
sigResToTable <- function(fulldf, taxlist){

    df <- subset(fulldf, fulldf$Taxon %in% taxlist)
    df <- df[order(df$Taxon),]
    df$pval <- round(df$pval, digits = 4)
    df[df$Method == "ANCOM", "pval"] <- 0
    df <- dcast(df, Timepoint + Method ~ Taxon, value.var = "pval")

    tspos <- grep("Method", colnames(df)) + 1
    endpos <- ncol(df)
    df[df$Method == "ANCOM", tspos:endpos][df[df$Method == "ANCOM", tspos:endpos] == 0] <-
        "significant"
    df[tspos:endpos][df[tspos:endpos] < 0.001] <- "<0.001"
    df[df$Method == "ANCOM", tspos:endpos][is.na(df[df$Method == "ANCOM", tspos:endpos])] <- "n.s."

    df <- t(df)
    colnames(df) <- paste("p", as.character(df["Method", ]), df["Timepoint", ], sep = "_")
    df <- df[3:nrow(df), ]

    df <- df[taxlist, ]

    return(df)
}

# Function to calculate mean statistics for each taxon in each group
source("Inputs/taxaSummarizer.R")

groupMeanSD <- function(phyloObj, taxlist, var, sigdec = 3){
    df <- relAbundSummary(phyloObj, byVariable = var)[taxlist,]
    meanDF <- sapply(levels(sample_data(phyloObj)[[var]]), function(x)
        as.character(paste(unlist(round(df[paste("Mean", var, x, sep = "_")], sigdec)), " ± ",
        unlist(round(df[paste("SD", var, x, sep = "_")], sigdec)), sep = "")))
    if(length(taxlist) == 1){
      meanDF <- data.frame(Taxon = taxlist, t(meanDF))
    } else {
      meanDF <- as.data.frame(cbind(taxlist, meanDF), stringsAsFactors = FALSE)
    }
    colnames(meanDF) <- c("Taxon", paste("MeanSD", levels(sample_data(phyloObj)[[var]])))
    return(meanDF)
}

# Collect p-values & taxon relative abundances into dataframes per level

# OTUs
resAllSharedOTUsTable <- cbind(sigResToTable(resAll, gsub(" .*", "", sharedOtus)),
    groupMeanSD(newPD, gsub(" .*", "", sharedOtus), "PD_TP"))
rownames(resAllSharedOTUsTable) <- sharedOtus
resAllSharedOTUsTable$LitMatch <- gsub(".* ", "", rownames(resAllSharedOTUsTable)) %in% litGenSpec

# Genera
resAllSharedGenTable <- cbind(sigResToTable(resAll, sharedGen), groupMeanSD(newPDgen, sharedGen,
    "PD_TP"))
resAllSharedGenTable$LitMatch <- resAllSharedGenTable$Taxon %in% subset(litSigs, Level ==
    "genus")$Taxon.mentioned

# Families
resAllSharedFamTable <- cbind(sigResToTable(resAll, sharedFam), groupMeanSD(newPDfam, sharedFam,
    "PD_TP"))
resAllSharedFamTable$LitMatch <- resAllSharedFamTable$Taxon %in% subset(litSigs, Level ==
    "family")$Taxon.mentioned

## Combine the three tables:
resAllSharedTables <- rbind(
    data.frame(resAllSharedFamTable, Level = "Family"),
    data.frame(resAllSharedGenTable, Level = "Genus"),
    data.frame(resAllSharedOTUsTable, Level = "OTU"))

# Reorder columns for export
```

```
resAllSharedTablesTrim <- resAllSharedTables[, c("Level", "LitMatch", "MeanSD.control_baseline",
    "MeanSD.Parkinson_baseline", "p_ANCOM_baseline", "p_RF_baseline", "p_DESeq2_baseline",
    "MeanSD.control_followup", "MeanSD.Parkinson_followup", "p_ANCOM_followup", "p_RF_followup",
    "p_DESeq2_followup")]

# Reorder rows for export
resAllSharedTablesTrim$LitMatch <- factor(resAllSharedTablesTrim$LitMatch)
levels(resAllSharedTablesTrim$LitMatch) <- c("no", "yes")

resAllSharedTablesTrim <- resAllSharedTablesTrim[order(resAllSharedTablesTrim$Level,
    resAllSharedTablesTrim$LitMatch,  decreasing = FALSE),]
```

Table 8

Export table:

```
kable(resAllSharedTablesTrim[, 1:7], col.names = c("Level", "LitMatch", "MeanSD control baseline",
    "MeanSD Parkinson baseline", "p_ANCOM baseline", "p_RF baseline", "p_DESeq2 baseline"))
```

	Level	LitMatch	MeanSD control baseline	MeanSD Parkinson baseline	p_ANCOM baseline	p_RF baseline	p_DESeq2 baseline
Puniceicoccaceae	Family	no	0.045 ± 0.103	0.01 ± 0.029	n.s.	0.0099	0.9928
Bifidobacteriaceae	Family	yes	2.629 ± 2.719	6.528 ± 8.573	significant	0.1287	0.1512
Prevotellaceae	Family	yes	4.345 ± 9.22	0.725 ± 2.12	significant	0.0099	0.0018
Rikenellaceae	Family	yes	2.201 ± 2.035	3.494 ± 2.899	n.s.	0.0297	0.0749
Lachnospiraceae	Family	yes	22.478 ± 10.241	16.48 ± 9.121	n.s.	0.0198	0.9896
Pasteurellaceae	Family	yes	0.036 ± 0.097	0.009 ± 0.027	n.s.	0.0891	0.6251
Lactobacillaceae	Family	yes	0.032 ± 0.124	0.28 ± 1.199	n.s.	0.1287	0.9928
Clostridium_-XlVa	Genus	no	1.971 ± 2.631	1.83 ± 3.265	n.s.	0.1683	<0.001
Bifidobacterium	Genus	yes	2.628 ± 2.717	6.524 ± 8.57	significant	0.0891	0.2442
Roseburia	Genus	yes	7.014 ± 6.944	3.588 ± 4.096	n.s.	0.0198	0.1467
Prevotella	Genus	yes	4.187 ± 9.003	0.659 ± 2.102	n.s.	0.0495	0.0417
Anaerotruncus	Genus	yes	0.056 ± 0.079	0.071 ± 0.084	n.s.	0.0792	0.9863
Blautia	Genus	yes	2.419 ± 1.516	1.829 ± 1.598	n.s.	0.0396	0.7419
Lactobacillus	Genus	yes	0.031 ± 0.125	0.278 ± 1.197	n.s.	0.3564	0.9892
Otu0003 Roseburia	OTU	yes	6.506 ± 6.71	3.115 ± 3.742	n.s.	0.0198	0.3421
Otu0007 Bifidobacterium	OTU	yes	1.563 ± 1.808	4.579 ± 6.456	significant	0.0792	0.6127
Otu0024 Blautia	OTU	yes	0.904 ± 0.875	0.649 ± 0.903	n.s.	0.0198	0.9997
Otu0027 Ruminococcus	OTU	yes	0.679 ± 0.959	0.437 ± 0.895	n.s.	0.0891	0.9997
Otu0030 Alistipes	OTU	yes	0.404 ± 0.861	0.955 ± 1.392	significant	0.0099	0.9997
Otu0036 Roseburia	OTU	yes	0.483 ± 0.757	0.468 ± 1.279	n.s.	0.0891	0.9997
Otu0041 Alistipes	OTU	yes	0.278 ± 0.247	0.448 ± 0.499	n.s.	0.099	0.9997
Otu0055 Prevotella	OTU	yes	0.344 ± 1.243	0.197 ± 1.048	n.s.	0.1287	0.9997
Otu0062 Blautia	OTU	yes	0.446 ± 0.576	0.266 ± 0.358	n.s.	0.0297	0.0485
Otu0098 Bacteroides	OTU	yes	0.105 ± 0.239	0.29 ± 0.599	n.s.	0.0297	0.4988
Otu0109 Ruminococcus	OTU	yes	0.343 ± 1.142	0.122 ± 0.844	n.s.	0.4653	0.9997
Otu0110 Ruminococcus	OTU	yes	0.189 ± 0.332	0.136 ± 0.279	n.s.	0.0198	0.9997
Otu0131 Bacteroides	OTU	yes	0.183 ± 0.654	0.079 ± 0.256	n.s.	0.4059	0.9982
Otu0363 Lactobacillus	OTU	yes	0.019 ± 0.107	0.056 ± 0.225	n.s.	0.7921	0.9997
Otu0379 Alistipes	OTU	yes	0.001 ± 0.003	0.041 ± 0.2	n.s.	0.3168	0.9997
Otu0464 Lactobacillus	OTU	yes	0.002 ± 0.009	0.004 ± 0.016	n.s.	0.4455	0.9997

	Level	LitMatch	MeanSD control baseline	MeanSD Parkinson baseline	p__ANCOM baseline	p__RF baseline	p__DESeq2 baseline
Otu0468 Faecalibacterium	OTU	yes	0.016 ± 0.026	0.009 ± 0.019	n.s.	0.7228	0.9997
Otu0513 Anaerotruncus	OTU	yes	0.006 ± 0.006	0.011 ± 0.011	n.s.	0.0297	0.9976

```r
kable(resAllSharedTablesTrim[, 8:ncol(resAllSharedTablesTrim)], col.names = c("MeanSD control
    followup", "MeanSD Parkinson followup", "p_ANCOM  followup", "p_RF followup", "p_DESeq2
    followup"))
```

	MeanSD control followup	MeanSD Parkinson followup	p__ANCOM followup	p__RF followup	p__DESeq2 followup
Puniceicoccaceae	0.032 ± 0.061	0.01 ± 0.03	significant	0.0099	0.9949
Bifidobacteriaceae	2.189 ± 3.531	5.919 ± 8.256	significant	0.0396	0.1499
Prevotellaceae	2.972 ± 4.882	1.395 ± 3.474	significant	0.0792	0.0015
Rikenellaceae	2.479 ± 2.559	2.836 ± 2.131	n.s.	0.3861	0.1717
Lachnospiraceae	21.787 ± 8.964	17.753 ± 8.198	n.s.	0.4158	0.9949
Pasteurellaceae	0.059 ± 0.28	0.014 ± 0.052	n.s.	0.1584	0.5849
Lactobacillaceae	0.04 ± 0.159	0.226 ± 0.867	n.s.	0.0396	0.9949
Clostridium__XlVa	2.136 ± 2.589	1.501 ± 2.076	n.s.	0.0198	<0.001
Bifidobacterium	2.188 ± 3.53	5.917 ± 8.256	significant	0.0396	0.41
Roseburia	6.683 ± 6.162	4.395 ± 5.298	significant	0.0099	0.1214
Prevotella	2.85 ± 4.854	1.306 ± 3.355	significant	0.0297	0.0432
Anaerotruncus	0.119 ± 0.501	0.098 ± 0.163	n.s.	0.6931	0.99
Blautia	2.63 ± 1.847	2.429 ± 2.334	n.s.	0.6436	0.9269
Lactobacillus	0.04 ± 0.159	0.221 ± 0.862	n.s.	0.0297	0.9964
Otu0003 Roseburia	6.144 ± 5.933	4.109 ± 5.027	n.s.	0.0198	0.2749
Otu0007 Bifidobacterium	1.518 ± 3.155	4.029 ± 6.243	significant	0.0891	0.6924
Otu0024 Blautia	1.057 ± 1.14	0.712 ± 0.884	n.s.	0.0198	0.9998
Otu0027 Ruminococcus	0.832 ± 1.515	0.604 ± 1.23	n.s.	0.9604	0.9998
Otu0030 Alistipes	0.711 ± 1.627	0.623 ± 1.134	n.s.	0.5347	0.9998
Otu0036 Roseburia	0.534 ± 0.771	0.282 ± 0.689	n.s.	0.0297	0.9998
Otu0041 Alistipes	0.33 ± 0.342	0.412 ± 0.37	n.s.	0.7129	0.9998
Otu0055 Prevotella	0.493 ± 1.758	0.342 ± 1.413	n.s.	0.0792	0.9998
Otu0062 Blautia	0.328 ± 0.46	0.217 ± 0.299	n.s.	0.198	0.1111
Otu0098 Bacteroides	0.111 ± 0.22	0.231 ± 0.407	n.s.	0.0792	0.371
Otu0109 Ruminococcus	0.369 ± 1.175	0.001 ± 0.002	significant	0.0099	0.9998
Otu0110 Ruminococcus	0.205 ± 0.519	0.14 ± 0.322	n.s.	0.1386	0.9998
Otu0131 Bacteroides	0.209 ± 0.613	0.049 ± 0.205	significant	0.0198	0.9952
Otu0363 Lactobacillus	0.016 ± 0.122	0.006 ± 0.035	n.s.	0.0495	0.9998
Otu0379 Alistipes	0.004 ± 0.033	0.047 ± 0.236	n.s.	0.0198	0.9998
Otu0464 Lactobacillus	0.001 ± 0.006	0.044 ± 0.236	n.s.	0.0099	0.9998
Otu0468 Faecalibacterium	0.021 ± 0.033	0.007 ± 0.016	n.s.	0.0495	0.9998
Otu0513 Anaerotruncus	0.006 ± 0.005	0.012 ± 0.019	n.s.	0.2079	0.9952

```r
## Export
write.csv(resAllSharedTablesTrim, "Outputs/table_8.csv")
```

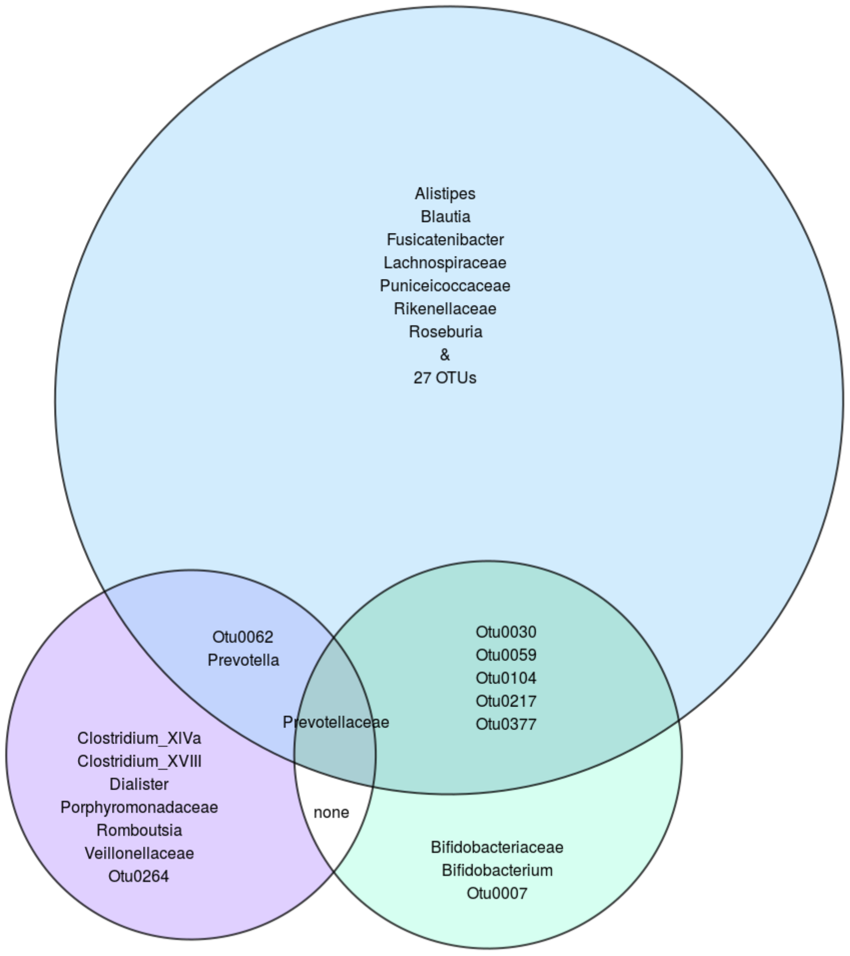
Euler diagrams of the shared taxa in the results lists:

Figure 9

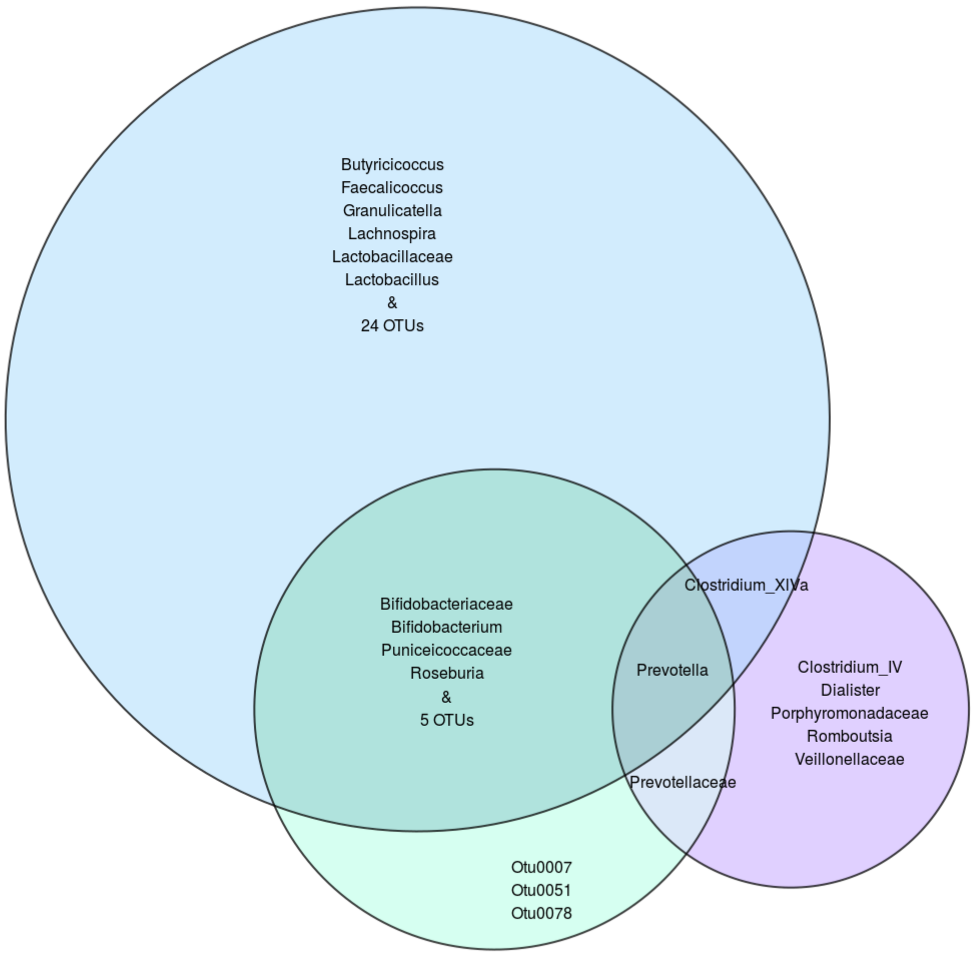
```r
# Summarized data for the plots:
vennDFbl <- table(subset(resAllOnlySigs, Timepoint == "baseline")[, c("Taxon", "Method")])
vennDFfu <- table(subset(resAllOnlySigs, Timepoint == "followup")[, c("Taxon", "Method")])

vennValues <- data.frame(row.names = c("baseline", "followup"))

# Taxa detected by each method alone
vennValues$ANCOM <- c(sum(vennDFbl[, 1]), sum(vennDFfu[, 1]))
vennValues$DESeq2 <- c(sum(vennDFbl[, 2]), sum(vennDFfu[, 2]))
vennValues$RandomForest <- c(sum(vennDFbl[, 3]), sum(vennDFfu[, 3]))
```

```r
# Taxa detected by pairs of methods
vennValues$DS_AN <- c(sum(rowSums(vennDFbl[, 1:2]) == 2), sum(rowSums(vennDFfu[, 1:2]) == 2))
vennValues$AN_RF <- c(sum(rowSums(vennDFbl[, c(1,3)]) == 2), sum(rowSums(vennDFfu[, c(1,3)]) == 2))
vennValues$DS_RF <- c(sum(rowSums(vennDFbl[, 2:3]) == 2), sum(rowSums(vennDFfu[, 2:3]) == 2))

# Taxa detected by all three
vennValues$AN_DS_RF <- c(sum(rowSums(vennDFbl) == 3), sum(rowSums(vennDFfu) == 3))

# Options for plotting:
library("eulerr")

eulerr_options(quantities = list(fontsize = 6))
euCols <- setNames(
  c("#c8ffec", "#d5c0ff", "#c3e6fc", "#CFE0F6", "#95d9d2", "#aec6fc", "#8dbfc6"),
  c("an", "ds", "rf", "ds+an", "an+rf", "ds+rf", "all"))

# Baseline

eulerBL <- as.vector(t(vennValues[1,]))
names(eulerBL) <- c("ANCOM", "DESeq2", "RandomForest", "ANCOM&DESeq2", "DESeq2&RandomForest",
    "ANCOM&RandomForest", "ANCOM&DESeq2&RandomForest")
eulerBLfit <- euler(eulerBL, shape = "circle", input = "union")

# Labels for the detected taxa

taxLabelSort <- function(taxa){
  if(length(grep("Otu", taxa)) < 5){
    tax_sorted <- paste(paste(sort(grep("Otu", taxa, invert = TRUE, value = TRUE)), collapse = "\n"),
                        paste(grep("Otu", taxa, value = TRUE), collapse = "\n"), sep = "\n")
  } else {
    tax_sorted <- paste(paste(sort(grep("Otu", taxa, invert = TRUE, value = TRUE)), collapse="\n"),
        "&", paste(length(grep("Otu", taxa)), "OTUs"), sep = "\n")
  }
  return(tax_sorted)
}

eulerTaxaBL <- vector(length = 7)
names(eulerTaxaBL) <- c("ds", "an", "rf", "ds+an", "ds+rf", "an+rf", "all")

eulerTaxaBL["an"] <- taxLabelSort(names(which(vennDFbl[,"ANCOM"] == 1 & rowSums(vennDFbl) == 1)))
eulerTaxaBL["ds"] <- taxLabelSort(names(which(vennDFbl[,"DESeq2"] == 1 & rowSums(vennDFbl) == 1)))
eulerTaxaBL["rf"] <- taxLabelSort(names(which(vennDFbl[,"RF"] == 1 & rowSums(vennDFbl) == 1)))
eulerTaxaBL["ds+rf"] <- paste(names(which(vennDFbl[which(rowSums(vennDFbl) == 2),][,"ANCOM"] == 0)),
    collapse = "\n")
eulerTaxaBL["an+rf"] <- paste(names(which(vennDFbl[which(rowSums(vennDFbl)==2),][,"DESeq2"] == 0)),
    collapse = "\n")
eulerTaxaBL["ds+an"] <- "none"
eulerTaxaBL["all"] <- names(which(rowSums(vennDFbl) == 3))

euColsBL <- euCols[names(eulerTaxaBL)]
euColsBL["ds+an"] <- "white"
euBL <- plot(eulerBLfit, labels = FALSE, alpha = 0.75, quantities = eulerTaxaBL, fills = euColsBL)

# Follow-up

eulerTaxaFU <- vector(length = 7)
names(eulerTaxaFU) <- names(euCols)

eulerTaxaFU["an"] <- taxLabelSort(names(which(vennDFfu[,"ANCOM"] == 1 & rowSums(vennDFfu) == 1)))
eulerTaxaFU["ds"] <- taxLabelSort(names(which(vennDFfu[,"DESeq2"] == 1 & rowSums(vennDFfu) == 1)))
eulerTaxaFU["rf"] <- taxLabelSort(names(which(vennDFfu[,"RF"] == 1 & rowSums(vennDFfu) == 1)))
```

```r
eulerTaxaFU["ds+rf"] <- names(which(vennDFfu[which(rowSums(vennDFfu) == 2),][,"ANCOM"] == 0))
eulerTaxaFU["an+rf"] <- taxLabelSort(names(which(vennDFfu[which(rowSums(vennDFfu)==2),][,"DESeq2"]
    == 0)))
eulerTaxaFU["ds+an"] <- names(which(vennDFfu[which(rowSums(vennDFfu) == 2),][,"RF"] == 0))
eulerTaxaFU["all"] <- names(which(rowSums(vennDFfu) == 3))

eulerFU <- as.vector(t(vennValues[2,]))
names(eulerFU) <- c("ANCOM", "DESeq2", "RandomForest", "ANCOM&DESeq2", "ANCOM&RandomForest",
    "DESeq2&RandomForest", "ANCOM&DESeq2&RandomForest")
eulerFUfit <- euler(eulerFU, shape = "circle", input = "union")

# Make plots

library("grid")

euBL <- plot(eulerBLfit, labels = FALSE, alpha = 0.75, quantities = eulerTaxaBL, fills = euColsBL)
euBL
```



```r
euFU <- plot(eulerFUfit, labels = FALSE, alpha = 0.75, quantities = eulerTaxaFU, fills = euCols)
euFU
```



```r
blank_panel <- grid::rectGrob(gp = grid::gpar(col = "white"))

## Export to pdf
## (with added, manually placed text labels)

pdf("Outputs/fig9_temp.pdf", width = halfpage, height = maxhi*0.8, useDingbats = FALSE)

grid.arrange(euBL, blank_panel, euFU, ncol = 1, heights = c(1, 0.1, 1))
grid.text("A.", gp = gpar(font = 2, size = 12), x = unit(0.05, "npc"), y = unit(0.98, "npc"))
grid.text("B.", gp = gpar(font = 2, size = 12), x = unit(0.05, "npc"), y = unit(0.46, "npc"))

grid.text("RandomForest", gp = gpar(font = 2), x = unit(0.54, "npc"), y = unit(0.965, "npc"))
grid.text("DESeq2", gp = gpar(font = 2), x = unit(0.12, "npc"), y = unit(0.725, "npc"))
grid.text("ANCOM", gp = gpar(font = 2), x = unit(0.53, "npc"), y = unit(0.735, "npc"))

grid.text("RandomForest", gp = gpar(font = 2), x = unit(0.435, "npc"), y = unit(0.435, "npc"))
grid.text("DESeq2", gp = gpar(font = 2), x = unit(0.86, "npc"), y = unit(0.225, "npc"))
grid.text("ANCOM", gp = gpar(font = 2), x = unit(0.5, "npc"), y = unit(0.255, "npc"))

dev.off()
```

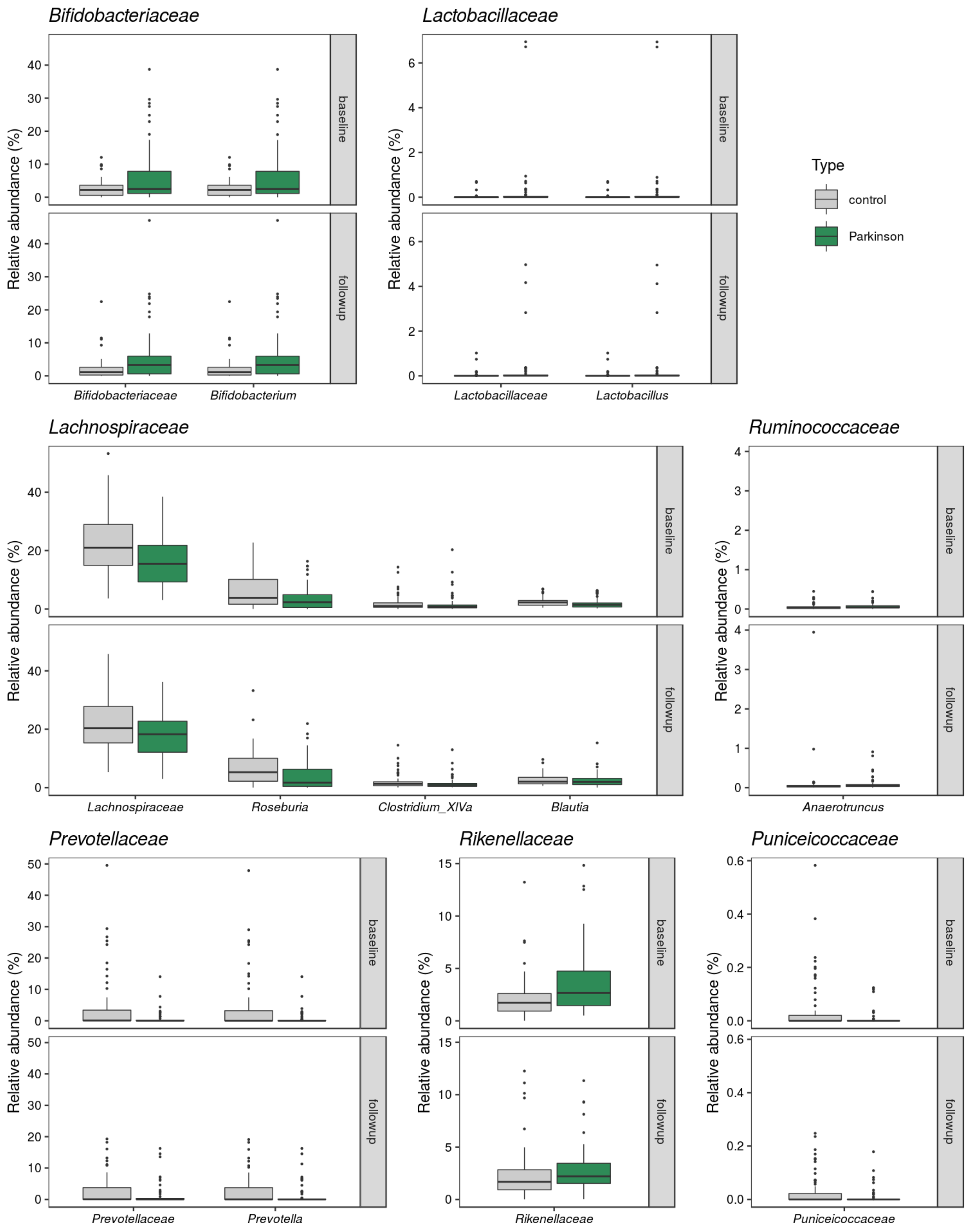
```
## png
##   2
```

```r
# Embed fonts
embedFonts("Outputs/fig9_temp.pdf", outfile = "Outputs/figure9.pdf")
```

Box plots showing the abundances of all the taxa of interest:

Figure 10

```r
colnames(sharedSigAbunds)[grep("Otu", colnames(sharedSigAbunds))] <- gsub(" ", "\n",
    colnames(sharedSigAbunds)[grep("Otu", colnames(sharedSigAbunds))])

# Only for genera and families
sharedSigAbundsGF <- sharedSigAbunds[, -grep("Otu", colnames(sharedSigAbunds))]

# sort taxa per taxonomy?
sharedTaxaPerTax <- sort(colnames(sharedSigAbundsGF[, 3:(ncol(sharedSigAbundsGF))]))

# Plotting function
diffPlot <- function(df){
 ggplot(df, aes(variable, value, fill = Parkinson)) +
  geom_boxplot(outlier.size = 0.1, lwd = 0.25) +
  facet_grid(Timepoint~.) +
  scale_fill_manual(values = c("gray80", "seagreen4"),
                    labels = c("control", "Parkinson"),
                    name = "Type") +
  theme_bw(base_size = 9) +
  xlab(NULL) +
  ylab("\nRelative abundance (%)") +
  theme(legend.key.height = unit(1.5, "line"),
     panel.grid = element_blank(),
     legend.position = "none",
     axis.text = element_text(color = "black"),
     axis.text.x = element_text(face = "italic"),
     plot.title = element_text(face = "italic"))
}

abundPlotsAll <- list()

# Make a table of the taxonomy information
taxForPlot <- rbind(data.frame(Genus = NA,
                               Family = setNames(sharedFam, sharedFam)),
         as.data.frame(tax_table(newPDgen)[sharedGen, c("Genus", "Family")]))

abundPlotsAll[["Bifidobacteriaceae"]] <- diffPlot(
  melt(sharedSigAbunds[, c(rownames(subset(taxForPlot, Family == "Bifidobacteriaceae")),
                           "Parkinson", "Timepoint")])) +
    ggtitle("Bifidobacteriaceae")

abundPlotsAll[["Lachnospiraceae"]] <- diffPlot(
  melt(sharedSigAbunds[, c(rownames(subset(taxForPlot, Family == "Lachnospiraceae")),
                           "Parkinson", "Timepoint")])) +
  ggtitle("Lachnospiraceae")

abundPlotsAll[["Lactobacillaceae"]] <- diffPlot(
  melt(sharedSigAbunds[, c(rownames(subset(taxForPlot, Family == "Lactobacillaceae")),
                           "Parkinson", "Timepoint")])) +
  ggtitle("Lactobacillaceae")

abundPlotsAll[["Prevotellaceae"]] <- diffPlot(
  melt(sharedSigAbunds[, c(rownames(subset(taxForPlot, Family == "Prevotellaceae")),
                           "Parkinson", "Timepoint")])) +
    ggtitle("Prevotellaceae")

abundPlotsAll[["Puniceicoccaceae"]] <- diffPlot(
  melt(sharedSigAbunds[, c(rownames(subset(taxForPlot, Family == "Puniceicoccaceae")),
                           "Parkinson", "Timepoint")])) +
 ggtitle("Puniceicoccaceae")

abundPlotsAll[["Rikenellaceae"]] <- diffPlot(melt(
  sharedSigAbunds[, c(rownames(subset(taxForPlot, Family == "Rikenellaceae")),
                      "Parkinson", "Timepoint")])) +
    ggtitle("Rikenellaceae")

abundPlotsAll[["Ruminococcaceae"]] <- diffPlot(
  melt(sharedSigAbunds[, c(rownames(subset(taxForPlot, Family == "Ruminococcaceae")),
                           "Parkinson", "Timepoint")])) +
    ggtitle("Ruminococcaceae")

apleg <- g_legend(diffPlot(melt(
  sharedSigAbunds[, c(sharedTaxaPerTax[1], "Parkinson", "Timepoint")])) +
    theme(legend.position = "right"))

blank_panel <- grid::rectGrob(gp = grid::gpar(col = "white"))

abundPlotsByTax <- arrangeGrob(
 arrangeGrob(abundPlotsAll[["Bifidobacteriaceae"]],
             abundPlotsAll[["Lactobacillaceae"]], apleg,
             nrow = 1, widths = c(1.85, 1.85, 1.1)),
  arrangeGrob(abundPlotsAll[["Lachnospiraceae"]],
              abundPlotsAll[["Ruminococcaceae"]],
              widths = c(3.15, 1.25)),
 arrangeGrob(abundPlotsAll[["Prevotellaceae"]],
             abundPlotsAll[["Rikenellaceae"]],
             abundPlotsAll[["Puniceicoccaceae"]],
             nrow = 1, widths = c(1.85, 1.3, 1.3)),
 ncol = 1)

grid.arrange(abundPlotsByTax)
```



```
## Export to pdf
ggsave(abundPlotsByTax, filename = "Outputs/figure10.pdf", device = cairo_pdf,
       height = maxhi*0.8, width = fullpage, units = "in")
```

Progression

Data filtering

Starting with the phyloseq object that only has the samples that were selected for the progression comparisons, trim to taxa that are present in around 1/10 samples and drop the “unclassified” bin.

```
round(nrow(sample_data(progPhy)) / 10)
```

```
## [1] 11
```

```
trim_co2 <- round(nrow(sample_data(progPhy)) / 10)

progPhyTrim <- filter_taxa(progPhy, function(x) sum(x > 1) > trim_co2 & sum(x) > 999, prune = TRUE)

progPhyGen <- collapseTaxLevel(progPhy, level = "Genus", fixUnclassifieds = FALSE)
progPhyTrimGen <- filter_taxa(progPhyGen, function(x) sum(x > 1) > trim_co2, prune = TRUE)
progPhyTrimGen <- subset_taxa(progPhyTrimGen, Genus != "unclassified")

progPhyFam <- collapseTaxLevel(progPhy, level = "Family", fixUnclassifieds = FALSE)
progPhyTrimFam <- filter_taxa(progPhyFam, function(x) sum(x > 1) > trim_co2, prune = TRUE)
progPhyTrimFam <- subset_taxa(progPhyTrimFam, Family != "unclassified")
```

ANCOM

Compare timepoints separately, correcting for COMT inhibitor use.

```
# Function for comparisons
ancom2prog <- function(phyloObj, adjp = 2){
  otu_data <- data.frame(Sample.ID = sample_data(phyloObj)$Subject,
      as.data.frame(as.matrix(t(otu_table(phyloObj)))))
  meta_data <- data.frame(Sample.ID = sample_data(phyloObj)$Subject,
                        ProgCat = sample_data(phyloObj)$ProgCat,
                        COMT = sample_data(phyloObj)$meds_COMT_inhibitor)
  res <- ANCOM.main(OTUdat = otu_data,
                  Vardat = meta_data,
                  adjusted = TRUE,
                  repeated = FALSE,
                  main.var = "ProgCat",
                  adj.formula = "COMT",
                  repeat.var = NULL,
                  longitudinal = FALSE,
                  random.formula = NULL,
                  multcorr = adjp,
                  sig = 0.05,
                  prev.cut = 0.9)
  return(res)
}

# Run comparisons

# OTUs
## NB: this step takes a while (but less than an hour)

# Baseline
anProgOTUsBL <- ancom2prog(subset_samples(progPhyTrim, Timepoint == "baseline"))
# Followup
anProgOTUsFU <- ancom2prog(subset_samples(progPhyTrim, Timepoint == "followup"))

# Genera

# Baseline
anProgGenBL <- ancom2prog(subset_samples(progPhyTrimGen, Timepoint == "baseline"))
```

```
#Followup
anProgGenFU <- ancom2prog(subset_samples(progPhyTrimGen, Timepoint == "followup"))

# Families

# Baseline
anProgFamBL <- ancom2prog(subset_samples(progPhyTrimFam, Timepoint == "baseline"))
#Followup
anProgFamFU <- ancom2prog(subset_samples(progPhyTrimFam, Timepoint == "followup"))
```

There are no hits at all on the genus and family levels, only a handful of OTUs for both timepoints.

Collect the results (on the levels that had some):

Table S6A

```
# Function for getting results if there are any (or just an empty data.frame if not):
checkAnRes <- function(ancom_out, lvl, tp){

  ancom_res <- getAnRes(ancom_out)

  if(length(ancom_res) == 0){
    res <- data.frame()
  } else {
    res <- data.frame(Taxon = ancom_res, Level = lvl, Timepoint = tp)
  }
  return(res)
}

anProgRes <- rbind(checkAnRes(anProgOTUsBL, lvl = "OTU", tp = "baseline"),
                   checkAnRes(anProgOTUsFU, lvl = "OTU", tp = "followup"),
                   checkAnRes(anProgGenBL, lvl = "genus", tp = "baseline"),
                   checkAnRes(anProgGenFU, lvl = "genus", tp = "followup"),
                   checkAnRes(anProgFamBL, lvl = "family", tp = "baseline"),
                   checkAnRes(anProgFamFU, lvl = "family", tp = "followup"))

# Table for exporting, with genus classification information for OTUs
anProgExp <- anProgRes
anProgExp$Taxon <- as.character(anProgExp$Taxon)
anProgExp[anProgExp$Level == "OTU", "Taxon"] <- gsub("_", " ", paste(anProgExp[anProgExp$Level ==
    "OTU", "Taxon"], " (", tax_table(progPhyTrim)[anProgExp[anProgExp$Level == "OTU", "Taxon"],
    "Genus"], ")", sep = ""))
```

```
kable_styling(kable(anProgExp), full_width = FALSE)
```

Taxon	Level	Timepoint
Otu0148 (Bifidobacterium)	OTU	followup
Otu0327 (Lachnospiraceae unclassified)	OTU	followup

```
## Export
write.csv(anProgExp, "Outputs/table_s6a.csv")
```

Random forests

Run separately for baseline and follow-up, not corrected for confounders.

```
# Function for random forest classifiers
rfProgCat <- function(phyloObj, rngseed, treen){
    predictors <- t(otu_table(phyloObj))
    response <- sample_data(phyloObj)$ProgCat
    rf.data <- data.frame(response, predictors)
    set.seed(rngseed)
    rfres <- rfPermute(response~., data = rf.data, ntree = treen, num.cores = 4)
    return(rfres)
}

## Run comparisons:

## OTUs
rfOtusProgBL <- rfProgCat(subset_samples(progPhyTrim, Timepoint == "baseline"), 483029, 500)
rfSigOtusProgBL <- getRFsigs(rfOtusProgBL, "OTU")
rfSigOtusProgBL$Timepoint <- "baseline"

rfOtusProgFU <- rfProgCat(subset_samples(progPhyTrim, Timepoint == "followup"), 132118, 500)
rfSigOtusProgFU <- getRFsigs(rfOtusProgFU, "OTU")
rfSigOtusProgFU$Timepoint <- "followup"

# Genera
rfGenProgBL <- rfProgCat(subset_samples(progPhyTrimGen, Timepoint == "baseline"), 900097, 500)
rfSigGenProgBL <- getRFsigs(rfGenProgBL, "Genus")
rfSigGenProgBL$Timepoint <- "baseline"

rfGenProgFU <- rfProgCat(subset_samples(progPhyTrimGen, Timepoint == "followup"), 562913, 500)
rfSigGenProgFU <- getRFsigs(rfGenProgFU, "Genus")
rfSigGenProgFU$Timepoint <- "followup"

# Families
rfFamProgBL <- rfProgCat(subset_samples(progPhyTrimFam, Timepoint == "baseline"), 553580, 500)
rfSigFamProgBL <- getRFsigs(rfFamProgBL, "Family")
rfSigFamProgBL$Timepoint <- "baseline"

rfFamProgFU <- rfProgCat(subset_samples(progPhyTrimFam, Timepoint == "followup"), 526037, 500)
rfSigFamProgFU <- getRFsigs(rfFamProgFU, "Family")
rfSigFamProgFU$Timepoint <- "followup"
```

Table S6B

Collect results and export (here showing only first 10 rows) :

```
rfProgRes <- rbind(rfSigOtusProgBL, rfSigOtusProgFU, rfSigGenProgBL,
                   rfSigGenProgFU, rfSigFamProgBL, rfSigFamProgFU)

# Trim to only significant results and reorganize for export
rfProgResultsExp <- subset(rfProgRes, MeanDecreaseGini.pval < 0.05)
rfProgResultsExp <- rfProgResultsExp[, c("Level", "Timepoint", "Taxon", "MeanDecreaseGini",
    "MeanDecreaseGini.pval")]
rfProgResultsExp <- rfProgResultsExp[order(rfProgResultsExp$Level, rfProgResultsExp$Timepoint,
    rfProgResultsExp$Taxon),]
rfProgResultsExp[rfProgResultsExp$Level == "OTU", "Taxon"] <- gsub("_", " ",
    paste(rfProgResultsExp[rfProgResultsExp$Level == "OTU", "Taxon"], " (",
    tax_table(progPhyTrim)[rfProgResultsExp[rfProgResultsExp$Level == "OTU", "Taxon"], "Genus"],
    ")", sep = ""))
rownames(rfProgResultsExp) <- NULL
```

```
kable_styling(kable(head(rfProgResultsExp, 10), digits = 4), font_size = 10, full_width = FALSE)
```

Level	Timepoint	Taxon	MeanDecreaseGini	MeanDecreaseGini.pval
Family	baseline	Eubacteriaceae	0.9975	0.0297
Family	baseline	Streptococcaceae	1.7012	0.0099
Family	baseline	Synergistaceae	0.8288	0.0495
Family	followup	Actinomycetaceae	1.3350	0.0297
Family	followup	Anaeroplasmataceae	0.9584	0.0099
Genus	baseline	Anaerotruncus	0.7630	0.0396
Genus	baseline	Cloacibacillus	0.3297	0.0396
Genus	baseline	Eubacterium	0.6376	0.0099
Genus	baseline	Intestinimonas	0.7467	0.0297
Genus	followup	Actinomyces	0.6551	0.0396

```r
## Export
write.csv(rfProgResultsExp, "Outputs/table_s6b.csv")

# Reorder for downstream comparisons:
rfProgRes <- rfProgRes[, c("Taxon", "Level", "Timepoint", "MeanDecreaseGini.pval")]
colnames(rfProgRes)[4] <- "pval"
```

Estimate model significance:

```r
# Estimating model significance
# (This is SLOOOW)
rfModSigsProg <- rbind(
    getRFmodelSig(rfOtusProgBL, subset_samples(progPhyTrim, Timepoint == "baseline")),
    getRFmodelSig(rfOtusProgFU, subset_samples(progPhyTrim, Timepoint == "followup")),
    getRFmodelSig(rfGenProgBL, subset_samples(progPhyTrimGen, Timepoint == "baseline")),
    getRFmodelSig(rfGenProgFU, subset_samples(progPhyTrimGen, Timepoint == "followup")),
    getRFmodelSig(rfFamProgBL, subset_samples(progPhyTrimFam, Timepoint == "baseline")),
    getRFmodelSig(rfFamProgFU, subset_samples(progPhyTrimFam, Timepoint == "followup"))
    )
```

Table of the model significance:

Table S7

```r
colnames(rfModSigsProg) <- c("pValue", "realOOB", "permOOB")
rfModSigsProg <- as.data.frame(rfModSigsProg)
rfModSigsProg$Level <- rep(c("OTU", "Genus", "Family"), each = 2)
rfModSigsProg$Timepoint <- rep(c("baseline", "followup"), 3)

rfModSigsProg <- rfModSigsProg[, c("Level", "Timepoint", "realOOB", "permOOB", "pValue")]
rfModSigsProg <- rfModSigsProg[order(rfModSigsProg$Level), ]
rownames(rfModSigsProg) <- NULL

kable_styling(kable(rfModSigsProg, digits = 3), full_width = FALSE)
```

Level	Timepoint	realOOB	permOOB	pValue
Family	baseline	0.286	0.286	0.232
Family	followup	0.321	0.286	0.798
Genus	baseline	0.304	0.286	0.660
Genus	followup	0.321	0.286	0.910
OTU	baseline	0.304	0.286	0.728
OTU	followup	0.268	0.286	0.044

```r
## Export
write.csv(rfModSigsProg, "Outputs/table_s7.csv")
```

None of the models are better than chance at classifying the samples into progressed and stable.

DESeq2

Timepoints run separately, correcting for COMT inhibitor medication.

```r
library("DESeq2")

# Function for comparisons
dsProgTpCat <- function(phyloObj, baseline = TRUE){
    if(baseline == FALSE){
        phO <- subset_samples(phyloObj, Timepoint == "followup")
    } else {
        phO <- subset_samples(phyloObj, Timepoint == "baseline")
    }
    ds <- phyloseq_to_deseq2(phO, ~ meds_COMT_inhibitor + ProgCat)
    ds <- DESeq(ds, fitType = "parametric", sfType = "poscounts")
    dsres <- rbind(
      data.frame(cbind(as(results(ds, cooksCutoff = FALSE, name = "meds_COMT_inhibitor_1_vs_0"),
          "data.frame"), as(tax_table(ph0), "matrix")), Variable = "meds_COMT_inhibitor"),
      data.frame(cbind(as(results(ds, cooksCutoff = FALSE, name = "ProgCat_progressed_vs_stable"),
          "data.frame"), as(tax_table(ph0), "matrix")), Variable = "ProgCat_progressed_vs_stable"))
    if(baseline == FALSE){
        dsres$Timepoint <- "followup"
    } else {
        dsres$Timepoint <- "baseline"
    }
    return(dsres)
}

# Run comparisons

# OTUs
dsProgOtusBL <- dsProgTpCat(phyloObj = progPhyTrim, baseline = TRUE)
dsProgOtusFU <- dsProgTpCat(phyloObj = progPhyTrim, baseline = FALSE)

# Genera
dsProgGenBL <- dsProgTpCat(phyloObj = progPhyTrimGen, baseline = TRUE)
dsProgGenFU <- dsProgTpCat(phyloObj = progPhyTrimGen, baseline = FALSE)

# Families
dsProgFamBL <- dsProgTpCat(phyloObj = progPhyTrimFam, baseline = TRUE)
dsProgFamFU <- dsProgTpCat(phyloObj = progPhyTrimFam, baseline = FALSE)

# Collect results:
dsProgRes <- data.frame(rbind(dsProgOtusBL, dsProgOtusFU), Level = "OTU")
dsProgRes <- rbind(dsProgRes, data.frame(rbind(dsProgGenBL, dsProgGenFU), Level = "Genus"))
dsProgRes <- rbind(dsProgRes, data.frame(rbind(dsProgFamBL, dsProgFamFU), Genus = NA, Level =
    "Family"))
```

Table S6C

Rearrange results data frame; export (here only showing first 10 rows):

```
# Reorganize table to have a single "Taxon" variable instead of full taxonomy:
dsProgRes[dsProgRes$Level == "Family", "Taxon"] <- as.character(dsProgRes[dsProgRes$Level ==
    "Family", "Family"])
dsProgRes[dsProgRes$Level == "Genus", "Taxon"] <- as.character(dsProgRes[dsProgRes$Level == "Genus",
    "Genus"])
dsProgRes[dsProgRes$Level == "OTU", "Taxon"] <- substr(rownames(dsProgRes[dsProgRes$Level ==
    "OTU",]), start = 1, stop = 7)
dsProgRes <- dsProgRes[, c("Level", "Timepoint", "Taxon", "Variable", "log2FoldChange", "lfcSE",
    "pvalue", "padj")]

# Results for exporting final table
dsProgExp <- subset(dsProgRes, padj < 0.05)
levels(dsProgExp$Variable) <- c("COMT inhibitor", "Progression")
dsProgExp$Variable <- factor(dsProgExp$Variable, levels = c("Progression", "COMT inhibitor"))
dsProgExp$Level <- factor(dsProgExp$Level, levels = c("Family", "Genus", "OTU"))
dsProgExp <- dsProgExp[order(dsProgExp$Level,  dsProgExp$Timepoint, dsProgExp$Variable,
    dsProgExp$Taxon),]
rownames(dsProgExp) <- NULL
```

```
kable_styling(kable(head(dsProgExp, 10), digits = 4), font_size = 10, full_width = FALSE)
```

Level	Timepoint	Taxon	Variable	log2FoldChange	lfcSE	pvalue	padj
Family	baseline	Streptococcaceae	Progression	2.2304	0.5388	0.0000	0.0013
Family	baseline	Enterococcaceae	COMT inhibitor	14.6724	4.6902	0.0018	0.0351
Family	baseline	Peptostreptococcaceae	COMT inhibitor	-2.9684	0.9556	0.0019	0.0351
Family	followup	Anaeroplasmataceae	Progression	7.5962	1.3797	0.0000	0.0000

Level	Timepoint	Taxon	Variable	log2FoldChange	lfcSE	pvalue	padj
Family	followup	Oxalobacteraceae	Progression	-2.7586	0.9027	0.0022	0.0277
Family	followup	Prevotellaceae	Progression	-4.8780	1.0187	0.0000	0.0000
Family	followup	Verrucomicrobiaceae	Progression	-2.2722	0.7763	0.0034	0.0316
Family	followup	Lachnospiraceae	COMT inhibitor	-1.5118	0.3315	0.0000	0.0001
Family	followup	Lactobacillaceae	COMT inhibitor	6.2970	0.9224	0.0000	0.0000
Family	followup	Ruminococcaceae	COMT inhibitor	-1.0925	0.2406	0.0000	0.0001

```
## Export
write.csv(dsProgExp, "Outputs/table_s6c.csv")

# Trim results for downstream analyses
dsProgRes <- dsProgRes[, c("Taxon", "Level", "Timepoint", "padj", "Variable")]
```

Compare and contrast results

Find out if there are any overlapping taxa between methods and/or timepoints for the progression variable:

```
source("Inputs/taxaSummarizer.R") # function for making summaries of relative abundances

# Additional fixes to the DS2 results:
dsProgRes <- subset(dsProgRes, Variable == "ProgCat_progressed_vs_stable")
dsProgRes$Variable <- NULL
colnames(dsProgRes)[4] <- "pval"

# Combine results from the three methods:
progResAll <- rbind(data.frame(anProgRes, pval = 0, Method = "ANCOM"),
                    data.frame(rfProgRes, Method = "RF"),
                    data.frame(dsProgRes, Method = "DS2"))
rownames(progResAll) <- NULL

# Dataframe with only significant taxa:
progResAllSigs <- subset(progResAll, pval < 0.05)

## Are there any overlapping taxa between methods?

# Baseline
blProgShared <- names(which(table(c(
  as.character(subset(progResAllSigs, Method == "ANCOM" & Timepoint == "baseline")$Taxon),
    as.character(subset(progResAllSigs, Method == "RF" & Timepoint == "baseline")$Taxon),
    as.character(subset(progResAllSigs, Method == "DS2" & Timepoint == "baseline")$Taxon))) > 1))
blProgShared
```

```
## [1] "Otu0111"          "Streptococcaceae"
```

```
# Followup
fuProgShared <- names(which(table(c(
  as.character(subset(progResAllSigs, Method == "ANCOM" & Timepoint == "followup")$Taxon),
    as.character(subset(progResAllSigs, Method == "RF" & Timepoint == "followup")$Taxon),
    as.character(subset(progResAllSigs, Method == "DS2" & Timepoint == "followup")$Taxon))) > 1))
fuProgShared
```

```
## [1] "Anaeroplasmataceae" "Asteroleplasma"     "Otu0084"
## [4] "Otu0115"            "Otu0148"            "Otu0241"
## [7] "Otu0327"
```

```
# Any of these also shared between timepoints?
intersect(blProgShared, fuProgShared)
```

```
## character(0)
```

```r
# No!

# Are there overall any significant taxa shared between timepoints?
# (detected as significant with at least one method)
tpProgShared <- names(which(table(c(
  unique(as.character(subset(progResAllSigs, Timepoint == "baseline")$Taxon)),
  unique(as.character(subset(progResAllSigs, Timepoint == "followup")$Taxon)))) > 1))
tpProgShared
```

```
## [1] "Otu0042"    "Otu0049"    "Otu0115"    "Otu0118"    "Otu0166"
## [6] "Otu0222"    "Otu0268"    "Otu0327"    "Prevotella"
```

```r
progSharedTaxa <- sort(unique(c(blProgShared, fuProgShared, tpProgShared)))
```

Make a table of the taxa that are significant at either timepoint according to more than one tool, or at both timepoints according to at least one tool.

```r
# Collect the overlapping taxa into a dataframe
progSharedSigs <- subset(progResAll, progResAll$Taxon %in% progSharedTaxa)
progSharedSigs <- dcast(progSharedSigs, Taxon + Level ~  Timepoint + Method, value.var = "pval")

progSharedSigs[!is.na(progSharedSigs$baseline_ANCOM), "baseline_ANCOM"] <- "significant"
progSharedSigs[!is.na(progSharedSigs$followup_ANCOM), "followup_ANCOM"] <- "significant"
progSharedSigs[is.na(progSharedSigs$baseline_ANCOM), "baseline_ANCOM"] <- "n.s."
progSharedSigs[is.na(progSharedSigs$followup_ANCOM), "followup_ANCOM"] <- "n.s."

progSharedSigs$Level <- factor(progSharedSigs$Level, levels = c("Family", "Genus", "OTU"))
progSharedSigs <- progSharedSigs[order(progSharedSigs$Level, progSharedSigs$Taxon),]

# Add mean relative abundances
progSharedSigs$Taxon <- as.character(progSharedSigs$Taxon)

progSharedSigsExp <- cbind(progSharedSigs, rbind(
  groupMeanSD(progPhyFam, subset(progSharedSigs, Level == "Family")$Taxon, "ProgTP"),
    groupMeanSD(progPhyGen, subset(progSharedSigs, Level == "Genus")$Taxon, "ProgTP"),
    groupMeanSD(progPhy, subset(progSharedSigs, Level == "OTU")$Taxon, "ProgTP")))

# Delete redundant second Taxon column
progSharedSigsExp[, grep("Taxon", colnames(progSharedSigsExp))[2]] <- NULL

# Mark taxa with overlap between methods
progSharedSigsExp$MethodOverlap <- progSharedSigsExp$Taxon %in% unique(c(blProgShared, fuProgShared))

# Mark taxa with overlap between timepoints
progSharedSigsExp$TimepointOverlap <- progSharedSigsExp$Taxon %in% tpProgShared

# Add genus classifications for OTUs
progSharedSigsExp[progSharedSigsExp$Level == "OTU", "Taxon"] <- gsub("_", " ",
    paste(progSharedSigsExp[progSharedSigsExp$Level == "OTU", "Taxon"],
    tax_table(progPhy)[progSharedSigsExp[progSharedSigsExp$Level == "OTU", "Taxon"], "Genus"]))

# Reorder
progSharedSigsExp <- progSharedSigsExp[, c("Taxon", "Level", "MethodOverlap", "TimepointOverlap",
    "MeanSD stable_baseline", "MeanSD progressed_baseline", "baseline_ANCOM", "baseline_RF",
    "baseline_DS2", "MeanSD stable_followup", "MeanSD progressed_followup", "followup_ANCOM",
    "followup_RF", "followup_DS2")]
rownames(progSharedSigsExp) <- progSharedSigsExp$Taxon
progSharedSigsExp$Taxon <- NULL
```

Table 9

Export table:

```r
# Baseline results
kable(progSharedSigsExp[, 1:8], digits = 4, col.names = c("Level", "Method Overlap", "Timepoint
    Overlap", "MeanSD stable baseline", "MeanSD progressed baseline", "p_ANCOM baseline", "p_RF
    baseline", "p_DS2 baseline"))
```

	Level	Method Overlap	Timepoint Overlap	MeanSD stable baseline	MeanSD progressed baseline	p_-ANCOM baseline	p_RF baseline	p_DS2 baseline
Streptococcaceae	Family	TRUE	FALSE	0.207 ± 0.584	0.614 ± 1.316	n.s.	0.0099	0.0013
Anaeroplasmataceae	Family	TRUE	FALSE	0.426 ± 1.792	0.371 ± 1.125	n.s.	0.1980	0.9661
Prevotella	Genus	FALSE	TRUE	0.964 ± 2.571	0.147 ± 0.493	n.s.	0.5545	0.0000
Asteroleplasma	Genus	TRUE	FALSE	0.191 ± 1.022	0.302 ± 1.113	n.s.	0.4554	0.7673
Otu0148 Bifidobacterium	OTU	TRUE	FALSE	0.06 ± 0.252	0.143 ± 0.498	n.s.	0.4653	0.8621
Otu0327 Lachnospiraceae unclassified	OTU	TRUE	TRUE	0.013 ± 0.032	0.104 ± 0.282	n.s.	0.0495	0.9651
Otu0118 Ruminococcaceae unclassified	OTU	FALSE	TRUE	0.084 ± 0.102	0.246 ± 0.221	n.s.	0.0099	0.0944
Otu0166 Ruminococcaceae unclassified	OTU	FALSE	TRUE	0.069 ± 0.102	0.166 ± 0.203	n.s.	0.0198	0.5306
Otu0222 Phascolarctobacterium	OTU	FALSE	TRUE	0.091 ± 0.281	0 ± 0	n.s.	0.4356	0.0018
Otu0111 Streptococcus	OTU	TRUE	FALSE	0.127 ± 0.369	0.46 ± 1.187	n.s.	0.0495	0.0069
Otu0042 Coprococcus	OTU	FALSE	TRUE	0.344 ± 1.26	0.002 ± 0.002	n.s.	0.7327	0.0003
Otu0268 Desulfovibrio	OTU	FALSE	TRUE	0.059 ± 0.168	0.034 ± 0.131	n.s.	0.4752	0.0000
Otu0115 Lachnospiraceae unclassified	OTU	TRUE	TRUE	0.167 ± 0.404	0.607 ± 2.296	n.s.	0.9208	0.0185
Otu0049 Ruminococcaceae unclassified	OTU	FALSE	TRUE	0.26 ± 0.707	0.595 ± 0.919	n.s.	0.0396	0.8639
Otu0241 Clostridiales unclassified	OTU	TRUE	FALSE	0.029 ± 0.054	0.044 ± 0.064	n.s.	0.4455	0.6979
Otu0084 Clostridium IV	OTU	TRUE	FALSE	0.313 ± 0.978	0.335 ± 0.726	n.s.	0.4851	0.8639

```r
# Follow-up results
kable(progSharedSigsExp[, 9:ncol(progSharedSigsExp)], digits = 4, col.names = c("MeanSD stable
    followup", "MeanSD progressed followup", "p_ANCOM followup", "p_RF followup", "p_DS2 followup"))
```

	MeanSD stable followup	MeanSD progressed followup	p_ANCOM followup	p_RF followup	p_DS2 followup
Streptococcaceae	0.299 ± 0.904	0.398 ± 0.74	n.s.	0.6634	0.1764
Anaeroplasmataceae	0.133 ± 0.596	0.311 ± 1.05	n.s.	0.0099	0.0000

	MeanSD stable followup	MeanSD progressed followup	p__ANCOM followup	p__RF followup	p__DS2 followup
Prevotella	1.966 ± 4.04	0.187 ± 0.678	n.s.	0.3861	0.0000
Asteroleplasma	0.133 ± 0.596	0.31 ± 1.051	n.s.	0.0495	0.0000
Otu0148 Bifidobacterium	0.051 ± 0.125	0.744 ± 1.324	significant	0.0099	0.3298
Otu0327 Lachnospiraceae unclassified	0.021 ± 0.091	0.178 ± 0.357	significant	0.0099	0.0157
Otu0118 Ruminococcaceae unclassified	0.135 ± 0.261	0.165 ± 0.14	n.s.	0.0396	0.8383
Otu0166 Ruminococcaceae unclassified	0.082 ± 0.097	0.194 ± 0.227	n.s.	0.0396	0.6027
Otu0222 Phasco-larctobacterium	0.043 ± 0.137	0 ± 0	n.s.	0.9109	0.0000
Otu0111 Streptococcus	0.217 ± 0.643	0.254 ± 0.501	n.s.	0.7426	0.4380
Otu0042 Coprococcus	0.511 ± 1.44	0.004 ± 0.011	n.s.	0.5248	0.0000
Otu0268 Desulfovibrio	0.042 ± 0.102	0.001 ± 0.002	n.s.	0.7921	0.0093
Otu0115 Lachnospiraceae unclassified	0.04 ± 0.086	0.776 ± 2.121	n.s.	0.0099	0.0143
Otu0049 Ruminococcaceae unclassified	0.286 ± 0.663	0.56 ± 0.896	n.s.	0.0297	0.2972
Otu0241 Clostridiales unclassified	0.015 ± 0.022	0.084 ± 0.159	n.s.	0.0396	0.0200
Otu0084 Clostridium IV	0.071 ± 0.091	0.275 ± 0.345	n.s.	0.0099	0.0006

```r
## Export
write.csv(progSharedSigsExp, "Outputs/table_9.csv")
```

Make plots of all overlapping genera and families (either method or timepoint overlap), and the family *Prevotellaceae*, which is a taxon of interest and was significant with DESeq2 at follow-up (but not at baseline).

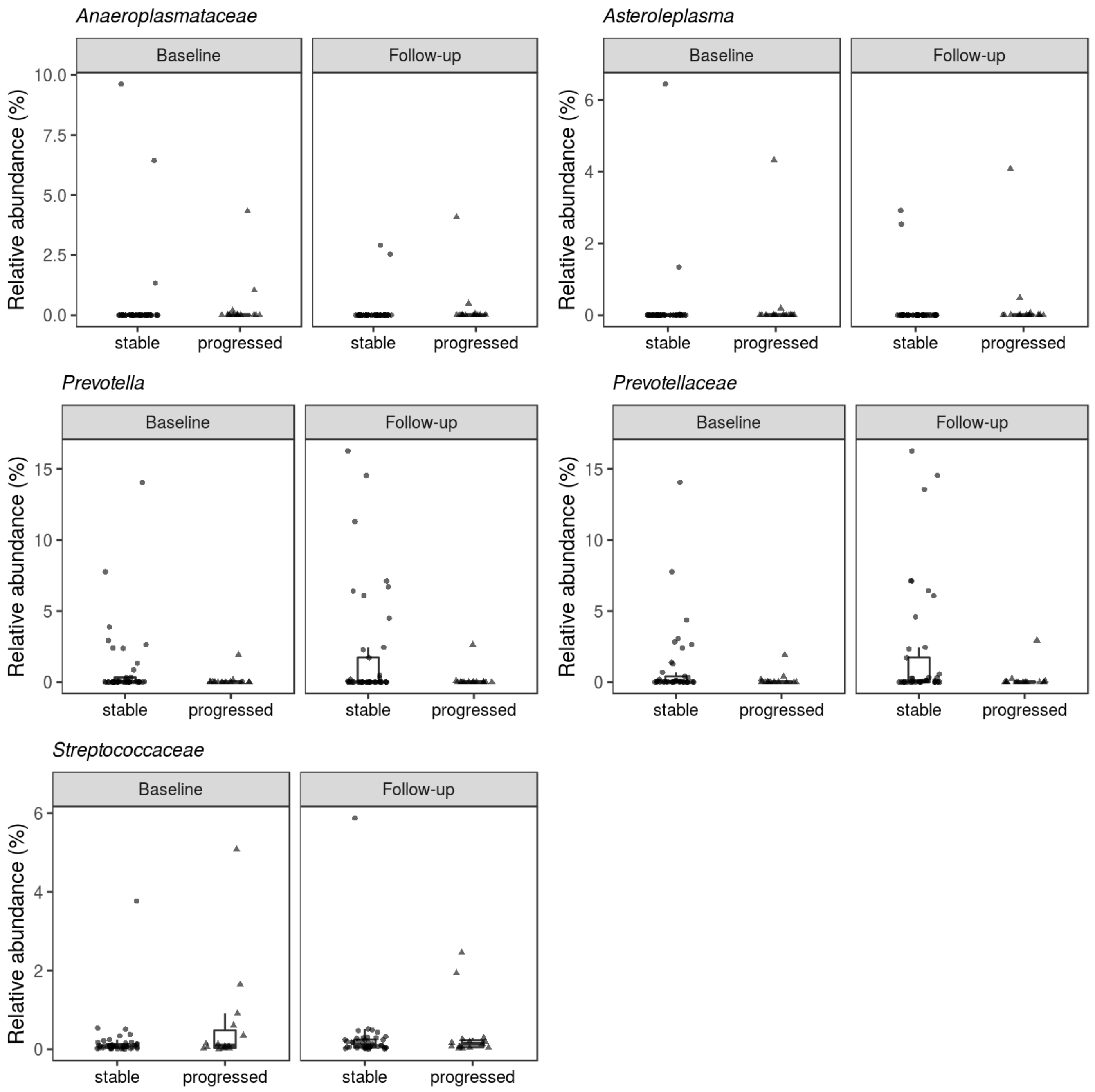
Figure 11

```r
# Plotting function
progDiffBoxPlot <- function(df, taxon){
    p <- ggplot(df, aes(y = df[[taxon]], x = ProgCat, shape = ProgCat)) +
        geom_boxplot(width = 0.2, outlier.shape = NA) +
        geom_jitter(width = 0.2, alpha = 0.6, size = 1) +
        theme_bw() +
    xlab(NULL) +
        ggtitle(taxon) +
    facet_grid(~Timepoint) +
        scale_fill_manual(values = c("gray95", "gray10")) +
        scale_x_discrete(labels = c("stable", "progressed")) +
        ylab("Relative abundance (%)") +
        theme(legend.position = "none",
              panel.grid = element_blank(),
              axis.text.x = element_text(color = "black"),
              plot.title = element_text(size = 10, face = "italic"))
  return(p)
}

progSharedGen <- progSharedTaxa[-c(grep("Otu", progSharedTaxa), grep("ceae", progSharedTaxa))]

progSharedFam <- c(grep("ceae", progSharedTaxa[-grep("Otu", progSharedTaxa)], value = TRUE),
    "Prevotellaceae")

# Make table with relative abundances
progAbunds <- data.frame(row.names = rownames(sample_data(progPhy)))
progAbunds[, progSharedGen] <- t(prop.table(otu_table(progPhyGen), 2) * 100)[, progSharedGen]
progAbunds[, progSharedFam] <- t(prop.table(otu_table(progPhyFam), 2) * 100)[, progSharedFam]
```

```
# Add metadata
progAbunds$ProgCat <- sample_data(progPhy)$ProgCat
progAbunds$Timepoint <- sample_data(progPhy)$Timepoint
levels(progAbunds$Timepoint) <- c("Baseline", "Follow-up")

progSharedSort <- sort(c(progSharedGen, progSharedFam))
progSharedSort
```

```
## [1] "Anaeroplasmataceae" "Asteroleplasma"      "Prevotella"
## [4] "Prevotellaceae"     "Streptococcaceae"
```

```
# Plot taxa
progPlots <- lapply(progSharedSort, function(x) progDiffBoxPlot(progAbunds, x))
progPlotsGrobs <- do.call("arrangeGrob", progPlots)

grid.arrange(progPlotsGrobs)
```



```
## Export to pdf
ggsave(progPlotsGrobs, filename = "Outputs/figure11.pdf", device = cairo_pdf,
       width = fullpage, height = maxhi*0.7, units = "in")
```

PD phenotypes (TD vs PIGD)

Data setup

Trim to taxa present in at least 1/10 samples, and remove the "unclassified" bin.

```
nrow(sample_data(progPhyPhe)) / 10 # 10.4
```

```
## [1] 10.4
```

```
trim_co3 <- round(nrow(sample_data(progPhyPhe)) / 10)

progPhyPheO <- filter_taxa(progPhyPhe, function(x) sum(x > 1) > trim_co3 & sum(x) > 999, prune =
    TRUE)

progPhyPheG <- filter_taxa(collapseTaxLevel(progPhyPhe, level = "Genus", fixUnclassifieds = FALSE),
    function(x) sum(x > 1) > trim_co3, prune = TRUE)
progPhyPheG <- subset_taxa(progPhyPheG, Genus != "unclassified")

progPhyPheF <- filter_taxa(collapseTaxLevel(progPhyPhe, level = "Family", fixUnclassifieds = FALSE),
    function(x) sum(x > 1) > trim_co3, prune = TRUE)
progPhyPheF <- subset_taxa(progPhyPheF, Family != "unclassified")
```

ANCOM

Compare TD vs PIGD with ANCOM (not correcting for any confounders):

```
# Function for comparisons:
ancom2JanCat <- function(phyloObj, adjp = 2){
  otu_data <- data.frame(Sample.ID = sample_data(phyloObj)$Subject,
      as.data.frame(as.matrix(t(otu_table(phyloObj)))))
  meta_data <- data.frame(Sample.ID = sample_data(phyloObj)$Subject,
                          JankovicClass = sample_data(phyloObj)$JankovicClass)
  res <- ANCOM.main(OTUdat = otu_data,
                    Vardat = meta_data,
                    adjusted = FALSE,
                    repeated = FALSE,
                    main.var = "JankovicClass",
                    adj.formula = NULL,
                    repeat.var = NULL,
                    longitudinal = FALSE,
                    random.formula = NULL,
                    multcorr = adjp,
                    sig = 0.05,
                    prev.cut = 0.90)
  return(res)
}

# OTUs
# NB: Takes some time to finish!

anPheOtuBL<-ancom2JanCat(subset_samples(progPhyPheO, Timepoint == "baseline"))
anPheOtuFU<-ancom2JanCat(subset_samples(progPhyPheO, Timepoint == "followup"))

# Genera
anPheGenBL <- ancom2JanCat(subset_samples(progPhyPheG, Timepoint == "baseline"))
anPheGenFU <- ancom2JanCat(subset_samples(progPhyPheG, Timepoint == "followup"))

# Families
anPheFamBL <- ancom2JanCat(subset_samples(progPhyPheF, Timepoint == "baseline"))
anPheFamFU <- ancom2JanCat(subset_samples(progPhyPheF, Timepoint == "followup"))
```

Collect the results, which were exported as Supplementary table 3A.

Table S8A

```
# Any hits found?
anJanOTUs <- rbind(checkAnRes(anPheOtuBL, lvl = "OTU", tp = "baseline"),
                   checkAnRes(anPheOtuFU, lvl = "OTU", tp = "followup"),
                   checkAnRes(anPheGenBL, lvl = "genus", tp = "baseline"),
                   checkAnRes(anPheGenFU, lvl = "genus", tp = "followup"),
                   checkAnRes(anPheFamBL, lvl = "family", tp = "baseline"),
                   checkAnRes(anPheFamFU, lvl = "family", tp = "followup"))
```

```
kable_styling(kable(anJanOTUs), full_width = FALSE)
```

Taxon	Level	Timepoint
Otu0048	OTU	baseline
Otu0170	OTU	baseline

```
## Export
write.csv(anJanOTUs, "Outputs/table_s8a.csv")
```

Only two OTUs for baseline, nothing for follow-up. No genera or families at either level.

DESeq2

```
# Testing function
dsJanTpCat <- function(phyloObj, baseline = TRUE){
  if(baseline == FALSE){
    phO <- subset_samples(phyloObj, Timepoint == "followup")
  } else {
    phO <- subset_samples(phyloObj, Timepoint == "baseline")
  }
  ds <- phyloseq_to_deseq2(phO, ~ JankovicClass)
  ds <- DESeq(ds, fitType = "parametric", sfType = "poscounts")
  dsres<-cbind(as.data.frame(results(ds, cooksCutoff = FALSE, name = "JankovicClass_TD_vs_PIGD")),
      as.matrix(tax_table(phO)))
  if(baseline == FALSE){
    dsres$Timepoint <- "followup"
  } else {
    dsres$Timepoint <- "baseline"
  }
  return(dsres)
}

# Run comparisons

# OTUs
dsJanOtusBL <- dsJanTpCat(phyloObj=progPhyPheO, baseline = TRUE)
dsJanOtusFU <- dsJanTpCat(phyloObj=progPhyPheO, baseline = FALSE)

# Genera
dsJanGenBL <- dsJanTpCat(phyloObj = progPhyPheG, baseline = TRUE)
dsJanGenFU <- dsJanTpCat(phyloObj = progPhyPheG, baseline = FALSE)

# Families
dsJanFamBL <- dsJanTpCat(phyloObj = progPhyPheF, baseline = TRUE)
dsJanFamFU <- dsJanTpCat(phyloObj = progPhyPheF, baseline = FALSE)
```

Table S8B

Collect the results and export (here showing only first 10 rows).

```r
# Collect results
dsJanRes <- rbind(data.frame(rbind(dsJanOtusBL, dsJanOtusFU), Level = "OTU"),
                  data.frame(rbind(dsJanGenBL, dsJanGenFU), Level = "Genus"),
                  data.frame(rbind(dsJanFamBL, dsJanFamFU), Genus = NA, Level = "Family"))

dsJanResSigs <- subset(dsJanRes, padj < 0.05)

dsJanResSigs[dsJanResSigs$Level == "Family", "Taxon"] <-
    as.character(dsJanResSigs[dsJanResSigs$Level == "Family", "Family"])
dsJanResSigs[dsJanResSigs$Level == "Genus", "Taxon"] <- as.character(dsJanResSigs[dsJanResSigs$Level
    == "Genus", "Genus"])
dsJanResSigs[dsJanResSigs$Level == "OTU", "Taxon"] <-
    substr(rownames(dsJanResSigs[dsJanResSigs$Level == "OTU",]), start = 1, stop = 7)
dsJanResSigs[dsJanResSigs$Level == "OTU", "Taxon"] <- paste(dsJanResSigs[dsJanResSigs$Level ==
    "OTU", "Taxon"], " (", gsub("_", " ", tax_table(progPhyPhe)[dsJanResSigs[dsJanResSigs$Level ==
    "OTU", "Taxon"], "Genus"]), ")", sep ="")

dsJanResSigs <- dsJanResSigs[, c("Level", "Timepoint", "Taxon", "log2FoldChange", "lfcSE", "pvalue",
    "padj")]
```

```r
kable_styling(kable(head(dsJanResSigs, 10), digits = 4), font_size = 10, full_width = FALSE)
```

	Level	Timepoint	Taxon	log2FoldChange	lfcSE	pvalue	padj
Otu0012	OTU	baseline	Otu0012 (Bacteroides)	1.8238	0.5327	0.0006	0.0246
Otu0019	OTU	baseline	Otu0019 (Prevotella)	-3.0993	0.8683	0.0004	0.0160
Otu0031	OTU	baseline	Otu0031 (Escherichia/Shigella)	-2.6344	0.7969	0.0009	0.0273
Otu0047	OTU	baseline	Otu0047 (Butyrivibrio)	4.5339	1.0628	0.0000	0.0024
Otu0063	OTU	baseline	Otu0063 (Acidaminococcaceae unclassified)	6.4045	1.7359	0.0002	0.0134
Otu0082	OTU	baseline	Otu0082 (Asteroleplasma)	-5.5362	1.6691	0.0009	0.0273
Otu0093	OTU	baseline	Otu0093 (Clostridiales unclassified)	6.3962	1.2701	0.0000	0.0001
Otu0115	OTU	baseline	Otu0115 (Lachnospiraceae unclassified)	5.4087	0.9346	0.0000	0.0000
Otu0178	OTU	baseline	Otu0178 (Bacteria unclassified)	7.0462	1.8853	0.0002	0.0133
Otu0197	OTU	baseline	Otu0197 (Deltaproteobacteria unclassified)	-3.6637	1.1336	0.0012	0.0314

```r
## Export
write.csv(dsJanResSigs, "Outputs/table_s8b.csv")
```

Session info

The R packages and their specific versions used for these analyses were the following:

```r
devtools::session_info()
```

```
##  Session info
##  setting  value
##  version  R version 3.5.1 (2018-07-02)
##  os       Ubuntu 16.04.5 LTS
##  system   x86_64, linux-gnu
##  ui       X11
##  language en_GB.utf8:
##  collate  en_GB.UTF-8
##  ctype    en_GB.UTF-8
##  tz       Europe/Helsinki
##  date     2019-01-15
##
##   Packages
##  package              * version   date       lib source
##  abind                  1.4-5     2016-07-21 [1] CRAN (R 3.5.1)
##  acepack                1.4.1     2016-10-29 [1] CRAN (R 3.5.1)
##  ade4                   1.7-13    2018-08-31 [1] CRAN (R 3.5.1)
##  annotate               1.60.0    2018-10-30 [1] Bioconductor
##  AnnotationDbi          1.44.0    2018-10-30 [1] Bioconductor
##  ape                    5.2       2018-09-24 [1] CRAN (R 3.5.1)
##  assertthat             0.2.0     2017-04-11 [1] CRAN (R 3.5.1)
##  backports              1.1.2     2017-12-13 [1] CRAN (R 3.5.1)
##  base64enc              0.1-3     2015-07-28 [1] CRAN (R 3.5.1)
##  bindr                  0.1.1     2018-03-13 [1] CRAN (R 3.5.1)
##  bindrcpp               0.2.2     2018-03-29 [1] CRAN (R 3.5.1)
##  Biobase              * 2.42.0    2018-10-30 [1] Bioconductor
##  BiocGenerics         * 0.28.0    2018-10-30 [1] Bioconductor
##  BiocParallel         * 1.16.2    2018-11-28 [1] Bioconductor
##  biomformat             1.10.0    2018-10-30 [1] Bioconductor
##  Biostrings             2.50.1    2018-11-06 [1] Bioconductor
##  bit                    1.1-14    2018-05-29 [1] CRAN (R 3.5.1)
##  bit64                  0.9-7     2017-05-08 [1] CRAN (R 3.5.1)
##  bitops                 1.0-6     2013-08-17 [1] CRAN (R 3.5.1)
##  blob                   1.1.1     2018-03-25 [1] CRAN (R 3.5.1)
##  callr                  3.0.0     2018-08-24 [1] CRAN (R 3.5.1)
##  checkmate              1.8.5     2017-10-24 [1] CRAN (R 3.5.1)
##  class                  7.3-14    2015-08-30 [4] CRAN (R 3.5.0)
##  cli                    1.0.1     2018-09-25 [1] CRAN (R 3.5.1)
##  cluster                2.0.7-1   2018-04-09 [4] CRAN (R 3.5.0)
##  codetools              0.2-15    2016-10-05 [1] CRAN (R 3.5.1)
##  colorspace             1.3-2     2016-12-14 [1] CRAN (R 3.5.1)
##  crayon                 1.3.4     2017-09-16 [1] CRAN (R 3.5.1)
##  data.table             1.11.8    2018-09-30 [1] CRAN (R 3.5.1)
##  DBI                    1.0.0     2018-05-02 [1] CRAN (R 3.5.1)
##  DelayedArray         * 0.8.0     2018-10-30 [1] Bioconductor
##  deldir                 0.1-15    2018-04-01 [1] CRAN (R 3.5.1)
##  desc                   1.2.0     2018-05-01 [1] CRAN (R 3.5.1)
##  DESeq2               * 1.22.1    2018-11-05 [1] Bioconductor
##  devtools             * 2.0.1     2018-10-26 [1] CRAN (R 3.5.1)
##  digest                 0.6.18    2018-10-10 [1] CRAN (R 3.5.1)
##  dplyr                  0.7.8     2018-11-10 [1] CRAN (R 3.5.1)
##  e1071                * 1.7-0     2018-07-28 [1] CRAN (R 3.5.1)
##  eulerr               * 5.0.0     2018-11-05 [1] CRAN (R 3.5.1)
##  evaluate               0.12      2018-10-09 [1] CRAN (R 3.5.1)
##  exactRankTests       * 0.8-29    2017-03-01 [1] CRAN (R 3.5.1)
##  factoextra           * 1.0.5     2017-08-22 [1] CRAN (R 3.5.1)
##  foreach                1.4.4     2017-12-12 [1] CRAN (R 3.5.1)
##  foreign                0.8-71    2018-07-20 [4] CRAN (R 3.5.1)
##  Formula                1.2-3     2018-05-03 [1] CRAN (R 3.5.1)
##  fs                     1.2.6     2018-08-23 [1] CRAN (R 3.5.1)
##  genefilter             1.64.0    2018-10-30 [1] Bioconductor
##  geneplotter            1.60.0    2018-10-30 [1] Bioconductor
##  GenomeInfoDb         * 1.18.1    2018-11-12 [1] Bioconductor
##  GenomeInfoDbData       1.2.0     2018-12-04 [1] Bioconductor
##  GenomicRanges        * 1.34.0    2018-10-30 [1] Bioconductor
##  ggfortify            * 0.4.5     2018-05-26 [1] CRAN (R 3.5.1)
##  ggplot2              * 3.1.0     2018-10-25 [1] CRAN (R 3.5.1)
##  ggpubr                 0.2       2018-11-15 [1] CRAN (R 3.5.1)
##  ggrepel                0.8.0     2018-05-09 [1] CRAN (R 3.5.1)
##  glue                   1.3.0     2018-07-17 [1] CRAN (R 3.5.1)
##  goftest                1.1-1     2017-04-03 [1] CRAN (R 3.5.1)
##  gridExtra            * 2.3       2017-09-09 [1] CRAN (R 3.5.1)
##  gtable                 0.2.0     2016-02-26 [1] CRAN (R 3.5.1)
##  highr                  0.7       2018-06-09 [1] CRAN (R 3.5.1)
##  Hmisc                  4.1-1     2018-01-03 [1] CRAN (R 3.5.1)
##  hms                    0.4.2     2018-03-10 [1] CRAN (R 3.5.1)
##  htmlTable              1.12      2018-05-26 [1] CRAN (R 3.5.1)
##  htmltools              0.3.6     2017-04-28 [1] CRAN (R 3.5.1)
##  htmlwidgets            1.3       2018-09-30 [1] CRAN (R 3.5.1)
##  httr                   1.3.1     2017-08-20 [1] CRAN (R 3.5.1)
##  igraph                 1.2.2     2018-07-27 [1] CRAN (R 3.5.1)
##  IRanges              * 2.16.0    2018-10-30 [1] Bioconductor
##  iterators              1.0.10    2018-07-13 [1] CRAN (R 3.5.1)
##  jsonlite               1.5       2017-06-01 [1] CRAN (R 3.5.1)
##  kableExtra           * 0.9.0     2018-05-21 [1] CRAN (R 3.5.1)
##  knitr                * 1.20      2018-02-20 [1] CRAN (R 3.5.1)
##  labeling               0.3       2014-08-23 [1] CRAN (R 3.5.1)
##  lattice              * 0.20-38   2018-11-04 [1] CRAN (R 3.5.1)
##  latticeExtra           0.6-28    2016-02-09 [1] CRAN (R 3.5.1)
##  lazyeval               0.2.1     2017-10-29 [1] CRAN (R 3.5.1)
##  locfit                 1.5-9.1   2013-04-20 [1] CRAN (R 3.5.1)
##  magrittr               1.5       2014-11-22 [1] CRAN (R 3.5.1)
##  mapdata                2.3.0     2018-03-30 [1] CRAN (R 3.5.1)
##  maps                   3.3.0     2018-04-03 [1] CRAN (R 3.5.1)
##  MASS                   7.3-51.1  2018-11-01 [4] CRAN (R 3.5.1)
##  Matrix                 1.2-15    2018-11-01 [4] CRAN (R 3.5.1)
##  matrixStats          * 0.54.0    2018-07-23 [1] CRAN (R 3.5.1)
##  measurements         * 1.3.0     2018-12-09 [1] CRAN (R 3.5.1)
##  memoise                1.1.0     2017-04-21 [1] CRAN (R 3.5.1)
##  mgcv                   1.8-26    2018-11-21 [4] CRAN (R 3.5.1)
##  multtest               2.38.0    2018-10-30 [1] Bioconductor
##  munsell                0.5.0     2018-06-12 [1] CRAN (R 3.5.1)
##  nlme                 * 3.1-137   2018-04-07 [4] CRAN (R 3.5.0)
##  nnet                   7.3-12    2016-02-02 [4] CRAN (R 3.5.0)
##  permute              * 0.9-4     2016-09-09 [1] CRAN (R 3.5.1)
##  phyloseq             * 1.26.0    2018-10-30 [1] Bioconductor
##  pillar                 1.3.0     2018-07-14 [1] CRAN (R 3.5.1)
##  pkgbuild               1.0.2     2018-10-16 [1] CRAN (R 3.5.1)
##  pkgconfig              2.0.2     2018-08-16 [1] CRAN (R 3.5.1)
##  pkgload                1.0.2     2018-10-29 [1] CRAN (R 3.5.1)
##  plyr                 * 1.8.4     2016-06-08 [1] CRAN (R 3.5.1)
##  polyclip               1.9-1     2018-07-27 [1] CRAN (R 3.5.1)
##  polylabelr             0.1.0     2018-11-02 [1] CRAN (R 3.5.1)
##  prettyunits            1.0.2     2015-07-13 [1] CRAN (R 3.5.1)
##  processx               3.2.0     2018-08-16 [1] CRAN (R 3.5.1)
##  ps                     1.2.1     2018-11-06 [1] CRAN (R 3.5.1)
##  purrr                  0.2.5     2018-05-29 [1] CRAN (R 3.5.1)
##  R6                     2.3.0     2018-10-04 [1] CRAN (R 3.5.1)
##  randomForest         * 4.6-14    2018-03-25 [1] CRAN (R 3.5.1)
##  RColorBrewer         * 1.1-2     2014-12-07 [1] CRAN (R 3.5.1)
##  Rcpp                   1.0.0     2018-11-07 [1] CRAN (R 3.5.1)
##  RCurl                  1.95-4.11 2018-07-15 [1] CRAN (R 3.5.1)
##  readr                  1.2.1     2018-11-22 [1] CRAN (R 3.5.1)
##  remotes                2.0.2     2018-10-30 [1] CRAN (R 3.5.1)
##  reshape2             * 1.4.3     2017-12-11 [1] CRAN (R 3.5.1)
##  rfPermute            * 2.1.6     2018-07-07 [1] CRAN (R 3.5.1)
##  rfUtilities          * 2.1-3     2018-02-21 [1] CRAN (R 3.5.1)
##  rhdf5                  2.26.0    2018-10-30 [1] Bioconductor
##  Rhdf5lib                1.4.1     2018-11-22 [1] Bioconductor
##  rlang                   0.3.0.1   2018-10-25 [1] CRAN (R 3.5.1)
##  rmarkdown               1.10      2018-06-11 [1] CRAN (R 3.5.1)
##  rpart                   4.1-13    2018-02-23 [1] CRAN (R 3.5.1)
##  rprojroot               1.3-2     2018-01-03 [1] CRAN (R 3.5.1)
##  RSQLite                 2.1.1     2018-05-06 [1] CRAN (R 3.5.1)
##  rstudioapi              0.8       2018-10-02 [1] CRAN (R 3.5.1)
##  rvest                   0.3.2     2016-06-17 [1] CRAN (R 3.5.1)
##  S4Vectors             * 0.20.1    2018-11-09 [1] Bioconductor
##  scales                  1.0.0     2018-08-09 [1] CRAN (R 3.5.1)
##  sessioninfo             1.1.1     2018-11-05 [1] CRAN (R 3.5.1)
##  spatstat                1.57-1    2018-11-04 [1] CRAN (R 3.5.1)
##  spatstat.data           1.4-0     2018-10-04 [1] CRAN (R 3.5.1)
##  spatstat.utils          1.13-0    2018-10-31 [1] CRAN (R 3.5.1)
##  stringi                 1.2.4     2018-07-20 [1] CRAN (R 3.5.1)
##  stringr                 1.3.1     2018-05-10 [1] CRAN (R 3.5.1)
##  SummarizedExperiment * 1.12.0    2018-10-30 [1] Bioconductor
##  survival                2.43-3    2018-11-26 [4] CRAN (R 3.5.1)
##  swfscMisc               1.2       2016-08-23 [1] CRAN (R 3.5.1)
##  tensor                  1.5       2012-05-05 [1] CRAN (R 3.5.1)
##  tibble                  1.4.2     2018-01-22 [1] CRAN (R 3.5.1)
##  tidyr                 * 0.8.2     2018-10-28 [1] CRAN (R 3.5.1)
##  tidyselect              0.2.5     2018-10-11 [1] CRAN (R 3.5.1)
##  usethis               * 1.4.0     2018-08-14 [1] CRAN (R 3.5.1)
##  vegan                 * 2.5-3     2018-10-25 [1] CRAN (R 3.5.1)
##  viridisLite             0.3.0     2018-02-01 [1] CRAN (R 3.5.1)
##  withr                   2.1.2     2018-03-15 [1] CRAN (R 3.5.1)
##  XML                     3.98-1.16 2018-08-19 [1] CRAN (R 3.5.1)
##  xml2                    1.2.0     2018-01-24 [1] CRAN (R 3.5.1)
##  xtable                  1.8-3     2018-08-29 [1] CRAN (R 3.5.1)
##  XVector                 0.22.0    2018-10-30 [1] Bioconductor
##  yaml                    2.2.0     2018-07-25 [1] CRAN (R 3.5.1)
##  zlibbioc                1.28.0    2018-10-30 [1] Bioconductor
##
## [1] /home/local/vaho/R/x86_64-pc-linux-gnu-library/3.5
## [2] /usr/local/lib/R/site-library
## [3] /usr/lib/R/site-library
## [4] /usr/lib/R/library
```