Imports

```
md"# Imports"
```

```
import Pkg, Revise; Pkg.activate(Base.current_project())
```

```
  Activating project at `/DATA/cossio/SAM/2024/SamApp2024.jl`
```

```
import Makie, CairoMakie
```

```
import CSV, HDF5
```

```
import FASTX, Infernal
```

```
import SamApp2024
```

```
import RestrictedBoltzmannMachines as RBMs
```

```
import Rfam
```

```
import StatsBase, KernelDensity
```

```
using BioSequences: LongRNA
```

```
using DataFrames: DataFrame
```

```
using Distributions: Gamma, logpdf, pdf, Poisson
```

```
using LinearAlgebra: Diagonal, eigen
```

```
using Makie: @L_str
```

```
using NaNStatistics: nansum
```

```
using Random: bitrand
```

```
using Statistics: cor, mean
```

```
using StatsBase: countmap
```

Plot

```
md"# Plot"
```

```
Rfam_cm = Infernal.cmfetch(Rfam.cm(), "RF00162");
```

```
RF00162_seed_stk =
▸ (out = "/tmp/jl_KJi92rmN2l", stdout = "/tmp/jl_g3y9wLe96B", stderr = "/tmp/jl_4vCtudi1iN")
```

```
RF00162_seed_stk = Infernal.esl_afetch(Rfam.seed(), "RF00162")
```

```
RF00162_seed_match_cols = findall(≠('.'), SamApp2024.stockholm_ss(RF00162_seed_stk.out));
```

```
RF00162_seed_afa =
▸ (out = "/tmp/jl_cVIrYBgnex", stdout = "/tmp/jl_6NchYpGVWB", stderr = "/tmp/jl_GhKROx2XWB")
```

```
RF00162_seed_afa = Infernal.esl_reformat("AFA", RF00162_seed_stk.out;
informat="STOCKHOLM") # WARNING: this has inserts marked as '-'
```

```
RF00162_seed_records =
▸ [FASTX.FASTA.Record:                                    , FASTX.FASTA.Record:
   description: "AF027868.1/5245-5154"                      description: "AF269983.1/571-67
```

```
RF00162_seed_records = collect(FASTX.FASTA.Reader(open(RF00162_seed_afa.out)))
```

```
RF00162_seed_seqs_noinserts = LongRNA{4}.([FASTX.sequence(record)
[RF00162_seed_match_cols] for record in RF00162_seed_records]);
```

```
# trimmed (no inserts) aligned fasta
```

```
RF00162_hits_afa = Infernal.cmalign(Rfam_cm.out, Rfam.fasta_file("RF00162");
matchonly=true, outformat="AFA");
# these are already aligned and without inserts
```

```
RF00162_hits_sequences = LongRNA{4}.(FASTX.sequence.
(FASTX.FASTA.Reader(open(RF00162_hits_afa.out))));
```

```
# emit sequences from Rfam CM model
Rfam_cm_emitted_sequences_afa = Infernal.cmemit(Rfam_cm.out; N=5000, aligned=true,
outformat="AFA");
```

```
▸ [108nt RNA Sequence:                                                    , 108nt RNA S
  CUCCUAUCCAGAGGGACGGAGGGAGUGGCCCUGUAUAGU GCCAUACCCUCUAG-----AUG--GCAAACGGAUAGGAG UUGCCAUCUGA
```

```
begin
    Rfam_cm_emitted_sequences = FASTX.sequence.
    (FASTX.FASTA.Reader(open(Rfam_cm_emitted_sequences_afa.out)));
    Rfam_cm_emitted_sequences = [filter(!=('.'), filter(!islowercase, seq)) for seq in
    Rfam_cm_emitted_sequences];
    Rfam_cm_emitted_sequences = LongRNA{4}.(Rfam_cm_emitted_sequences);
end
```

▸ [108nt RNA Sequence: ACCCUAUCAAGAGUGGUUGAAGAGCUGGUCUUAUGAAAC…GCCAAGUCCGACAGGAUGGGAGUUCCGGAAAGAUGGGAG , 108nt RNA S UACUUAUUCAG

```
begin
    # aligned hits, used to train a new noiseless CM model (in Stockholm format, without inserts!)
    RF00162_hits_stk = Infernal.cmalign(Rfam_cm.out, Rfam.fasta_file("RF00162"); matchonly=true);
    # fit new CM model using full alignment (without inserts), and without entropic noise
    Refined_cm = Infernal.cmbuild(RF00162_hits_stk.out; enone=true);

    # emit sequences from Refined CM model
    Refined_cm_emitted_sequences_afa = Infernal.cmemit(Refined_cm.cmout; N=5000, aligned=true, outformat="AFA");
    Refined_cm_emitted_sequences = FASTX.sequence.(FASTX.FASTA.Reader(open(Refined_cm_emitted_sequences_afa.out)));
    # remove inserts
    Refined_cm_emitted_sequences = [filter(!=('.'), filter(!islowercase, seq)) for seq in Refined_cm_emitted_sequences];
    @assert only(unique(length.(Refined_cm_emitted_sequences))) == 108
    Refined_cm_emitted_sequences = LongRNA{4}.(Refined_cm_emitted_sequences);
end
```

```
# use saved RBM samples
sampled_v = SamApp2024.rbm2022samples(); # faster
```

▸ [119.98, 122.13, 117.93, 125.82, 126.6, 118.9, 118.78, 119.73, 114.7, 118.16, 121.37, 120.02, 12

```
begin
    # Infernal scores of hits, using Rfam CM model
    RF00162_hits_Rfam_cm_scores = Infernal.cmalign_parse_sfile(
        Infernal.cmalign(
            Rfam_cm.out,
            Infernal.esl_reformat("FASTA", RF00162_hits_afa.out; informat="AFA").out;
            glob=true, informat="FASTA"
        ).sfile
    ).bit_sc;

    # Infernal scores of hits, using Refined CM model
    RF00162_hits_Refined_cm_scores = Infernal.cmalign_parse_sfile(
        Infernal.cmalign(
            Refined_cm.cmout,
            Infernal.esl_reformat("FASTA", RF00162_hits_afa.out; informat="AFA").out;
            glob=true, informat="FASTA"
        ).sfile
    ).bit_sc;
end
```

▸ [105.91, 69.36, 103.03, 101.23, 107.89, 85.26, 60.42, 117.34, 102.88, 107.64, 98.22, 100.8, 110.

```
begin
    # Infernal scores of Refined CM samples
    _tmpfasta = tempname()
    FASTX.FASTA.Writer(open(_tmpfasta, "w")) do writer
        for (n, seq) in enumerate(Refined_cm_emitted_sequences)
            ismissing(seq) && continue
            write(writer, FASTX.FASTA.Record(string(n), filter(!=('-'), string(seq))))
        end
    end
    # Infernal scores
    Refined_cm_emitted_sequences_infernal_scores = Infernal.cmalign_parse_sfile(
        Infernal.cmalign(Refined_cm.cmout, _tmpfasta; glob=true, informat="FASTA").sfile
    ).bit_sc;


    # Infernal scores of Rfam CM samples
    _tmpfasta = tempname()
    FASTX.FASTA.Writer(open(_tmpfasta, "w")) do writer
        for (n, seq) in enumerate(Rfam_cm_emitted_sequences)
            ismissing(seq) && continue
            write(writer, FASTX.FASTA.Record(string(n), filter(!=('-'), string(seq))))
        end
    end
    # Infernal scores
    Rfam_cm_emitted_sequences_infernal_scores = Infernal.cmalign_parse_sfile(
        Infernal.cmalign(Rfam_cm.out, _tmpfasta; glob=true, informat="FASTA").sfile
    ).bit_sc;

    # Infernal scores of RBM samples
    _tmpfasta = tempname()
    FASTX.FASTA.Writer(open(_tmpfasta, "w")) do writer
        for (n, seq) in enumerate(SamApp2024.rnaseq(sampled_v))
            @assert !ismissing(seq)
            write(writer, FASTX.FASTA.Record(string(n), filter(!=('-'), string(seq))))
        end
    end

    # Rfam CM
    RBM_samples_Rfam_CM_infernal_scores = Infernal.cmalign_parse_sfile(
        Infernal.cmalign(Rfam_cm.out, _tmpfasta; glob=true, informat="FASTA").sfile
    ).bit_sc;

    # Refined CM
    RBM_samples_Refined_CM_infernal_scores = Infernal.cmalign_parse_sfile(
        Infernal.cmalign(Refined_cm.cmout, _tmpfasta; glob=true, informat="FASTA").sfile
    ).bit_sc;

end
```

```
# sites that have some non-zero fluctuations
# We need to separate frozen sites below because otherwise cor and eigen give NaN, infinities, and fail
_variable_sites_flag = vec(all(0 .< mean(SamApp2024.onehot(RF00162_hits_sequences); dims=3) .< 1; dims=1));
```

```
_variable_sites = findall(_variable_sites_flag);
```

```
RF00162_hits_var_sites_only = SamApp2024.onehot(RF00162_hits_sequences)[:,
_variable_sites, :];
```

```
RF00162_hits_cor = cor(reshape(RF00162_hits_var_sites_only, :,
size(RF00162_hits_var_sites_only, 3)); dims=2);
```

```
RF00162_hits_eig = eigen(RF00162_hits_cor);


```

```
# remap the variable sites eigenvectors back to the original consensus sequence numbering
begin
    RF00162_hits_eigvec = zeros(5, 108, size(RF00162_hits_eig.vectors, 1))
    for n in 1:size(RF00162_hits_eig.vectors, 1)
        vec(view(RF00162_hits_eigvec, :, _variable_sites, n)) .=
    RF00162_hits_eig.vectors[:, n]
    end
end
```

```
__proj_hits = reshape(SamApp2024.onehot(RF00162_hits_sequences), 5*108, :)' *
reshape(RF00162_hits_eigvec, 5*108, :);
```

```
__proj_rbm = reshape(sampled_v, 5*108, :)' * reshape(RF00162_hits_eigvec, 5*108, :);
```

```
__proj_refined_cm = reshape(SamApp2024.onehot(Refined_cm_emitted_sequences), 5*108, :)'
* reshape(RF00162_hits_eigvec, 5*108, :);
```

```
__proj_rfam_cm = reshape(SamApp2024.onehot(Rfam_cm_emitted_sequences), 5*108, :)' *
reshape(RF00162_hits_eigvec, 5*108, :);
```

```
# load SHAPE data
shape_data_045 = SamApp2024.load_shapemapper_data_pierre_demux_20230920(; demux=true);
```

```
shape_data_rep0 = SamApp2024.select_conditions_20231002(shape_data_045,
filter(endswith("_rep0"), shape_data_045.conditions));
```

```
shape_data_rep45 = SamApp2024.select_conditions_20231002(shape_data_045,
filter(endswith("_rep45"), shape_data_045.conditions));
```

```
_idx_not_missing_seqs = findall(!ismissing, shape_data_rep0.aligned_sequences);
```

```
shape_sequences_onehot = SamApp2024.onehot(LongRNA{4}.
(shape_data_rep0.aligned_sequences[_idx_not_missing_seqs]));
```

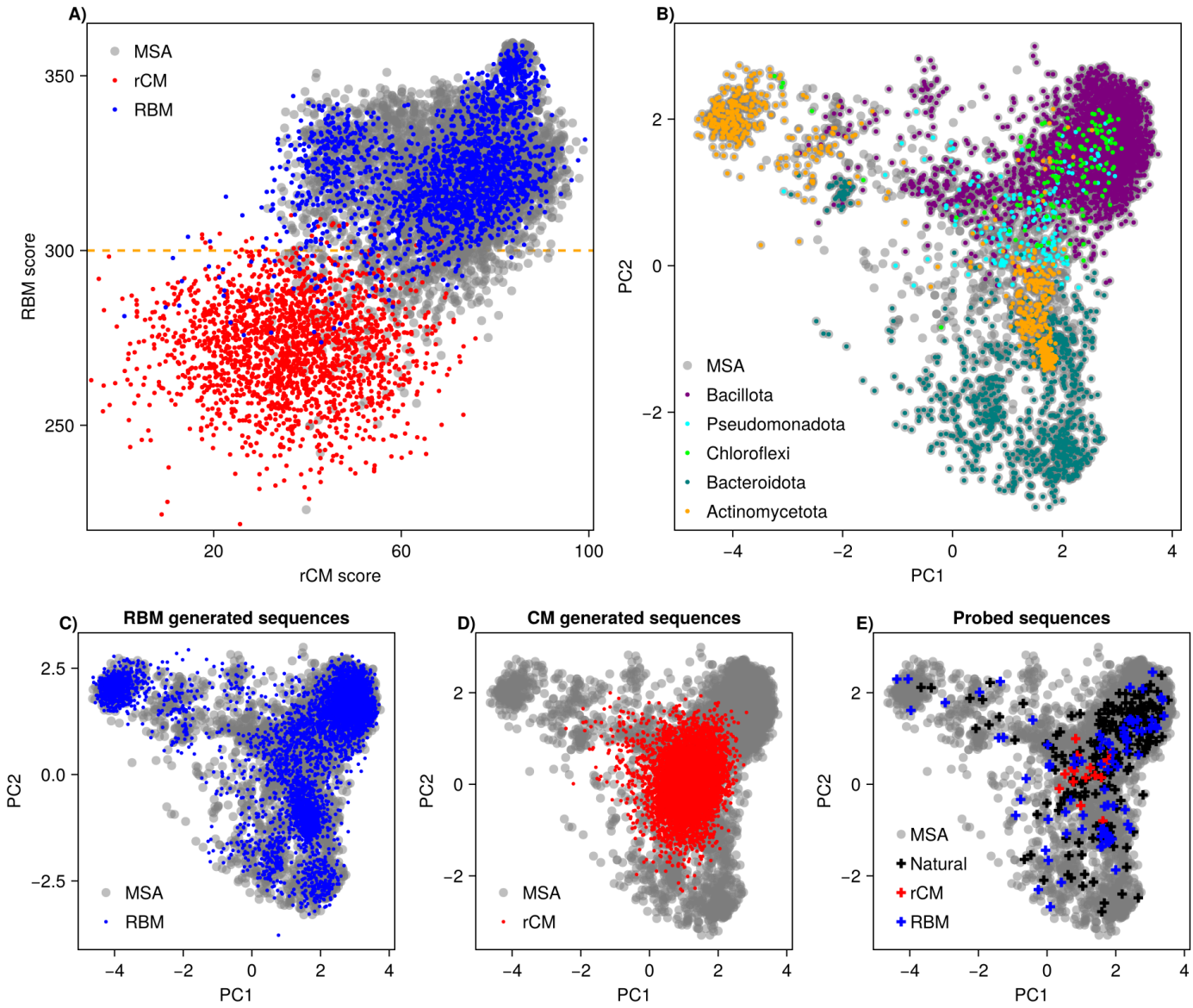
```
__proj_probed = reshape(shape_sequences_onehot, 5*108, :)' *
reshape(RF00162_hits_eigvec, 5*108, :);
```

```
_probed_origin = shape_data_rep0.aptamer_origin[_idx_not_missing_seqs];
```

```
["Bacilli", "Bacilli", "Tissierellia", "Clostridia", "Bacilli", "Bacilli", "Bacilli", "Bacilli"
```

```
begin
    hits_tax_df = SamApp2024.rf00162_hits_taxonomy();
    hits_tax_df.taxonomy_split = [ismissing(tax) ? missing : split(tax, "; ") for tax in
    hits_tax_df.taxonomy]
    hits_tax_df.taxa_1 = [ismissing(tax) ? missing : length(tax) ≥ 1 ? tax[1] : missing
    for tax in hits_tax_df.taxonomy_split];
    hits_tax_df.taxa_2 = [ismissing(tax) ? missing : length(tax) ≥ 2 ? tax[2] : missing
    for tax in hits_tax_df.taxonomy_split];
    hits_tax_df.taxa_3 = [ismissing(tax) ? missing : length(tax) ≥ 3 ? tax[3] : missing
    for tax in hits_tax_df.taxonomy_split];
end
```

```
hits_tax_cnt = countmap(split(join(filter(!ismissing, filter(!ismissing,
hits_tax_df.taxonomy)), "; "), "; "));
```




```
let fig = Makie.Figure()

    ax = Makie.Axis(fig[1,1][1,1], xlabel="rCM score", ylabel="RBM score", width=400,
    height=400, xticks=20:40:130, xgridvisible=false, ygridvisible=false)
    Makie.hlines!(ax, 300, color=:orange, linestyle=:dash, linewidth=2)
    #Makie.scatter!(ax, RF00162_hits_Refined_cm_scores, -
    RBMs.free_energy(SamApp.rbm2022(), SamApp.onehot(RF00162_hits_sequences)),
    label="Natural", color=(:gray, 0.5), markersize=10)
    Makie.scatter!(
        ax, RF00162_hits_Rfam_cm_scores, -RBMs.free_energy(SamApp2024.rbm2022(),
        SamApp2024.onehot(RF00162_hits_sequences)), label="MSA", color=(:gray, 0.5),
        markersize=10
    )
    Makie.scatter!(ax,
        #Refined_cm_emitted_sequences_infernal_scores[1:2000],
        Rfam_cm_emitted_sequences_infernal_scores[1:2000],
        -RBMs.free_energy(SamApp2024.rbm2022(),
    SamApp2024.onehot(Refined_cm_emitted_sequences))[1:2000],
        label="rCM", color=:red, markersize=5
    )
    Makie.scatter!(ax,
        #RBM_samples_Refined_CM_infernal_scores[1:2000],
        RBM_samples_Rfam_CM_infernal_scores[1:2000],
        -RBMs.free_energy(SamApp2024.rbm2022(), sampled_v)[1:2000],
        label="RBM", color=:blue, markersize=5
    )
```

```
    Makie.xlims!(ax, -7, 101)
    Makie.ylims!(ax, 220, 365)
    Makie.axislegend(ax, position=:lt, framevisible=false)


    _colors = [:purple, :cyan, :lime, :teal, :orange]
    _c = 0

    # Natural
    ax = Makie.Axis(fig[1,1][1,2], xlabel="PC1", ylabel="PC2", width=400, height=400,
    xgridvisible=false, ygridvisible=false) #title="Natural sequences")
    Makie.scatter!(ax, __proj_hits[:, end], __proj_hits[:, end - 1], markersize=10,
    label="MSA", color=(:gray, 0.5))
    for t = unique(hits_tax_df.taxa_2)
        ismissing(t) && continue
        hits_tax_cnt[t] > 100 || continue
        _c += 1
        Makie.scatter!(ax,
            __proj_hits[replace(hits_tax_df.taxa_2 .== t, missing => false), end],
            __proj_hits[replace(hits_tax_df.taxa_2 .== t, missing => false), end - 1],
            markersize=5, label=t, color=_colors[_c])
    end
    Makie.axislegend(ax, position=(-0.05, -0.03), framevisible=false)

    ax = Makie.Axis(fig[2,1][1,1], xlabel="PC1", ylabel="PC2", width=250, height=250,
    xgridvisible=false, ygridvisible=false, title="RBM generated sequences")
    Makie.scatter!(ax, __proj_hits[:, end], __proj_hits[:, end - 1], markersize=10,
    label="MSA", color=color=(:gray, 0.5))
    Makie.scatter!(ax, __proj_rbm[:, end], __proj_rbm[:, end - 1], markersize=4,
    label="RBM", color=:blue)
    Makie.axislegend(ax, position=:lb, framevisible=false)

    ax = Makie.Axis(fig[2,1][1,2], xlabel="PC1", ylabel="PC2", width=250, height=250,
    xgridvisible=false, ygridvisible=false, title="CM generated sequences")
    Makie.scatter!(ax, __proj_hits[:, end], __proj_hits[:, end - 1], markersize=10,
    label="MSA", color=color=(:gray, 0.5))
    Makie.scatter!(ax, __proj_rfam_cm[:, end], __proj_rfam_cm[:, end - 1], markersize=4,
    label="rCM", color=:red)
    Makie.axislegend(ax, position=:lb, framevisible=false)

    ax = Makie.Axis(fig[2,1][1,3], xlabel="PC1", ylabel="PC2", width=250, height=250,
    xgridvisible=false, ygridvisible=false, title="Probed sequences")
    Makie.scatter!(ax, __proj_hits[:, end], __proj_hits[:, end - 1], markersize=10,
    color=(:gray, 0.5), label="MSA")
    Makie.scatter!(ax,
        __proj_probed[(_probed_origin .== "RF00162_seed70") .| (_probed_origin .==
    "RF00162_full30"), end],
        __proj_probed[(_probed_origin .== "RF00162_seed70") .| (_probed_origin .==
    "RF00162_full30"), end - 1],
        markersize=10, color=:black, label="Natural", marker=:cross
    )
    Makie.scatter!(ax,
        __proj_probed[_probed_origin .== "RF00162_syn_inf", end],
        __proj_probed[_probed_origin .== "RF00162_syn_inf", end - 1],
        markersize=10, color=:red, label="rCM", marker=:cross
    )
    Makie.scatter!(ax,
        __proj_probed[_probed_origin .== "RF00162_syn_rbm", end],
        __proj_probed[_probed_origin .== "RF00162_syn_rbm", end - 1],
        markersize=10, color=:blue, label="RBM", marker=:cross
```

```
    )
    Makie.axislegend(ax, position=:lb, framevisible=false, patchlabelgap=-3)

    Makie.Label(fig[1,1][1,1][1,1,Makie.TopLeft()], "A)", font=:bold)
    Makie.Label(fig[1,1][1,2][1,1,Makie.TopLeft()], "B)", font=:bold)
    Makie.Label(fig[2,1][1,1][1,1,Makie.TopLeft()], "C)", font=:bold)
    Makie.Label(fig[2,1][1,2][1,1,Makie.TopLeft()], "D)", font=:bold)
    Makie.Label(fig[2,1][1,3][1,1,Makie.TopLeft()], "E)", font=:bold)

    # fig[0,4] = Makie.Label(fig, "Natural sequences", font=:bold)
    # fig[0,5] = Makie.Label(fig, "Generated sequences", font=:bold)

    Makie.resize_to_layout!(fig)
    #Makie.save("/workspaces/SamApp.jl/notebooks/2024-03-14 New paper
    figures/Figures/PCA.pdf", fig)
    fig
```